НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
Кафедра автоматизованих систем обробки інформації та управління

УДК 004.93

«До захисту допущено»
В.о. завідувача кафедри
_____ *О.А.Павлов*
(підпис) (ініціали, прізвище)
“___”_____2019 р.

Дипломний проект
на здобуття ступеня бакалавра

з напряму підготовки *6.050101 «Комп’ютерні науки»*

на тему: « Система розпізнавання архітектурних стилів будівель за зображеннями»

Виконала: студентка *4* курсу, групи *ІС-52*
Новіченко Неля Валеріївна _____
(прізвище, ім’я, по батькові) (підпис)

Керівник *доц. Сперкач М.О.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з графічної документації *ст.викл. Халус О.А.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент *доц. каф. АУТС, к.т.н., доц. Писаренко А.В.* _____
(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 (-ка) _____
(підпис)

Київ – 2019 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет (інститут) *інформатики та обчислювальної техніки*
(повна назва)

Кафедра *автоматизованих систем обробки інформації та управління*
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки (програма професійного спрямування) *6.050101*

«Комп’ютерні науки» («Інформаційні управляючі системи та технології»)

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ *О.А. Павлов*
(підпис) (ініціали, прізвище)
“___”_____2019 р.

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЕКТ СТУДЕНТУ

Новіченко Нелі Валеріївни
(прізвище, ім’я, по батькові)

1. Тема проекту «Система розпізнавання архітектурних стилів будівель за зображеннями»

керівник проекту *Сперкач Майя Олегівна, доцент*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від “*23*”*квітня* 2019 р. №1181-с

2. Термін подання студентом проекту “*03*”*червня* 2019 року

3. Вихідні дані до проекту

Технічне завдання

4. Зміст пояснювальної записки

1. Загальні положення: основні визначення та терміни, опис предметного середовища, огляд ринку програмних продуктів, постановка задачі

2. Інформаційне забезпечення: вхідні дані, вихідні дані, опис структури бази даних

3. Математичне забезпечення: змістовна та математична постановки задачі, обґрунтування та опис методу розв’язання

4. Програмне та технічне забезпечення: засоби розробки, вимоги до технічного забезпечення, архітектура програмного забезпечення, побудова звітів

5. Технологічний розділ: керівництво користувача, методика випробувань програмного продукту

5. Перелік графічного матеріалу

1. Схема структурна діяльності

2. Схема структурна варіантів використань

3. Схема бази даних

4. Схема структурна компонентів програмного забезпечення

5. Схема структурна класів програмного забезпечення

6. Схема структурна послідовності

7. Креслення вигляду екранних форм

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання «*15*» *лютого 2019 року*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1.	*Вивчення рекомендованої літератури*	*20.02.2019*	
2.	*Аналіз існуючих методів розв'язання задачі*	*28.02.2019*	
3.	*Постановка та формалізація задачі*	*10.03.2019*	
4.	*Розробка інформаційного забезпечення*	*25.03.2019*	
5.	*Алгоритмізація задачі*	*01.04.2019*	
6.	*Обґрунтування використовуваних технічних засобів*	*16.04.2019*	
7.	*Розробка програмного забезпечення*	*21.04.2019*	
8.	*Налагодження програми*	*05.05.2019*	
9.	*Виконання графічних документів*	*08.05.2019*	
10.	*Оформлення пояснювальної записки*	*10.05.2019*	
11.	*Подання ДП на попередній захист*	*30.05.2019*	
12.	*Подання ДП на основний захист*	*03.06.2019*	
13.	*Подання ДП рецензенту*	*05.06.2019*	

Студент _____ Н.В. Новіченко
(підпис)

Керівник проекту _____ М.О. Сперкач
(підпис)

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
			Документи текстові		
1	А4	ДП ІС-5219.1181-с.ПЗ	Пояснювальна записка	82	
2	А4	ДП ІС-5219.1181-с.ТЗ	Технічне завдання	9	
3	А4	ДП ІС-5219.1181-с.ПМВ	Програма та методика випробувань	14	
			Матеріали графічні		
4	А4	ДП ІС-5219.1181-с.ССД	Схема структурна діяльності	1	
5	А4	ДП ІС-5219.1181-с.ССВ	Схема структурна варіантів використань	1	
6	А4	ДП ІС-5219.1181-с.СБД	Схема бази даних	1	
7	А4	ДП ІС-5219.1181-с.ССК	Схема структура компонентів	1	
			програмного забезпечення		
8	А4	ДП ІС-5219.1181-с.ССК	Схема структурна класів програмного	1	
			забезпечення		
9	А3	ДП ІС-5219.1181-с.ССП	Схема структурна послідовності	2	
10	А3	ДП ІС-5219.1181-с.КЕ	Креслення вигляду екранних форм	2	

Зм.	Арк.	ПІП	Підп.	Дата	ДП ІС-5219.1181-с.ВДП		
Розробн.		Новіченко Н.В.			Відомість дипломного проекту	Літ.	Лист 1 / Листів 1
Керівн.		Сперкач М.О.					
Консульт.						КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-52	
Н/контр.		Халус О.А.					
В.о.зав.каф.		Павлов О.А.					

Пояснювальна записка
до дипломного проекту

на тему: Система розпізнавання архітектурних стилів будівель за зображеннями

Київ – 2019 року

АНОТАЦІЯ

Структура та обсяг роботи. Пояснювальна записка дипломного проекту складається з шести розділів, містить 18 рисунків, 21 таблиць, 1 додаткок, 17 джерел.

Дипломний проект присвячений розробці системи класифікації зображень з метою визначення архітектурних стилів будівель.

В дипломному проекті розглянуті методи класифікації цифрових зображень, засновані на машинному навчанні за допомогою нейронних мереж. Система вирішує задачу документування культурної спадщини та дозволяє зменшити помилки при визначенні архітектурного стилю.

У розділі загальні положення описано предметне середовище, процес діяльності та опис функціональної моделі системи. Також у розділі описано порівняння системи з наявними аналогами та описані мета та призначення розробки системи.

У розділі з інформаційного забезпечення були визначені дані для навчання системи, вхідні та вихідні дані до комплексу задач, були розроблені вимоги до зображень для аналізу, що відповідають поставленим цілям проекту.

Розділ математичного забезпечення присвячений обґрунтуванню обраного підходу навчання системи, що дозволив збільшити точність результатів.

Розділ програмного забезпечення описує основні засоби розробки комплексу задач, висунуті вимоги до технічного забезпечення. В цьому розділі обрано та обґрунтовано архітектуру програмного забезпечення.

У технологічному розділі описана інструкція користувача та проведене тестування комплексу задач.

МАШИННЕ НАВЧАННЯ, НЕЙРОННІ МЕРЕЖІ, КЛАСИФІКАЦІЯ ЗОБРАЖЕНЬ, АРХІТЕКТУРНІ СТИЛІ

ABSTRACT

Structure and scope of work. Diploma project consists of six sections, contains 18 drawings, 21 tables, 1 applications, 17 sources.

The diploma project is devoted to the development of tasks for the classification of images in order to determine the architectural styles of the buildings. Automatic methods for the classification of images during the analysis of architectural objects solve the problem of documenting cultural heritage and significantly reduce mistakes in sorting: usually a large number of images are processed and this is a tedious task, the process of classification by experts is prone to errors and takes a lot of time. The correct classification allows to study and analyze cultural heritage more effectively.

In the diploma project were considered methods of classification of digital images, based on machine learning with the help of neural networks.

The section on information provision define the data for training neural network, input and output data to a set of tasks, requirements for images for analysis, which corresponds to the set objectives of the project.

The section of mathematical support is devoted to substantiation of the chosen approach of training the system, which allows to increase the accuracy of the results.

The software section describes the main tools for developing a set of tasks, the requirements for technical support. This section defines and justifies the software architecture.

The technology section describes the user's manual and tests a set of tasks.

MACHINE LEARNING, NEURAL NETWORKS, IMAGE CLASSIFICATION, ARCHITECTURAL STYLES.

ЗМІСТ

ВСТУП

У наш час збереження архітектурної спадщини є актуальним питанням, адже кожен народ пишається культурними здобутками своїх предків і намагається зберегти твори архітектури минулого і сучасності. Саме в архітектурі відображаються ідеї та цінності свого часу, вона носить соціокультурну цінність та має економічний вплив (наприклад, туризм). Стихійні лиха, забруднення навколишнього середовища, зміна клімату, природна деградація або ж вандалізм загрожують збереженню архітектурних споруд. Проте в час швидкого розвитку інформаційних систем людство отримало можливість зберегти дані про архітектурну спадщину.

Існують міжнародні організації, такі як CIPA (Comité International de la Photogrammétrie Architecturale – Міжнародний комітет архітектурної фотограмметрії) та ISPRS (International Society for Photogrammetry and Remote Sensing), що спеціалізуються на технологіях вимірювання та візуалізації документації та збереження спадщини. Їх завданням є дослідження еволюції архітектури, моніторинг і нагляд, виявлення патологій і порушень, комп'ютерна допомога відновлення споруд, застосування методів віртуальної та розширеної реальності, цифрових каталогів, інтеграції в географічних інформаційних системах (ГІС) і в інформаційному моделюванні будівель (BIM), поширення і багато іншого [1].

Швидкий аналіз параметрів будівель є важливим завданням для досягнення цілей таких організацій. На жаль, підхід до цифрової документації у сфері культурної спадщини ще не використовують. Саме сучасні технології можуть бути потужним інструментом, що допоможе покращити класичний стандарт вимірювання та документування спадщини та може створити нові методології.

Збереження та підтримка належного стану архітектурних споруд неможливе без документування архітектурної спадщини, адже саме документація дозволяє відстежити еволюцію та зміни стану будівлі. Документація може бути представлена в будь-якій формі:

– 3D-моделі;
– фотографії;
– термографи;
– мультиспектральні зображення;
– історичні документи.

Найрозповсюдженішими в наш час є фотографії, креслення та історичні документи.

Для реконструкції та відновлення споруджень вкрай необхідним постає визначення архітектурного стилю будівлі, що має бути внесений до опису будівлі при її документуванні. Цей процес вимагає від спеціалістів аналізу великої кількості даних та допускає можливість помилки.

Взагалі, архітектурний стиль — сукупність основних рис та ознак архітектури певного історичного часу і місця, яка проявляється у функціональних, конструктивних, мистецьких особливостях будов. Архітектурний стиль несе в собі інформацію про конструкцію, архітектурну форму, використанні будівельних та обробних матеріалів і період будівництва споруди, саме тому саме ця характеристика є важливим параметром і має бути визначена точно [1].

Метою проекту є спростити процес визначення архітектурного стилю будівлі та знизити похибку при визначенні стилю за рахунок аналізу зображень отриманих із звичайних цифрових камер або камери телефону, що посприяє збереженню, підтримці та відновленню будівель і є вкрай важливим на сьогоднішній день.

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Опис предметного середовища

В архітектурі виділяють більше 25 архітектурних стилів будівель. Архітектурний стиль — сукупність основних рис та ознак архітектури певного історичного часу і місця, яка проявляється у функціональних, конструктивних, мистецьких особливостях будов. Саме архітектура – це одночасно наука і мистецтво проектування будівель, а також власне система будівель та споруд, які формують просторове середовище для життя і діяльності людей відповідно до законів краси. Вона визначає ідейну спрямованість, масштаб, техніку виконання та загальні принципи композиції скульптури, живопису, елементів декоративно-прикладного мистецтва, що є компонентами органічного рішення загальної художньої завдання. Архітектурний стиль відрізняється власними стійкими художніми формами, описує похідну епоху і визначає культурне обличчя епохи та країни у певний період.

Наприклад, раціоналізм – напрямок в архітектурі 20 ст., намагається не тільки відобразити в архітектурі ті чи інші життєві процеси, а й активно впливати на ці процеси «винайти» для них нові форми. Характеризується цей стиль естетизацією сучасної техніки (що схоже на техніцизм), пошуки виразності простих геометричних форм, відмова від декору, інтерес до пропорціям, до кольору, до синтезу архітектури з іншими мистецтвами. Зовнішні ознаки – геометрична простота форм, асиметрія, мінімалізм у декору, використання металу, залізобетону та скла. Раціоналізм – спрощеність і стандарт.

Найбільш поширеними архітектурними стилями є:

- античная архітектура – поєднання грецького ордера, італійської арки і циліндричного зводу;
- романський стиль – рельєфна площина, лаконічність; основні кольори – коричневий, червоний, зелений, білий, сірий, чорний; дуги, прямі, горизонтальні і вертикальні лінії; поширені повторювані геометричні або рослинні малюнки та зали з відкритими стельовими балками і опорами по центру, присутня замкова і лицарська тематика – смолоскипи, обладунки, герби, бої, зброї;
- готика – каркасна система, стрілчасті арки, вузькі і високі вежі і колони, багато прикрашений фасад з різьбленими деталями; для будівництва використовується камінь;
- відродження – горизонтальні архітектурні композиції, чіткість та раціональність планів, фасадів та інтер'єрів ренесансних будівель; основні геометричні фігури – квадрат, прямокутник, куб і куля;
- бароко – контрасти і суперечливість, пишність, химерність, театральність і нагромадження деталей;
- класицизм – строгість, логічність ясності і монументальність, звернення до форм античного, властиві симетрично-осьові композиції, стриманість декоративного оздоблення;
- модерн – відмова від прямих ліній і кутів на користь більш природних ліній, використання нових технологій – метал, скло, прагнення до створення естетично красивих і функціональних будівель; конструктивні елементи такі як сходи, двері, стовпи, балкони художньо оброблені;
- постмодернізм – раціональні, вільні, симетричні і асиметричні форми; легкі конструкції, просторі, іноді на всю стіну, розсувні, відкидні або ж поворотні вікна; телескопічні, розсувні, складчасті

двері; характерна бежева гамма, сріблястий, «металік», перламутровий та флуоресцентний кольори [2].

1.1.1 Опис процесу діяльності

У роботі буде йти мова про розпізнавання архітектурних стилів будівель за зображенням. З розвитком технологій комп'ютерного зору та постійно зростаючу кількість цифрової документації з'явилася необхідність у автоматизації процесу аналізу інформації. Робота присвячена аналізу різного формату зображень, отриманих з різних цифрових камер.

При документуванні архітектурної спадщини зберігаються зображення будівель, процес опису може бути автоматизований: експерту достатньо завантажити чітке зображення будівлі та зробити аналіз отриманих даних. Дотепер, для проведення подібного процесу необхідно було долучити групу експертів, що розглядали опис будівлі архітектором, якщо такий наявний та робили висновки.

Для вирішення проблеми пропонується створити сервіс, в основі якого будуть використані технології машинного зору – машинне навчання. Необхідно створити алгоритм, за допомогою якого машина могла б розрізнити різні архітектурні стилі по зображенню. Для цього буде використано такий метод машинного навчання як навчання по прецедентах, а саме контрольоване навчання або ж навчання з учителем. Суть полягає у тому, що на вхід алгоритму подається велика кількість фотографій з маркерами – описами стилів будівель. Машина сама вибирає ознаки, за якими вона відрізняє архітектурні стилі між собою.

У процесі вирішенні поставленої задачі можна виділити такі етапи:

– створення вибірки великої кількості зображень для навчання;
– маркування вибірки – аналіз архітектурного стилю кожного зображення будівлі;

– навчання алгоритму та коригування похибок;
– створення сервісу для підготовки вхідного зображення до аналізу;
– тестування алгоритму.

Перші 2 етапи є найбільш важливими, адже саме від якості даних буде залежать результат навчання.

Користувач системи отримує результат аналізу за допомогою заздалегідь навченого алгоритму. Розглянемо типові дії, що має виконати користувач для аналізу зображення та збереження результатів до системи за допомогою схеми діяльності, представлену у частині графічного матеріалу.

1.1.2 Опис функціональної моделі

Спроектуємо функціональну модель системи, де визначимо дійових осіб (акторів) та дії, які вони можуть виконувати в рамках системи.

У системі розпізнавання архітектурних стилів будівель функціонує лише один актор – користувач.

Користувач може проводити аналіз зображення, збереження отриманих результатів та перегляд збережених результатів.

Розглянемо дії, що може виконати користувач системи, представленні на схемі варіантів використання, представлену у частині графічного матеріалу.

Розглянемо детальніше деякі з варіантів використання:

– перегляд документації: користувач має доступ до детального опису можливостей системи та до інструкції, як інтерпретувати отримані результати; користувач може ознайомитися з переліком архітектурних стилів, що були розглянуті при навчанні системи;
– аналіз архітектурного стилю будівлі: на сторінці веб-застосування користувач має можливість завантажити зображення будівлі з власного комп’ютера, відправити зображення на аналіз та отримати

результати у вигляді в відсоткового відношення даного зображення до певного архітектурного стилю; якщо користувач увійшов в системи в свій обліковий запис, він має можливість зберегти результати аналізу до бази, що дає можливість проводити масовий аналіз зображень з інтерпретацією результатів після завершення;

– перегляд збережених результатів: авторизований користувач має можливість перегляду попередніх збережених результатів, сортувати картинки за стилями та датою аналізу.

Отже, користувач системи, що має обліковий запис, отримує ширший функціонал ніж гість.

1.2 Огляд наявних аналогів

У ході пошуку схожих вирішень не було виявлено застосувань з схожою функціональністю. Проте можна порівняти різні підходи до розробки алгоритму класифікації. Розглянемо деякі з них детальніше:

– алгоритм, заснований на згорткових нейронних мережах (CNNs) розроблений Кріс Песто: розроблений для вирішення питання класифікації будинків Америки та спроможній розрізняти стилі, притаманні для архітектури американських будинків. Вибірка для навчання засновувалась на 5 поширених архітектурних стилів у США [3];

– робота Гаяне Шалунц, Юлл Хашимуса і Роберт Саблатинга «Класифікація архітектурного стилю фасаду будівлі» без використання машинного навчання: алгоритм використовує підхід на основі градієнтних напрямків з використанням SVM на 4 класи для класифікації архітектурного стилю з особливим фокусом на «зворотному процесуальному моделюванні» – використовуючи

зображення для створення генеративної процедурної моделі для 3D-графіки [6];

– «Класифікація зображень архітектурної спадщини методами глибокого навчання» виконана Хосе Ламасом, Педро М. Леронесом та іншими, основною метою поставили оцінку застосовності різних методів класифікації зображень архітектурної спадщини. В результаті даної роботи були отримані різні навчені нейронні сітки, спроможні розрізняти різні частини будівель (колони, горгульї, дзвіниці, тощо) [7].

1.3 Постановка задачі

1.3.1 Призначення розробки

Розробка призначена для визначення архітектурних стилів будівель за зображенням різного формату та якості.

1.3.2 Мета та задачі розробки

Метою розробки є спрощення процесу визначення архітектурного стилю будівлі та зниження похибки при визначенні стилю за рахунок аналізу отриманого зображення.

Завдяки створеній системі аналіз архітектурної спадщини та її документування може бути автоматизоване, що дуже важливо для зменшення часу на документування та підвищення достовірності. Це посприяє збереженню, підтримці в належному стані та відновленню будівель за рахунок можливості дослідження еволюції архітектурної споруди. Робота присвячена також підтримці аналізу зображень різного формату, адже в наш час основним джерелом інформації про будівлі будуть зображення – швидкий спосіб передачі інформації. Користувачі можуть отримувати

зображення з різних камер, що відрізняються певними характеристиками: формат (лінійний або матричний), оптика (фіксований або взаємозамінний), розмір (повний кадр, середній формат, чотири третини тощо), видошукач (рефлекс або беззеркальний), за сегментами (споживчим або професійним), обсяг (промисловий чи ні). У даному випадку, навіть найпростіші камери зазвичай мають достатню якість і можуть бути використані для вирішення завдання.

Для досягнення поставленої мети мають бути вирішенні такі задачі:

- створення достатньо великої та точної вибірки зображень та класифікація кожного зображення до певного стилю;
- розробка алгоритму, здатного с високою точністю класифікувати зображення (підбір типу нейронних сіток, що дасть найкращий результат для поставленої задачі);
- підтримка аналізу зображень різного розміру та якості;
- навчання нейронної мережі на створеній вибірці;
- розробка веб-застосування, що дозволить завантажувати зображення для аналізу;
- розробка веб-застосування, що дозволить зберігати результати аналізу до бази для задокументованої інформації про будівлі;
- підтримка можливості збереження результатів аналізу та перегляд еволюції зміни будівлі;
- створення документації та інструкції для користувачів системи щодо інтерпретації отриманих результатів.

Висновок до розділу

У даному розділі було розглянуто предметне середовище, короткий опис архітектурних стилів будівель, їх відмінності між собою та особливості, було наведено приклади зображень для найбільш поширених архітектурних

стилів. Дана робота присвячена аналізу зображень архітектурних споруд для документування архітектурної спадщини, з використанням зображень різної якості та формату, отриманих з цифрових камер. При описі предметного середовища було визначено та описано процес діяльності, що наведений у структурній схемі діяльності. Іншим результатом є створення функціональної моделі системи: опис користувачів системи, їх ролі та можливі дії у системі. Функціональна модель представлена у вигляді структурної схеми варіантів використання.

Було сформовано постановку задачі аналізу архітектурних стилів та визначено мету розробки – зменшення часу на документування та підвищення достовірності. Це посприяє збереженню, підтримці в належному стані та відновленню будівель за рахунок можливості дослідження еволюції архітектурної споруди. Для досягнення поставленої мети необхідно вирішити ряд задач, що описані вище у розділі.

Результатом цього розділу також є огляд наявних аналогів програмного забезпечення та порівняння їх між собою. На даний момент систем зі схожим функціоналом не було виявлено. Було проведено аналіз та порівняння існуючих алгоритмів для вирішення задачі розпізнавання архітектурних стилів будівель та описано їх особливості.

2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Вхідні дані

Вхідні дані для задачі визначення архітектурного стилю будівлі можна розділити на дві категорії:

- вхідні дані для створення системи (дата сет для навчання системи);
- вхідні дані користувача (зображення для аналізу, електронна адреса та ім’я для створення облікового запису та інформація про зображення).

Розглянемо детально кожну з категорій.

Для навчання нейронних мереж необхідна велика кількість зображень, хоча показано, що можливо оптимізувати структуру нейронної сітки так, щоб використовувати менші набори даних. У будь-якому випадку доцільно використовувати набори даних, які вільно доступні для відтворення експериментів і перевірки отриманих результатів [3].

Хоча існує багато загальнодоступних банків зображень, зображень з тегами, все одно задача пошуку конкретних наборів даних архітектурної спадщини, готових до використання, не може бути вирішена так просто. Для навчання нейронної мережі був створений набір з близько 2 000 зображень, класифікованих у 25 типах архітектурних елементів.

Розглянемо вибірку зображень різних архітектурних стилів, що були виділені для навчання нейронної мережі:

- achaemenid – 42 зображення;
- american craftsman – 195 зображень;
- american foursquare – 59 зображень;
- ancient egyptian – 256 зображень;

- art Deco – 366 зображень;
- art Nouveau – 450 зображень;
- baroque – 239 зображень;
- bauhaus – 92 зображення;
- beaux Arts – 191 зображення;
- byzantine – 111 зображень;
- chicago School – 153 зображення;
- colonial – 177 зображень;
- deconstructivism – 213 зображень;
- edwardian – 79 зображень;
- georgian – 154 зображення;
- gothic – 109 зображень;
- greek Revival – 327 зображень;
- Iinternational – 207 зображень;
- novelty – 212 зображень;
- palladian – 113 зображень;
- postmodern – 163 зображення;
- queen Anne – 425 зображень;
- romanesque – 107 зображень;
- russian Revival – 165 зображень;
- tudor Revival – 162 зображення.

Вибірка охоплює архітектурні споруди різного часу та призначення, що дозволить отримати алгоритм, спроможній аналізувати споруди не тільки сучасної архітектури, що полегшить задачу документування архітектурної спадщини. Вибірка включає в себе архітектурні споруди давнього часу, що мають особливий зовнішній вигляд (такі споруди, як піраміди Єгипту), достатньо велику кількість храмів, збудованих у різний період часу та

приклади «дивних» будинків архітектурних стилів деконструктивізм та постмодерну. Усі використані зображення наявні у вільному доступі та були отримані в основному з дата сету, створеного Dr. Zhe XU, DataHub, фотосховища Flickr та Вікісховища [4].

Вхідні дані системи визначення архітектурних стилів будівель включають в себе:

– зображення, завантажене користувачем;
– не обов’язкові дані: назва зображення, опис, дата створення зображення;
– дані користувача для створення облікового запису: електронна адреса та ім’я.

Цих вхідних даних достатньо для створення сховища документованих архітектурних споруд.

2.2 Вихідні дані

Вихідними даними аналізу зображення є об’єкт, що містить інформацію у частках від одиниці про відношення будівлі на зображені до певного архітектурного стилю. Даний формат дозволяє оцінити у відсотках ймовірність відношення будівлі до певного стилю.

Сукупність таких результатів являє собою сховище для будівель, яку користувач системи може використовувати для документування архітектурної спадщини.

2.3 Опис структури бази даних

Розглянемо структуру бази даних для збереження результатів аналізу, представлену у частині графічного матеріалу.

Розглянемо призначення кожної таблиці:

– result – таблиця для збереження результатів аналізу, містить опис аналізу, дату аналізу та має зв’язки з таблицями user, image та style;
– user – таблиця з інформацією про користувача системи;
– image – містить унікальний ідентифікатор зображення та двійкове представлення зображення;
– style – системна таблиця, що містить наявні в системі архітектурні стилі.

Розглянемо детальніше структуру таблиць бази даних та призначення кожного поля в таблицях (таблиці 2.1 – 2.4).

Таблиця 2.1 – Опис таблиці Result

Код	**Опис**	**Тип даних**	**Довжина**	**Первинний**	**Обов’язкове**
id	Унікальний ідентифікатор запису у таблиці	INT		X	X
title	Назва аналізу	NVARCHAR	150		X
description	Опис аналізу	NVARCHAR	MAX		
date	Дата проведення аналізу	DATE			X
styleId	Посилання на архітектурний стиль	INT			X
userId	Посилання на користувача	INT			X
imageId	Посилання на збережене зображення	INT			X

Таблиця 2.2 – Опис таблиці Image

Код	**Опис**	**Тип даних**	**Довжина**	**Первинний**	**Обов'язкове**
id	Унікальний ідентифікатор запису у таблиці	INT		X	X
date	Дата створення зображення	DATE			
name	Назва зображення	NVARCHAR	100		
format	Формат зображення	NVARCHAR	50		
comment	Примітки користувача	NVARCHAR	1000		

Таблиця 2.3 – Опис таблиці Style

Код	**Опис**	**Тип даних**	**Довжина**	**Первинний**	**Обов'язкове**
id	Унікальний ідентифікатор запису у таблиці	INT		X	X
name	Назва архітектурного стилю	NVARCHAR	100		X
comment	Опис архітектурного стилю	NVARCHAR	500		

Таблиця 2.4 – Опис таблиці User

Код	**Опис**	**Тип даних**	**Довжина**	**Первинний**	**Обов'язкове**
id	Унікальний ідентифікатор запису у таблиці	INT		X	X
firstName	Зображення у бінарному вигляді	NVARCHAR	100		X
lastName	Дата створення зображення	NVARCHAR	100		X
email	Назва зображення	NVARCHAR	100		X
identityId	Формат зображення	INT			X

У таблиці Image зберігається мета-інформація про зображення, але файл зображення зберігається у файловій системі серверу. Цей підхід було обрано спираючись на результат після великої кількості тестів та аналізів, проведених компанією Microsoft: якщо зображення або документи зазвичай мають розмір нижче 256 КБ, зберігання їх у стовпці VARBINARY у базі даних є більш ефективним, якщо розмір зображень або документів зазвичай перевищує 1 МБ, зберігання їх у файловій системі є більш ефективним [5].

Автори рекомендують використовувати окрему таблицю для зберігання мета-інформації зображень – тобто не зберігати зображення будівлі в таблиці результатів, а зберігайте їх у окремій таблиці. Таким чином, таблиця Result може залишатися «легкою» і ефективною.

Дана реалізація дозволить веб-застосуванню завантажувати зображення окремо від мета-інформації і за вимогою, що дає більшу гнучкість. Так як зображення є публічним ресурсом, сервер зможе віддавати його за посиланням будь-якому клієнту.

2.4 Структура масивів інформації

При навчанні нейронної мережі було сформовано model.json файл. Його призначення – інтерпретація результатів, що отримано на виході нейронної мережі до предметної області.

Структура файлу – це пари «число – рядок» що дозволяють з отриманого на виході числа дізнатися, до якого архітектурного стилю відноситься зображення.

```
{
   "6" : "baroque",
   "7" : "bauhaus",
   "23" : "russianRevival",
   "21" : "queenAnne",
   "3" : "ancientEgyptian",
   "20" : "postmodern",
   "11" : "colonial",
   "4" : "artDeco",
   "24" : "tudorRevival",
   "22" : "romanesque",
   "17" : "international",
   "12" : "deconstructivism",
   "19" : "palladian",
   "15" : "gothic",
   "8" : "beauxArts",
   "9" : "byzantine",
   "14" : "georgian",
   "13" : "edwardian",
   "18" : "novelty",
   "0" : "achaemenid",
   "2" : "americanFoursquare",
   "5" : "artNouveau",
   "10" : "chicagoSchool",
   "1" : "americanCraftsman",
   "16" : "greekRevival"
}
```

Рисунок 2.1 – Вміст файлу опису стилів

Висновок до розділу

У розділі інформаційного забезпечення було описано вхідні дані, що були використані для створення системи та вхідні дані користувача. При створенні алгоритму розпізнавання архітектурних стилів було використано нейронну мережу, що була навчена на великій вибірці зображень, що відносяться до різних архітектурних стилів. У розділі вхідних даних описано джерела зображень, що були використані для створення вибірки а також яка кількість зображень певного архітектурного стилю була включена до вибірки. Також у розділі вхідні дані описана вхідна інформація від користувача, що необхідна для користування системою.

Вихідними даними є результати аналізу та база даних результатів.

У розділі було описано структуру зберігання даних у базі даних та представлена схема бази даних. Було наведено опис кожної таблиці та призначення кожного стовбця.

Крім бази даних у системі наявні інші структури масивів інформації, що детально описані у розділі структур масивів інформації. Файл типу json містить інформацію про архітектурні стилі, що може розпізнати навчений алгоритм, та відповідність кожного стилю певному числовому значенню, з яким працює алгоритм.

3 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Змістовна постановка задачі

Існує вибірка великого розміру з зображеннями будівель різних архітектурних стилів та заданий перелік цих стилів. Зображення мають різний розмір та якість. Кожне зображення має ярлик – назва архітектурного стилю. Необхідно на основі навчаючої вибірки створити алгоритм, що спроможній за вхідним зображенням класифікувати будівлю до одного з заданих архітектурних стилів.

3.2 Математична постановка задачі

Розглянемо базову нейронну мережу та її архітектуру. Нейронна мережа являє собою взаємопов'язану збірку простих елементів обробки, об’єднань вузлів або індивідуальних вузлів, чия поведінка моделює поведінку нейронів людини. Саме тому вузли називають нейронами. Здатність мережі оброблювати дані зберігається в вагових коефіцієнтах зв’язків між нейронами мережі, отриманих в результаті процесу адаптації або навчання на наборі шаблонів. Структура нейрону зображена на рисунку 3.1 [8].

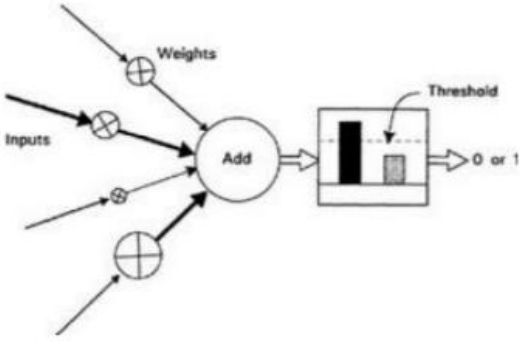


Рисунок 3.1 – Схема просто нейрону

Нейронна мережа складається з довільної кількості шарів, що обєднують нейрони. Всі нейрони мають звязки с нейронами з попереднього і наступного шарів. Перший шар є вхідним шаром, кількість нейронів на ньому – кількість вхідних параметрів. Останній шар є вихідним шаром, кількість нейронів на якому відповідає кількісті можливих вихідних даних. Інші шари між вхідним і вихідним шарами називають прихованим шаром, нейрони якого будуть обробляти і передавати "повідомлення" від попереднього шару до наступного. Кожен нейрон отримає деякі входи від нейронів у попередньому шарі, виконує обробку повідомлення, отримуючи значення порогової функції для вхідного значення, та передає нейронам з наступного шару [9].

Навчання нейронної мережі полягає у коригуванні ваг та порогової функції за допомогою ітераційного процесу. Розглянемо загальний процес навчання нейронної мережі:

Крок 1: Ініціалізація всіх ваг $w_{ij}^{(l)}$ звязків нейронної мережі, де $w_{ij}^{(l)}$ – вага звязку від нейрону *i* у *(l – 1)* шарі до нейрону *j* у шарі *l*.

Крок 2: Виконати наступні дії:

Крок 2.1: Беремо одні дані з навчаючої вибірки і встановлюємо значення вхідного шару мережі (шар 0) у значення даних та значення результату у вихідний шар.

Крок 2.2: Розраувати значення для входу для кожного нейрону у наступному прихованому шарі $x_j^{(l)}$ та застосувати функцію активації, щоб отримати віхідне значення для наступного прихованого шару. Повторити для всіх наступних шарів:

$$X^{(l)} = Y^{(l-1)}W^{(l-1)}, \qquad W^{(l-1)} \in \mathrm{R}(\mathrm{d}(l-1) \times \mathrm{d}(l)),$$

$$Y^{(l)} = \sigma\left(X^{(l)}\right), X^{(l)} \; and \; Y^{(l)} \in \mathrm{R}(\mathrm{d}(l) \times 1),$$

в яких $\mathrm{d}(l)$ означає кількість нейронів в шарі l.

Крок 3: Зворотний прогін $W^{(l)} = W^{(l)} - \eta dW^{(l)}$

Крок 3.1: Розрахувати матрицю градієнтів dW для ваг з предостаннього шару до останнього – вихідного шару *(l = n)*. Нехай похибка обчислень $E = \frac{1}{2}(label_j^{(n)} - y_i^{(n)})^2$, тоді градієнт кожного вага між останнім та передостаннім шаром:

$$\delta^{(n)} = -(E - Y^{(n)})\sigma'(X^{(n)}), \qquad \delta^{(n)} \in R(d(n) \times 1),$$

$$dW^{(n-1)} = (Y^{(n-1)})^T \delta^{(n)}$$

Крок 3.2: Розрахувати значення градієнту на попередніх шарах (*l = n – 1*):

$$\delta^{(l)} = W^{(l)}(\delta^{(l+1)}),$$

$$dW^{(l-1)} = (Y^{(l-1)})^T \delta^{(l)}$$

Повторюмо процес доки всі ваги нейронної мережі не будуть оновлені.

Крок 4: Якщо ще не виконали задану кількість ітерацій – перейти на шаг 2. Інакше – завершити процес навчання.

Для вирішення задачі класифікації будемо використовувати згорткову нейронну мережу. Згорткові нейронні мережі (CNN) – це особлива архітектура штучних нейронних мереж, що заснована на механізмах зорової кори, саме тому цю архітектуру використовують для класифікації зображень. Алгоритм CNN включає в себе 2 основних процеса: згортка та обєднання.

Розглянемо процес більш детально: зображення передається через ряд згорткових, нелінійних об'єднуючих шарів (шарів спрощення) і повністю пов'язаних шарів, а потім генерує вихід.

На шар згортки завантажується зображення (матриця з пікселів розміру n × n). Далі вибирається матрицю, що є частиною загальної матриці, яку називають фільтром (або нейроном, або ядром) за наступним принципом:

$$x_{ij}^{(l)} = \sigma(\mathrm{b} + \sum_{r=0}^{n}\sum_{c=0}^{n} W_{r,c}x_{i+r,j+c}^{(l-1)})$$

Потім з цієї матриці отримується згортка шляхом множення значень фільтру на початкові значення «пікселів». Ці значення підсумовуються, в результаті чого отримують одне число (рисунок 3.2).

Ця операція повторюється для кожного фільтру, що отримується переміщенням вбік (або вниз) 1 одиницю, виконуючи подібну операцію. Після проходження фільтра по всіх позиціях отримують матрицю, але меншу, ніж вхідну матрицю.

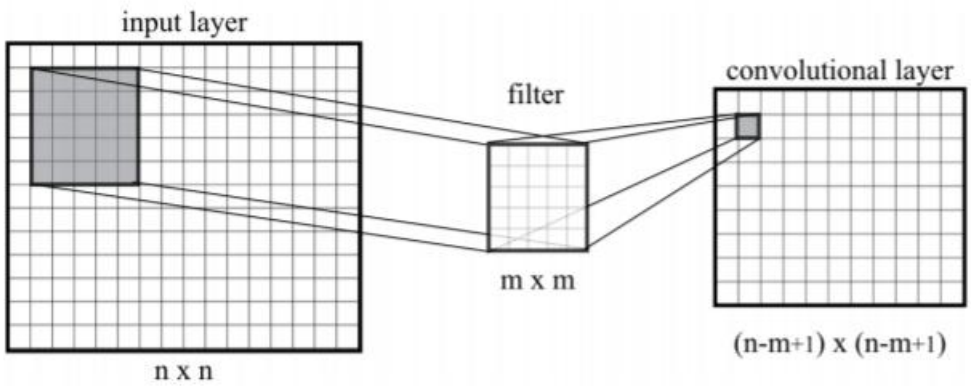


Рисунок 3.2 – Процес згортки

Нейронна мережа складається з декількох згорткових мереж, поєднаних з обєднуючими шарами. Обєднуючі шари завжди йдуть після згорткових та мають функцію активації, що «виділяє» кордони на зображенні і робить матрицю більш «контрастною». Зазвичай використовують стандарну функцію активації наступного вигляду:

$$y(x) = (1 + e^{-x})^{-1},$$

або

$$y(x) = \tanh(x).$$

Ця операція аналогічна то процесу виділення людиною кордонів та меж предметів на зображенні. Процес обєднання(спрощення) зображений на рисунку 3.3.

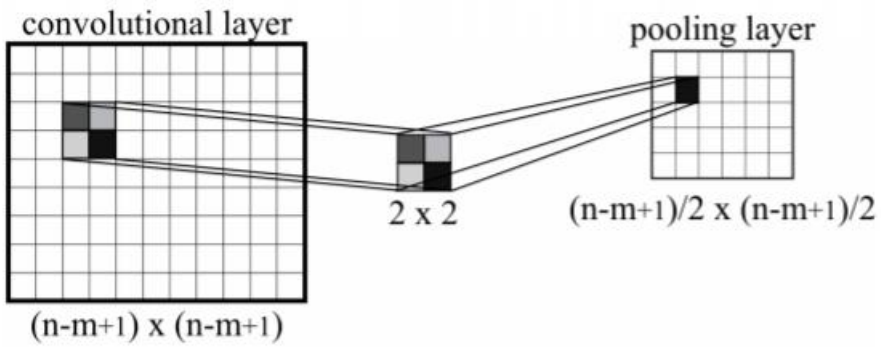


Рисунок 3.3 – Процес обєднання(спрощення)

Обєднуючий шар дозволяє коригувати розміри матриці пікселів – зменшувати розмірність стовбців та рядків.

Після прогону матриці на вище описаних шарах вона готова до аналізу нейронною мережою, тому що зображення достатньо зменшене в розмірах. Матриця потрапляє на «fully connected layer», де відбувається проекція матриці на вектор вхідного шару [9][10].

Саме така структура дозволяє працювати з зображеннями різних розмірів та якості.

3.3 Обґрунтування методу розв’язання

До того, як CNN набули популярності, для класифікації зображень використовували наступні методи: розробники виділяли певні «особливості» на зображеннях – певні об’єкти і саме на цих об’єктах виконувалася класифікація за допомогою таких алгоритмів, як SVM. Деякі алгоритми працювали на основі значень рівнів пікселів, як виділених об’єктах.

CNN можна розглядати як автоматичні екстрактори об'єктів із зображення за допомогою своєї структури. Порівняно з SVM алгоритмом, CNN активно використовує інформацію про сусідні пікселі, що дозволяє більш ефективно спрощувати зображення, чого не дозволяє зробити SVM.

3.4 Опис методів розв'язання

Розглянемо загальну спрощену модель CNN з входом 32×32×3 [18].

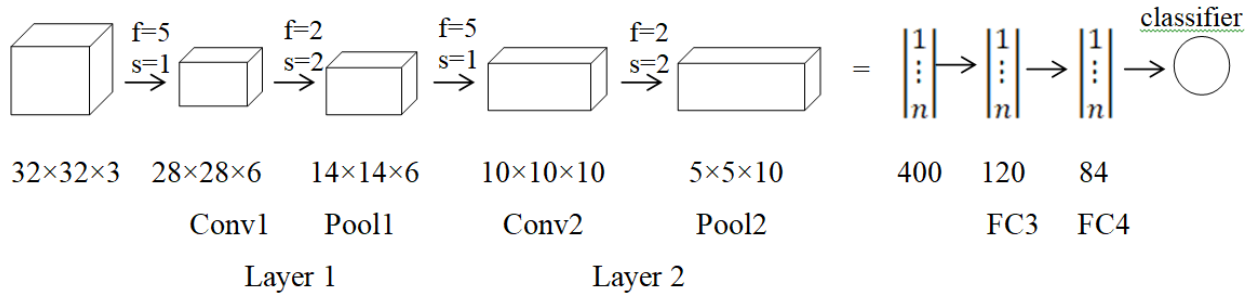


Рисунок 3.4 – Приклад структури CNN

На початку мережа складається з декількох поєднаних шарів згортки і об'єднання, за якими слідують декілька повністю пов'язаних шарів на кінці. Як видно з наведеного вище прикладу, висота і ширина вхідних даних зменшуються, коли ми йдемо глибше в мережу (від 32×32 до 5×5) і кількість каналів збільшується (з 3 до 10).

Розглянемо роботу шару згортки – процес максимальної згортки, на прикладі матриці 4×4.

$$\begin{bmatrix} 1 & 3 & 2 & 1 \\ 2 & 9 & 1 & 1 \\ 1 & 3 & 2 & 3 \\ 5 & 6 & 1 & 2 \end{bmatrix}$$

Для даного прикладу візьмемо розмір фільтра 2 і крок – 2. Ці числа називають гіпкрпарамтрами шару обєднання. Для кожного блоку матриці розміром 2x2 обираємо максимальне число. Тобто, для входу на шар матриці $n_h \times n_w \times n_c$, виходом буде:

$$[\ \{\ (n_h - f) / s + 1\} \times \{\ (n_w\ -\ f\_\ /\ s + 1\ \} \times \{\ n_c\ \}\ ].$$

Результатом максисальної згортки буде матриця 2×2:

$$\begin{bmatrix} 4 & 3 & 1 & 1 \\ 2 & 8 & 1 & 1 \\ 1 & 3 & 2 & 3 \\ 5 & 7 & 7 & 2 \end{bmatrix} \longrightarrow \begin{bmatrix} 8 & 1 \\ 7 & 7 \end{bmatrix}$$

Середня згортка відрізняється від максимальної тим, що результуюча матриця заповнюється не максимальним значенням, а середнім [17].

Висновок до розділу

В даному розділі була сформульована змістовна та математична постановки задачі навчання нейронної мережі для класифікації зображень будівель.

У розділі змістовної постановки задачі описана задача навчання нейронної мережі, вхідні дані та їх особливості. У розділі математичної постановки задачі було розглянуто структуру нейрона, архітектуру нейронної мережі, процес згортки та об’єднання. Було наведено покроковий алгоритм навчання нейронної мережі.

У розділі обґрунтування методу розв’язання було обґрунтовано вибір структури нейронної мережі та описано її порівняння з іншими популярними видами архітектури нейронних мереж. У розділі опису методів розв’язання представлено на прикладах принцип роботи CNN.

Після аналізу різних архітектур нейронних мереж можна зробити висновок, що CNN є найбільш ефективними для задачі класифікації зображень.

4 СТЕМОЮ ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

4.1 Засоби розробки

Для реалізації програмного продукту було використано мову програмування Python, фреймворк Flask, платформу .NET core та фреймворк для створення SPA – Angular. Для зберігання результатів аналізу викоритстовується MS SQL Server. Розробка виконувалася в Visual Studio Community та Visual Studio Code.

Python – популярна платформа, що використовується для досліджень і розробки виробничих систем завдяки своїй простоті. Python – це мова програмування з великою кількістю модулів, пакунків і бібліотек, що забезпечує великий вибір методів для досягнення завдання. Найбільш популярні бібліотеки Python – це NumPy, SciPy, Scikit-Learn, Matplotlib, що використовуються в data science та data analysis. Вони також широко використовуються для створення масштабованих алгоритмів машинного навчання. Python має готові реалізації популярних методів машинного навчання, такі як класифікація, регресія, рекомендації та кластеризація [11].

Для створення та навчання нейронної мережі бело використоно Python версії 3.5.2 та пакетний менеджер pip. Були використані наступні бібіліотеки Python:

— h5py – це інтерфейс Pythonic для формату бінарних даних HDF5. Дозволяє зберігати величезну кількість числових даних і легко маніпулювати цими даними з NumPy;
— imageai – бібліотека для роботи з компютерним баченням – розпізнавання зображень та розпізнавання відео;

- keras – бібліотека, що містить API для роботи з згортковими нейронними мережами та дозволяє оброблювати зображення як на CPU, так і на GPU. Бібілотека використовує TensorFlow;
- matplotlib – бібліотека для побудови графіків функцій, гістограми, діаграми, графіки розсіювання, тощо;
- numpy – бібліотека для роботи з матрицями. Має велику кількість готових функцій, призначених для роботи з багатовимірними масивами. Зображення для нейронної мережі представлене у вигляді матриці пікселів, тому дана бібліотека була корисною;
- opencv-python – неофіційний, але дуже поширений пакет OpenCV для Python. Дозволяє працювати з зображеннями і виконувати такі операції, як редагування пікселів, геометричні перетворення та інші функції обробки зображень;
- pillow – бібліотека для роботи з растровими зображеннями;
- scipy – бібліотека з містить спеціальні функції для модулі для оптимізації, обробки зображень, генетичних алгоритмів, розв'язування звичайних диференціальних рівнянь та багато іншого. Дана бібілотека розроблюється та підтримується розробниками Scilab та Matlab;
- tensorflow – бібліотека з відкритим вихідним кодом для створення моделей для машинного навчання, розроблена компанією Google [13].

Flask – мікрофреймворк для створення веб застосувань, що має базовий функціонал і підтримує базові можливості. За замовчуванням Flask не включає шар абстракції бази даних, перевірку форми і ще деякі можливості, для яких вже існують спеціальні бібліотеки, але їх можна додати, розширюючи мікрофреймворк додатковими можливостями. Численні

розширення забезпечують інтеграцію баз даних, перевірку форм, обробку завантажень, різні технології відкритої аутентифікації та багато іншого. [12]

.NET Core – це крос-платформений фреймворк з відкритим кодом, створений компанією Microsoft для розробки бібліотек, консольних та веб застосувань, що підтримує такі мови програмування, як C#, F# та частково VB. ASP.NET Core – технологія для створення різного типу веб-застосувань: від невеликих веб-сайтів до великих веб-порталів і веб-сервісів. ASP.NET Core працює поверх крос-платформеного середовища .NET Core, що може бути розгорнута на різних операційних системах: Windows, Mac OS X, Linux. Для розгортання веб-додатків можна використовувати традиційний IIS, або крос-платформенний веб-сервер Kestrel.

Angular – JavaScript фреймворк для створення веб, мобільних та десктопних застосувань. Є популярним завдяки Web Workers та server-side rendering. Має велику кількість пакетів для роботи з даними та їх зберігання в застосуванні. Дозволяє використовувати реактивне програмування.

Для збереження даних про Задачі (графи, розбиття, інформацію про виконання алгоритму) було використано MsSQL Server. Ця СУБД перш за все виділяється високою надійністю. Це досягається за рахунок застосування різних базових технологій, таких як створення відмовостійких кластерів, віддзеркалення, надання різноманітних засобів для роботи з журналами. Наступне гідність Microsoft SQL Server – це її можливості щодо масштабування і висока продуктивність. Особливу увагу в Microsoft SQL Server приділено питанням безпеки. Microsoft SQL Server активно застосовується сьогодні для створення корпоративних систем. Ця СУБД посідає перше місце на ринку за кількістю продажів копій [14].

Для доступу до бази даних у застосування було використано платформу Entity Framework (EF) Core. (EF) Core — це легка, розширювана, відкрита і крос-платформна версія популярної технології

доступу до даних Entity Framework і слугує об'єктно-реляційним маппером (O/RM), що дозволяє розробникам працювати з базою даних за допомогою об'єктів. Основними перевагами використання цієї платформи є можливість створювати базу даних та таблиці на основі класів, створених розробником, що виключає необхідність написання великої кількості коду доступу до даних. За допомогою EF Core доступ до даних здійснюється за допомогою моделі. Модель складається з класів сутностей і об'єкта контексту, який представляє сеанс з базою даних, що дозволяє запитувати і зберігати дані. Платформа надає можливість генерувати модель з існуючої бази даних. Вносити зміни до бази даних, створеної за допомогою цієї платформи безпечно та легко використовувати EF Migrations [16].

4.2 Вимоги до технічного забезпечення

Для правильної роботи даної програми до складу технічних засобів повинні входити будь-який комп’ютер з встановленим веб-браузером та доступом в Інтернет.

4.3 Архітектура програмного забезпечення

Програмний продукт складається з 3 застосувань:

- Web API застосування «arcitecture-recognition-api» для збереження даних користувача та збереження результатів аналізу;
- Web API застосування «arcitecture-prediction-api» – це мікро сервіс для розпізнавання архітектурних стилів зображень з використанням нейронної мережі;
- Angular застосування інтерфейсу користувача «arcitecture-recognition-client».

Дана архітектура дає велику гнучкість дозволяючи розроблювати та оновлювати певну частину програми незалежно від інших частин. Даний підхід зменшує час розробки при роботі командою та дозволяє вносити зміни до нейронної мережі, застосування представлення або застосування для збереження даних незалежно один від одного.

Веб-застосування «arcitecture-prediction-api» проводить аналіз зображення, надісланого до сервісу за допомогою HTTP запиту і проводить аналіз для зображення за допомогою навченої нейронної мережі. Ця частина була винесена до окремого сервісу так як це публічний API.

Розглянемо детальніше взаємодію компонентів застосування «arcitecture-prediction-api» за допомогою схеми компонентів, представленої у частині графічного матеріалу.

Схема ілюструє зв’язки між сторонніми бібліотеками, що були використані для створення застосування.

Застосунок «arcitecture-recognition-api» має трирівневу архітектуру. Даний тип архітектури програмного забезпечення є популярним при створенні клієнт-серверної системи, дозволяючи користувальницькому інтерфейсу веб-застосування бути переробленим або модернізованим, не впливаючи на нижню функціональну логіку бізнесу та доступу до даних. Ця архітектурна система часто ідеально підходить для вбудовування та інтеграції стороннього програмного забезпечення в існуючу програму. Ця гнучкість інтеграції також робить її ідеальною для вбудовування аналітичного програмного забезпечення в вже існуючі програми та часто використовується постачальниками вбудованих аналітиків з цієї причини. Тобто, трирівнева архітектура надає багато можливостей для розширення та розвитку програмного продукту [15].

Рівні застосування:

– Web API – приймає вхідні дані у вигляді HTTP-запитів по мережі (GET / POST / і т.д.) і повертає дані, форматовані як JSON / HTML / XML, і т.д.;
– рівень бізнес-логіки – рівень додатка містить функціональну бізнес-логіку, яка керує основними можливостями програми;
– рівень даних – рівень даних складається з бази даних / системи зберігання даних і рівня доступу до даних.

Схема структурна залежності рівнів застосування «arcitecture-recognition-api» представлена на рисунку 4.1.

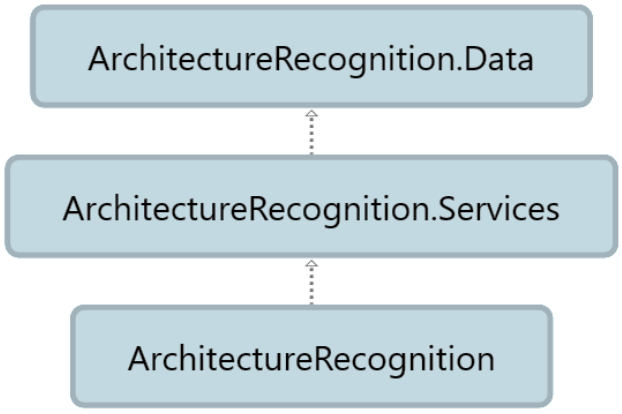


Рисунок 4.1 – Схема структурна залежності застосування «arcitecture-recognition-api»

Для збереження даних користувача до бази даних було використано платформу EF Core, що оперує бізнес-сутностями застосування для збереження даних та доступу до них. Розглянемо схему класів бізнес-сутностей застосування «arcitecture-recognition-api», представлену у частині графічного матеріалу.

Наведена вище схема описує залежності між моделями, що використовуються на нижньому рівні даних та описують інтерфейси для

структури бази даних. Розглянемо детальніше моделі, що наведені на схемі класів:

– RecognitionResult – клас для збереження аналізу зображення, зберігає інформацію про дату збереження, опис, назву. Містить посилання на користувача системи, що зберіг аналіз та посилання на зображення, що було збережене;
– User – модель, що описує користувача системи, має посилання на системну базу користувачів, необхідну для збереження інформації про користувачів, створених Identity Provider фреймворком .NET Core;
– Image – клас для роботи з зображенням, містить функції для роботи з зображеннями;
– Style – перелік архітектурних стилів, що відповідає переліку стилів, які була навчена розпізнавати нейронна мережа, є відповідністю styles.json з застосування «arcitecture-prediction-api».

У застосування наявні сервіси та контролери, що приймають та відправляють HTTP запити.

Програмний продукт складається з великої кількості інтерфейсів та класів. Для більш глибокого розуміння внутрішніх процесів обміну інформацією між застосуваннями та класами в межах одного застосування розглянемо реакцію системи на такі дії користувача як авторизація в систему та аналіз зображення.

У частині графічного матеріалу представлений процес авторизації користувача на схемі послідовності. У даному процесі задіяні класи з застосувань інтерфейсу користувача та веб-сервісу, що взаємодіють між собою через HTTP протокол.

Процес визначення архітектурного стилю будівлі та збереження результатів аналізу наведений у частині графічного матеріалу на схемі послідовності.

4.4 Специфікація функцій

Розглянемо специфікацію публічних функцій (публічний API) застосувань.

Таблиця 4.1 – Функції класів програмного забезпечення

Назва	Опис
Застосунок «arcitecture-prediction-api» клас «app.py»:	
@app.route('/api/recognition', methods=['POST']) def get_styles()	Контроллер, що приймає HTTP запити типу 'POST' та оброблює їх. Інтерфейс контролеру: запит має містити зображення для аналізу у полі request.files['image']. Метод повертає один з можливих відповідей: – status code 200 та json об’єкт з описом стилів та ймовірностей відношення до нього; – status code 400 при відсутності секції files у HTTP запиті; – status code 400 при відсутності файлів у секції files у HTTP запиті.
def allowed_file(filename)	Повертає рядок в необхідному форматі з обмеженнями у типах файлів. Дозволені типи файлів для аналізу – 'png', 'jpg', 'jpeg'.
Застосунок «arcitecture-prediction-api» клас «prediction.py»:	

Продовження таблиці 4.1

Назва	**Опис**
def getModel()	Ініціалізує об’єкт для розпізнавання архітектурного стилю будівлі на зображенні: встановлює шлях до навченої моделі нейронної мережі та до файлу стилів styles.json. Повертає завантажену модель.
def predict(filename)	Проводить аналіз зображення та повертає об’єкт з описом стилів та ймовірностей відношення до нього.
Застосунок «arcitecture-prediction-api» клас «main.py»:	
main()	Ініціалізує flask застосування з описаними конфігураціями. Запускає застосування.
Застосунок «arcitecture-recognition-api» проект «ArchitectureRecognition» клас «AuthController.cs»:	
[HttpPost("sign-in")] [AllowAnonymous] public async Task<IActionResult> SignInAsync([FromBody] SignInModel signInModel)	Контроллер, що приймає HTTP запити типу 'POST' та виконує вхід користувача до системи. Приймає модель для входу в систему (обєкт з полями логін та пароль) з тіла запиту. Метод повертає один з можливих відповідей: – status code 200, json об’єкт з JWT токеном та описом користувача; – status code 400 при не валідному запиту; – status code 401 якщо не знайдено користувача з заданим логіном та паролем.

Продовження таблиці 4.1

Назва	**Опис**
[HttpPost("sign-up")] public async Task<IActionResult> SignUp([FromBody] SignUpModel signUpModel)	Контроллер, що приймає HTTP запити типу 'POST' та виконує реєстрацію користувача у системі. Приймає модель реєстрації з інформацією про користувача з тіла запиту. Метод повертає один з можливих відповідей: – status code 200, json об’єкт з JWT токеном та описом зареєстрованого користувача; – status code 400 при не валідному запиту; – status code 400 якщо користувач з заданим логіном вже існує в системі.
Застосунок «arcitecture-recognition-api» проект «ArchitectureRecognition» клас «RecognitionResultController.cs»:	
[HttpGet()] public async Task<ActionResult<RecognitionResultDto[]>> GetAllAsync()	Приймає HTTP запити типу ‘GET’ та повертає усі результати аналізу користувача ситеми. Метод повертає один з можливих відповідей: – status code 200 та массив результатів для даного користувача;

Продовження таблиці 4.1

Назва	**Опис**
[HttpGet("{id}")] public async Task<ActionResult<RecognitionResultDto>> GetByIdAsync(int id)	Приймає HTTP запити типу ‘GET’ та повертає результат аналізу користувача ситеми з заданим ідентифікатором. Метод повертає один з можливих відповідей: – status code 200 та результат; – status code 400 якщо ідентифікатор не валідний; – status code 401 якщо в базу не знайдено запису з заданим ідентифікатором для даного користувача.
[HttpPost()] public async Task<ActionResult<RecognitionResultDto>> CreateAsync([FromBody][Required]RecognitionResultDto resultDto, [FromForm]IFormFile file)	Приймає HTTP запити типу 'POST' та створює у базі запис про результат для переданої моделі та зображення. Метод повертає один з можливих відповідей: – status code 200 та створенний у базі результат; – status code 400 якщо модель не валідна;
[HttpPut("{id}")] public async Task<ActionResult> UpdateAsync([Range(1, int.MaxValue)]int id, [FromBody]RecognitionResultDto taskDto)	Приймає HTTP запити типу 'PUT' та видаляє у базі запис про результат для переданого ідентифікатору. Метод повертає один з можливих відповідей: – status code 200 та оновлений у базі обєкт; – status code 400 якщо модель не валідна;

Продовження таблиці 4.1

Назва	**Опис**
[HttpDelete("{id}")] public async Task<ActionResult> DeleteAsync([Range(1, int.MaxValue)]int id)	Приймає HTTP запити типу 'DELETE' та оновлює у базі запис про результат для переданої модел. Метод повертає один з можливих відповідей: – status code 200 та оновлений у базі обєкт; – status code 400 якщо ідентифікатор не валідний.
Застосунок «arcitecture-recognition-api» проект «ArchitectureRecognition» клас «UserController.cs»:	
[HttpGet("current")] public async Task<IActionResult> GetCurrentUserAsync()	Приймає HTTP запити типу ‘GET’ та повертає інформацію про користувача системи за переданим токеном. Метод повертає один з можливих відповідей: – status code 200 та результат; – status code 401 якщо токен не є валідним.

Висновок до розділу

У розділі програмного та технічного забезпечення було розглянуто технології, що використовувалися для створення програмного забезпечення – Python Flask Microframework, .NET Core Web API, СУБД Ms SQL та фреймворк для створення SPA – Angular. Було обґрунтовано вибір технологій та наведені переваги їх використання.

У розділі архітектура програмного забезпечення було наведено схеми класів та компонентів застосування, що були створенні для вирушення задачі

аналізу архітектурних стилів будівель. Основні бізнес-процеси системи – авторизація користувача та аналіз зображення були розглянуті у вигляді взаємодії класів та представленні на схемах послідовності. Так як система складається з 2 мікро сервісів та застосування інтерфейсу користувача, то обмін повідомленнями між ними здійснюється через протокол HTTP і представлений на схемах.

Опис архітектури програмного забезпечення представлений для трьох застосувань. На схемі класів бізнес сутностей наведено структуру класів, що відповідає структурі бази даних.

Даний розділ містить специфікацію функцій, що складають публічний API застосувань. Саме ці функції доступні клієнтам застосування і описують контракт взаємодії по протоколу HTTP.

5 ТЕХНОЛОГІЧНИЙ РОЗДІЛ

5.1 Керівництво користувача

Користувач для початку роботи з програмним забезпеченням має відкрити браузер і перейти на http://localhost:4200. Після загрузки файлів користувач побачить домашню сторінку, представлену на рисунку 5.1.

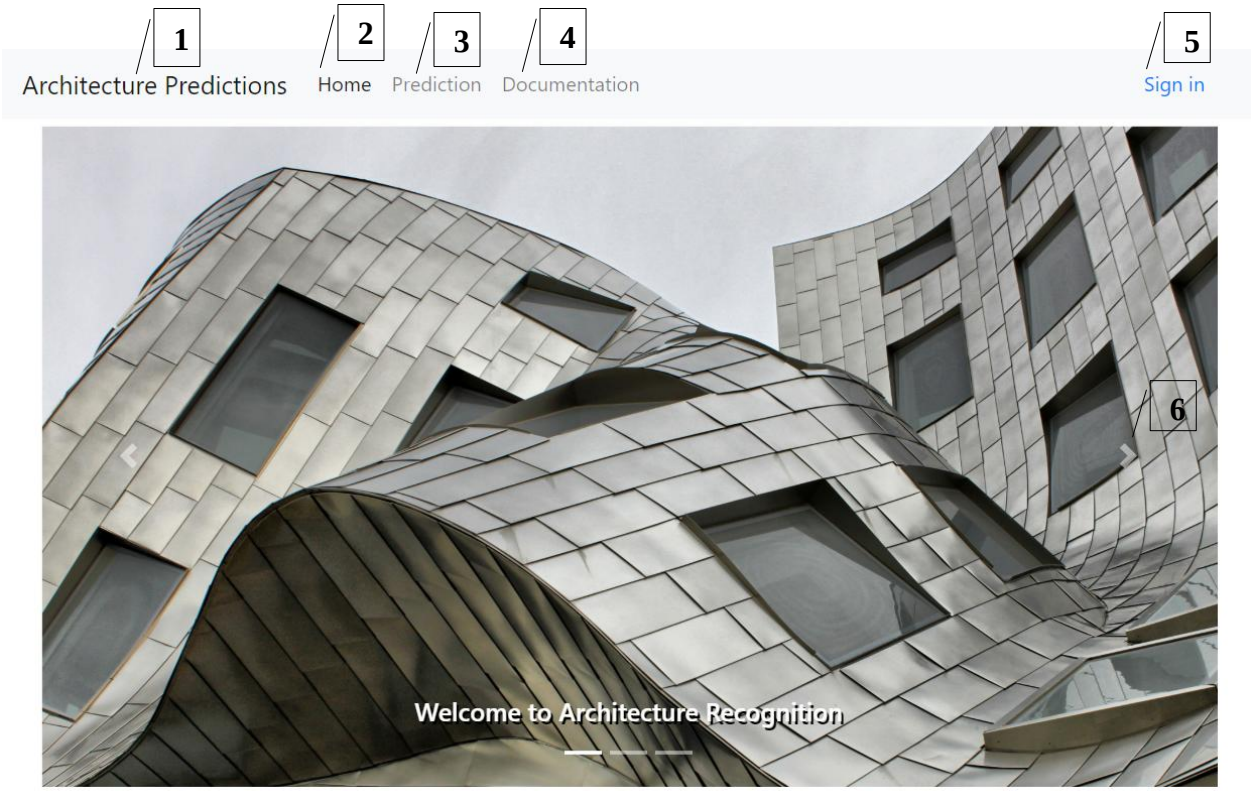


Рисунок 5.1 – Домашня сторінка

Розглянемо детальніше домашню сторінку користувача, головне меню сайту та вміст сторінки, що доступний незареєстрованому користувачу:

Кнопка 1: Назва застосування – перехід на домашню сторінку.

Кнопка 2: Посилання на домашню сторінку, що є поточною.

Кнопка 3: Посилання на сторінку для визначення архітектурних стилів зображень.

Кнопка 4: Посилання на сторінку з документацією.

Кнопка 5: Посилання на сторінку авторизації.

Кнопка 6: Кнопка для зміни зображення на наступне.

Перейдемо на сторінку визначення архітектурного стилю і завантажимо зображення для аналізу (рисунок 5.2).

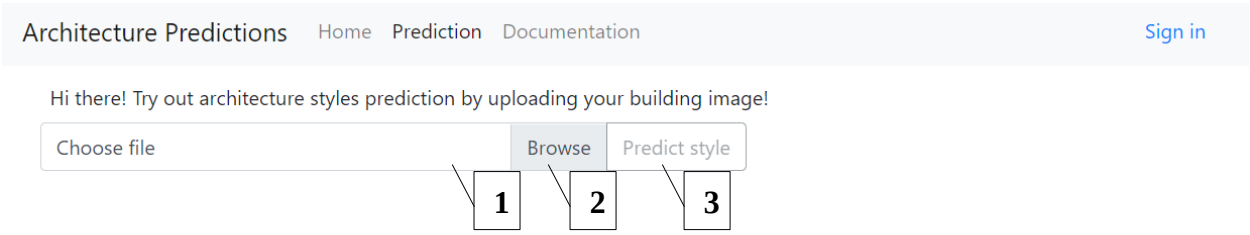


Рисунок 5.2 – Сторінка визначення архітектурного стилю будівлі

На сторінці наявний компонент для завантаження зображень. Під цифрою 1 місце для відображення повідомлення про призначення компоненти або ж для відображення імя завантаженого файлу. Далі йдуть кнопки обрати файл - 2, та відправити зображення на аналіз – 3, що є не активною поки зображення не обране. Натиснемо кнопку вибору зображення.

Рисунок 5.3 – Вибір зображення для аналізу

На рисунку 5.3 зображене стандартне вікно вибору файлів, що налаштоване для відображення файлів з розширенням 'png', 'jpg' та 'jpeg'. Виберемо один з файлів та натиснемо «Відкрити». Після завантаження файлу користувач зможе побачити зображення та його назву на сторінці.

Рисунок 5.4 – Завантажене зображення

Натискаємо кнопку для відправки на аналіз.

Рисунок 5.5 – Аналіз зображення

Рисунок 5.6 – Результат аналізу зображення

Так як користувач не увійшов в свій обліковий запис, він не може зберегти результат аналіз до системи. Тому натиснемо на кнопку входу в систему.

Рисунок 5.7 – Форма авторизації

Якщо користувач не має облікового запису в системі, він може створити її через форму реєстрації, натиснувши на кнопку зареєструватися.

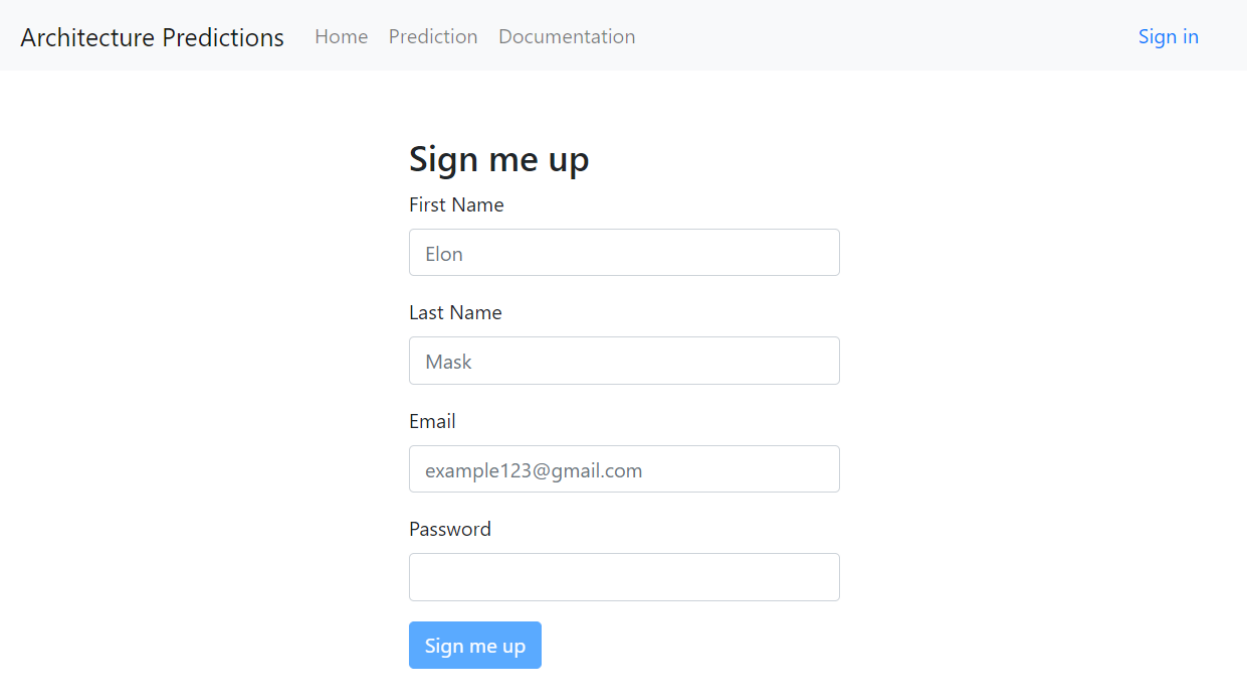


Рисунок 5.8 – Форма реєстрації

Після авторизації користувача в системі в головному меню з’являється посилання на додаткову сторінку – сторінка збережених результатів. Також користувач тепер зможе зберегти результати аналізу будівлі.

Рисунок 5.9 – Сторінка аналізу зображень для авторизованого користувача з формою для збереження результату

Після збереження результату перейдемо на сторінку збережених результатів, зображену на рисунку 5.10.

Рисунок 5.10 – Сторінка збережених результатів

На сторінці відображені збережені користувачем результати у вигляді таблиці. Розглянемо детальніше меню користувача та таблицю:

Кнопка 1: Для авторизованого користувача сайту доступна сторінка збережених результатів.

Кнопка 2: Імя користувача, авторизованого в системі.

Кнопка 3: Кнопка виходу з системи.

Кнопка 4: Дана іконка позначає поля, по яким можливе сортування даних в таблиці.

Кнопка 5: Іконка видалення результату.

1. Зображення будівлі для якого проведений аналіз. При натисканні на зображення відкривається вікно з збільшеним зображенням (рисунок 5.11).

Рисунок 5.11 – Перегляд зображення

При переході по посиланню документація користувач потрапляє на сторінку з описом архітектурних стилів та результатів. Сторінка показана на рисунку 5.12.

Рисунок 5.12 – Сторінка документації

5.2 Випробування програмного продукту

Розглянемо методики випробувань програмного продукту, порядок випробувань та опис тестів для перевірки відповідності програмного продукту функціональним вимогам, що описані у технічному завданні.

5.2.1 Мета випробувань

Метою випробувань являється перевірка відповідності функцій комплексу задач система розпізнавання архітектурних стилів будівель за зображеннями вимогам технічного завдання.

5.2.2 Загальні положення

Випробування проводяться на основі наступних документів:

– ГОСТ 34.603−92. Інформаційна технологія. Види випробувань автоматизованих систем;
– ГОСТ РД 50-34.698-90. Автоматизовані системи вимог до змісту.

5.2.3 Результати випробувань

Кожен застосування програмного продукту був перевірений на відповідність функціональним вимогам. Зміст та результати випробувань розглянуті нижче у таблицях 5.1 – 5.16.

Розглянемо зміст та результати тестування. Тести включають перевірку основної функціональності – навігація сайтом, авторизація та реєстрація, аналіз зображень та збереження результатів. Тестові сценарії та результат їх проходження наведений у таблицях 5.1 – 5.16.

Таблиця 5.1 – Тестування форми авторизації на непорожні поля

Тест:	Форма авторизації перевіряє введені дані на не пусті поля
Початковий стан системи:	Відкрита сторінка з формою авторизації
Дія:	Натискаємо кнопку «вхід» з пустими полями логін та пароль
Очікуваний результат:	Поля для вводу логіна та пароля виділені червоним кольором; повідомлення «Email is required!» та «Password is required!» з’явилися поряд з полями для вводу;
Фактичний результат співпадає з очікуваним:	Так

Таблиця 5.2 – Тестування форми авторизації на відкриття домашньої сторінки

Тест:	Форма авторизації при успішній авторизації відкриває домашню сторінку
Початковий стан системи:	Відкрита сторінка з формою реєстрації
Дія:	Вводимо правильний логін та пароль і натискаємо кнопку «вхід»
Очікуваний результат:	Відкривається домашня сторінка; В меню відображається ім’я поточного користувача
Фактичний результат співпадає з очікуваним:	Так

Таблиця 5.3 – Тестування форми авторизації на зберігання JWT в Local Storage

Тест:	Форма авторизації при успішній авторизації зберігає згенерований JWT в Local Storage
Початковий стан системи:	Відкрита сторінка з формою реєстрації
Дія:	Вводимо правильний логін та пароль і натискаємо кнопку «вхід»
Очікуваний результат:	В dev tools веб-браузера, в вкладці application, розділ local storage для домену http://localhost:4200 існує не пусте значення з ключем «rawToken»
Фактичний результат співпадає з очікуваним:	Так

Таблиця 5.4 – Тестування форми реєстрації на непорожні поля

Тест:	Форма реєстрації перевіряє введені дані
Початковий стан системи:	Відкрита сторінка з формою реєстрації
Дія:	Натискаємо кнопку «зареєструватися» з пустими полями ім’я, прізвище, логін та пароль
Очікуваний результат:	Усі незаповнені поля для вводу виділені червоним кольором; повідомлення «First name is required!», «Last name is required!», «Email is required!» та «Password is required!» з’явилися поряд з полями для вводу; користувач не перенаправлений на домашню сторінку; JWT збережений в Local Storage
Фактичний результат співпадає з очікуваним:	Так

Таблиця 5.5 – Тестування меню користувача – гостя

Тест:	В меню користувача відображені посилання на доступні незареєстрованому користувачу системи
Початковий стан системи:	Відкрита домашня сторінка; користувач не ввійшов в систему
Дія:	-
Очікуваний результат:	Користувач бачить назву застосування, посилання на сторінки «аналіз зображення», «документація» та «вхід у систему»
Фактичний результат співпадає з очікуваним:	Так

Таблиця 5.6 – Тестування меню авторизованого користувача

Тест:	В меню користувача відображені посилання на доступні авторизованому користувачу
Початковий стан системи:	Відкрита домашня сторінка; користувач ввійшов в систему
Дія:	-
Очікуваний результат:	Користувач бачить назву застосування, посилання на сторінки «аналіз зображення», «збережені результати», «документація», своє ім’я та кнопку «вихід з системи»
Фактичний результат співпадає з очікуваним:	Так

Таблиця 5.7 – Тестування меню користувача, посилання на сторінку аналізу

Тест:	Посилання на сторінку аналізу зображень в меню користувача відкриває сторінку аналізу зображень
Початковий стан системи:	Відкрита домашня сторінка
Дія:	Натискаємо на посилання «Prediction»
Очікуваний результат:	Перехід на сторінку аналізу зображень
Фактичний результат співпадає з очікуваним:	Так

Таблиця 5.8 – Тестування меню користувача, посилання на сторінку з результатами

Тест:	Посилання на сторінку збережених результатів в меню користувача відкриває сторінку з збереженими результатами
Початковий стан системи:	Відкрита домашня сторінка; користувач увійшов в свій обліковий запис
Дія:	Натискаємо на посилання «Saved Results»
Очікуваний результат:	Перехід на сторінку з збереженими результатами
Фактичний результат співпадає з очікуваним:	Так

Таблиця 5.9 – Тестування меню користувача, посилання на сторінку документацією

Тест:	Посилання на сторінку збережених результатів в меню користувача відкриває сторінку з документацією
Початковий стан системи:	Відкрита домашня сторінка
Дія:	Натискаємо на посилання «Documentation»
Очікуваний результат:	Перехід на сторінку з документацією
Фактичний результат співпадає з очікуваним:	Так

Таблиця 5.10 – Тестування аналізу зображення для гостя

Тест:	На сторінці аналізу зображень гість має змогу завантажити зображення та провести аналіз, але не може зберегти результат
Початковий стан системи:	Відкрита сторінка аналізу зображень; користувач не авторизований
Дія:	Завантажуємо зображення будівлі і натискаємо кнопку «Predict»
Очікуваний результат:	На сторінці відображається анімація завантаження результатів; коли аналіз завершується, результат аналізу архітектурних стилів відображений
Фактичний результат співпадає з очікуваним:	Так

Таблиця 5.11 – Тестування аналізу зображення для авторизованого користувача

Тест:	На сторінці аналізу зображень авторизований користувач має змогу завантажити зображення та провести аналіз, після чого може зберегти результат
Початковий стан системи:	Відкрита сторінка аналізу зображень; користувач авторизований
Дія:	Завантажуємо зображення будівлі і натискаємо кнопку «Predict»
Очікуваний результат:	На сторінці відображається анімація завантаження результатів; коли аналіз завершується, результат аналізу архітектурних стилів відображений; на сторінці відображена форма для вводу детальної інформації про аналіз для збереження; на сторінці відображена кнопка для збереження результату
Фактичний результат співпадає з очікуваним:	Так

Таблиця 5.12 – Тестування виходу з системи користувача

Тест:	Вихід з системи має видалити дані поточного користувача
Початковий стан системи:	Відкрита домашня сторінка; користувач авторизований
Дія:	Натискаємо кнопку вихід з системи
Очікуваний результат:	В меню не відображається ім'я користувача, посилання на сторінку збережених результатів та кнопка вихід з системи; відображається посилання на сторінку авторизації; в dev tools веб-браузера, в вкладці application, розділ local storage для домену http://localhost:4200 не існує значення з ключем «rawToken»
Фактичний результат співпадає з очікуваним:	Так

Таблиця 5.13 – Тестування сортування результатів за назвою

Тест:	Кнопка сортування за назвою має сортувати збережені результати
Початковий стан системи:	Відкрита домашня сторінка; користувач авторизований
Дія:	Зберегти 2 результати аналізу з різним значенням назви; перейти на сторінку збережених результатів і відсортувати за назвою
Очікуваний результат:	У таблиці збережених результатів рядки відсортовані в алфавітному порядку
Фактичний результат співпадає з очікуваним:	Так

Таблиця 5.14 – Тестування сортування результатів за назвою стилю

Тест:	Кнопка сортування за назвою стилю має сортувати збережені результати
Початковий стан системи:	Відкрита домашня сторінка; користувач авторизований
Дія:	Зберегти 2 результати аналізу з різним архітектурним стилем; перейти на сторінку збережених результатів і відсортувати за назвою стилю
Очікуваний результат:	У таблиці збережених результатів рядки відсортовані в алфавітному порядку за значенням архітектурного стилю
Фактичний результат співпадає з очікуваним:	Так

Таблиця 5.15 – Тестування сортування результатів за датою проведення аналізу

Тест:	Кнопка сортування за датою проведення аналізу має сортувати збережені результати
Початковий стан системи:	Відкрита домашня сторінка; користувач авторизований
Дія:	Зберегти 2 результати аналізу з інтервалом в більш ніж хвилина; перейти на сторінку збережених результатів і відсортувати за датою проведення аналізу
Очікуваний результат:	У таблиці збережених результатів рядки відсортовані в порядку зростання дати
Фактичний результат співпадає з очікуваним:	Так

Таблиця 5.16 – Тестування сортування результатів за датою зображення

Тест:	Кнопка сортування за датою зображення має сортувати збережені результати
Початковий стан системи:	Відкрита домашня сторінка; користувач авторизований
Дія:	Зберегти 2 результати аналізу з різними датами, коли було зроблене зображення; перейти на сторінку збережених результатів і відсортувати за датою зображення
Очікуваний результат:	У таблиці збережених результатів рядки відсортовані в порядку зростання дати зображення
Фактичний результат співпадає з очікуваним:	Так

Висновок до розділу

У технологічному розділі було розглянуто детально опис програмного продукту та представлені його екранні форми.

У розділі керівництво користувача розглянуто детально користувацький інтерфейс. Представлений опис меню користувача – призначення кожного посилання в меню, процес реєстрації та авторизації користувача. Було розглянуто сторінки аналізу архітектурного стилю будівлі за зображенням, процес завантаження зображення та збереження отриманих результатів. Для сторінки збережених результатів розглянуто процес сортування результатів у таблиці, можливість видалити результати та як детально переглянути зображення.

У розділі випробування програмного продукту наведено мету та загальні положення випробувань. У розділі наведені тесові сценарії, що перевіряють відповідність програмного продукту функціональним вимогам, представленим у технічному завданні. Також наведені тестові сценарії

перевіряють коректність перевірки введених у форми авторизації та реєстрації даних.

ЗАГАЛЬНІ ВИСНОВКИ

У пояснювальный записці розглянуто детальний опис дипломного проекту, що присвячений розробці комплексу задач класифікації зображень з метою визначення архітектурних стилів будівель.

Визначено мету та призначення розробки, що описана у розділі загальних положеннь.

У розділі опису предметного середовища описано предметну область дипломного проекту. Для реалізації комплексу задач важливо розуміти що таке архітектурний стиль та відмінності між певними стилями. Саме в цьому розділі представлений короткий опис стилів, що розглядаються в рамках системи та особливості кожного з них, що дозволяють легко відрізняти їх між собою. У цьому розділі описано важливість та проблему визначення архітектурних стилів.

Розглянуто процес діяльності користувача та описано функціональну модель за допомогою схеми варіантів використання – дії, що можуть виконувати різні ролі користувачів у системі. Саме ця схема відображає функціональні вимоги до системи та описує можливі ролі користувачів та їх відмінності.

Порівняння системи з іншими рішеннями, а також розглянуто різні варіанти реалізації задачі класифікації зображень за допомогою різних алгоритмів наведене у розділі наявних аналогів.

Розділ інформаційного забезпечення призначений для опису даних, якими оперує система та структур зберігання даних. Описано вхідні дані системи – вибірка великої кількості зображень, відсортованих за архітектурним стилем, до якого вони належать. Також у розділі вхідних даних описано архітектурні стилі, для яких створена вибірка та кількість

зобраень кожного архітектурного стилю та ресурси, що використовувалися для створення вибірки.

У розділі вихідних даних описані дані, що отримує користувач у результаті аналізу та модель, що отримана в результаті навчання алгоритму. Опис структури бази даних, таблиць та колонок представлений у розділі опис структури бази даних. У ньому наведено призначення та обгрунтування вибору способу зберігання зображень, для яких проводиться аналіз. Також описано структуру і призначення інших масивів інформації, що використовуються у застосування.

Математичне забезпечення – розділ, що містить змістовну та математичну постановку задачі: описує обрані алгоритми для розвязання задачі та демонструє приклади їх застосування.

Опис та обгрунтування вибору засобів розробки наведено у розділі стемою програмне та технічне забезпечення. Програмний продукт є веб-застосуванням і вимагає певного технічного забезпечення, що описано у розділі вимог до технічного забезпечення. У даному розділі описано архітектуру програмного забезпечення: представлено детальний опис застосувань, що були створені для розвязання задачі, описано схему компонентів, схему залежностей застосувань, класів бізнес-сутностей. Основні процеси у системі та демонстрація внутрішньої реалізації та взаємодії окремих класів представлена за допомогою схем послідовності авторизації користувача та схеми послідовності визначення архітектурного стилю та збереження результату. Для публічних інтерфейсів програми наведена специфікація функцій. У технологічному розділі наведено керівництво користувача та екранні форми застосування. Основні тестові сценарії та результат проходження тестів описаний у розділі випробування програмного продукту.

Програмний продукт відповідає функціональним вимогам, описаним у технічному завданні до дипломного проекту.

ПЕРЕЛІК ПОСИЛАНЬ

1. Jose Llamas, Pedro M. Lerones, Roberto Medina, Eduardo Zalama, Jaime Gómez-García-Bermejo. Classification of Architectural Heritage Images Using Deep Learning Techniques / Jose Llamas, Pedro M. Lerones, Roberto Medina, Eduardo Zalama, Jaime Gómez-García-Bermejo // Applied Sciences — Open Access Journal. – Basel, Switzerland, 26 september 2017. 25 с. – (Appl. Sci. 2017, 7, 992; doi:10.3390/app7100992).
2. Земляна і. О. Найбільш значимі архітектурні стилі [електронний ресурс] / Ірина Ілександрівна Земляна // класна оцінка. – 2011. – Режим доступу до ресурсу: http://klasnaocinka.com.ua/ru/article/naibolee-znachimie-arkhitekturnie-stili.html.
3. Chris P. Classifying U.S. Houses by Architectural Style Using Convolutional Neural Networks / Pesto Chris. – Stanford: Stanford University. – 9 с.
4. Z. Xu, D. Tao, Y. Zhang, J. Wu, A. Tsoi, "Architectural Style Classification using Multinomial Latent Logistic Regression", European Conference on Computer Vision (ECCV2014), 2014.
5. Russell S. To BLOB or Not To BLOB: Large Object Storage in a Database or a Filesystem / S. Russell, V. Catharine, G. Jim. // Microsoft Research, Microsoft Corporation, One Microsoft Way. – 2006. – С. 2 – 11.
6. Shalunts, Gayane, Yll Haxhimusa, and Robert Sablatnig. "Architectural style classification of building facade windows." International Symposium on Visual Computing. Springer Berlin Heidelberg, 2011.
7. Classification of Architectural Heritage Images Using Deep Learning Techniques Jose Llamas 1,* ID , Pedro M. Lerones 1 , Roberto Medina 1 , Eduardo Zalama 2 and Jaime Gómez-García-Bermejo 2 ID

https://pdfs.semanticscholar.org/1a1c/4e75c74b715fcc0903a044c4f7aa3d3bbf1c.pdf.

8. Kevin G. An introduction to neural networks / Gurney Kevin. – London and New York: Taylor & Francis e-Library, 2004. – 317 с. – (ROUTLEDGE). – (0-203-45151-1).
9. Sorokina K. Image Classification with Convolutional Neural Networks [Електронний ресурс] / Ksenia Sorokina // A Medium Corporation. – 201. – Режим доступу до ресурсу: https://medium.com/@ksusorokina/image-classification-with-convolutional-neural-networks-496815db12a8.
10. Chen W. Convolutional Neural Network for Image Classification / W. Chen, X. Yang.. – (Johns Hopkins University).
11. Machine Learning with Python – Hyderabad: Tutorials Point (I) Pvt. Ltd., 2018. – 68 с.
12. Welcome to Flask [Електронний ресурс] // Pallets Team. – 2010. – Режим доступу до ресурсу: http://flask.pocoo.org/docs/1.0/.
13. Olafenwa M. ImageAI Documentation Release 2.0.2 / M. Olafenwa, J. Olafenwa., 2019. – 31 с.
14. Сравнение баз данных: Microsoft Sql Server и Microsoft Visual Foxpro [Електронний ресурс] // SoftClipper.net. – 2019. – Режим доступу до ресурсу: https://softclipper.net/foxpro-i-sql/sravnenie-baz-dannykh-microsoft-sql-server-i-microsoft-visual-foxpro.html.
15. 3-Tier Architecture: A Complete Overview [Електронний ресурс] // Jinfonet Software, Inc.. – 2019. – Режим доступу до ресурсу: https://www.jinfonet.com/resources/bi-defined/3-tier-architecture-complete-overview/.
16. Miller R. Entity Framework Core / Rowan Miller. // docs.microsoft.com. – 2017. – С. 1.

17. Pulkit S. A Comprehensive Tutorial to learn Convolutional Neural Networks from Scratch (deeplearning.ai Course #4) / Sharma Pulkit. // Analytics Vidhya. – 2018. – №3. – С. 1.

ДОДАТОК А

Тексти програмного коду

Система розпізнавання архітектурних стилів будівель за зображеннями

(Найменування програми (документа))

DVD-R

(Вид носія даних)

11 арк, 8000 Кб

(Обсяг програми (документа) , арк.,) Кб)

Київ – 2019 року

```
[Route("api/[controller]")]
[ApiController]
public class AuthController : ControllerBase
{
    private readonly SignInManager<AuthUser> _signInManager;
    private readonly UserManager<AuthUser> _userManager;
    private readonly IConfiguration _configuration;
    private readonly IUserService _appUserService;
    public AuthController(SignInManager<AuthUser> signInManager,
        UserManager<AuthUser> userManager,
        IUserService appUserService,
        IConfiguration configuration)
    {
        _signInManager = signInManager ?? throw new ArgumentNullException(nameof(signInManager
));
        _userManager = userManager ?? throw new ArgumentNullException(nameof(userManager));
        _configuration = configuration ?? throw new ArgumentNullException(nameof(configuration
));
        _appUserService = appUserService ?? throw new ArgumentNullException(nameof(appUserServ
ice));
    }

    [HttpPost("sign-in")]
    [AllowAnonymous]
    public async Task<IActionResult> SignInAsync([FromBody] SignInModel signInModel)
    {
        if (!ModelState.IsValid)
        {
            return BadRequest(ModelState);
        }
        Microsoft.AspNetCore.Identity.SignInResult result =
            await _signInManager.PasswordSignInAsync(signInModel.Login, signInModel.Password,
false, false);

        if (!result.Succeeded)
        {
            return Unauthorized();
        }
        AuthUser authUser = _userManager.Users.FirstOrDefault(user => user.UserName == signInM
odel.Login);
        User appUser = await _appUserService.GetByIdAsync(authUser.Id);
        string token = GenerateJwtTokenString(authUser);

        return Ok(new { user = appUser, token });
    }
    [HttpPost("sign-up")]
    public async Task<IActionResult> SignUp([FromBody] SignUpModel signUpModel)
    {
        if (!ModelState.IsValid)
        {
            return BadRequest(ModelState);
        }
        var authUser = new AuthUser
        {
            UserName = signUpModel.SignInModel.Login
        };

        var result = await _userManager.CreateAsync(authUser, signUpModel.SignInModel.Password
);
        if (!result.Succeeded)
        {
            return BadRequest(result.Errors);
        }
```

```
}

        await _signInManager.SignInAsync(authUser, false);
        string authUserId = _userManager.Users.FirstOrDefault(user => user.UserName == authUse
r.UserName).Id;

        var appUser = new User
        {
            Id = authUserId,
            FirstName = signUpModel.FirstName,
            LastName = signUpModel.LastName
        };
         await _appUserService.AddAsync(appUser);
        string token = GenerateJwtTokenString(authUser);
        return Ok(new { user = appUser, token });
    }

    private string GenerateJwtTokenString(AuthUser user)
    {
        long unixNowSeconds = DateTimeOffset.Now.ToUnixTimeSeconds();
        long expirationTime = unixNowSeconds + 10 * 3600;
        var claims = new List<Claim>
            {
                new Claim(JwtRegisteredClaimNames.Sub, user.Id),
                new Claim(JwtRegisteredClaimNames.Iat, unixNowSeconds.ToString()),
            };

        var securityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(_configuration["Jwt:
Key"]));
        var credentials = new SigningCredentials(securityKey, SecurityAlgorithms.HmacSha256);

        var token = new JwtSecurityToken(
            _configuration["Jwt:Issuer"],
            _configuration["Jwt:Audience"],
            claims,
            expires: DateTime.Now.AddMinutes(10),
            signingCredentials: credentials
        );
        return new JwtSecurityTokenHandler().WriteToken(token);
    }
}
[Route("api/[controller]")]
[ApiController]
public class RecognitionResultController : ControllerBase
{
    private readonly IRecognitionResultService _service;
    private readonly IRecognitionResultMapper _mapper;
    private readonly IHostingEnvironment _env;
    public RecognitionResultController(IRecognitionResultService service, IRecognitionResultMa
pper mapper, IHostingEnvironment env)
    {
        _service = service ?? throw new ArgumentNullException(nameof(service));
        _mapper = mapper ?? throw new ArgumentNullException(nameof(mapper));
        _env = env;
    }
    [HttpGet()]
    public async Task<ActionResult<RecognitionResultDto[]>> GetAllAsync()
    {
        string userId = User.FindFirst(JwtRegisteredClaimNames.Sub).Value;
        //string userId = 0;
        List<RecognitionResult> recognitionResults = await _service.GetAllAsync(userId);
        if (recognitionResults == null || !recognitionResults.Any())
        {
```

```
            return Ok(Array.Empty<RecognitionResult>());
        }

        RecognitionResultDto[] recognitionResultDtos = recognitionResults.Select(recognitionRe
sult => _mapper.MapToDto(recognitionResult)).ToArray();
        return Ok(recognitionResultDtos);
    }


    [HttpGet("{id}")]
    public async Task<ActionResult<RecognitionResultDto>> GetByIdAsync(int id)
    {
        if (id <= 0)
        {
            return BadRequest();
        }

        RecognitionResult recognitionResult = await _service.GetByIdAsync(id);

        if (recognitionResult == null)
        {
            return NotFound();
        }

        RecognitionResultDto dto = _mapper.MapToDto(recognitionResult);
        return Ok(dto);
    }

    [HttpPost()]
    public async Task<ActionResult<RecognitionResultDto>> CreateAsync([FromForm]SaveRecognitio
nDto dto)
    {
        if (!ModelState.IsValid)
        {
            return BadRequest(ModelState);
        }
        var resultDto = JsonConvert.DeserializeObject<RecognitionResultDto>(dto.resultDto);
        IFormFile file = dto.file;

        string userId = User.FindFirst(JwtRegisteredClaimNames.Sub).Value;

        var PathWithFolderName = Path.Combine(Directory.GetCurrentDirectory(), "StaticFiles//"
);

        if (!Directory.Exists(PathWithFolderName))
        {
            // Try to create the directory.
            DirectoryInfo di = Directory.CreateDirectory(PathWithFolderName);
        }

        var fileNameExtention = file.FileName.Split(".");

        var imagePath = string.Join("", fileNameExtention.Take(fileNameExtention.Length - 1))
            + Guid.NewGuid()
            + "."
            + fileNameExtention[fileNameExtention.Length - 1];

        using (var fileStream = new FileStream(PathWithFolderName + imagePath, FileMode.Create
))
        {
            await file.CopyToAsync(fileStream);
        }
```

```
            resultDto.ImagePath = "/images/" + imagePath;
            var task = _mapper.MapToEntity(resultDto, userId);
            var created = await _service.AddRecognitionResultAsync(task);
            return Ok(_mapper.MapToDto(created));
        }

        [HttpPut("{id}")]
        public async Task<ActionResult> UpdateAsync([Range(1, int.MaxValue)]int id, [FromBody]Reco
    gnitionResultDto taskDto)
        {
            if (!ModelState.IsValid)
            {
                return BadRequest(ModelState);
            }
            string userId = User.FindFirst(JwtRegisteredClaimNames.Sub).Value;
            RecognitionResult recognitionResult = _mapper.MapToEntity(taskDto, userId);
            await _service.UpdateRecognitionResultAsync(id, recognitionResult);
            return NoContent();
        }

        [HttpDelete("{id}")]
        public async Task<ActionResult> DeleteAsync([Range(1, int.MaxValue)]int id)
        {
            if (!ModelState.IsValid)
            {
                return BadRequest(ModelState);
            }
            await _service.DeleteAsync(id);
            return NoContent();
        }
    }
    [Route("api/[controller]")]
    [ApiController]
    public class UserController : ControllerBase
    {
        private readonly IUserService _appUserService;
        public UserController(IUserService appUserService)
        {
            _appUserService = appUserService ?? throw new ArgumentNullException(nameof(appUserServ
    ice));
        }

        [HttpGet("current")]
        public async Task<IActionResult> GetCurrentUserAsync()
        {
            string currentUserId = HttpContext.User.FindFirstValue(ClaimTypes.NameIdentifier);
            string userId = User.FindFirst(JwtRegisteredClaimNames.Sub).Value;

            if (userId == null)
            {
                return Unauthorized();
            }
            User currentAppUser = await _appUserService.GetByIdAsync(userId);
            return Ok(currentAppUser);
        }
    }

    public interface IRecognitionResultRepository
    {
        void AddRecognitionResult(Entities.RecognitionResult recognitionResult);

        Task<List<Entities.RecognitionResult>> GetAllRecognitionResultsAync(string userId);
```

```
        Task<Entities.RecognitionResult> FindByIdAsync(int id);

        void Update(Entities.RecognitionResult recognitionResult);

        void Delete(Entities.RecognitionResult recognitionResult);
    }

    public interface IUnitOfWork
    {
        Task<int> SaveAsync();
    }
    public interface IUserRepository
    {
        void Add(Entities.User user);

        Task<Entities.User> GetByIdAsync(string id);
    }

    public sealed class AppDbContext : DbContext
    {
        public AppDbContext(DbContextOptions<AppDbContext> options) : base(options)
        {
        }

        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            modelBuilder.Entity<User>(user =>
            {
                user.Property(u => u.FirstName).IsRequired();
                user.Property(u => u.Id)
                    .IsRequired()
                    .ValueGeneratedNever();
                user.Property(u => u.LastName).IsRequired();
            });
            modelBuilder.Entity<RecognitionResult>(recognitionResults =>
            {
                recognitionResults.Property(tl => tl.AuthUserId)
                    .IsRequired();
                recognitionResults.Property(tl => tl.ImagePath)
                    .IsRequired();
                recognitionResults.Property(tl => tl.Style)
                    .IsRequired();

            });
            base.OnModelCreating(modelBuilder);
        }

        public DbSet<User> Users { get; set; }
        public DbSet<RecognitionResult> RecognitionResults { get; set; }

    }
    public sealed class RecognitionResultRepository : IRecognitionResultRepository
    {
        private readonly AppDbContext _dbContext;
        public RecognitionResultRepository(AppDbContext dbContext)
        {
            _dbContext = dbContext ?? throw new ArgumentNullException(nameof(dbContext));
        }
        public void AddRecognitionResult(RecognitionResult task)
        {
            _dbContext.Add(task);
        }
        public async Task<List<RecognitionResult>> GetAllRecognitionResultsAync(string userId)
```

```
    {
        return await _dbContext.RecognitionResults.Where(r => r.AuthUserId == userId).ToListAs
ync();
    }
    public async Task<RecognitionResult> FindByIdAsync(int id)
    {
        return await _dbContext.RecognitionResults.FindAsync(id);
    }
    public void Update(RecognitionResult recognitionResult)
    {
        _dbContext.RecognitionResults.Update(recognitionResult);
    }
    public void Delete(RecognitionResult recognitionResult)
    {
        _dbContext.Entry(recognitionResult).State = EntityState.Deleted;
    }
}
public sealed class UnitOfWork : IUnitOfWork
{
    private readonly AppDbContext _dbContext;
    public UnitOfWork(AppDbContext dbContext)
    {
        _dbContext = dbContext ?? throw new ArgumentNullException(nameof(dbContext));
    }
    public async Task<int> SaveAsync()
    {
        return await _dbContext.SaveChangesAsync();
    }
}
public sealed class UserRepository : IUserRepository
{
    private readonly AppDbContext _dbContext;

    public UserRepository(AppDbContext dbContext)
    {
        _dbContext = dbContext ?? throw new ArgumentNullException(nameof(dbContext));
    }
    public void Add(User user)
    {
        _dbContext.Add(user);
    }
    public async Task<User> GetByIdAsync(string id)
    {
        return await _dbContext.Users.FindAsync(id);
    }
}
public interface IRecognitionResultService
{
    Task<Data.Entities.RecognitionResult> AddRecognitionResultAsync(Data.Entities.RecognitionR
esult recognitionResult);

    System.Threading.Tasks.Task UpdateRecognitionResultAsync(int id, Data.Entities.Recognition
Result recognitionResult);

    Task<List<Data.Entities.RecognitionResult>> GetAllAsync(string userId);

    Task<Data.Entities.RecognitionResult> GetByIdAsync(int id);

    System.Threading.Tasks.Task DeleteAsync(int id);
}

public sealed class RecognitionResultService : IRecognitionResultService
{
```

```
    private readonly IRecognitionResultRepository _recognitionResultRepository;
    private readonly IUnitOfWork _unitOfWork;

    public RecognitionResultService(IRecognitionResultRepository recognitionResultRepository,
        IUnitOfWork unitOfWork)
    {
        _recognitionResultRepository = recognitionResultRepository ?? throw new ArgumentNullEx
ception(nameof(recognitionResultRepository));
        _unitOfWork = unitOfWork ?? throw new ArgumentNullException(nameof(unitOfWork));
    }

    public async Task<Data.Entities.RecognitionResult> AddRecognitionResultAsync(Data.Entities
.RecognitionResult recognitionResult)
    {
        if (recognitionResult == null)
        {
            throw new ArgumentNullException(nameof(RecognitionResult));
        }
        _recognitionResultRepository.AddRecognitionResult(recognitionResult);
        await _unitOfWork.SaveAsync();
        return recognitionResult;
    }

    public async Task DeleteAsync(int id)
    {
        if (id <= 0)
        {
            throw new ArgumentException("Id must be >= 0", nameof(id));
        }
        Data.Entities.RecognitionResult recognitionResult = await _recognitionResultRepository
.FindByIdAsync(id);

        if (recognitionResult == null)
        {
            throw new ArgumentException("not fount", nameof(id));
        }
        _recognitionResultRepository.Delete(recognitionResult);
        await _unitOfWork.SaveAsync();
    }


    public async Task<List<Data.Entities.RecognitionResult>> GetAllAsync(string userId)
    {
        return await _recognitionResultRepository.GetAllRecognitionResultsAync(userId);
    }
    public async Task<Data.Entities.RecognitionResult> GetByIdAsync(int id)
    {
        if (id <= 0)
        {
            throw new ArgumentException("Id must be more then zero", nameof(id));
        }
        return await _recognitionResultRepository.FindByIdAsync(id);
    }
    public async Task UpdateRecognitionResultAsync(int id, Data.Entities.RecognitionResult rec
ognitionResult)
    {
        if (id <= 0)
        {
            throw new ArgumentException("Id must be more then zero", nameof(id));
        }
        if (recognitionResult == null)
        {
            throw new ArgumentNullException(nameof(RecognitionResult));
```

```
        }
        Data.Entities.RecognitionResult entity = await _recognitionResultRepository.FindByIdAs
ync(id);

        if (entity == null)
        {
            throw new ArgumentException("not found", nameof(id));
        }
        entity.Title = recognitionResult.Title;
        entity.Description = recognitionResult.Description;
        entity.Date = recognitionResult.Date;
        entity.ImageDate = recognitionResult.ImageDate;
        entity.ImageName = recognitionResult.ImageName;
        entity.Style = recognitionResult.Style;
        await _unitOfWork.SaveAsync();
    }
}
public class Program
{
    public static void Main(string[] args)
    {
        CreateWebHostBuilder(args).Build().Run();
    }
    public static IWebHostBuilder CreateWebHostBuilder(string[] args) =>
        WebHost.CreateDefaultBuilder(args)
            .UseStartup<Startup>();
}
public class Startup
{
    public Startup(IConfiguration configuration)
    {
        Configuration = configuration;
    }
    public IConfiguration Configuration { get; }

    //This method gets called by the runtime. Use this method to add services to the containe
    public void ConfigureServices(IServiceCollection services)
    {
        services.AddDbContext<AuthDbContext>(options =>
        {
            options.UseSqlServer(Configuration.GetConnectionString("AuthDb"));
        });

        services.AddIdentity<AuthUser, IdentityRole>()
            .AddEntityFrameworkStores<AuthDbContext>()
            .AddDefaultTokenProviders();
        services.Configure<IdentityOptions>(options =>
        {
            options.Password.RequireDigit = false;
            options.Password.RequiredLength = 1;
            options.Password.RequiredUniqueChars = 0;
            options.Password.RequireLowercase = false;
            options.Password.RequireNonAlphanumeric = false;
            options.Password.RequireUppercase = false;
        });

        JwtSecurityTokenHandler.DefaultInboundClaimTypeMap.Clear();
        services.AddAuthentication(options =>
        {
            options.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;
            options.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;
        })
            .AddJwtBearer(options =>
```

```
            {
                options.TokenValidationParameters = new TokenValidationParameters
                {
                    ValidateIssuer = true,
                    ValidateAudience = true,
                    ValidateLifetime = true,
                    ValidateIssuerSigningKey = true,
                    ValidIssuer = Configuration["Jwt:Issuer"],
                    ValidAudience = Configuration["Jwt:Audience"],
                    IssuerSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(Configu
ration["Jwt:Key"]))
                };
            });
        services.AddMvc();
        services.AddCors();
        services.RegisterDependencies(Configuration);
    }

    // This method gets called by the runtime. Use this method to configure the HTTP request p
ipeline.
    public void Configure(IApplicationBuilder app, IHostingEnvironment env, AuthDbContext auth
DbContext, AppDbContext appDbContext)
    {
        if (env.IsDevelopment())
        {
            app.UseDeveloperExceptionPage();
        }
        else
        {
            app.UseHsts();
        }
        app.UseStaticFiles(new StaticFileOptions
        {
            FileProvider = new PhysicalFileProvider(
                Path.Combine(Directory.GetCurrentDirectory(), "StaticFiles")),
            RequestPath = "/images"
        });
        app.UseCors(builder =>
        {
            builder.AllowAnyOrigin();
            builder.AllowAnyHeader();
            builder.AllowAnyMethod();
            builder.AllowCredentials();
        });
        app.UseHttpsRedirection();
        app.UseAuthentication();
        app.UseMvc();
        authDbContext.Database.EnsureCreated();
        appDbContext.Database.EnsureCreated();
    }
}
```

```
import json
from flask_cors import CORS, cross_origin
from config import Configuration
from prediction import predict
import sys
import os
import numpy
from PIL import Image
from flask import Flask, flash, request, redirect, url_for, jsonify, abort
from werkzeug.utils import secure_filename
```

```
UPLOAD_FOLDER = ''
ALLOWED_EXTENSIONS = set(['png', 'jpg', 'jpeg'])

app = Flask(__name__)
app.config.from_object(Configuration)
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

def allowed_file(filename):
    return '.' in filename and \
           filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS


@app.route('/api/recognition', methods=['POST'])
def get_styles():
    # check if the post request has the file part
    if 'image' not in request.files:
        flash('No file part')
        return abort(400)
    file = request.files['image']
    # if user does not select file, browser also
    # submit an empty part without filename
    if file.filename == '':
        flash('No selected file')
        return abort(400)
    if file and allowed_file(file.filename):
        filename = secure_filename(file.filename)
        filePath = os.path.join(app.config['UPLOAD_FOLDER'], filename)
        file.save(filePath)
        result = predict(filename)
        os.remove(filePath)
        print(result)
        print(json.dumps(result))
        return json.dumps(result)

@NgModule({
  declarations: [
    AppComponent,
    HomeComponent,
    PredictionComponent,
    ResultsTableComponent,
    DocumentationComponent,
    NavBarComponent,
    SignInComponent,
    SignUpComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    HttpClientModule,
    FormsModule,
    DataTableModule
  ],
  providers: [
    { provide: Window, useValue: window },
    { provide: HTTP_INTERCEPTORS, useClass: TokenInterceptor, multi: true }
  ],
  bootstrap: [AppComponent]
})
export class AppModule { }

const routes: Routes = [
  { path: 'home', component: HomeComponent },
  { path: 'sign-in', component: SignInComponent },
```

```
  { path: 'sign-up', component: SignUpComponent },
  { path: 'prediction', component: PredictionComponent },
  { path: 'results', component: ResultsTableComponent,  canActivate: [AuthGuard] },
  { path: 'documentation', component: DocumentationComponent },
  { path: '', redirectTo: 'home', pathMatch: 'full' }
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
import { HttpInterceptor, HttpRequest, HttpHandler, HttpEvent } from '@angular/common/http';
import { Observable } from 'rxjs';
import { Injectable } from '@angular/core';
import { JwtService } from '../services/jwt.service';

@Injectable({
    providedIn: 'root'
  })
export class TokenInterceptor implements HttpInterceptor {

    constructor(private tokenService: JwtService) {}

    intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {
        req = req.clone({
            setHeaders: {
                Authorization: `Bearer ${this.tokenService.getRawToken()}`
            }
        });

        console.log('TokenInterceptor', req);
        return next.handle(req);
    }
}
import { Injectable } from '@angular/core';
import { CanActivate, ActivatedRouteSnapshot, RouterStateSnapshot, Router } from
'@angular/router';
import { UserService } from '../services/user.service';
import { Observable } from 'rxjs';

@Injectable({
  providedIn: 'root'
})
export class AuthGuard implements CanActivate {

  constructor(private router: Router, private userService: UserService) { }

  canActivate(
    next: ActivatedRouteSnapshot,
    state: RouterStateSnapshot): boolean {
    if (this.userService.currentUser) {
      return true;
    } else {
      this.router.navigateByUrl('/sign-in');
      return false;
    }
  }
}
```

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”
Кафедра автоматизованих систем обробки інформації та управління

УЗГОДЖЕНО	**ЗАТВЕРДЖУЮ**
Керівник проекту	**Завідувач кафедри**
_____ *М.О. Сперкач*	_____ *О.А. Павлов*
(підпис) (ініціали, прізвище)	(підпис) (ініціали, прізвище)
“15” квітня 2019 р.	“15” квітня 2019 р.

Система розпізнавання архітектурних стилів будівель за зображеннями

ТЕХНІЧНЕ ЗАВДАННЯ

Шифр *ДП ІС-5219.1180-с.ТЗ*

на 9 сторінках

Київ – 2019 року

ЗМІСТ

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Повне найменування системи та її умовне позначення

Повне найменування системи: «Система розпізнавання архітектурних стилів будівель за зображеннями».

Коротке найменування системи: «Система розпізнавання архітектурних стилів будівель за зображеннями».

1.2 Найменування організації-замовника та організацій-учасників робіт

Замовником проекту є кафедра Автоматизованих систем обробки інформації та управління факультету інформатики та обчислювальної техніки Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського». Представник замовника: доцент кафедри АСОІУ Сперкач Майя Олегівна. Адреса замовника: м. Київ, п-кт Перемоги 37.

Розробником системи є студентка групи ІС-52 кафедри Автоматизованих систем обробки інформації та управління Національного технічного університету України "Київський політехнічний інститут ім. І. Сікорського" Новіченко Неля Валеріївна.

1.3 Перелік документів, на підставі яких створюється система

При розробці системи і створення проектно-експлуатаційної документації виконавець повинен керуватися вимогами наступних нормативних документів:

- ДСТУ 19.201-78. Технічне завдання. Вимоги до змісту і оформлення;
- ДСТУ 34.601-90. Комплекс стандартів на автоматизовані системи. Автоматизовані системи. Стадії створення;
- ДСТУ 34.201-89. Інформаційні технології. Комплекс стандартів на автоматизовані системи. Види, комплексність і позначення документів при створенні автоматизованих систем.

1.4 Планові терміни початку і закінчення роботи зі створення системи

Плановий строк початку роботи по створенню системи 8 лютого 2019 року. Плановий строк кінця роботи по створенню системи для громадської системи фіксації правопорушень та її адміністрування 31 травня 2019 року.

2 ПРИЗНАЧЕННЯ І ЦІЛІ СТВОРЕННЯ СИСТЕМИ

2.1 Призначення системи

Розробка призначена для визначення архітектурних стилів будівель за зображенням різного формату та якості.

2.2 Мета створення системи

Метою розробки є спрощення процесу визначення архітектурного стилю будівлі та зниження похибки при визначенні стилю за рахунок аналізу отриманого зображення.

Завдяки створеній системі аналіз архітектурної спадщини та її документування може бути автоматизоване, що дуже важливо для зменшення часу на документування та підвищення достовірності. Це посприяє збереженню, підтримці в належному стані та відновленню будівель за рахунок можливості дослідження еволюції архітектурної споруди.

Для досягнення поставленої мети мають бути вирішенні такі задачі:

- створення достатньо великої та точної вибірки зображень та класифікація кожного зображення до певного стилю;
- розробка алгоритму, здатного с високою точністю класифікувати зображення (підбір типу нейронних сіток, що дасть найкращий результат для поставленої задачі);
- підтримка аналізу зображень різного розміру та якості;
- навчання нейронної мережі на створеній вибірці;
- розробка веб-застосування, що дозволить завантажувати зображення для аналізу;
- розробка веб-застосування, що дозволить зберігати результати аналізу до бази для задокументованої інформації про будівлі;
- підтримка можливості збереження результатів аналізу та перегляд еволюції зміни будівлі;

створення документації та інструкції для користувачів системи щодо інтерпретації отриманих результатів.

3 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Вимоги до функціональних характеристик

Функціональні вимоги до системи:

- Система надає можливість користувачу створити обліковій запис.
- Система надає можливість користувачу увійти в обліковий запис.
- Система надає можливість користувачу здійснити аналіз архітектурного стилю.
    1) Система надає можливість користувачу завантажити зображення будівлі.
    2) Система надає можливість користувачу зробити запит для аналізу архітектурного стилю.
    3) Система надає можливість користувачу переглянути результат аналізу.
    4) Система надає можливість користувачу зберегти результат аналізу.
    5) Система надає можливість користувачу ввести додаткову інформацію про зображення та аналіз.
- Система надає можливість користувачу переглянути попередньо збережені результати аналізу.
- Система надає можливість користувачу переглянути документацію та інструкції щодо інтерпретації результатів аналізу.

3.2 Вимоги до надійності

Система має зберігати приватну інформацію користувача (пароль від облікового запису) у зашифрованому вигляді. Всі дані, що вводяться користувачем, мають бути перевірені на формальну правильність.

3.3 Умови експлуатації

Для експлуатації розроблюваної системи користувач має мати персональний комп’ютер з встановленим веб-браузером та доступом в інтернет.

Всі користувачі системи повинні дотримуватися правил експлуатації електронної обчислювальної техніки.

3.4 Вимоги до складу і параметрів технічних засобів

Для коректної роботи веб-застосування мінімальними вимогами до апаратної частини ПК є:

- об’єм оперативної пам’яті не менше 6 ГБ;
- процесор з тактовою частотою не нижче 1 ГГц;
- наявність підключення до інтернету;

4 СТАДІЇ І ЕТАПИ РОЗРОБКИ

У таблиці 5.1 наведено календарний план робіт та терміни їх виконання.

Таблиця 5.1 – Календарний план виконання робіт

№ з/п	Назва етапів розробки програмного продукту	Строк виконання
1.	Встановлення необхідного програмного забезпечення для розробки програмного продукту	*15.04.2019*
2.	Проектування нейронної мережі	*18.04.2019*
3.	Навчання алгоритму розпізнавати архітектурні стилі	*25.04.2019*
4.	Розробка веб-сервісу для розпізнавання архітектурних стилів	*05.05.2019*
5.	Розробка бази даних	*09.05.2019*
6.	Розробка веб-сервісу для зберігання результатів аналізу	*13.05.2019*
7.	Розробка інтерфейсу користувача	*16.05.2019*
8.	Написання тестів	*19.05.2019*

5 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ СИСТЕМИ

5.1 Види випробувань

Види випробувань узгоджуються з замовником до проведення випробувань.

Здача програмного забезпечення відбувається на комп’ютері виконавця завдання з встановленим програмним забезпеченням, що відповідає вимогам.

Всі випробування, які проводилися для перевірки програмного продукту, наведені у пояснювальній записці до дипломного проекту. Методика тестувань наведена детально у пояснювальній записці. Випробуванням підлягає перевірка основних функцій системи. Було проведене функціональне, модульне та інтеграційне тестування.

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”
Кафедра автоматизованих систем обробки інформації та управління

УЗГОДЖЕНО

Керівник проекту

_____ *М.О. Сперкач*
(підпис) (ініціали, прізвище)

“13” травня 2019 р.

ЗАТВЕРДЖУЮ

В.о. завідувача кафедри

_____ *О.А.Павлов*
(підпис) (ініціали, прізвище)

“14” травня 2019 р.

Система розпізнавання архітектурних стилів будівель за зображеннями

ПРОГРАМА ТА МЕТОДИКА ВИПРОБУВАНЬ

Шифр *ДП ІС-5219.1181-с.ПМВ*

на 14 сторінках

Київ – 2019 року

ЗМІСТ

1 ОБ'ЄКТ ВИПРОБУВАННЯ

Об’єктом тестових випробувань є веб-застосунок розпізнавання архітектурних стилів будівель за зображеннями.

1.1 Найменування програми

Повне найменування системи: Система розпізнавання архітектурних стилів будівель за зображеннями.

1.2 Область застосування

Документування архітектурних стилів будівель, вимірювання та візуалізації документації та збереження спадщини.

2 МЕТА ВИПРОБУВАНЬ

Метою випробувань є перевірка відповідності системи розпізнавання архітектурних стилів будівель за зображеннями вимогам технічного завдання.

3 ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

3.1 Вимоги до функціональних характеристик

Функціональні характеристики системи мають відповідати функціональним вимогам функціональних характеристик наведених у технічному завданні дипломного проекту.

3.1.1 Вимоги до складу виконуваних функцій

Функціональні вимоги до системи:

1. Система надає можливість користувачу створити обліковій запис.
2. Система надає можливість користувачу увійти в обліковий запис.
3. Система надає можливість користувачу здійснити аналіз архітектурного стилю.
    1.1 Система надає можливість користувачу завантажити зображення будівлі.
    1.2 Система надає можливість користувачу зробити запит для аналізу архітектурного стилю.
    1.3 Система надає можливість користувачу переглянути результат аналізу.
    1.4 Система надає можливість користувачу зберегти результат аналізу.
    1.5 Система надає можливість користувачу ввести додаткову інформацію про зображення та аналіз.
4. Система надає можливість користувачу переглянути попередньо збережені результати аналізу.
5. Система надає можливість користувачу переглянути документацію та інструкції щодо інтерпретації результатів аналізу.

4 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

Документація програмного забезпечення має включати в себе специфікацію функцій публічного програмного інтерфейсу, технічне завдання, опис обраних технологій, текст програмного коду та випробування програмного продукту.

Специфікація функцій публічного програмного інтерфейсу передбачає опис функцій, їх призначення, можливі варіанти результатів та опис контрактів, через які відбувається взаємодія.

Технічне завдання містить опис системи – найменування системи, найменування замовника, календарний план робіт, постановку задання, функціональні та нефункціональні вимоги до програмного забезпечення. У технічному завданні має бути наведено призначення та мета створення системи.

Опис обраних технологій для розробки системи містить обґрунтування вибору технологій, порівняння з аналогами, переваги обраних технологій та короткий опис.

Тексти програмного коду містить в собі символічний запис програми на мові програмування, що була обрана в результаті вибору технології програмування.

ПМВ містить опис випробування програми та тестові сценарії для перевірки програмного продукту на відповідність функціональним вимогам.

5 СКЛАД І ПОРЯДОК ВИПРОБУВАНЬ

Тести включають перевірку основної функціональності – навігація сайтом, авторизація та реєстрація, аналіз зображень та збереження результатів. Тестові сценарії та результат їх проходження наведений у таблицях 5.1 – 5.16.

Таблиця 5.1 – Тестування форми авторизації на непорожні поля

Тест:	Форма авторизації перевіряє введені дані на не пусті поля
Початковий стан системи:	Відкрита сторінка з формою авторизації
Дія:	Натискаємо кнопку «вхід» з пустими полями логін та пароль
Очікуваний результат:	Поля для вводу логіна та пароля виділені червоним кольором; повідомлення «Email is required!» та «Password is required!» з’явилися поряд з полями для вводу;
Фактичний результат співпадає з очікуваним:	Так

Таблиця 5.2 – Тестування форми авторизації на відкриття домашньої сторінки

Тест:	Форма авторизації при успішній авторизації відкриває домашню сторінку
Початковий стан системи:	Відкрита сторінка з формою реєстрації
Дія:	Вводимо правильний логін та пароль і натискаємо кнопку «вхід»
Очікуваний результат:	Відкривається домашня сторінка; В меню відображається ім’я поточного користувача
Фактичний результат співпадає з очікуваним:	Так

Таблиця 5.3 – Тестування форми авторизації на зберігання JWT в Local Storage

Тест:	Форма авторизації при успішній авторизації зберігає згенерований JWT в Local Storage
Початковий стан системи:	Відкрита сторінка з формою реєстрації
Дія:	Вводимо правильний логін та пароль і натискаємо кнопку «вхід»
Очікуваний результат:	В dev tools веб-браузера, в вкладці application, розділ local storage для домену http://localhost:4200 існує не пусте значення з ключем «rawToken»
Фактичний результат співпадає з очікуваним:	Так

Таблиця 5.4 – Тестування форми реєстрації на непорожні поля

Тест:	Форма реєстрації перевіряє введені дані
Початковий стан системи:	Відкрита сторінка з формою реєстрації
Дія:	Натискаємо кнопку «зареєструватися» з пустими полями ім’я, прізвище, логін та пароль
Очікуваний результат:	Усі незаповнені поля для вводу виділені червоним кольором; повідомлення «First name is required!», «Last name is required!», «Email is required!» та «Password is required!» з’явилися поряд з полями для вводу; користувач не перенаправлений на домашню сторінку; JWT збережений в Local Storage
Фактичний результат співпадає з очікуваним:	Так

Таблиця 5.5 – Тестування меню користувача – гостя

Тест:	В меню користувача відображені посилання на доступні незареєстрованому користувачу системи
Початковий стан системи:	Відкрита домашня сторінка; користувач не ввійшов в систему
Дія:	-
Очікуваний результат:	Користувач бачить назву застосунку, посилання на сторінки «аналіз зображення», «документація» та «вхід у систему»
Фактичний результат співпадає з очікуваним:	Так

Таблиця 5.6 – Тестування меню авторизованого користувача

Тест:	В меню користувача відображені посилання на доступні авторизованому користувачу
Початковий стан системи:	Відкрита домашня сторінка; користувач ввійшов в систему
Дія:	-
Очікуваний результат:	Користувач бачить назву застосунку, посилання на сторінки «аналіз зображення», «збережені результати», «документація», своє ім’я та кнопку «вихід з системи»
Фактичний результат співпадає з очікуваним:	Так

Таблиця 5.7 – Тестування меню користувача, посилання на сторінку аналізу

Тест:	Посилання на сторінку аналізу зображень в меню користувача відкриває сторінку аналізу зображень

Продовження таблиці 5.7

Початковий стан системи:	Відкрита домашня сторінка
Дія:	Натискаємо на посилання «Prediction»
Очікуваний результат:	Перехід на сторінку аналізу зображень
Фактичний результат співпадає з очікуваним:	Так

Таблиця 5.8 – Тестування меню користувача, посилання на сторінку з результатами

Тест:	Посилання на сторінку збережених результатів в меню користувача відкриває сторінку з збереженими результатами
Початковий стан системи:	Відкрита домашня сторінка; користувач увійшов в свій обліковий запис
Дія:	Натискаємо на посилання «Saved Results»
Очікуваний результат:	Перехід на сторінку з збереженими результатами
Фактичний результат співпадає з очікуваним:	Так

Таблиця 5.9 – Тестування меню користувача, посилання на сторінку документацією

Тест:	Посилання на сторінку збережених результатів в меню користувача відкриває сторінку з документацією
Початковий стан системи:	Відкрита домашня сторінка
Дія:	Натискаємо на посилання «Documentation»

Продовження таблиці 5.9

Очікуваний результат:	Перехід на сторінку з документацією
Фактичний результат співпадає з очікуваним:	Так

Таблиця 5.10 – Тестування аналізу зображення для гостя

Тест:	На сторінці аналізу зображень гість має змогу завантажити зображення та провести аналіз, але не може зберегти результат
Початковий стан системи:	Відкрита сторінка аналізу зображень; користувач не авторизований
Дія:	Завантажуємо зображення будівлі і натискаємо кнопку «Predict»
Очікуваний результат:	На сторінці відображається анімація завантаження результатів; коли аналіз завершується, результат аналізу архітектурних стилів відображений
Фактичний результат співпадає з очікуваним:	Так

Таблиця 5.11 – Тестування аналізу зображення для користувача

Тест:	На сторінці аналізу зображень авторизований користувач має змогу завантажити зображення та провести аналіз, після чого може зберегти результат
Початковий стан системи:	Відкрита сторінка аналізу зображень; користувач авторизований
Дія:	Завантажуємо зображення будівлі і натискаємо кнопку «Predict»

Продовження таблиці 5.11

Очікуваний результат:	На сторінці відображається анімація завантаження результатів; коли аналіз завершується, результат аналізу архітектурних стилів відображений; на сторінці відображена форма для вводу детальної інформації про аналіз для збереження; на сторінці відображена кнопка для збереження результату
Фактичний результат співпадає з очікуваним:	Так

Таблиця 5.12 – Тестування виходу з системи користувача

Тест:	Вихід з системи має видалити дані поточного користувача
Початковий стан системи:	Відкрита домашня сторінка; користувач авторизований
Дія:	Натискаємо кнопку вихід з системи
Очікуваний результат:	В меню не відображається ім'я користувача, посилання на сторінку збережених результатів та кнопка вихід з системи; відображається посилання на сторінку авторизації; в dev tools веб-браузера, в вкладці application, розділ local storage для домену http://localhost:4200 не існує значення з ключем «rawToken»
Фактичний результат співпадає з очікуваним:	Так

Таблиця 5.13 – Тестування сортування результатів за назвою

Тест:	Кнопка сортування за назвою має сортувати збережені результати
Початковий стан системи:	Відкрита домашня сторінка; користувач авторизований
Дія:	Зберегти 2 результати аналізу з різним значенням назви; перейти на сторінку збережених результатів і відсортувати за назвою
Очікуваний результат:	У таблиці збережених результатів рядки відсортовані в алфавітному порядку
Фактичний результат співпадає з очікуваним:	Так

Таблиця 5.14 – Тестування сортування результатів за назвою стилю

Тест:	Кнопка сортування за назвою стилю має сортувати збережені результати
Початковий стан системи:	Відкрита домашня сторінка; користувач авторизований
Дія:	Зберегти 2 результати аналізу з різним архітектурним стилем; перейти на сторінку збережених результатів і відсортувати за назвою стилю
Очікуваний результат:	У таблиці збережених результатів рядки відсортовані в алфавітному порядку за значенням архітектурного стилю
Фактичний результат співпадає з очікуваним:	Так

Таблиця 5.15 – Тестування сортування результатів за датою проведення аналізу

Тест:	Кнопка сортування за датою проведення аналізу має сортувати збережені результати
Початковий стан системи:	Відкрита домашня сторінка; користувач авторизований
Дія:	Зберегти 2 результати аналізу з інтервалом в більш ніж хвилина; перейти на сторінку збережених результатів і відсортувати за датою проведення аналізу
Очікуваний результат:	У таблиці збережених результатів рядки відсортовані в порядку зростання дати
Фактичний результат співпадає з очікуваним:	Так

Таблиця 5.16 – Тестування сортування результатів за датою зображення

Тест:	Кнопка сортування за датою зображення має сортувати збережені результати
Початковий стан системи:	Відкрита домашня сторінка; користувач авторизований
Дія:	Зберегти 2 результати аналізу з різними датами, коли було зроблене зображення; перейти на сторінку збережених результатів і відсортувати за датою зображення
Очікуваний результат:	У таблиці збережених результатів рядки відсортовані в порядку зростання дати зображення
Фактичний результат співпадає з очікуваним:	Так

6 МЕТОДИ ВИПРОБУВАНЬ

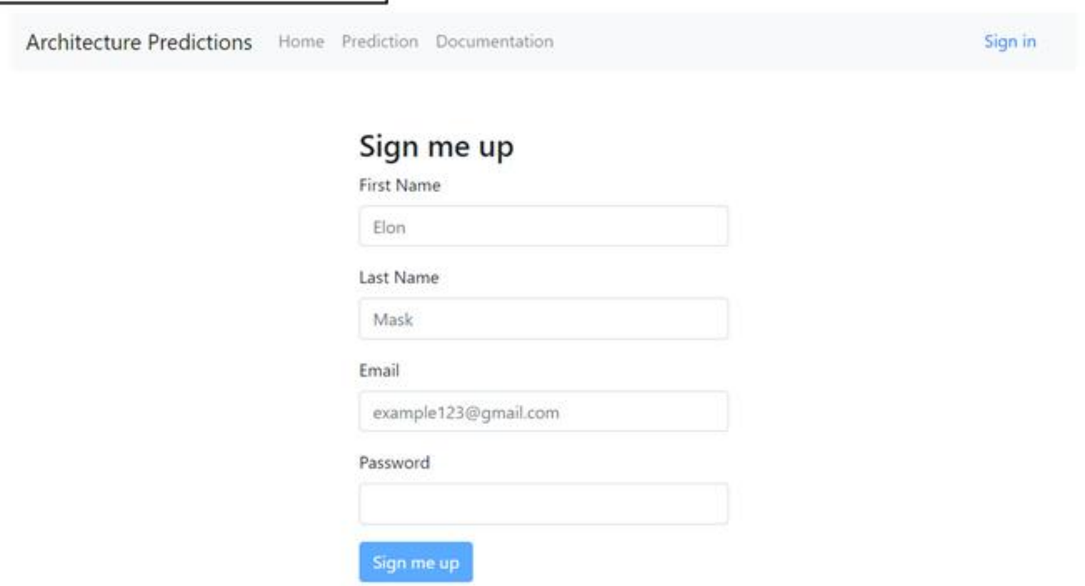
Метод даних випробувань – ручне тестування на інтеграційному рівні. Всі випробування пройдено успішно.

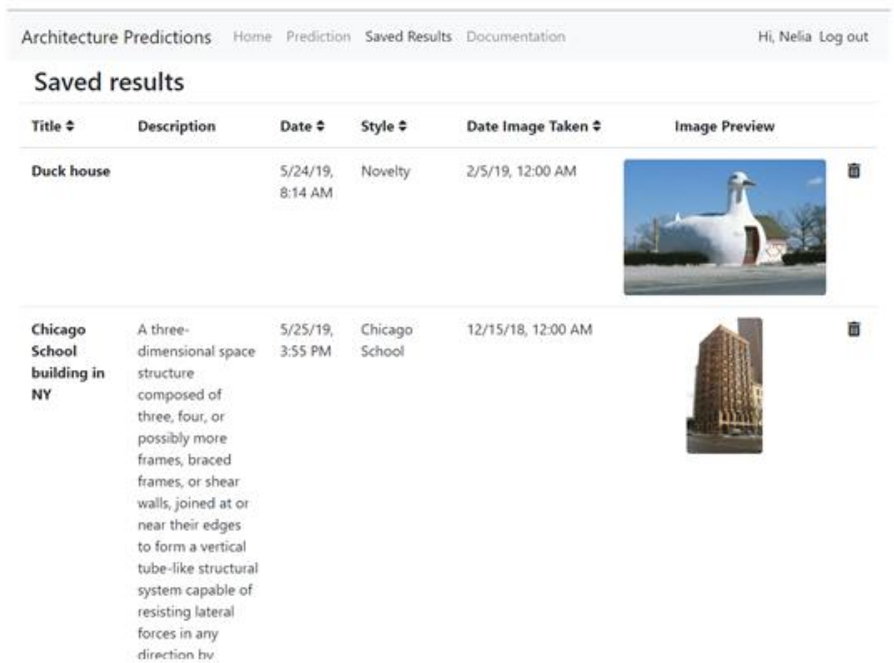
Графічний матеріал
до дипломного проекту

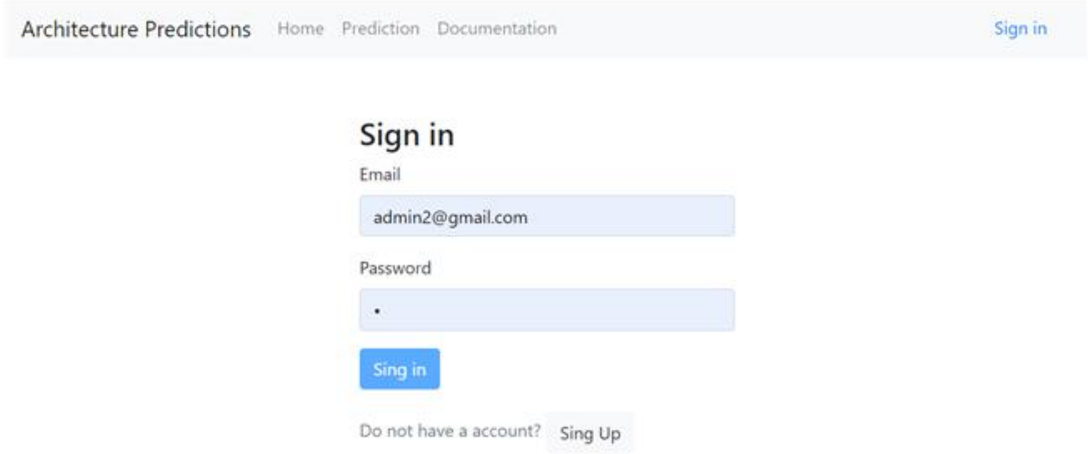
на тему: Система розпізнавання архітектурних стилів будівель за зображеннями

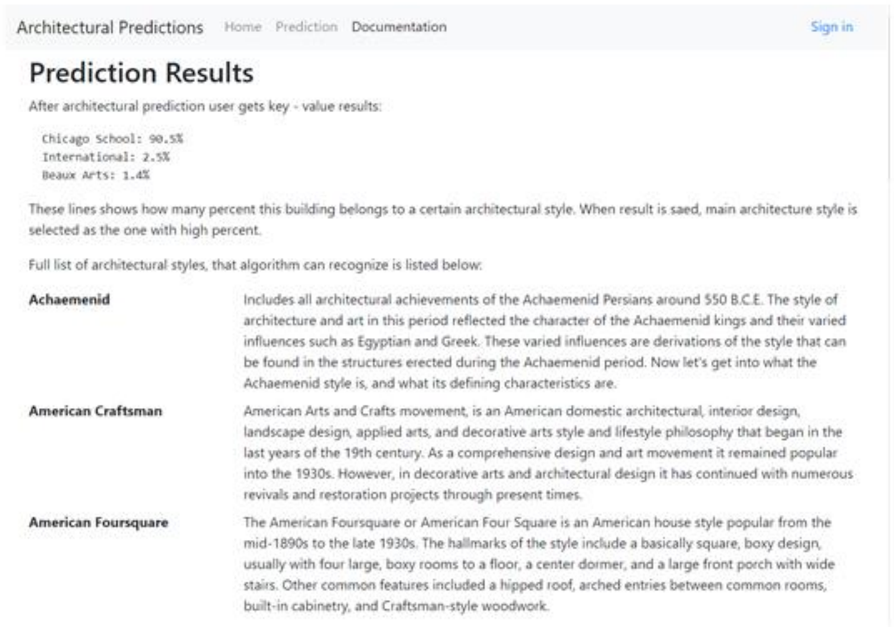
Київ – 2019 року

ДП ІС-5216.1181-с.КЕ

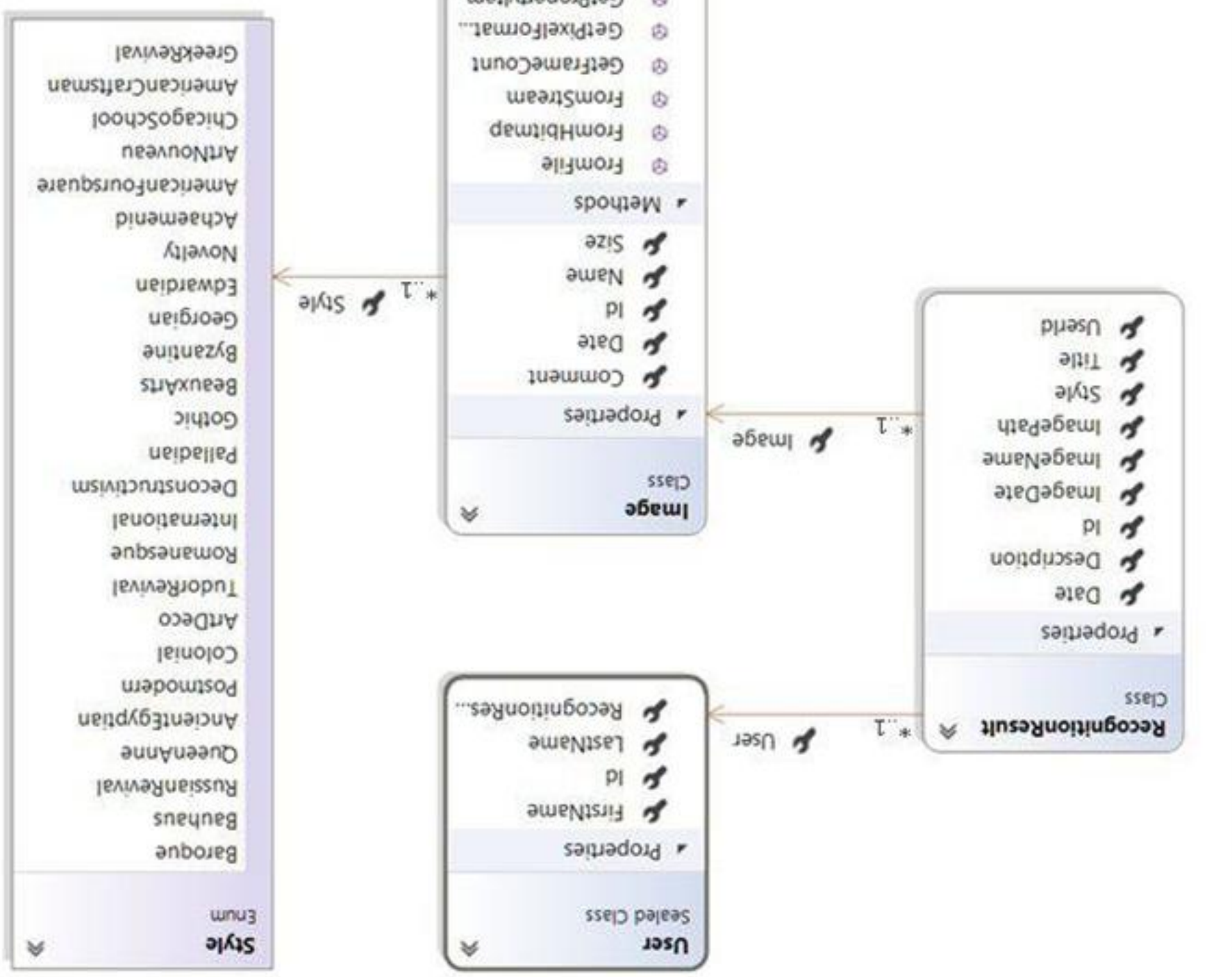








					ДП ІС-5219.1181-с.КЕ		
						Літера	Маса / Масштаб
Зм.	Арк.	№ документа	Підпис	Дата	Креслення вигляду екранних форм		
Розробив		Новіченко Н.В.				Аркуш 2	Аркушів 2
Перевірив		Сперкач М.О.					
Т. кон.					Система розпізнавання архітектурних стилів будівель за зображеннями	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-52	
Н. кон.		Халус О.А.					
Затвердив		Сперкач М.О.					



ДП ІС-5219.1181-с.ССК

Зм.	Арк	№ докумeнта	Підпис	Дата	ДП ІС-5219.1181-с.ССК		
Розробив		Новіченко Н.В.			Структурна схема класів програмного забезпечення	Літера	Маса	Масштаб
Перевірив		Снєжкін М.О.				Аркуш 1	Аркушів 1	
Т. контр.								
Н. контр.		Харус О.А.			Система розпізнавання архітектурних спливів будівель за зображеннями	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-52		
Затвердив		Снєжкін М.О.						

ДП ІС-5216.1181-с.КЕ

Architecture Predictions Home Prediction Documentation Sign in

Hi there! Try out architecture styles prediction by uploading your building image!

18_337px-Wyandotte_Building_in_Columbus.jpg Browse Predict style

Chicago School: 90.5%
International: 2.5%
Beaux Arts: 1.4%

Fill in details

Title:

Chicago School building in NY

Description:

A three-dimensional space structure composed of three, four, or possibly more frames, braced frames, or shear walls, joined at or near their edges to form a vertical tube-like structural system capable of resisting lateral forces in any direction by cantilevering from the foundation.

Date image taken:

12/15/2018

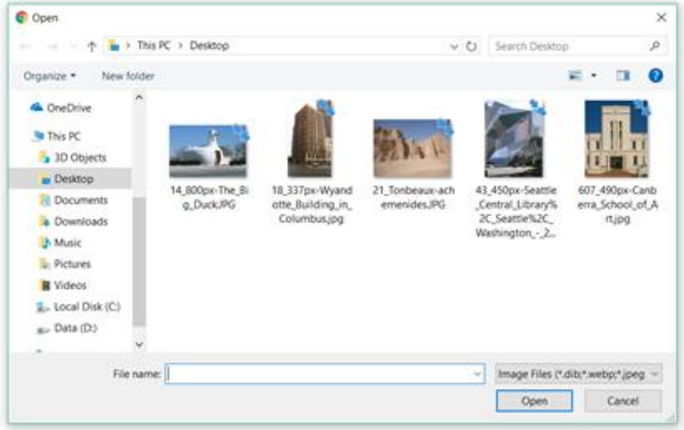
Save results

Architecture Predictions Home Prediction Documentation Sign in

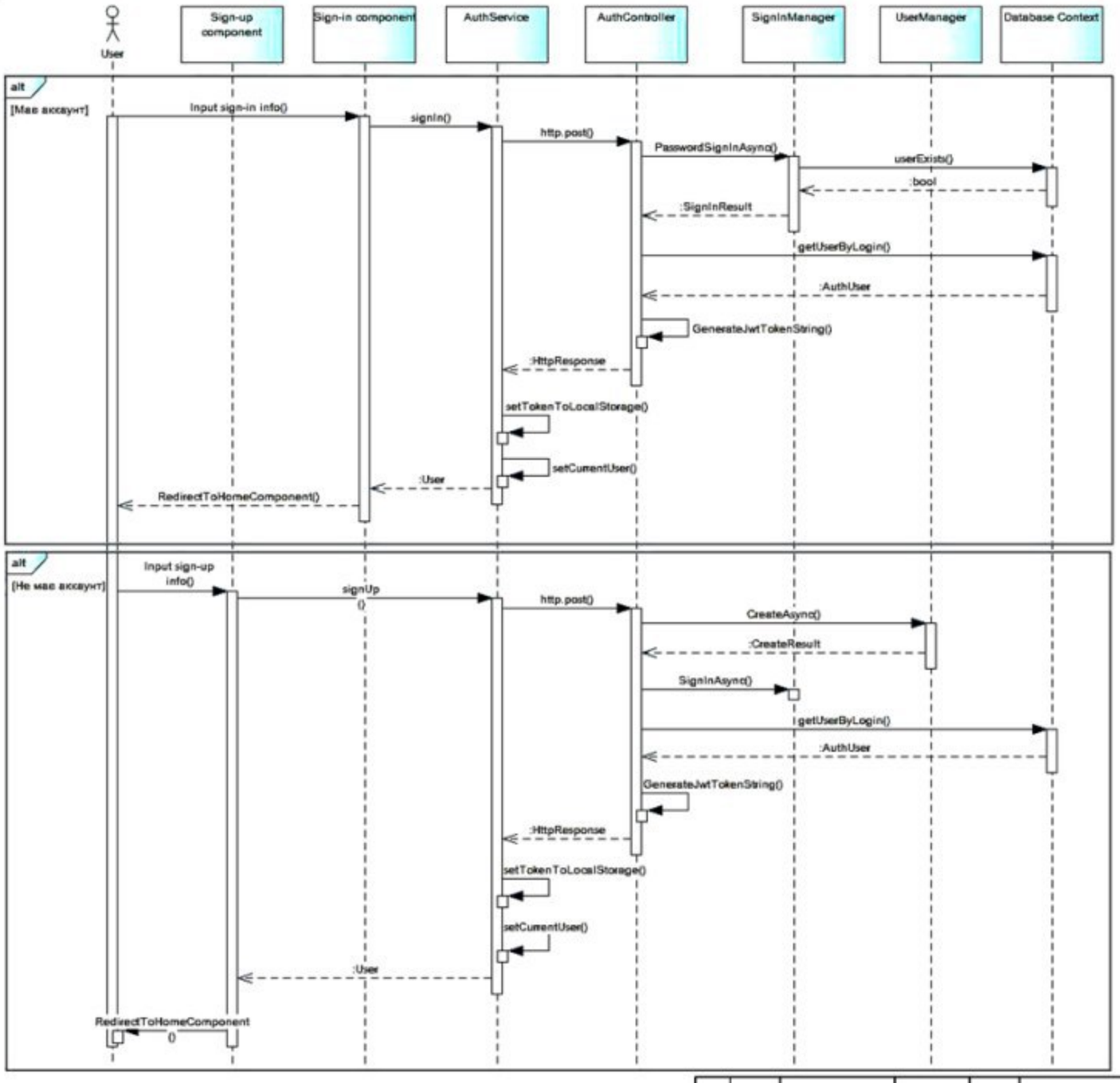
Architecture Predictions Home Prediction Documentation Sign in

Hi there! Try out architecture styles prediction by uploading your building image!

Choose file Browse Predict style

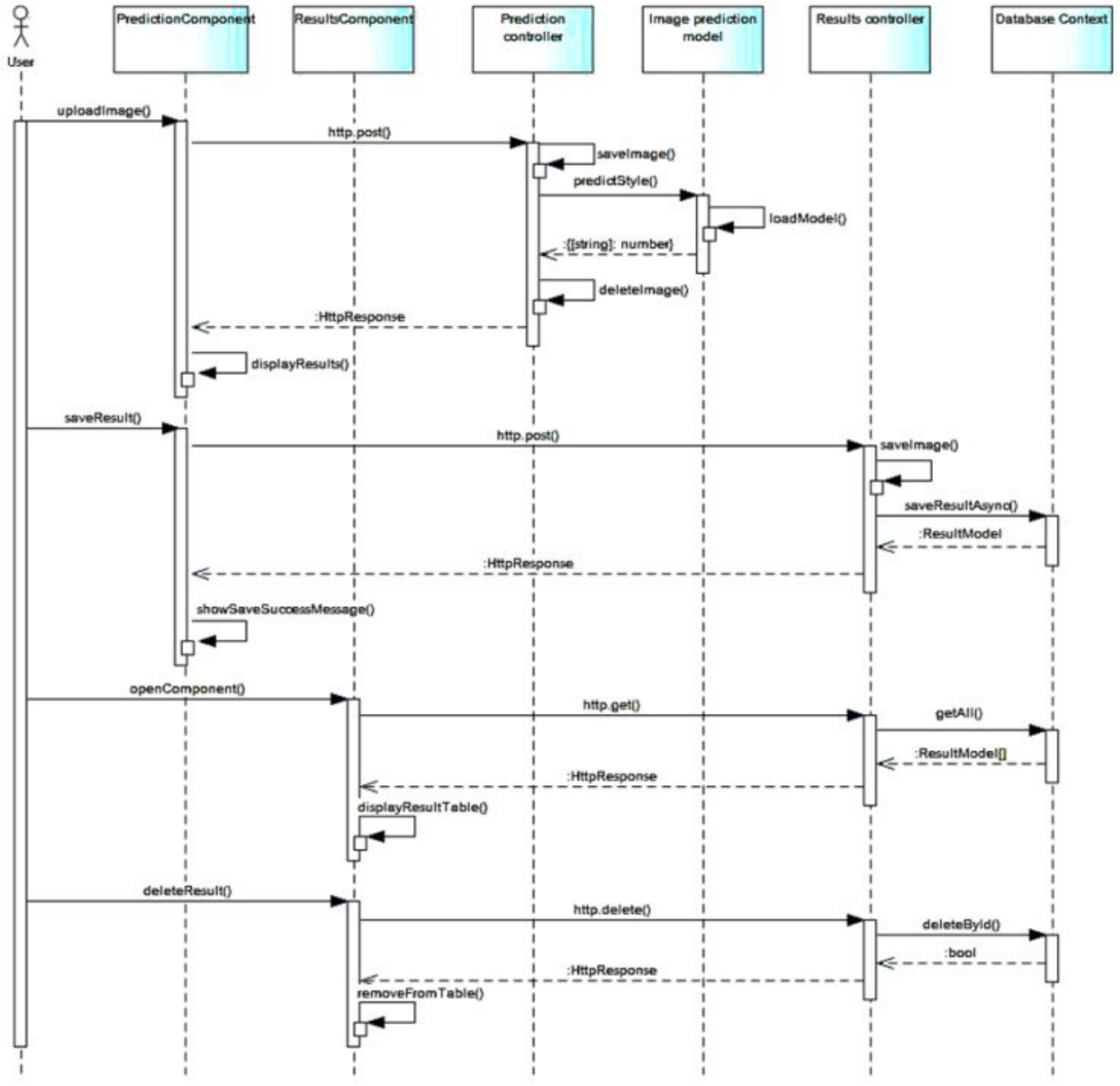


					ДП ІС-5219.1181-с.КЕ	
Зм.	Арк.	№ документа	Підпис	Дата	Креслення вигляду екранних форм	Літера / Маса / Масштаб
Розробив		Новіченко Н.В.				
Перевірив		Сперкач М.О.				Аркуш 1 / Аркушів 2
Т. кон.					Система розпізнавання архітектурних стилів будівель за зображеннями	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-52
Н. кон.		Халус О.А.				
Затвердив		Сперкач М.О.				



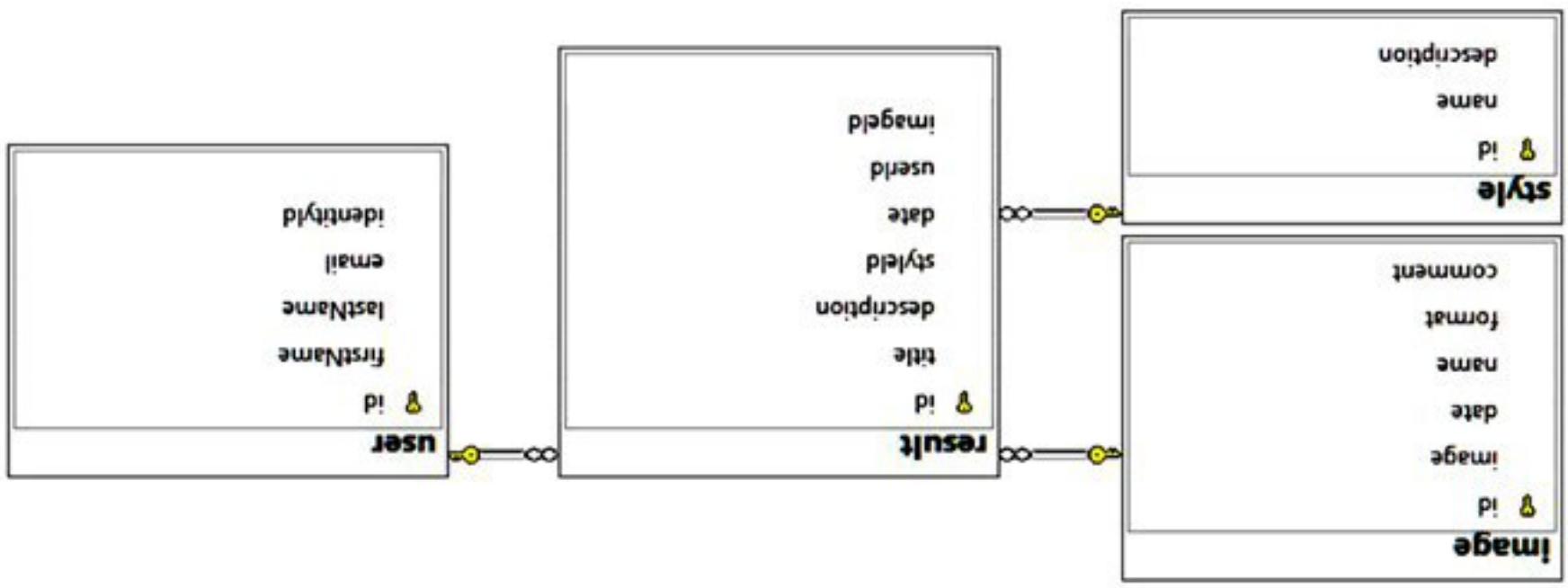
					ДП ІС-5219.1181-с.ССП	
Зм.	Арк.	№ документа	Підпис	Дата	Структурна схема послідовності	Літера / Маса / Масштаб
Розробив		Новіченко Н.В.				
Перевірив		Сперкач М.О.				Аркуш 1 / Аркушів 2
Т. кон.					Система розпізнавання архітектурних стилів будівель за зображеннями	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-52
Н. кон.		Халус О.А.				
Затвердив		Сперкач М.О.				

ДП ІС-5216.1181-с.ССП



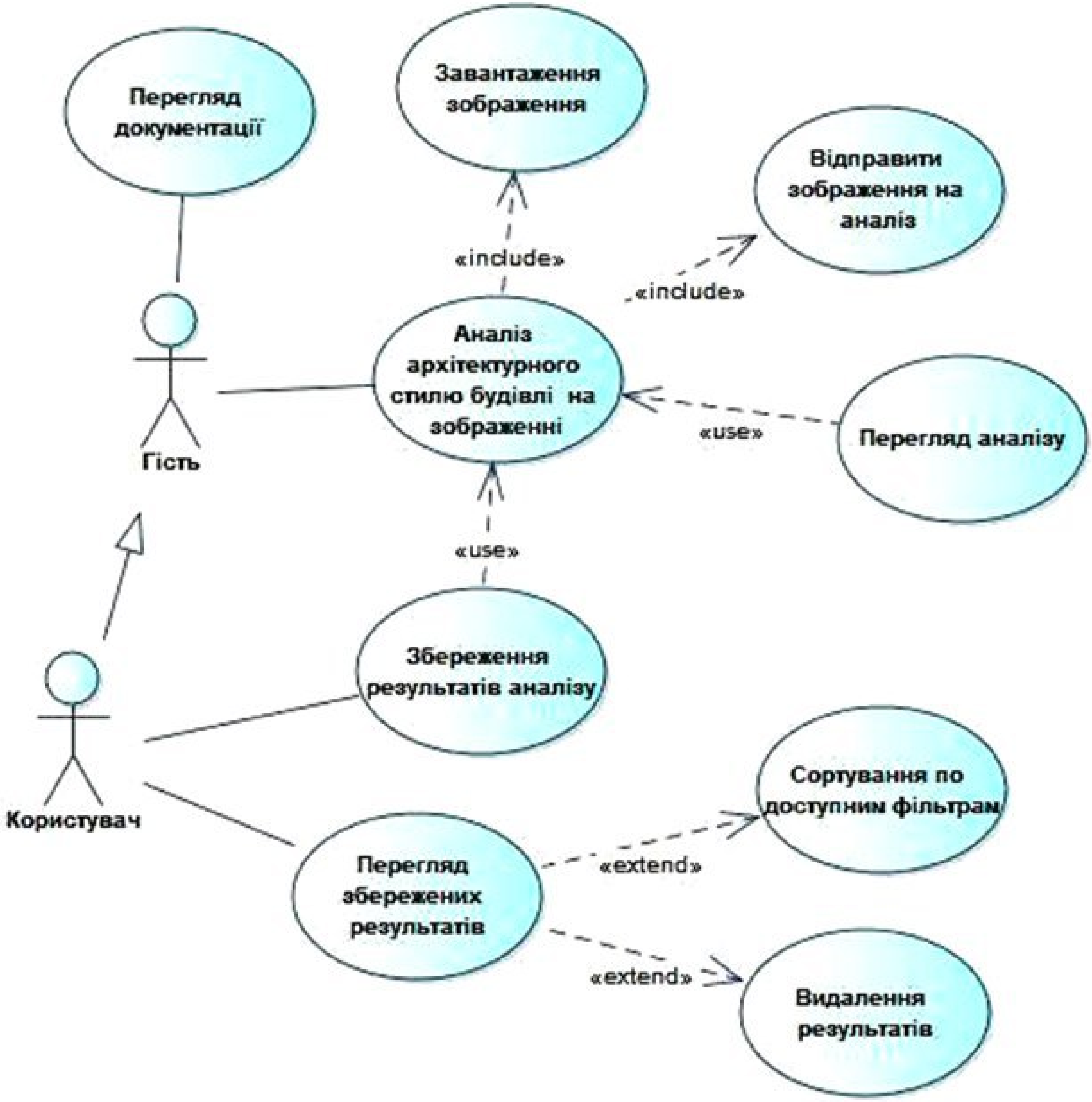
					ДП ІС-5219.1181-с.ССП	
Зм.	Арк.	№ документа	Підпис	Дата	Структурна схема послідовності	Літера / Маса / Масштаб
Розробив		Новіченко Н.В.				
Перевірив		Сперкач М.О.				Аркуш 2 / Аркушів 2
Т. кон.					Система розпізнавання архітектурних стилів будівель за зображеннями	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-52
Н. кон.		Халус О.А.				
Затвердив		Сперкач М.О.				

Зм.	Арк	№ докумeнта	Пiдпис	Дата	ДП ІС-5219.1181-с.СБД				
Розробив		Новiченко Н.В.			Схема бази даних		Лiтера	Маса	Масштаб
Перевiрив		Снєжкін М.О.					Аркуш 1		Аркушів 1
Т. конс.									
Н. конт.		Харус О.А.			Системa розпізнавання архітектурних спецiй будівель за зображеннями		КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-52		
Затвердив		Снєжкін М.О.							



ДП ІС-5219.1181-с.СБД

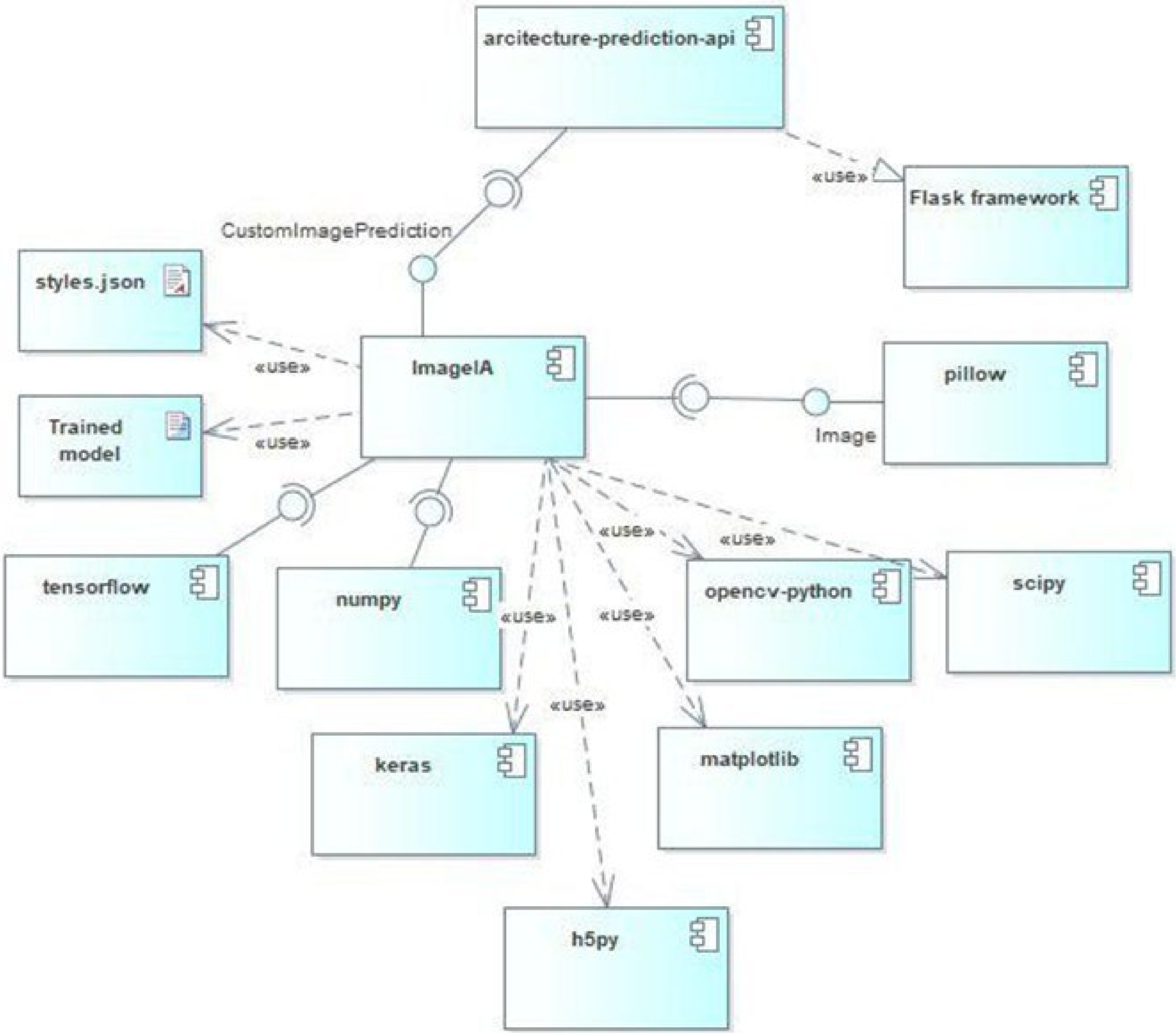
ДП ІС-5219.1181-с.ССВ



					ДП ІС-5219.1181-с.ССВ
Зм.	Арк.	№ документа	Підпис	Дата	
Розробив		Новіченко Н.В.			*Структурна схема варіантів використань*
Перевірив		Сперкач М.О.			Літера / Маса / Масштаб
Т. кон.					Аркуш 1 / Аркушів 1
Н. кон.		Халус О.А.			*Система розпізнавання архітектурних стилів будівель за зображеннями*
Затвердив		Сперкач М.О.			*КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-52*

					ДП ІС-5219.1181-с.ССД		
						Літера	Маса	Масштаб
Зм.	Арк.	№ документа	Підпис	Дата	Структурна схема діяльності			
Розробив		Новіченко Н.В.				Аркуш 1	Аркушів 1	
Перевірив		Сперкач М.О.						
Т. кон.					Система розпізнавання архітектурних стилів будівель за зображеннями	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-52		
Н. кон.		Халус О.А.						
Затвердив		Сперкач М.О.						

ДП ІС-5219.1181-с.ССК



					ДП ІС-5219.1181-с.ССК		
Зм.	Арк.	№ документа	Підпис	Дата	Структурна схема компонентів програмного забезпечення	Літера	Маса	Масштаб
Розробив		Новіченко Н.В.						
Перевірив		Сперкач М.О.				Аркуш 1	Аркушів 1	
Т. кон.					Система розпізнавання архітектурних стилів будівель за зображеннями	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-52		
Н. кон.		Халус О.А.						
Затвердив		Сперкач М.О.						