Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра системного аналізу та інформаційних технологій

Бакалаврська кваліфікаційна робота на тему:

«ІНФОРМАЦІЙНА РОЗВАЖАЛЬНА СИСТЕМА РОЗПІЗНАВАННЯ ЕМОЦІЙ»

Виконала: студентка 4 курсу, групи 2ІСТ-21б спеціальності 126 – «Інформаційні системи та технології»
_____ Софія ФІЯЛО
Керівник: к.т.н., доц. каф. САІТ
_____ Георгій ГОРЯЧЕВ
«04» 06 2025 р.
Рецензент: доц. каф. АІІТ
_____ Ілона БОГАЧ
«12» 06 2025 р.

Допущено до захисту

Завідувач кафедри САІТ

_____ д.т.н., проф. Віталій МОКІН

«06» 06 2025 р.

Вінниця ВНТУ – 2025 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра системного аналізу та інформаційних технологій
Рівень вищої освіти – перший (бакалаврський)
Галузь знань – 12 Інформаційні технології
Спеціальність 126 – «Інформаційні системи та технології»
Освітньо-професійна програма – Прикладні інформаційні технології

ЗАТВЕРДЖУЮ
Завідувач кафедри САІТ
д.т.н., проф. Віталій МОКІН
"28" 03 2025 року

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТКІ
Фіяло Софії Геннадіївні

1. Тема роботи: «Інформаційна розважальна система розпізнавання емоцій»
керівник роботи: Горячев Г. В., к.т.н., доц. каф. САІТ
затверджені наказом ВНТУ від «20» 03 2025 року №97
2. Термін подання студентом роботи 31.05.2025 р.
3. Вихідні дані до роботи:
1) Відеопотік з камери користувача для виявлення обличчя та визначення емоцій.
2) Перелік базових емоцій для розпізнавання (радість, сум, злість, подив тощо).
3) Вимоги до форматів виводу результату (текст, графік, озвучення).
4) Збереженні зображення з найкращими емоціями користувач
5) Локальна база даних з статистикою та результатами розпізнавання користувача.
4. Зміст текстової частини:
Аналіз проблематики інформаційної системи розпізнавання емоцій, IT-інфраструктура та вибір технологій, Проєктування та розробка інформпційної розважальної системи розпізнавання емоцій.
5. Перелік ілюстративного матеріалу:
1) Контекстні діаграми;
2) Архітектура інформаційної системи;
3) Діаграма USE CASE інформаційної системи;.
4) Блок-схема алгоритму роботи програми

6. Консультанти розділів роботи:

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

7. Дата видачі завдання 28.03.2025р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Аналіз проблематики інформаційної системи розпізнавання емоцій	01.04.2025	18.04.2025	
2	IT-інфраструктура та вибір технологій	18.04.2025	26.04.2025	
3	Проєктування та розробка інформаційної системи розпізнавання емоцій	27.04.2025	19.05.2025	
4	Висновки	19.05.2025	20.05.2025	
5	Оформлення матеріалів до захисту БДР	22.05.2025	03.06.2025	
6	Попередній захист БКР, доопрацювання, рецензування БКР	08.06.2025	10.06.2025	
7	Захист БКР ЕК	16.06.2025	17.06.2025	

Студент _____ Софія ФІЯЛО

Керівник роботи _____ Георгій ГОРЯЧЕВ

АНОТАЦІЯ

Бакалаврська кваліфікаційна робота складається з 84 сторінок формату А4, на яких є 36 рисунків, в списку використаних джерел міститься 27 найменування.

Мета даної роботи є розробка інформаційної розважальної системи, що здійснює розпізнавання емоцій користувача в режимі реального часу. Під час виконання роботи було досліджено сучасні підходи до визначення емоцій за відеопотоком з камери, а також можливості інтеграції таких методів у інтерфейс розважального мобільного застосунку. Розроблено програмну систему, яка виконує автоматичне виявлення обличчя, розпізнавання емоцій, збереження результатів та візуалізацію статистики. Окрім цього, система забезпечує голосове озвучення результатів та ігрову взаємодію з користувачем. Основу розпізнавання складає використання бібліотеки DeepFace у поєднанні з Python та графічним середовищем PyQt5.

Результати розробки демонструють можливість побудови легкого, але функціонального застосунку, орієнтованого на підвищення емоційного залучення користувача через візуальну й аудіоінтеракцію.

Ключові слова: розпізнавання емоцій, інформаційна система, DeepFace, PyQt5, розважальний застосунок, відеоаналіз, статистика, голосове озвучення.

ABSTRACT

Bachelor's qualification work consists of 84 pages of the A4 format, on which there are 36 drawings, the list of sources used contains 27 names.

The purpose of this work is to develop an information entertainment system that recognizes the user's emotions in real time. During the work, modern approaches to determining emotions from the video stream from the camera were investigated, as well as the possibility of integrating such methods into the interface of an entertaining mobile application. A software system has been developed that automatically detects faces, recognizes emotions, saves results and visualizes statistics. In addition, the system provides voice dubbing of results and game interaction with the user. The basis of recognition is the use of the DeepFace library in combination with Python and the PyQt5 graphical environment.

The results of the development demonstrate the possibility of building a lightweight but functional application aimed at increasing the user's emotional involvement through visual and audio interaction.

Keywords: emotion recognition, information system, DeepFace, PyQt5, entertainment application, video analysis, statistics, voice dubbing.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасному цифровому середовищі багато уваги приділяється персоналізації взаємодії між користувачем та інформаційними системами. Одним із перспективних напрямів є використання емоційного стану користувача як елемента зворотного зв'язку. Розпізнавання емоцій в реальному часі дозволяє покращити інтерактивність програмного забезпечення, особливо у сфері розважальних застосунків. Такі системи можуть не лише розважати, а й адаптувати свій контент відповідно до настрою користувача, що значно підвищує залучення та позитивне сприйняття. Саме тому розробка інформаційної розважальної системи з функцією автоматичного розпізнавання емоцій є актуальним завданням, яке поєднує технічну складову з психологічними аспектами взаємодії людини і комп'ютера.

Мета і задачі дослідження. Метою дослідження є розробка інформаційної розважальної системи, яка розпізнає емоційний стан користувача на основі зображення з вебкамери, озвучує результат і візуалізує статистику емоцій. Для досягнення мети необхідно вирішити наступні задачі:

- Проаналізувати сучасні методи визначення емоцій за зображенням обличчя;
- Обрати програмні засоби для розпізнавання емоцій;
- Розробити архітектуру системи, що дає змогу інтерактивно взаємодіяти з користувачем;
- Реалізувати програму з графічним інтерфейсом;
- Здійснити збереження результатів та виведення статистики про користувача.

Об'єктом дослідження є процес взаємодії користувача з інформаційною системою, що включає в себе реакцію на емоції користувача.

Предметом дослідження є методи та засоби програмної реалізації розпізнавання людських емоцій на основі зображення обличчя в реальному часі. Також інтерфейс системи, що дає змогу забезпечити зворотний зв'язок у розважальному форматі.

Публікації. За результатами виконання даної роботи опубліковано тези доповідей на тему: «Інтерактивна інформаційно-розважальна система розпізнавання емоцій на основі виразу обличчя» Міжнародна науково-практична інтернет-конференція *«Молодь в науці: дослідження, проблеми, перспективи»* (Вінниця, 2025 р.) [1].

1 АНАЛІЗ ПРОБЛЕМАТИКИ ІНФОРМАЦІЙНОЇ СИСТЕМИ РОЗПІЗНАВАННЯ ЕМОЦІЙ

1.1 Аналіз проблематики системи розпізнавання емоцій

Інформаційно-розважальні технології постійно розширюється охоплюючи нові методи взаємодії з користувачем. Одним з сучасних напрямків є розробка нових систем, які реагують не лише на дії користувача, а й на його емоційний стан. Завдяки цьому формується більш індивідуальний досвід, підвищувати зацікавленість до взаємодії та залученість у процес.

Додатковою складністю є завдання розпізнавання емоцій в умовах щоденного використання персонального комп'ютера або ноутбука. Не всі користувачі мають ідеальні умови для зйомки: освітлення, фон, якість веб-камери – все це впливає на точність результатів. Самі емоції проявляються у кожної людини по-своєму [2].

При створенні таких систем основною проблемою є поєднання швидкості й точності. З одного боку необхідно, щоб зображення оброблялось майже миттєво – у реальному часі [3]. Але водночас важливо, щоб система не помилялась під час розпізнавання емоцій. Для цього потрібні справді добре налаштовані алгоритми, які здатні швидко знайти обличчя на кадрі та зрозуміти його вираз.

У випадку з розважальними застосунками недостатньо просто «побачити» емоцію – потрібно якось на неї реагувати. Наприклад, програма може її озвучити або занести в статистику [4]. Отже, мова не лише про технічне виявлення, а й у продуманій взаємодії з користувачем, яка виглядає логічною й природною.

Окремою проблемою є подальше використання розпізнаної інформації. У випадку з розважальними застосунками недостатньо просто «побачити» емоцію – потрібно якось на неї реагувати наприклад, озвучити її або зберегти в статистику для подальшої обробки. Завдання виходить за межі технічного розпізнавання та охоплює проєктування повноцінної логіки взаємодії. Для реалізації системи розпізнавання емоцій важливо чітко окреслити основні етапи

її роботи — від обробки вхідного зображення до аналізу виразу обличчя та формування результату. На рисунку 1.1 наведено загальну блок-схему, яка ілюструє послідовність дій, що виконує система.

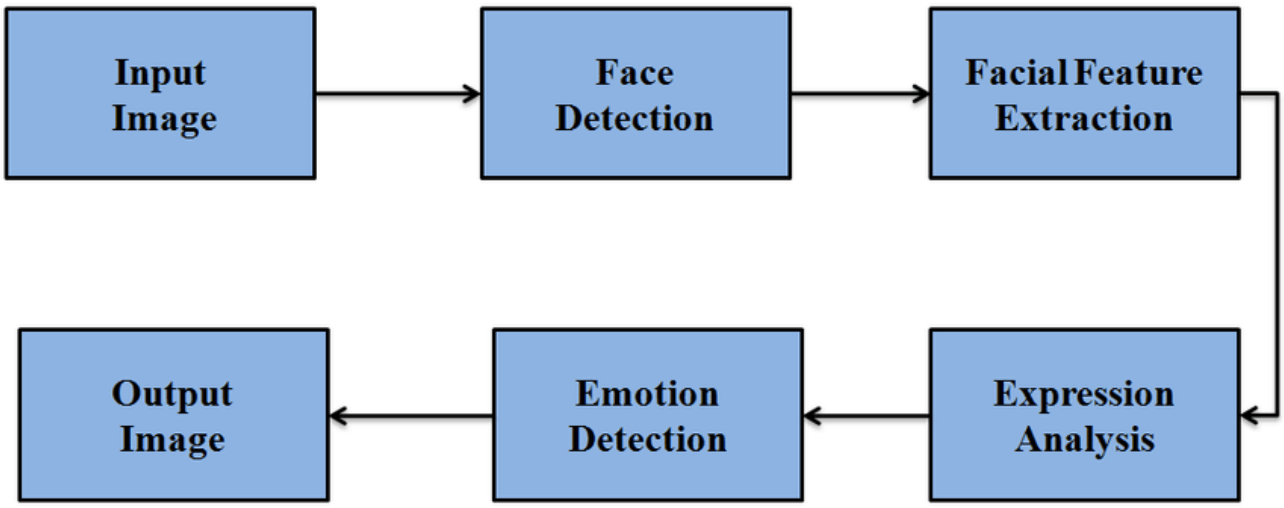

Рисунок 1.1 - Загальна схема роботи системи розпізнавання емоцій за зображенням обличчя

1.2 Аналіз ринку інформаційних систем з елементами емоційної взаємодії

Користувачі очікують не лише коректної роботи програмного забезпечення, а й викликали емоції. При цьому важливими є не лише технічні характеристики, а й зручність та привабливість використання системи. Саме тому за останні роки все більше з'являється програм, які намагаються враховувати емоційний стан користувача і будувати з ним більш живу взаємодію.

1.2.1 Глобальні тенденції

Великі компанії, такі як Microsoft, IBM, Google та Amazon, давно працюють над розпізнаванням емоцій. Вони інвестують у це великі кошти, особливо в напрямки, де це поєднується з машинним навчанням, біометрією та

нейромережами. Наприклад, у IBM Watson є інструменти для аналізу емоцій у текстах, а в Amazon Rekognition – для роботи з відео.

Такі розробки вже знаходять застосування в різних сферах наприклад: маркетинг, медицина, розваги та інші. У розважальних програмах, наприклад ігри, де персонажі реагують на емоції гравця, або застосунки, що змінюють музику та візуальний супровід відповідно до виразу обличчя користувача.

Графік зростання ринку:

- За оцінками, у 2022 році світовий ринок технологій, що пов'язані з розпізнаванням емоцій, перевищив позначку в 23 мільярди доларів США.
- Прогнози свідчать, що до 2030 року цей показник може зрости більш ніж у чотири рази – до понад 90 мільярдів доларів.
- Щорічне зростання ринку становить у середньому понад 20%, що робить цей напрям одним із найдинамічніших в IT-сфері [5].

Світовий ринок технологій розпізнавання емоцій стрімко розвивається, що зумовлено зростаючим попитом на персоналізовані цифрові рішення в освіті, медицині, маркетингу та розвагах. На рисунку 1.2 представлено прогноз динаміки зростання ринку компонентів систем розпізнавання емоцій (програмне забезпечення та послуги) в період з 2020 по 2030 роки.

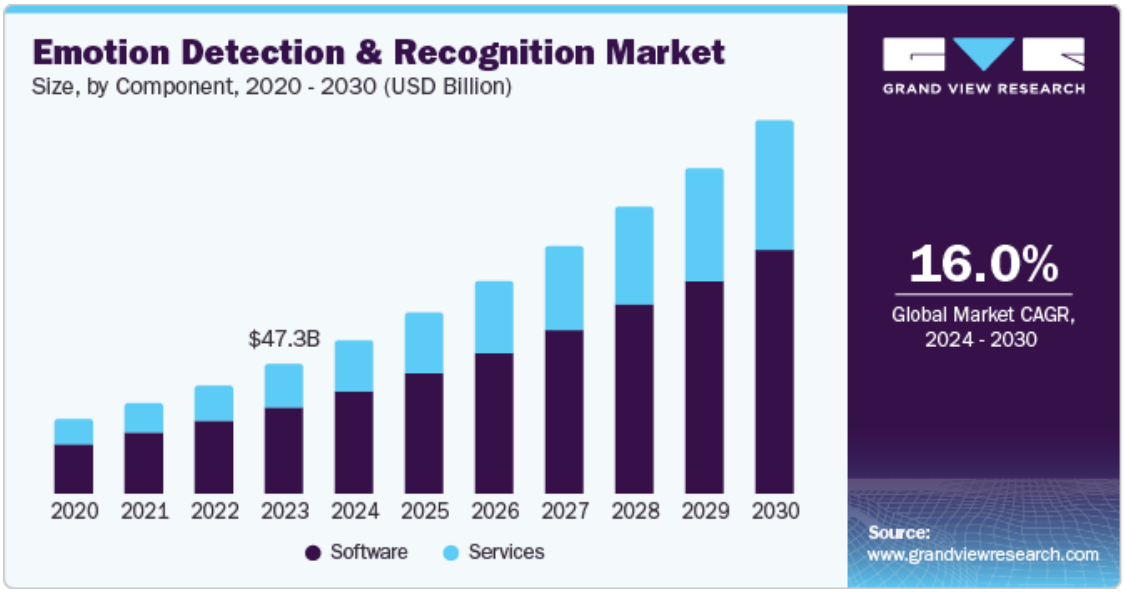

Рисунок 1.2 - Динаміка зростання світового ринку технологій розпізнавання емоцій

1.2.2 Стан українського ринку

В Україні подібні технології тільки починають використовуватись у бізнесі та розробці. Наявні проєкти здебільшого зосереджені в галузі освіти, психології, телемедицини та гейміфікованого навчання. В розважальному секторі переважають спроби створення мобільних додатків або веб-платформ для роботи з емоціями, але повноцінні програмні продукти з локальною обробкою та зворотним зв’язком (озвученням, статистикою) є рідкістю.

Для більшості студентських і освітніх проєктів великою перевагою є можливість реалізації таких систем із відкритими бібліотеками – наприклад, DeepFace, OpenCV, PyQt, gTTS тощо. Це дає змогу створити повноцінний застосунок навіть без складної серверної архітектури. Сьогодні на ринку існує низка рішень для розпізнавання емоцій, які відрізняються за ліцензією, платформою використання та можливостями інтеграції. На рисунку 1.3 наведено порівняння кількох популярних систем, що дозволяє обрати оптимальний інструмент залежно від потреб користувача та специфіки проєкту.

Система	Тип	Платформа	Розпізнає емоції	Працює офлайн	Потребує інтернет	Безкоштовність
Affectiva	Комерційна	Web/API	7+	Ні	Так	Ні
FaceReader	Комерційна	Desktop	6+	Так	Ні	Ні
DeepFace	Open Source	Python	7	Так	Ні	Так
Azure Emotion API	Хмарна	Azure Cloud	8+	Ні	Так	Ні (обмежено)

Рисунок 1.3 – Порівняльна таблиця популярних рішень для емоційного розпізнавання

1.2.3 Основні недоліки комерційних рішень

Незважаючи на те, що великі платформи пропонують потужні інструменти для розпізнавання емоцій, вони не позбавлені недоліків.

Більшість таких сервісів є платними та мають обмежену кількість запитів.

Їх робота залежить від підключення до Інтернету потрібно передавати дані на сервер і очікувати відповіді, що займає не мало часу. Також не варто забувати про конфіденційність, передача зображень, файлів на сторонні сервери часто викликає сумніви у безпеці, так як не відомо куди попадуть нашлі картинки в подальшому.

Якщо сервіс має технічний збій, додаток просто не зможе працювати, бо повністю залежить від хмари і потрібен час на те, щоб його полагодити.

У зв'язку з цим створення локального автономного застосунку без використання сторонніх API є доцільним рішенням. Це не лише знижує ризики, пов'язані з передачею даних, а й дає змогу зберігати повну незалежність від зовнішніх платформ [6].

1.2.4 Переваги розробки власної системи

Розробка персонального застосунку має кілька вагомих плюсів:

- Повна гнучкість у побудові логіки взаємодії з користувачем.
- Можливість адаптувати інтерфейс під конкретну цільову аудиторію.
- Відсутність витрат на сторонні API.
- Розширюваність: у майбутньому можна додати інші емоційні реакції, голосові помічники, збереження історії тощо.

Така система може використовуватись не лише в розвагах, а й у психоемоційному моніторингу, в освітніх програмах для дітей, при розробці інтерактивних навчальних модулів.

1.3 Огляд існуючих інформаційних систем з функцією розпізнавання емоцій

У світі існують системи, які інтегрують функціонал емоційної взаємодії посилаючись на аналіз зображення обличчя або голосу. Переважна частина таких рішень використовується у сферах маркетингу, освіти та охорони здоров'я однак деякі знаходять застосування в програмному забезпеченні з елементами

емоційної взаємодії для підвищення залученості користувачів. Розглянемо найбільш поширені приклади даних систем.

Affectiva – це платформа, яка надає API для розпізнання емоційного стану користувача на основі аналізу обличчя або тональності голосу. Система може розрізняти до 20 параметрів виразу обличчя, а саме радість, злість, здивування, смуток та інше. Основна цільова аудиторія рекламні агентства та автомобільна індустрія. Для локального використання ця система не підходить, оскільки обробка зображення здійснюється через хмарний сервіс [7]. Серед відомих рішень для розпізнавання емоцій можна виокремити комерційну платформу Affectiva, яка використовує аналіз обличчя в реальному часі. На рисунку 1.4 показано приклад роботи цієї системи при обробці зображення користувача, де відображаються виявлені емоційні параметри та мімічні ознаки.

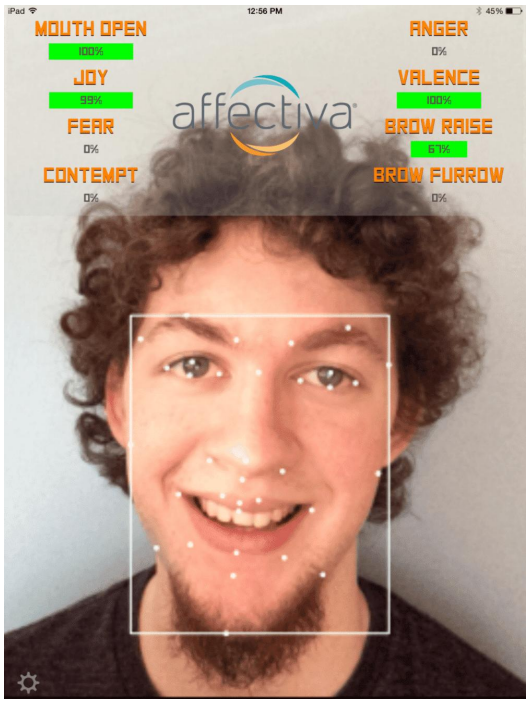

Рисунок 1.4 – Приклад використання Affectiva для аналізу емоцій

FaceReader – це програминие забезпечення для настільних комп'ютерів, що широко використовується у наукових дослідженнях. Програма дозволяє в режимі реального часу аналізувати вирази обличчя, визначати рівень залучення, уваги та навіть оцінювати пульс. Розробка орієнтована на дослідницьке середовище тому вимагає серйозного апаратного забезпечення. Інтерфейс

складний для пересічного користувача, а ціна досить висока, що обмежує її застосування в побуті [8]. Ще одним відомим рішенням для аналізу емоцій є програма FaceReader, яка дозволяє деталізовано аналізувати міміку обличчя користувача, визначати не лише емоції, а й активність окремих м’язів, рівень емоційного збудження та інші показники. На рисунку 1.5 представлено приклад інтерфейсу цієї системи з візуалізацією емоційних реакцій.

Рисунок 1.5 – Інтерфейс програми FaceReader

Microsoft Azure Emotion API – це хмарне рішення яке було представлено компанією Microsoft, що входить до складу. Дозволяє виконувати емоційний аналіз зображень або відео з допомогою попередньо натренованих моделей. Результати передаються у вигляді відсоткового ймовірнісного розподілу за кожною з базових емоцій. Потребує постійного інтернет-з’єднання, ключа доступу та облікового запису Microsoft. Має обмежену кількість безкоштовних запитів на місяць [9-10]. Одним із рішень, яке пропонує хмарну інфраструктуру для розпізнавання емоцій, є Azure Emotion API від Microsoft. Цей сервіс дозволяє визначати емоційний стан користувача на основі зображення обличчя та видає результати у вигляді ймовірнісного розподілу. На рисунку 1.6 показано приклад обробки зображення з виявленням емоції "Surprise" у двох осіб.

Рисунок 1.6 – Результат обробки зображення в Azure Emotion API

Replika – це чат-бот, який побудований на нейронних мережах і має на меті емоційно підтримати користувача. Хоча, функція аналізу міміки не реалізована, система активно оцінює текстову комунікацію, визначає настрій і відповідно підбирає відповіді. ей приклад демонструє, що емоційна взаємодія може реалізовуватись навіть без відеоаналізу, а виключно на ґрунтуючись лише на аналізі поведінки або мови [11]. Окрім відеоаналізу, емоційну взаємодію з користувачем можна реалізувати й іншими способами, зокрема через аналіз текстової комунікації. Прикладом такого підходу є застосунок Replika — віртуальний співрозмовник, що використовує нейронні мережі для створення підтримувальної емоційної взаємодії. На рисунку 1.7 продемонстровано його інтерфейс, орієнтований на збереження позитивного настрою користувача.

Рисунок 1.7 – Емоційно-орієнтований інтерфейс чат-бота Replika

Існують також і інші мобільні додатки, такі як Emotient, EmoReact, Moodie, які включають в себе базові механізми емоційного розпізнавання. Проте більшість з них є або експериментальними, або обмеженими у функціональності, або вже припинили підтримку.

1.4 Проблеми та виклики при реалізації систем розпізнавання емоцій

Незважаючи на помітне зростання зацікавленості у використанні емоційно-орієнтованих інтерфейсів у сучасному програмному забезпеченні, реалізація подібних систем на практиці стикається з рядом об’єктивних труднощів. Вони пов’язані не лише з технічними параметрами, а й із психологічними особливостями сприйняття емоцій, етичними питаннями, а в окремих випадках – і з юридичними обмеженнями щодо збору та обробки біометричних даних.

Особливо складним є завдання створення системи, яка не просто визначає емоцію, а й реагує на неї коректно, без затримок і помилкових спрацювань. Розглянемо основні проблеми, з якими стикаються розробники таких систем.

1.4.1 Точність та індивідуальні особливості

Кожен відчуває та показує емоції по різному. Скажімо, у когось «усмішка» виражається лише легким підняттям куточків губ, а в іншого – широкою мімікою з активною участю очей. У деяких людей природні вирази обличчя можуть здаватись стриманими або, навпаки, дуже емоційними, що може заплутати алгоритми.

Ще однією проблемою є культурні відмінності. Емоції, які в одній культурі можуть сприйматися як позитивні, в іншій можуть виглядати нещирими або й образливими. Це ускладнює створення універсальних моделей для розпізнавання емоцій.

До того ж, більшість загально доступних наборів даних для навчання моделей (datasets) мають обмежену репрезентативність часто там переважають обличчя людей однієї раси, статі чи віку. У результаті система, яка показує високу точність у тестових умовах, може погано працювати на практиці при широкому розмаїтті користувачів. Розпізнавання емоцій ускладнюється не лише якістю зображення, але й природною варіативністю міміки в різних людей. Один і той самий емоційний стан може проявлятися по-різному залежно від індивідуальних особливостей обличчя. На рисунку 1.8 продемонстровано приклади типових емоцій, які суттєво відрізняються за вираженням, навіть при однаковому емоційному контексті.

Рисунок 1.8 - Приклад варіацій міміки у різних людей при однаковій емоції

1.4.2 Реалізація в реальному часі

У застосунках з елементами емоційного зворотного зв’язку важливо забезпечити швидку реакцію системи. Якщо людина показала якусь емоцію, програма має майже одразу її розпізнати й щось зробити у відповідь – наприклад, показати повідомлення, змінити картинку чи озвучити результат. Якщо ж реакція затримується навіть на кілька секунд, це вже псує загальне враження.

Щоб усе працювало плавно, потрібно добре налаштувати як сам алгоритм розпізнавання, так і всю програму. Особливо важливо, щоб обробка відео не гальмувала інтерфейс. Якщо комп’ютер слабкий, усе може працювати повільно, і користувачеві буде незручно.

Ще одна річ, яку часто забувають – це стабільність відео з камери. Якщо камера “скаче” по кадрах або підвисає, то система може інтерпретувати емоції по-різному навіть на схожих зображеннях, що виглядає як помилка.

Ще одним важливим аспектом для систем розпізнавання емоцій у реальному часі є стабільність відеопотоку та швидкість обробки. Навіть незначні затримки або зниження частоти кадрів можуть спричинити помилкове розпізнавання або пропуск обличчя. На рисунку 1.9 показано приклад ситуації, коли система виявляє два обличчя, але працює із затримкою понад 3 секунди на кадр.

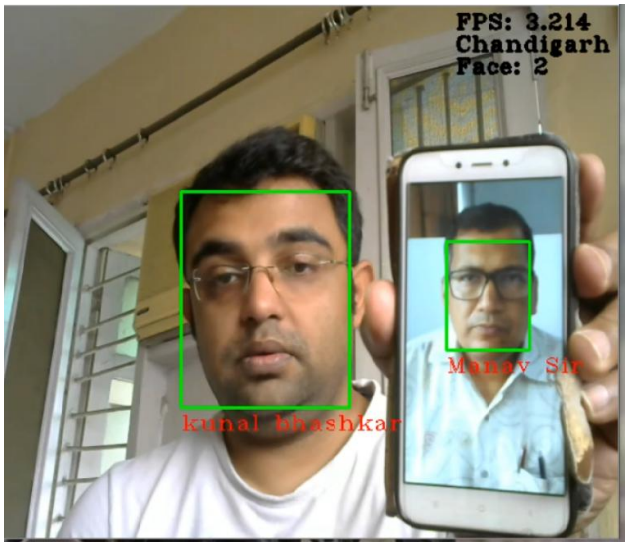

Рисунок 1.9 – Затримка при обробці зображення в реальному часі

1.4.3 Технічні обмеження пристроїв

Не всі користувачі мають сучасну або потужну техніку. Наприклад, старенький ноутбук чи дешева вебкамера часто дають зображення такої якості, що алгоритми просто “губляться” і не можуть нічого нормально розпізнати.
Типові проблеми, які зустрічаються найчастіше:

- Занадто маленька роздільна здатність – обличчя виходить розмитим.
- Погане світло: або темно, або яскраве світло засвічує половину кадру.
- Камера “розфокусується” під час рухів, особливо якщо в неї немає автофокусу.
- Волосся, борода, окуляри – все це може частково закривати обличчя.

Через це навіть найточніша модель іноді видає повну нісенітницю, бо вона працює з неякісним зображенням. Тому важливо, щоб розробник передбачив якісь прості фільтри – наприклад, покращення яскравості або контрасту. А якщо камера зовсім погано знімає – варто хоча б вивести попередження для користувача. Якість вхідного зображення має вирішальне значення для точності роботи системи розпізнавання емоцій. Один із ключових факторів — це освітлення: при слабкому світлі система може неправильно інтерпретувати емоції або не розпізнати обличчя взагалі. На рисунку 1.10 наведено приклад того, як якість освітлення впливає на сприйняття обличчя системою.

Рисунок 1.10 – Вплив якості камери та освітлення на розпізнавання емоцій

1.4.4 Інтерфейс та сприйняття користувачем

Навіть якщо система працює точно, це не означає, що вона буде зручною для користувача. Особливо у випадку програмного забезпечення з інтерактивною взаємодією – інтерфейс має бути інтуїтивно зрозумілим, зрозумілим і викликати приємні емоції, а не роздратування.

Користувач повинен чітко розуміти, що саме відбувається: що аналізується, навіщо і який результат. Добре, якщо система показує це у зрозумілій формі – наприклад, у вигляді смайлика, голосового повідомлення чи навіть просто тексту.

Також важливо дати користувачу можливість контролювати процес: поставити на паузу, подивитись історію результатів, або взагалі вимкнути камеру. Інакше може скластися враження, що за ним "підглядають", особливо якщо програма не пояснює, що вся обробка відбувається тільки на комп’ютері й нікуди не передається.

Саме тому розробнику варто подумати не тільки про технічну частину, а й про те, як усе виглядатиме “з боку” користувача. Інтерфейс має бути простим і викликати довіру – інакше навіть перспективна ідея не буде позитивно сприйнята користувачем. Окрім точності алгоритмів, важливу роль відіграє зручність взаємодії з користувачем. Інтерфейс системи розпізнавання емоцій має бути не лише інформативним, а й інтуїтивно зрозумілим. На рисунку 1.11 представлено приклад інтерфейсу професійної системи емоційного аналізу, де одночасно відображаються мімічні показники, інтенсивність емоцій і графік змін у динаміці.

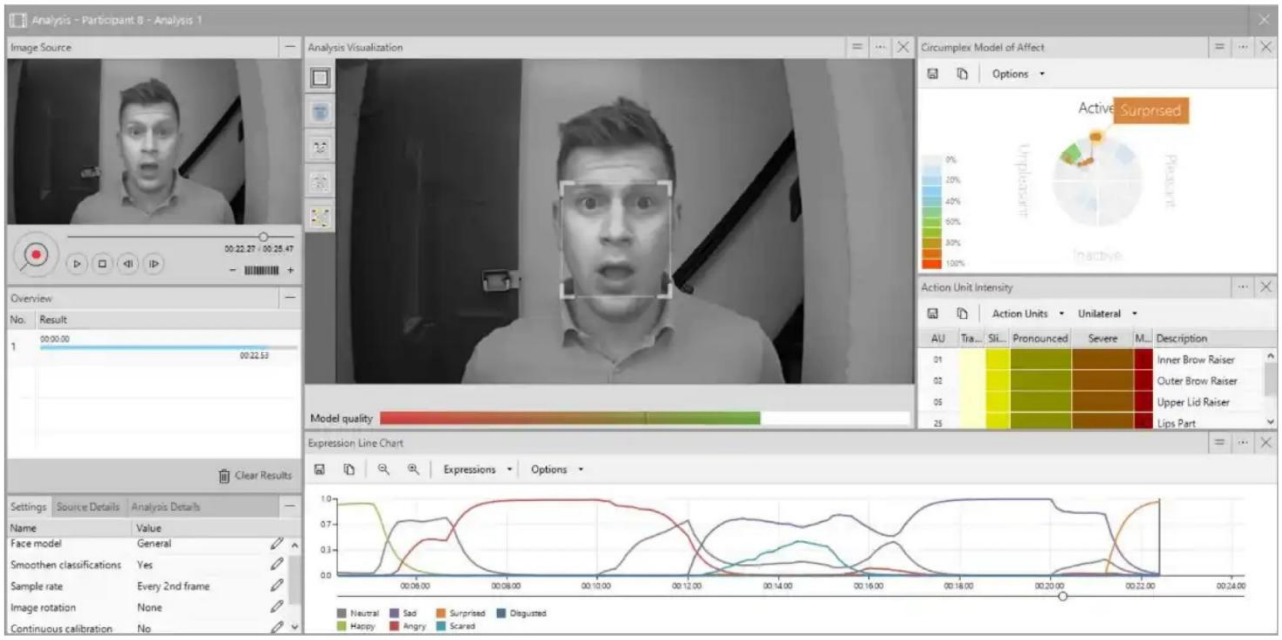

Рисунок 1.11 – Приклад інтерфейсу системи розпізнавання емоцій

1.5 Висновки

У даному розділі було детально проаналізовано основні аспекти, що стосуються створення інформаційної розважальної системи з розпізнаванням емоцій. Було з’ясовано, що проблема емоційної взаємодії в цифрових системах є актуальною, особливо у сфері розваг, де важливо забезпечити глибше залучення користувача.

Проведений аналіз показав, що на світовому ринку вже існують потужні платформи, які дозволяють виконувати розпізнавання емоцій (Affectiva, FaceReader, Azure Emotion API), але більшість із них мають суттєві обмеження: потребують підключення до Інтернету, є комерційними, складними в налаштуванні або не дають можливості повністю контролювати процес обробки даних.

Крім того, було встановлено, що серед безкоштовних або відкритих рішень не вистачає простих у використанні десктопних програм, які могли б поєднувати розпізнавання емоцій, .візуальний зворотний зв’язок та інтерфейс демонстрації результатів взаємодії. Це підтверджує актуальність розробки власної програми,

яка б працювала локально, зберігала результати та могла б бути використана як демонстраційна чи навчальна.

Загалом, зроблені у цьому розділі висновки стали основою для формування технічних вимог до системи та обґрунтування вибору інструментів, що буде описано у наступному розділі.

2 ІТ-ІНФРАСТРУКТУРА ТА ВИБІР ТЕХНОЛОГІЙ

2.1 Вибір ІТ-інфраструктури та забезпечення роботи системи

ІТ-інфраструктура створеної інформаційної розважальної системи розпізнавання емоцій є простою та локальною, оскільки програма розроблена як десктопний застосунок, який працює без підключення до Інтернету. Незважаючи на відсутність серверної частини, система все одно має певну архітектуру, набір компонентів, а також вимоги до середовища розгортання [12].

Архітектура системи

Архітектура проєкту побудована за моделлю "клієнтська частина з локальною обробкою". Користувач запускає програму на своєму комп’ютері, який виконує всі функції:

- захоплює зображення з вебкамери;
- обробляє дані в режимі реального часу;
- розпізнає емоцію за допомогою DeepFace;
- зберігає результати в локальні файли JSON;
- візуалізує прогрес через графіки;
- озвучує результат голосовим повідомленням.

Усі процеси виконуються на одному пристрої, тому можна сказати, що програма реалізована у вигляді локального автономного рішення.

Компоненти та середовище розгортання

- Операційна система: Windows 10 або вище
- Мова програмування: Python 3.10
- Середовище розробки: PyCharm
- Віртуальне середовище: venv з бібліотеками (PyQt5, OpenCV, DeepFace, gTTS, matplotlib)
- Формати збереження даних: JSON-файли
- Пристрої: Вебкамера, мікрофон (для озвучення)

Програма зібрана у .exe за допомогою pyinstaller і не потребує встановлення Python на цільовій машині. Це дозволяє запускати застосунок на будь-якому ПК з Windows без додаткового налаштування.

Безпека та стійкість

Основною перевагою з точки зору безпеки є локальна обробка усіх даних. Програма не надсилає зображення або результати в мережу, не зберігає персональні фото в Інтернеті. Усі файли (включаючи збережені емоції, статистику) зберігаються на комп'ютері користувача у папці програми.

Використання відкритих бібліотек із широкою підтримкою забезпечує стійкість системи. Навіть якщо окремі функції (наприклад, озвучення через gTTS) не працюють через відсутність Інтернету, програма продовжує виконувати основні завдання – розпізнавання емоцій, збереження результатів, побудову графіків.

Моніторинг та логування

У межах проєкту використовується базове логування через консоль. Під час розробки повідомлення про хід роботи, помилки або виключення виводяться через print(). Це дозволяє оперативно виявляти проблеми, наприклад, при помилці розпізнавання облич або при недоступності модуля озвучення.

У майбутньому, при масштабуванні, можна реалізувати додатковий лог-файл для збереження подій, або інтегрувати простий інтерфейс для перегляду помилок.

Розроблена система взаємодіє з користувачем у реальному часі через вебкамеру, здійснюючи обробку зображення, розпізнавання емоцій, озвучування результату та збереження статистики. На рисунку 2.1 зображено загальну схему IT-інфраструктури, яка демонструє послідовність компонентів системи та взаємозв'язок між ними.

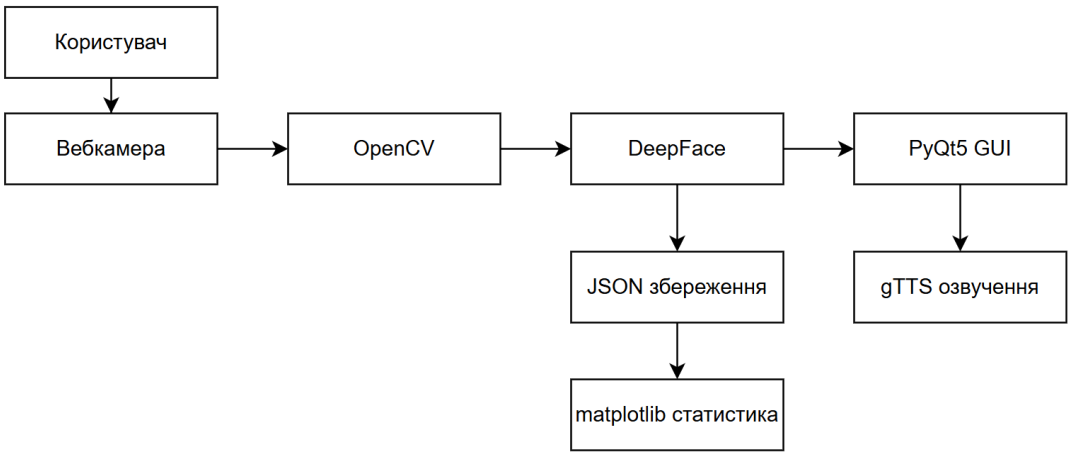

Рисунок 2.1 – Схема IT-інфраструктури системи розпізнавання емоцій

2.2 Формування системних вимог

Перед початком розробки будь-якого програмного забезпечення необхідно чітко визначити його функціональні можливості, обмеження та умови, в яких програма має працювати. Особливо це важливо для систем, які працюють з обробкою зображення в реальному часі, як у випадку з інформаційною розважальною системою розпізнавання емоцій.

Основна мета створюваної системи – забезпечити зручну взаємодію користувача з комп’ютером через емоційний зворотній зв’язок. Для цього програмі потрібно щоб вона фіксувала дані виразу обличчя з веб-камери, для того щоб в подальшому розпізнавати емоцію, показувати результат у зручній формі, зберігати статистику та за можливості озвучувати відповідь.

2.2.1 Загальна характеристика програми

Програмне забезпечення реалізовано у вигляді настільного (десктопного) додатка з графічним інтерфейсом, який працює без підключення до Інтернету. Уся обробка даних виконується локально, що важливо для забезпечення конфіденційності користувача. Розробка буде здійснюватись мовою Python із використанням відкритих бібліотек.

2.2.2 Функціональні вимоги

До функціональних вимог належать ті дії, які програма повинна виконувати. На основі поставленої мети, сформульовано наступні основні функції системи:

- Отримання відеопотоку з вебкамери.

Програма повинна автоматично підключатись до камери користувача та виводити потік у реальному часі.

- Виявлення обличчя на відео.

Алгоритм повинен ідентифікувати область обличчя на зображенні, навіть при неідеальних умовах (освітлення, положення голови).

- Розпізнавання емоцій користувача.

Система має аналізувати міміку і визначати базову емоцію (радість, сум, злість, подив, страх тощо).

- Виведення результату.

Емоція повинна відображатись у зручному форматі – текстом, піктограмою, графіком або голосом.

- Озвучення емоцій.

При бажанні користувача – результат має озвучуватись через синтез мови (наприклад, за допомогою gTTS або pyttsx3).

- Збереження статистики.

Дані про розпізнані емоції зберігаються у файл або базу даних, що дозволяє переглядати історію.

- Візуалізація статистики.

Користувач повинен мати можливість подивитися графік розподілу емоцій за весь період роботи. На рисунку 2.2 показано послідовність обробки даних у системі: від отримання зображення до збереження та візуалізації результатів.

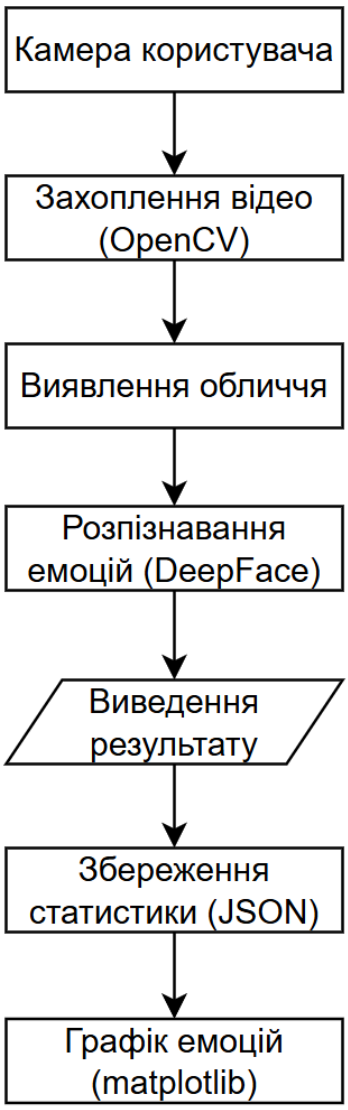

Рисунок 2.2 – Загальна послідовність обробки даних у системі розпізнавання емоцій

2.2.3 Нефункціональні вимоги

Окрім основних функцій, існують загальні вимоги до стабільності, безпеки та зручності роботи з програмою:

- Автономність.

Програма повинна працювати без підключення до Інтернету – вся логіка реалізується на локальному пристрої.

- Конфіденційність.

Зображення користувача не повинні зберігатись або передаватись у хмару. Уся обробка виконується на локальному рівні.

- Простота інтерфейсу.

Інтерфейс повинен бути інтуїтивно зрозумілим, без зайвих вікон або складних налаштувань.

- Мінімальні системні вимоги.

Програма повинна працювати на звичайному комп'ютері з середніми характеристиками:

1. процесор не нижче Intel Core i3;
2. 4 ГБ оперативної пам'яті;
3. вебкамера з мінімальною роздільною здатністю 640x480;
4. ОС: Windows 10 або вище.

2.2.4 Технічні вимоги до середовища розробки

Розробка програми передбачається на мові Python (версія 3.10 і вище) з використанням таких технологій:

- PyQt5 – для створення графічного інтерфейсу користувача;
- OpenCV – для роботи з відеопотоком і виділення облич;
- DeepFace – для розпізнавання емоцій;
- gTTS / pyttsx3 – для озвучення емоцій;
- matplotlib / pandas – для побудови статистичних графіків;
- локальний файл JSON – для збереження результатів розпізнавання [13-14].

2.3 Огляд інформаційних технологій для розробки десктопних застосунків

У межах цієї бакалаврської кваліфікаційної роботи реалізовано настільний застосунок, який розпізнає емоції користувача через вебкамеру. Важливою

частиною підготовки до реалізації є вибір відповідних інформаційних технологій. Враховуючи, що система повинна мати графічний інтерфейс, працювати без інтернету, обробляти відеопотік у реальному часі та бути стабільною, було проаналізовано кілька можливих варіантів засобів розробки.

Одним із ключових рішень стало формування концепції реалізації додатка саме у форматі десктопної програми, а не веб-додатку. Такий підхід має низку переваг. По-перше, немає необхідності у налаштуванні серверної частини або браузерної підтримки – усе працює безпосередньо на комп'ютері користувача. По-друге, це дає змогу працювати з локальними пристроями, такими як вебкамера, без додаткових дозволів чи посередників. Крім того, локальна обробка забезпечує вищу конфіденційність, що є важливим фактором у роботі з відеоданими обличчя.

Під час розгляду варіантів були коротко проаналізовані доступні бібліотеки для побудови графічного інтерфейсу користувача на Python. Найпростішим рішенням є Tkinter, який вбудовано у стандартну бібліотеку Python. Але через обмежений зовнішній вигляд і слабку підтримку сучасного дизайну він не підходить для реалізації складніших інтерфейсів. Інший варіант – Kivy – більше підходить для мобільних застосунків, а його зовнішній вигляд на ПК не є звичним для користувача. Electron, який базується на вебтехнологіях, теж розглядався, але був відкинутий через великі обсяги файлів, вимогу знань JavaScript і складність налаштування.

У результаті вибір було зупинено на бібліотеці PyQt5, яка поєднує зручність програмування на Python зі всіма можливостями повноцінного настільного графічного інтерфейсу. Ця технологія дозволяє створювати вікна, кнопки, відображення зображень, таблиці, а також легко інтегрується з іншими бібліотеками, такими як OpenCV, matplotlib і DeepFace. PyQt5 підтримує розбиття інтерфейсу на окремі компоненти, що робить розробку структурованою і зручною [15-18].

Крім цього, важливою перевагою обраної технології є її стабільність та підтримка в спільноті Python-розробників. Документація до PyQt5 містить приклади рішень багатьох типових задач, що значно полегшує процес створення

застосунку навіть у рамках навчального проєкту.

Таким чином, у ході аналізу інформаційних технологій для розробки десктопних програм було зроблено вибір на користь PyQt5 як основного інструменту для створення інтерфейсу. Такий вибір є обґрунтованим і відповідає вимогам, поставленим до системи: автономність, простота використання, можливість роботи з камерою та виведення результатів у зручній формі.

На рисунку 2.3 зображено головне вікно створеної системи, розробленої із використанням PyQt5. Інтерфейс містить вікно для відображення відеопотоку, елементи керування та зону виведення результатів.

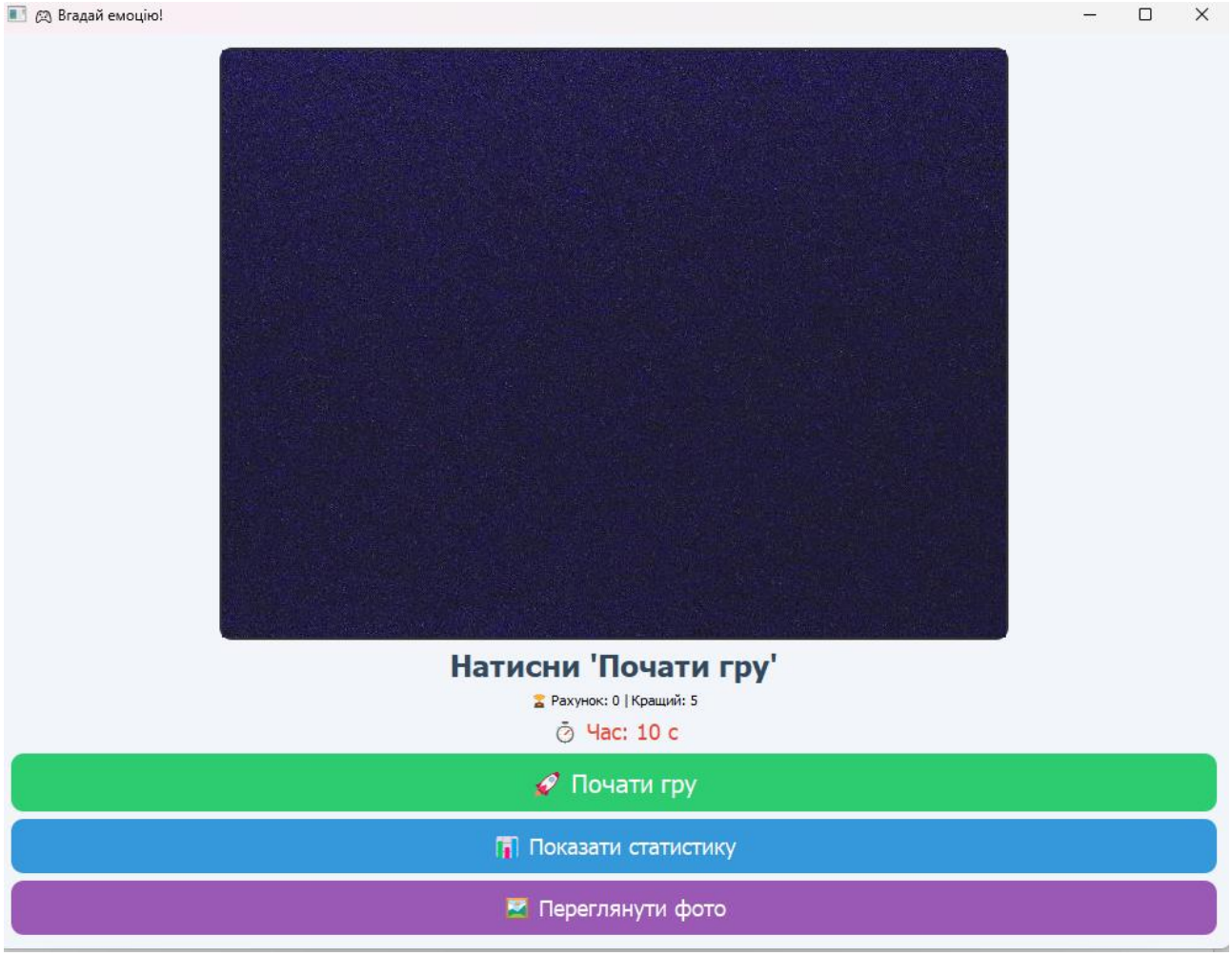

Рисунок 2.3 – Приклад інтерфейсу розробленого застосунку на основі PyQt5

2.4 Аналіз можливостей мови програмування Python

Python є однією з найпопулярніших мов програмування у світі, особливо в галузях, пов’язаних з обробкою зображень, штучним інтелектом, машинним навчанням, аналізом даних і створенням прототипів. Її популярність пояснюється не тільки простим синтаксисом, але й широкою підтримкою бібліотек, активною спільнотою та доступністю інструментів для різних задач.

Під час вибору мови для створення інформаційної розважальної системи з розпізнаванням емоцій важливо було врахувати декілька факторів: наявність бібліотек для роботи з відео, підтримку аналізу облич, можливість створення графічного інтерфейсу, а також простоту інтеграції всіх компонентів в один робочий застосунок. Саме Python повністю задовольняє ці умови.

Однією з головних переваг Python є його модульність. Наприклад, для роботи з відео й камерою використовується бібліотека OpenCV, яка дозволяє захоплювати потік у реальному часі та обробляти його з мінімальними затримками. Для розпізнавання емоцій обрано бібліотеку DeepFace – вона побудована поверх сучасних моделей комп’ютерного зору, зокрема VGG-Face, Google FaceNet, OpenFace, DeepID тощо, і підтримує визначення кількох базових емоцій без потреби у додатковому навчанні моделі.

Ще однією сильною стороною Python є можливість побудови графічного інтерфейсу через бібліотеку PyQt5. Це дозволяє створити повноцінний десктопний застосунок з кнопками, вікнами, відображенням відео з камери, а також взаємодією з користувачем [19].

Також Python має інструменти для роботи зі звуком. Зокрема в цій програмі реалізовано вимову розпізнаних емоцій. Це здійснюється за допомогою бібліотеки gTTS (Google Text-to-Speech), що перетворює текст у мову. Як альтернатива, можна використовувати pyttsx3, який працює без підключення до Інтернету.

Ще один плюс – зберігання та візуалізація результатів. Для реалізації цього використовуються стандартні інструменти Python: matplotlib – для побудови

графіків, pandas – для комфортної роботи з даними, JSON – для локального збереження результатів.

Варто зазначити, що Python має добру сумісність із середовищами розробки на кшталт Visual Studio Code, де є інтегровані інструменти відлагодження, автодоповнення коду, запуску проєкту та роботи з віртуальними середовищами. Це значно спрощує розробку і підтримку проєкту.

Незважаючи на те, що Python – інтерпретована мова, її продуктивності достатньо для проєктів, які не потребують обробки сотень відеопотоків одночасно. У межах розроблюваної системи, коли обробляється зображення з однієї вебкамери і виконується лише розпізнавання емоцій, його можливостей цілком достатньо [20].

Таким чином, Python є оптимальним вибором для реалізації даної системи. Він забезпечує доступ до всіх необхідних інструментів, дозволяє швидко створити стабільний прототип і в разі потреби – масштабувати проєкт або перенести його на інші платформи.

```
import sys
import os
import cv2
import json
import uuid
import random
from datetime import datetime
from PyQt5 import QtWidgets, QtGui, QtCore
from deepface import DeepFace
from gtts import gTTS
import pygame
import matplotlib.pyplot as plt

class EmotionGameBot(QtWidgets.QWidget):  3 usages  new *
    def __init__(self):  new *
        super().__init__()
```

Рисунок 2.4 – Фрагмент коду з використанням бібліотек Python для створення інтерфейсу, обробки відео та емоційного аналізу

```
    def update_frame(self):  1 usage  new *
        if not self.capture.isOpened():
            return
        ret, frame = self.capture.read()
        if not ret:
            return
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        faces = self.face_cascade.detectMultiScale(gray)
        for (x, y, w, h) in faces:
            roi = frame[y:y + h, x:x + w]
            now = QtCore.QTime.currentTime()
            if self.last_emotion_check.msecsTo(now) > 1500:
                try:
                    result = DeepFace.analyze(roi, actions=['emotion'], enforce_detection=False)
                    res = result[0] if isinstance(result, list) else result
                    emotion = res['dominant_emotion']
                    emotion_score = res['emotion'][emotion]
                    self.detected_emotion = emotion
```

Рисунок 2.5 – Код використання DeepFace для аналізу емоцій користувача

2.5 Аналіз бібліотек і фреймворків, використаних у проєкті

Під час розробки програмного забезпечення виникла потреба реалізувати широкий спектр функціональних можливостей, зокрема побудову графічного інтерфейсу, обробку відеопотоку з камери, розпізнавання емоцій користувача, озвучення результатів та збереження статистичних даних. Щоб усе це не писати “з нуля”, було прийнято рішення використовувати вже готові інструменти – бібліотеками, які існують давно й добре себе зарекомендували. Усі вони мають відкритий код, їх можна безкоштовно використовувати та модифікувати, що дуже зручно для навчального та некомерційного проєкту.

Для побудови вікон, кнопок, вкладок і взагалі всього, що бачить користувач на екрані, було використано PyQt5. Це досить відома бібліотека, яка дозволяє створювати інтерфейс майже так само гнучко, як у професійних програмах. Було реалізовано інтерфейс, у якому є все потрібне: відео з камери, результат розпізнавання, статистика, кнопки керування. Ще одним важливим аспектом є те, що PyQt5 органічно інтегрується з мовою Python, що дозволяє реалізувати повноцінний графічний інтерфейс без потреби в додатковому налаштуванні серверної частини чи вивченні веб-технологій, таких як HTML або JavaScript, як це зазвичай потрібно при створенні веб-застосунків [21].

Для захоплення відео з веб-камери використали використали OpenCV. Ця бібліотека відома всім, хто хоч раз працював з комп’ютерним зором. Вона дозволяє “зловити” потік з камери, брати з нього кадри, конвертувати їх у

зручний формат і шукати на зображенні обличчя. Було використано вбудовану функцію з каскадами Хаара, яка досить швидко й точно знаходить обличчя навіть у не дуже якісному відео. У процесі тестування було помічено, що на слабких комп'ютерах теж усе працює стабільно – OpenCV тут реально допомогла.

Розпізнавання емоцій реалізовано за допомогою DeepFace. Це сучасна бібліотека, яка вже має вбудовані моделі, натреновані на великих наборах даних. Підключення бібліотеки DeepFace дозволило уникнути потреби у самостійному навчанні нейромереж, оскільки вона вже містить попередньо натреновані моделі. До бібліотеки передається зображення обличчя, після чого повертається результат із визначенням домінантної емоції. DeepFace підтримує розпізнавання таких емоцій, як радість, злість, смуток, здивування, відраза тощо. За нашими спостереженнями, найкраще вона працює при нормальному освітленні та камері середньої якості [22].

Щоб зробити програму трохи "живішою", Було реалізовано озвучення результатів, що забезпечує емоційний зворотний зв'язок із користувачем.

Для реалізації озвучення результатів було використано бібліотеку gTTS (Google Text-to-Speech), яка дає змогу швидко створювати аудіофайли з будь-якого текстового вмісту, зокрема українською мовою. Таким чином, коли програма розпізнає емоцію, вона може сказати, наприклад: "У вас радість!" – це додає елемент ігрової взаємодії. Проте gTTS функціонує лише за умови наявності підключення до Інтернету, тому як альтернативу було обрано pyttsx3 – бібліотеку, яка забезпечує озвучення тексту локально, без необхідності доступу до мережі. Це є важливим чинником для збереження конфіденційності, оскільки жодна інформація не передається на сторонні сервери.

Збереження результатів було реалізовано у простий і надійний спосіб шляхом використання формату JSON. Для кожного користувача автоматично створюється окремий файл, у якому зберігаються відповідні дані: дата, виявлена емоція, кількість набраних балів тощо.Якщо користувач запускає програму знову – вся історія залишається. Це зручно й не потребує складної бази даних. Крім того, найкращий результат зберігається окремо в текстовому файлі, щоб

його можна було швидко показати при наступному запуску – наприклад, для відображення “рекорду”.

Для цього використали бібліотеку matplotlib, яка дає можливість будувати різні графіки. У даному випадку дійшли до рішення обмежитися лінійною діаграмою, яка показує, як змінювались результати користувача з часом. Це просто, але наочно. Графік будується автоматично на основі даних з JSON-файлу. Для обробки цих даних також використовується pandas – дуже зручна бібліотека, яка дозволяє швидко “підчистити” структуру, відсортувати дати, підрахувати середні значення. Хоча бібліотека була використана лише в базовому обсязі, у майбутньому можливе легке розширення статистичного функціоналу.

Загалом, обраний набір бібліотек забезпечив можливість об’єднати всі складові системи в єдину функціональну структуру. Завдяки цьому не виникло потреби у створенні власних рішень з нуля, що дозволило зосередити увагу на логіці роботи програмного забезпечення та реалізації взаємодії з користувачем.

Завдяки відкритим технологіям, вдалося реалізувати повноцінну локальну програму, яка не вимагає серверів, встановлюється без складнощів і працює автономно. Це особливо важливо, якщо користувач хоче мати повний контроль над своїми даними.

2.6 Аналіз середовища розробки

Під час роботи над інформаційною системою, яка аналізує емоції користувача використовували середовище розробки PyCharm. Цей інструмент було рекомендовано ще на початкових етапах розробки, і після тестування стало очевидно, що для проєктів на Python він є одним із найзручніших рішень. Усе, що потрібно для написання коду, налаштування бібліотек, тестування і навіть базового контролю версій – вже є «в комплекті». Це суттєво зекономило час, особливо коли кількість файлів і логіки почала збільшуватись.

Ми активно використовували підсвічування синтаксису, автодоповнення, зручні підказки та вбудований термінал, через який можна було швидко

встановити нові пакети. Наприклад, якщо десь була помилка у виклику функції або невірна назва змінної – PyCharm це одразу показував. Це, здається, дрібниця, але коли проєкт складається з десятків файлів – такі речі критично важливі. Особливо корисною виявилася функція швидкого переходу до визначень функцій і класів, що дозволило значно пришвидшити орієнтацію в структурі коду.

Ще на самому початку було прийнято рішення не ставити всі бібліотеки «всередину системи», а створити віртуальне середовище (venv). Це окремий простір, де зберігаються всі залежності саме для цього проєкту. Такий підхід дозволив уникнути конфліктів, які могли виникнути при використанні кількох проєктів на одному комп'ютері. Крім того, це зробило розгортання на інших машинах набагато простішим: достатньо активувати середовище і все працює, як треба.

Уся розробка здійснювалась мовою Python версії 3.10, яка забезпечувала підтримку всіх бібліотек, необхідних для реалізації функціоналу проєкту.

Серед них були PyQt5 (інтерфейс), OpenCV (робота з відео), DeepFace (розпізнавання емоцій), gTTS і pyttsx3 (озвучення), matplotlib і pandas (графіки й обробка статистики). Кожна з них встановлювалась через pip, і, чесно кажучи, кілька разів виникали ситуації, коли потрібно було видалити пакет і поставити іншу версію – це теж легко вирішувалося в межах venv.

Щоб не плутатись у файлах і не «ламати» логіку, була побудувана чітка структура проєкту в PyCharm. На рисунку 2.5 можна побачити, як усе виглядало: окремі файли для логіки, інтерфейсу, озвучення, збереження результатів, а також віртуальне середовище та конфігураційні файли. Така структура виникла не одразу – спочатку було все прописано просто"в одному файлі", але з часом розбили код на частини, і це суттєво полегшило підтримку. Такий підхід рекомендується в багатьох методичних матеріалах, тому під час розробки було дотримано базових правил оформлення структури проєкту.

Загалом, робота в PyCharm виявилась комфортною. Це середовище не нав'язує складних налаштувань, але при цьому дає багато інструментів, які

справді допомагають. Навіть коли щось ішло не так, помилки можна було швидко знайти, перевірити лог, зробити відкат змін – і продовжити працювати.

Завдяки вказаним можливостям вдалося уникнути затримок у процесі розробки та оперативно перейти від створення прототипу до реалізації повнофункціонального програмного рішення.

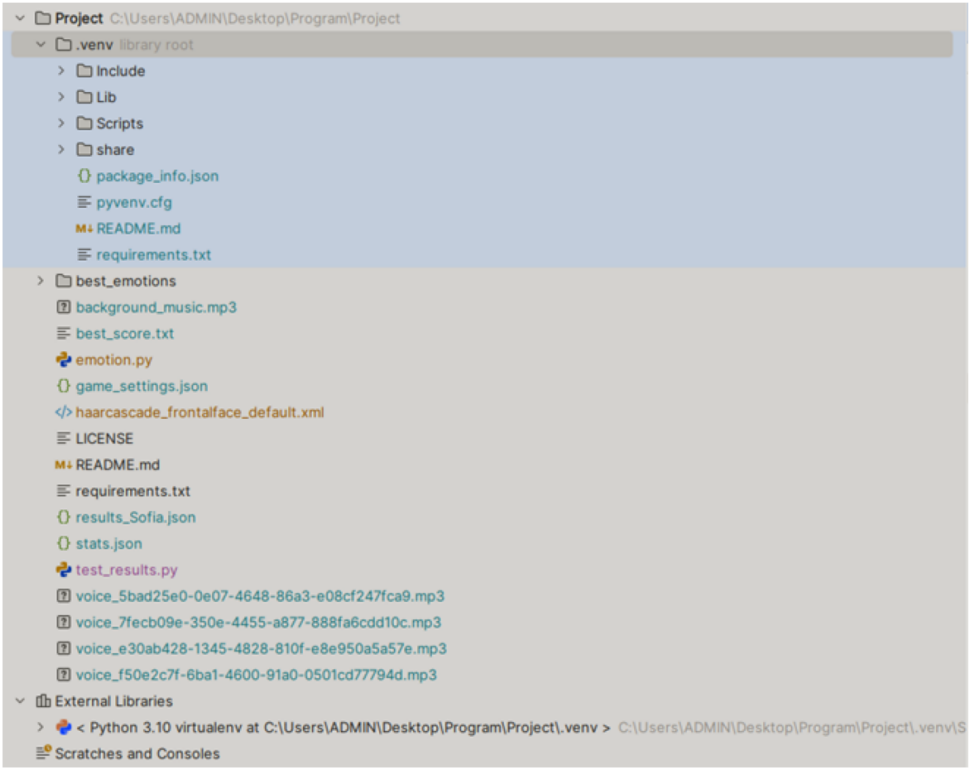

Рисунок 2.6 – Робоче середовище PyCharm із відкритим проєктом застосунку

Коли було розпочато розробку цієї програми, з самого початку було зрозуміло, що вона має працювати як звичайний застосунок на комп’ютері. Тобто, щоб людина могла просто відкрити файл і запустити, не заморочуючись із тим, як встановити Python чи інші залежності. Було прийнято рішення зробити .exe-файл, який можна буде поширювати як звичайний додаток. Для цього скористалися програмою pyinstaller – вона дозволяє зібрати все в один виконуваний файл, включно з усіма картинками, моделями, налаштуваннями.

Щоб усе працювало правильно, довелось трохи попрацювати з тим, як правильно вказати шляхи до ресурсів. Був момент, коли через неправильний шлях одна з моделей просто не завантажувалась і програма “падала”. Після

кількох ітерацій налаштувань було досягнуто стабільної роботи застосунку. У результаті користувач може просто двічі клацнути по виконуваному файлу, і програма запуститься незалежно від наявності Python на пристрої.

Вся розробка велася у PyCharm. Цим середовищем я користуюсь не вперше, і воно справді зручне, особливо коли треба відстежити помилки. Часто запускали не весь код, а тільки певні частини – наприклад, окремий модуль, щоб перевірити, чи працює розпізнавання або озвучення. PyCharm дає змогу зупинити виконання в конкретному місці, подивитись, що зберігається у змінних, і зрозуміти, де щось “ламається”.

Особливо це допомогло, коли працювали з відеопотоком. Іноді бували помилки, які важко було побачити одразу, бо все йшло “в реальному часі”. Саме тоді дебагер і перегляд стека викликів дозволяли зрозуміти, що саме викликає збій. Якщо чесно, без цих інструментів довелося б витратити набагато більше часу.

Ще варто згадати, що PyCharm побудований на платформі IntelliJ. Тобто сама програма складається з базової частини і додаткових плагінів. Наприклад, підтримка Python – це не вбудована функція, а модуль, який можна вимкнути чи замінити. Це корисно, бо дозволяє налаштувати середовище під конкретний проєкт. У межах розроблюваної системи залишили тільки те, що справді використовувалося. Завдяки цьому PyCharm не гальмував, працював стабільно, і нічого зайвого не заважало.

Загалом, комбінація PyCharm + pyinstaller виявилась зручною. Було кілька моментів, коли все “ламалось” на рівному місці, але з часом усе вдалося налаштувати. Тепер користувачам не потрібно думати про встановлення, налаштування чи командний рядок – достатньо просто запустити файл, і програма працює.

У програмному забезпеченні реалізовано функції збереження результатів розпізнавання емоцій користувача, побудови графіків із динамікою змін емоційного стану, а також озвучення виявлених емоцій. Уся обробка інформації здійснюється виключно локально на пристрої користувача, без використання мережевих з’єднань або сторонніх серверів, що забезпечує високий рівень

конфіденційності. Важливо наголосити, що створене програмне забезпечення не є грою. Це повноцінна інформаційна система, орієнтована на демонстрацію можливостей емоційного зворотного зв’язку між людиною та комп’ютером у межах інтерфейсу взаємодії. Розважальні елементи в інтерфейсі мають допоміжний характер і використовуються виключно для підвищення інтересу користувача до досліджуваного функціоналу.

2.7 Технології візуалізації та роботи з даними

Одним із важливих елементів розважальної системи розпізнавання емоцій є можливість відображення статистики проходження гри та збереження результатів для подальшого аналізу. З цією метою в проєкті були використані прості, але потужні засоби для роботи з даними у Python, зокрема бібліотеки matplotlib, pandas та формат JSON.

Для побудови графіків у застосунку використовується бібліотека matplotlib – одна з найвідоміших у Python для візуалізації даних. У нашій програмі з її допомогою будуються лінійні графіки, які показують, як змінювався результат користувача в різні дні. Це дозволяє зробити систему більш наочною та цікавою для користувача, оскільки він бачить свій прогрес у вигляді наочних діаграм.

Візуалізація здійснюється у вікні після натискання кнопки “Показати статистику”. Дані для графіка беруться з JSON-файлу, де зберігається історія кожного проходження гри з кількістю балів і датою. Matplotlib дозволяє змінювати стиль графіка, кольори, додавати підписи, що робить графік більш інформативним.

Для зберігання і обробки статистичних даних використовується формат JSON. У кожного гравця створюється окремий файл з іменем на кшталт results_Ім’я.json, де в масиві зберігаються об’єкти з полями "дата", "кількість балів" та "максимальний результат". Формат JSON обрано через його простоту, зручність і універсальність – його можна легко обробити, зчитати, змінити або експортувати.

Крім того, при потребі роботи з більшими масивами даних або складнішими обчисленнями використовується бібліотека pandas. Вона дозволяє без перешкод створити таблицю (DataFrame), та брати дані з JSON, та фільтрувати, групувати їх. У нашому випадку pandas може використовуватись для підрахунку середнього результату або знаходження найкращого дня проходження гри.

Цей підхід повністю відповідає формату локального застосунку, оскільки вся обробка та збереження здійснюється на комп'ютері користувача, без використання баз даних чи зовнішніх серверів. Це забезпечує не лише автономність, а й конфіденційність – дані не передаються третім сторонам.

Таким чином, комбінація matplotlib, pandas і JSON дозволила реалізувати повний цикл: збирання статистики, її зберігання, обробку та наочне представлення у вигляді графіка. Це зробило систему більш зручною, інтерактивною та інформативною для користувача.

На рисунку 2.7 показано приклад побудови графіка з результатами користувача за допомогою бібліотеки matplotlib.

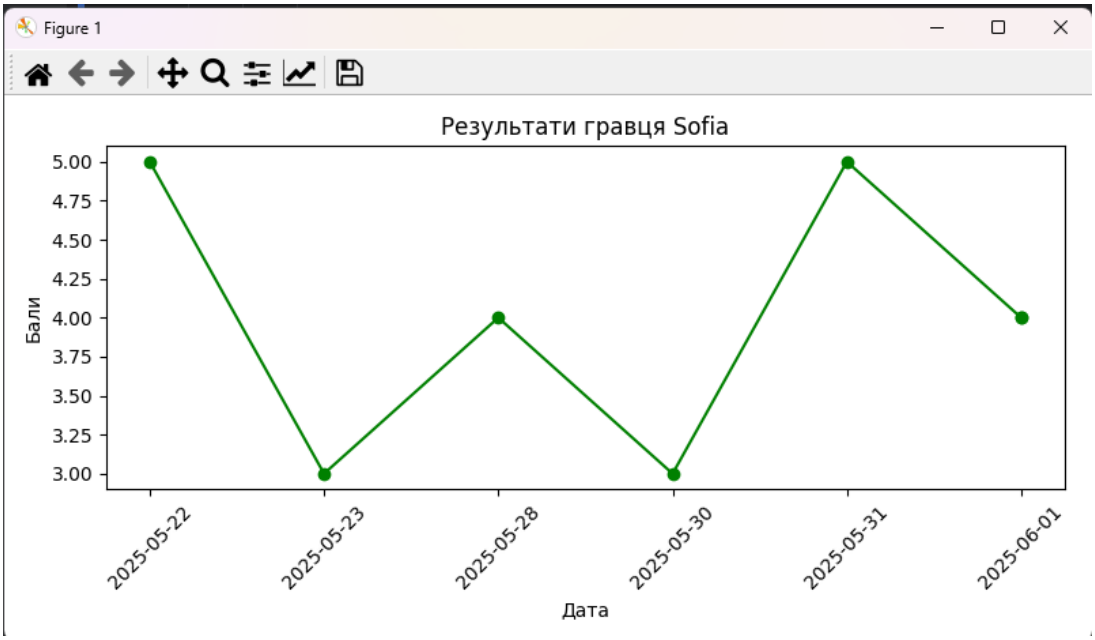

Рисунок 2.7 – Графік результатів користувача, побудований з використанням matplotlib

2.8 Висновки

У цьому розділі узагальнено основні технічні рішення, що були застосовані під час створення програмного забезпечення. Було розглянуто низку альтернатив, проте остаточний вибір зупинився на тих інструментах і технологіях, які найкраще відповідали поставленим завданням. Пріоритет надавався простоті використання, стабільності роботи та можливості локального функціонування системи без необхідності підключення до мережі чи додаткового встановлення компонентів.

У процесі проєктування було обрано мову програмування Python як основну, оскільки вона забезпечує швидку інтеграцію необхідних бібліотек і спрощує процес розробки та тестування системи. Серед переваг Python також варто відзначити підтримку великої кількості бібліотек, що дозволило забезпечити функціональність системи без створення власних низькорівневих модулів.

Серед використаних бібліотек — PyQt5 (інтерфейс користувача), OpenCV (обробка відеопотоку), DeepFace (розпізнавання емоцій), gTTS (озвучення), matplotlib і pandas (виведення статистики), а також формат JSON (локальне збереження результатів). Таке поєднання дозволило забезпечити автономну роботу системи без використання баз даних чи зовнішніх серверів.

Для розробки застосунку було використано середовище PyCharm, яке надало можливість створення ізольованого віртуального середовища та зручного керування залежностями. Завершальна збірка виконуваного файлу здійснювалася за допомогою pyinstaller, що дозволило створити інсталяційний .exe-файл для запуску програми на інших пристроях без додаткових налаштувань.

Особливу увагу було приділено інтеграції всіх компонентів системи. Незважаючи на відсутність підключення до мережі, камера, модуль розпізнавання, інтерфейс користувача та механізми збереження статистики функціонують як єдиний цілісний програмний комплекс. Такий підхід сприяє

підвищенню надійності системи та зменшенню ймовірності збоїв під час експлуатації.

Отже, обрані інструменти та технології дозволили реалізувати повністю автономний застосунок, який поєднує елементи штучного інтелекту, мультимедіа та зручний інтерфейс користувача, не потребуючи складного налаштування або серверної інфраструктури.

3 ПРОЕКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ РОЗВАЖАЛЬНОЇ СИСТЕМИ РОЗПІЗНАВАННЯ ЕМОЦІЙ

У цьому розділі розглядається інфраструктурна складова інформаційної розважальної системи з розпізнаванням емоцій. Детально описано компоненти, середовище розгортання, архітектуру, а також засоби забезпечення стійкої та безпечної роботи програмного забезпечення.

3.1 UML-діаграми

Перш ніж приступати до реалізації програмного забезпечення, важливо заздалегідь продумати його логіку, структуру та взаємодію окремих частин. Найзручніший спосіб зробити це – створити наочне представлення майбутньої системи. Для таких завдань використовують UML (*Unified Modeling Language*) – уніфіковану мову моделювання, яка дозволяє описати компоненти, зв’язки між ними та поведінку системи.

UML – це набір спеціальних графічних позначень, за допомогою яких розробник може візуалізувати ключові частини програми: структуру об’єктів, сценарії використання, порядок виконання дій тощо. Це значно спрощує розуміння логіки роботи ще до початку кодування й дає змогу вчасно помітити потенційні помилки або недоліки в архітектурі. Також UML є корисним під час супроводу та подальшого доопрацювання програмного продукту.

Діаграми UML застосовуються не лише на етапі проєктування, а й під час підготовки документації. У випадку бакалаврської кваліфікаційної роботи це має особливе значення, адже важливо не лише реалізувати програму, а й чітко показати, як вона функціонує на рівні логіки та взаємодії елементів.

У цьому підрозділі наведено основні діаграми, які стосуються інформаційної розважальної системи розпізнавання емоцій. Вони допомагають зрозуміти, як побудована система, як вона працює і яким чином користувач взаємодіє з нею [23-25].

У межах проєкту використано такі типи UML-діаграм:

– Діаграма варіантів використання (Use Case) – демонструє, які дії може виконати користувач і як система реагує на них;

– Діаграма класів – описує основні об’єкти програми, їхні властивості та функції;

– Діаграма активності – показує послідовність виконання дій або етапи обробки інформації;

– Діаграма послідовностей (Sequence Diagram) – ілюструє обмін повідомленнями між компонентами під час виконання сценаріїв;

– За потреби – діаграма компонентів або пакетів, яка дозволяє побачити внутрішню архітектуру застосунку.

Кожна з наведених діаграм супроводжується коротким поясненням: що вона показує, для чого була створена і як саме пов’язана з реальною роботою програми.

На рисунку 3.1 представлено діаграму випадків використання, яка демонструє основні дії користувача та взаємодію з елементами системи.

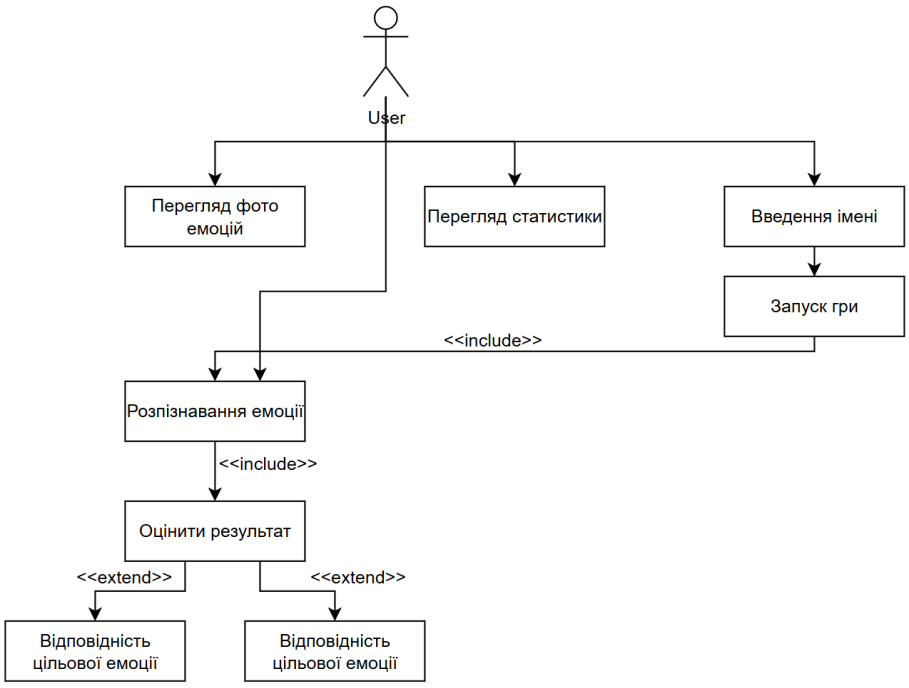

Рисунок 3.1 – Діаграма випадків використання

На діаграмі випадків використання, наведеній на рисунку 3.1, представлено основні дії, які може виконувати користувач у межах розробленої

розважально інформаційної системи розпізнавання емоцій. Центральним актором виступає User – звичайний користувач (гравець), який взаємодіє із системою через графічний інтерфейс, реалізований у PyQt5.

Користувач має можливість:

- ввести своє ім’я, що ініціалізує персоналізовану гру;
- запустити гру, що активує процес розпізнавання емоцій;
- переглянути статистику, де система виводить історію попередніх результатів у вигляді графіка;
- переглянути збережені фото, які система автоматично зберігає після вдалого розпізнавання емоції.

Основна дія – “Розпізнавання емоції” – включає в себе внутрішню піддію “Оцінити результат”, яка, у свою чергу, може мати розширення (extend) у вигляді перевірки на відповідність цільовій емоції. Це відображає логіку гри: система задає емоцію, користувач демонструє її, і якщо розпізнана емоція відповідає цільовій – зараховується бал.

Використання зв’язків <<include>> та <<extend>> дозволяє структурувати сценарії взаємодії та показати, що деякі дії (наприклад, оцінка результату) завжди відбуваються в межах основного сценарію, а інші – є необов’язковими або залежать від певних умов.

Ця діаграма є важливою частиною опису функціональності, оскільки дозволяє ще до реалізації зрозуміти, які саме дії виконує користувач, і як система реагує на ці дії. Вона також може бути корисною під час майбутнього розширення програми або додавання нових функцій.

На рисунку 3.2 наведено діаграму послідовності «Розпізнавання емоцій».

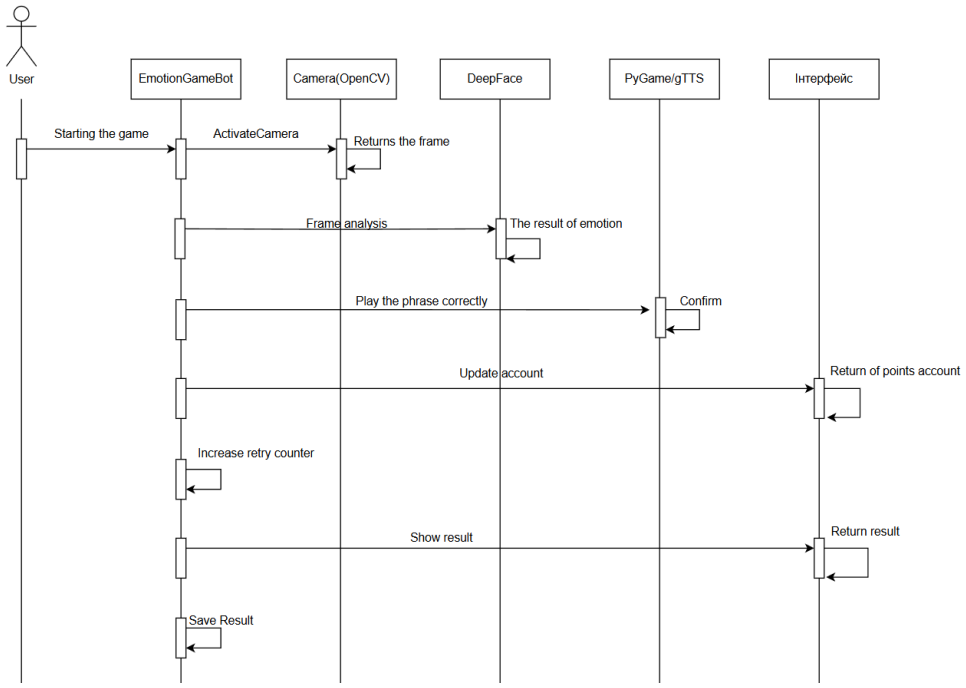

Рисунок 3.2 – Діаграма послідовності «Розпізнавання емоцій»

На діаграмі послідовності, наведеній на рисунку 3.2, відображено основний сценарій взаємодії користувача з розробленою інформаційною розважальною системою. Ця діаграма демонструє обмін повідомленнями між різними компонентами програми – від початку гри до завершення аналізу та збереження результатів.

Все починається з дії користувача – запуску гри. Основний клас EmotionGameBot ініціює активацію камери за допомогою бібліотеки OpenCV. Камера повертає відеокадр, який далі передається на аналіз у DeepFace. Цей компонент виконує розпізнавання емоції на основі отриманого зображення.

Результат аналізу використовується для визначення правильності виконання завдання (чи збігається вираз обличчя з очікуваною емоцією). У разі успішного розпізнавання система через модуль PyGame/gTTS озвучує фразу-підтвердження. Після цього оновлюється рахунок гравця та виводиться результат у графічному інтерфейсі.

У випадку помилки або завершення спроб, програма зберігає результат у файл (формат JSON) для подальшого перегляду в статистиці. Така послідовність операцій забезпечує логічну та природну взаємодію між модульними компонентами, дозволяючи легко масштабувати або вдосконалити систему.

Діаграма є корисною не лише для розуміння поточного функціоналу, а й для подальшого тестування та розширення, оскільки вона чітко відображає всі кроки обміну повідомленнями між частинами системи.

На рисунку 3.3 наведено діаграму кооперації системи розпізнавання емоцій.

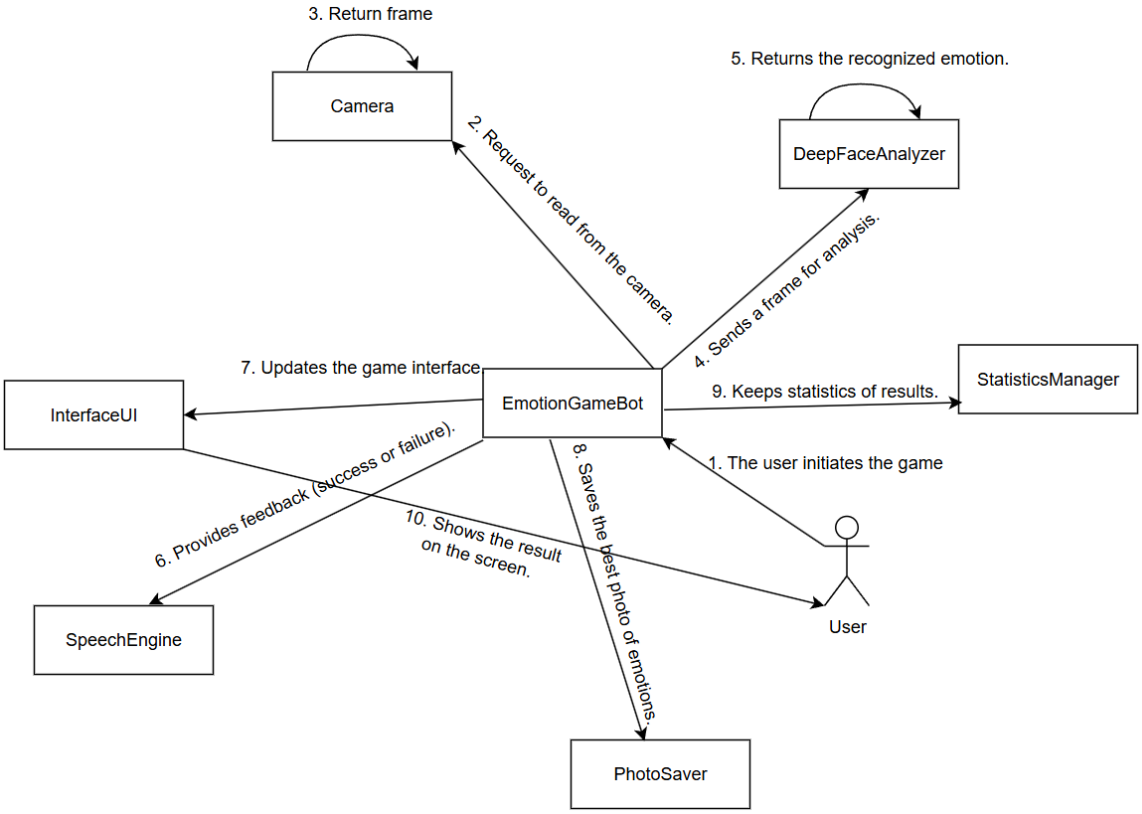

Рисунок 3.3 - Діаграма кооперації системи розпізнавання емоцій

На рисунку 3.3 показана діаграма кооперації – це така схема, де добре видно, як між собою взаємодіють головні об’єкти, з яких складається наша програма. Ця діаграма допомагає чітко зрозуміти, що саме відбувається всередині програми, коли користувач починає гру, а програма визначає емоцію.

В самому центрі цієї схеми знаходиться основний клас нашого проєкту – він називається EmotionGameBot. Цей клас фактично керує всім процесом гри, тобто саме він відповідає за запуск інших об’єктів, за контроль їх роботи, передачу даних і збереження інформації про результати. Якщо говорити простіше, це щось типу головного менеджера, який стежить, щоб все відбувалося в правильному порядку.

Отже, користувач запускає гру (це позначено на схемі цифрою 1). Після цього наш головний клас EmotionGameBot звертається до камери, точніше до об’єкта, який називається Camera. Це відбувається, щоб отримати з камери конкретний кадр – зображення обличчя користувача (2). Камера отримує цей запит, робить кадр і повертає його назад до нашого класу (3).

Потім отримане зображення передається до іншого дуже важливого модуля програми, який називається DeepFaceAnalyzer. Це той самий модуль, який і займається визначенням емоції користувача (4). Цей модуль працює на основі спеціальних алгоритмів і нейронних мереж, які раніше навчилися розпізнавати емоції за виразом обличчя. Коли DeepFaceAnalyzer завершує обробку, він повертає результат аналізу назад до головного класу (5). Тобто тепер програма вже знає, яку саме емоцію показує користувач.

Наступний крок, який виконує програма – це перевірка, чи правильно користувач виконав завдання (чи справді він показав ту емоцію, яку було потрібно). Якщо емоцію визначено правильно, програма запускає голосове підтвердження, яке каже користувачу, що все добре. За це відповідає модуль, який називається SpeechEngine (6).

Одночасно з цим інший модуль – InterfaceUI, тобто модуль інтерфейсу – оновлює інформацію на екрані (7). Тепер користувач бачить, що його результат зарахований, що завдання виконано правильно. Крім цього, найкращий кадр з отриманих під час гри зберігається за допомогою спеціального модуля під назвою PhotoSaver (8). Цей модуль відповідає за те, щоб фото зберігалося у файлі і його можна було потім переглянути.

Після того як зроблено всі ці кроки, результати (яка емоція була показана, кількість балів, успіх чи невдача користувача) записуються до окремого модуля – це StatisticsManager (9). Саме він зберігає інформацію про кожну гру, яку проводив користувач. В кінці-кінців всі дані, які були записані, система показує на екрані (10). Тобто користувач бачить, що саме вийшло в результаті його спроби, скільки він отримав балів і взагалі який прогрес за грою.

Така діаграма кооперації дуже корисна. Вона дозволяє побачити, як саме між собою пов’язані модулі програми, хто з них що робить, хто кому передає

дані і коли саме це відбувається. І головне, завдяки такій схемі легко зрозуміти, яку саме роль виконує кожен компонент системи. Це дуже зручно, якщо потрібно щось змінити, додати нові функції або знайти помилку.

На рисунку 3.4 наведено діаграму станів, яка описує повну логіку, переходів між станами у процесі гри.

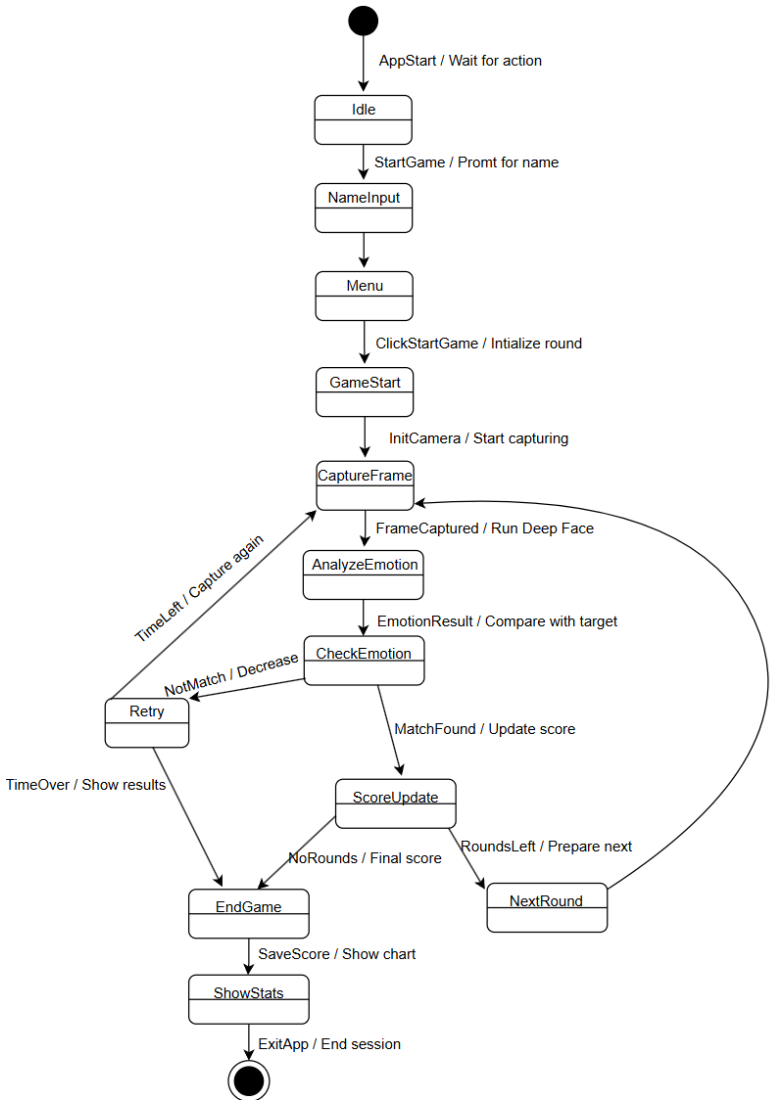

Рисунок 3.4 – Діаграма станів для логіки гри в системі розпізнавання емоцій

На рисунку 3.5 наведено діаграму класів інформаційної розважальної системи розпізнавання емоцій.

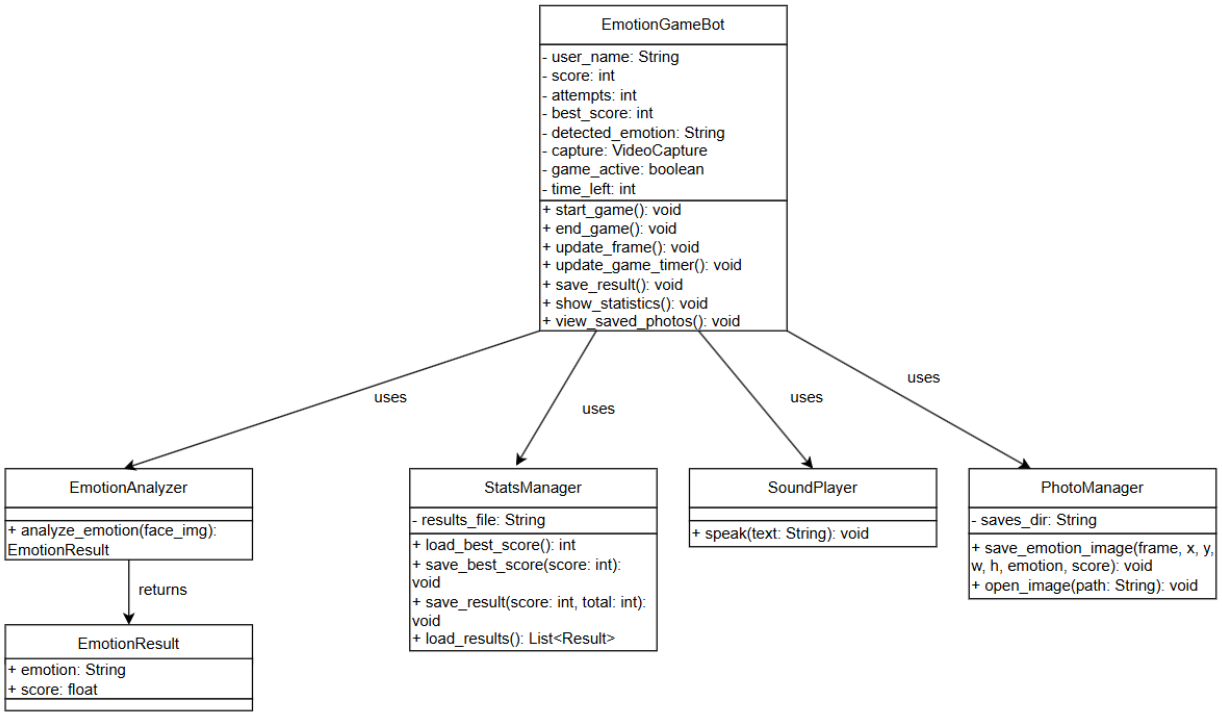

Рисунок 3.5 – Діаграма класів інформаційної розважальної системи розпізнавання емоцій

На рисунку 3.5 показано діаграму класів – тобто таку схему, яка дозволяє зрозуміти, які саме основні класи використані у програмі і як саме вони пов’язані між собою. Діаграма класів дуже важлива, бо з її допомогою можна чітко побачити структуру програми, і які в ній є головні та допоміжні елементи. Простими словами, така схема показує, з чого складається програма і хто що робить всередині.

У самому центрі цієї діаграми розташований найголовніший клас програми, який має назву EmotionGameBot. Це клас, який відповідає взагалі за всю гру: він запускає таймери, керує камерою, оновлює вікна, показує результати і відповідає за виклики інших частин програми. Тобто це такий керівник, який роздає завдання іншим класам, збирає результати, і відображає користувачу, що відбувається в програмі. Сам по собі цей клас не робить всю роботу сам, він взаємодіє ще з кількома допоміжними класами, які йому потрібні для різних завдань.

Тепер більш детально про ці додаткові класи, які видно на схемі:

- Перший клас має назву EmotionAnalyzer. Він займається тим, що аналізує вираз обличчя користувача. Для цього використовується модуль DeepFace – це спеціальна бібліотека, яка вміє визначати емоції за фотографією. Коли цей клас аналізує обличчя, він повертає спеціальний об’єкт, який називається EmotionResult. У цьому об’єкті є два параметри: сама назва емоції (наприклад, радість чи здивування) і ще окремо число, яке показує, наскільки точно ця емоція визначилась (тобто, наскільки програма впевнена в результаті).
- Наступний клас називається StatsManager. Він відповідає за статистику гри. Його завдання полягає в тому, щоб записувати результати кожного раунду, кращі досягнення користувача, і зберігати все це у файлах формату JSON. Це зроблено спеціально, щоб можна було легко переглядати статистику і бачити свій прогрес або результати попередніх ігор.
- Третій клас – це SoundPlayer. Він потрібен для озвучення результатів гри. В цьому класі використано одну з двох бібліотек – або gTTS, яка працює через Інтернет, або pyttsx3, яка озвучує текст без доступу до мережі. Таким чином користувач чує голосові підказки і повідомлення про те, що завдання виконано успішно, чи навпаки – помилково.
- Останній клас – PhotoManager. Він відповідає за збереження найкращих фотографій користувача з конкретними емоціями, які були правильно розпізнані. Крім цього, він може відкривати ці зображення, якщо користувач захоче переглянути свої фото після гри.

На діаграмі класи з’єднані стрілками, які мають позначку "uses" (тобто використовує). Це означає, що центральний клас – EmotionGameBot – просто користується послугами інших класів, коли йому це потрібно. Він не успадковує їх, а лише викликає їхні методи чи функції. Такий підхід дозволяє легко керувати всією системою, змінювати її частини або додавати нові без складнощів.

Взагалі, така архітектура класів дуже зручна, тому що вона дозволяє програмі бути модульною. Це означає, що будь-який клас можна змінювати окремо від інших. Наприклад, якщо виникне потреба в іншому модулі розпізнавання емоцій чи новому способі зберігання статистики – достатньо буде поміняти або доопрацювати відповідний клас, і при цьому не потрібно буде

сильно переписувати всю програму. Це дуже важливо для майбутнього розвитку та вдосконалення проєкту.

3.2 Розробка архітектури інформаційної розважальної системи розпізнавання емоцій

Архітектура програми – це по суті те, як вона влаштована всередині. Тобто з яких саме частин вона складається, які функції виконує кожна частина, як ці частини пов’язані між собою і як саме відбувається взаємодія всіх модулів. Під час розробки інформаційної розважальної системи з розпізнаванням емоцій особливу увагу було приділено зрозумілості всієї її структури.

По-перше, щоб легше було написати код, а по-друге, щоб можна було у майбутньому швидко щось змінювати або додавати якісь нові функції без переписування половини програми [26].

На початковому етапі розробки було визначено основні функціональні завдання програмного забезпечення. У спрощеному вигляді вони формулюються наступним чином: потрібно отримати відео з вебкамери, знайти на відео обличчя людини, потім визначити, яку саме емоцію показує користувач, порівняти цю емоцію з тією, яка потрібна у грі, і вже потім вивести результат користувачу. Також після цього треба озвучити повідомлення, зберегти найкращий кадр (тобто фото), записати усі ці результати для статистики і ще побудувати графік, щоб було зрозуміло, як змінюється результат з часом. Для забезпечення стабільної, швидкої та безперебійної роботи було прийнято рішення відмовитися від використання мережевих підключень.

Вся програма працює виключно на комп’ютері користувача без під’єднання до інтернету.

Сама програма створювалася мовою Python, точніше Python 3.10. Для її створення були використані такі популярні бібліотеки: PyQt5 (це така бібліотека, яка допомагає зробити графічний інтерфейс), OpenCV (вона працює з відео і камерами), DeepFace (це спеціальний модуль, який визначає емоції по обличчю),

matplotlib (для побудови графіків), gTTS (для озвучування результатів) і ще json (для запису статистики у файл).

Центральний і найважливіший клас, навколо якого будується вся програма, називається EmotionGameBot. По суті, це як головний менеджер всього проєкту. Саме він запускає вебкамеру, бере відео, запускає таймери, перевіряє, що робить користувач, показує результати гри на екрані, викликає модулі для аналізу емоцій, зберігає інформацію і будує графіки.

З метою спрощення обслуговування та підтримки програмного забезпечення було прийнято рішення винести окремі функції в невеликі незалежні модулі, що відповідає принципам модульної архітектури.

У нас є модуль EmotionAnalyzer, який займається лише аналізом виразів обличчя та розпізнає емоції. Другий модуль називається StatsManager – його завдання зберігати результати і вести статистику. Ще один модуль – PhotoManager – він відповідає за збереження кращих кадрів, зроблених під час гри. І є ще модуль SoundPlayer, який відповідає тільки за озвучення повідомлень користувачу.

Така структура програми була обрана не випадково. Це дуже зручний спосіб, тому що кожен модуль відповідає за свою конкретну задачу і не заважає іншим. Наприклад, якщо треба буде щось змінити у способі аналізу емоцій, не доведеться перебирати весь код. Можна просто зайти в модуль EmotionAnalyzer, внести там зміни і все буде працювати далі нормально. Це економить дуже багато часу.

Оскільки програма є локальною, то для її роботи не потрібно жодних серверів чи додаткових мережевих ресурсів. Вся інформація, яка обробляється (фото, результати, статистика), залишається тільки на пристрої користувача. Це дуже важливо, бо користувачам не треба хвилюватися за конфіденційність своїх даних, і сама програма працює швидко, без затримок, які бувають при підключенні до інтернету [27].

На рисунку 3.6 якраз зображена така загальна схема всієї архітектури програми. Можна легко побачити, як працює кожна частина: як центральний клас взаємодіє з іншими модулями, як відбувається процес обробки зображення,

збереження даних і як побудована логіка гри. Завдяки такій схемі дуже просто зрозуміти, що за чим йде і як саме модулі взаємодіють один з одним.

Таким чином, обрана архітектура забезпечила простоту, стабільність і логічну структуру системи. Вона повністю відповідає визначеним на початку вимогам та передбачає можливість подальшого розширення функціоналу або внесення змін без значних витрат часу.

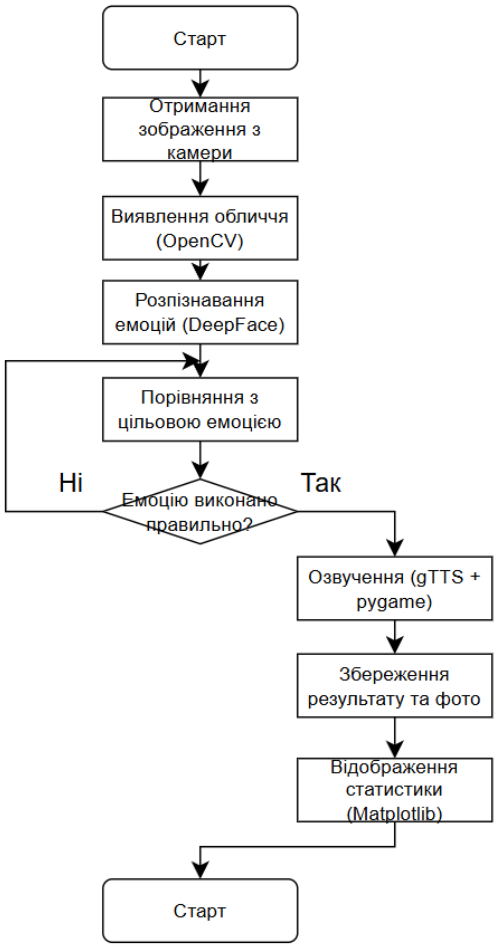

Рисунок 3.6 - Блок-схема архітектури роботи інформаційної розважальної системи розпізнавання емоцій

3.3 Програмна реалізація інформаційної розважальної системи розпізнавання емоцій

Програмна реалізація створеної системи базується на мові програмування Python з використанням модульного підходу. Основу становить графічний

інтерфейс, реалізований засобами PyQt5, і логіка гри, зосереджена в класі EmotionGameBot. У межах одного вікна реалізовано всі ключові функції: запуск гри, обробка емоцій, збереження результатів та статистики, виведення графіків і перегляд фото.

Після запуску програми користувачу пропонується ввести ім’я. Це реалізовано через вікно введення, створене за допомогою QInputDialog. Введене ім’я використовується для формування окремого JSON-файлу, в якому зберігатиметься персональна статистика проходжень гри.

Інтерфейс гри містить відео з камери, інструкції до гри, рахунок, таймер та три основні кнопки: «Почати гру», «Показати статистику», «Переглянути фото». Кнопки мають контрастні кольори, великі шрифти та чітку візуальну ієрархію, що робить інтерфейс зрозумілим навіть для нових користувачів.

Після натискання кнопки «Почати гру» активується метод start_game(), який обнуляє рахунок, генерує нову цільову емоцію та запускає таймер гри. У грі передбачено 5 раундів, кожен з яких має обмеження за часом. Таймер реалізовано через QTimer, який кожну секунду викликає метод update_game_timer(). Паралельно відбувається оновлення кадрів з камери методом update_frame(), у якому кожні півтори секунди виконується аналіз зображення за допомогою бібліотеки DeepFace.

Якщо в кадрі виявлено обличчя, виконується розпізнавання емоції. Визначена емоція порівнюється з цільовою. Якщо збіг є, гравець отримує бал, лунає голосове підтвердження через gTTS, а кадр із обличчям зберігається у папку best_emotions як зображення у форматі .png. При збігу або завершенні часу переходять до наступного раунду.

Після завершення гри викликається метод end_game(), який підводить підсумки. Статистика записується у форматі JSON в персональний файл гравця. У ньому зберігається дата, результат та кількість раундів. Для візуалізації прогресу реалізовано побудову графіка за допомогою matplotlib, де осі представляють дату і кількість балів.

Додатково реалізовано перегляд найкращих фото по кожній емоції, які зберігаються лише у разі досягнення кращого результату. Для цього створено

окреме вікно з QScrollArea, де зображення відображаються в одному рядку, мають мініатюру та відкриваються в один клік.

Також присутня функція озвучення результатів. Для цього використано бібліотеку gTTS, яка генерує звуковий файл з текстом підтвердження. Звук відтворюється через pygame.mixer. У разі відсутності підключення до Інтернету передбачена заміна на pyttsx3, який працює локально.

Уся логіка зберігається в одному класі, але розбита на окремі методи: start_game(), update_frame(), speak(), save_result(), show_statistics(), view_saved_photos() та інші. Кожен метод має свою відповідальність, що полегшує розуміння структури програми, її налагодження та доопрацювання.

У підсумку, реалізована система є цілісним інтерактивним продуктом, який реагує на дії користувача, фіксує успіхи, зберігає візуальні докази правильних відповідей і мотивує до повторного проходження через доступну статистику.

На одному з ключових етапів роботи системи відбувається захоплення кадру з камери та обробка обличчя для подальшого аналізу емоцій. Відповідний фрагмент коду зображено на рисунку 3.7.

```
def update_frame(self):  1 usage  new *
    if not self.capture.isOpened():
        return
    ret, frame = self.capture.read()
    if not ret:
        return
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = self.face_cascade.detectMultiScale(gray)
    for (x, y, w, h) in faces:
        roi = frame[y:y + h, x:x + w]
        now = QtCore.QTime.currentTime()
        if self.last_emotion_check.msecsTo(now) > 1500:
            try:
                result = DeepFace.analyze(roi, actions=['emotion'], enforce_detection=False)
                res = result[0] if isinstance(result, list) else result
                emotion = res['dominant_emotion']
                emotion_score = res['emotion'][emotion]
                self.detected_emotion = emotion
```

Рисунок 3.7 – Код обробки кадру з камери та розпізнавання емоцій

Після завершення кожної гри результати проходження зберігаються в локальний файл. Збереження відбувається у форматі JSON за допомогою методу save_result(), що показано на рисунку 3.8.

```
def save_result(self):  2 usages  new *
    data = []
    if os.path.exists(self.results_file):
        with open(self.results_file, "r") as f:
            data = json.load(f)
    data.append({"score": self.score, "total": self.total_rounds, "date": datetime.now().isoformat()})
    with open(self.results_file, "w") as f:
        json.dump(data, f, indent=2)
```

Рисунок 3.8 – Фрагмент коду збереження результатів у JSON-файл

У результаті роботи системи формується персональний файл користувача, де фіксується дата проходження гри, отримані бали та загальна кількість раундів. Приклад сформованого JSON-файлу подано на рисунку 3.9.


```
[
  {
    "score": 5,
    "total": 5,
    "date": "2025-05-22T14:55:12.722637"
  },
  {
    "score": 3,
    "total": 5,
    "date": "2025-05-23T15:27:07.499968"
  },
  {
    "score": 4,
    "total": 5,
    "date": "2025-05-28T15:27:07.499968"
  },
  {
    "score": 3,
    "total": 5,
    "date": "2025-05-30T15:27:07.499968"
  },
  {
    "score": 5,
    "total": 5,
    "date": "2025-05-31T15:27:07.499968"
  },
  {
    "score": 4,
    "total": 5,
    "date": "2025-06-01T15:27:07.499968"
  },
  {
    "score": 4,
    "total": 5,
    "date": "2025-06-01T15:27:07.499968"
  }
]
```


Рисунок 3.9 – Файл результатів користувача у форматі JSON

3.4 Інтерфейс користувача та взаємодія з системою

Графічний інтерфейс програми – це одна з найбільш важливих речей, особливо якщо мова йде про додаток для розваг. Якщо він незручний або виглядає складним, користувачі просто не захочуть ним користуватись, навіть якщо сама програма працює чудово. Тому для нас було принципово важливо зробити інтерфейс зрозумілим та максимально простим для користувача. У нашому додатку для розпізнавання емоцій цей момент був особливо важливим, тому що людина має швидко зрозуміти, як саме взаємодіяти з системою, що від неї вимагається, і де знаходяться потрібні функції.

Перед початком написання коду було доцільно створити макет майбутнього застосунку. Такий підхід є загальноприйнятою практикою в середовищі розробників, адже дозволяє заздалегідь візуалізувати інтерфейс користувача та структуру програми. Для цієї мети було обрано сервіс Figma, який вирізняється простотою у використанні та дає змогу швидко розробити якісний дизайн.

Під час створення макету детально продумано розташування ключових елементів інтерфейсу. Основне вікно, що демонструє зображення з вебкамери, розміщене в центральній частині – саме воно є орієнтиром для користувача щодо виразу емоцій та роботи системи. Також визначено зручне розташування кнопок керування: запуск основного режиму роботи, перегляд статистичних даних, доступ до галереї збережених зображень, а також таймер, який відображає залишок часу для виконання завдання.

З метою підвищення зручності передбачено використання текстових підказок – вони дозволяють швидко зорієнтуватися новим користувачам. Особливу увагу приділено простоті інтерфейсу: всі елементи інтуїтивно розміщені та доступні без додаткових дій. Це сприяє комфортній взаємодії навіть для користувачів без досвіду роботи з подібним програмним забезпеченням.

На рисунку 3.10 представлено дизайн-макет інтерфейсу, створений у середовищі Figma. Завдяки попередньому плануванню структура інтерфейсу

вийшла логічною та зручною для реалізації, що значно оптимізувало подальший процес розробки програмного продукту.

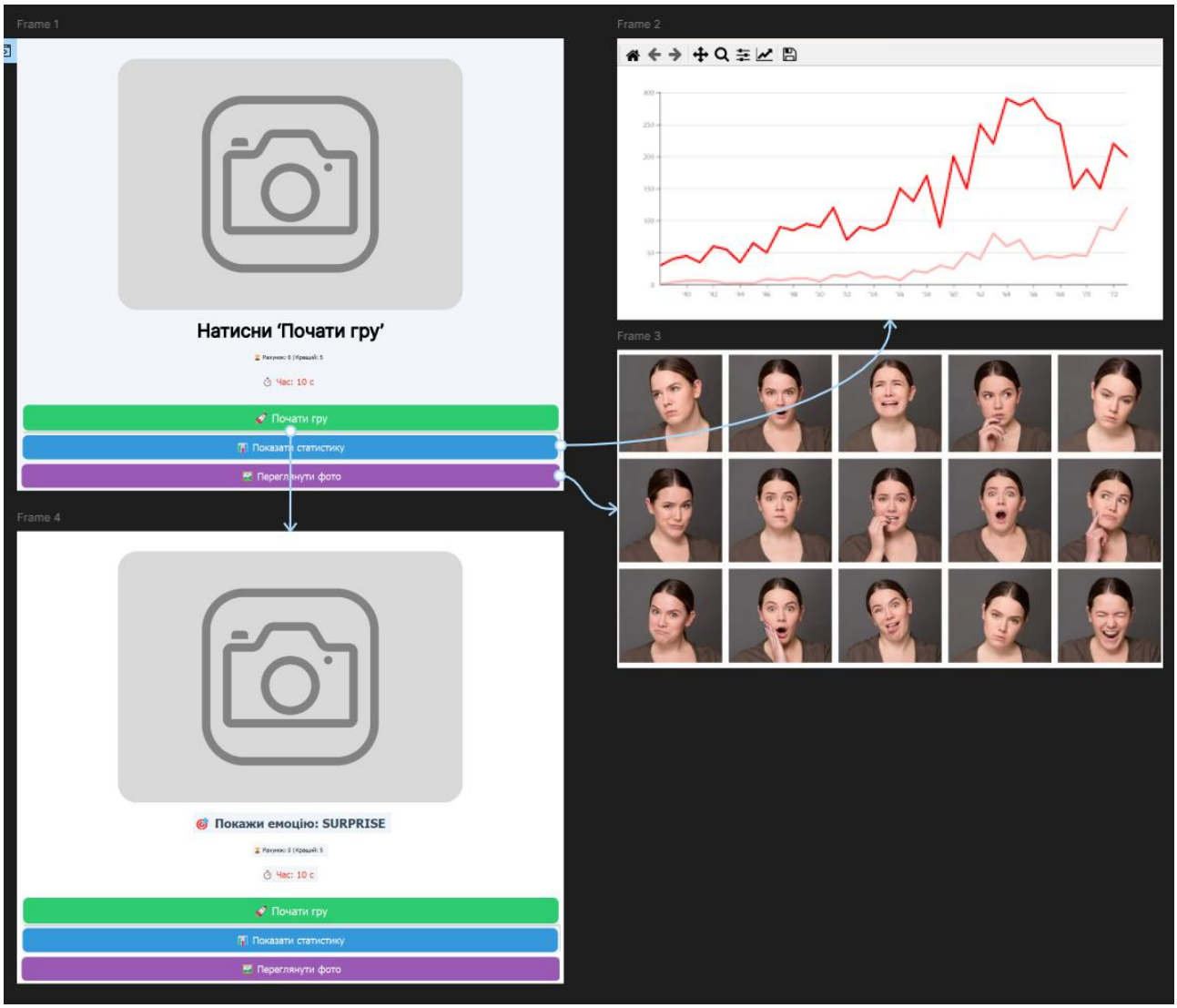

Рисунок 3.10 – Дизайн інтерфейсу користувача в Figma

Після затвердження дизайн-макету було розпочато реалізацію графічного інтерфейсу безпосередньо в програмному середовищі. Для цього обрано бібліотеку PyQt5, яка є ефективним інструментом для створення інтерфейсів у середовищі Python. У процесі реалізації було збережено ключові елементи, визначені на етапі макетування, що дозволило досягти високого рівня відповідності між початковим дизайном і кінцевим виглядом інтерфейсу. Завдяки цьому структура залишилася зрозумілою, а користувацький досвід — інтуїтивно зручним.

Основна частина екрану – це вікно з відеопотоком, яке показує користувача через вебкамеру. Це вікно займає центральну частину екрана, щоб користувач міг легко бачити себе, контролювати свій вираз обличчя і одразу бачити реакцію програми на свої емоції.

Під відеопотоком було розміщено блок із текстовими підказками. У ньому чітко вказується, яку саме емоцію слід продемонструвати на поточному етапі взаємодії. Це рішення сприяє кращому розумінню користувачем вимог системи та знижує ризик помилкового виконання дій.

Трохи нижче розташований таймер, який дозволяє орієнтуватися в часі, що залишився до завершення поточної сесії. Таймер дуже корисний, адже він показує, скільки часу ще лишилося на виконання завдання. Це дозволяє користувачу зорієнтуватись, чи варто поспішати, чи він має достатньо часу, щоб показати емоцію правильно.

Також важливим елементом є блок, де відображається поточний рахунок користувача. Завдяки йому гравець завжди бачить свої досягнення, скільки правильних емоцій він вже показав і як змінюється його результат протягом гри.

Крім цього, на екрані розташовані три великі кнопки: «Почати гру», «Показати статистику» та «Переглянути фото». Всі ці кнопки зроблені великими, зручними і розташовані в такому місці, щоб користувач легко їх помітив і зміг швидко натиснути, не шукаючи довго по екрану.

Особливу увагу було приділено кольорам інтерфейсу. Кожна кнопка має свій колір, який асоціюється з певною дією. Наприклад, кнопка запуску гри має зелений колір, тому що зелений зазвичай означає «старт». Кнопка статистики зроблена синьою, а кнопка перегляду фото – помаранчевою. Також на кнопках є піктограми, які додатково допомагають зрозуміти, що саме ця кнопка робить.

Ще одним важливим моментом був вибір шрифтів. Було свідомо обрано великі та добре читабельні шрифти, які забезпечують комфортне сприйняття текстової інформації навіть з відстані. Такий підхід спрямований на підвищення зручності користування інтерфейсом для широкого кола користувачів.

Це важливо, щоб користувачеві не доводилося нахилятися ближче до екрану чи витрачати час на розпізнавання написів.

Ігровий екран зроблено максимально просто. Там немає жодних зайвих елементів, які могли б відволікати користувача від гри. Все зроблено таким чином, щоб увага гравця була спрямована виключно на завдання – показати певну емоцію і отримати за це бали.

Коли користувач завершує спробу, програма одразу дає зворотний зв’язок: показує, наскільки вдало він виконав завдання, озвучує результат і оновлює рахунок на екрані. Це дозволяє гравцю одразу побачити, наскільки він був успішним, і зробити висновки на майбутнє.

На рисунку 3.11 представлено фактичний вигляд інтерфейсу, реалізованого з використанням бібліотеки PyQt5. Усі елементи були створені відповідно до попередньо розробленого макету, що дозволило забезпечити узгодженість дизайну та зручність користування.

В центрі розташоване основне вікно з камерою, нижче – інформаційні блоки з інструкцією, таймером та кнопками керування. Завдяки простоті та продуманому розміщенню всіх елементів інтерфейс вийшов максимально зручним, зрозумілим та приємним для користувача.

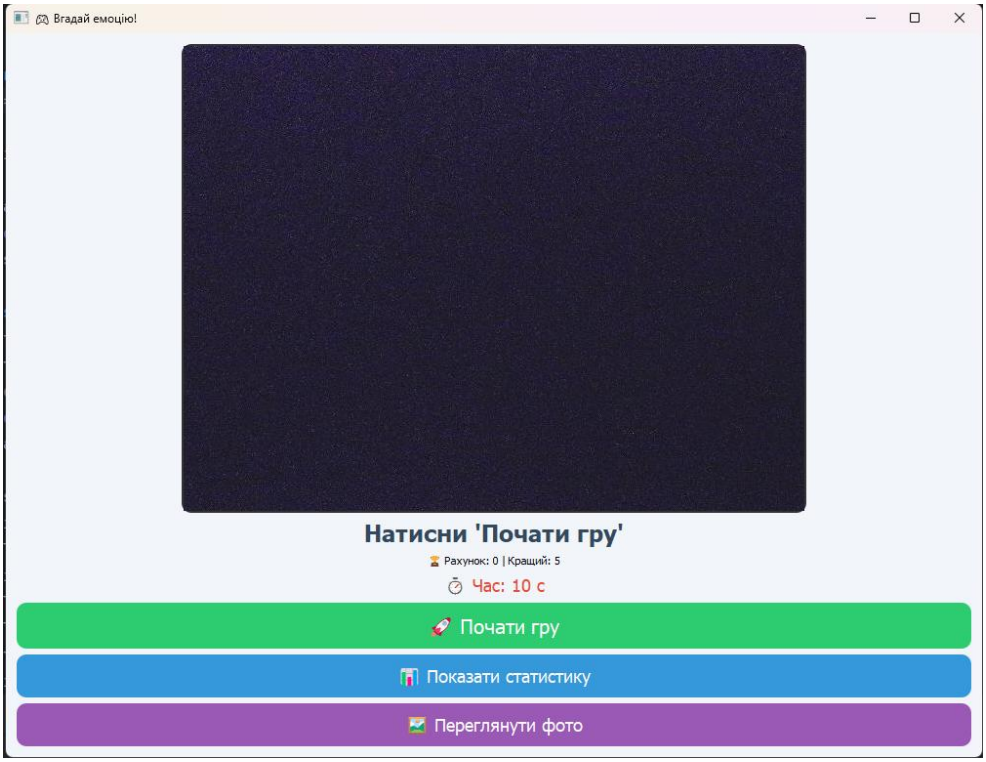

Рисунок 3.11 – Основне вікно інтерфейсу системи з розпізнаванням емоцій

Користувач має змогу переглянути результати проходження гри у вигляді графіка. Для цього використовуються дані, що зберігаються у JSON-файлі, та бібліотека matplotlib для побудови діаграми. На рисунку 3.12 показано приклад графічного відображення прогресу користувача.

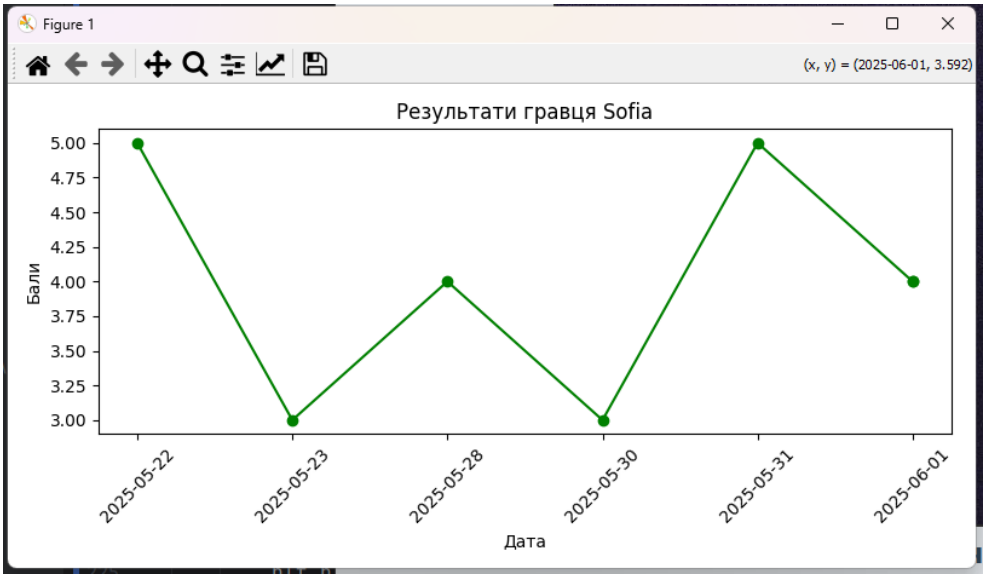

Рисунок 3.12 – Відображення статистики результатів гри користувача

Щоб користувач міг легко переглядати свої найкращі емоції, у програмі передбачене окреме вікно для фотографій, які зберігаються після вдалого проходження гри. Ці фото автоматично додаються в спеціальну галерею й відображаються у вигляді мініатюр у горизонтальному списку. Якщо користувач натискає на зображення, воно відкривається в повному розмірі, що дозволяє детально його роздивитися.

Уся структура інтерфейсу побудована так, щоб навіть недосвідченому користувачеві було зрозуміло, як ним користуватися. Елементи розташовані логічно, а дії супроводжуються короткими підказками: як текстовими, так і голосовими. Це допомагає швидко зорієнтуватися, навіть якщо людина відкриває додаток уперше.

Загалом інтерфейс програми поєднує простий зовнішній вигляд, продуману функціональність та зручність у користуванні. Такий підхід не тільки підсилює позитивний досвід взаємодії з додатком, а й робить саму гру доступною для широкого кола користувачів.

3.5 Алгоритм функціонування та обробка емоцій в реальному часі

Однією з важливих переваг створеної системи є те, що вона працює практично без затримок — усе відбувається в реальному часі. Як тільки камера захоплює відео, одразу запускається обробка, і вже за мить користувач бачить результат. Щоб усе працювало так швидко і без збоїв, логіка системи була побудована чітко та злагоджено: кадри постійно оновлюються, а емоції розпізнаються одразу ж після отримання зображення.

Гра стартує, коли користувач натискає кнопку «Почати гру». Система одразу активує таймер і починає опрацьовувати відео з камери. Зображення регулярно обробляється: воно перетворюється у відтінки сірого, і програма визначає, чи є на кадрі обличчя. Якщо є — відповідна частина вирізається та передається у DeepFace для аналізу.

Далі програма визначає емоцію, яка, на її думку, є на обличчі. Якщо ця емоція збігається з тією, яку мав показати користувач — гра зараховує відповідь як правильну, зберігає бал і переходить до наступного кроку. Якщо ж емоція не відповідає — гравець має ще кілька спроб до завершення часу.

Ми врахували реальні обставини, які можуть завадити точному розпізнаванню: наприклад, якщо обличчя частково перекрите, світло змінюється або людина трохи повертає голову. Щоб система не зависала і не перевантажувалась, кадри обробляються з паузою в 1,5 секунди — цього цілком достатньо, щоб зберегти точність і водночас уникнути надмірного навантаження.

Уся робота системи відбувається на комп’ютері користувача. Немає потреби в інтернеті чи доступі до якихось серверів. Це підвищує швидкість і, що не менш важливо, — захищає особисті дані. Для озвучення результатів використовується або gTTS, або pyttsx3 — залежно від того, чи є з’єднання з мережею.

Кожен етап гри зберігається у файл JSON. Якщо гравець показав правильну емоцію, система також робить знімок обличчя з найкращим виразом. Після цього оновлюється інтерфейс і гра переходить до нового завдання.

Саме завдяки такій логіці дій користувач не відчуває затримок — усе працює плавно, послідовно і без зайвих очікувань. Результат видно одразу, а взаємодія з програмою здається «живою» і природною.

3.6 Висновки

У цьому розділі детально висвітлено ключові технічні аспекти створення застосунку, який розпізнає емоції користувача. Описано, як працюють усі модулі, яким чином вони взаємодіють, і яку функцію виконує кожен із них. Також пояснено, чому було обрано саме це програмне середовище, мову та бібліотеки, а UML-діаграми допомогли краще зрозуміти загальну архітектуру.

Сама програма працює як локальний застосунок, тобто не потребує підключення до інтернету. Це зроблено для забезпечення максимальної швидкодії. Відео з камери обробляється одразу, а емоція визначається практично без затримок. Система враховує зміну освітлення, рухи голови, часткове перекриття обличчя — завдяки цьому вона добре адаптується до реальних умов. Усі обчислення виконуються локально, а результати гри записуються в JSON. Крім того, зберігається найкраще фото з емоцією користувача.

Інтерфейс розроблявся так, щоб користувач зміг легко орієнтуватися в ньому без додаткових інструкцій. Основні елементи (кнопки, підказки, таймер і вікно відео) розташовані логічно, що дозволяє швидко зорієнтуватися навіть під час першого запуску. Також реалізовано зручний перегляд статистики, що стимулює користувача повертатися до гри.

Особливий акцент зроблено на тому, як система реагує на дії гравця в реальному часі. Уся логіка побудована так, щоб аналіз кадрів, перевірка міміки та реагування на емоцію відбувалися без пауз. Це особливо важливо в інтерактивних сценаріях, де навіть короткі затримки можуть вплинути на комфорт.

Загалом проєкт демонструє вдале поєднання зручності, швидкості й зрозумілості. Користуватися програмою легко навіть тим, хто бачить її вперше,

а всі процеси відбуваються плавно та без складнощів. Саме це дозволяє розглядати її як повноцінний інструмент для розважального використання.

ВИСНОВКИ

Результатом виконання бакалаврської кваліфікаційної роботи стало створення інформаційної розважальної системи, яка забезпечує розпізнавання емоцій користувача в реальному часі з використанням комп’ютерного зору та інструментів штучного інтелекту. Система функціонує локально, без залежності від зовнішніх сервісів, що дозволяє гарантувати швидкість реагування та конфіденційність даних користувача.

Розроблена інформаційна система охоплює повний життєвий цикл: від збору зображення з камери до аналізу емоцій, виведення результату у вигляді голосу чи графіка та збереження статистики. Порівняно з наявними рішеннями, запропонована система має переваги в автономності, простоті інтерфейсу та адаптації до неінтернет-залежних середовищ.

Система продемонструвала хороші показники стабільності, швидкодії та зручності використання. Візуальний інтерфейс, розроблений у PyQt5, забезпечує доступність навіть для користувачів без технічної підготовки. Всі рішення щодо вибору бібліотек та архітектурної моделі були обґрунтовані на основі порівняльного аналізу альтернатив.

Серед обмежень реалізації можна назвати відсутність можливості роботи з кількома користувачами одночасно та відсутність адаптивності до різних типів камер або освітлення. Водночас це створює перспективи для подальшого розвитку, які можуть включати підтримку багатокористувацького режиму, інтеграцію з хмарними сервісами або мобільну версію.

Розроблена інформаційна система є прикладом ефективного поєднання елементів штучного інтелекту та зручного інтерфейсу користувача у форматі локального настільного застосунку. Вона демонструє потенціал технологій емоційної взаємодії для розважальних і навчальних застосувань, а також може стати основою для більш складних емоційно-орієнтованих рішень у майбутньому.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Фіяло С.Г., Горячев Г.В. Інтерактивна інформаційно-розважальна система розпізнавання емоцій на основі виразу обличчя. Міжнародна науково-практична інтернет-конференція «Молодь в науці: дослідження, проблеми, перспективи (МН- 2025)». URL: https://conferences.vntu.edu.ua/index.php/mn/mn2025/paper/viewFile/24602/20794

2. Використання нейронної мережі для розпізнавання емоцій 2022 - URL: https://er.nau.edu.ua/items/1e9f4cd3-4f60-4847-8d0e-e63b12ed575a

3. Ekman, P. (1992). An argument for basic emotions. Cognition & Emotion, 6(3–4), 169–200.

4. Cowie, R., Douglas-Cowie, E., Savvidou, S., et al. (2000). 'FEELTRACE': An instrument for recording perceived emotion in real time. ISCA Workshop on Speech and Emotion.

5. McStay, A. (2018). Emotional AI: The Rise of Empathic Media. SAGE Publishing. URL: https://www.researchgate.net/publication/326294717_Emotional_AI_The_Rise_of_Empathic_Media

6. Що нам потрібно знати про штучний інтелект у розпізнаванні емоцій у 2024 році URL: https://uk.shaip.com/blog/what-we-need-to-know-about-ai-in-emotion-recognition/

7. Affectiva URL: https://www.affectiva.com/

8. FaceReader URL: https://facereader-online.com/

9. Microsoft Azure Emotion API URL: https://thirdeyedata.ai/azure-emotion-api/

10. What is the Azure AI Face service? URL: https://learn.microsoft.com/en-us/azure/ai-services/computer-vision/overview-identity

11. Replika URL: https://replika.com/

12. Що таке IT-інфраструктура, якою вона буває і з чого складається URL: https://cases.media/article/sho-take-it-infrastruktura-yakoyu-vona-buvaye-i-z-chogo-skladayetsya

13. Riverbank Computing. (2023). PyQt5 Reference Guide. URL:

https://www.riverbankcomputing.com/static/Docs/PyQt5/

14. Bradski, G. (2000). The OpenCV Library. Dr. Dobb's Journal of Software Tools.

15. PyQt5 URL: https://pypi.org/project/PyQt5/

16. OpenCV URL: https://opencv.org/

17. Matplotlib: Visualization with Python URL: https://matplotlib.org/

18. DeepFace: A Popular Open Source Facial Recognition Library URL: https://viso.ai/computer-vision/deepface/

19. Навчальний посібник із PyQt5 із прикладами: проектування графічного інтерфейсу за допомогою PyQt Python URL: https://www.guru99.com/uk/pyqt-tutorial.html

20. Що таке комп'ютерний зір (Computer Vision CV) URL: https://thetransmitted.com/adlucem/shho-take-kompyuternyj-zir-computer-vision-cv/

21. Працюємо з нейромережами: як ми навчили камеру розпізнавати обличчя, щоб обійтися без перепусток в офіс URL: https://dou.ua/lenta/articles/office-face-recognition-system/

22. DeepFace для вдосконаленого розпізнавання обличчя URL: https://www.unite.ai/uk/deepface-for-advanced-facial-recognition/

23. Як будувати UML-діаграми. Розбираємо три найпопулярніші варіанти URL: https://dou.ua/forums/topic/40575/

24. UML для бізнес-моделювання: для чого потрібні діаграми процесів URL: https://evergreens.com.ua/ua/articles/uml-diagrams.html

25. Для чого потрібні UML діаграми? UML: https://foxminded.ua/uml-diagramy/

26. Архітектура та проектування програмного забезпечення UML: https://surli.cc/simquc

27. Zhang, K., Zhang, Z., & Li, Z. (2016). Joint face detection and alignment using multitask cascaded convolutional networks. IEEE Signal Processing Letters, 23(10), 1499–1503.

Додаток А

(обов'язковий)

Технічне завдання на бакалаврську кваліфікаційну роботу

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації

ЗАТВЕРДЖУЮ
Завідувач кафедри САІТ
_____ д.т.н., проф. Віталій МОКІН
"____"_____ 2025 р.

ТЕХНІЧНЕ ЗАВДАННЯ
на бакалаврську дипломну роботу
ІНФОРМАЦІЙНА РОЗВАЖАЛЬНА СИСТЕМА РОЗПІЗНАВАННЯ ЕМОЦІЙ
08-34.БКР.024.00.000 ТЗ

Керівник: к.т.н., доц.
_____ Георгій ГОРЯЧЕВ
«__» _____ 2025 р.
Розробив студент гр. 2ІСТ-20б
_____ Софія ФІЯЛО
«__» _____ 2025 р.

Вінниця 2025

1. Підстава для проведення робіт.

Підставою для виконання роботи є наказ №__ по ВНТУ від «__» _____2025р., та індивідуальне завдання на БКР, затверджене протоколом №__ засідання кафедри САІТ від «__» _____ 2025р.

2. Джерела розробки:

1) Андрущенко Т. В., Баранчук А. В. Технології штучного інтелекту: підручник. – Вінниця: ВНТУ, 2020. – 312 с.;

2) Бобровник С. В. Основи програмування на Python : навч. посіб. / С. В. Бобровник. – Київ : Ліра-К, 2020. – 210 с.

3) Сидоренко В.Л. Інформаційні системи: навч. посібник. – К.: Ліра-К, 2021. – 248 с.

3. Мета і призначення роботи.

Метою дослідження є розроблення інформаційної розважальної системи, здатної розпізнавати емоції користувача в реальному часі для створення інтерактивної взаємодії.

4. Вихідні дані для проведення робіт:

Зображення з вебкамери користувача, бібліотеки DeepFace, OpenCV, PyQt5, JSON-дані зі статистикою проходження гри, середовище розробки PyCharm.

5. Методи дослідження:

Застосування методів комп'ютерного зору, елементів машинного навчання, синтезу мовлення, аналізу емоцій за допомогою бібліотек Python.

6. Етапи роботи і терміни їх виконання:

a) Аналіз предметної області _____ – _____
b) Аналіз проблематики емоційної взаємодії _____ – _____
c) Вибір оптимальних інформаційних технологій _____ – _____
d) Розроблення інформаційної системи розпізнавання емоцій на програмні застосунки _____ – _____
e) Оформлення матеріалів до захисту БКР _____ – _____

7. Очікувані результати та порядок реалізації

Отримання повнофункціональної розважальної системи, яка розпізнає емоції користувача, озвучує результат, веде статистику, має зрозумілий графічний інтерфейс.

8. Вимоги до розробленої документації

Текстова та ілюстративна частини оформлені згідно з «Методичними вказівками до виконання бакалаврських кваліфікаційних робіт для студентів спеціальностей: 126 «Інформаційні системи та технології» (освітня програма «Прикладні інформаційні технології»)».

9. Порядок приймання роботи

Публічний захист «___» _____ 2025 р.
Початок розробки «___» _____ 2025 р.
Граничні терміни виконання БКР «___» _____ 2025 р.

Розробила студентка групи 2ІСТ-21б _____ Софія ФІЯЛО

Додаток Б

(обов'язковий)

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: «Інформаційна розважальна система розпізнавання емоцій»

Тип роботи: бакалаврська кваліфікаційна робота

Підрозділ: кафедра САІТ, ФІІТА, гр. 2ІСТ-21б

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 0,52 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне):

- Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

- У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

- У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

Віталій МОКІН, зав. каф. САІТ _____
(підпис)

Євгеній КРИЖАНОВСЬКИЙ, доц. каф. САІТ _____
(підпис)

Особа, відповідальна за перевірку _____ Сергій ЖУКОВ
(підпис)

З висновком експертної комісії ознайомлена

Керівник _____ Георгій ГОРЯЧЕВ, к.т.н., доц. каф. САІТ
(підпис)

Здобувач _____ Софія ФІЯЛО
(підпис)

Додаток В

(довідниковий)

Фрагмент лістингу програми

```
import sys
import os
import cv2
import json
import uuid
import random
from datetime import datetime
from PyQt5 import QtWidgets, QtGui, QtCore
from deepface import DeepFace
from gtts import gTTS
import pygame
import matplotlib.pyplot as plt

class EmotionGameBot(QtWidgets.QWidget):
    def __init__(self):
        super().__init__()
        self.user_name = self.ask_user_name()
        self.settings_file = "game_settings.json"
        self.results_file = f"results_{self.user_name}.json"
        self.saves_dir = "best_emotions"
        os.makedirs(self.saves_dir, exist_ok=True)
        self.best_emotion_scores = {}

        self.running = True
        self.voice_index = 0
        self.target_emotion = random.choice(['happy', 'sad', 'angry', 'surprise'])
        self.score = 0
        self.attempts = 0
        self.total_rounds = 5
        self.time_per_round = 10
```

```
        self.time_left = self.time_per_round
        self.game_active = False
        self.best_score = self.load_best_score()
        self.detected_emotion = ""

        self.initUI()
        self.capture = cv2.VideoCapture(0)

        self.timer = QtCore.QTimer()
        self.timer.timeout.connect(self.update_frame)
        self.timer.start(30)

        self.game_timer = QtCore.QTimer()
        self.game_timer.timeout.connect(self.update_game_timer)
        self.game_timer.start(1000)

        self.face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
"haarcascade_frontalface_default.xml")
        self.last_emotion_check = QtCore.QTime.currentTime()

        pygame.mixer.init()

    def ask_user_name(self):
        name, ok = QtWidgets.QInputDialog.getText(self, "Ім'я гравця", "Введіть своє ім'я:")
        return name if ok and name else f"Player_{random.randint(100,999)}"

    def initUI(self):
        self.setWindowTitle('🎮 Вгадай емоцію!')
        self.setGeometry(200, 100, 1000, 740)
        self.setStyleSheet("background-color: #f2f6fa;")

        self.video_label = QtWidgets.QLabel(self)
        self.video_label.setFixedSize(640, 480)
        self.video_label.setAlignment(QtCore.Qt.AlignCenter)
        self.video_label.setStyleSheet("border: 2px solid #333; border-radius: 10px;")
```

```
        self.info_label = QtWidgets.QLabel("Натисни 'Почати гру'")
        self.info_label.setAlignment(QtCore.Qt.AlignCenter)
        self.info_label.setStyleSheet("font-size: 24px; font-weight: bold; color: #34495e;")

        self.score_label = QtWidgets.QLabel(f" Рахунок: {self.score} | Кращий: {self.best_score}")
        self.score_label.setAlignment(QtCore.Qt.AlignCenter)

        self.timer_label = QtWidgets.QLabel(f" Час: {self.time_left} с")
        self.timer_label.setAlignment(QtCore.Qt.AlignCenter)
        self.timer_label.setStyleSheet("font-size: 18px; color: #e74c3c;")

        self.start_button = QtWidgets.QPushButton(" Почати гру")
        self.start_button.clicked.connect(self.start_game)
        self.start_button.setStyleSheet("padding: 10px; background: #2ecc71; color: white; font-
size: 20px; border-radius: 10px;")

        self.stats_button = QtWidgets.QPushButton(" Показати статистику")
        self.stats_button.clicked.connect(self.show_statistics)
        self.stats_button.setStyleSheet("padding: 10px; background: #3498db; color: white; font-
size: 18px; border-radius: 10px;")

        self.view_photos_button = QtWidgets.QPushButton(" Переглянути фото")
        self.view_photos_button.clicked.connect(self.view_saved_photos)
        self.view_photos_button.setStyleSheet("padding: 10px; background: #9b59b6; color: white;
font-size: 18px; border-radius: 10px;")

        layout = QtWidgets.QVBoxLayout()
        layout.addWidget(self.video_label, alignment=QtCore.Qt.AlignCenter)
        layout.addWidget(self.info_label)
        layout.addWidget(self.score_label)
        layout.addWidget(self.timer_label)
        layout.addWidget(self.start_button)
        layout.addWidget(self.stats_button)
        layout.addWidget(self.view_photos_button)
```

```
    self.setLayout(layout)

def start_game(self):
    self.score = 0
    self.attempts = 0
    self.time_left = self.time_per_round
    self.game_active = True
    self.next_target()

def next_target(self):
    self.target_emotion = random.choice(['happy', 'sad', 'angry', 'surprise'])
    self.info_label.setText(f" Покажи емоцію: {self.target_emotion.upper()}")
    self.score_label.setText(f" Рахунок: {self.score} | Кращий: {self.best_score}")
    self.time_left = self.time_per_round

def update_game_timer(self):
    if not self.game_active:
        return
    self.time_left -= 1
    self.timer_label.setText(f" Час: {self.time_left} с")
    if self.time_left <= 0:
        self.attempts += 1
        if self.attempts >= self.total_rounds:
            self.end_game()
        else:
            self.next_target()

def update_frame(self):
    if not self.capture.isOpened():
        return
    ret, frame = self.capture.read()
    if not ret:
        return
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = self.face_cascade.detectMultiScale(gray)
```

```
        for (x, y, w, h) in faces:
            roi = frame[y:y + h, x:x + w]
            now = QtCore.QTime.currentTime()
            if self.last_emotion_check.msecsTo(now) > 1500:
                try:
                    result = DeepFace.analyze(roi, actions=['emotion'], enforce_detection=False)
                    res = result[0] if isinstance(result, list) else result
                    emotion = res['dominant_emotion']
                    emotion_score = res['emotion'][emotion]
                    self.detected_emotion = emotion

                    if self.game_active and self.detected_emotion == self.target_emotion:
                        self.score += 1
                        self.attempts += 1
                        self.save_emotion_image(frame, x, y, w, h, emotion, emotion_score)
                        self.speak(True)
                        if self.attempts >= self.total_rounds:
                            self.end_game()
                        else:
                            self.next_target()
                except Exception as e:
                    print("DeepFace помилка:", e)
                self.last_emotion_check = now

            cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
            cv2.putText(frame, self.detected_emotion, (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.9,
(0, 255, 0), 2)

        rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        h, w, ch = rgb.shape
        bytes_per_line = ch * w
        qt_img = QtGui.QImage(rgb.data, w, h, bytes_per_line, QtGui.QImage.Format_RGB888)
        pix = QtGui.QPixmap.fromImage(qt_img)
        self.video_label.setPixmap(pix)
```

```
def save_emotion_image(self, frame, x, y, w, h, emotion, score):
    current_best = self.best_emotion_scores.get(emotion, 0)
    if score > current_best:
        self.best_emotion_scores[emotion] = score
        filename = os.path.join(self.saves_dir, f"{self.user_name}_{emotion}_BEST.png")
        face_img = frame[y:y + h, x:x + w]
        cv2.imwrite(filename, face_img)

def speak(self, correct=True):
    txt = "Правильно!" if correct else f"Твій результат: {self.score} із {self.total_rounds}"
    try:
        tts = gTTS(text=txt, lang='uk')
        filename = f"voice_{uuid.uuid4()}.mp3"
        tts.save(filename)
        sound = pygame.mixer.Sound(filename)
        sound.play()
        while pygame.mixer.get_busy():
            pygame.time.Clock().tick(10)
        os.remove(filename)
    except Exception as e:
        print(f"TTS помилка: {e}")

def end_game(self):
    self.game_active = False
    self.info_label.setText("Гру завершено ")
    if self.score > self.best_score:
        self.best_score = self.score
        self.save_best_score()
    self.save_result()

def load_best_score(self):
    try:
        with open("best_score.txt", "r") as f:
            return int(f.read())
```

```
        except:
            return 0

    def save_best_score(self):
        with open("best_score.txt", "w") as f:
            f.write(str(self.best_score))

    def save_result(self):
        data = []
        if os.path.exists(self.results_file):
            with open(self.results_file, "r") as f:
                data = json.load(f)
        data.append({"score": self.score, "total": self.total_rounds, "date":
datetime.now().isoformat()})
        with open(self.results_file, "w") as f:
            json.dump(data, f, indent=2)

    def show_statistics(self):
        if not os.path.exists(self.results_file):
            QtWidgets.QMessageBox.information(self, "Статистика", "Немає збережених
результатів")
            return
        with open(self.results_file, "r") as f:
            data = json.load(f)
        dates = [item["date"][:10] for item in data]
        scores = [item["score"] for item in data]
        plt.figure(figsize=(8, 4))
        plt.plot(dates, scores, marker='o', linestyle='-', color='green')
        plt.title(f"Результати гравця {self.user_name}")
        plt.xlabel("Дата")
        plt.ylabel("Бали")
        plt.xticks(rotation=45)
        plt.tight_layout()
        plt.show()
```

```
    def view_saved_photos(self):
        folder = self.saves_dir
        if not os.path.exists(folder) or not any(fname.lower().endswith(('.png', '.jpg')) for fname in
os.listdir(folder)):
            QtWidgets.QMessageBox.information(self, "Фото", "Немає збережених фото.")
            return

        viewer_window = QtWidgets.QWidget()
        viewer_window.setWindowTitle("Збережені емоції")
        layout = QtWidgets.QVBoxLayout()
        scroll_area = QtWidgets.QScrollArea()
        scroll_area.setWidgetResizable(True)
        container = QtWidgets.QWidget()
        img_layout = QtWidgets.QHBoxLayout()
        img_layout.setAlignment(QtCore.Qt.AlignTop)

        for file in sorted(os.listdir(folder)):
            if file.lower().endswith(('.png', '.jpg')):
                full_path = os.path.join(folder, file)
                pix = QtGui.QPixmap(full_path).scaledToHeight(200,
QtCore.Qt.SmoothTransformation)
                label = QtWidgets.QLabel()
                label.setPixmap(pix)
                label.setAlignment(QtCore.Qt.AlignTop)
                label.setCursor(QtGui.QCursor(QtCore.Qt.PointingHandCursor))
                label.mousePressEvent = lambda event, path=full_path: os.startfile(path)
                img_layout.addWidget(label)

        container.setLayout(img_layout)
        scroll_area.setWidget(container)
        layout.addWidget(scroll_area)
        viewer_window.setLayout(layout)
        viewer_window.resize(900, 300)
        viewer_window.show()
```

```
        self.viewer_window = viewer_window

    def closeEvent(self, event):
        self.running = False
        self.timer.stop()
        self.game_timer.stop()
        if self.capture.isOpened():
            self.capture.release()
        pygame.mixer.quit()
        event.accept()

if __name__ == '__main__':
    app = QtWidgets.QApplication(sys.argv)
    win = EmotionGameBot()
    win.show()
    sys.exit(app.exec_())
```

Додаток Г

(обов’язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

ІНФОРМАЦІЙНА РОЗВАЖАЛЬНА СИСТЕМА РОЗПІЗНАВАННЯ ЕМОЦІЙ

Нормоконтроль: к.т.н., доцент

_____ Сергій ЖУКОВ

«___» _____ 2025 р.

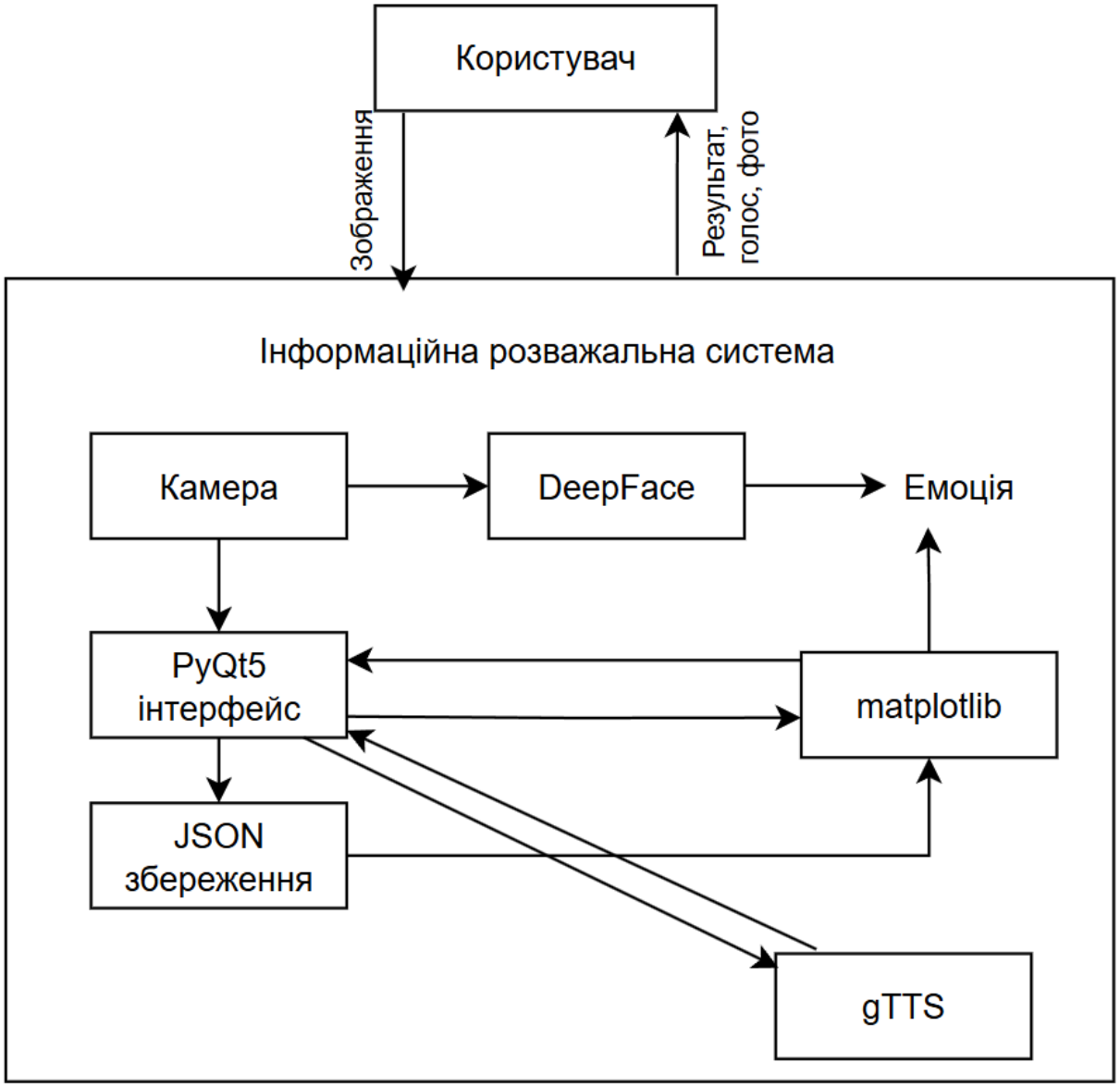

Рисунок Г.1 – Контекстна діаграма інформаційної розважальної системи розпізнавання емоцій

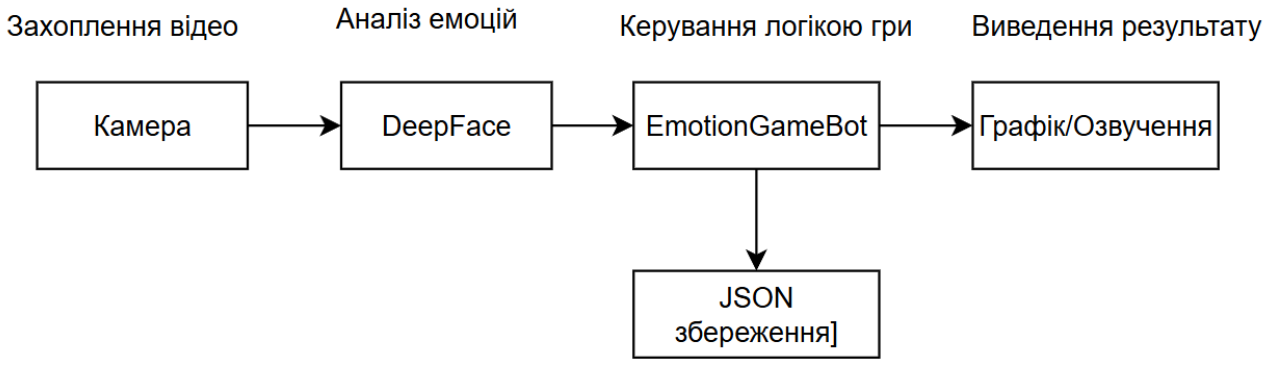

Рисунок Г.2 – Діаграма модулів системи

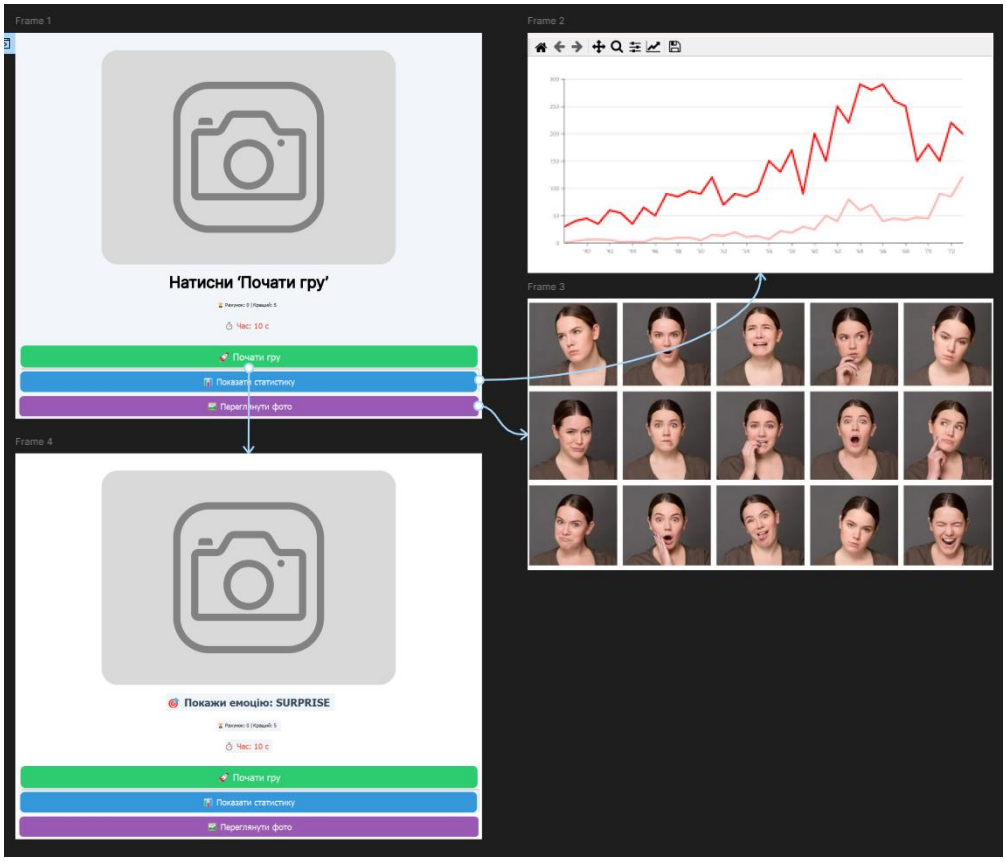

Рисунок Г.3 – Макет-дизайн інтерфейсу (Figma)

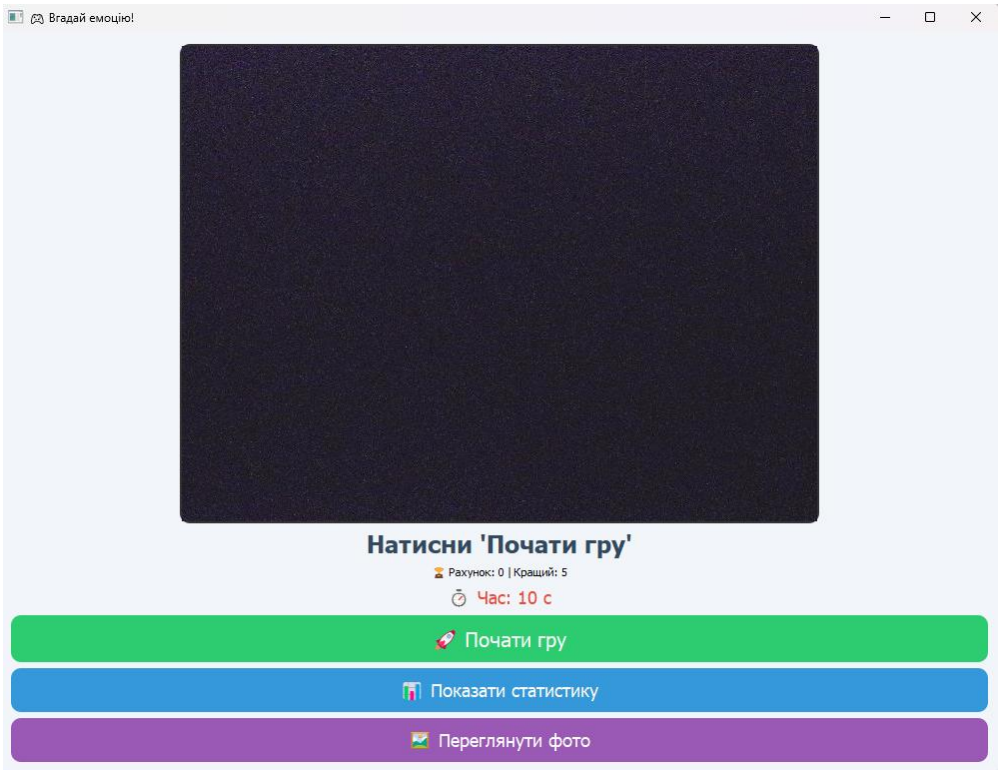

Рисунок Г.4 – Screenshot застосунку «Головне меню»

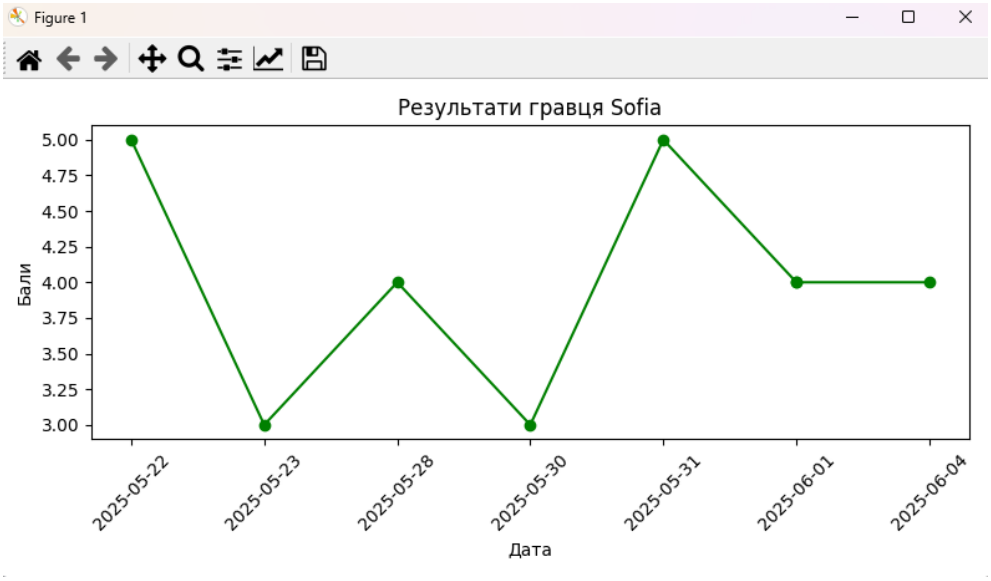

Рисунок Г.5 – Screenshot застосунку «Перегляд статистики»

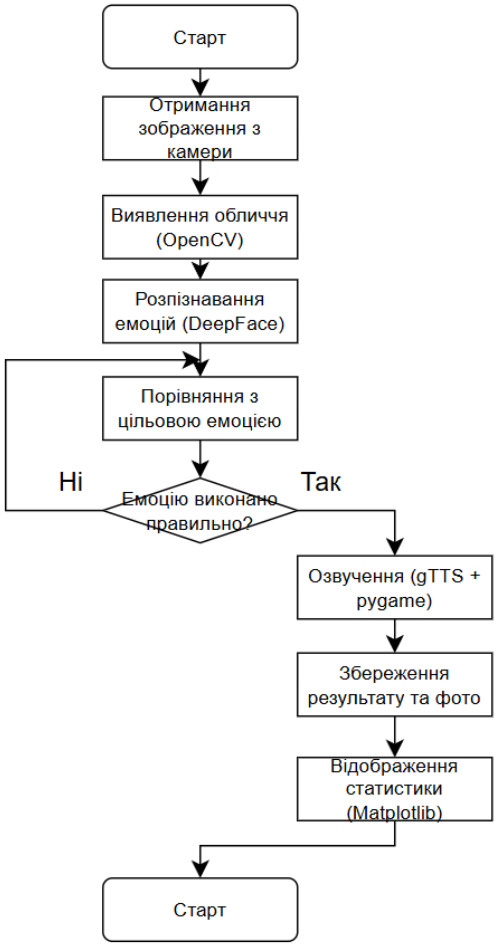

Рисунок Г.6 – Блок-схема алгоритму роботи програми