

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”**

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено:

Завідувач кафедри

Сергій СТИРЕНКО

(підпис)

“ ” _____ 2022 р.

Дипломний проект

на здобуття ступеня бакалавра

за освітньо-професійною програмою “Комп’ютерні системи та мережі”

спеціальності 123 “Комп’ютерна інженерія”

на тему: Програмний прикладний інтерфейс для доступу до інформаційних
ресурсів

Виконав : студент 4 курсу, групи ІО-83
(шифр групи)

Дяченко Владислав Геннадійович

(прізвище, ім’я, по батькові)

(підпис)

Керівник професор, к.т.н. Клименко І. А.

(посада, науковий ступінь, вчене звання, прізвище та ініціали)

(підпис)

Консультант (нормоконтроль) професор, д.т.н., Сімоненко В. П.

(назва розділу)

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

(підпис)

Рецензент _____

(посада, науковий ступінь, вчене звання, прізвище та ініціали)

(підпис)

Засвідчую, що у цьому дипломному
проекті немає запозичень з праць інших
авторів без відповідних посилань.

Студент _____

(підпис)

Київ – 2022 р.

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”**

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалавр)

Освітньо-професійна програма

“Комп’ютерні системи та мережі”

спеціальності 123 “Комп’ютерна інженерія”

ЗАТВЕРДЖУЮ
Завідувач кафедри
Сергій СТИРЕНКО

(підпис)

“ ” _____ 2022 р.

ЗАВДАННЯ

на бакалаврський дипломний проект студента

Дяченка Владислава Геннадійовича

1. Тема проекту Програмний прикладний інтерфейс для доступу до інформаційних ресурсів

керівник проекту Клименко Ірина Анатоліївна, доцент, к.т.н.,

(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від 10 червня 2022 року №1033-с

2. Термін здачі студентом закінченого проекту 30 травня 2022 р.

3. Вихідні дані до проекту технічна документація, теоретичні дані.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які розробляються)

Розділ 1. Найменування та область застосування.

Розділ 2. Огляд технологій для розробки системи.

Розділ 3. Деталі розробки системи.

Розділ 4. Дослідження та аналіз розробленої системи.

5. Перелік графічного матеріалу (з точним позначенням обов'язкових креслень) структурна схема системи, функціональна схема, алгоритм дій програмного забезпечення.

6. Консультанта проєкту, з вказівкою розділів проєкту, які до них вносяться

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Нормоконтроль	Сімоненко В. П.		

7. Дата видачі завдання «30» серпня 2021 р.

Календарний план

№ п/п	Найменування етапів дипломного проєкту	Терміни виконання етапів проєкту	Примітки
1.	<i>Затвердження теми проєкту</i>	<i>10.12.2021-15.12.2021</i>	
2.	<i>Вивчення та аналіз завдання</i>	<i>15.12.2021-15.03.2022</i>	
3.	<i>Розробка архітектури та загальної структури системи</i>	<i>15.03.2022-25.03.2022</i>	
4.	<i>Розробка структур окремих підсистем</i>	<i>25.03.2022-5.04.2022</i>	
5.	<i>Програмна реалізація системи</i>	<i>5.04.2022-15.04.2022</i>	
6.	<i>Оформлення пояснювальної записки</i>	<i>15.04.2022-20.05.2022</i>	
7.	<i>Захист програмного продукту</i>	<i>25.04.2022</i>	
8.	<i>Передзахист</i>	<i>23.05.2022</i>	
9.	<i>Захист</i>	<i>14.06.2022</i>	

Студент-дипломник _____ Владислав ДЯЧЕНКО
(підпис)

Керівник проєкту _____ Ірина КЛИМЕНКО
(підпис)

АНОТАЦІЯ

Дипломна робота складається з 4 розділів, 3 додатків, 1 таблиці та 49 рисунків.

В дипломній роботі спеціаліста запропонований та реалізований програмний прикладний інтерфейс для доступу до інформаційних ресурсів. Розроблена програма дозволяє полегшити процес знаходження, вивчення інформації. Ціллю програми є полегшення умов адаптації нових співробітників на великих проектах за рахунок легкого доступу до інформації, зрозумілого інтерфейсу.

У роботі проведено аналіз роботи розробленої програми. Описано функціонал та архітектуру проекту для вирішення поставлених задач. Програма написана на мові програмування Python версії 3.9 з використанням фреймворку Django. За керування базами даних відповідає SQLite3.

ANNOTATION

Thesis consists of 4 sections, 3 appendices, 1 table and _ pictures.

In the diploma work of the specialist the software application interface for access to information resources is offered and realized. The developed program allows to facilitate the process of finding and studying information. The aim of the program is to facilitate the adaptation of new employees on large projects through easy access to information, clear interface.

The analysis of the developed program is carried out in the work. The functionality and architecture of the project for solving the tasks are described. The program is written in the Python programming language version 3.9 using the Django framework. SQLite3 is responsible for database management.

справки	Формат	Значення	Найменування	Кіл. листів	№ екземпля	Додаток
			Документація загальна			
			Знову розроблена			
	<i>A4</i>	<i>ІАЛЦ.467200.002 ТЗ</i>	Програмний прикладний	3		
			Інтерфейс для доступу			
			До інформаційних ресурсів			
			Технічне завдання			
	<i>A4</i>	<i>ІАЛЦ.467200.003 ПЗ</i>	Програмний прикладний	68		
			Інтерфейс для доступу			
			До інформаційних ресурсів			
			Пояснювальна записка			
	<i>A4</i>	<i>ІАЛЦ.467200.004 Д1</i>	Програмний прикладний	1		
			Інтерфейс для доступу			
			До інформаційних ресурсів			
			Структурна схема системи			
	<i>A4</i>	<i>ІАЛЦ.4672008.005 Д2</i>	Програмний прикладний	1		
			Інтерфейс для доступу			
			До інформаційних ресурсів			
			Функціональна схема			
	<i>A4</i>	<i>ІАЛЦ.467200.006 Д3</i>	Програмний прикладний	1		
			Інтерфейс для доступу			
			До інформаційних ресурсів			
			Алгоритм дій програмного забезпечення			
	<i>A4</i>	<i>ІАЛЦ.467200.007 Д4</i>	Програмний прикладний	35		
			Інтерфейс для доступу			
			До інформаційних ресурсів			
			Текст програмного коду			

					<i>ІАЛЦ.467200.001 ОА</i>						
Зм	Лист	№ докум.	Підп	Дата							
Розроб		Дяченко В.Г			<i>Програмний прикладний інтерфейс для доступу до інформаційних ресурсів</i> Опис альбому			Літ.	Аркуш	Аркушів	
Перев		Клименко І.А.							1	1	
					<i>НТУУ “КПІ” ФІОТ ІО-83</i>						

ТЕХНІЧНЕ ЗАВДАННЯ
ДО ДИПЛОМНОГО ПРОЄКТУ

на тему: «Програмний прикладний інтерфейс для доступу до інформаційних
ресурсів»

Київ – 2022

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ.....	2
2. ПІДСТАВИ ДЛЯ РОЗРОБКИ.....	2
3. МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ.....	2
4. ДЖЕРЕЛА РОЗРОБКИ.....	2
5. ТЕХНІЧНІ ВИМОГИ.....	3
5.1. Вимоги до розробленого продукту.....	3
5.2. Вимоги до програмного забезпечення.....	3
5.3. Вимоги до апаратної частини.....	3
6. ЕТАПИ РОЗРОБКИ.....	3

					ІАЛЦ.467200.002 ТЗ							
		№ докум.	Підпис	Дата								
Розробив		Дяченко В.Г.			Програмний прикладний інтерфейс для доступу до інформаційних ресурсів Технічне завдання			Літ.	Аркуш	Аркушів		
Перевірів		Клименко І.А.								1	3	
Н. Контр.		Сімоненко В. П.										
Затвердив												

1 НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Дане технічне завдання поширюється на розробку програмний прикладний інтерфейс для доступу до інформаційних ресурсів, а також на подальшу підтримку та вдосконалення розробленої програми.

Областю застосування цієї програми є полегшення пошуку потрібної інформації, її вивченні, полегшені адаптації нових працівників масштабних проектів.

2 ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки даної системи є завдання для виконання роботи кваліфікаційно-освітнього рівня «бакалавр інженерії програмного забезпечення», який був затверджений факультетом “Інформатики та обчислювальної техніки” кафедрою обчислювальної техніки Національного технічного Університету України «Київський Політехнічний інститут ім. Ігоря Сікорського».

3 МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою та призначенням даної роботи є розробка програмного прикладного інтерфейсу для доступу до інформаційних ресурсів, що дозволить ефективно вирішувати дану задачу.

4 ДЖЕРЕЛА РОЗРОБКИ

Джерелом розробки даного дипломного проекту є офіційні документації, публікації та статті в мережі Інтернет на дану тему, науково-технічна література.

					ІАЛЦ.467200.002 ТЗ	Арк.
						2
Зм.	Арк.	№ докум.	Підпис	Дата		

5 ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до розробленого продукту

Розроблена система має виконувати такі вимоги:

- Простий і інтуїтивно-зрозумілий інтерфейс програми.
- Надати можливість користувачам власноруч публікувати інформацію.
- Надати можливість користувачам можливість пошуку інформації по темам.
- Надати можливість користувачам власноруч створювати теми і кімнати для вибраної теми.
- Надати вичерпну та зрозумілу документацію для розробленого продукту.

5.2. Вимоги до програмного забезпечення

- ОС Windows, Mac чи Linux.
- Visual Studio Code 2017 версії або вище.

5.3. Вимоги до апаратної частини

- ЦП не менше ніж Intel® Core (TM) i3-2100T.
- ROM не менше ніж 64 ГБ.
- RAM не менше ніж 4 ГБ.

6 ЕТАПИ РОЗРОБКИ

Назва етапів виконання	Термін виконання
Затвердження теми роботи	10.12.2021-15.12.2021
Вивчення та аналіз завдання	15.12.2021-15.03.2022
Розробка архітектури та загальної структури системи	15.03.2022-25.03.2022
Розробка структур окремих частин системи	25.03.2022-5.04.2022
Програмна реалізація системи	5.04.2022-15.04.2022
Виправлення помилок	15.04.2022-20.05.2022
Оформлення пояснювальної записки	25.04.2022

					ІАЛЦ.467200.002 ТЗ	Арк.
						3
Зм.	Арк.	№ докум.	Підпис	Дата		

**ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО ДИПЛОМНОГО ПРОЄКТУ**

на тему: «Програмний прикладний інтерфейс для доступу до інформаційних
ресурсів»

Київ – 2022

ЗМІСТ

ВСТУП.....	6
РОЗДІЛ 1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ	8
1.1 Огляд існуючих проблем	8
1.2 Огляд наявних програм для адаптації нових співробітників	9
1.3 Документація проекту	10
1.4 Месенджери	14
1.5 Stack Overflow	17
1.6 Опис розроблюваної програми	19
1.7 Вимоги до програмного продукту	20
1.8 Порівняння розроблюваної програми з іншими.....	20
ВИСНОВОК ДО РОЗДІЛУ 1.....	22
РОЗДІЛ 2. ОГЛЯД ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ ПРОГРАМИ	23
2.1 Технології для реалізації серверної частини	23
2.1.1 Python.....	23
2.1.2 Java	27
2.1.3 C#	29
2.1.4 JavaScript	30
2.1.5 Rust.....	32
2.1.5 Обрання мови програмування.....	33
2.2 Підсистема зберігання даних	36
ВИСНОВОК ДО РОЗДІЛУ 2.....	40
РОЗДІЛ 3. ДЕТАЛІ РОЗРОБКИ ПРОГРАМИ.....	41

3.1 Налаштування проекту.	41
3.2 Створення моделей в базі даних та їх логіки.....	44
3.3 Створення компонентів програми	47
3.3.1. Початкова сторінка.....	47
3.1.3 Кімната, вигляд, створення, редагування	48
3.3.3 Авторизація	52
3.4 Динамічна генерація HTML	53
ВИСНОВОК ДО РОЗДІЛУ 3.....	56
РОЗДІЛ 4. ДОСЛІДЖЕННЯ ТА АНАЛІЗ РОЗРОБЛЕНОЇ ПРОГРАМИ	57
4.1 Запуск сервера	57
4.2 Інтерфейс веб-додатку	57
4.2.1 Початкова сторінка.....	57
4.2.2 Створення кімнати.....	59
4.2.3 Профіль користувача.....	61
4.2.4 Кімната	62
4.2.5 Авторизація	65
4.3. Відповідь сервера	65
ВИСНОВОК ДО РОЗДІЛУ 4.....	67
ВИСНОВКИ	68
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	69
Додаток 1	3
Додаток 2.....	5
Додаток 3.....	7
Додаток 4.....	9

ВСТУП

Існує безліч складних проектів, входження в які, іноді, бувають досить складними. Не завжди всі технології програмування відомі для працівника, який щойно узявся за новий проект, або попав у нову команду. Для адаптації та вивчення необхідних матеріалів може піти досить багато часу. При ефективному і цілеспрямованому вивченні саме потрібних тем – цей процес можна значно пришвидшитись. Проте існує питання, де саме шукати матеріал, як знайти відповіді на питання, де їх шукати. В інтернеті на даний час досить багато різних інформаційних джерел і ресурсів. Деякі навіть платні. В якому саме джерелі буде краще вивчати той чи інший матеріал досить важко визначити. Було б набагато легше, якби люди, які через це все пройшли, самі розповіли і виділили основну інформацію і теми які слід в першу чергу вивчити для комфортного входження в проект.

Мета дипломного проекту – пришвидшення адаптації, полегшення у вивченні потрібних тем. Новий працівник зосередиться на вивченні саме потрібного матеріалу, у правильно сформульованому змісті, не забиваючи голову непотрібними технологіями.

Для досягнення мети пропонується зробити веб-додаток під назвою StudyBuddy. З-за допомогою нього можна чітко сформулювати теми, створити кімнати за цими темами, де співробітники можуть ділитися повідомленнями з правильним порядком інформації і рекомендації, щодо технологій розробки проекту. Не треба буде кожному новому співробітнику, який увійшов у команду розказувати одну і ту саму інформацію. Досить буде перейти йому у веб-програму, де все буде змістовно описано по технологіям. І якщо якась технологія викликає сумніви, можна перейти в веб-додаток і оновити свої знання.

					ІАЛЦ.467200.003 ПЗ	Арк.
						6
Зм.	Арк.	№ докум.	Підпис	Дата		

У дипломній роботі оглянуто існуючі програмні рішення і визначено функції, які повинен містити розроблюваний веб-додаток. Також оглянуто програмні можливості і реалізація даної програми.

					ІАЛЦ.467200.003 ПЗ	Арк.
						7
Зм.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

1.1 Огляд існуючих проблем

В зв'язку з політичною ситуацією, яка виникла в Україні, досить багато джерел інформації стали недоступними для мешканців нашої країни. Десятки сайтів, звідки бралася інформація, відповіді на питання, довідники по певним засобам програмування були заблоковані, так як вони були створені країною агресором. Іншомовна документація залишилася, проте навчання на нерідній мові дається набагато важче. Тому на даний час пошук інформації став набагато складнішим. Досить важливо знати де саме можна дістати потрібну інформацію чи документацію, а саме головне в зрозумілому для нас вигляді. Адже сприйняття інформації у кожного різне, те ж саме можна сказати і про її розуміння.

Цим самим складнішим стає адаптація нового робітника у компанія. Досить великі і масштабні проекти складаються з різних фреймворків, засобів програмування. Новий співробітник може всі їх не знати і для кращого розуміння проекту, а також його подальшої роботи доведеться вивчати нове. В розділі 1 ми розглянемо основні програми та способи вивчення нового матеріалу для кращої адаптації співробітника. Наприклад, слід ознайомитись з фреймворком Django мови програмування Python. На рисунку 1.1 зображено, що отримаємо в пошуковій системі Google при даному запиті.

					ІАЛЦ.467200.003 ПЗ	Арк.
						8
Зм.	Арк.	№ докум.	Підпис	Дата		

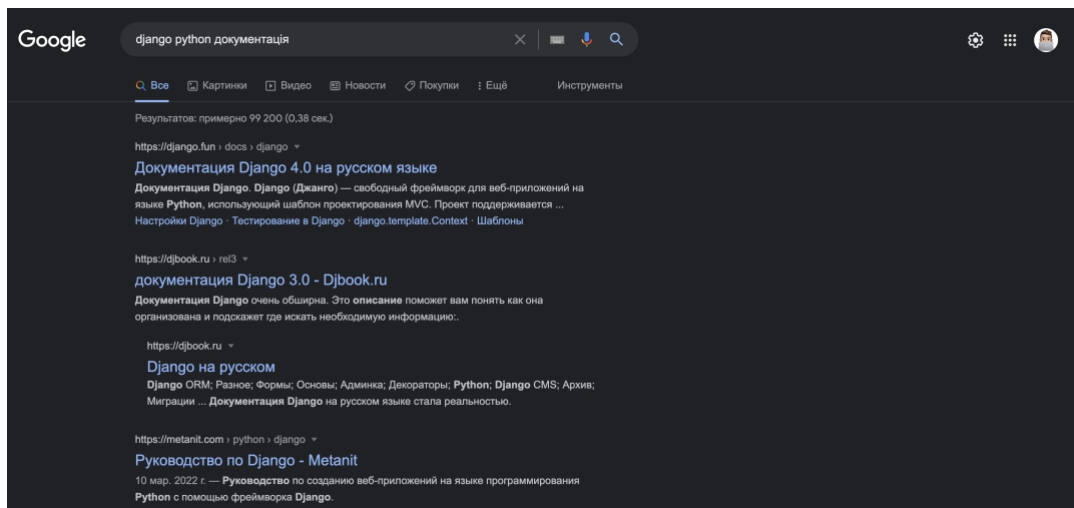


Рисунок. 1.1 – Пошук інформації в пошуковій системі Google

Перші 4 сайти, які видала пошукова система недоступні у нашій країні. Саме тому досить складно знайти ресурс, з якого можна чітко і зрозуміло знайти потрібну нам інформацію. Нижче будуть розглянуті основні, на нашу думку, засоби та методи пошуку інформації.

Задачі глобальної оптимізації застосовуються в різних областях сучасної науки і техніки. Вони надзвичайно важливі для розвитку економіки в цілому і таких галузей, як нанотехнології, мікроелектроніка, біологія, легка і важка промисловість. У зв'язку з цим розробка ефективних методів вирішення таких завдань і їх реалізація у вигляді прикладних програмних комплексів представляється актуальною науковою і практичною проблемою [3].

1.2 Огляд наявних програм для адаптації нових співробітників

Адаптація – звикання людини до нового середовища, нових умов(роботи). Другими словами це пристосування. Досить часто адаптація проходить складно. Саме від результату адаптації залежить наскільки швидко новий співробітник зможе почати працювати з новим проектом і скільки часу йому знадобиться для ознайомлення і розуміння уже готового продукту.

					ІАЛЦ.467200.003 ПЗ	Арк.
						9
Зм.	Арк.	№ докум.	Підпис	Дата		

Адаптуватися до вже сформованої соціальної групи складно. Люди часто сприймають новеньких несерйозно або холодно, однак це не означає, що так буде завжди. Готуйтеся до того, що на початку співпраці ви не відчуватимете морального емоційного розслаблення, як було раніше. Зміна проекту — це стрес, тому потрібно бути до нього готовим. Період пристосування до проекту та команди — неминучий етап для кожного. Упродовж цього часу спеціаліст лише вчиться виконувати посадові обов’язки на належному рівні та часто робить помилки.

На сьогоднішній день ринок ІТ-фахівців перегрітий через велику кількість кандидатів і пропозицій, конкуренції як між інженерами за проект, так і між компаніями за спеціалістами. Тому чим швидше новий працівник адаптується до нового проекту, нових засобів програмування – тим більше шансів у нього залишитися на роботі і досягти подальших успіхів. На даний момент, як таких аналогів немає. Можна виокремити документацію проекту, яка досить швидко допоможе розібратися в коді проекту, проте якщо, наприклад, в частині проекту буде новий фреймворк чи методи з якими новий працівник не стикався ще – йому досить складно буде повністю розібратися в коді, а в наслідок чого приступити до повноцінної роботи. Для порівняння продукту дипломної роботи, спочатку пройдемося по ринку і проаналізуємо можливі програми для спрощення адаптації нового працівника.

1.3 Документація проекту

Документація – це робоча область вашого проекту. У будь-який момент будь-який співробітник може швидко знайти потрібну йому інформацію і зрозуміти, як вирішити конкретні проблеми. А документи позбавлять вас від зайвих конфліктів, адже в них чітко прописано, хто що робить, хто за що відповідає, як працює система і що робити, якщо щось піде не так.

					ІАЛЦ.467200.003 ПЗ	Арк.
						10
Зм.	Арк.	№ докум.	Підпис	Дата		

Призначення документації наступне:

- забезпечити грамотну і злагоджену роботу усіх співробітників команди;
- вести облік ІТ-активів, планувати та оптимізувати витрати на їх обслуговування;
- отримати впевненість, що у разі збою система та всі підключені сервіси будуть швидко відновлені;
- актуалізувати нормативну базу підприємства, вчасно замінювати застарілі регламенти та інструкції;
- упорядкувати роботу ІТ-відділу, зробити її ефективною та зрозумілою іншим співробітникам.

І саме головне – описує та документує ІТ-інфраструктуру, щоб кожен краще розумів, як вона працює,. Ви можете приймати дійсно ефективні управлінські рішення. Документи бувають різними, і ви самі вирішуєте, які з них для вас головні, а без яких можна обійтися. Крупні компанії, сильно залежні від ІТ, розробляють весь комплекс документації.

Різні види документації технічного характеру створюються протягом життєвого циклу розробки програмного забезпечення. Є документація види документацій, а самі така, яка пояснює функціональність продукту, дозволяє обговорювати всі важливі питання, що виникають між замовником і командою, уніфікує інформацію, пов'язану з проектами. На рисунку 1.2 зображений приклад документації фреймворку Django.

					ІАЛЦ.467200.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		11

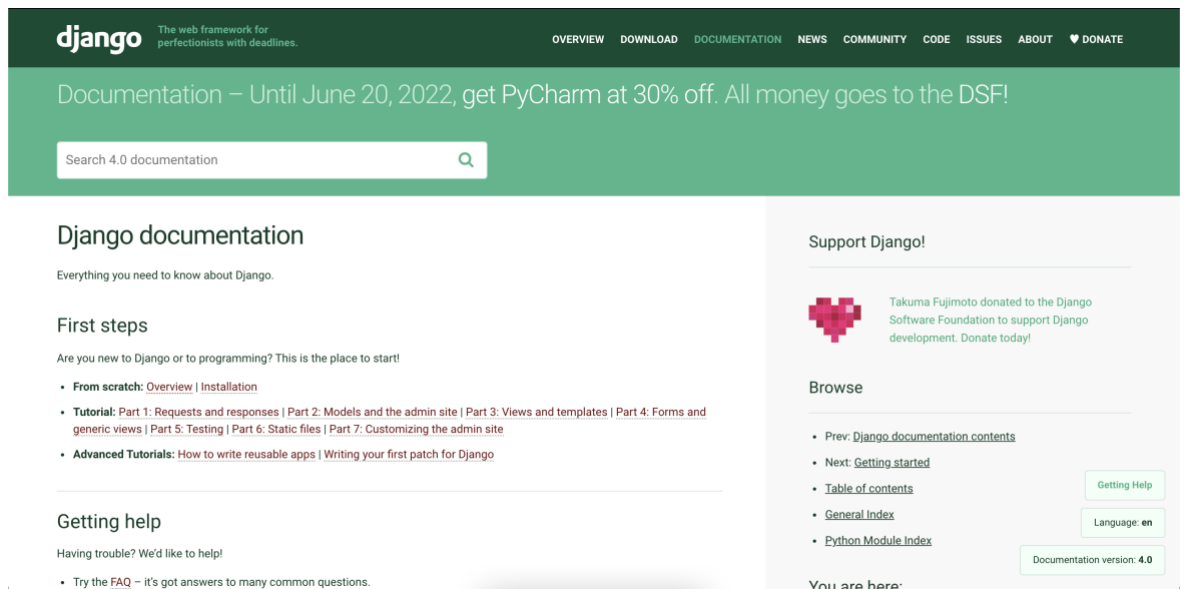


Рисунок 1.2 – Документація фреймворку Django

Від обраного підходу до розробки програмного забезпечення залежать види документації(рисунок 1.3), яку готує команда, та їх обсяг. Наприклад, підхід Agile базується на командній роботі, тісній співпраці з клієнтами та зацікавленими сторонами, гнучкості та здатності швидко реагувати на зміни. Agile спочатку не вимагає вичерпної документації. Не потрібно багато планувати заздалегідь, тому що в міру розвитку проекту ситуація може змінитися.

					ІАЛЦ.467200.003 ПЗ	Арк.
						12
Зм.	Арк.	№ докум.	Підпис	Дата		

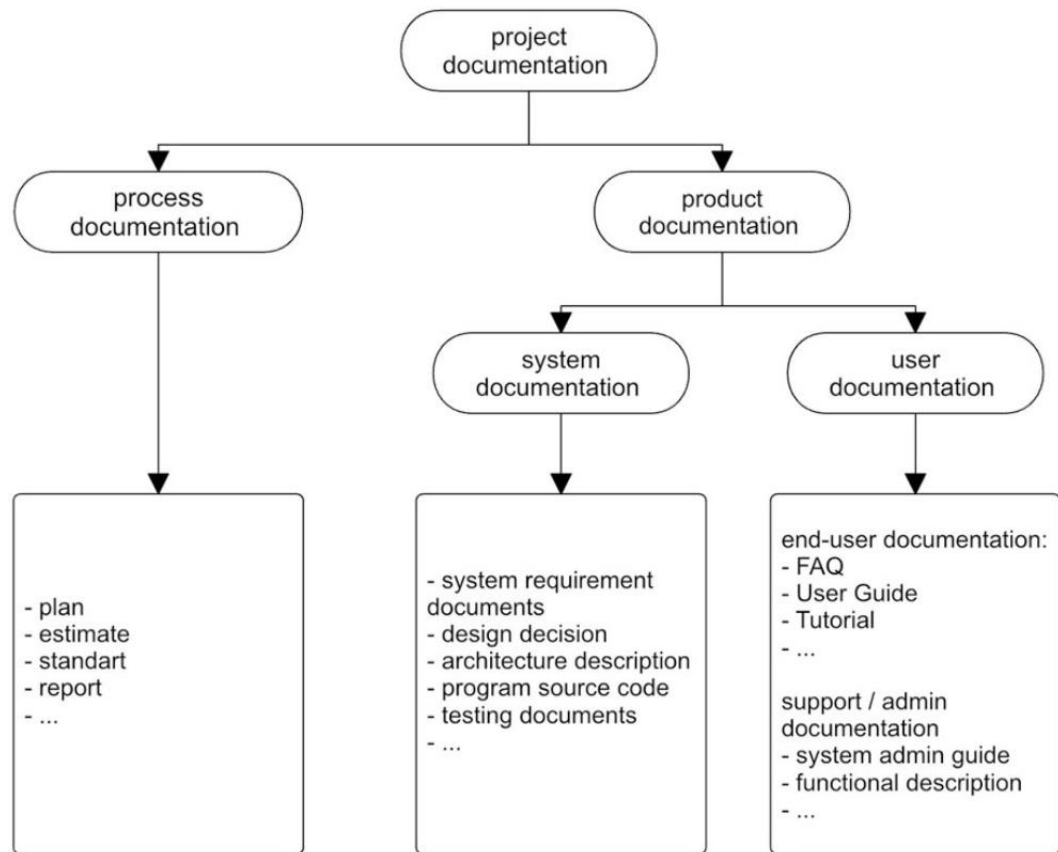


Рисунок 1.3 – Структурна схема документування

Усю документацію можна розділити на дві основні категорії:

- process documentation;
- product documentation.

Product documentation описує продукт, який розробляють. Він містить інструкції щодо виконання різних завдань з продуктом. Загалом можна сказати, що це вимоги, технічні характеристики, логіка та мануали. Два основних типи product documentation це:

- System documentation — документи, що описують саму систему та її частини. Це Architecture descriptions, Requirement documents, Program Source code, Testing documents, Design decisions,.

- User documentation — це інструкції, які підготовлені переважно для кінцевих користувачів продукту та команди підтримки. Це Tutorials, Installation, Reference manuals, Troubleshooting manuals, User guides.

Process documentation описують процес. Можуть бути Standards, Project plans, Test schedules, Reports, Meeting notes тощо.

1.4 Месенджери

Messenger — це запатентована програма та платформа для обміну миттєвими повідомленнями. Кількість таких програм на даний час дуже важко порахувати. Можна виділити найбільш актуальні та популярні. Сюди відносяться Telegram, Discord, Viber. Також вони зберігають історію переписок, картинки, відео, медіа. В якому плані вони можуть бути корисні новому працівнику? В даний час все більш поширеною стає робота на дому замість офісу. Тому дзвінки, переписки неминучі. Якщо ж ви новий працівник – у вас будуть з’являтися безліч різних питань, на які потрібно буде дати відповідь. Щоб дати вам відповідь, потрібно щоб учасник команди відволікався та тратив свій час, щоб дати вам відповідь, замість того, щоб далі працювати над проектом. Тому чим довше буде адаптація – тим більше часу та сил потратить новий співробітник та команда. Нижче приведений приклад одного з месенджерів – Discord:

					ІАЛЦ.467200.003 ПЗ	Арк.
						14
Зм.	Арк.	№ докум.	Підпис	Дата		

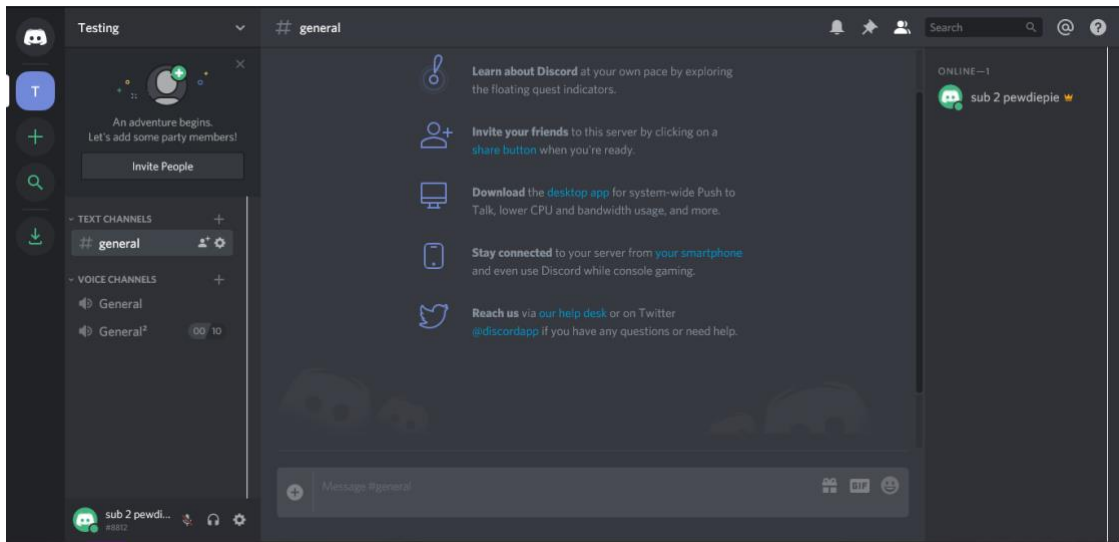


Рисунок 1.4 – Інтерфейс програми Discord

Discord приємний і простий на вид. Щоб ним користуватися не слід мати багато навичок, зможе навіть дитина. Проте якщо переписок буде дуже багато – повідомлення будуть губитися і їх важко буде потім знайти.

Розглянемо інтерфейс серверу та його функціональність:

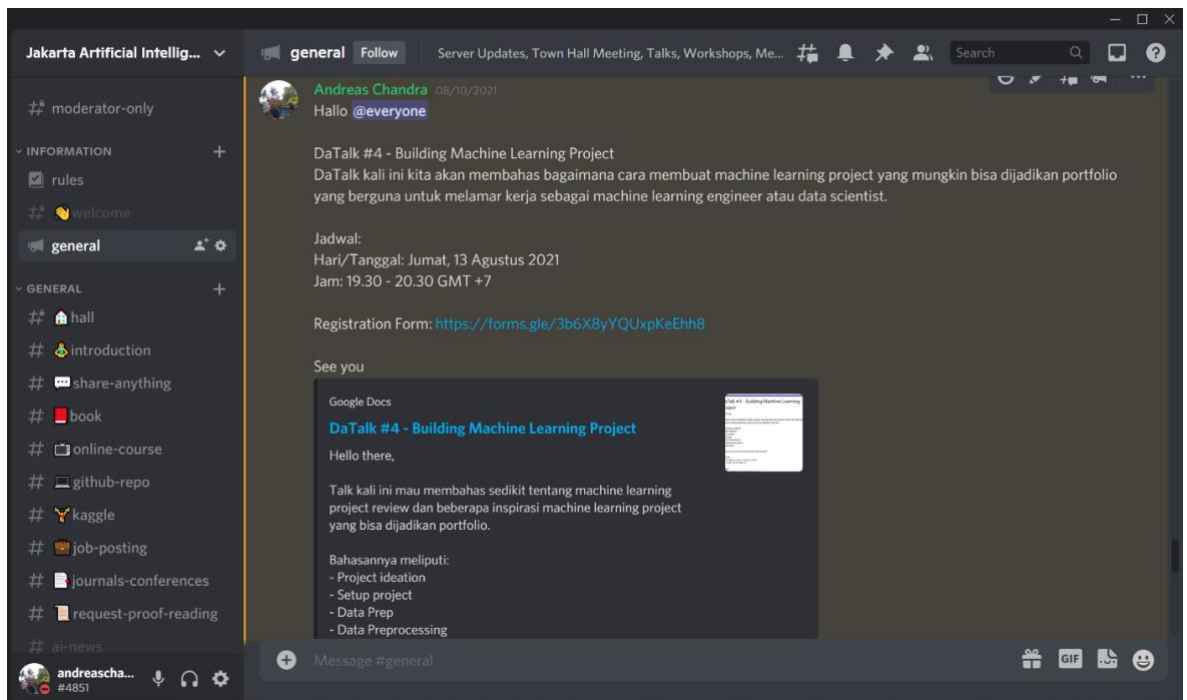


Рисунок 1.5 – Інтерфейс серверу в Discord.

					ІАЛЦ.467200.003 ПЗ	Арк.
						15
Зм.	Арк.	№ докум.	Підпис	Дата		

На кожному сервері можемо створити окремі кімнати, де можна спілкуватись. Також є можливість аудіо і відео дзвінків. Великим плюсом є можливість показу свого екрану. Ця програма може працювати в фоновому режимі і споживати мінімум ресурсу персонального комп'ютера, що є досить великим плюсом у її використанні. Великим плюсом цієї програми ще є те, що вона регулярно оновлюється. Також слід відзначити і стабільність серверів і безперебійну роботу. Не часто бувають випадки, коли Discord різко припиняє роботу або має іншу технічну проблему. Проте все ж таки, такі випадки трапляються.

Також розглянемо аналог Discord, а саме Телеграм. На даний момент телеграм стрімкіше розвивається за Discord. А останні оновлення, які включають в себе показ екрану і зовсім похитнули шанувальників Discord. Адже завдяки цьому він значно вирізнявся на фоні Телеграму. Проте, телеграм споживає більше ресурсів за Discord, що є значним мінусом. Телеграм має наступний вигляд:

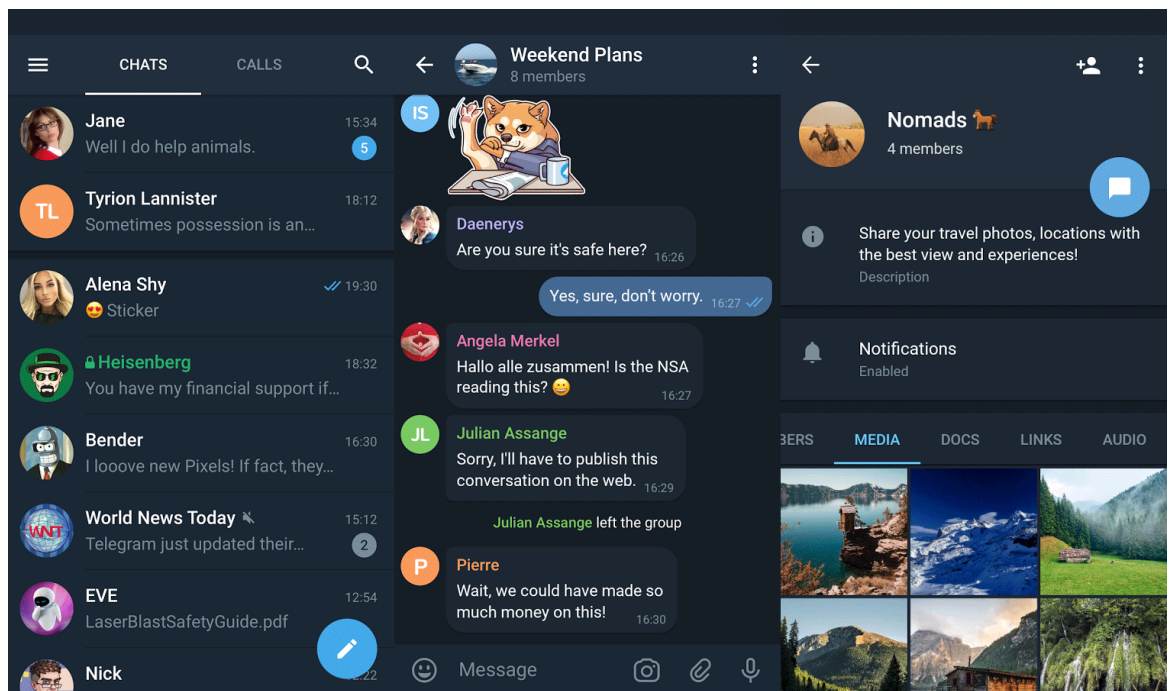


Рисунок 1.6 -Інтерфейс Телеграму

Слід відзначити ще те, що Телеграм має більш приємний інтерфейс і простіший у використанні за Discord. До плюсів Телеграму ще можна віднести налаштування фону, зміну теми, оформлення. Також, якщо порівняти цифри використання цих програм, можна дійти до висновку, що Телеграм в декілька разів популярніший за Discord. Сама ця цифра говорить, наскільки потужна ця програма. Тому, якщо зрівнявши їх, можна сказати, що Телеграм буде краще підходити для обговорюваною нами проблеми. Проте і вибравши Discord ви також нічого не втратите.

1.5 Stack Overflow

Початкова сторінка Stack Overflow має наступний вигляд:

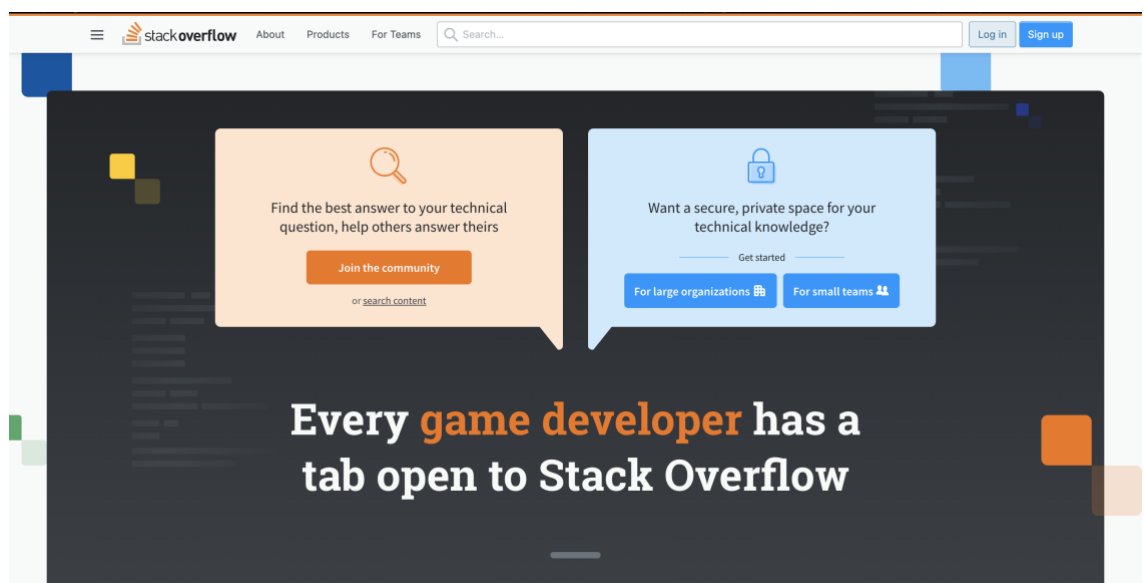


Рисунок 1.7 – Інтерфейс Stack Overflow

Stack Overflow — система питань і відповідей з програмування. Система відома на весь світ, адже кожен програміст хоча б раз в житті заходив на неї та шукав відповіді на свої питання. Stack Overflow приймає лише питання про програмування, які зосереджені на певній проблемі. Запитання більш широкого

					ІАЛЦ.467200.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		17

характеру — або ті відповіді, які за своєю природою є предметом думки — зазвичай відхиляються користувачами сайту та позначаються як закриті.

Завершальні запитання є основною відмінністю від інших сайтів запитань і відповідей. Механізм капітально відремонтований у 2013 році. Питання, відредаговані після того, як їх призупинили, тепер з'являються в черзі перегляду. Джефф Етвуд заявив у 2010 році, що повторювані запитання не розглядаються як проблема, а навпаки, вони є перевагою, якщо такі додаткові запитання залучають додатковий трафік на сайт за рахунок множення відповідних звернень за ключовими словами в пошукових системах.

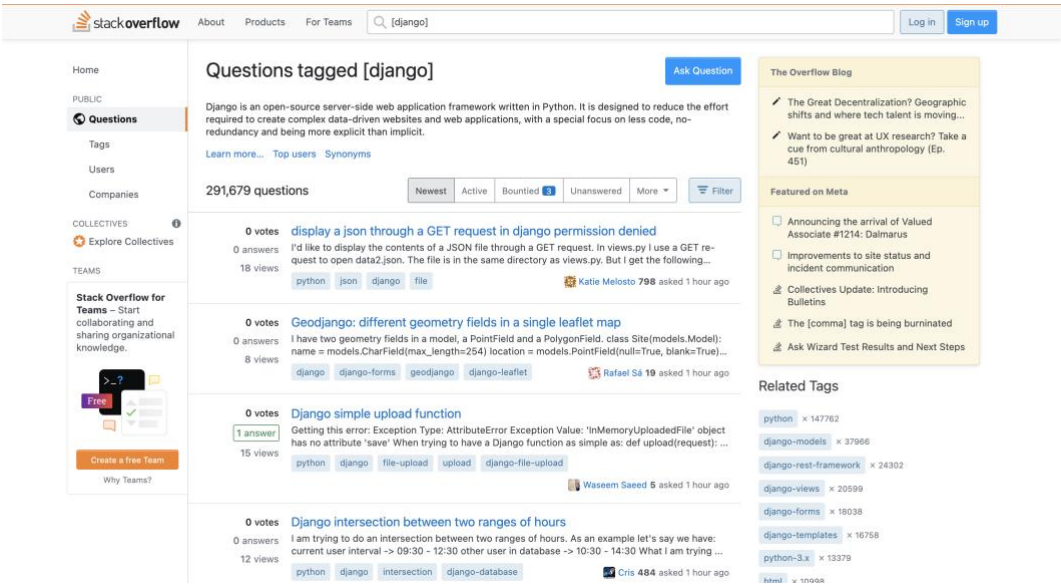


Рисунок 1.8 – Інтерфейс питання та відповідей на Stack Overflow

Весь контент, створений користувачами, ліцензується за ліцензією Creative Commons Attribute-ShareAlike версії 2.5, 3.0 або 4.0 залежно від дати розміщення вмісту.

Проте якщо новий працівник не знайде відповідь на своє запитання – він може його задати сам. Але відповідь отримає не відразу. Адже слід буде чекати до того часу, інший користувач дасть на нього відповідь. Може пройти як декілька хвилин, так і днів. Тому ця програма досить неоднозначна для

адаптації нового співробітника, скоріше для вирішення окремих питань з програмування.

1.6 Опис розроблюваної програми

Розроблювана нами програма буде включати основні переваги наведених вище програм та способів. Дана програма візьме переваги кожної з них для кращої адаптації нових співробітників у команді. Також ця програма може бути корисною навіть для простих користувачів, які мають намір щось вивчити. Саме головне правильно оформити зміст і ввести та описати потрібну інформації.

З месенджерів візьмемо можливість переписуватись. Проте для цього будемо створювати окремі кімнати з різними темами, на які будемо спілкуватись. Обмін повідомленнями буде проходити в реальному часі. Якщо проект досить великий, і має досить багато тем для обговорення – на кожен окрему тему можна буде створити свій сервер, на якому можна буде в різних кімнатах обговорювати конкретну тему. Таким чином обговорення відбуватися у спеціально відведеному для цього місці – не перешкоджаючи і не замінюючи інформаційні ресурси для вивчення нового матеріалу.

З документації візьмемо можливість детально описати проект. Для кожного фреймворку чи фрагменту коду можна буде зробити чіткий опис. При цьому на цьому сервері можна створити кімнату, де буде ознайомлення з самим фреймворком. Якщо розглядаючи документації функції чи класу чи чогось іншого, ви щось не можете понять – завжди можна перейти в кімнату, де буде опис та принцип функціонування самого фреймворку, що в свою чергу допоможе швидше розібратись з адаптацією чи навчанням.

З Stack Overflow візьмемо можливість створювати питання, де користувачі команди по можливості зможуть відповідати на ці питання та допомогти розібратись в тому чи іншому розділі.

					ІАЛЦ.467200.003 ПЗ	Арк.
						19
Зм.	Арк.	№ докум.	Підпис	Дата		

Тож, програма стане втіленням найкращих рис переглянутих вище програм, які мають покращити адаптацію нового співробітника. Тим паче дана програма має мати попит у вивченні нового матеріалу для простих користувачів.

1.7 Вимоги до програмного продукту

Враховуючи предметну область та аналіз програм, які призначені під цю програмну область можна виділити мінімальні вимоги до нашого проекту:

- простий і зрозумілий інтерфейс;
- можливість легко розмістити інформацію;
- поділ інформації на теми;
- пошук інформації по темам, кімнатам;
- кожен користувач має мати свій власний аккаунт;
- для кожного користувача різні права для зміни кімнат, повідомлень.

Щоб програма ефективно виконувала свою роль – слід чітко дотримуватись виконання мінімальних вимог до програми. Лише за цих умов програма принесе користь тому, хто буде її використовувати і полегшить досліджувану задачу.

1.8 Порівняння розроблюваної програми з іншими

Для кращого порівняння програм створимо таблицю порівняння.

Таблиця 1.1 - Порівняння програм

	Розроблювана програма	Документація	Telegram	Stack Overflow
Опис проекту	Містить	Містить	Не містить	Не містить

Продовження таблиці 1.1

	Розроблювана програма	Документація	Telegram	Stack Overflow
Надсилання повідомлень	Містить	Не містить	Містить	Не містить
Пошук по темам(розбиття матеріалу на підгрупи)	Містить	Містить	Не містить	Містить
Відео, аудіо дзвінки	Не містить	Не містить	Містить	Не містить
Виділення проблеми і обговорення рішення	Містить	Не містить	Містить	Містить

Як ми бачимо після порівняння поданого у таблиці 1.1. Наша програма краще виконує свою функцію за оглянуті програми в розділі 1. Проте у більш глобальному розумінні дані програми будуть значно краще за розроблювану нами програму. Так як програма направлена суто на полегшення вивчення того чи іншого матеріалу і для кращої адаптації. Саме з-за цього наша програма має перевагу за нашою таблицею. Якби ми брали інші, різносторонні критерії – розроблювана нами програми не виглядала б так домінуючи.

ВИСНОВОК ДО РОЗДІЛУ 1

Існує безліч методів та способів полегшення адаптації нового співробітника до нового проекту, команди або для набуття нових знань. В першому розділі було проаналізовано причини складності цього процесу. Та знайдено і розглянуто кращі підходи та варіанти для поставленої задачі.

Кожна з розглянутих програм мала як свої переваги так і недоліки. Було розглянуто основні недоліки і переваги програм для конкретної ситуації. Досить складно буде адаптуватись використовуючи лише одну програму. На основі розглянутих програм, а також враховуючи необхідні вимоги для рішення конкретної задачі було сформульовано основні функції нашої програми. Наш продукт – це втілення основних переваг кожної програми з урахуванням їх недоліків та виправленням їх. Готова програма повинна зробити адаптацію нового співробітника в команду легшою, а ознайомлення з матеріалом змістовним і зрозумілим.

В Розділі 1 визначили мінімальні вимоги до нашої програми, що дозволить в свою чергу краще поставити технічне завдання для його програмування.

					ІАЛЦ.467200.003 ПЗ	Арк.
						22
Зм.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ОГЛЯД ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ ПРОГРАМИ

Програмний прикладний інтерфейс для доступу до інформаційних ресурсів буде створено у форматі клієнт-серверного веб-застосунку.

Відповідно, задачею даного розділу буде розглянути різні мови програмування, фреймворки, СКБД, додаткові утиліти та сформувати стек технологій, які будуть найкраще підходити для розробки даної програми.

2.1 Технології для реалізації серверної частини

2.1.1 Python

Python – це інтерпретована, об’єктно-орієнтована мова програмування високого рівня з динамічною семантикою. Python дуже привабливий для швидкої розробки додатків, також привабливий для використання в якості мови сценаріїв або склеювання для з’єднання існуючих компонентів разом завдяки його високорівневі вбудованій структурі даних, яка поєднується з динамічною типізацією та з динамічним зв’язуванням. Він має простий синтаксис, який можна легко вивчати, підкреслює читабельність і, отже, цим самим знижує витрати на обслуговування програми. Python підтримує модулі та пакунки, що в свою чергу сприяє модульності програм і повторному використанню написаного коду. Великі стандартні бібліотеки і інтерпретатор Python доступні у вихідній або двійковій формі та безкоштовні для всіх основних платформ, а також можуть вільно поширюватися.

Нерідко програмісти захожуються в Python з-за його підвищену продуктивність, яку він забезпечує. Оскільки етапу компіляції немає, цикл який

					ІАЛЦ.467200.003 ПЗ	Арк.
						23
Зм.	Арк.	№ докум.	Підпис	Дата		

має вигляд: редагування-тестування-налагодження неймовірно швидкий. Налагоджувати програми у Python досить легко: помилка або неправильне введення ніколи не призведе до помилки сегментації. Натомість, коли інтерпретатор виявляє помилку, він створює виняток. Якщо програма не вловлює виняток, інтерпретатор друкує трасування стека, що в свою чергу допоможе виявити корінь проблеми. Налагоджувач на рівні вихідного коду дозволяє перевіряти локальні та глобальні змінні, оцінювати довільні вирази, встановлювати точки зупину, переходити через код по рядку і так далі. Сам налагоджувач написаний на самому Python, що свідчить про інтроспективну силу Python. З іншого боку, часто найшвидший спосіб налагодити програму без налагоджувача – це додати кілька операторів print до джерела: швидкий цикл редагування-тестування-налагодження робить цей простий підхід дуже ефективним і використовуваним.

Також у Python є свій Zen. Zen – лаконічні принципи щодо дизайну на Python, які записав Тім Пітерс. Його можна побачити увівши в інтерпретаторі `import this`. Результатом буде рисунок 2.1.

```
Beautiful is better than ugly.  
Explicit is better than implicit.  
Simple is better than complex.  
Complex is better than complicated.  
Flat is better than nested.  
Sparse is better than dense.  
Readability counts.  
Special cases aren't special enough to break the rules.  
Although practicality beats purity.  
Errors should never pass silently.  
Unless explicitly silenced.  
In the face of ambiguity, refuse the temptation to guess.  
There should be one-- and preferably only one --obvious way to do it.  
Although that way may not be obvious at first unless you're Dutch.  
Now is better than never.  
Although never is often better than *right* now.  
If the implementation is hard to explain, it's a bad idea.  
If the implementation is easy to explain, it may be a good idea.  
Namespaces are one honking great idea -- let's do more of those!
```

Рисунок 2.1 – Python Zen

					ІАЛЦ.467200.003 ПЗ	Арк.
						24
Зм.	Арк.	№ докум.	Підпис	Дата		

На рисунку 2.2 приведений код програми, що рахує факторіал уведеного числа, написаний на мові Python.

```
# importing math library
import math
# Taking input from user
num = int(input("Enter the number to find factorial: "))

# Finding factorial of the given number
fact = math.factorial(num)

print("Factorial of the given number ", num, " is: ", fact)
```

Рисунок 2.2 – Приклад коду написаного на мові програмування Python

Python має як переваги, так і недоліки. Серед переваг:

- Розширюваність - один із слоганів мови звучить як - Just Import! — що повністю пояснює, наскільки мову розширюємо та було розширено за останні роки. Існують бібліотеки та фреймворки під будь-який тип завдань та потреб. Також величезним плюсом є те, що можна використовувати C-код з Python.
- Простота синтаксису - код чистий і зрозумілий без зайвих дужок та виразів.
- Гнучкість — це одна з основних переваг мови, оскільки завдяки своїй гнучкості мова набула популярності серед багатьох розробників.
- Інтерпретованість. Інтерпретатор Python існує для всіх популярних платформ і за замовчуванням входить до більшості дистрибутивів Linux, а значить, є на більшості серверів «з коробки».

- PEP — єдиний стандарт для написання коду, що робить код підтримуваним і читати навіть при переході від одного програміста до іншого. Це підтримує популярність Python.
- Open Source — код інтерпретатора Python є відкритим, що дозволяє будь-кому, хто зацікавлений у розвитку мови взяти участь у її розробці та покращити її. Якщо дивитися деталі релізу однієї з версій мови, можна помітити, що великі частини нового функціоналу реалізовані сторонніми розробниками.
- Ком'юніті — навколо Python утворилося досить дружнє і приємне ком'юніті, яке готове прийти на допомогу будь-якому початківцю або вмілому розробнику і розібратися в його проблемі.

Але незважаючи на всі явні переваги, Python також має ряд недоліків, серед них:

- Продуктивність. Більшість розробників, та й сам автор мови, сходяться на думці, що Python не настільки швидкий, наскільки хотілося б. Це пов'язано з тим, що Python інтерпретується. Але навіть у порівнянні з іншими мовами, що інтерпретуються, помітно, що Python програє у продуктивності. Але це легко можна нівелювати за допомогою C реалізацій тієї чи іншої проблемної ділянки коду. В умовах сьогоденних потужностей це не дуже помітно.
- Global Interpreter Lock. На даний момент це є основною проблемою продуктивності Python, а також цим обумовлена погана реалізація багатопоточності. Код GIL не змінювався з першої версії мови. Це явно свідчить про те, що він застарів. Залишається сподіватися, що розробники приділять цьому увагу найближчими релізами.

Виходячи з вищепереліченого - всі недоліки мови з лишком нівелюються її перевагами, які набагато вагоміші в сьогоденних реаліях. І не варто забувати,

					ІАЛЦ.467200.003 ПЗ	Арк.
						26
Зм.	Арк.	№ докум.	Підпис	Дата		

що мова розвивається. Це дає надію, що всі недоліки будуть або скорочені, або усунуті зовсім.

Python - це відмінна мова програмування як для навчання, так і для реальної розробки. Він допомагає вирішити величезний спектр завдань!

2.1.2 Java

Java — це багатоплатформна, об'єктно-орієнтована та мережево-орієнтована мова. Це одна з найбільш використовуваних мов програмування. Java також використовується як обчислювальна платформа. Фрагмент коду вирахування факторіалу числа зображений на рисунку 2.3.

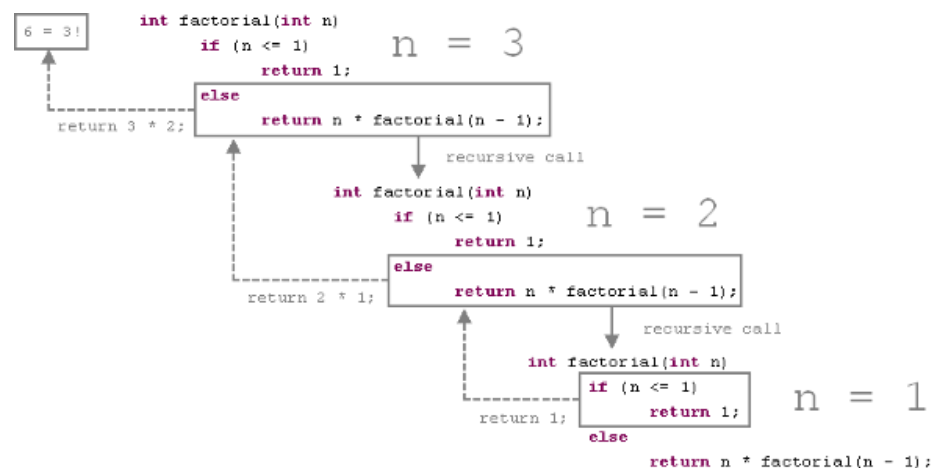


Рисунок 2.3 - Приклад коду написаний на мові програмування Java

Віртуальна машина Java (JVM) — це механізм, який забезпечує середовище виконання для керування кодом Java або програмами. Він перетворює байт-код Java на машинну мову. JVM є частиною середовища виконання Java (JRE). В інших мовах програмування компілятор створює машинний код для певної системи. Однак компілятор Java створює код для віртуальної машини, відомої як віртуальна машина Java.

Як і компілятор C, компілятор Java не створює власний виконуваний код для певної машини. Замість цього Java створює унікальний формат, який

називається байт-кодом. Він виконується відповідно до правил, викладених у специфікації віртуальної машини. Таким чином, Java є незалежною від платформи мовою.

Байт-код зрозумілий для будь-якої JVM, встановленої на будь-якій ОС. Коротше кажучи, вихідний код Java може працювати на всіх операційних системах.

Переваги Java:

- Легкість вивчення: дану мову програмування можна швидко вивчити і домогтися високої продуктивності програмування.
- Відкриті джерела: завдяки доступності вихідного програмного коду можна легко налагодити програму або налаштувати її під свої власні потреби.
- Розширення: існує безліч вільно доступних розширень та вихідного коду для виконання будь-яких прикладних задач, починаючи від маніпуляції з XML до доступу до каталогів. Програмісти також можуть використовувати цю масу готового коду для швидкого компілювання найсучасніших додатків.
- Безпека: віртуальна Java-машина повністю визначає і обмежує функціонування програми.
- Надійність: у мові відсутній механізм, потенційно призводячий до помилок.
- Переносимість: програми Java після одноразової трансляції в байт код можуть бути виконані на будь-якій платформі, яка підтримує віртуальну машину Java.

Недоліки Java:

- Мова в повному обсязі об'єктно-орієнтований: це означає, що мова не має таких властивостей мови, як індивідуальні змінні, множинне спадкування і.т.д.

					ІАЛЦ.467200.003 ПЗ	Арк.
						28
Зм.	Арк.	№ докум.	Підпис	Дата		

- Ресурсозатрати: для запуску застосунку потребує велику кількість ресурсів.

2.1.3 C#

C# — це сучасна, об'єктно-орієнтована та безпечна для типів мова програмування. Вона дозволяє розробникам створювати багато типів безпечних і надійних програм, які працюють у .NET. C# має свої коріння в сімействі мов C і буде відразу знайома програмістам, які вивчали C, C++, Java та JavaScript.

C# — це компонентно-орієнтована мова програмування. Вона надає мовні конструкції для безпосередньої підтримки цих концепцій, роблячи її природною мовою для створення та використання програмних компонентів. З моменту свого виникнення, в C# були додані функції для підтримки нових робочих навантажень і нових методів проектування програмного забезпечення. За своєю суттю C# є об'єктно-орієнтованою мовою. Фрагмент коду вирахування факторіалу числа зображений на рисунку 2.4.

```

0 references
static void Main(string[] args)
{
    int rslt= FactorialWithoutR(5);
    Console.WriteLine("Factorial is :{0}", rslt);
}

1 reference
public static int Factorial(int limit)
{
    // Using Recursion
    if (limit == 0)
    {
        return 1;
    }
    return limit * Factorial(limit - 1);
}

```

Рисунок 2.4 - Приклад коду написаний на мові програмування C#

До переваг C# можна віднести:

- Універсальність.

					ІАЛЦ.467200.003 ПЗ	Арк.
						29
Зм.	Арк.	№ докум.	Підпис	Дата		

- Швидкий компілятор.
- Безпека і швидкість виконання.
- Велика стандартна бібліотека.

До недостатків можна віднести:

- Перенасиченість синтаксису призводить до підвищення порогу входження.
- Погана типізація: номінальна система типів періодично заставляє писати більше коду ніж потрібно.

2.1.4 JavaScript

JavaScript (скорочення JS) — це легка, інтерпретована, об'єктно-орієнтована мова з першокласними функціями. Вона найбільш відома як мова сценаріїв для веб-сторінок, але також використовується в багатьох середовищах, що не є браузерами. JS заснована на прототипах, мультипарадигмальній мові сценаріїв, яка є динамічною та підтримує об'єктно-орієнтований, імперативний та функціональний стилі програмування.

JavaScript працює на стороні клієнта в Інтернеті, що можна використовувати для розробки/програмування поведінки веб-сторінок під час виникнення події. JavaScript — це легка для вивчення, а також потужна мова сценаріїв, широко використовується для керування поведінкою веб-сторінки. Фрагмент коду вирахування факторіалу числа зображений на рисунку 2.5.

```
function factorial(n) {
    if (n < 0) {
        return undefined;
    }
    if (n === 1 || n === 0) {
        return 1;
    }
    return n * factorial(n - 1);
}

let value = factorial(5);
```

Рисунок 2.5 - Приклад коду написаний на мові програмування JavaScript

Динамічні можливості JavaScript включають побудову об'єктів під час виконання, списки змінних параметрів, змінні функцій, створення динамічного сценарію (через eval), інтроспекцію об'єкта (через for ... in) і відновлення вихідного коду (програми JavaScript можуть декомпілювати тіла функцій назад у вихідний текст).

Переваги JS:

- Швидкість: мова є дуже швидкою, оскільки вона часто запускається відразу в браузері клієнта.
- Простота: синтаксис JS був натхнений від Java і його порівняно легко можна вивчити.
- Популярність: JS є однією з найбільш популярних мов програмування.
- Сумісність: JavaScript можна вставити на будь яку сторінку веб-браузера.
- Навантаження сервера: працює на стороні клієнта.
- Розширені інтерфейси: розробник може розширити функціональність веб-сторінки, написавши фрагмент коду для сторонніх доповнень.
- Універсальність.
- Оновлення: мова регулярно оновлюється.

До недоліків даної мови можна віднести наступне:

- Захист на стороні клієнта: оскільки код виконується на стороні клієнта, помилки та недогляди іноді можуть бути використані для зловмисних цілей.
- Підтримка браузера: різні браузери іноді по різному інтерпретують код, хоча сценарії на стороні сервера завжди дають однакові результати.

2.1.5 Rust

Rust було створено для забезпечення високої продуктивності, подібної до тієї, яку пропонують C і C++, але з акцентом на безпеку коду, недолік якої є ахіллесовою п'ятою цих двох мов. Однак у Rust є більше, ніж просто безпека пам'яті. Висока продуктивність при обробці великих обсягів даних, підтримка паралельного програмування, і це разом з ефективним компілятором — інші причини, чому відомі важковаговики програмного забезпечення зараз використовують цю мову програмування. Firefox, Dropbox, Cloudflare та багато інших компаній від стартапів до великих корпорацій використовують Rust у виробництві.

Rust — це мова програмування зі статичним типом, розроблена для забезпечення продуктивності та безпеки, особливо безпечного паралельного використання та керування пам'яттю. Його синтаксис схожий на синтаксис C++. Це проект з відкритим кодом, спочатку розроблений Mozilla Research. У 2021 році Rust Foundation взяв факел і керує розвитком мови.

Rust вирішує проблеми, з якими розробники C/C++ боролися протягом тривалого часу: помилки пам'яті та одночасне програмування. Це розглядається як його головна перевага. Однак це не єдині приклади того, чим Rust відрізняється від C++.

У Rust є два режими написання коду: Safe Rust і Unsafe Rust. Safe Rust накладає додаткові обмеження на програміста (наприклад, управління

					ІАЛЦ.467200.003 ПЗ	Арк.
						32
Зм.	Арк.	№ докум.	Підпис	Дата		

володінням об'єктом), тим самим забезпечуючи належну роботу коду. Unsafe Rust надає програмісту більше автономії (наприклад, він може працювати з необробленими C-подібними покажчиками), але код може зламатися. Фрагмент коду пошуку факторіалу на Rust зображено на рисунку 2.6.

```
pub fn factorial(num: u64) -> u64 {
    (1..=num).fold(1, |acc, v| acc * v)
}
```

Рисунок 2.6 - Приклад коду написаний на мові програмування Rust

До переваг Rust можна віднести:

- Лаконічний синтаксис, подібний до C.
- Кодовий аналізатор, що не припускає витоку пам'яті.
- Надійна система взаємодії з пам'яттю.

Недоліки Rust:

- Часті поломки компіляції.
- Повільніший за C.

2.1.5 Обрання мови програмування

Для виконання дипломного проекту було вибрано використовувати мову програмування Python. На ній швидко та зручно розробляти веб-додатки за допомогою фреймворку Django.

Django — це високорівневий веб-фреймворк Python, який дозволяє швидко розробляти безпечні та підтримувані веб-сайти. Створений досвідченими розробниками, Django займається більшою частиною клопоту веб-розробки, тому програмісти можуть зосередитися на написанні свого додатка без необхідності винаходити велосипед. Він є безкоштовним і відкритим, має процвітаючу та активну спільноту, чудову документацію.

Переваги Django:

					ІАЛЦ.467200.003 ПЗ	Арк.
						33
Зм.	Арк.	№ докум.	Підпис	Дата		

- Універсальність: його можна використовувати для створення практично будь-якого типу сайтів починаючи системами керування вмістом до соціальних мереж або нових сайтів.
- Безпека: він допомагає програмісту уникнути багато поширених помилок пов'язаних з безпекою, надаючи фреймворк для автоматичного захисту веб-сайту.
- Масштабність: фреймворк використовує архітектуру «не ділиться» на основі компонентів, тобто кожна частина архітектури не залежить від інших і, може бути замінена або змінена, якщо це потрібно.
- Повторне використання: код написаний з використанням принципів проектування та шаблонізації, в наслідок чого, його можна підтримувати та повторно використовувати.
- Портативність: Django написаний на Python, який працює на багатьох платформах.

Логотип фреймворку зображений на рисунку 2.7.



Рисунку 2.7 – Логотип фреймворку Django

MVT (Model View Template) — це шаблон проектування програмного забезпечення. Це набір з трьох важливих компонентів Model View і Template. Модель допомагає працювати з базою даних. Це рівень доступу до даних, який обробляє дані. Шаблон – це рівень презентації, який повністю обробляє частину

					ІАЛЦ.467200.003 ПЗ	Арк.
						34
Зм.	Арк.	№ докум.	Підпис	Дата		

інтерфейсу користувача. Представлення використовується для виконання бізнес-логіки та взаємодії з моделлю для перенесення даних і візуалізації шаблону. Хоча Django дотримується шаблону MVC, але зберігає власні конвенції. Отже, управлінням займається сам.

Нижче на рисунку 2.8 наведено графік, який показує потік керування на основі MVT.

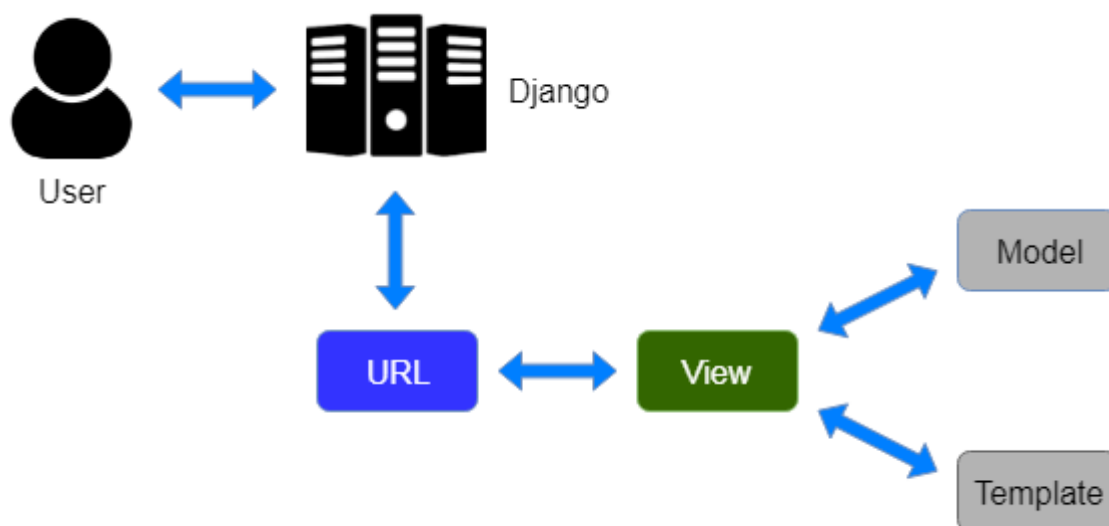


Рисунок 2.8 - Потік керування даних на основі MVT

Тут користувач запитує ресурс до Django, він працює як контролер і перевіряє доступний ресурс в URL. Якщо URL-мапи називають подання, яке взаємодіє з моделлю та шаблоном, воно відтворює шаблон. Django відповідає користувачеві та надсилає шаблон у відповідь.

Перевага цієї архітектурної моделі полягає в тому, що частина контролера, яка приймає введення користувача, обробляється самою платформою. Django приділяє більше уваги моделям (Model), уявленням (View) та шаблонам (Template). Різні компоненти слабо пов'язані. Кожна веб-програма, керована Django, має чітку мету і може бути змінена незалежно, не торкаючись інших частин.

2.2 Підсистема зберігання даних

PostgreSQL - одна з найпопулярніших систем управління базами даних. Сам проект postgresql еволюціонував з іншого проекту, який називався Ingres. Формально розвиток postgresql почався ще 1986 року. Тоді він називався POSTGRES. А в 1996 році проект був перейменований на PostgreSQL, що відображало більший акцент на SQL. І 8 липня 1996 року відбувся перший реліз товару. З того часу вийшло безліч версій postgresql. Поточною версією є версія 14. Однак, регулярно також виходять підверсії. PostgreSQL підтримується всім основних операційних систем - Windows, Linux, MacOS. Логотип PostgreSQL зображено на рисунку 2.9.



Рисунок 2.9 – Логотип PostgreSQL

Перевагами PostgreSQL будуть:

- підтримка БД необмеженого розміру;
- успадкування;
- потужні та надійні механізми транзакцій та реплікації;
- легка розширюваність;
- розширювана система вбудованих мов програмування та підтримка завантаження C-сумісних модулів.

MySQL – це популярний сервер баз даних, що використовується у різних додатках. SQL означає мову структурованих запитів - Structured Query

					ІАЛЦ.467200.003 ПЗ	Арк.
						36
Зм.	Арк.	№ докум.	Підпис	Дата		

Language, яку MySQL використовує для комунікації з іншими програмами. Крім того, MySQL має власні розширені функції SQL для того, щоб забезпечити користувачам додатковий функціонал. Логотип MySQL зображено на рисунку 2.10.



Рисунок 2.10 – Логотип MySQL

Переваги MySQL:

- Простота: легка у встановленні та у використанні.
- Підтримує необмежену кількість користувачів.
- Кількість рядків у таблицях може досягати 50 мільйонів.
- Висока швидкість виконання команд.
- Наявна проста і ефективна система безпеки.

MongoDB — це документно-орієнтована база даних NoSQL, яка використовується для зберігання великих обсягів даних. Замість використання таблиць і рядків, як у традиційних реляційних базах даних, MongoDB використовує колекції та документи. Документи складаються з пар ключ-значення, які є основною одиницею даних у MongoDB. Колекції містять набори документів і функцій, що є еквівалентом таблиць реляційної бази даних. MongoDB — це база даних, яка з'явилася приблизно в середині 2000-х років.

До переваг MongoDB можна віднести:

- Орієнтований на документ. Оскільки MongoDB є базою даних типу NoSQL, замість того, щоб мати дані у форматі реляційного типу, вона зберігає дані в документах. Це робить MongoDB дуже гнучким і адаптованим до реальної ситуації та вимог ділового світу.

- Спеціальні запити – MongoDB підтримує пошук за полями, запитами за діапазонами та пошук за регулярними виразами. Можна робити запити для повернення конкретних полів у документах.
- Індексування – індекси можна створювати для покращення ефективності пошуку в MongoDB. Будь-яке поле в документі MongoDB можна проіндексувати.
- Реплікація – MongoDB може забезпечити високу доступність за допомогою наборів реплік. Набір реплік складається з двох або більше екземплярів mongo DB. Кожен член набору реплік може виконувати роль первинної або вторинної репліки в будь-який час. Основною реплікою є основний сервер, який взаємодіє з клієнтом і виконує всі операції читання/запису. Вторинні репліки зберігають копію даних первинної за допомогою вбудованої реплікації. Коли первинна репліка виходить з ладу, набір реплік автоматично перемикається на додатковий, а потім стає основним сервером.
- Балансування навантаження – MongoDB використовує концепцію шардинга для горизонтального масштабування шляхом розподілу даних між кількома екземплярами MongoDB. MongoDB може працювати на кількох серверах, балансуючи навантаження та/або дублюючи дані, щоб підтримувати роботу системи в разі збою обладнання.

SQLite — це бібліотека в процесі роботи, яка реалізує автономний, безсерверний, транзакційний механізм баз даних SQL з нульовою конфігурацією. Це база даних з нульовою конфігурацією, що означає, що, як і інші бази даних, вам не потрібно налаштовувати її у вашій системі. Движок SQLite не є окремим процесом, як інші бази даних, ви можете зв'язати його статично або динамічно відповідно до ваших вимог із вашою програмою. SQLite отримує прямий доступ до своїх файлів зберігання. Логотип SQLite зображено на рисунку 2.11.



Рисунок 2.11 – Логотип SQLite

Переваги SQLite:

- Не потребує окремого серверного процесу або системи для роботи (безсерверна).
- Поставляється з нульовою конфігурацією, що означає, що налаштування чи адміністрування не потрібні.
- Повна база даних SQLite зберігається в одному кросплатформному дисковому файлі.
- SQLite дуже мала і легка, менше ніж 400 КБ повністю налаштований або менше 250 КБ без додаткових функцій.
- Є автономною, що означає відсутність зовнішніх залежностей.
- Транзакції повністю сумісні з ACID(Atomicity Consistency Isolation Durability), забезпечуючи безпечний доступ з кількох процесів або потоків.
- Підтримує більшість функцій мови запитів, які є в стандарті SQL92 (SQL2).
- Написаний на ANSI-C і надає простий у використанні API.
- Доступний на UNIX (Linux, Mac OS-X, Android, iOS) і Windows (Win32, WinCE, WinRT).

Для виконання дипломної роботи для системи керування базами даних була обрана SQLite, так як проект не надто масштабний. SQLite забезпечить простий і легкий доступ до даних, а також вона займатиме досить мало місця в пам'яті.

					ІАЛЦ.467200.003 ПЗ	Арк.
						39
Зм.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВОК ДО РОЗДІЛУ 2

У другому розділі були детально описані технології, які можна використати для розробки дипломної програми. Також було вибрано і обґрунтовано, які саме технології будуть використовуватись.

Спочатку були описані різні мови, які підходять для рішення написання програми. Було розглянуто переваги та недоліки кожної з мов, а також загальна інформація про кожну з мов та приклад коду.

Потім переглянувши усі переваги та недоліки було вибрано мову програмування, яка найкраще підійде для написання даної програми. Було розглянуто фреймворк обраної мови програмування. Розглянуто основні переваги та особливості даного фреймворку.

Також було розглянуто шаблон за допомогою якого буде розроблятися дана програма. Був наведений рисунок функціонування даного шаблону, а також був розглянутий принцип роботи даного шаблону.

У другій частині були розглянуті основні системи керування базами даних. Було описано кожну з наданих систем керування баз даних. Переглянуто їх переваги над іншими.

Проаналізувавши системи керування базами даних було обрано, яку саме будемо використовувати для розробки програми. Також було обґрунтування вибраної системи.

					ІАЛЦ.467200.003 ПЗ	Арк.
						40
Зм.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3. ДЕТАЛІ РОЗРОБКИ ПРОГРАМИ

Відповідно до описаних матеріалів вище (розділ 2) перейдемо до фази розробки програмного продукту. Для того, щоб створити гнучку програму, яка динамічно буде реагувати на дії користувача - розробку поділімо на дві частини. Перша це Frontend, друга Backend. Оскільки перша частина візуальна – більше уваги зосередимо на другій, адже в першу чергу це логіка самої програми, її функціональність.

3.1 Налаштування проекту.

Після створення початкового проекту(studybud) і додатку(base) налаштуємо setting.py для подальшої роботи з нашим проектом.

Створимо всі папки і файли які необхідні нам для роботи в подальшому. Файлова система проекту матиме зображена на рисунку 3.1:

					ІАЛЦ.467200.003 ПЗ	Арк.
						41
Зм.	Арк.	№ докум.	Підпис	Дата		

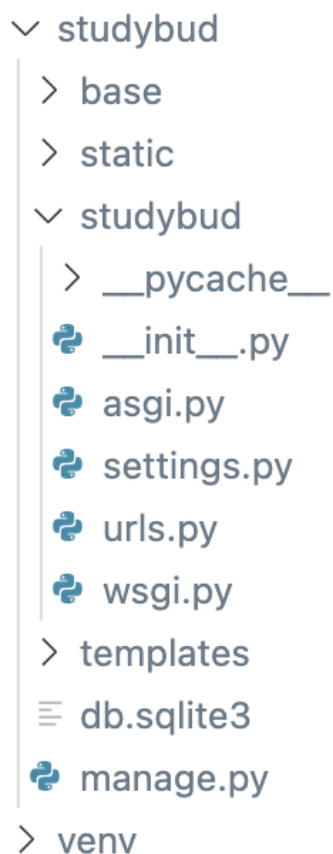


Рисунок 3.1 – Файловая система проекта

В папці `base` розташований додаток. В нього входять файли, які відповідають за логіку і поведінку проекту, деякі шаблони і файли, які відповідають за певний зовнішній вигляд. Вміст папки розглянемо детально:

```

  ✓ base
    > __pycache__
    > migrations
    > templates/base
    + __init__.py
    + admin.py
    + apps.py
    + forms.py
    + models.py
    + tests.py
    + urls.py
    + views.py
    > static

```

Рисунок 3.2 – Вміст папки base

Розглянемо за що відповідають файли даної папки:

- migrations – папка з міграціями бази даних;
- templates – шаблони зовнішнього вигляду певних сторінок;
- __init__.py – вказує, що дана папка являється пакетом;
- admin.py – відповідає за панель адміністратора;
- apps.py – налаштування додатку;
- forms.py – розширення заповнення форм даних
- models.py – моделі бази даних;
- test.py – тести в додатку;
- urls.py – відповідає за посилки в проєкті;
- views.py – логіка нашого додатку.

В папці static знаходяться статичні файли – це файли з розширенням .css, .js, а також зображення та іконки нашого проєкту.

В папці studybud знаходяться елементи, які відповідають за налаштування нашого проекту. А саме:

- settings.py – налаштування проекту;
- urls.py – посилки проекту;
- wsgi.py - взаємодії між нашим проектом в синхронному режимі, який виконується на стороні сервера, і самим вебсервером;
- asgi.py – взаємодії між нашим проектом в асинхронному режимі, який виконується на стороні сервера, і самим вебсервером;
- папка template – основні шаблони зовнішнього вигляду:
 - index.html – початкова сторінка;
 - navbar.html – навігаційна панель.
- db.sqlite3 – база даних проекту.
- manage.py – файл для управління проектом(запуск сервера, міграції баз даних).
- venv – папка з віртуальним середовищем.

Щоб Django бачив наш додаток – добавимо його в змінну INSTALLED_APP. Також добавимо шлях до папки з шаблонами, а також шлях до папки з статичними шляхами.

Тепер будемо розглядати розробку по частинам.

3.2 Створення моделей в базі даних та їх логіки

В нашому проекті будуть: тема, кімнати, повідомлення та користувачі. Django автоматично створює таблиці в базі даних для користувачів. Тому доведеться створити моделі для кімнати та повідомлень.

					ІАЛЦ.467200.003 ПЗ	Арк.
						44
Зм.	Арк.	№ докум.	Підпис	Дата		

```
class Topic(models.Model):
    name = models.CharField(max_length=200)

    def __str__(self) -> str:
        return self.name
```

Рисунок 3.3 – Клас, який відповідає за модель тем

```
class Room(models.Model):
    host = models.ForeignKey(User, on_delete=models.SET_NULL, null=True)
    topic = models.ForeignKey(Topic, on_delete=models.SET_NULL, null=True)
    name = models.CharField(max_length=200)
    description = models.TextField(null=True, blank=True)
    participants = models.ManyToManyField(User, related_name='participants', blank=True)
    updated = models.DateTimeField(auto_now=True)
    created = models.DateTimeField(auto_now_add=True)

    class Meta:
        ordering = ['-updated', '-created']

    def __str__(self):
        return str(self.name)
```

Рисунок 3.4 – Клас, який відповідає за модель кімнат

Тут ми можемо бачити, що для кожної кімнати буде визначатися власник(той, хто її створив), тема, назва, описання, учасники, останнього разу оновлено та дата створення.

Клас Meta відповідає за те, щоб кімнати сортувалися по останньому оновленню, а не даті створення.

Аргумент topic – наслідується від класу Topic(рисунок 3.3)

Аргумент participants – має взаємодію ManyToMany для того, щоб різні учасники могли приєднуватися до різних кімнат.

```

class Message(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    room = models.ForeignKey(Room, on_delete=models.CASCADE)
    body = models.TextField()
    updated = models.DateTimeField(auto_now=True)
    created = models.DateTimeField(auto_now_add=True)

    class Meta:
        ordering = ['-updated', '-created']

    def __str__(self):
        return self.body[:50]

```

Рисунок 3.4 – Клас, який відповідає за модель повідомлень

Цей клас відповідає за створення таблиць в базі даних, які будуть зберігати повідомлення. Давайте розглянемо, яка сама інформація буде зберігатися. Кожне повідомлення мати наступну інформацію: користувач, який написав, кімната, в якій було написано, текст повідомлення, останнє оновлення та створення повідомлення.

Також Django автоматично створить моделі для користувачів, сюди будуть входить логін та пароль користувача. Також буде зберігатись інформація його дій та дата входження. З моделями закінчимо.

Кожного разу після створення нової моделі слід проводити міграції даних моделей до баз даних. Приклад роботи і команди зображені на рисунку 3.5.

```

(venv) Vladislavs-MBP:studybud vladislav$ python3 manage.py makemigrations
Migrations for 'base':
  base/migrations/0005_alter_message_options.py
  - Change Meta options on message
(venv) Vladislavs-MBP:studybud vladislav$ python3 manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, base, contenttypes, sessions
Running migrations:
  Applying base.0005_alter_message_options... OK

```

Рисунок 3.5 – Міграції даних

3.3 Створення компонентів програми

3.3.1. Початкова сторінка

Спочатку створимо голосну сторінку нашого сайту для подальшої роботи з ним.

Для цього візуально представимо її, що там повинно бути і почнемо її розробляти поступово. Зверху у нас буде навігаційна панель, за допомогою якої ми будемо потрапляти на початкову сторінку, а також міститиме інформацію про користувача, якщо ж користувач не увійшов в свій обліковий запис допоможе зайти, або ж, якщо користувач авторизувався – вийти з кабінету. Зліва будуть присутні теми, при нажаті на них, зможемо перейти на ту чи іншу тему. По центру будуть відсортовані по даті створення кімнати(будуть присутні ім'я кімнат, користувачі, які їх створили, а також коли саме була створена кімната). Маючи візуальне представлення кімнати візьмемося до її програмування.

Спершу створим логіку нашої початкової сторінки в views.py. Для цього створим функцію home.

```
def home(request):

    q = request.GET.get('q') if request.GET.get('q') != None else ''
    rooms = Room.objects.filter(Q(topic__name__icontains=q) |
                                Q(name__icontains=q) |
                                Q(description__icontains=q)
                                )

    topics = Topic.objects.all()
    room_count = rooms.count()
    room_messages = Message.objects.filter(
        Q(room__topic__name__icontains=q)
    )

    context = { 'rooms': rooms, 'topics': topics, 'room_count': room_count, 'room_messages': room_messages }
    return render(request, 'base/home.html', context)
```

Рисунок 3.6 – Функція, яка відповідає за логіку початкової сторінки

					ІАЛЦ.467200.003 ПЗ	Арк.
						47
Зм.	Арк.	№ докум.	Підпис	Дата		

Ця функція в змінних зберігає всі кімнати(їх назви, теми, описи), теми, кількість кімнат, а також повідомлення, які написані в різних кімнатах.

Змінна context слугує для передавання інформації в наші html файли, для того, щоб наша програма могла працювати динамічно і змінювати свої значення при різних діях. Після створення логіки створимо візуальне представлення. Воно буде зберігатися в home.html та activity_component.html. Самі файли досить обширний. Тому звернемо увагу на ключові рядки файлів.

{% if request.user == message.user %} – перевірка, чи саме цей користувач, який зараз авторизований в системі створив повідомлення чи ні. Другими словами, повідомлення, яке написав користувач ніхто не спроможний видалити чи редагувати крім цього користувача. Також якщо користувач не авторизований – він також не зможе робити ніяких маніпуляцій з повідомленням. Така ж сама логіка описана в кімнатах.

Також кожна назва користувача є динамічною – при нажаті на неї ми будемо переходити на профіль користувача. Це дозволено за допомогою посилання на профіль, яке має вигляд - {% url 'user-profile' room.host.id %}.

3.1.3 Кімната, вигляд, створення, редагування

Створимо функції, які відповідатимуть за логіку кімнат, а також повідомлень.

					ІАЛЦ.467200.003 ПЗ	Арк.
						48
Зм.	Арк.	№ докум.	Підпис	Дата		

```

def room(request, pk):
    room = Room.objects.get(id=pk)

    room_messages = room.message_set.all().order_by('-created')
    participants = room.participants.all()
    if request.method == 'POST':
        Message.objects.create(
            user = request.user,
            room=room,
            body=request.POST.get('body')
        )
        room.participants.add(request.user)
    return redirect('room', pk=room.id)

context = { 'room': room, 'room_messages': room_messages, 'participants': participants }
return render(request, 'base/room.html', context)

```

Рисунок 3.7. – Відображення кімнати

Дана функція зберігає в змінній всі повідомлення, які були в кімнаті написані та її учасників. Щоб стати учасником кімнати слід написати хоча б одне повідомлення в кімнаті. За допомогою метода if перевіряємо, чи створює користувач кімнату, якщо так – додаємо дану кімнату до бази даних з полями: користувач, кімната та опис. В даному прикладі і надалі за допомогою змінної contex передаємо значення змінних в функції для їх обробки для візуалізації даних.

```

@login_required(login_url='/login')
def create_room(request):
    form = RoomForm()

    topics = Topic.objects.all()
    if request.method == 'POST':
        topic_name = request.POST.get('topic')
        topic, created = Topic.objects.get_or_create(name=topic_name)
        Room.objects.create(
            host=request.user,
            topic=topic,
            name=request.POST.get('name'),
            description=request.POST.get('description')
        )
        return redirect('home')

    context = {'form': form, 'topics': topics }
    return render(request, 'base/room_form.html', context)

```

Рисунок 3.8 – Створення кімнати

Дана функція відповідає за логіку створення кімнати. Вони обернена в декоратор для того, щоб не авторизований користувач не мав права створити кімнату. Також якщо користувач не авторизований – його поверне на сторінку, де він зможе увійти в свій обліковий запис, або створити новий акаунт. При створенні кімнати за допомогою `redirect('home')` користувача поверне на початку сторінку. Також у базі даних буде створена відповідна кімната з вказаними користувачем назвою, описом, темою.

```

@login_required(login_url='/login')
def update_room(request, pk):
    room = Room.objects.get(id=pk)
    form = RoomForm(instance=room)
    topics = Topic.objects.all()

    if request.user != room.host:
        return HttpResponse("Your are not allowed here!")

    if request.method == 'POST':
        topic_name = request.POST.get('topic')
        topic, created = Topic.objects.get_or_create(name=topic_name)
        room.name = request.POST.get('name')
        room.topic = topic
        room.description = request.POST.get('description')
        room.save()

        return redirect('home')

    context = {'form': form, 'topics': topics, 'room': room }
    return render(request, 'base/room_form.html', context)

```

Рисунок 3.9 – Редагування інформації кімнати

За допомогою цього представлення ми можемо змінити любі дані кімнати, навіть, її назву. До таблиці запишуться нові дані, стерши старі. При цьому, якщо авторизований користувач не є власником кімнати – його спливе повідомлення у якому говоритиметься, що він не має права оновити цю кімнату.

```

@login_required(login_url='/login')
def delete_room(request, pk):
    room = Room.objects.get(id=pk)

    if request.user != room.host:
        return HttpResponse("Your are not allowed here!")

    if request.method == 'POST':
        room.delete()
        return redirect('home')

    context = {'obj': room}
    return render(request, 'base/delete.html', context)

```

Рисунок 3.10 – Видалення кімнати

В функції вище описані дії програми при видаленні кімнати, при цьому всі дані з бази даних будуть стерти. Знову ж, якщо в користувача немає прав на видалення кімнати – він це зробити не зможе.

3.3.3 Авторизація

Створимо логіку яка буде відповідати за авторизацію. Для цього створимо функцію, яка відповідатиме за авторизацію користувач.

```
def login_page(request):

    page = 'login'

    if request.user.is_authenticated:
        return redirect('home')

    if request.method == "POST":
        username = request.POST.get('username').lower()
        password = request.POST.get('password')

        try:
            user = User.objects.get(username=username)
        except:
            messages.error(request, 'User does not exist.')

        user = authenticate(request, username=username, password=password)

        if user is not None:
            login(request, user)
            return redirect('home')
        else:
            messages.error(request, 'Username or password does not correct.')

    context = {'page': page}
    return render(request, 'base/login_register.html', context)
```

Рисунок 3.11 – Авторизації користувача

Дана функція перевіряє чи існує в базі даних користувач з таким іменем та паролем. Якщо такого користувача з таким іменем не існує, впливе відповідне вікно. Також, якщо користувач не заповнить форму вводу імені –

програма не стане перевіряти користувача. Лише при умові введення даних в відповідні поля і натисканні кнопки «Submit» програма перевірить дані і якщо вони вірні, увійде в аккаунт користувача.

Також, якщо користувач не реєструвався ще на сайті, передбачена сторінка реєстрації. Її логіка зображена на рисунку нижче:

```
def register_page(request):
    form = UserCreationForm()

    if request.method == 'POST':
        form = UserCreationForm(request.POST)
        if form.is_valid():
            user = form.save(commit=False)

            user.username = user.username.lower()
            user.save()
            login(request, user)
            return redirect('home')
        else:
            messages.error(request, 'An error occurred during registration')

    context = { 'form': form }
    return render(request, 'base/login_register.html', context)
```

Рисунок 3.12 –Реєстрації нового користувача

Після реєстрації дані, які увів користувач запишуться в базу даних. І він наступного разу зможе зайти під своїм іменем і паролем, який він увів раніше.

Також не можна буде створити аккаунт з простим паролем. Щоб пароль був вірний повинно бути як мінімум 8 символів, він не має бути занадто простим, містити цифри, великі і маленькі букви.

3.4 Динамічна генерація HTML

Найкращий підхід – це створити статичний і динамічний файли. Створимо статичні файли .html, які будуть використовувати динамічні файли .html. Це будуть: navbar.html, index.html. Сам index.html містить navbar.html. Це

створено для того, щоб спростити розуміння коду і для того, щоб легше було змінювати код при необхідності. Якщо, наприклад, доведеться змінити виключно навігаційну панель, чи панель активності – не доведеться шукати це все в одному файлі, а досить буде знайти виключно файл, який за це відповідає – і змінювати його, не боячись за те, що змінимо щось не те.

```
<!DOCTYPE html>
{% load static %}
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link rel="shortcut icon" href="assets/favicon.ico" type="image/x-icon" />
    <link rel="stylesheet" href="{% static 'styles/style.css' %}" />
    <title>StudyBuddy – Find study partners around the world!</title>
  </head>

  <body>
    {% include 'navbar.html' %}

    {% if messages %}
    <ul class="messages">
      {% for message in messages %}
        <li>
          {{ message }}
        </li>
      {% endfor %}
    </ul>
    {% endif %}

    {% block content %}

    {% endblock content %}
    <script src="{% static 'js/script.js' %}"></script>
  </body>
</html>
```

Рисунок 3.13 – Код index.html

```

{% load static %}

<header class="header header--loggedIn">
  <div class="container">
    <a href="{% url 'home' %}" class="header__logo">...
    </a>
    <form class="header__search" method="GET" action="{% url 'home' %}">...
    </form>
    <nav class="header__menu">
      <!-- Not Logged In -->
      <!-- Logged In -->
      {% if request.user.is_authenticated %}
        <div class="header__user">...
        </div>
      {% else %}
        <a href="{% url 'login' %}">...
        </a>
      {% endif %}
      <div class="dropdown-menu">
        <a href="settings.html" class="dropdown-link" ...
        >
        <a href="{% url 'logout' %}" class="dropdown-link" ...
        >
      </div>
    </nav>
  </div>
</header>

```

Рисунок 3.14 – Код navbar.html

При цьому увесь код наступних html файлів буде вставлено в початковий файл index.html а саме в поле {% block content %}. Такі динамічні дані значно полегшують програмування веб-додатку, цим самим економить багато часу і сил.

ВИСНОВОК ДО РОЗДІЛУ 3

В результаті проведеної роботи щодо розробки програмного прикладного інтерфейсу для доступу до інформаційних ресурсів можна виокремити наступне:

- було реалізовано раніше описані технології веб-програмування за допомогою фреймворку Django(розділ 2);
- були продемонстровані фрагменти коду у вигляді скріншотів коду;
- описана і реалізована логіка поведінки сайту;
- динамічно створена візуальна частина;
- детально описані основні методи, атрибути, властивості класів.

					ІАЛЦ.467200.003 ПЗ	Арк.
						56
Зм.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 4. ДОСЛІДЖЕННЯ ТА АНАЛІЗ РОЗРОБЛЕНОЇ ПРОГРАМИ

Базуючись на розробленому програмному прикладному інтерфейсі для доступу до інформаційних ресурсів метою даного розділу є показати результати її роботи.

4.1 Запуск сервера

Після розробки веб-програми запусимо наш сервер за допомогою команди: `python3 manage.py runserver`.

Перед тим, як запусити – слід активувати наше віртуальне середовище, в якому зберігаються всі необхідні бібліотеки для запуску нашого сервера. Після команди в терміналі можна найти посилання, за яким ми можемо перейти на наш сервер.

```
(venv) Vladislavs-MBP:studybud vladislav$ python3 manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
June 11, 2022 - 20:14:48
Django version 4.0.5, using settings 'studybud.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

Рисунок 4.1 – Результати команди `python3 manage.py runserver`.

4.2 Інтерфейс веб-додатку

4.2.1 Початкова сторінка

					ІАЛЦ.467200.003 ПЗ	Арк.
						57
Зм.	Арк.	№ докум.	Підпис	Дата		

Перейшовши за посиланням побачимо наступне:

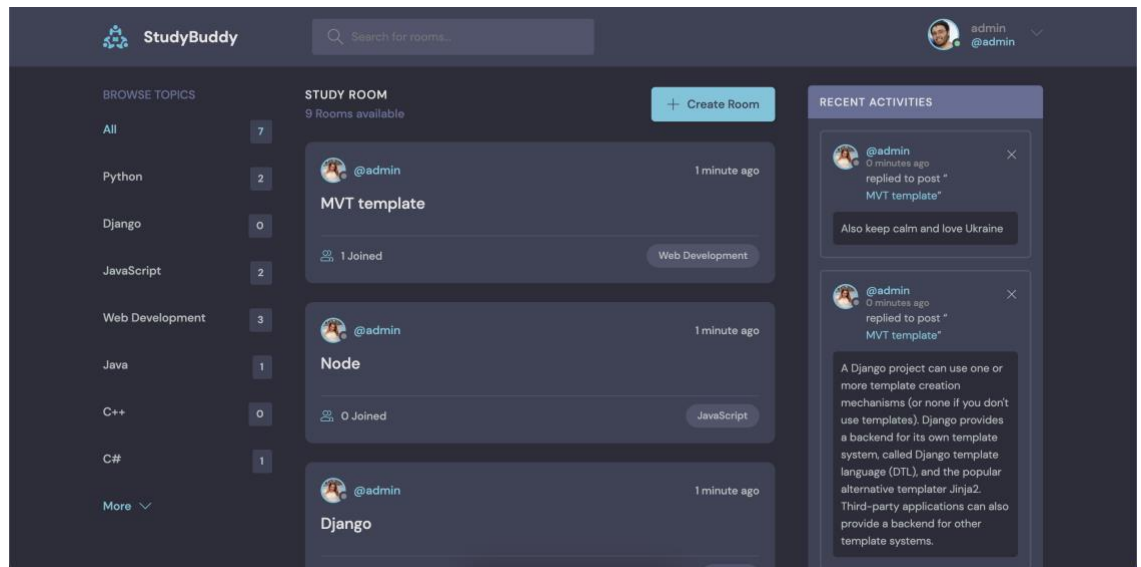


Рисунок 4.2 – Початкова сторінка

У нашій базі даних уже створено декілька кімнат, користувачів та тем для кращої демонстрації веб-програми. Розглянемо навігаційну панель. Зліва назва програми та її іконка. Назва завжди слугує силкою на нашу головну сторінку. По центру ми бачимо пошукову панель. За допомогою неї можна шукати потрібні кімнати чи теми. Зліва ми побачимо аватар користувача, його ім'я. Нажавши на нього вилізе невелике меню за допомогою якого ми можемо вийти з нашого облікового запису. Якщо користувач не авторизований ця частина матиме наступний вигляд:

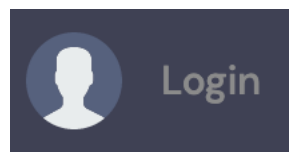


Рисунок 4.3 – Вигляд неавторизованого користувача

Зліва ми можемо побачити теми нашої веб програми. Там також відображається кількість створених кімнат на ту чи іншу тему. Теми слугують силками на всі кімнати, створені по цій темі.

По центру присутня кнопка «Create Room», за допомогою неї ми можемо створити нову кімнату. Також по центру ми бачимо список кімнат, а саме: хто створив, коли створили, тему кімнати, назву, скільки людей приєдналося.

Справа бачимо панель активності. В ній відображаються останні повідомлення, які були написані користувачами. Якщо ви власник повідомлення – у вас ще будем видний хрестик біля кожного вашого повідомлення. З-за допомогою нього ви можете запросто видалити ваше повідомлення не переходячи в кімнату.

4.2.2 Створення кімнати

Інтерфейс створення нової кімнати матиме наступний вигляд:

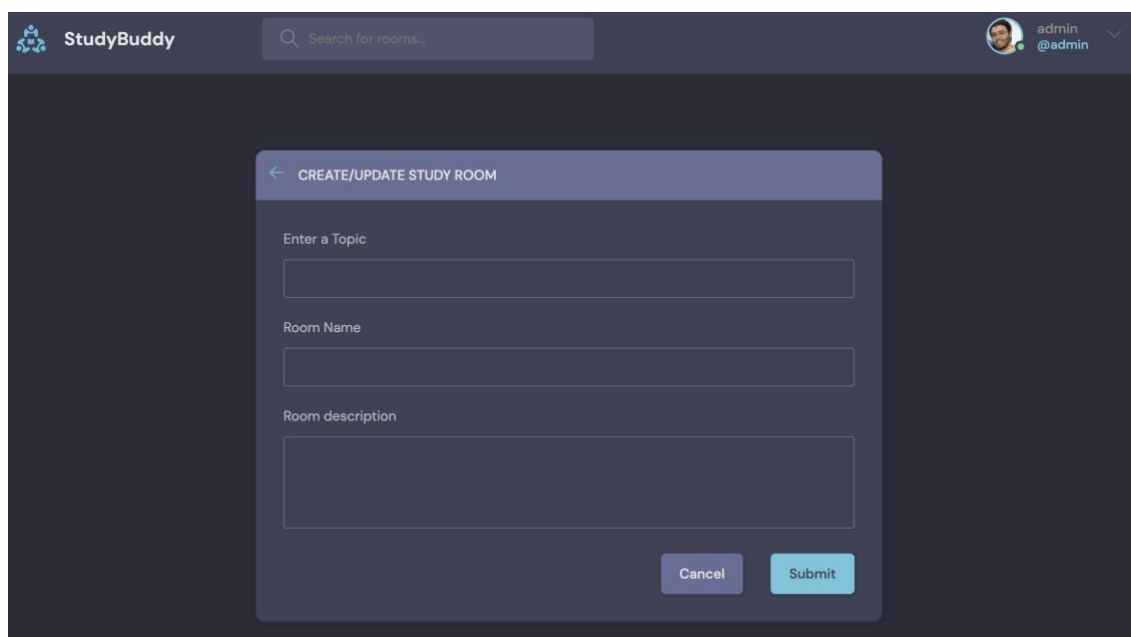


Рисунок 4.4 – Вигляд сторінки для створення кімнати

Таку сторінку можемо побачити, коли будемо створювати кімнату. Тут ми можемо написати тему, назву та опис нашої кімнати. При цьому тему можна як вибрати так і написати вручну:

Рисунок 4.5 – Вибір теми кімнати

Поля мають бути обов’язково всі заповнені. Якщо поля не заповнено – програма вам про це повідомить:

Рисунок 4.6 – Сповіщення про помилку.

4.2.3 Профіль користувача

Нажавши на будь якого користувача – ми перейдемо в його профіль. Він має наступний вигляд:

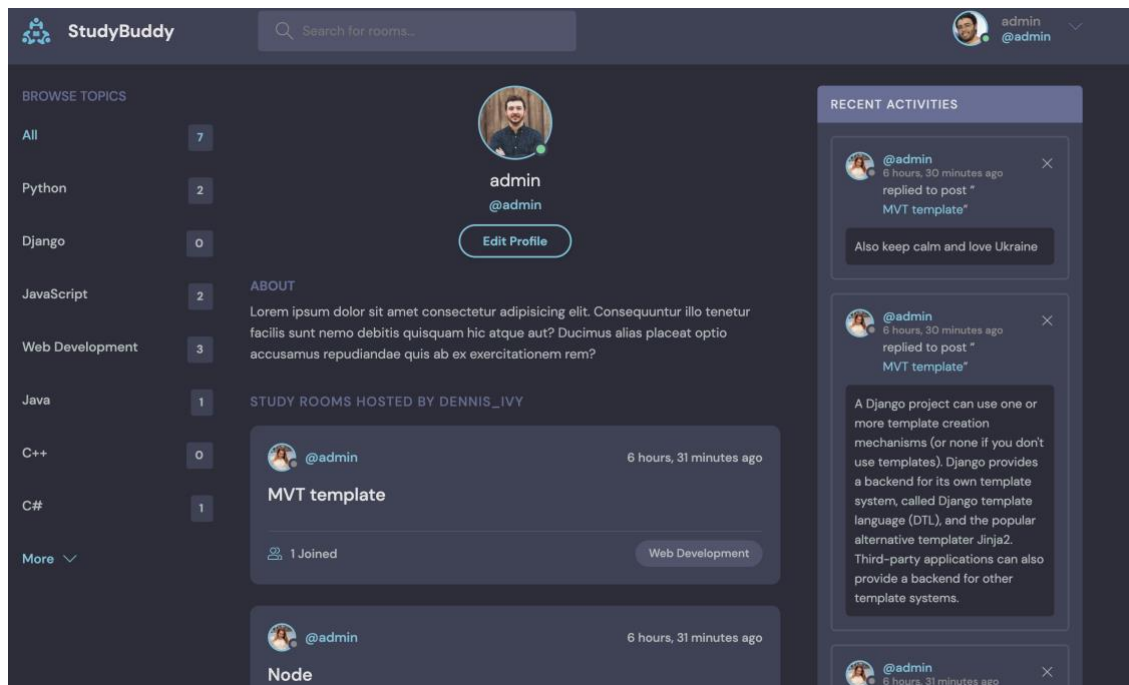


Рисунок 4.7 – Профіль користувача

Тут ми бачимо його аватар, ім'я, якщо це ваш аккаунт, ви можете його редагувати. Далі йде опис користувача. Нижче відображаються всі кімнати, які він створив. А справа, в меню активності відображаються повідомлення лише цього користувача, відсортовані по даті написанні.

4.2.4 Кімната

Нажавши на будь-яку кімнату – перейдемо в неї:

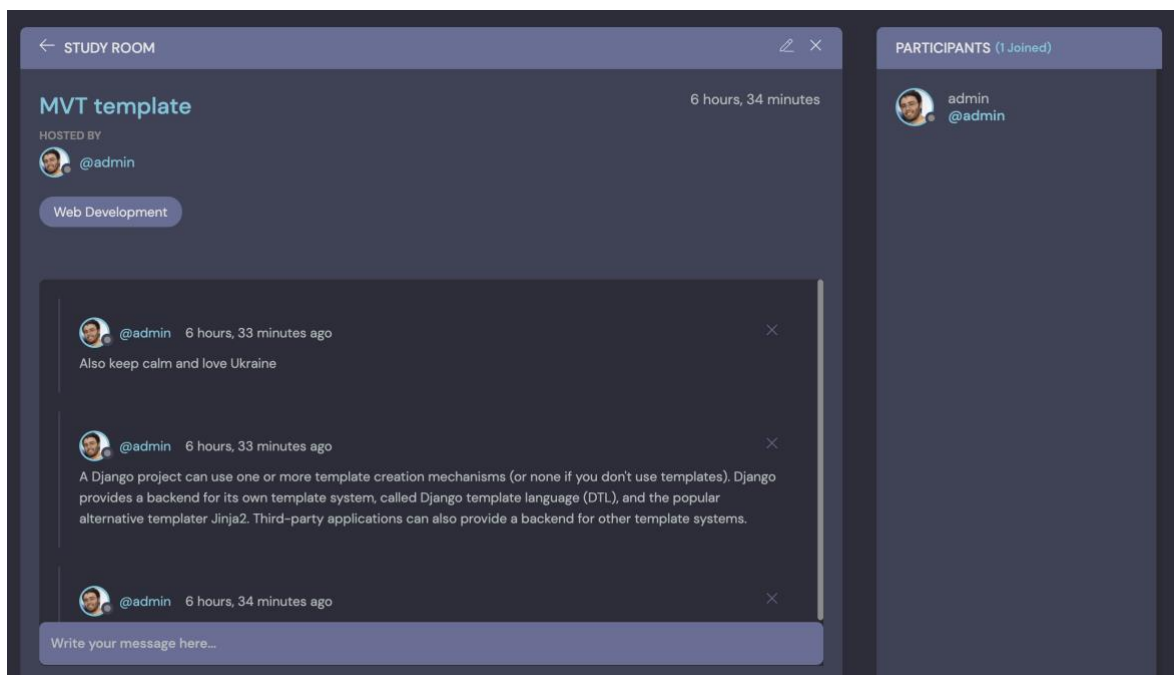


Рисунок 4.8 – Інтерфейс кімнати

Тут ми бачимо назву кімнати, ким була створена, коли саме та тему. Нижче ми можемо бачити повідомлення, які писалися в цю кімнату. Якщо це ваше повідомлення – ви маєте права, щоб його видалити. Якщо я вийду з облікового запису – ця можливість пропаде:

					ІАЛЦ.467200.003 ПЗ	Арк.
						62
Зм.	Арк.	№ докум.	Підпис	Дата		

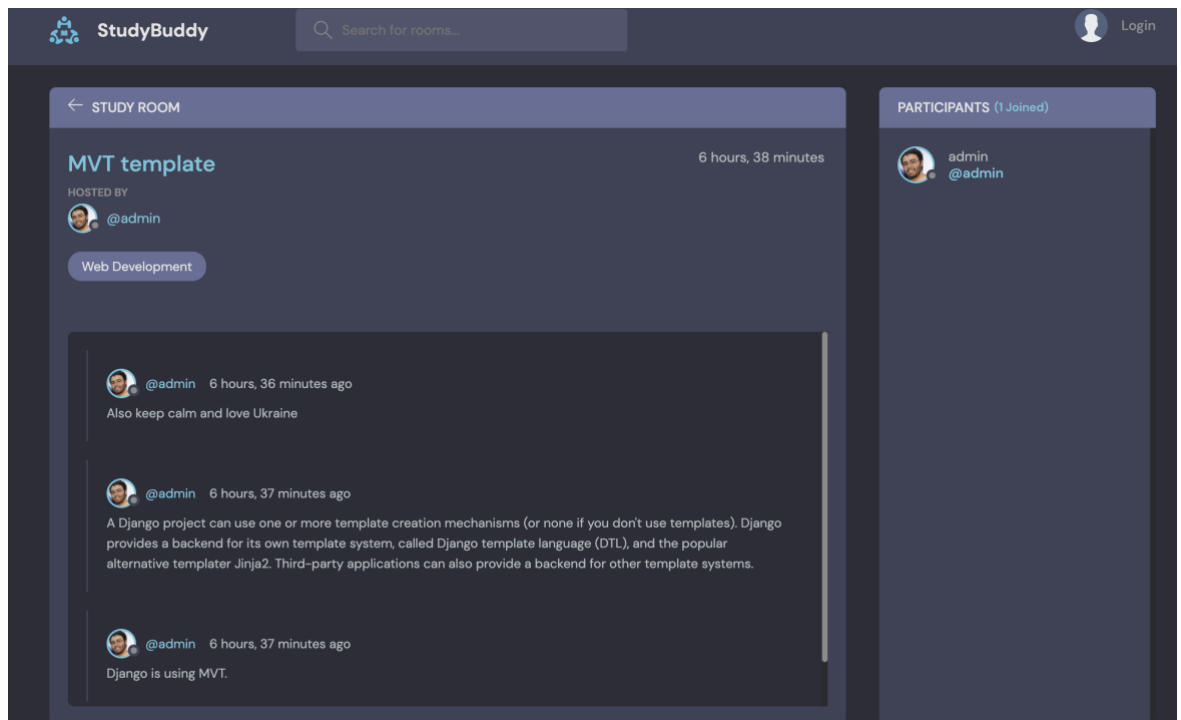


Рисунок 4.9 – Вигляд кімнати для неавторизованого користувача

Як бачимо, перегляд повідомлень дозволяється, проте можливості написати повідомлення немає. Також відсутнє меню редагування кімнати.



Рисунок 4.10 – Можливість редагування кімнати

Тут ми можемо як змінити кімнату, так і видалити її. При натисканні на редагування побачимо результат, який зображено на рисунку 4.11.

					ІАЛЦ.467200.003 ПЗ	Арк.
						63
Зм.	Арк.	№ докум.	Підпис	Дата		

Рисунок 4.11 – Редагування кімнати

При цьому старі дані відображаються автоматично, заново вводити не треба. Після редагування оновлені дані запишуться до баз даних. Допишемо до назви кімнати слово “test” і підтвердим зміни. Побачимо наступні зміни:

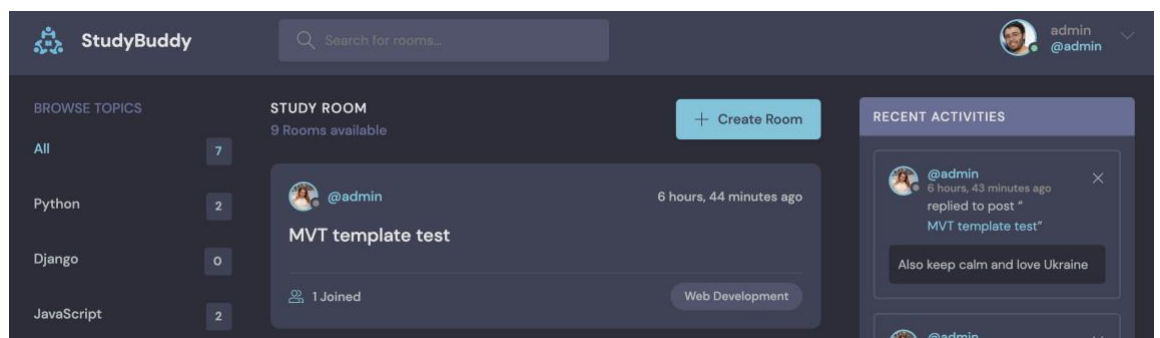


Рисунок 4.12 – Зміни в вигляді кімнати, після її редагування

4.2.5 Авторизація

Для того, щоб увійти в акаунт достатньо клікнути по іконці зображеній на рисунку 4.2. Ми перейдемо на сторінку авторизації, яка має наступний вигляд:



Рисунок 4.13 – Вигляд інтерфейсу реєстрації

Якщо ви ще не зареєстровані – є можливість створити новий акаунт. Для цього клікнем на «Sign up» і перейдемо на сторінку реєстрації, яка має наступний вигляд:

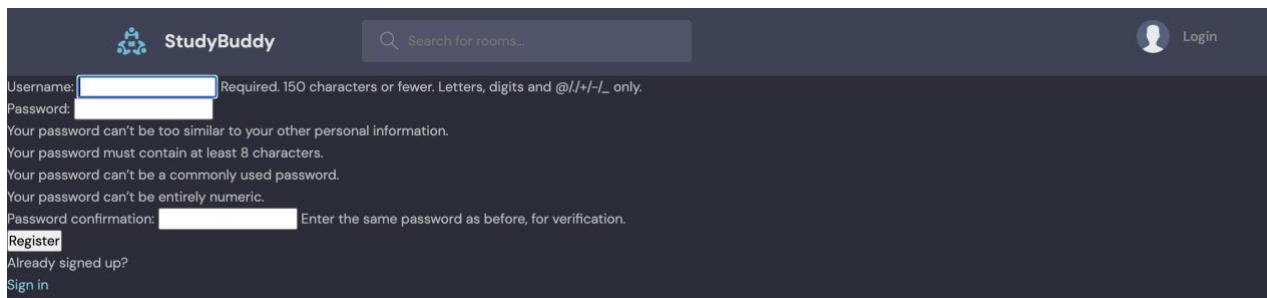


Рисунок 4.14 – Вигляд інтерфейсу реєстрації

4.3. Відповідь сервера

Щоб закінчити роботу сервера достатньо перейти в термінал і нажати відповідні клавіші. При цьому в терміналі ми можемо побачити нашу сесію і які протоколи виконувалися. Це має наступний вигляд:

					ІАЛЦ.467200.003 ПЗ	Арк.
						65
Зм.	Арк.	№ докум.	Підпис	Дата		

```

[11/Jun/2022 20:15:28] "GET / HTTP/1.1" 200 26205
[11/Jun/2022 20:15:29] "GET /static/images/logo.svg HTTP/1.1" 304 0
[11/Jun/2022 20:15:29] "GET /static/js/script.js HTTP/1.1" 304 0
[11/Jun/2022 20:15:29] "GET /static/styles/style.css HTTP/1.1" 304 0
Not Found: /assets/favicon.ico
[11/Jun/2022 20:15:29] "GET /assets/favicon.ico HTTP/1.1" 404 3940
[11/Jun/2022 20:18:32] "GET /create-room/ HTTP/1.1" 200 7281
Not Found: /create-room/assets/favicon.ico
[11/Jun/2022 20:18:32] "GET /create-room/assets/favicon.ico HTTP/1.1" 404 3976
[11/Jun/2022 20:18:46] "POST /create-room/ HTTP/1.1" 302 0
[11/Jun/2022 20:18:46] "GET / HTTP/1.1" 200 28228
[11/Jun/2022 20:18:50] "GET /create-room/ HTTP/1.1" 200 7281
[11/Jun/2022 20:18:54] "POST /create-room/ HTTP/1.1" 302 0
[11/Jun/2022 20:18:54] "GET / HTTP/1.1" 200 30252
[11/Jun/2022 20:18:57] "GET /create-room/ HTTP/1.1" 200 7281
[11/Jun/2022 20:19:16] "POST /create-room/ HTTP/1.1" 302 0
[11/Jun/2022 20:19:16] "GET / HTTP/1.1" 200 32291
[11/Jun/2022 20:19:20] "GET /delete-message/14/ HTTP/1.1" 200 4636
Not Found: /delete-message/14/assets/favicon.ico
[11/Jun/2022 20:19:20] "GET /delete-message/14/assets/favicon.ico HTTP/1.1" 404 3994
[11/Jun/2022 20:19:22] "POST /delete-message/14/ HTTP/1.1" 302 0
[11/Jun/2022 20:19:23] "GET / HTTP/1.1" 200 31126
[11/Jun/2022 20:19:24] "GET /delete-message/13/ HTTP/1.1" 200 4631
Not Found: /delete-message/13/assets/favicon.ico
[11/Jun/2022 20:19:24] "GET /delete-message/13/assets/favicon.ico HTTP/1.1" 404 3994
[11/Jun/2022 20:19:26] "POST /delete-message/13/ HTTP/1.1" 302 0
[11/Jun/2022 20:19:26] "GET / HTTP/1.1" 200 29966
[11/Jun/2022 20:19:27] "GET /delete-message/11/ HTTP/1.1" 200 4633
Not Found: /delete-message/11/assets/favicon.ico
[11/Jun/2022 20:19:27] "GET /delete-message/11/assets/favicon.ico HTTP/1.1" 404 3994
[11/Jun/2022 20:19:28] "POST /delete-message/11/ HTTP/1.1" 302 0
[11/Jun/2022 20:19:28] "GET / HTTP/1.1" 200 28806
[11/Jun/2022 20:19:31] "GET /room/10/ HTTP/1.1" 200 10163
Not Found: /room/10/assets/favicon.ico
[11/Jun/2022 20:19:31] "GET /room/10/assets/favicon.ico HTTP/1.1" 404 3964
[11/Jun/2022 20:19:50] "POST /room/10/ HTTP/1.1" 302 0
[11/Jun/2022 20:19:50] "GET /room/10/ HTTP/1.1" 200 11869
[11/Jun/2022 20:20:21] "POST /room/10/ HTTP/1.1" 302 0
[11/Jun/2022 20:20:21] "GET /room/10/ HTTP/1.1" 200 13553
[11/Jun/2022 20:20:38] "POST /room/10/ HTTP/1.1" 302 0
[11/Jun/2022 20:20:38] "GET /room/10/ HTTP/1.1" 200 14941
[11/Jun/2022 20:20:41] "GET / HTTP/1.1" 200 32610
[12/Jun/2022 02:31:40] "GET /logout/ HTTP/1.1" 302 0
[12/Jun/2022 02:31:41] "GET / HTTP/1.1" 200 29656
[12/Jun/2022 02:31:41] "GET /static/images/avatar.svg HTTP/1.1" 304 0
[12/Jun/2022 02:31:46] "GET /login/ HTTP/1.1" 200 4322
Not Found: /login/assets/favicon.ico
[12/Jun/2022 02:31:46] "GET /login/assets/favicon.ico HTTP/1.1" 404 3958
[12/Jun/2022 02:31:59] "POST /login/ HTTP/1.1" 302 0
[12/Jun/2022 02:31:59] "GET / HTTP/1.1" 200 32680
[12/Jun/2022 02:41:45] "GET /create-room/ HTTP/1.1" 200 7281
[12/Jun/2022 02:48:25] "GET / HTTP/1.1" 200 32680
Not Found: /profile.html
[12/Jun/2022 02:48:44] "GET /profile.html HTTP/1.1" 404 3922
Not Found: /favicon.ico
[12/Jun/2022 02:48:44] "GET /favicon.ico HTTP/1.1" 404 3919
[12/Jun/2022 02:48:50] "GET /profile/1/ HTTP/1.1" 200 29373

```

Рисунок 4.15 – Відповідь сервера

					ІАЛЦ.467200.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		66

ВИСНОВОК ДО РОЗДІЛУ 4

У четвертому розділі представлена демонстрація роботи програми розроблюваної під час всієї дипломної роботи. Надано основні вказівки щодо запуску і застосування веб-програми.

Були продемонстровані основні властивості веб-програми, її особливості. На прикладі скріншотів побачили можливості веб-програми.

Під час роботи з веб-програмою ніяких помилок та перебоїв не було, веб-програма працює як і було задумано.

					ІАЛЦ.467200.003 ПЗ	Арк.
						67
Зм.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

Під час виконання дипломної роботи було створено програмний прикладний інтерфейс для доступу до інформаційних ресурсів. Дана програма надає можливість своїм користувачам за допомогою веб-застосунку знаходити потрібну їм інформацію.

У першому розділі була розглянута область використання та аналоги серед програм і не тільки для рішення проблеми. Також у першому було порівняно аналоги, були розглянуті їхні загальні характеристики. Також з аналогів виокремили їх переваги, щоб потім реалізувати у дипломній програмі.

У другому розділі були детально описані мови програмування на яких можна було б написати дипломну веб-програму, а також розглянуто переваги та недоліки кожної з мов. Провівши аналіз, зробили вибір мови та обґрунтували цей вибір. Було розглянуто основні системи керування базами даних для збереження потрібної інформації веб-додатку. Розглянуто основні переваги цих систем. Після розгляду вибрано і обґрунтовано систему керування баз даними. Також у другій частині цього розділу обговорюється фреймворк та шаблон, які будуть використані для написання вихідного коду.

У третьому розділі було реалізовано раніше обґрунтовані (у Розділі II) фреймворки, бази даних та шаблони та їх функціонування. Були продемонстровані фрагменти коду у вигляді скріншотів розроблюваної веб-програми.

У четвертому розділі ми провели детальний аналіз розробленої веб-програми. У розділі представлений готовий веб-додаток, розроблюваний протягом всієї дипломної роботи. Надано загальні вказівки щодо використання застосунка, а також було розглянуто випадки неправильного користування і до яких ситуацій це може призвести.

Слід також зазначити, що під час розробки застосунку були враховані всі зазначені на початку вимоги, яким він мав відповідати.

					ІАЛЦ.467200.003 ПЗ	Арк.
						68
Зм.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Python 3.10.5 documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.python.org/3/>.
2. Django The web framework for perfectionists with deadlines [Електронний ресурс] – Режим доступу до ресурсу: <https://www.djangoproject.com/>.
3. Django MVT [Електронний ресурс] – Режим доступу до ресурсу: <https://www.javatpoint.com/django-mvt/>.
4. HTML Element Reference [Електронний ресурс] – Режим доступу до ресурсу: <https://www.w3schools.com/TAGs/default.asp/>.
5. Адаптація - [Електронний ресурс] – Режим доступу до ресурсу: <https://uk.wikipedia.org/wiki/%D0%90%D0%B4%D0%B0%D0%BF%D1%82%D0%B0%D1%86%D1%96%D1%8F>.
6. Основы CSS - Изучение веб-разработки - [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/ru/docs/Learn/Getting_started_with_the_web/CSS_basics.
7. Адаптація у новій компанії [Електронний ресурс] – Режим доступу до ресурсу: <https://www.shyshenko.com/blog/first-working-day>.
8. Преимущества и недостатки языка Python [Електронний ресурс] – Режим доступу до ресурсу: <https://blog.ithillel.ua/articles/perevagi-i-nedoliki-movi-python>.
9. Современный учебник JavaScript [Електронний ресурс] – Режим доступу до ресурсу: <https://learn.javascript.ru/>.
10. Rust [Електронний ресурс] – Режим доступу до ресурсу: <https://en.wikipedia.org/wiki/Rust>.

					ІАЛЦ.467200.003 ПЗ	Арк.
						69
Зм.	Арк.	№ докум.	Підпис	Дата		

11. What is DataBase [Електронний ресурс] – Режим доступу до ресурсу:
<https://www.techtarget.com/searchdatamanagement/definition/database#:~:text=A%20database%20is%20information%20that,data%2C%20financials%20and%20product%20information.> .
12. MySQL [Електронний ресурс] – Режим доступу до ресурсу:
<https://www.mysql.com/t.com/>.
13. sqlite3 — DB-API 2.0 interface for SQLite [Електронний ресурс] – Режим доступу до ресурсу:
<https://docs.python.org/3/library/sqlite3.html#:~:text=Source%20code%3A%20Lib%2Fsqlite3%2F,SQLite%20for%20internal%20data%20storage> .
14. SQLite – Overview [Електронний ресурс] – Режим доступу до ресурсу:
https://www.tutorialspoint.com/sqlite/sqlite_overview.htm .
15. ОПИСАНИЕ И ДОКУМЕНТИРОВАНИЕ ИТ-ПРОЕКТОВ [Електронний ресурс] – Режим доступу до ресурсу:
<https://itsource.com.ua/blog/opisanie-i-dokumentirovanie-it-proektov-must-have-dlya-uspeshnogo-biznesa/> .
16. Подготовка документации для IT-проектов: для разработчиков, руководителей, инвесторов [Електронний ресурс] – Режим доступу до ресурсу:
<https://internationalwealth.info/offshore-business-abroad/podgotovka-dokumentacii-dlja-it-proektov-dlja-razrabotchikov-rukovoditelej-investorov/> .
17. Mongo DB [Електронний ресурс] – Режим доступу до ресурсу:
<https://www.mongodb.com/> .
18. What is Java? Definition, Meaning & Features of Java Platforms [Електронний ресурс] – Режим доступу до ресурсу:
<https://www.guru99.com/java-platform.html> .
19. ПЕРЕВАГИ ТА НЕДОЛІКИ СИСТЕМ БАЗ ДАНИХ [Електронний ресурс] – Режим доступу до ресурсу:
https://stud.com.ua/93783/informatika/perevagi_nedoliki_sistem_danih .

- 20.Призначення баз даних. Переваги БД і їх недоліки. [Електронний ресурс] – Режим доступу до ресурсу: <https://infopedia.su/15x213e.html> .
- 21.SQLAlchemy ORM [Електронний ресурс] – Режим доступу до ресурсу: <https://lectureswww.readthedocs.io/6.www.sync/2.codding/9.databases/2.sqlalchemy/> .
- 22.Object-relational Mappers (ORMs) [Електронний ресурс] – Режим доступу до ресурсу: <https://www.fullstackpython.com/object-relational-mappers-orms.html> .
- 23.JavaScript Tutorial [Електронний ресурс] – Режим доступу до ресурсу: <https://www.w3schools.com/js/> .
- 24.From Zero to Front-end Hero [Електронний ресурс] – Режим доступу до ресурсу: <https://www.freecodecamp.org/news/from-zero-to-front-end-hero-part-1-7d4f7f0bff02/> .
- 25.Що таке робота в команді [Електронний ресурс] – Режим доступу до ресурсу: <https://blog.agrokebety.com/shcho-take-komandna-robota>.
- 26.A tour of the C# language [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: <https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/>.
- 27.PYPL PopularitY of Programming Language [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: <https://pypl.github.io/PYPL.html>.
- 28.Deep Dive Into Modern Web Development [Електронний ресурс] – Режим доступу до ресурсу: <https://fullstackopen.com/en/> .
- 29.MATT WATSON.What is C# used for? [Електронний ресурс] / MATT WATSON // STACKIFY PRODUCT & COMPANY UPDATES. – 2020. – Режим доступу до ресурсу: <https://stackify.com/what-is-c-used-for/>.
- 30.Top 10 Most Important Features Of C# [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: <https://www.c-sharpcorner.com/article/top-10-most-important-features-of-C-Sharp-programming/>.

ДОДАТОК 1

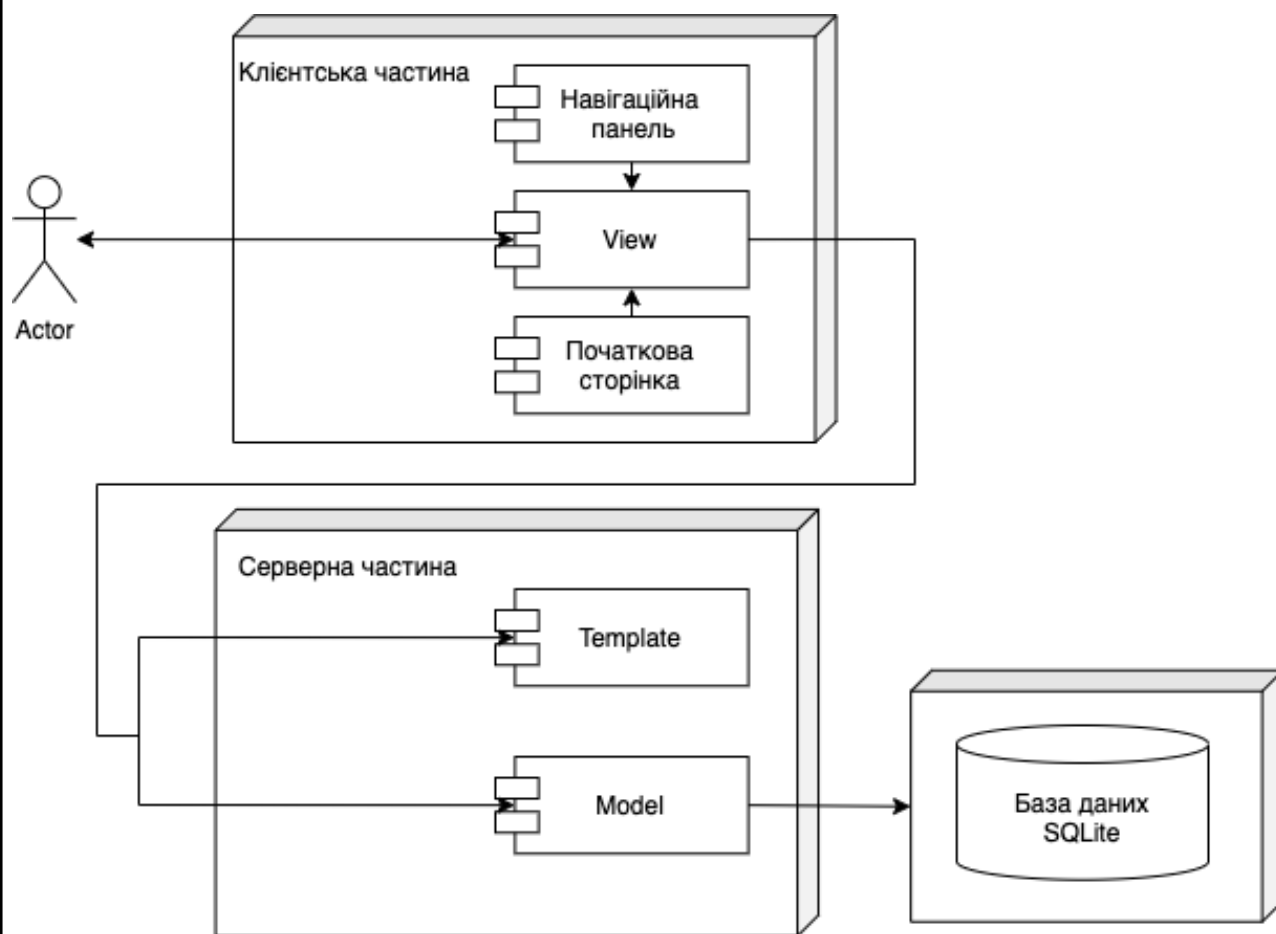
Програмний прикладний інтерфейс для доступу до
інформаційних ресурсів

Структурна схема системи

ІАЛЦ.467200.004 Д1

Аркушів 1

Київ 2022 р



					ІАЛЦ.467200.004 Д1			
		№ докум.	Підпис	Дата	Програмний прикладний інтрейс для доступу до інформаційних ресурсів Структурна схема системи	Літ.	Аркуш	Аркушів
Розробив	Дяченко В.Г.						1	1
Перевірив	Клименко І.А.							
Н. Контр.	Сімоненко В. П.					НТУУ КПІ ім. Ігоря Сікорського, ФІОТ, ІО-83		
Затвердив								

ДОДАТОК 2

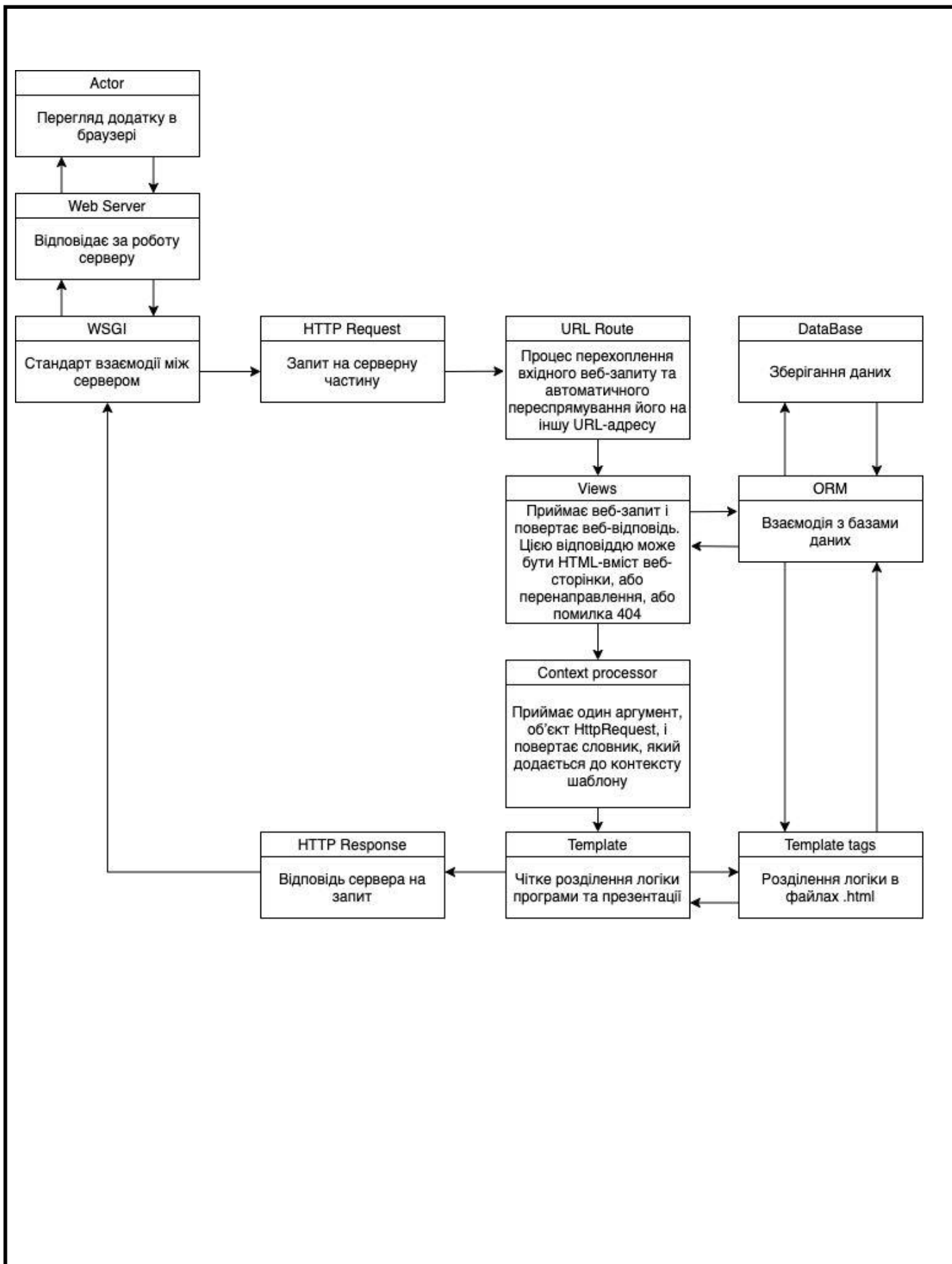
Програмний прикладний інтерфейс для доступу до інформаційних
ресурсів

Функціональна схема

ІАЛЦ.467200.005 Д2

Аркушів 1

Київ 2022 р



					ІАЛЦ.467200.005 Д2		
		№ докум.	Підпис	Дата			
Розробив	Дяченко В.Г.				Програмний прикладний інтерфейс для доступу до інформаційних ресурсів Функціональна схема (діаграма класів)		
Перевірів	Клименко І.А.						
Н. Контр.	Сімоненко В. П.						
Затвердив							
					Літ.	Аркуш	Аркушів
						1	1
					НТУУ КПІ ім. Ігоря Сікорського, ФІОТ, ІО-83		

ДОДАТОК 3

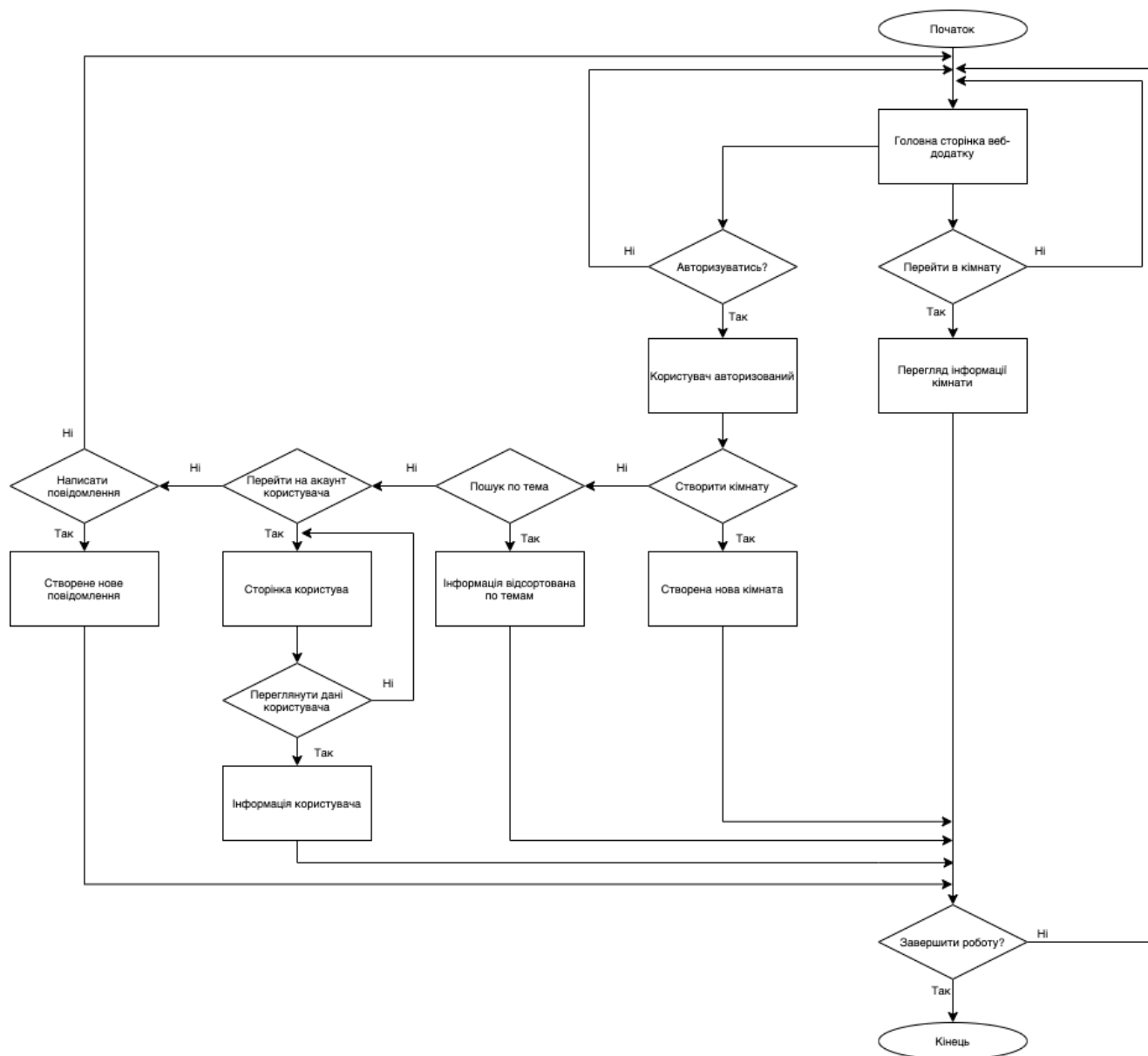
Програмний прикладний інтерфейс для доступу до інформаційних
ресурсів

Алгоритм дій програмного забезпечення

ІАЛЦ.467200.006 ДЗ

Аркушів 1

Київ 2022 р



					ІАЛЦ.467200.006 ДЗ							
		№ докум.	Підпис	Дата								
Розробив		Дяченко В.Г.			Програмний прикладний інтерфейс для доступу до інформаційних ресурсів Алгоритм дій програм- ного забезпечення			Літ.	Аркуш	Аркушів		
Перевірив		Клименко І.А.								1	1	
								НТУУ КПІ ім. Ігоря Сікорського, ФІОТ, ІО-83				
Н. Контр.		Сімоненко В. П.										
Затвердив												

ДОДАТОК 4

Програмний прикладний інтерфейс для доступу до
інформаційних ресурсів

Текст програмного коду
ІАЛЦ.467200.007 Д4

Аркушів 41

Київ 2022 р

					ІАЛЦ.467200.007 Д4	Арк.
						9
Зм.	Арк.	№ докум.	Підпис	Дата		

```

"""
Django settings for studybud project.

Generated by 'django-admin startproject' using Django 4.0.4.

For more information on this file, see
https://docs.djangoproject.com/en/4.0/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/4.0/ref/settings/
"""

from pathlib import Path

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

# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/4.0/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-0*k*ou2@2au-6&%&le#rb_^wt26im%4wc5@m@lscunh^mrm@l('

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

ALLOWED_HOSTS = []

# Application definition

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

    'base.apps.BaseConfig',
]

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

ROOT_URLCONF = 'studybud.urls'

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

```

					ІАЛЦ.467200.007 Д4	Арк.
						10
Зм.	Арк.	№ докум.	Підпис	Дата		

```

        'context_processors': [
            'django.template.context_processors.debug',
            'django.template.context_processors.request',
            'django.contrib.auth.context_processors.auth',
            'django.contrib.messages.context_processors.messages',
        ],
    },
],
WSGI_APPLICATION = 'studybud.wsgi.application'

# Database
# https://docs.djangoproject.com/en/4.0/ref/settings/#databases

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

# Password validation
# https://docs.djangoproject.com/en/4.0/ref/settings/#auth-password-validators

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

# Internationalization
# https://docs.djangoproject.com/en/4.0/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_TZ = True

# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/4.0/howto/static-files/

STATIC_URL = 'static/'

STATICFILES_DIRS = [
    BASE_DIR / 'static'
]

# Default primary key field type

```

					ІАЛЦ.467200.007 Д4	Арк.
						11
Зм.	Арк.	№ докум.	Підпис	Дата		

```
# https://docs.djangoproject.com/en/4.0/ref/settings/#default-auto-field

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'

from django.forms import ModelForm
from .models import Room

class RoomForm(ModelForm):
    class Meta:
        model = Room
        fields = '__all__'
        exclude = ['host', 'participants']

from django.db import models
from django.contrib.auth.models import User

class Topic(models.Model):
    name = models.CharField(max_length=200)

    def __str__(self) -> str:
        return self.name

class Room(models.Model):
    host = models.ForeignKey(User, on_delete=models.SET_NULL, null=True)
    topic = models.ForeignKey(Topic, on_delete=models.SET_NULL, null=True)
    name = models.CharField(max_length=200)
    description = models.TextField(null=True, blank=True)
    participants = models.ManyToManyField(User, related_name='participants', blank=True)
    updated = models.DateTimeField(auto_now=True)
    created = models.DateTimeField(auto_now_add=True)

    class Meta:
        ordering = ['-updated', '-created']

    def __str__(self):
        return str(self.name)

class Message(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    room = models.ForeignKey(Room, on_delete=models.CASCADE)
    body = models.TextField()
    updated = models.DateTimeField(auto_now=True)
    created = models.DateTimeField(auto_now_add=True)

    class Meta:
        ordering = ['-updated', '-created']

    def __str__(self):
        return self.body[:50]

from django.urls import path
from . import views

urlpatterns = [
```

Зм.	Арк.	№ докум.	Підпис	Дата

ІАЛЦ.467200.007 Д4

Арк.

12

```

path('login/', views.login_page, name='login'),
path('logout/', views.logout_user, name='logout'),
path('register/', views.register_page, name='register'),
path("", views.home, name='home'),
path('room/<str:pk>', views.room, name='room'),
path('profile/<str:pk>', views.user_profile, name='user-profile'),

path('create-room/', views.create_room, name='create-room'),
path('update-room/<str:pk>', views.update_room, name='update-room'),
path('delete-room/<str:pk>', views.delete_room, name='delete-room'),
path('delete-message/<str:pk>', views.delete_message, name='delete-message'),
]

from curses.panel import top_panel
from pydoc_data.topics import topics
from django.http import HttpResponse
from django.shortcuts import render, redirect
from django.contrib.auth.decorators import login_required
from django.db.models import Q
from django.contrib.auth.models import User
from django.contrib.auth import authenticate, login, logout
from django.contrib.auth.forms import UserCreationForm
from django.contrib import messages
from .models import Message, Room, Topic
from .forms import RoomForm

def login_page(request):

    page = 'login'

    if request.user.is_authenticated:
        return redirect('home')

    if request.method == "POST":
        username = request.POST.get('username').lower()
        password = request.POST.get('password')

        try:
            user = User.objects.get(username=username)
        except:
            messages.error(request, 'User does not exist.')

        user = authenticate(request, username=username, password=password)

        if user is not None:
            login(request, user)
            return redirect('home')
        else:
            messages.error(request, 'Username or password does not correct.')

    context = {'page': page}
    return render(request, 'base/login_register.html', context)

def logout_user(request):
    logout(request)

```

					ІАЛЦ.467200.007 Д4	Арк.
						13
Зм.	Арк.	№ докум.	Підпис	Дата		

```

        return redirect('home')

def register_page(request):
    form = UserCreationForm()

    if request.method == 'POST':
        form = UserCreationForm(request.POST)
        if form.is_valid():
            user = form.save(commit=False)

            user.username = user.username.lower()
            user.save()
            login(request, user)
            return redirect('home')
        else:
            messages.error(request, 'An error occurred during registration')

    context = { 'form': form }
    return render(request, 'base/login_register.html', context)

def home(request):

    q = request.GET.get('q') if request.GET.get('q') != None else ''
    rooms = Room.objects.filter(Q(topic__name__icontains=q) |
                                Q(name__icontains=q) |
                                Q(description__icontains=q)
                                )

    topics = Topic.objects.all()
    room_count = rooms.count()
    room_messages = Message.objects.filter(
        Q(room__topic__name__icontains=q)
    )

    context = { 'rooms': rooms, 'topics': topics, 'room_count': room_count, 'room_messages':
room_messages }
    return render(request, 'base/home.html', context)

def room(request, pk):
    room = Room.objects.get(id=pk)

    room_messages = room.message_set.all().order_by('-created')
    participants = room.participants.all()
    if request.method == 'POST':
        Message.objects.create(
            user = request.user,
            room=room,
            body=request.POST.get('body')
        )
        room.participants.add(request.user)
        return redirect('room', pk=room.id)

    context = { 'room': room, 'room_messages': room_messages, 'participants': participants }
    return render(request, 'base/room.html', context)

def user_profile(request, pk):
    user = User.objects.get(id=pk)
    room_messages = user.message_set.all()
    topics = Topic.objects.all()
    rooms = user.room_set.all()

```

					ІАЛЦ.467200.007 Д4	Арк.
						14
Зм.	Арк.	№ докум.	Підпис	Дата		

```

        context = { 'user': user, 'rooms': rooms, 'room_messages': room_messages, 'topics':
topics }
        return render(request, 'base/profile.html', context)

```

```

@login_required(login_url='/login')
def create_room(request):
    form = RoomForm()

    topics = Topic.objects.all()
    if request.method == 'POST':
        topic_name = request.POST.get('topic')
        topic, created = Topic.objects.get_or_create(name=topic_name)
        Room.objects.create(
            host=request.user,
            topic=topic,
            name=request.POST.get('name'),
            description=request.POST.get('description')
        )
        return redirect('home')

    context = {'form': form, 'topics': topics }
    return render(request, 'base/room_form.html', context)

```

```

@login_required(login_url='/login')
def update_room(request, pk):
    room = Room.objects.get(id=pk)
    form = RoomForm(instance=room)
    topics = Topic.objects.all()

    if request.user != room.host:
        return HttpResponse("Your are not allowed here!")

    if request.method == 'POST':
        topic_name = request.POST.get('topic')
        topic, created = Topic.objects.get_or_create(name=topic_name)
        room.name = request.POST.get('name')
        room.topic = topic
        room.description = request.POST.get('description')
        room.save()

        return redirect('home')

    context = {'form': form, 'topics': topics, 'room': room }
    return render(request, 'base/room_form.html', context)

```

```

@login_required(login_url='/login')
def delete_room(request, pk):
    room = Room.objects.get(id=pk)

    if request.user != room.host:
        return HttpResponse("Your are not allowed here!")

    if request.method == 'POST':
        room.delete()
        return redirect('home')

    context = {'obj': room}
    return render(request, 'base/delete.html', context)

```

```

@login_required(login_url='/login')
def delete_message(request, pk):

```

					ІАЛЦ.467200.007 Д4	Арк.
						15
Зм.	Арк.	№ докум.	Підпис	Дата		

```

message = Message.objects.get(id=pk)

if request.user != message.user:
    return HttpResponse("Your are not alloed here!")

if request.method == 'POST':
    message.delete()
    return redirect('home')

context = {'obj': message}
return render(request, 'base/delete.html', context)

```

#erik

#Zfewkfw2342352

```

<div class="activities">
  <div class="activities__header">
    <h2>Recent Activities</h2>
  </div>
  {% for message in room_messages %}
  <div class="activities__box">
    <div class="activities__boxHeader roomListRoom__header">
      <a href="{% url 'user-profile' message.user.id %}" class="roomListRoom__author">
        <div class="avatar avatar--small">
          
        </div>
        <p>
          @{{message.user}}
          <span>{{message.created|timesince}} ago</span>
        </p>
      </a>
      {% if request.user == message.user %}
      <div class="roomListRoom__actions">
        <a href="{% url 'delete-message' message.id %}">
          <svg version="1.1" xmlns="http://www.w3.org/2000/svg" width="32" height="32"
viewBox="0 0 32 32">
            <title>remove</title>
            <path
              d="M27.314 6.019l-1.333-1.333-9.98 9.981-9.981-9.981-1.333 1.333 9.981
9.981-9.981 9.98 1.333 1.333 9.981-9.98 9.98 9.98 1.333-1.333-9.98-9.98 9.98-9.981z"
            ></path>
          </svg>
        </a>
      </div>
      {% endif %}
    </div>
    <div class="activities__boxContent">
      <p>replied to post “<a href="{% url 'room' message.room.id
%}”>{{message.room}}</a>”</p>
      <div class="activities__boxRoomContent">
        {{message.body}}
      </div>
    </div>
  </div>
  {% endfor %}
</div>

{% extends 'index.html' %}

{% block content %}

  <form method="POST"action="">
    {% csrf_token %}
    <p>Are you sure you want to delete "{{obj}}?" </p>

```

					ІАЛЦ.467200.007 Д4	Арк.
						16
Зм.	Арк.	№ докум.	Підпис	Дата		

```

        <a href="{{request.META.HTTP_REFERER}}">Go Back</a>
        <input type="Submit" value="Confirm">
    </form>

{% endblock content %}

{% for room in rooms %}
<div class="roomListRoom">
    <div class="roomListRoom__header">
        <a href="{% url 'user-profile' room.host.id %}" class="roomListRoom__author">
            <div class="avatar avatar--small">
                
            </div>
            <span>@{{room.host.username}}</span>
        </a>
        <div class="roomListRoom__actions">
            <span>{{room.created|timesince}} ago</span>
        </div>
    </div>
    <div class="roomListRoom__content">
        <a href="{% url 'room' room.id %}">{{room.name}}</a>
    </div>
    <div class="roomListRoom__meta">
        <a href="{% url 'room' room.id %}" class="roomListRoom__joined">
            <svg version="1.1" xmlns="http://www.w3.org/2000/svg" width="32" height="32"
viewBox="0 0 32 32">
                <title>user-group</title>
                <path
                    d="M30.539 20.766c-2.69-1.547-5.75-2.427-8.92-2.662 0.649 0.291 1.303 0.575
1.918 0.928 0.715 0.412 1.288 1.005 1.71 1.694 1.507 0.419 2.956 1.003 4.298 1.774 0.281
0.162 0.456 0.487 0.456 0.85v4.65h-4v2h5c0.553 0 1-0.447 1-1v-5.65c0-1.077-0.56-2.067-1.461-
2.584z"
                ></path>
                <path
                    d="M22.539 20.766c-6.295-3.619-14.783-3.619-21.078 0-0.901 0.519-1.461 1.508-
1.461 2.584v5.65c0 0.553 0.447 1 1 1h22c0.553 0 1-0.447 1-1v-5.651c0-1.075-0.56-2.064-1.461-
2.583zM22 28h-20v-4.65c0-0.362 0.175-0.688 0.457-0.85 5.691-3.271 13.394-3.271 19.086 0
0.282 0.162 0.457 0.487 0.457 0.849v4.651z"
                ></path>
                <path
                    d="M19.502 4.047c0.166-0.017 0.33-0.047 0.498-0.047 2.757 0 5 2.243 5 5s-2.243
5-5 5c-0.168 0-0.332-0.030-0.498-0.047-0.424 0.641-0.944 1.204-1.513 1.716 0.651 0.201 1.323
0.331 2.011 0.331 3.859 0 7-3.141 7-7s-3.141-7-7-7c-0.688 0-1.36 0.131-2.011 0.331 0.57
0.512 1.089 1.075 1.513 1.716z"
                ></path>
                <path
                    d="M12 16c3.859 0 7-3.141 7-7s-3.141-7-7-7c-3.859 0-7 3.141-7 7s3.141 7 7 7zM12
4c2.757 0 5 2.243 5 5s-2.243 5-5 5-5-2.243-5-5c0-2.757 2.243-5 5-5z"
                ></path>
            </svg>
            {{room.participants.all.count}} Joined
        </a>
        <p class="roomListRoom__topic">{{room.topic.name}}</p>
    </div>
</div>
{% endfor %}

{% extends 'index.html'%}

{% block content %}
    <main class="layout layout--3">
        <div class="container">
            <!-- Topics Start -->

```

					ІАЛЦ.467200.007 Д4	Арк.
						17
Зм.	Арк.	№ докум.	Підпис	Дата		

```

{% include 'base/topics_component.html' %}
<!-- Topics End -->

<!-- Room List Start -->
<div class="roomList">
  <div class="mobile-menu">
    <form class="header__search">
      <label>
        <svg version="1.1" xmlns="http://www.w3.org/2000/svg" width="32" height="32"
viewBox="0 0 32 32">
          <title>search</title>
          <path
            d="M32 30.5861-10.845-10.845c1.771-2.092 2.845-4.791 2.845-7.741 0-
6.617-5.383-12-12-12s-12 5.383-12 12c0 6.617 5.383 12 12 12 2.949 0 5.649-1.074 7.741-
2.845l10.845 10.845 1.414-1.414zM12 22c-5.514 0-10-4.486-10-10s4.486-10 10-10c5.514 0 10
4.486 10 10s-4.486 10-10 10z"
          ></path>
        </svg>
        <input placeholder="Search for posts" />
      </label>
    </form>
    <div class="mobile-menuItems">
      <a class="btn btn--main btn--pill" href="#">Browse Topics</a>
      <a class="btn btn--main btn--pill" href="#">Recent Activities</a>
    </div>
  </div>
  <div class="roomList__header">
    <div>
      <h2>Study Room</h2>
      <p>{{room_count}} Rooms available</p>
    </div>
    <a class="btn btn--main" href="{% url 'create-room' %}">
      <svg version="1.1" xmlns="http://www.w3.org/2000/svg" width="32" height="32"
viewBox="0 0 32 32">
        <title>add</title>
        <path
          d="M16.943 0.943h-1.885v14.115h-14.115v1.885h14.115v14.115h1.885v-
14.115h14.115v-1.885h-14.115v-14.115z"
        ></path>
      </svg>
      Create Room
    </a>
  </div>

  {% include 'base/feed_component.html' %}

</div>
<!-- Room List End -->

<!-- Activities Start -->
{% include 'base/activity_component.html' %}
<!-- Activities End -->
</div>
</main>
{% endblock content %}

{% extends 'index.html' %}

{% block content %}
{% if page == 'login' %}
<div>
  <form method="POST" action="">
    {% csrf_token %}

    <label>Username:</label>

```

					ІАЛЦ.467200.007 Д4	Арк.
						18
Зм.	Арк.	№ докум.	Підпис	Дата		

```

        <input type="text" name="username" placeholder="Enter UserName" />

        <label>Password:</label>
        <input type="password" name="password" placeholder="Enter Password" />

        <input type="submit" value="Login" />
    </form>

    <p>Haven`t signed up yet?</p>
    <a href="{% url 'register' %}">Sign up</a>
</div>
{% else %}
<div>
    <form method="POST" action="">
        {% csrf_token %}
        {{ form.as_p }}
        <input type="submit" value="Register" />
    </form>

    <p>Already signed up?</p>
    <a href="{ url 'login' }">Sign in</a>
</div>
{% endif %}
{% endblock content %}

{% extends 'index.html' %}

{% block content %}
    <main class="profile-page layout layout--3">
        <div class="container">
            <!-- Topics Start -->
            {% include 'base/topics_component.html' %}
            <!-- Topics End -->

            <!-- Room List Start -->
            <div class="roomList">
                <div class="profile">
                    <div class="profile__avatar">
                        <div class="avatar avatar--large active">
                            
                        </div>
                    </div>
                    <div class="profile__info">
                        <h3>{{user.username}}</h3>
                        <p>@{{user.username}}</p>
                        <a href="edit-user.html" class="btn btn--main btn--pill">Edit Profile</a>
                    </div>
                    <div class="profile__about">
                        <h3>About</h3>
                        <p>
tenetur                Lorem ipsum dolor sit amet consectetur adipisicing elit. Consequuntur illo
                            facilis sunt nemo debitis quisquam hic atque aut? Ducimus alias placeat optio
                            accusamus repudiandae quis ab ex exercitationem rem?
                        </p>
                    </div>
                </div>
            </div>

            <div class="roomList__header">
                <div>
                    <h2>Study Rooms Hosted by dennis_ivy</a>
                </h2>
            </div>
        </div>
    </main>
</div>

```

					ІАЛЦ.467200.007 Д4	Арк.
						19
Зм.	Арк.	№ докум.	Підпис	Дата		

```

        {% include 'base/feed_component.html' %}
    </div>
    <!-- Room List End -->

    <!-- Activities Start -->
    {% include 'base/activity_component.html' %}
    <!-- Activities End -->
</div>
</main>
{% endblock content %}

{% extends 'index.html' %}

{% block content %}

<main class="profile-page layout layout--2">
    <div class="container">
        <!-- Room Start -->
        <div class="room">
            <div class="room__top">
                <div class="room__topLeft">
                    <a href="{% url 'home' %}">
                        <svg version="1.1" xmlns="http://www.w3.org/2000/svg" width="32" height="32"
viewBox="0 0 32 32">
                            <title>arrow-left</title>
                            <path
                                d="M13.723 2.286l-13.723 13.714 13.719 13.714 1.616-1.611-10.96-
10.96h27.625v-2.286h-27.625l10.965-10.965-1.616-1.607z"
                                ></path>
                            </svg>
                        </a>
                        <h3>Study Room</h3>
                    </div>
                    {% if room.host == request.user %}
                    <div class="room__topRight">
                        <a href="{% url 'update-room' room.id %}">
                            <svg
                                enable-background="new 0 0 24 24"
                                height="32"
                                viewBox="0 0 24 24"
                                width="32"
                                xmlns="http://www.w3.org/2000/svg"
                            >
                                <title>edit</title>
                                <g>
                                    <path d="m23.5 22h-15c-.276 0-.5-.224-.5-.5s.224-.5.5-.5h15c.276 0
.5.224.5.5s-.224.5-.5.5z" />
                                </g>
                                <g>
                                    <path
                                        d="m2.5 22c-.131 0-.259-.052-.354-.146-.123-.123-.173-.3-.133-
.468l1.09-4.625c.021-.09.067-.173.133-.239l14.143-14.143c.565-.566 1.554-.566 2.121 0l2.121
2.121c.283.283.439.66.439 1.061s-.156.778-.439 1.061l-14.142 14.141c-.065.066-.148.112-
.239.133l-4.625 1.09c-.038.01-.077.014-.115.014zm1.544-4.873-.872 3.7 3.7-.872 14.042-
14.041c.095-.095.146-.22.146-.354 0-.133-.052-.259-.146-.354l-2.121-2.121c-.19-.189-.518-
.189-.707 0zm3.081 3.283h.01z"
                                    >
                                </g>
                                <g>
                                    <path

```

					ІАЛЦ.467200.007 Д4	Арк.
						20
Зм.	Арк.	№ докум.	Підпис	Дата		

```

d="m17.889 10.146c-.128 0-.256-.049-.354-.146l-3.535-3.536c-.195-
.195-.195-.512 0-.707s.512-.195.707 0l3.536 3.536c.195.195.512 0 .707-.098.098-.226.146-
.354.146z"
      />
    </g>
  </g>
</svg>
</a>
<a href="{% url 'delete-room' room.id %}">
  <svg version="1.1" xmlns="http://www.w3.org/2000/svg" width="32" height="32"
viewBox="0 0 32 32">
    <title>remove</title>
    <path
      d="M27.314 6.019l-1.333-1.333-9.98 9.981-9.981-9.981-1.333 1.333 9.981
9.981-9.981 9.98 1.333 1.333 9.981-9.98 9.98 9.98 1.333-1.333-9.98-9.98 9.98-9.981z"
    ></path>
  </svg>
</a>
</div>
{% endif %}
<!-- <button class="action-button" data-id="120" data-delete-
url="https://randomuser.me/api/3324923"
data-edit-url="profile.html">
  <svg version="1.1" xmlns="http://www.w3.org/2000/svg" width="32" height="32"
viewBox="0 0 32 32">
    <title>ellipsis-horizontal</title>
    <path
      d="M16 7.843c-2.156 0-3.908-1.753-3.908-3.908s1.753-3.908 3.908-3.908c2.156
0 3.908 1.753 3.908 3.908s-1.753 3.908-3.908 3.908zM16 1.98c-1.077 0-1.954 0.877-1.954
1.954s0.877 1.954 1.954 1.954c1.077 0 1.954-0.877 1.954-1.954s-0.877-1.954-1.954-1.954z">
    </path>
    <path
      d="M16 19.908c-2.156 0-3.908-1.753-3.908-3.908s1.753-3.908 3.908-3.908c2.156
0 3.908 1.753 3.908 3.908s-1.753 3.908-3.908 3.908zM16 14.046c-1.077 0-1.954 0.877-1.954
1.954s0.877 1.954 1.954 1.954c1.077 0 1.954-0.877 1.954-1.954s-0.877-1.954-1.954-1.954z">
    </path>
    <path
      d="M16 31.974c-2.156 0-3.908-1.753-3.908-3.908s1.753-3.908 3.908-3.908c2.156
0 3.908 1.753 3.908 3.908s-1.753 3.908-3.908 3.908zM16 26.111c-1.077 0-1.954 0.877-1.954
1.954s0.877 1.954 1.954 1.954c1.077 0 1.954-0.877 1.954-1.954s-0.877-1.954-1.954-1.954z">
    </path>
  </svg>
</button> -->
</div>
<div class="room__box scroll">
  <div class="room__header scroll">
    <div class="room__info">
      <h3>{{room.name}}</h3>
      <span>{{room.created|timesince}}</span>
    </div>
    <div class="room__hosted">
      <p>Hosted By</p>
      <a href="{% url 'user-profile' room.host.id %}" class="room__author">
        <div class="avatar avatar--small">
          
        </div>
        <span>@{{room.host}}</span>
      </a>
    </div>
    <!-- <div class="room__details">
      Lorem ipsum dolor sit, amet consectetur adipisicing elit. Quasi, tenetur
laudantium? Dicta repellendus
rerum consectetur, voluptatem repudiandae eum ea porro cupiditate provident
nulla, deserunt, ab ipsum
corporis laboriosam deleniti molestias?

```

					ІАЛЦ.467200.007 Д4	Арк.
						21
Зм.	Арк.	№ докум.	Підпис	Дата		

```

    </div> -->
    <span class="room__topics">{{room.topic}}</span>
</div>
<div class="room__conversation">
    <div class="threads scroll">
        {% for message in room_messages %}
        <div class="thread">
            <div class="thread__top">
                <div class="thread__author">
                    <a href="{% url 'user-profile' user.id %}" class="thread__authorInfo">
                        <div class="avatar avatar--small">
                            
                        </div>
                        <span>@{{message.user.username}}</span>
                    </a>
                    <span class="thread__date">{{message.created|timesince}} ago</span>
                </div>
                {% if request.user == message.user %}
                <a href="{% url 'delete-message' message.id %}">
                    <div class="thread__delete">
                        <svg version="1.1" xmlns="http://www.w3.org/2000/svg" width="32"
height="32" viewBox="0 0 32 32">
                            <title>remove</title>
                            <path
                                d="M27.314 6.019l-1.333-1.333-9.98 9.981-9.981-9.981-1.333 1.333
9.981 9.981-9.981 9.98 1.333 1.333 9.981-9.98 9.98 9.98 1.333-1.333-9.98-9.98 9.98-9.981z"
                            ></path>
                        </svg>
                    </div>
                </a>
                {% endif %}
            </div>
            <div class="thread__details">
                {{message.body}}
            </div>
        </div>
        {% endfor %}
    </div>
</div>
</div>
{% if request.user.is_authenticated %}
<div class="room__message">
    <form method="POST" action="">
        {% csrf_token %}
        <input type="text" name="body" placeholder="Write your message here..." />
    </form>
</div>
{% endif %}
</div>

<!-- Room End -->

<!-- Start -->
<div class="participants">
    <h3 class="participants__top">Participants <span>({{participants.count}}
Joined)</span></h3>
    <div class="participants__list scroll">
        {% for user in participants %}
        <a href="{% url 'user-profile' user.id %}" class="participant">
            <div class="avatar avatar--medium">
                
            </div>
            <p>
                {{user.username}}
                <span>@{{user.username}}</span>
            </p>
        </a>
        </div>
        </div>
    </div>

```

					ІАЛЦ.467200.007 Д4	Арк.
						22
Зм.	Арк.	№ докум.	Підпис	Дата		

```

        </p>
        </a>
        {% endfor %}
    </div>
</div>
<!-- End -->
</div>
</main>

{% endblock content %}

<div class="topics">
    <div class="topics__header">
        <h2>Browse Topics</h2>
    </div>
    <ul class="topics__list">
        <li>
            <a href="{% url 'home' %}" class="active">All <span>{{topics.count}}</span></a>
        </li>

        {% for topic in topics %}
        <li>
            <a href="{% url 'home' %}?q={{topic.name}}">{{topic.name}}
            <span>{{topic.room_set.all.count}}</span></a>
        </li>
        {% endfor %}
    </ul>
    <a class="btn btn--link" href="topics.html">
        More
        <svg version="1.1" xmlns="http://www.w3.org/2000/svg" width="32" height="32"
viewBox="0 0 32 32">
            <title>chevron-down</title>
            <path d="M16 21l-13-13h-3l16 16 16-16h-3l-13 13z"></path>
        </svg>
    </a>
</div>

<!DOCTYPE html>
{% load static %}
<html lang="en">
    <head>
        <meta charset="UTF-8" />
        <meta http-equiv="X-UA-Compatible" content="IE=edge" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0" />
        <link rel="shortcut icon" href="assets/favicon.ico" type="image/x-icon" />
        <link rel="stylesheet" href="{% static 'styles/style.css' %}" />
        <title>StudyBuddy - Find study partners around the world!</title>
    </head>

    <body>
        {% include 'navbar.html' %}

        {% if messages %}
        <ul class="messages">
            {% for message in messages %}
            <li>
                {{ message }}
            </li>
            {% endfor %}
        </ul>
        {% endif %}

        {% block content %}

        {% endblock content %}
    </body>
</html>

```

					ІАЛЦ.467200.007 Д4	Арк.
						23
Зм.	Арк.	№ докум.	Підпис	Дата		

```

    <script src="{% static 'js/script.js' %}"></script>
  </body>
</html>

{% load static %}

<header class="header header--loggedIn">
  <div class="container">
    <a href="{% url 'home' %}" class="header__logo">
      
      <h1>StudyBuddy</h1>
    </a>
    <form class="header__search" method="GET" action="{% url 'home' %}">
      <label>
        <svg version="1.1" xmlns="http://www.w3.org/2000/svg" width="32" height="32"
viewBox="0 0 32 32">
          <title>search</title>
          <path
            d="M32 30.5861-10.845-10.845c1.771-2.092 2.845-4.791 2.845-7.741 0-6.617-
5.383-12-12-12s-12 5.383-12 12c0 6.617 5.383 12 12 12 2.949 0 5.649-1.074 7.741-2.845l10.845
10.845 1.414-1.414zM12 22c-5.514 0-10-4.486-10-10s4.486-10 10-10c5.514 0 10 4.486 10 10s-
4.486 10-10 10z"
          ></path>
        </svg>
        <input name="q" placeholder="Search for rooms..." />
      </label>
    </form>
    <nav class="header__menu">
      <!-- Not Logged In -->
      <!-- Logged In -->
      {% if request.user.is_authenticated %}
      <div class="header__user">
        <a href="profile.html">
          <div class="avatar avatar--medium active">
            
          </div>
          <p>{{ request.user.username }} <span>@{{ request.user.username }}</span></p>
        </a>
        <button class="dropdown-button">
          <svg version="1.1" xmlns="http://www.w3.org/2000/svg" width="32" height="32"
viewBox="0 0 32 32">
            <title>chevron-down</title>
            <path d="M16 21-13-13h-3l16 16 16-16h-3l-13 13z"></path>
          </svg>
        </button>
      </div>
      {% else %}
      <a href="{% url 'login' %}">
        
        <p>Login</p>
      </a>
      {% endif %}
      <div class="dropdown-menu">
        <a href="settings.html" class="dropdown-link">
          >>svg version="1.1" xmlns="http://www.w3.org/2000/svg" width="32" height="32"
viewBox="0 0 32 32">
            <title>tools</title>
            <path
              d="M27.465 32c-1.211 0-2.35-0.471-3.207-1.328l-9.392-9.391c-2.369 0.898-
4.898 0.951-7.355 0.15-3.274-1.074-5.869-3.67-6.943-6.942-0.879-2.682-0.734-5.45 0.419-8.004
0.135-0.299 0.408-0.512 0.731-0.572 0.32-0.051 0.654 0.045 0.887 0.277l5.394 5.395 3.586-
3.586-5.394-5.395c-0.232-0.232-0.336-0.564-0.276-0.887s0.272-0.596 0.572-0.732c2.552-1.152
5.318-1.295 8.001-0.418 3.274 1.074 5.869 3.67 6.943 6.942 0.806 2.457 0.752 4.987-0.15
7.358l9.392 9.391c0.844 0.842 1.328 2.012 1.328 3.207-0 2.5-2.034 4.535-4.535 4.535zM15.101
19.102c0.26 0 0.516 0.102 0.707 0.293l9.864 9.863c0.479 0.479 1.116 0.742 1.793 0.742 1.398

```

					ІАЛЦ.467200.007 Д4	Арк.
						24
Зм.	Арк.	№ докум.	Підпис	Дата		

```

0 2.535-1.137 2.535-2.535 0-0.668-0.27-1.322-0.742-1.7931-9.864-9.863c-0.294-0.295-0.376-
0.74-0.204-1.119 0.943-2.090 1.061-4.357 0.341-6.555-0.863-2.631-3.034-4.801-5.665-5.666-
1.713-0.561-3.468-0.609-5.145-0.16414.986 4.988c0.391 0.391 0.391 1.023 0 1.4141-5 5c-0.188
0.188-0.441 0.293-0.707 0.293s-0.52-0.105-0.707-0.2931-4.987-4.988c-0.45 1.682-0.397 3.436
0.164 5.146 0.863 2.631 3.034 4.801 5.665 5.666 2.2 0.721 4.466 0.604 6.555-0.342 0.132-
0.059 0.271-0.088 0.411-0.088z"
    ></path>
  </svg>
  Settings</a>
  >
  <a href="{% url 'logout' %}" class="dropdown-link"
    ><svg version="1.1" xmlns="http://www.w3.org/2000/svg" width="32" height="32"
viewBox="0 0 32 32">
    <title>sign-out</title>
    <path
      d="M3 0h22c0.553 0 1 0 1 0.5531-0 3.447h-2v-2h-20v28h20v-2h210 3.447c0
0.553-0.447 0.553-1 0.553h-22c-0.553 0-1-0.447-1-1v-30c0-0.553 0.447-1 1-1z"
    ></path>
    <path
      d="M21.879 21.29311.414 1.414 6.707-6.707-6.707-6.707-1.414 1.414 4.293
4.293h-14.172v2h14.1721-4.293 4.293z"
    ></path>
  </svg>
  Logout</a>
  >
</div>
</nav>
</div>
</header>

```

```

<!-- {% if request.user.is_authenticated %}
<p>Hello {{ request.user }}</p>
<a href="{% url 'logout' %}">Logout</a>
{% else %}
<a href="{% url 'login' %}">Login</a>
{% endif %}

<hr> -->

```

```

#!/usr/bin/env python
"""Django's command-line utility for administrative tasks."""
import os
import sys

def main():
    """Run administrative tasks."""
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'studybud.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)

if __name__ == '__main__':
    main()

```

					ІАЛЦ.467200.007 Д4	Арк.
						25
Зм.	Арк.	№ докум.	Підпис	Дата		

```

from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('base.urls')),
]

body{
    background-color: aquamarine;
}

@import
url("https://fonts.googleapis.com/css2?family=DM+Sans:wght@400;500;700&display=swap");

:root {
    --color-main: #71c6dd;
    --color-main-light: #e1f6fb;
    --color-dark: #3f4156;
    --color-dark-medium: #51546e;
    --color-dark-light: #696d97;
    --color-light: #e5e5e5;
    --color-gray: #8b8b8b;
    --color-light-gray: #bdbdbd;
    --color-bg: #2d2d39;
    --color-success: #5dd693;
    --color-error: #fc4b0b;
}

/*===== base styles =====*/

* {
    font-family: "DM Sans", -apple-system, BlinkMacSystemFont, "Segoe UI", Roboto, Oxygen,
    Ubuntu, Cantarell, "Open Sans",
    "Helvetica Neue", sans-serif;
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    text-rendering: optimizeLegibility;
    /* color: inherit; */
    font-size: inherit;
}

html {
    font-size: 56.25%;
}

@media only screen and (min-width: 1200px) {
    html {
        font-size: 62.5%;
    }
}

@media only screen and (min-width: 2100px) {
    html {
        font-size: 75%;
    }
}

body {
    line-height: 1.6;

```

					ІАЛЦ.467200.007 Д4	Арк.
						26
Зм.	Арк.	№ докум.	Підпис	Дата		

```

font-weight: 400;
font-size: 1.5rem;
color: var(--color-light-gray);
background-color: var(--color-bg);
min-height: 100vh;
}

img {
width: 100%;
}

a {
display: inline-block;
color: var(--color-main);
text-decoration: none;
}

/*===== components =====*/
.container {
max-width: 120rem;
width: 90%;
margin: auto;
}

.btn {
background-color: transparent;
border: none;
display: inline-flex;
align-items: center;
gap: 1rem;
cursor: pointer;
transition: all ease-in-out 0.3s;
padding: 1rem 2rem;
border-radius: 5px;
box-shadow: 1px 1px 3px 1px rgba(0, 0, 0, 0.15);
font-weight: 500;
}

.btn--link {
border-radius: 0;
padding: 0;
color: var(--color-main);
box-shadow: none;
}

.btn--link:hover {
text-decoration: underline;
}

.btn--main {
background-color: var(--color-main);
color: var(--color-dark);
}

.btn:hover {
opacity: 0.9;
}

.btn--dark {
background-color: var(--color-dark-light);
color: var(--color-light);
}

.btn > svg {
fill: currentColor;
}

```

					ІАЛЦ.467200.007 Д4	Арк.
						27
Зм.	Арк.	№ докум.	Підпис	Дата		

```

width: 1.6rem;
height: 1.6rem;
}

.btn--pill {
border-radius: 10rem;
font-size: 1.4rem;
font-weight: 700;
padding: 6px 2.5rem;
color: var(--color-main);
background: transparent;
border: 2px solid var(--color-main);
}

.action-button {
background: transparent;
border: none;
outline: none;
cursor: pointer;
}

.avatar {
position: relative;
display: inline-block;
border-radius: 50%;
border: 2px solid var(--color-main);
}

.avatar img {
display: block;
border-radius: 50%;
object-fit: cover;
object-position: center;
}

.avatar::after {
content: "";
display: block;
position: absolute;
background-color: var(--color-gray);
z-index: 111;
border-radius: 50%;
border: 0.3rem solid var(--color-dark);
}

.avatar.active::after {
background-color: var(--color-success);
}

.avatar.avatar--small img {
width: 2.8rem;
height: 2.8rem;
}

.avatar.avatar--small:after {
width: 0.7rem;
height: 0.7rem;
bottom: 0px;
right: -6px;
}

.avatar.avatar--medium img {
width: 3.6rem;
height: 3.6rem;
border-radius: 50%;
}

```

					ІАЛЦ.467200.007 Д4	Арк.
						28
Зм.	Арк.	№ докум.	Підпис	Дата		

```

}

.avatar.avatar--medium:after {
  width: 0.7rem;
  height: 0.7rem;
  bottom: 0px;
  right: -6px;
}

.avatar.avatar--large img {
  display: block;
  width: 8rem;
  height: 8rem;
  border-radius: 50%;
}

.avatar.avatar--large:after {
  width: 1rem;
  height: 1rem;
  bottom: 2px;
  right: 3.5px;
}

.scroll::-webkit-scrollbar {
  width: 0.6rem;
  background-color: rgb(41, 41, 46);
}

.scroll::-webkit-scrollbar-thumb {
  border-radius: 1rem;
  background-color: var(--color-gray);
}

.dropdown-menu {
  z-index: 111;
  position: absolute;
  top: 5rem;
  right: 0.5rem;
  background: var(--color-dark-light);
  border-radius: 5px;
  box-shadow: 1px 1px 3px 1px rgba(0, 0, 0, 0.15);
  overflow: hidden;
  display: none;
}

.dropdown-menu.show {
  display: block;
}

.dropdown-menu a {
  padding: 1.2rem 4rem;
  display: block;
  color: var(--color-light) !important;
  font-weight: 500;
  font-size: 1.4rem;
}

.dropdown-menu a:hover {
  background-color: var(--color-dark-medium);
}

.dropdown-menu > a:not(:last-child) {
  border-bottom: 1px solid var(--color-dark-medium);
}

```

					ІАЛЦ.467200.007 Д4	Арк.
						29
Зм.	Арк.	№ докум.	Підпис	Дата		

```

.dropdown-menu a svg {
  fill: var(--color-light);
}

.mobile-menu {
  margin-bottom: 3rem;
}

.mobile-menuItems {
  display: flex;
  align-items: center;
  justify-content: center;
  gap: 1rem;
}

@media screen and (min-width: 500px) {
  .mobile-menu {
    display: none;
  }
}

/*=====
=>  Header Section
=====*/

.header {
  padding: 1.5rem;
  background-color: var(--color-dark);
}

.header > .container {
  display: flex;
  gap: 9.5rem;
}

.header__logo,
.header__user {
  display: flex;
  gap: 2rem;
  align-items: center;
}

.header__logo > img {
  height: 3.2rem;
  width: 3.2rem;
}

.header__logo > h1 {
  font-weight: 700;
  font-size: 2rem;
  color: var(--color-light);
}

.header__search > label {
  background-color: var(--color-dark-medium);
  padding: 1.3rem 1rem;
  display: flex;
  align-items: center;
  gap: 1rem;
  border-radius: 4px;
}

.header__search svg {
  fill: var(--color-gray);
  width: 2rem;

```

					ІАЛЦ.467200.007 Д4	Арк.
						30
Зм.	Арк.	№ докум.	Підпис	Дата		

```

height: 2rem;
margin-left: 1rem;
}

.header__search input {
width: 30rem;
background: transparent;
border: none;
outline: none;
color: var(--color-light);
}

@media screen and (max-width: 800px) {
.header__search input {
width: 20rem;
}

.header > .container {
gap: 3rem;
}
}

@media screen and (max-width: 700px) {
.header__logo h1 {
display: none;
}
}

@media screen and (max-width: 500px) {
.header__search {
display: none;
}
}

.header__menu {
margin-left: auto;
position: relative;
}

.header__menu a {
display: flex;
gap: 1.5rem;
align-items: center;
font-weight: 500;
text-decoration: none;
color: var(--color-gray);
}

.header__menu img {
height: 3.6rem;
}

.header__menu p {
line-height: 1.2;
}

.header__menu span {
color: var(--color-main);
font-weight: 500;
font-size: 1.4rem;
display: block;
}

.header__menu svg {
width: 1.6rem;
}

```

					ІАЛЦ.467200.007 Д4	Арк.
						31
Зм.	Арк.	№ докум.	Підпис	Дата		

```

height: 1.6rem;
fill: var(--color-dark-light);
}

.dropdown-button {
background: transparent;
border: 0;
outline: 0;
cursor: pointer;
}

.dropdown-button:hover svg {
fill: var(--color-main);
}

/*=====
=> Layout
=====*/

.layout {
margin-top: 2.4rem;
}

.layout > .container {
display: flex;
justify-content: space-between;
align-items: flex-start;
}

.layout--3 > .container > div:first-child {
flex-basis: 18%;
max-width: 22.5rem;
}

.layout--3 > .container > div:nth-child(2) {
flex-basis: 50%;
}

.layout--3 > .container > div:last-child {
flex-basis: 25%;
}

.layout--2 > .container > div:first-child {
flex-basis: 72%;
}

.layout--2 > .container > div:last-child {
flex-basis: 25%;
}

/*===== Layout Box =====*/

.layout__box {
width: 90%;
max-width: 48rem;
min-height: 40rem;
position: absolute;
top: 50%;
left: 50%;
transform: translate(-50%, -46%);
background-color: var(--color-dark);
border-radius: 1rem;
box-shadow: 1px 1px 6px 3px rgba(0, 0, 0, 0.1);
overflow: hidden;
}

```

					ІАЛЦ.467200.007 Д4	Арк.
						32
Зм.	Арк.	№ докум.	Підпис	Дата		

```

.layout__boxHeader {
  display: flex;
  padding: 1.5rem;
  background-color: var(--color-dark-light);
}

.layout__boxTitle {
  display: flex;
  gap: 1.5rem;
  align-items: center;
}

.layout__boxTitle h3 {
  text-transform: uppercase;
  font-weight: 500;
  color: var(--color-light);
}

.layout__boxTitle svg {
  width: 1.6rem;
  height: 1.6rem;
  fill: var(--color-main);
}

.layout__body {
  margin: 3rem;
}

@media screen and (max-width: 900px) {
  .activities,
  .topics {
    display: none;
  }

  .layout--3 > .container > div:nth-child(2) {
    flex-basis: 100%;
  }
}

/*=====
=> Topics
=====*/

.form__group {
  margin-bottom: 2rem;
  width: 100%;
}

.form__split {
  display: flex;
  gap: 1.5rem;
}

.form__group label {
  display: block;
  font-size: 1.5rem;
  margin-bottom: 1rem;
}

.form__group input,
.form__group textarea,
.form__group select {
  background: transparent;
  border: 1px solid var(--color-dark-light);
  padding: 1rem;
}

```

					ІАЛЦ.467200.007 Д4	Арк.
						33
Зм.	Арк.	№ докум.	Підпис	Дата		

```

border-radius: 3px;
width: 100%;
color: var(--color-light);
font-weight: 500;
outline: none;
}

.form__group input:focus,
.form__group textarea:focus {
border-color: var(--color-main);
}

.form__group textarea {
background: transparent;
height: 10rem;
resize: none;
}

.form__group select {
color: var(--color-gray);
font-weight: 400;
}

.form__group select option {
background-color: var(--color-dark-light);
color: var(--color-light);
padding: 0 10rem;
}

.form__action {
display: flex;
justify-content: flex-end;
gap: 3rem;
}

.form__hide {
position: absolute;
left: -9999px;
}

.form__avatar label {
text-align: center;
font-size: 1.8rem;
font-weight: 500;
color: var(--color-main);
cursor: pointer;
}

.form__avatar label:hover {
text-decoration: underline;
}

/*=====
=> Topics
=====*/

.topics__header {
margin-bottom: 2rem;
}

.topics__header h2 {
text-transform: uppercase;
font-weight: 500;
color: var(--color-dark-light);
}

```

					ІАЛЦ.467200.007 Д4	Арк.
						34
Зм.	Арк.	№ докум.	Підпис	Дата		

```

.topics__list {
  list-style: none;
}

.topics__list li a {
  display: flex;
  justify-content: space-between;
  margin-bottom: 3rem;
  font-weight: 500;
  color: var(--color-light-gray);
  transition: all 0.3s ease-in-out;
}

.topics__list li a.active,
.topics__list li a:hover {
  color: var(--color-main);
}

.topics__list li a span {
  padding: 0.5rem 1rem;
  background-color: var(--color-dark);
  border-radius: 3px;
  font-size: 1.3rem;
  font-weight: 700;
  letter-spacing: 1px;
}

.topics-page a:hover {
  text-decoration: underline;
}

.topics-page .topics__list li:not(:last-child) a {
  margin: 2rem 0;
  padding-bottom: 1rem;
  text-decoration: none;
  border-bottom: 1px solid var(--color-dark-medium);
}

.topics-page .header__search {
  display: block;
}

@media screen and (max-width: 500px) {
  .mobile-menu .header__search {
    display: block;
    margin-bottom: 2.4rem;
  }
}

/*=====
=> Room List
=====*/

.roomList__header {
  display: flex;
  justify-content: space-between;
  align-items: center;
  margin-bottom: 2.4rem;
}

.roomList__header h2 {
  text-transform: uppercase;
  font-weight: 500;
  color: var(--color-light);
}

```

					ІАЛЦ.467200.007 Д4	Арк.
						35
Зм.	Арк.	№ докум.	Підпис	Дата		

```

    letter-spacing: 1px;
}

.roomList__header p {
    font-weight: 500;
    color: var(--color-dark-light);
}

/*===== Room List Room =====*/

.roomListRoom {
    margin-bottom: 2.4rem;
    background-color: var(--color-dark);
    border-radius: 1rem;
    padding: 2rem;
}

.roomListRoom__header {
    display: flex;
    justify-content: space-between;
    align-items: center;
}

.roomListRoom__author {
    font-weight: 500;
    display: flex;
    align-items: center;
    gap: 1rem;
}

.roomListRoom__actions {
    display: flex;
    align-items: flex-start;
    gap: 1rem;
    position: relative;
}

.roomListRoom__actions span {
    font-size: 1.4rem;
    font-weight: 500;
}

.roomListRoom__actions svg {
    fill: var(--color-main);

    width: 1.6rem;
    height: 1.6rem;
}

.roomListRoom__content {
    margin: 1rem 0;
}

.roomListRoom__content a {
    font-size: 2rem;
    font-weight: 500;
    margin-bottom: 1.5rem;
    color: var(--color-light);
    transition: all 0.3s ease-in-out;
}

.roomListRoom__content a:hover {
    color: var(--color-main);
}

```

					ІАЛЦ.467200.007 Д4	Арк.
						36
Зм.	Арк.	№ докум.	Підпис	Дата		

```

.roomListRoom__meta {
  border-top: 1px solid var(--color-dark-medium);
  padding-top: 1rem;
  display: flex;
  align-items: center;
  justify-content: space-between;
}

.roomListRoom__joined {
  color: var(--color-light-gray);
  display: flex;
  align-items: center;
  gap: 1rem;
  font-size: 1.4rem;
  font-weight: 500;
}

.roomListRoom__joined svg {
  fill: var(--color-main);
  width: 1.6rem;
  height: 1.6rem;
}

.roomListRoom__topic {
  padding: 5px 1.5rem;
  background-color: var(--color-dark-medium);
  border-radius: 5rem;
  font-weight: 500;
  font-size: 1.3rem;
}

/*=====
=>  Activities
=====*/

.activities {
  background: var(--color-dark);
  border-radius: 5px;
  overflow: hidden;
}

.activities__header h2 {
  background-color: var(--color-dark-light);
  text-transform: uppercase;
  font-weight: 500;
  padding: 1rem 1.5rem;
  color: var(--color-light);
  letter-spacing: 1px;
  font-size: 1.4rem;
}

.activities__box {
  margin: 1.5rem;
  padding: 1.5rem;
  border: 2px solid var(--color-dark-medium);
  border-radius: 5px;
}

.activities__boxHeader p {
  font-size: 1.4rem;
  line-height: 1.3;
}

.activities__boxHeader p span {
  color: var(--color-gray);
}

```

					ІАЛЦ.467200.007 Д4	Арк.
						37
Зм.	Арк.	№ докум.	Підпис	Дата		

```

font-size: 1.2rem;
display: block;
}

.activities__boxContent {
margin-left: 4.2rem;
}

.activities__boxContent {
font-size: 1.4rem;
}

.activities__boxContent a:hover {
text-decoration: underline;
}

.activities__boxRoomContent {
background: var(--color-bg);
padding: 1rem;
border-radius: 5px;
margin-top: 1rem;
margin-left: -4.2rem;
}

.roomListRoom__actions svg {
fill: var(--color-light-gray);
}

/*=====
=> Create Room
=====*/

.create-room.layout .layout__box {
max-width: 68rem;
}

/*=====
=> Update Account
=====*/

.update-account.layout .layout__box {
max-width: 68rem;
}

/*=====
=> Delete Item
=====*/

.delete-item.layout .layout__box {
max-width: 68rem;
}

/*=====
=> Auth
=====*/

.auth__tagline {
text-align: center;
margin-bottom: 3rem;
color: var(--color-main);
font-weight: 500;
font-size: 1.8rem;
}

.auth .layout__boxHeader {
text-align: center;

```

					ІАЛЦ.467200.007 Д4	Арк.
						38
Зм.	Арк.	№ докум.	Підпис	Дата		

```

    justify-content: center;
}

.auth__action {
    margin-top: 3rem;
    text-align: center;
}

/*=====
=>  Settings
=====*/

.settings__avatar {
    margin-bottom: 3rem;
    text-align: center;
    margin: 0 auto;
    display: flex;
    justify-content: center;
}

.settings__avatar .avatar {
    margin: 1rem;
}

/*=====
=>  Profile
=====*/

.profile {
    margin-bottom: 3rem;
}

.profile__avatar {
    text-align: center;
}

.profile__info {
    text-align: center;
}

.profile__info h3 {
    font-size: 2rem;
    color: var(--color-light);
    font-weight: 400;
}

.profile__info p {
    color: var(--color-main);
    font-weight: 500;
    margin-bottom: 1rem;
}

.profile__about {
    margin-top: 2rem;
}

.profile__about h3 {
    text-transform: uppercase;
    color: var(--color-dark-light);
    margin-bottom: 0.5rem;
}

.profile-page .roomList__header {
    margin-bottom: 1.5rem;
}

```

					ІАЛЦ.467200.007 Д4	Арк.
						39
Зм.	Арк.	№ докум.	Підпис	Дата		

```

.profile-page .roomList_header h2 {
  color: var(--color-dark-light);
}

/*=====
=> Room
=====*/

.room,
.participants {
  background: var(--color-dark);
  max-height: 87.5vh;
  border-radius: 0.7rem;
  overflow: hidden;
  position: relative;
}

@media screen and (max-width: 900px) {
  .participants {
    display: none;
  }

  .layout--2 > .container > div:first-child {
    flex-basis: 100%;
  }
}

.room__top,
.participants__top {
  background: var(--color-dark-light);
  display: flex;
  justify-content: space-between;
  align-items: center;
  padding: 1rem 2rem;
  position: relative;
}

.room__top svg,
.thread__top svg {
  width: 1.6rem;
  height: 1.6rem;
  fill: var(--color-light);
  cursor: pointer;
}

.room__topLeft {
  display: flex;
  align-items: flex-end;
  gap: 1rem;
}

.room__topLeft h3,
.participants__top {
  text-transform: uppercase;
  font-weight: 500;
  color: var(--color-light);
}

.room__topLeft svg {
  width: 1.6rem;
  height: 1.6rem;
  fill: var(--color-light);
}

.room__topRight {

```

					ІАЛЦ.467200.007 Д4	Арк.
						40
Зм.	Арк.	№ докум.	Підпис	Дата		

```

display: flex;
column-gap: 1em;
}

.room__topRight svg {
  fill: var(--color-main-light);
}

.room__header {
  max-height: 30vh;
  overflow-y: auto;
  position: absolute;
  width: 95%;
  background: var(--color-dark);
  z-index: 999;
  top: 4.4rem;
  padding-top: 2rem;
  padding-bottom: 1rem;
}

@media screen and (max-width: 500px) {
  .room__header {
    top: 4.3rem;
    padding-right: 2rem;
  }
}

.room__box {
  padding-left: 2rem;
  padding-right: 2rem;
  height: 80.5vh;
  /* overflow-y: auto; */
  padding-bottom: 0;
}

@media screen and (max-width: 500px) {
  .room__box {
    padding-left: 2.5rem;
    padding-right: 2rem;
    height: 80.5vh;
    overflow-y: auto;
    padding-bottom: 0;
  }
}

.room__info {
  display: flex;
  justify-content: space-between;
}

.room__info h3 {
  font-size: 2.4rem;
  font-weight: 500;
  color: var(--color-main);
}

.room__hosted p {
  text-transform: uppercase;
  color: var(--color-gray);
  font-size: 1.2rem;
  font-weight: 700;
  line-height: 2;
}

.room__author {

```

					ІАЛЦ.467200.007 Д4	Арк.
						41
Зм.	Арк.	№ докум.	Підпис	Дата		

```

display: flex;
gap: 1rem;
align-items: center;
margin-bottom: 1rem;
transition: all 0.3s ease-in-out;
}

.room__author:hover {
text-decoration: underline;
}

.room__topics {
padding: 0.5rem 1.5rem;
background: var(--color-dark-light);
color: var(--color-light);
display: inline-block;
font-size: 1.4rem;
border-radius: 1.5rem;
margin: 1rem 0;
}

.room__conversation {
margin-top: 1rem;
margin-bottom: 4rem;
height: 64%;
}

.threads h3 {
text-transform: uppercase;
font-weight: 500;
color: var(--color-gray);
}

.threads {
background: var(--color-bg);
border-radius: 0.7rem;
overflow-y: auto;
height: 100%;
margin-top: 28vh;
padding: 0 2rem 4rem 2rem;
}

.thread {
border-left: 2px solid var(--color-dark);
padding-left: 1rem;
margin: 2rem 0;
padding: 2rem;
}

.thread__top {
display: flex;
align-items: center;
justify-content: space-between;
}

.thread__top svg {
fill: var(--color-dark-light);
}

.thread__author {
display: flex;
align-items: center;
gap: 1.5rem;
font-size: 1.4rem;
}

```

					ІАЛЦ.467200.007 Д4	Арк.
						42
Зм.	Арк.	№ докум.	Підпис	Дата		

```

.thread__authorInfo {
  display: flex;
  align-items: center;
  gap: 1rem;
}

.thread__details {
  font-size: 1.4rem;
  margin-top: 0.5rem;
}

.room__message {
  padding: 2rem;
  position: absolute;
  z-index: 111;
  bottom: 0;
  left: 0;
  right: 0;
  background: transparent;
}

.room__message > form > input {
  resize: none;
  background-color: var(--color-dark-light);
  color: var(--color-light);
  border: none;
  outline: none;
  border-radius: 0.7rem;
  height: 4.5rem;
  width: 100%;
  margin-top: -1rem;
  padding: 1.2rem;
  font-size: 1.4rem;
  font-weight: 500;
  position: relative;
}

.room__message > form > input::placeholder {
  color: var(--color-light-gray);
}

.participants__top span {
  color: var(--color-main);
  font-size: 1.3rem;
  text-transform: none;
}

.participants__top {
  justify-content: flex-start;
  gap: 0.5rem;
}

.participants__list {
  padding: 2rem;
  height: 82.5vh;
  overflow-y: scroll;
  padding-bottom: 0;
}

.participant {
  display: flex;
  align-items: center;
  gap: 1.5rem;
  margin-bottom: 2rem;
}

```

					ІАЛЦ.467200.007 Д4	Арк.
						43
Зм.	Арк.	№ докум.	Підпис	Дата		

```

}

.participant p {
  color: var(--color-light-gray);
  line-height: 1.2;
}

.participant span {
  display: block;
  font-weight: 500;
  color: var(--color-main);
  font-weight: 1.4rem;
}

// // Actions:

// const closeButton = `Edit</a>
//   <form action="${deleteUrl}" method="delete">
//     <button type="submit">Delete</button>
//   </form>`;
//   button.insertAdjacentElement('afterend', popup);

```

					ІАЛЦ.467200.007 Д4	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		44

```

//    });
//  });
// }

// Menu

const dropdownMenu = document.querySelector(".dropdown-menu");
const dropdownButton = document.querySelector(".dropdown-button");

if (dropdownButton) {
  dropdownButton.addEventListener("click", () => {
    dropdownMenu.classList.toggle("show");
  });
}

// Upload Image
const photoInput = document.querySelector("#avatar");
const photoPreview = document.querySelector("#preview-avatar");
if (photoInput)
  photoInput.onchange = () => {
    const [file] = photoInput.files;
    if (file) {
      photoPreview.src = URL.createObjectURL(file);
    }
  };

// Scroll to Bottom
const conversationThread = document.querySelector(".room__box");
if (conversationThread) conversationThread.scrollTop = conversationThread.scrollHeight;

```

					ІАЛЦ.467200.007 Д4	Арк.
						45
Зм.	Арк.	№ докум.	Підпис	Дата		