Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

РОЗРОБЛЕННЯ ВЕБЗАСТОСУНКУ РОЗПІЗНАВАННЯ ТА ІДЕНТИФІКАЦІЇ КОРИСТУВАЧІВ БАНКІВСЬКОЇ СИСТЕМИ
(тема)

Виконав:
студент 4 курсу, групи ІТІНФ-20-1
Ларін І.П.
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник доц. Творошенко І.С.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Кобилін О.А.
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Ларіну Івану Павловичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення вебзастосунку розпізнавання та ідентифікації користувачів банківської системи

затверджена наказом університету від 20 травня 2024 року № 464 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 26 травня 2024 р.

3. Вихідні дані до роботи науково-методична та науково-технічна література, матеріали конференцій, дані інтернет-мережі, бібліотека комп'ютерного зору з відкритим кодом OpenCV та DeepFace, фреймворк Spring, фреймоворк для розробки вебзастосунків Flask.

4. Перелік питань, що потрібно опрацювати в роботі

1. Аналіз існуючих вебзастосунків розпізнавання та ідентифікації користувачів інформаційних систем в україні та за кордоном.

2. Моделювання структури вебзастосунку розпізнавання та ідентифікації користувачів банківської системи.

3.Розроблення вебзастосунку розпізнавання та ідентифікації користувачів банківської системи.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Актуальність проблеми розроблення вебзастосунку розпізнавання та ідентифікації користувачів банківської системи, постановка задачі, схема структури вебзастосунку, схема структури бази даних, тестові зображення.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	08.04.2024	
2	Аналіз завдання, підбір літератури	08.04.24-17.04.24	
3	Аналіз літератури з досліджуваної проблеми	18.04.24-18.04.24	
4	Аналіз технічних засобів	19.04.24-22.04.24	
5	Моделювання структири застосунку	22.04.24-24.04.24	
6	Програмна реалізація	25.04.24-23.05.24	
7	Оформлення пояснювальної записки	24.05.24-26.05.24	
8	Перевірка на плагіат	27.05.24	
9	Рецензування	28.05.24	
10	Підготовка презентації та доповіді	29.05.24-02.06.24	
11	Занесення роботи в електронний архів	03.06.24	
12	Попередній захист кваліфікаційної роботи	03.06.24	

Дата видачі завдання 8 квітня 2024 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Творошенко І.С.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 88 с., 4 табл., 35 рис., 2 дод., 30 джерел.

РОЗПІЗНАВАННЯ ОБЛИЧЧЯ, ІДЕНТИФІКАЦІЯ КОРИСТУВАЧА, ВЕБЗАСТОСУНОК, БАНКІВСЬКА СИСТЕМА, АВТОРИЗАЦІЯ, РЕЄСТРАЦІЯ, ШВИДКА АВТОРИЗАЦІЯ.

Об’єктом роботи є розроблення вебзастосунку розпізнавання та ідентифікації користувачів банківської системи.

Метою роботи є розробка вебзастосунку банківської системи на основі механізмів швидкої авторизації користувачів.

При створенні структури застосунку використано методи системного аналізу. Розглянуто наявні інструменти для розпізнавання та ідентифікації обличчя. Виявлено особливості методів Eigenfaces, FaceNet та OpenFace з подальшим застосування та тестуванням щодо поставленої задачі.

У результаті роботи розроблено вебзастосунок банківської системи. Реалізовано функції реєстрації, авторизації, розпізнавання та ідентифікації обличчя.

FACE RECOGNITION, USER IDENTIFICATION, WEB APPLICATION, BANKING SYSTEM, AUTHORIZATION, REGISTRATION, FAST AUTHORIZATION.

The object of the work is to develop a web application for recognizing and identifying users of the banking system.

The purpose of the work is to develop a web application of the banking system based on mechanisms for fast user authorization.

Methods of system analysis were used to create the application structure. Existing tools for face recognition and identification were considered. The features of the Eigenfaces, FaceNet and OpenFace methods are identified with further application and testing in relation to the task.

As a result, the banking system web application was designed and developed. The functions of registration, authorization, face recognition and identification were implemented.

ЗМІСТ

Перелік умовних позначень, символів, одиниць, скорочень і термінів 7
Вступ 8
1 Аналіз існуючих вебзастосунків розпізнавання та ідентифікації користувачів інформаційних систем в україні та за кордоном 9
1.1 Сучасний стан розвитку вебзастосунків розпізнавання та ідентифікації користувачів інформаційних систем в Україні та за кордоном 9
1.1.1 Вебзастосунки з функцією розпізнавання обличчя 10
1.1.2 Вебзастосунки з розпізнаванням та ідентифікацією обличчя 15
1.1.3 API-застосунки 19
1.2 Аналіз літературних джерел щодо існуючих підходів моделювання та реалізації вебзастосунків розпізнавання та ідентифікації користувачів автоматизованих систем 22
1.3 Постановка задачі 25
2 Моделювання структури вебзастосунку розпізнавання та ідентифікації користувачів банківської системи 26
2.1 Особливості архітектури даних вебзастосунку банківської системи 26
2.2 Особливості бази даних вебзастосунку банківської системи 27
3 Розроблення вебзастосунку розпізнавання та ідентифікації користувачів банківської системи 36
3.1 Вибір інструментальних засобів для реалізації поставленої задачі 36
3.2 Етапи розроблення вебзастосунку розпізнавання та ідентифікації користувачів банківської системи 37
3.2.1 Етап планування 38
3.2.2 Розроблення Java складової 38

3.2.3 Розроблення Python складової 52
3.2.4 Ручне тестування розробленого застосунку та аналіз результатів 60
3.2.5 Тестування методів оцінки схожості зображень та аналіз результатів 69
3.3 Перспективи подальшої роботи 71
Висновки 73
Перелік джерел посилання 74
Додаток А Тестові зображення 78
Додаток Б Реалізація перевірки відкриття сторінки 79

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Touch ID – розпізнавання за відбитком пальця

Face ID – розпізнавання за допомогою обличчя

Фреймворк – програмна інфраструктура (бібліотека) для полегшення розробки

Вебзастосунок – застосунок, запущений у браузері (клієнт), а сервером є вебсервер

ШІ – штучний інтелект

Embedding – вектор дробових чисел отриманих від аналізу зображення, найважливіші ознаки зображення

Eigenfaces – алгоритм розпізнавання обличчя

FaceNet – алгоритм розпізнавання обличчя

OpenFace – алгоритм розпізнавання обличчя

OpenCV – Open Computer Vision Library (бібліотека для роботи з обличчям)

LSTM – Long Short-Term Memory (довга короткочасна пам’ять, вид архітектури нейронної мережі)

CNN – Convolutional Neural Network (алгоритм розпізнавання обличчя)

AUC – Area Under the Curve (площа, обмежена кривою та віссю абсцис)

URL-адреса – Uniform Resource Locator (адреса сайту у мережі Інтернет)

LWF – Learning Without Forgetting (набір даних Labeled Faces in the Wild)

ВСТУП

У наш час велика кількість послуг, які раніше можна було отримати тільки прийшовши особисто, існує у мережі Інтернет. Наразі в інтернеті можна купляти речі, оформлювати документи. Так само у людей не завжди є можливість прийти безпосередньо у банк і отримати певні послуги. Тому інтернет-банкінг став достатньо поширеним явищем і деякі головні банки країни надають можливість встановити застосунок або скористатись ним онлайн.

З розвитком вебзастосунків, розвиваються і способи нелегального доступу до інформації та ресурсів користувачів. Використання звичайних способів захисту (паролей) не завжди допомагають захистити особисті дані.

Людина має певні фізичні особливості, які є унікальними: риси обличчя, голос, відбитки пальців. Протягом останніх років біометричні дані та їхнє розпізнавання все частіше використовуватися у захисті даних. Наприклад, використання біометрії можна побачити в інтернеті. Такі дані складніше, а іноді навіть майже неможливо підробити, що надає більшу надійність системі.

Саме тому, на даний момент, розробка вебзастосунку банківської системи, який би вмів зберігати обличчя при реєстрації, а також розпізнавати та ідентифікувати його при авторизації є актуальним. Така програма могла б допомогти розширити використання біометричних даних у сучасних системах, а також надати додаткову зручність та безпеку користувачам, шляхом пришвидшення процесу авторизації.

1 АНАЛІЗ ІСНУЮЧИХ ВЕБЗАСТОСУНКІВ РОЗПІЗНАВАННЯ ТА ІДЕНТИФІКАЦІЇ КОРИСТУВАЧІВ ІНФОРМАЦІЙНИХ СИСТЕМ В УКРАЇНІ ТА ЗА КОРДОНОМ

1.1 Сучасний стан розвитку вебзастосунків розпізнавання та ідентифікації користувачів інформаційних систем в Україні та за кордоном

Людина у своєму житті прагне до спрощення та надійності. Замість того, щоб рахувати вручну, вона придумала машини, які це роблять за неї. Замість того, щоб міркувати, яке слово і як правильно написати наступним у реченні, з’явилась технологія, яка пропонує варіанти, виходячи з того, що було написано раніше. Коли постала потреба у отриманні та аналізі біометричних даних, фахівці вирішили втілити технології розпізнавання та ідентифікації особистості.

Кожна людина є унікальною особистістю, але біологічних рис, за якими можна точно відрізнити одну людину від іншої, не так і багато. Наразі існує 2 основних способи ідентифікації: Touch ID та Face ID.

Touch ID хоча і набув розповсюдження у телефонах та у більшості ноутбуків, але він не отримав великого розповсюдження у вебзастосунках. Він використовується замість паролів при вході в систему та підтвердженні певних дій (часто може бути замінений Face ID або бути в комбінації). Для отримання відбитку пальця використовується сканер, який є часто або окремим пристроєм, або є вмонтованим в інший пристрій, на якому відкривається вебзастосунок, наприклад, телефон або ноутбук. Оскільки біометричні дані зберігає безпосередньо пристрій, то, якщо вебзастосунок допускає використання Touch ID, тоді для отримання даних використовується застосунок-посередник, який на пристрої порівнює відбитки і надсилає браузеру результат.

На відміну від попереднього варіанту, взаємодія з вебкамерами є більш поширеною серед вебзастосунків: використовується для розпізнавання та

отримання інформації про людей на фото, для аналізу та обробки облич на відео тощо. Робота з камерою важлива для сервісів з онлайн-трансляціями та відеодзвінками, наприклад, Google Meet та Discord. Наразі більша частина пристроїв має вебкамери і за потреби може використовувати розпізнавання обличчя, наприклад, для Face ID.

Відбиток пальця людини вже є унікальним і неповторним. Саме тому взаємодія з Touch ID полягає у простому порівнянні без необхідності у глибокому аналізі. Розпізнавання обличчя та ідентифікація за допомогою Face ID має похибку через те, що риси обличчя людей є схожими, а інколи ідентичними. Через доступність взаємодії з Face ID, а також необхідність ґрунтовного дослідження застосунків з цією функцією, сфокусуємо увагу на тих вебзастосунках, що мають можливість розпізнавання обличчя.

Вебзастосунків, які надають можливість по ідентифікації обличчя отримувати певні права, не так багато. Для того, щоб зрозуміти їх проблеми, достатньо розглянути інші програми, які теж використовують розпізнавання та ідентифікацію обличчя. Серед таких проєктів можна виділити три основних типи: ті які лише розпізнають обличчя, ті які розпізнають та ідентифікують обличчя та API-застосунки, які використовуються, як окремі програми, що надають функції розпізнавання та ідентифікації іншому застосунку, вони не є його внутрішньою системою. Українські проєкти з можливістю розпізнавання та ідентифікації користувачів є на деяких мобільних платформах (Monobank, Дія), але вони або не мають desktop версії, або ці функції у ній недоступні.

1.1.1 Вебзастосунки з функцією розпізнавання обличчя

Особливість застосунків з функцією розпізнавання обличчя у тому, що вони лише розпізнають обличчя, його риси для покращення та додавання ефектів.

На рисунках 1.1 та 1.2 показана робота вебверсії застосунку Zoom. У ньому розпізнавання обличчя використовується для розмиття чи заміни фону.

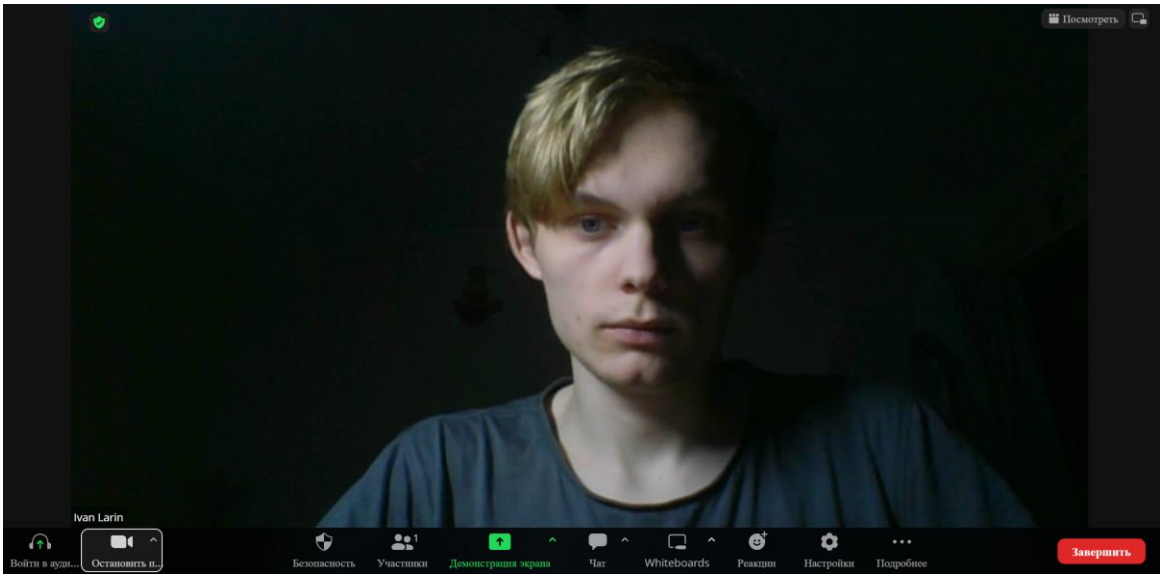


Рисунок 1.1 – Зображення в Zoom з вимкненим розпізнаванням обличчя

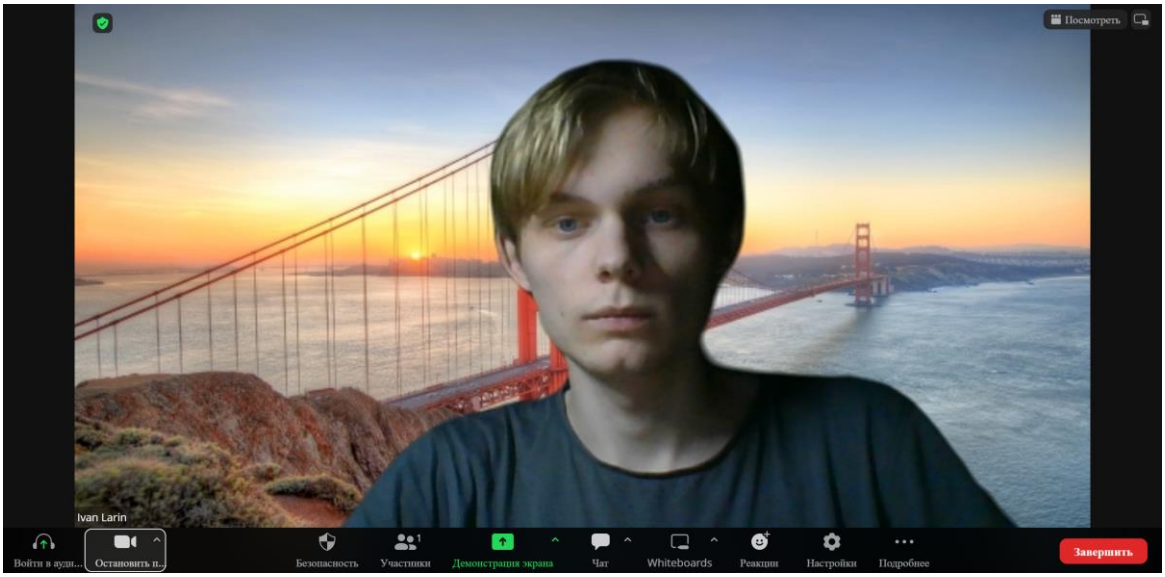


Рисунок 1.2 – Зображення в Zoom розпізнає людину і додає фон на решту зображення

Переваги застосунку Zoom:

– застосунок розпізнає риси обличчя і розмиває силует по краях для запобігання різких переходів до фону, а також графічних помилок;

– застосунок здатний підтримувати розпізнавання обличчя більше, ніж одного користувача, без проблем з продуктивністю.

Недоліки застосунку Zoom:

– при наявності поганого освітлення або різких рухах може бути некоректно розпізнана частина зображення і замінена фоном (рис. 1.3);

– функція налаштовується просто, але має дуже обмежений функціонал (додавання фонів);

– на більш старих пристроях робота з розпізнаванням обличчя може викликати затримки у системі через збільшений обсяг вживаємих ресурсів.

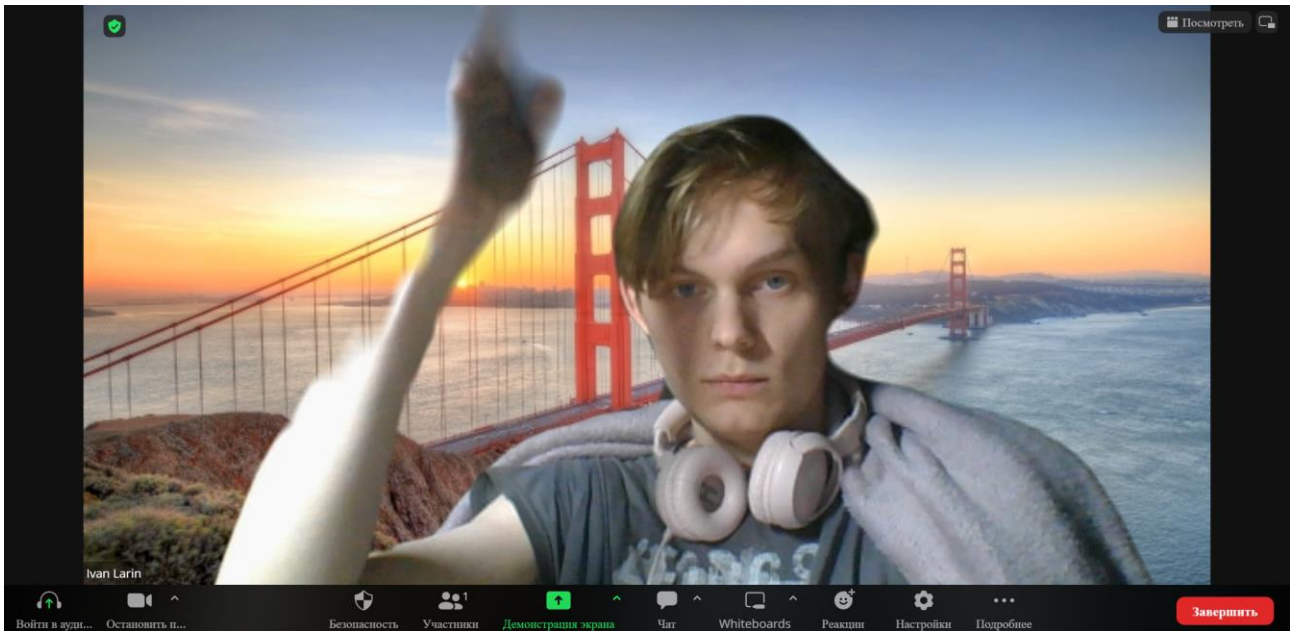


Рисунок 1.3 – Відсутня частина руки на зображенні

Google Meet – сервіс відеотелефонного зв’язку, розроблений компанією Google (рис. 1.4).

Переваги Google Meet:

– широкий функціонал налаштувань: можна змінити фон, можна додати ефекти на зображення (рис. 1.5);

– захист: застосунок надає можливість надавати або відмовляти у доступі, в залежності від того, чи розпізнано обличчя користувача;

– використання функцій розпізнавання обличчя істотно не збільшує навантаження на систему.

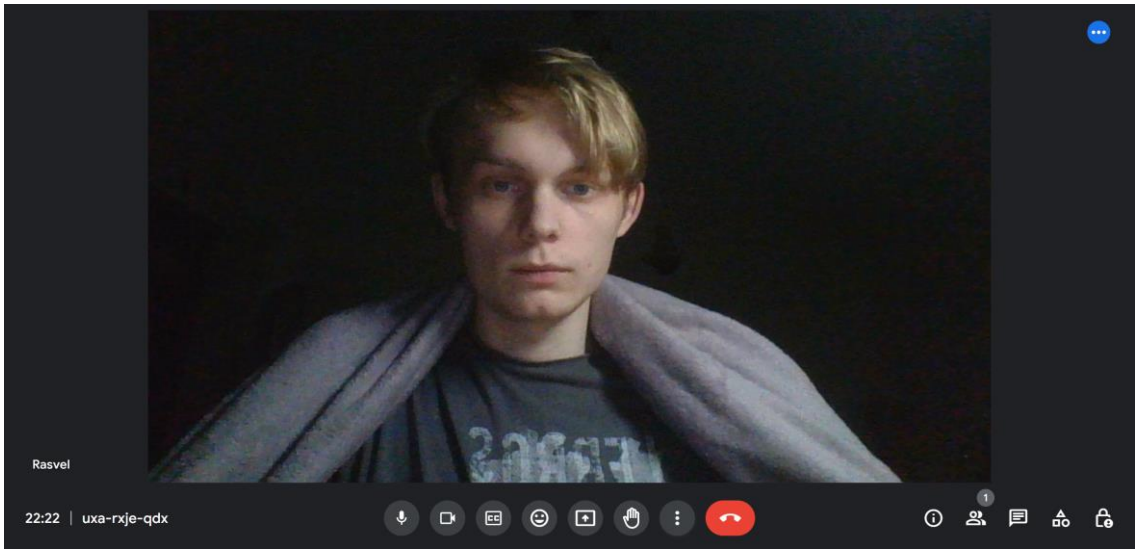


Рисунок 1.4 – Оригінальне зображення у Google Meet

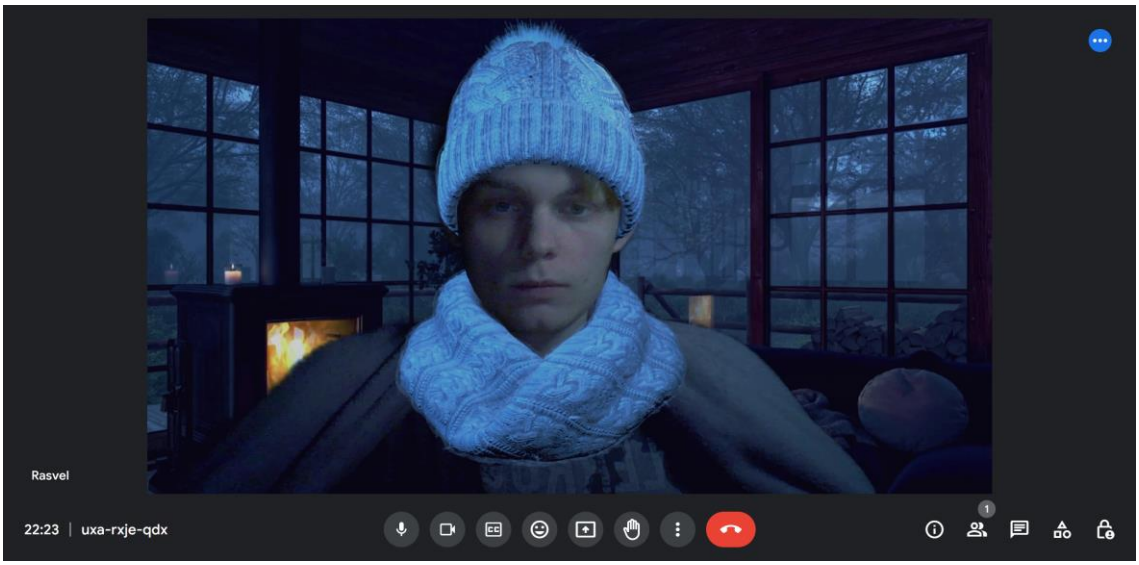


Рисунок 1.5 – Зображення у Google Meet з використанням фільтрів (фон, маска, колір)

Недоліки Google Meet:

– можливі проблеми з розпізнаванням обличчя через проблеми з камерою або хвороби;

– певні функції є доступними лише при наявності платної версії;

– методи розпізнавання не завжди можуть відпрацьовувати коректно і відмовляти у доступі особам, яким він має бути наданий;

– гірша якість зображення, що передається, це призводить до некоректного розпізнавання;

– використання функцій розпізнавання обличчя може впливати на якість зв'язку через збільшення навантаження на інтернет.

Microsoft Teams – це застосунок із пакету Office 365 від Microsoft (рис. 1.6, 1.7).

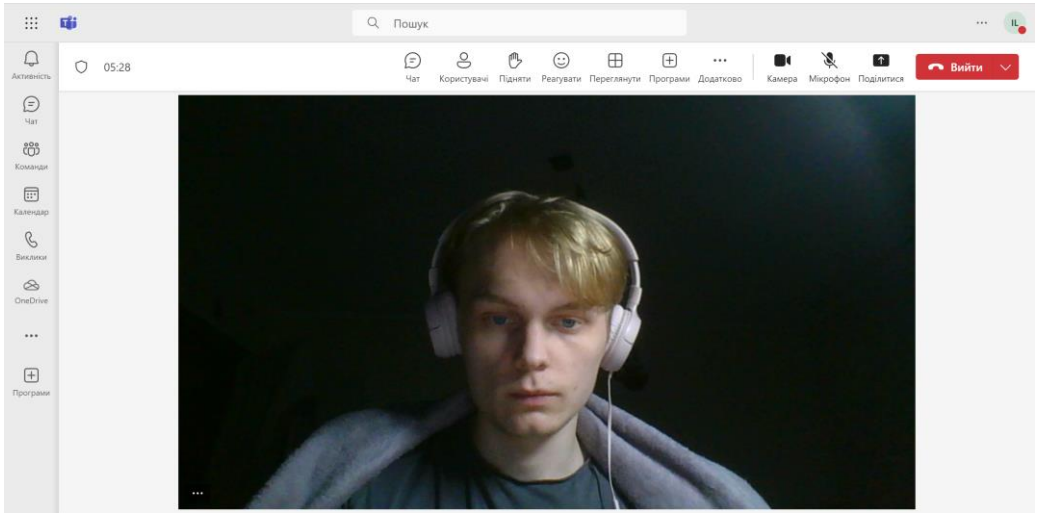


Рисунок 1.6 – Оригінальне зображення у Microsoft Teams

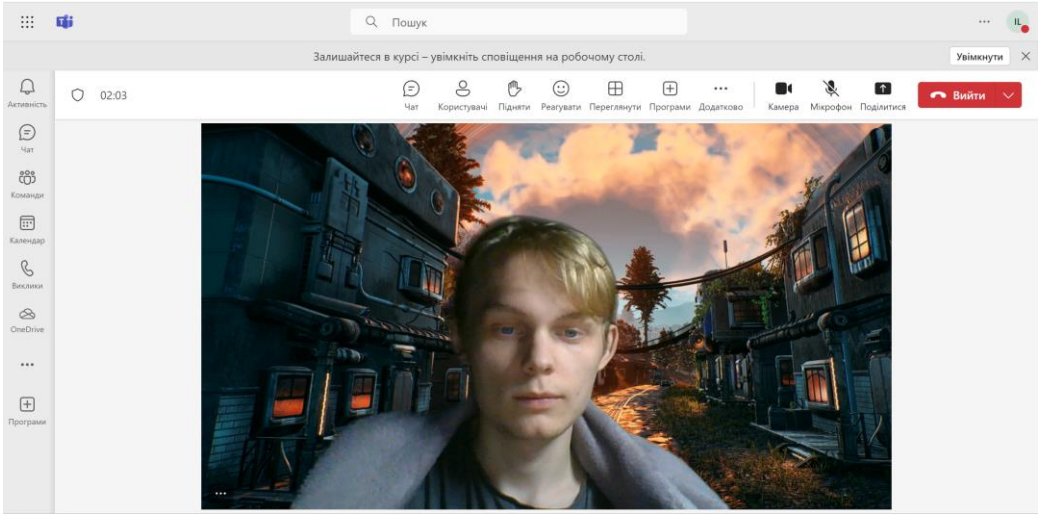


Рисунок 1.7 – Зображення у Microsoft Teams з використанням фільтрів (фон, маска, колір)

Microsoft Teams має багато можливостей для обміну даними та виконання дзвінків.

Переваги Microsoft Teams:

– можливе створення профілю для розпізнавання за голосом та обличчям;

– можливий логін за допомогою розпізнавання обличчя при наявності профілю розпізнавання;

– можна ввімкнути або увімкнути надання доступу до конференції за допомогою профілів розпізнавання.

Недоліки Microsoft Teams:

– зберігання біометричної інформації може призвести до її несанкціонованого використання;

– особа не завжди може бути ідентифікована через некоректне зчитування біометричних даних;

– відносно велика вартість для отримання усіх функцій.

Усі зазначені застосунки мають спільну проблему у вигляді некоректної роботи у випадку поганої камери або освітлення, а також відсутності зворотного зв’язку щодо зауважень про виявлені проблеми. Користувач не завжди може інтуітивно зрозуміти, як йому виправити проблему або у чому вона полягає.

1.1.2 Вебзастосунки з розпізнаванням та ідентифікацією обличчя

Наступний тип вебзастосунків – програми з функціями розпізнавання та ідентифікації обличчя. Основна відмінність цього типу застосунків, у тому, що вони не просто виявляють обличчя людини чи людей на фото і на основі цієї інформації накладають фільтри, а вони аналізують та порівнюють отримані дані з інформацією на інших фото.

Таким чином програми мають змогу групувати зображення або видавати інформацію, яка саме людина знаходиться на зображенні.

Одним з прикладів використання таких функцій є групування фото по особах у Google Photo (рис. 1.8).

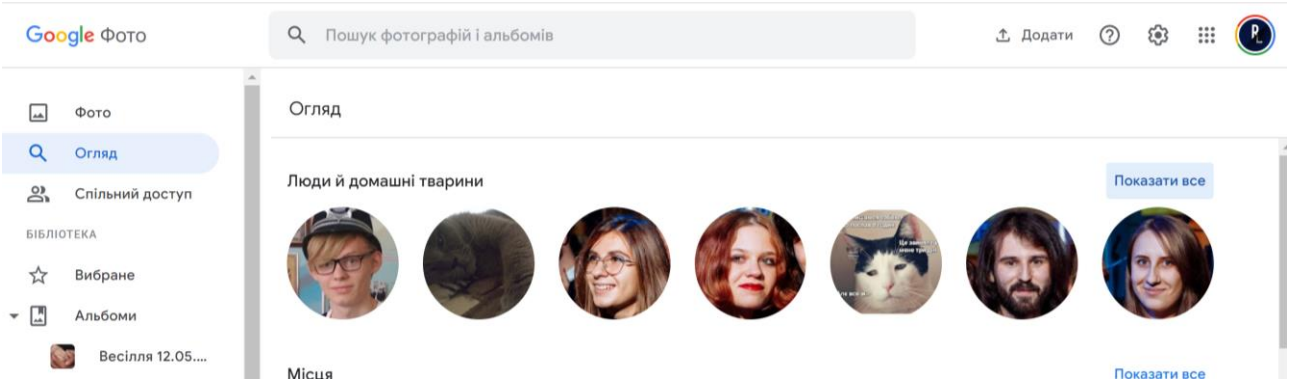


Рисунок 1.8 – Використання розпізнавання облич для групування фото

Застосунок аналізує фотографії і розпізнає на них обличчя. Якщо це обличчя повторюється, воно формує для нього папку.

Переваги Google Photo:

– якщо на фото є декілька людей для якиз існують групи, то фото попаде в обидві;

– може ідентифікувати обличчя на відео і фото;

– розпізнає людей на зображенні з поганою освітленістю;

– відрізняє людей від тварин;

– люди, зображення яких трапляється менше двох разів опускаються.

Недоліки Google Photo:

– якщо людина А є на фото А з освітленням, яке є на фото Б де людини А немає, фото Б може потрапити у папку для людини А;

– якщо у людини є незначні відмінності в зовнішності, може формуватися декілька папок;

– якість групування залежить від кількості фото конкретної особи;

– методи можуть помилятися і робити категорії навіть для людей, де обличчя зустрічається один раз.

В інтернеті є застосунки, які дозволяють по фото отримати інформацію про те, на яку відому особистість схожа персона на фото. Як приклад, можна навести сайт starbyface.com, який можна знайти одним з перших при пошуку схожих застосунків.

Переваги starbyface.com:

– велика база матеріалів для навчання методу;

– метод може виявляти окремі частини обличчя (нос, брови, тощо) та аналізувати їх, а також показувати на зображення область аналізу;

– метод може аналізувати схожість з особами різної статі та групувати їх;

– застосунок здатний розпізнавати зображення з поганим освітленням;

– застосунок здатний розпізнавати та порівнювати фото особи з різних ракурсів і приходити до схожих результатів (рис. 1.9 а)).

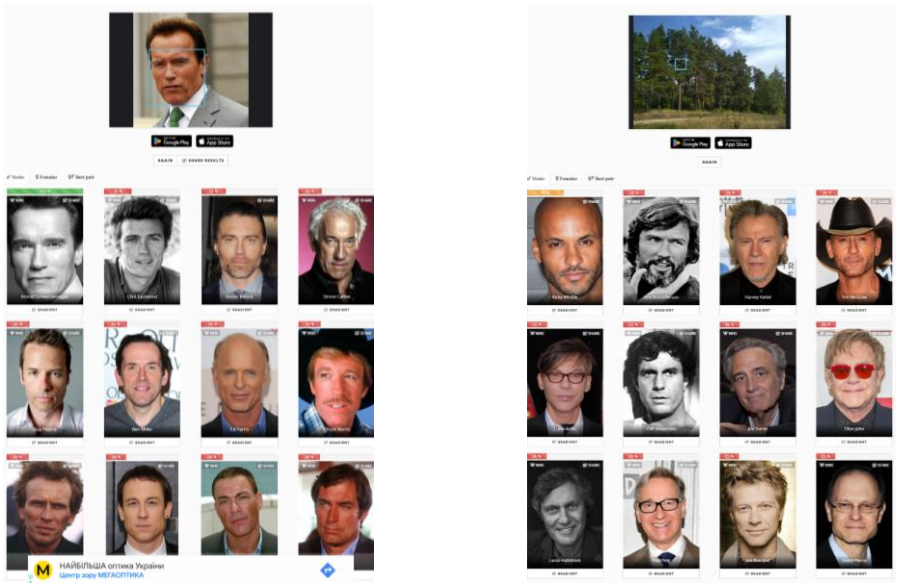
Недоліки starbyface.com:

– застосунок не завжди розпізнає відсутність людей на фото і знаходить риси обличчя людини у об’єктах (рис. 1.9 б));

– якість порівняння залежить безпосередньо від даних,які були задані для навчання застосунку на початку;

– застосунок не завжди коректно розпізнає стать особи на фото і може ідентифікувати чоловіка, як жінку.

Іншим застосунком, на який можна звернути увагу, є pinkmirror.com, який дозволяє проаналізувати красу обличчя, а також багато інших параметрів: стан шкіри, стать, довжину носа, тощо. Приклад аналізу показано у додатку А.



а) б)

Рисунок 1.9 – Результати аналізу зображень:
а) з наявним обличчям на зображенні; б) з відсутнім обличчям на зображенні

Переваги pinkmirror.com:

– застосування різних методів, які можуть дати різну інформацію по різних категоріях;

– аналіз супроводжений повноцінними текстовими коментарями;

– риси, що аналізуються показуються на зображенні;

– навність коментарів, якщо фото не підходить під запитувані застосунком параметри.

Недоліки pinkmirror.com:

– більша кількість функцій доступна лише в платній версії;

– більш коректний аналіз можливий при наявності 10 і більше проаналізованих облич;

– складність у підборі фото.

Варто зауважити, що в цьому напрямку також існують розробки, у яких брали участь українці. У цьому ключі, необхідно згадати про схожу розробку словацької компанії SYTOSS Ltd., команда якої складається переважно з українців. Система знаходиться у обмеженому доступі і активній розробці.

1.1.3 API-застосунки

Останній тип програм, якому необхідно приділити увагу, це API-застосунки. Це тип, який зазвичай неможливо протестувати, зайшовши на сайт, його функція полягає у тому, що застосунок приймає вхідні дані і видає відповідь. Таким чином, підключивши API до системи, розробник зможе не розробляти сервіс для розпізнавання та верифікації обличчя, а вже мати ці функції у своїй системі. Серед таких проєктів варто виділити Amazon Rekognition.

Переваги Amazon Rekognition:

– точність розпізнавання;

– можливість розширення системи. застосунок вміє працювати з різними розмірами даних;

– наявність документації, що спрощує розробку;

– різноманітність функціоналу. застосунок підтримує розпізнавання обличчя, порівняння облич, розпізнавання емоцій, тощо.

Недоліки Amazon Rekognition:

– відносно велика ціна та модель оплати pay-as-you-go, коли вона залежить від кількості використаних ресурсів;

– проблема приватності, оскільки накопичення даних відбувається саме в межах застосунку;

– складність взаємодії з іншими застосунками, оскільки функціонал системи спеціалізований на роботі з іншими мікросервісами від Amazon.

Іншим прикладом варто розглянути застосунок від компанії Luxand.

Переваги Luxand:

– як і рішення від Amazon, пропонує різний функціонал;

– пропонує методики, які більше орієнтовані на ідентифікацію обличчя і позиціонується як застосунок, для роботи з системи, де необхідна надійність і захист;

– рішення спроєктовано, щоб опрацьовувати результати швидко, що підходить для обробки відео або систем, де потрібна мінімальна затримка у відповіді застосунку;

– дозволяє роботу з різними мовами та фреймворками.

Недоліки Luxand:

– відносно висока ціна і малий спектр можливостей безкоштовної версії;

– неможливість адаптації під власні потреби. застосунок може програвати певним методам у надійності та швидкості, і не дозволяти покращити рішення;

– як і в минулому застосунку, залишається проблема приватності через накопичення застосунків біометричних даних.

Проаналізувавши застосунки, виявлено необхідність вдосконалювати сучасні вебзастосунки стосовно розпізнавання та ідентифікації обличчя. Тему розроблення вебзастосунку розпізнавання та ідентифікації користувачів банківської системи обрано для розробки вебзастосунку з обома функціями, який буде доступний в форматі API, із застосуванням новітніх бібліотек, підходів та фреймворків, враховуючи переваги та недоліки аналогів.

Альтернативним рішенням окремо хочеться розглянути такий застосунок як OpenCV, який є безкоштовним рішенням.

Переваги OpenCV:

– Open Source рішення, яке дає змогу редагувати вихідний код під власні потреби, а також відсутність плати за використання;

– використання різних спеціалізованих стандартних методів, яке дозволяє використовувати застосунок для різних потреб;

– підтримка різних мов програмування, а також кросплатформенність;

– наявність літератури що спрощує процес вивчення та використання застосунку.

Недоліки OpenCV:

– наявність широкої функціональності може затрудняти старт для новачків;

– певні методи можуть показувати гіршу продуктивність, ніж альтернативні застосунки;

– запропоновані методи в основному є низькорівневими, що може змусити розробника писати велику кількість власного коду;

– нові версії застосунку можуть вносити істотні зміни, які можуть призвести до несумісності з іншими бібліотеками.

Таким чином, для вдосконалення вебзастосунків необхідно реалізувати наступні функції:

– реалізація графічного інтерфейсу. Необхідно розробити інтуітивно-зрозумілий інтефейс, який буде давати можливість користувачеві зробити фотографію з необхідними системі параметрами (розмір, направлення погляду, тощо);

– вирішення проблеми приватності. Необхідно реалізувати захист накоплених біометричних даних за допомогою найсучасніших рішень шифрування для попередження несанкціонованого доступу;

– розробка порад при недотриманні вимог системи до зображень. Якщо в випадку користувача вимоги до зображення недотримані, то необхідно показувати повідомлення стосовно проблеми (наприклад, «заяскраве освітлення, змініть яскравість у приміщенні»);

– вирішення проблеми захисту. За допомогою підбору даних для навчання системи та комбінування методів розпізнавання обличчя досягти найвищого можливого рівня точності з попередженням різноманітних способів омани системи. Наприклад, через використання спуфінгу.

Отже, удосконалення існуючих та розроблення принципово нових застосунків для розпізнавання та ідентифікації об’єктів на зображеннях є актуальним вирішенням сучасних прикладних задач.

1.2 Аналіз літературних джерел щодо існуючих підходів моделювання та реалізації вебзастосунків розпізнавання та ідентифікації користувачів автоматизованих систем

У роботі [1] автори описують проблему безпеки в Інтернеті, зосереджуючись на фішингу, як одному з основних видів атак. Фішинг є формою шахрайства, при якому зловмисники, відомі як фішери, намагаються викрасти особисту інформацію користувачів, таку як дані для входу на вебсайти, паролі, номери кредитних карток та іншу конфіденційну інформацію. Це відбувається через імітацію довірених вебсайтів, щоб ввести користувачів в оману та ввели свої особисті дані на фішинговому сайті. Автори статті підкреслюють важливість боротьби з фішингом та необхідність розвитку систем виявлення фішингових вебсайтів для захисту користувачів Інтернету від можливої небезпеки. Це є актуальною проблемою, оскільки в Інтернеті постійно з’являються нові способи атак і шахрайства, тому користувачі повинні бути захищені від них. У статті пропонуються три різні методи виявлення фішингових вебсайтів на основі глибокого навчання: довга короткочасна пам’ять (LSTM), згорткова нейронна мережа (CNN) та комбінований підхід LSTM-CNN. Ці методи базуються на аналізі текстових та структурних характеристик вебсторінок, що дозволяє виявити фішингові сайти з високою точністю. Експериментальні результати демонструють, що запропоновані методи мають високу точність виявлення фішингу: 99,2%, 97,6% та 96,8% для CNN, LSTM-CNN та LSTM відповідно. Це свідчить про ефективність запропонованих методів у виявленні фішингових атак. Отримані ймовірності можна вважати надійними оскільки:

– обсяг даних для тренування є достатньо об’ємним: набір містить ознаки URL-адрес (20000 записів по 80 ознак);

– автори зазначають, що використовували відповідні та узгоджені дані для навчання системи, що сприяє достовірності результатів;

– зазначено, що використано метод SelectKBest для відбору найкращих ознак та метод MinMaxScaler для масштабування даних. Дані кроки попередньої обробки можуть покращити якість навчання моделі.

У дослідженні [2] авторами розглядається проблема банківського шахрайства та застосування штучного інтелекту для її виявлення. За останні роки через пандемію COVID-19 відбувся масовий перехід багатьох банківських операцій на онлайн-платформи, що сприяло зростанню обсягів банківського шахрайства через створення багатьох благодійних фондів, які можуть використовуватися зловмисниками для обману користувачів. Робота зосереджується на використанні методів машинного навчання для аналізу та виявлення шахрайських банківських операцій в Інтернеті. Основна наукова новизна дослідження полягає в розробці моделей машинного навчання для ефективного виявлення шахрайських банківських операцій та використання методів попередньої обробки банківських даних для подальшого аналізу та вибору оптимальних результатів. Робота також детально розглядає різні підходи для підвищення точності знаходження шахрайських операцій, зокрема, обробку незбалансованих даних, трансформацію та інженерію ознак. Запропонована модель, яка ґрунтується на штучних нейронних мережах, демонструє високу ефективність у виявленні шахрайських транзакцій. Результати різних методів представлено візуально, при цьому логістична регресія показала найкращі результати з AUC приблизно 0,946. Стекове узагальнення показало ще кращі показники з AUC – 0,954.

У роботі [3] досліджується вплив штучного інтелекту на кібербезпеку банків у Катарі, враховуючи загрозу кібератак та несанкціонованого доступу в банківській сфері. Банківська галузь у Катарі визнає необхідність застосування ШІ для створення ефективної системи кіберзахисту, спрямованої на мінімізацію ризиків кібернападів. Дослідження базується на тематичному аналізі інтерв’ю з 9 експертами банківської галузі Катару. Під час дослідження вдалося виявити чотири ключові теми:

– ШІ є основним інструментом підвищення кібербезпеки банків в Катарі, що підкреслює значення використання інноваційних технологій для захисту від кіберзагроз;

– банки стикаються з проблемами у використанні ШІ для підвищення кібербезпеки, що може включати в себе складнощі в інтеграції нових систем, недостатню експертизу та нестачу кваліфікованих кадрів;

– ШІ може бути використаний деструктивно і становити загрозу кібербезпеці банків у Катарі, якщо не вживатимуться належні заходи безпеки;

– використовувані інструменти на основі ШІ можуть мати вразливості, що можуть бути використані зловмисниками, що вимагає постійного вдосконалення систем кіберзахисту.

У роботі [4] автори роблять висновок, що питання безпеки залишається одним із найважливіших у банківській сфері. Однією із основних загроз є крадіжка особистих даних, що може відбуватися через незахищеність пристроїв, які легко піддаються кібератакам. Для зменшення ризику крадіжок особистих даних та підвищення рівня безпеки, автори зазначають важливість біометрії. Біометричні дані, такі як відбитки пальців, райдужна оболонка ока, розпізнавання обличчя тощо, є унікальними для кожної людини, тому їх важко підробити. У статті проводиться аналіз можливих біометричних рішень для проблем безпеки в банківських системах, а також реалізується модель ідентифікації та аутентифікації на основі відбитків пальців. Результати даної роботи можуть слугувати прикладами при розробці системи з ідентифікацією за обличчям.

У роботі [5] автори концентруються на аналізі та розробці моделі, яка використовує Face mesh для виявлення та розпізнавання облич. Face mesh – це технологія, яка дозволяє точно визначати контури обличчя на зображеннях, незалежно від умов освітлення та фону. Даний фактор дозволяє моделі працювати ефективно в різних умовах і розпізнавати обличчя чоловіків і жінок різного віку та рас. Для навчання моделі використано зображення із набору даних Labelled Faces in the Wild (LWF), а також

зображення, зняті в реальному часі. Під час тестування модель порівнює контури обличчя на тестовому зображенні із контурами обличчя на навчальних зображеннях. Якщо вони збігаються, модель розпізнає ім’я людини, в іншому випадку вона видає результат як «невідомий». Вказана модель досягає вражаючої точності у розпізнаванні облич – 94,23%, що свідчить про ефективність використання технологій глибокого навчання та обробки зображень для автоматичного розпізнавання облич.

Таким чином, проаналізувавши праці [1 – 20], можна зробити висновок, що тема розпізнавання та ідентифікації користувача за обличчям потребує подальшого дослідження. Крім того, встановлено, що не існує у відкритому доступі вебзастосунку банківської системи, яка використовує механізми розпізнавання та ідентифікації користувачів за обличчям.

1.3 Постановка задачі

Таким чином, розробка вебзастосунку банківської системи з можливістю розпізнавання та ідентифікації користувача за обличчям є актуальною темою.

Об’єктом роботи є розроблення вебзастосунку розпізнавання та ідентифікації користувачів банківської системи.

Метою роботи є розробка вебзастосунку банківської системи на основі механізмів швидкої авторизації користувачів.

Для досягнення поставленої мети необхідно:

– проаналізувати наявні вебзастосунки, що використовують розпізнавання та ідентифікацію обличчя, зрозуміти їхні переваги та недоліки;

– проаналізувати наявні методи аналізу та ідентифікації обличчя, порівняти, та визначити чи є необхідним використання одного або декількох методів;

– розробити вебзастосунок, який сформує зображення користувача при реєстрації та використає обраний метод ідентифікації під час авторизації.

2 МОДЕЛЮВАННЯ СТРУКТУРИ ВЕБЗАСТОСУНКУ РОЗПІЗНАВАННЯ ТА ІДЕНТИФІКАЦІЇ КОРИСТУВАЧІВ БАНКІВСЬКОЇ СИСТЕМИ

2.1 Особливості архітектури даних вебзастосунку банківської системи

Для реалізаці вебзастосунку банківської системи було використано мікросервісний підхід у проєктуванні архітектури.

Мікросервісна архітектура – архітектурний стиль, який передбачає використання декількох сервісів, кожен з яких працює в межах власного процесу і спілкується з іншими за допомогою протоколів передачі даних, наприклад http.

Переваги мікросервісів:

– кожен з сервісів розгортається та запускається окремо, що пришвидшує час запуску системи загалом;

– кожен з сервісів масштабується окремо;

– при змінах в одному сервісі не обов’язково перезапускати інші, можна перезапустити лише змінений мікросервіс;

– використання мікросервісного підходу дозволяє використовувати різні мови програмування;

– при наявності помилки в одному з мікросервісів, система загалом зможе продовжувати роботу. У випадку, якщо зламаний мікросервіс не мав логіки критичної для роботи системи.

Недоліки мікросервісів:

– проблема у розгортанні системи. Наявність великої кількості мікросервісів може створити проблему при запуску;

– проблема у знаходженні місця проблеми. При наявності мікросервісної архітектури, розробники можуть стикнутися з неможливістю одразу виявити, у якому місці система дала збій.

У випадку поточної реалізації вебзастосунку банківської системи архітектура передбачає наявність двох мікросервісів: «bank-microservice», «face-id-microservice». Архітектура застосунку показана на рисунку 2.1.

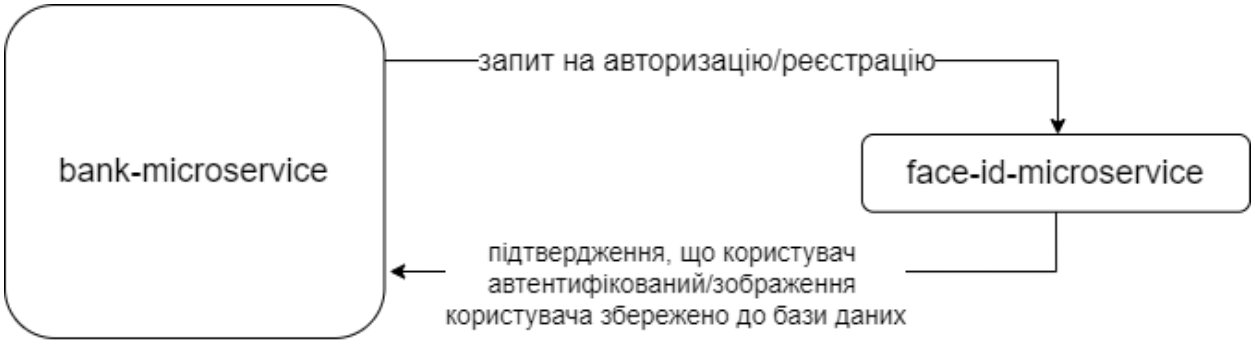


Рисунок 2.1 – Архітектура вебзастосунку банківської системи

«bank-microservice» відповідає за процеси реєстрації та авторизації, а також інші процеси у персональному кабінеті користувача, які не є частиною автенфікації.

«face-id-microservice» відповідає за роботу з Face ID під час авторизації чи реєстрації.

2.2 Особливості бази даних вебзастосунку банківської системи

Для збереження даних про користувача вебзастосунок має базу даних. База даних має вигляд зображений на рисунку 2.2.

Поточна база даних використовується для зберігання усіх даних про користувача і задовольняє наступні бізнес-вимоги:

– електрона пошта, номер паспорту, номер акаунта, ідентифікаційний номер, номер телефону мають бути унікальними в межах бази даних;

– номер картки, IBAN мають бути унікальними в межах бази даних;

– користувач може мати багато карток, картка має одного користувача;

– користувач може мати лише одне збережене зображення свого обличчя;

– користувач може використовувати лише один паролю для входу в систему.

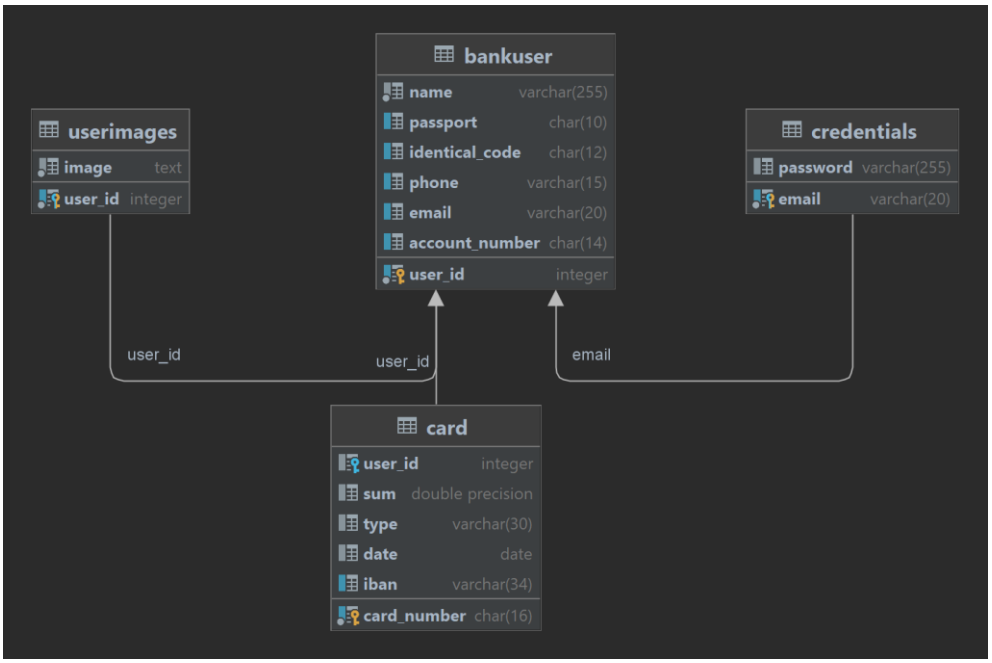


Рисунок 2.2 – Загальний вигляд бази даних банківської системи

Далі буде детально розглянуто кожну з таблиць 2.1 – 2.4.

Перша таблиця – bankuser (табл. 2.1).

Таблиця 2.1 – Таблиця bankuser вебзастосунку банківської системи

Data	Type	Example
user_id	Integer (counter)	1
name	String	Ivan Larin
passport	String	0000000000
identical_code	String	111111111111
phone	String	+380960000001
email	String	1@gmail.com
account_number	string	11111100000000

Таблиця bankuser зберігає основні дані про користувача. Усі дані цієї таблиці, окрім імені, мають бути унікальними.

Наступна таблиця – credentials (табл. 2.2).

Таблиця 2.2 – Таблиця credentials вебзастосунку банківської системи

Data	Type	Example
email	String	1
password	String	fFGQn5XB6biGo0h80+vtaQ==

Таблиця credentials зберігає пароль користувача. Пароль зберігається шифрованим, що ускладнює його крадіжку. Поле email є первинним ключем і посилається на email у таблиці bankuser.

Ще одна таблиця – userimages (табл. 2.3).

Таблиця 2.3 – Таблиця userimages вебзастосунку банківської системи

Data	Type	Example
user_id	Integer (counter)	1
image	bytea	/9j/4AgcJP2P/wCHcn7E/wD0QHTP/Ay6/wDjtf/Z…

Таблиця userimages зберігає дані про зображення користувача. Поле user_id є первинним ключем і посилається на user_id у таблиці bankuser.

Остання таблиця – card (табл. 2.4).

Таблиця card зберігає дані про картки користувача. Поле user_id посилається на user_id у таблиці bankuser, але не є первинним ключем і може повторюватись, що дає можливість користувачу мати декілька карток. Поля card_number, iban мають бути унікальними.

Таблиця 2.4 – Таблиця card вебзастосунку банківської системи

Data	Type	Example
user_id	Integer (counter)	1
card_number	String	4604066508035563
sum	Double	101.50
type	String	VISA
date	Date	2028-05-01
iban	String	UA211324561100000000000000004604

Лістинг 2.1 Код для створення бази даних:

```
CREATE TABLE BankUser
(
    user_id        SERIAL primary key,
    name           varchar(255) NOT NULL,
    passport       char(10) unique,
    identical_code char(12) ,
    phone          varchar(15) ,
    email          varchar(20) unique check (email LIKE '%@%mail.com'),
    account_number char(14)
);

CREATE TABLE Card
(
    card_number char(16) primary key,
    user_id     INT REFERENCES BankUser (user_id),
    sum         float check (sum >= 0),
    type        varchar(30),
    date        date,
```

```
    IBAN        varchar(34)
);

CREATE TABLE Credentials
(
    email    varchar(20) PRIMARY KEY REFERENCES BankUser (email),
    password varchar(255)
);

CREATE TABLE UserImages (
    user_id SERIAL PRIMARY KEY REFERENCES BankUser (user_id),
    image TEXT not null
)
```

Архітектура мікросервісу для аналізу обличчя дозволяє розширювати систему будь-якими методами, які можуть порівняти 2 обличчя.

У даній роботі були використані наступні методи:

– Eigenfaces;

– FaceNet;

– OpenFace.

Якщо для роботи методу Eigenfaces необхідне саме зображення, то для роботи OpenFace та FaceNet необхідні отримані з зображення дані (embedding). Embedding надають найважливішу інформацію про зображення, їх порівняння дає можливість робити висновок про схожість зображень.

Методи OpenFace та FaceNet схожі за своїм принципом, оскільки не мають відкритої поступової реалізації. Вони лише отримують вхідні дані і модель видає вихідний результат.

Лістинг 2.2 Програмна реалізація методу Eigenfaces:

```
import cv2
import numpy as np

def calculate_similarity(image1, image2):
    image1_flat = image1.flatten()
    image2_flat = image2.flatten()

    mean_image = (image1_flat + image2_flat) / 2

    image1_centered = image1_flat - mean_image
    image2_centered = image2_flat - mean_image

    covariance_matrix = np.cov([image1_centered, image2_centered])
    eigenvalues, eigenvectors = np.linalg.eig(covariance_matrix)

    idx = eigenvalues.argsort()[::-1]
    eigenvalues = eigenvalues[idx]
    eigenvectors = eigenvectors[:, idx]

    similarity = np.dot(image1_centered, image2_centered) / (
        np.linalg.norm(image1_centered) *
np.linalg.norm(image2_centered))

    similarity_percentage = (similarity + 1) * 50

    return similarity_percentage
```

Схема методу Eigenfaces показана на рисунку 2.3.

Рисунок 2.3 – Метод Eigenfaces

Якщо два зображення однакові та метод, що отримує embedding один, то і embedding для обох зображень будуть однакові.

Загальна схема методів OpenFace та FaceNet показана на рисунку 2.4.

Рисунок 2.4 – Загальна схема методів OpenFace та FaceNet

Програмна реалізація методів OpenFace та FaceNet показана далі.

Лістинг 2.3 Програмна реалізація методу OpenFace:

```
from deepface import DeepFace as df

import cv2
from deepface import DeepFace

def get_embeddings_from_image(image):
    embeddings = DeepFace.represent(image, model_name='OpenFace', enforce_detection=False)[0]['embedding']
    return embeddings

def run(image1, image2):
```

```
    embedding1 = get_embeddings_from_image(image1)
    embedding2 = get_embeddings_from_image(image2)
    similarity = df.verify(embedding1, embedding2, model_name='OpenFace')
    distance = similarity['distance']
    max_distance = 4
    similarity_percent = (1 - distance / max_distance) * 100
    result = max(0, similarity_percent)
    return result
```

Лістинг 2.4 Програмна реалізація методу FaceNet:

```
from deepface import DeepFace as df, DeepFace

def get_embeddings_from_image(image):
    embeddings = DeepFace.represent(image, model_name='Facenet', enforce_detection=False)[0]['embedding']
    return embeddings

def run(image1, image2):
    embedding1 = get_embeddings_from_image(image1)
    embedding2 = get_embeddings_from_image(image2)

    similarity = df.verify(embedding1, embedding2, model_name='Facenet')
    distance = similarity['distance']
    max_distance = 4  # Maximum distance in Facenet's embedding space
    similarity_percent = (1 - distance / max_distance) * 100  # Convert distance to percentage
    result = max(0, similarity_percent)
    return result
```

3 РОЗРОБЛЕННЯ ВЕБЗАСТОСУНКУ РОЗПІЗНАВАННЯ ТА ІДЕНТИФІКАЦІЇ КОРИСТУВАЧІВ БАНКІВСЬКОЇ СИСТЕМИ

3.1 Вибір інструментальних засобів для реалізації поставленої задачі

Одним з критично важливих етапів під час розробки застосунку є вибір технологій, які будуть використані у роботі. Після ретельного вивчення сучасних технологій та мов програмування прийнято рішення про використання Spring Boot для бекенд-розробки на Java та Flask для фронтенд-розробки. Використано IntelliJ IDEA та PyCharm відповідно.

Spring Boot, відомий фреймворк на основі Java, пропонує повний набір інструментів для полегшення розробки програмного забезпечення, включаючи тестування, налагодження та розгортання. У сукупності з Flask, який став фронтенд-рішенням, вони дали можливість використовувати дві різні мови програмування разом, що продемонстровано використанням мікросервісної архітектури.

Використання Spring Boot з інтеграцією Spring Security посилили механізми захисту застосунку. Spring Security, надійний захисник для Java-застосунків, забезпечив надійні функції автентифікації та авторизації, гарантуючи цілісність даних і конфіденційність користувачів. Безшовна інтеграція Spring Security з Spring Boot полегшила впровадження безпечних протоколів автентифікації та заходів контролю доступу, підвищивши стійкість вебзастосунків до потенційних загроз безпеці.

Для інтеграції функцій розпізнавання та ідентифікації обличчя використано бібліотеки OpenCV та DeepFace.

OpenCV – бібліотека комп’ютерного зору з відкритим вихідним кодом, забезпечила застосунок множиною інструментів для обробки зображень, що дозволяє вирішувати такі завдання, як виявлення об’єктів, розпізнавання облич і маніпулювання зображеннями. Її універсальний характер і детальна

документація спростили процес інтеграції, надавши програмі найсучасніші можливості візуального аналізу.

DeepFace – спеціалізована система для аналізу облич доповнила OpenCV, пропонуючи складні методи розпізнавання облич і моделі глибокого навчання. Використовуючи можливості розпізнавання обличчя, застосунок DeepFace зміг розпізнати риси обличчя з надзвичайною точністю, покращивши автентифікацію користувачів і персоналізацію користувацького досвіду.

PyCharm та IntelliJ IDEA – редактори з відкритим вихідним кодом на основі широкого спектру мов програмування і фреймворків стали ідеальними платформами для розробки інтерфейсу. Їх набори інструментів, включаючи інтегровані термінали, підтримку системи контролю версій та відладчики, полегшили процес розробки, забезпечивши безперешкодну інтеграцію та оптимізацію робочих процесів.

3.2 Етапи розроблення вебзастосунку розпізнавання та ідентифікації користувачів банківської системи

Розроблення вебзастосунку розпізнавання та ідентифікації користувачів банківської системи є комплексним і під час його реалізації реалізовано наступні етапи:

- етап планування;
- розроблення Java складової;
- розроблення Python складової.

3.2.1 Етап планування

На даному етапі проведено роботу з предметною областю банківської системи і сформовано подальший план розробки проєкту. Покроковий опис виконаної роботи наведено далі:

– аналіз предметної області та планування розробки: на даному етапі проведено аналіз предметної області «Банківська система», сформовано вимоги для застосунку, визначено, які дані необхідно заповнювати та зберігати для користувача. Сформовано план розроблення вебзастосунку;

– проєктування та створення бази даних. Спроєктовано структуру бази даних: визначені необхідні поля, типи даних, зв’язки між таблицями.

3.2.2 Розроблення Java складової

На даному етапі реалізовано мікросервіс на мові Java для роботи з базою даних, процесами реєстрації та авторизації, розроблено дизайн сторінок реєстрації та авторизації.

Опис виконаних кроків:

– ініціалізація Java проєкту: засобами IntelliJ IDEA створено новий проєкт Spring. Налаштування при ініціалізації показано на рисунку 3.1;

– підключення необхідних залежностей до проєкту. Шляхом оновлення файлу pom.xml до проєкту додано можливість працювати із Spring Security, Spring MVC, тощо. Усі необхідні залежностей надано у лістингу 3.1;

– ініціалізація бази даних. Лістингом 3.2 ініціалізовано Docker-контейнер, а за допомогою лістингу 2.1 створено базу даних. Також у базу даних додано 2 технічних таблиці необхідних для коректної взаємодії з Spring Security. Код для їхнього створення показано у лістингу 3.3;

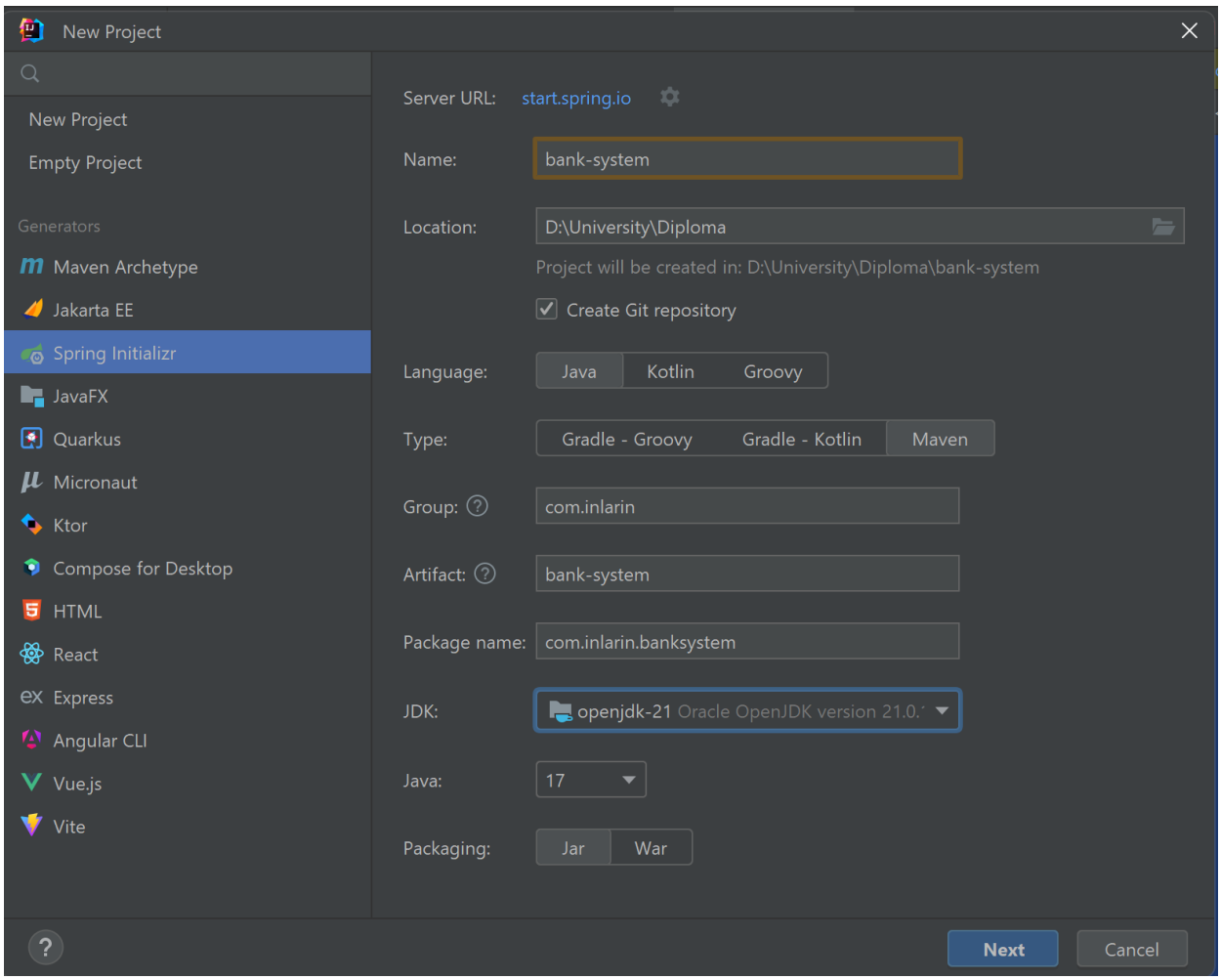


Рисунок 3.1 – Налаштування при ініціалізації нового Spring проєкту у середовищі IntelliJ IDEA

Лістинг 3.1 Перелік необхідних залежностей для Java проєкту:

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-thymeleaf</artifactId>
    </dependency>
```

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
    <groupId>org.thymeleaf.extras</groupId>
    <artifactId>thymeleaf-extras-springsecurity5</artifactId>
    <version>3.1.0.RELEASE</version>
</dependency>
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <version>1.18.30</version>
    <optional>true</optional>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-tomcat</artifactId>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
</dependency>
<dependency>
```

```
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-test</artifactId>
    <version>6.0.0</version>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>com.microsoft.sqlserver</groupId>
    <artifactId>mssql-jdbc</artifactId>
    <version>12.1.0.jre11-preview</version>
</dependency>
<dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter-api</artifactId>
    <version>5.9.1</version>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>org.mockito</groupId>
    <artifactId>mockito-junit-jupiter</artifactId>
    <version>4.9.0</version>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-jdbc</artifactId>
    <version>6.0.3</version>
</dependency>
<dependency>
    <groupId>org.springframework.session</groupId>
    <artifactId>spring-session-core</artifactId>
```

```
            <version>3.0.0</version>
        </dependency>
        <dependency>
            <groupId>org.springframework.session</groupId>
            <artifactId>spring-session-jdbc</artifactId>
            <version>3.0.0</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>6.0.3</version>
        </dependency>
        <dependency>
            <groupId>org.postgresql</groupId>
            <artifactId>postgresql</artifactId>
            <scope>runtime</scope>
        </dependency>
```

Лістинг 3.2 Код для ініціалізації Docker контейнера:

```
version: '3'
services:
  db:
    image: postgres:latest
    container_name: bank_db
    restart: no
    environment:
      POSTGRES_PASSWORD: 1
      POSTGRES_USER: rasvel
      POSTGRES_DB: db
```

```
    volumes:
      - bank_db_data:/var/lib/postgresql/data
    ports:
      - "5432:5432"
volumes:
  bank_db_data:
    driver: local
```

Лістинг 3.3 Код для створення технічних таблиць Spring Session:

```
CREATE TABLE SPRING_SESSION
(
    PRIMARY_ID          CHAR(36) NOT NULL,
    SESSION_ID          CHAR(36) NOT NULL,
    CREATION_TIME       BIGINT   NOT NULL,
    LAST_ACCESS_TIME    BIGINT   NOT NULL,
    MAX_INACTIVE_INTERVAL INT    NOT NULL,
    EXPIRY_TIME         BIGINT   NOT NULL,
    PRINCIPAL_NAME      VARCHAR(100),
    CONSTRAINT SPRING_SESSION_PK PRIMARY KEY (PRIMARY_ID)
);
CREATE UNIQUE INDEX SPRING_SESSION_IX1 ON SPRING_SESSION (SESSION_ID);
CREATE INDEX SPRING_SESSION_IX2 ON SPRING_SESSION (EXPIRY_TIME);
CREATE INDEX SPRING_SESSION_IX3 ON SPRING_SESSION (PRINCIPAL_NAME);
CREATE TABLE SPRING_SESSION_ATTRIBUTES
(
```

```
    SESSION_PRIMARY_ID CHAR(36)     NOT NULL,
    ATTRIBUTE_NAME     VARCHAR(200) NOT NULL,
    ATTRIBUTE_BYTES    BYTEA        NOT NULL,
    CONSTRAINT SPRING_SESSION_ATTRIBUTES_PK PRIMARY KEY (SESSION_PRIMARY_ID, ATTRIBUTE_NAME),
    CONSTRAINT SPRING_SESSION_ATTRIBUTES_FK FOREIGN KEY (SESSION_PRIMARY_ID) REFERENCES SPRING_SESSION (PRIMARY_ID) ON DELETE CASCADE
);
```

– конфігурація застосунку: створено класи SecurityConfig та SessionConfig для налаштування роботи Spring Security, а також можливості реєстрації та авторизації. Лістинг класів наведено відповідно у лістингах 3.4 та 3.5;

Лістинг 3.4 Реалізація SecurityConfig:

```
@Configuration
@EnableWebSecurity
@RequiredArgsConstructor
public class SecurityConfig {
   private final CredentialsService credentialsService;
   @Bean
   public PasswordEncoder passwordEncoder() {
      return new CustomEncoder();
   }
   @Bean
   public AuthenticationProvider authenticationProvider() {
      DaoAuthenticationProvider provider = new DaoAuthenticationProvider();
      provider.setUserDetailsService(credentialsService);
```

```
        provider.setPasswordEncoder(passwordEncoder());
        return provider;
    }
    @Bean
    public AuthenticationManager authenticationManager() {
        return new ProviderManager(Collections.singletonList(authenticationProvider()));
    }
```

Лістинг 3.5 Реалізація SessionConfig:

```
@Configuration
@EnableWebSecurity
public class SessionConfig {
    @Bean
    public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
        http.csrf().disable().authorizeHttpRequests().
                requestMatchers("/resources/**", "/css/**", "/loader/**", "/js/**").permitAll().
requestMatchers("/login").authenticated().anyRequest().permitAll().
and().formLogin().loginPage("/login").defaultSuccessUrl("/main").permitAll().
and().logout().invalidateHttpSession(true).clearAuthentication(true).
                logoutRequestMatcher(new AntPathRequestMatcher("/logout")).
                logoutSuccessUrl("/login").permitAll() .and().
                securityContext().
                securityContextRepository(securityContextRepository());
        http.sessionManagement(httpSecuritySessionManagementConfigurer ->
```

```
httpSecuritySessionManagementConfigurer.sessionCreationPolicy(SessionCreatio
nPolicy.ALWAYS));
            return http.build();
        }
        @Bean
        public CorsFilter corsFilter() {
            CorsConfiguration corsConfiguration = new CorsConfiguration();
            corsConfiguration.setAllowedOrigins(List.of("http://127.0.0.1:5000"));
            corsConfiguration.setAllowedMethods(List.of("GET", "POST", "PUT",
"DELETE"));
            corsConfiguration.setAllowedHeaders(List.of("*"));
            corsConfiguration.setAllowCredentials(true);
            UrlBasedCorsConfigurationSource source = new
UrlBasedCorsConfigurationSource();
            source.registerCorsConfiguration("/**", corsConfiguration);
            return new CorsFilter(source);
        }
        @Bean
        @Scope(value = ConfigurableBeanFactory.SCOPE_SINGLETON)
        public SecurityContextRepository securityContextRepository() {
            return new HttpSessionSecurityContextRepository();
        }
    }
```

– розробка дизайну: на цьому етапі розроблено дизайн для сторінок авторизації та реєстрації. Основним елементом сторінок є наявність елементу <form> та полів <input>, які містять значення, зазначенні у таблиці 2.1 при реєстрації, електрону пошту та пароль при авторизації. Приклад елементу <form> наведено у лістингу 3.6;

Лістинг 3.6 Елемент <form> на сторінці реєстрації:

```
<form method="post" th:action="@{/registration}">
        <div class="field"><label>Введіть ПІБ</label>
          <input name="name" pattern="([A-Z][a-z]+\s?)+"
placeholder="Введіть ПІБ" required type="text"></div>
        <div class="field"><label>Введіть номер паспорту</label>
          <input name="passport" pattern="[0-9]{10}"
placeholder="Введіть номер паспорту" required
              type="text"></div>
        <div class="field"><label>Введіть ідентифікаційний
код</label>
          <input name="identicalCode" pattern="[0-9]{12}"
placeholder="Введіть ідентифікаційний код"
              required type="text"></div>
        <div class="field"><label>Введіть e-mail</label>
          <input name="email" pattern="([a-z0-9]\.?){1,15}@[a-
z]{2,10}.[a-z]{2,5}"
              placeholder="Введіть e-mail"
              required type="text"></div>
        <div class="field"><label>Введіть номер телефону</label>
          <input name="phone" pattern="\+[1-9][0-9]{0,12}"
placeholder="Введіть номер телефону" required
              type="text"></div>
        <div class="field"><label>Введіть пароль</label>
          <input name="password" placeholder="Введіть пароль"
required type="password">
        </div>
        <div class="permission">
          <label class="form-control">
```

```
                    <input name="permission" type="checkbox" />
                    <p>Чи надаєте ви згоду <a>на обробку персональних даних</a></p>
                  </label>
                </div>
                <div class="field"><input type="submit" value="Підтвердити"></div>
              </form>
```

– розробка API. Створено 2 класи: UserController, PhotoController, які позначено анотацією @Controller. Клас UserController відповідає за можливість відображення створених раніше дизайнів сторінок, реєстрацію та авторизацію користувачів. Клас PhotoController відповідає за збереження та отримання фото користувачів з бази даних. Основні функції класів наведено у лістингах 3.7 та 3.8 відповідно.

Лістинг 3.7 Реалізація класу UserController:

```
@RestController
@RequiredArgsConstructor
@Slf4j
public class UserController {
   private final AuthenticationProvider authenticationProvider;
   private final SecurityContextRepository securityContextRepository;
  @GetMapping("/login")
   public ModelAndView login() {
      return new ModelAndView("/login");
   }
   @GetMapping("/logout")
   public ModelAndView logout() {
      return new ModelAndView("redirect:/login");
```

```
    }
    @GetMapping("/main")
    public ModelAndView main() {
        log.info("main");
        BankUser bankUser = userManagerService.getAuthenticatedUser();
        ModelAndView modelAndView = new ModelAndView("main");
        modelAndView.getModelMap().addAttribute("username",
bankUser.getName());
        modelAndView.getModelMap().addAttribute("cards",
bankUser.getCards());
        modelAndView.getModelMap().addAttribute("identicalCode",
bankUser.getIdenticalCode());
        modelAndView.getModelMap().addAttribute("recipientAccount",
bankUser.getAccountNumber());
        modelAndView.getModelMap().addAttribute("transactions",
userManagerService.getUserTransactions(bankUser));
        return modelAndView;
    }
    @GetMapping("/register")
    public ModelAndView register() {
        return new ModelAndView("register");
    }
    @PostMapping("/registration")
    public ModelAndView register(BankUser bankUser,
                                 @RequestParam String password,
                                 @RequestParam(name = "permission", required = false)
Boolean confirmation) throws Exception {
        ModelAndView modelAndView = new ModelAndView("/register");
        if (confirmation && userManagerService.checkUser(bankUser)) {
            BankUser newBankUser = userManagerService.addUser(bankUser);
```

```
            credentialsService.saveCredentials(bankUser, password);
            String encodedId = EncryptionService.encryptLong(newBankUser.getUserId());
            ModelAndView embeddingsModel = new ModelAndView("redirect:http://127.0.0.1:5000/face/register");
            embeddingsModel.getModelMap().addAttribute("id", encodedId);
            return embeddingsModel;
        } else {
            return modelAndView;
        }
    }
    @PostMapping("/authentication")
    public ModelAndView authentication(@RequestParam String email) throws Exception {
        BankUser user = userManagerService.findUserByEmail(email);
        String encodedId = EncryptionService.encryptLong(user.getUserId());
        ModelAndView embeddingsModel = new ModelAndView("redirect:http://127.0.0.1:5000/face/login");
        embeddingsModel.getModelMap().addAttribute("id", encodedId);
        return embeddingsModel;
    }
    @GetMapping("/authenticate/{userId}")
    public ModelAndView authenticate(@PathVariable(value = "userId") String encodedId, HttpServletRequest request, HttpServletResponse response) throws Exception {
        log.info("authentication started");
        Long id = EncryptionService.decryptLong(encodedId);
        BankUser bankUser = userManagerService.findUserById(id);
        UserDetails userCredentials = credentialsService.loadUserByUsername(bankUser.getEmail());
```

```
            UsernamePasswordAuthenticationToken authReq
                    = new UsernamePasswordAuthenticationToken(userCredentials,
                    userCredentials.getPassword());
            Authentication auth = authenticationProvider.authenticate(authReq);
            SecurityContext sc = SecurityContextHolder.createEmptyContext();
            sc.setAuthentication(auth);
            securityContextRepository.saveContext(sc,request,response);
            log.info("authentication finished");
            log.info(userManagerService.getAuthenticatedUserEmail());
            return new ModelAndView("redirect:/main");
        }
    }
```

Лістинг 3.8 Реалізація класу PhotoController:

```
    @RestController
    @RequiredArgsConstructor
    public class PhotoController {
        @PostMapping("/receive")
        public void saveEmbeddings(@RequestBody ImagePojo imagePojo)
throws Exception {
            if(imagePojo.getImage() == null){
                throw new RuntimeException("Image is wrong");
            }
            Long id = EncryptionService.decryptLong(imagePojo.getId());
            String image = imagePojo.getImage();
            photoService.saveEmbeddings(id, image);
        }
        @GetMapping("/image/{userId}")
        public String getImage(@PathVariable String userId) throws Exception {
```

```
        Long id = EncryptionService.decryptLong(userId);
        return photoService.getImage(id);
    }
}
```

3.2.3 Розроблення Python складової

На даному етапі розроблено мікросервіс на мові Python для роботи з зображеннями та дизайн сторінки авторизації за FaceID.

Опис виконаних кроків:

– ініціалізація Python проєкту: за допомогою середовища PyCharm створено новий Python проєкт. Налаштування при ініціалізації показано на рисунку 3.2;

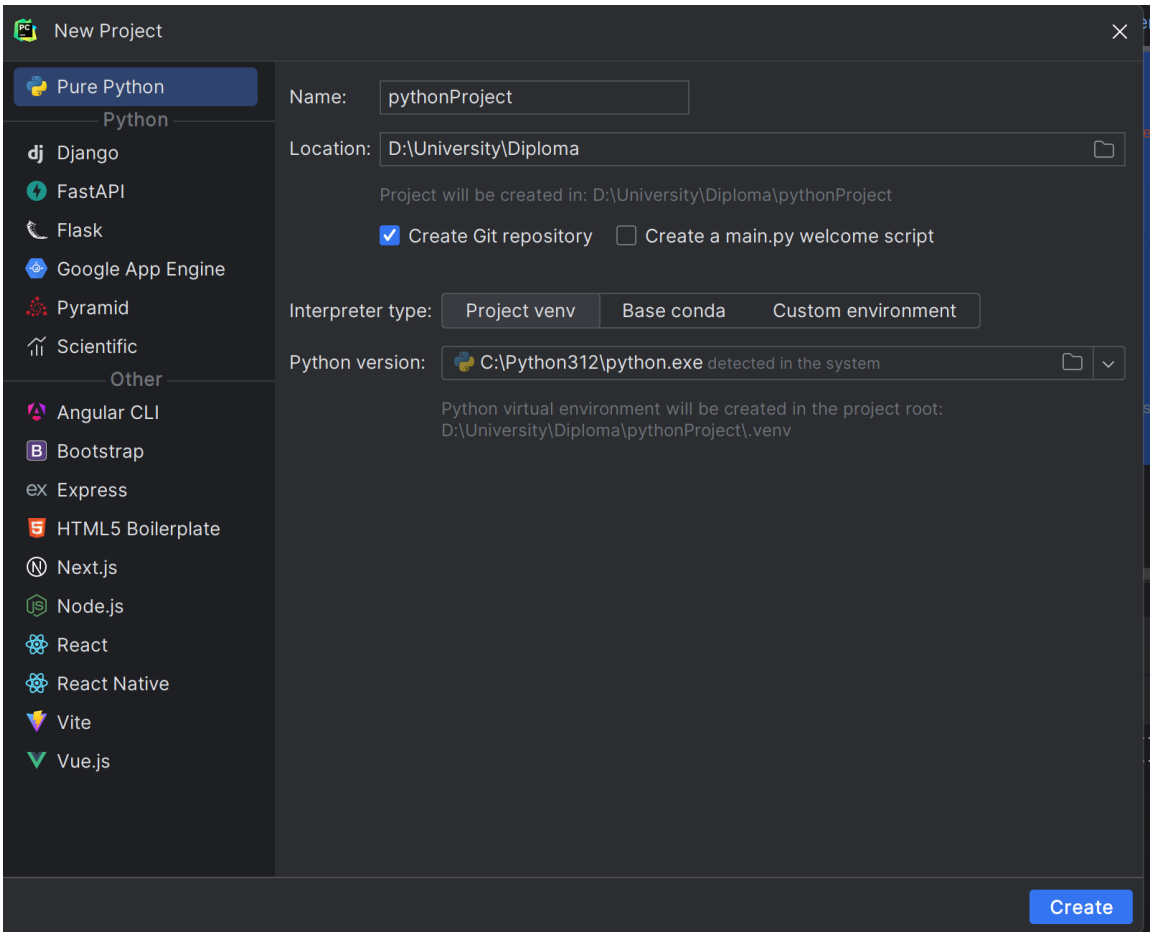


Рисунок 3.2 – Приклад ініціалізації Python проєкту

– розробка дизайну: на цьому етапі розроблено дизайн для сторінки авторизації. Основним компонентом є елемент <img> та його властивість src, яка відповідає за показ відео з вебкамери. Приклад коду сторінки авторизації за FaceID наведено у лістингу 3.9;

Лістинг 3.9 Приклад дизайну сторінки авторизації за FaceID:

```
<!DOCTYPE html>
<html lang="" xmlns="http://www.w3.org/1999/xhtml">
<head>
  <meta charset="UTF-8">
  <title>Main</title>
  <link href="{{ url_for('static', filename='main.css') }}" rel="stylesheet" type="text/css" />
  <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
</head>
<body>
<div id="timer"></div>
<div id="image">
  <div id="finish-description">Обробка фото...</div>
  <img id="web-camera" src="{{ url_for('video_feed') }}" alt="Webcam Image" />
  <div id="comment"></div>
</div>
<script src="{{ url_for('static', filename='script.js') }}"></script>
</body>
</html>
```

– розробка API. Створено файли: camera_functions.py, detect_init_face_functions.py. Перший файл містить усі функції, які відповідають за увімкнення вебкамери, відстеження обличчя, аналіз

зображення, тощо. Код функцій з файлу camera_functions.py наведено у лістингу 3.10. У файлі detect_init_face_functions.py написані функції, що відповідають за виклик методів порівняння облич, отримання зображення користувача з бази даних, тощо. Основні функції з файлу detect_init_face_functions.py показані у лістингу 3.11.

Лістинг 3.10 Функції роботи з камерою з файлу camera_functions.py:

```
import base64
import cv2
import numpy as np
import requests
from functions import detect_init_face_functions as face_detect
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
video_capture = cv2.VideoCapture(0)
comment = ''
def stopCamera():
    global streaming
    streaming = False
    video_capture.release()
def make_snapshot(user_id):
    image_in_bytes = get_image()
    print('byte[]', image_in_bytes)
    url = 'http://127.0.0.1:8080/receive'
    payload = {'id': user_id, 'image': image_in_bytes}
    requests.post(url, json=payload)
    return 'snapshot made'
def generate_frames():
    brightness_threshold_low = 90
    brightness_threshold_high = 110
```

```
        while True:
            if video_capture is not None:
                success, frame = video_capture.read()
                if not success:
                    break
                else:
                    errors = []
                    outer_center = (int(frame.shape[1] // 2), int(frame.shape[0] // 2))
                    outer_size = (int(frame.shape[0] // 3.5), int(frame.shape[1] // 3.5))
                    cv2.ellipse(frame, outer_center, outer_size, 0, 0, 360, (255, 255, 255), 2)
                    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
                    mean_intensity = np.mean(gray)
                    average_intensity = mean_intensity
                    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))
                    if len(faces) > 0:
                        face = faces[0]
                        print(average_intensity)
                         if average_intensity <= brightness_threshold_low or average_intensity >= brightness_threshold_high:
                            if average_intensity <= brightness_threshold_high:
                                errors.append("Зображення занадто темне. Змініть освітлення")
                             elif average_intensity >= brightness_threshold_low:
                                errors.append("Зображення занадто світле. Змініть освітлення")
                        (x, y, w, h) = face
                        inner_center = ((x + x + w) // 2, (y + y + h) // 2)
                        inner_size = (w // 2, h // 2)
```

```
                    isFaceInArea = check_if_face_in_area((outer_center, outer_size, 0), (inner_center, inner_size, 0))
                    if not isFaceInArea:
                        errors.append(
                            "Обличчя розпізнано, але знаходиться поза межами зони ідентифікації. Будь ласка, розташуйте обличчя у білій зоні")
                    elif isFaceInArea and len(errors) == 0:
                        errors.append("allow to photo")
                    color = (0, 255, 0) if isFaceInArea else (0, 0, 255)
                    cv2.ellipse(frame, inner_center, inner_size, 0, 0, 360, color, 2)
                else:
                    errors.append("Ми не змогли ідентифікувати ваше обличчя. Будь ласка, змініть його положення")
                frame = cv2.flip(frame, 1)
                global comment
                comment = ' '.join(errors)
                ret, buffer = cv2.imencode('.jpg', frame)
                frame = buffer.tobytes()
                yield (b'--frame\r\n'
                       b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')
    def get_comment():
        if comment is None:
            return ''
        return comment
    def check_if_face_in_area(outer_ellipse, inner_ellipse):
        outer_rectangle = Rectangle(outer_ellipse)
        inner_rectangle = Rectangle(inner_ellipse)
        return compare_rectangles(outer_rectangle, inner_rectangle)
    def compare_rectangles(outer, inner):
        return (outer.a.x <= inner.a.x and outer.a.y <= inner.a.y and
```

```
            outer.b.x >= inner.b.x and outer.b.y <= inner.b.y and
            outer.c.x >= inner.c.x and outer.c.y >= inner.c.y and
            outer.d.x <= inner.d.x and outer.d.y >= inner.d.y)
def get_image():
    if video_capture is not None:
        success, frame = video_capture.read()
        if not success:
            return 'error'
        else:
            face = face_detect.detect_face(frame)[0]
            _, img_encoded = cv2.imencode('.jpg', face)
            return base64.b64encode(img_encoded.tobytes()).decode('utf-8')
def get_image_from_database(id):
    url = 'http://127.0.0.1:8080/image/' + id
    response = requests.get(url)
    return response.content
class Rectangle:
    def __init__(self, ellipse):
        center, size, _ = ellipse
        w, h = size
        self.a = Point(center[0] - w / 2, center[1] - h / 2)
        self.b = Point(center[0] + w / 2, center[1] - h / 2)
        self.c = Point(center[0] + w / 2, center[1] + h / 2)
        self.d = Point(center[0] - w / 2, center[1] + h / 2)
class Point:
    def __init__(self, x, y):
        self.x = x
        self.y = y
```

Лістинг 3.11 Функції ідентифікації обличчя з файлу detect_init_face_functions.py:

```
import base64
import importlib
import os
import queue
import threading
from dataclasses import dataclass
from typing import List, Dict
import cv2
import numpy as np
import requests
from deepface import DeepFace
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
def detect_face(image):
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))
    return faces
def extract_face_embedding(image, face):
    x, y, w, h = face
    face_img = image[y:y + h, x:x + w]
    return DeepFace.represent(face_img, model_name='Facenet')
def compare_faces(user_id, internal_image):
    url = 'http://127.0.0.1:8080/image/'+str(user_id)
    response = requests.get(url)
    image1 = revert_image_from_base64(response.content)
    image2 = revert_image_from_base64(internal_image)
    return run_comparing(image1, image2)
```

```
        def run_comparing(image1, image2):
            modules = load_modules_from_folder()
            return run_modules(modules, image1, image2)
        def load_modules_from_folder():
            folder_path = "folder_path"
            modules = []
            for file_name in os.listdir(folder_path):
                if file_name.endswith(".py") and file_name != "__init__.py":
                    module_name = file_name[:-3]  # Remove the .py extension
                    spec = importlib.util.spec_from_file_location(module_name,
os.path.join(folder_path, file_name))
                    module = importlib.util.module_from_spec(spec)
                    spec.loader.exec_module(module)
                    modules.append(module)
            return modules
        def run_modules(modules, image1, image2):
            comparison_results = []
            threads = []
            que = queue.Queue()
            for module in modules:
                if hasattr(module, "run") and callable(module.run):
                    print(f"Running module {module.__name__}...")
                    thread = threading.Thread(target=lambda q, arg1, arg2:
q.put(module.run(arg1, arg2)),
                                              args=(que, image1, image2))
                    threads.append(thread)
                    thread.start()
                else:
                    print(f"Module {module.__name__} does not have a 'run' function or
it's not callable.")
            for thread in threads:
```

```
            thread.join()
        while not que.empty():
            comparison_results.append(que.get())
        return comparison_results
    def revert_image_from_base64(base64_str):
        img_data = base64.b64decode(base64_str)
        nparr = np.frombuffer(img_data, np.uint8)
        img = cv2.imdecode(nparr, cv2.IMREAD_COLOR)
        return img
```

Таким чином, застосування комбінації мов програмування Python та Java дозволило забезпечити надійність та ефективність функціональних можливостей під час розроблення вебзастосунку;

– розробка методів аналізу обличчя: розроблено та імплементовано методи аналізу обличчя: eigenfaces, OpenFace, FaceNet (лістинги 2.2 – 2.4).

3.2.4 Ручне тестування розробленого застосунку та аналіз результатів

Для проведення ручного тестування запущено застосунок у локальному просторі.

Після переходу за посиланням http://localhost:8080 можна побачити, що застосунок запущений коректно і початковою сторінкою є сторінка авторизації (рис. 3.3).

Для того, щоб протестувати можливість авторизації необхідно створити користувача, зробивши перехід на сторінку реєстрації, яка зображена на рисунку 3.4.

Після реєстрації користувач одразу попадає на головну сторінку застосунку (рис. 3.5), на якій може подати запит на створення банківської картки, побачити свої прізвище та ім’я, вийти з акаунту. Для того, щоб протестувати авторизацію за Face ID, зроблено вихід із застосунку.

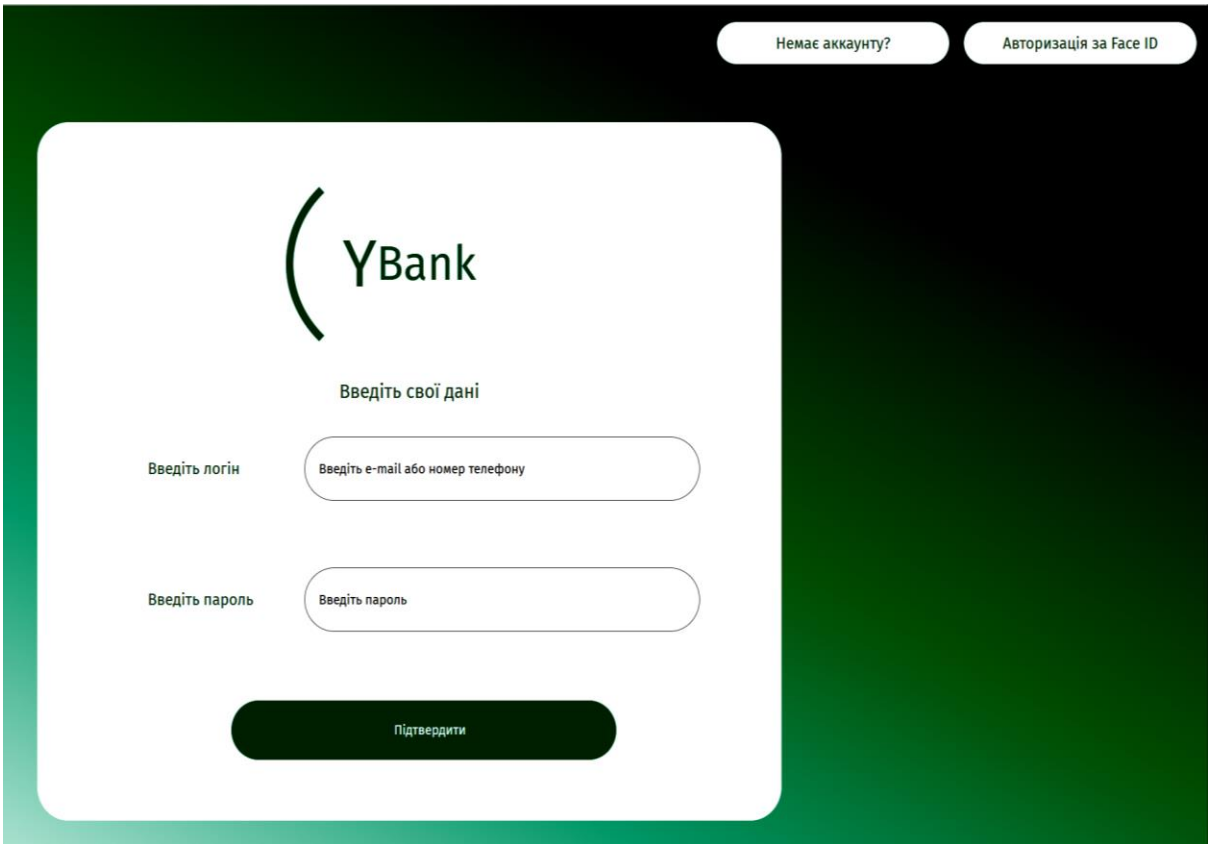


Рисунок 3.3 – Сторінка авторизації

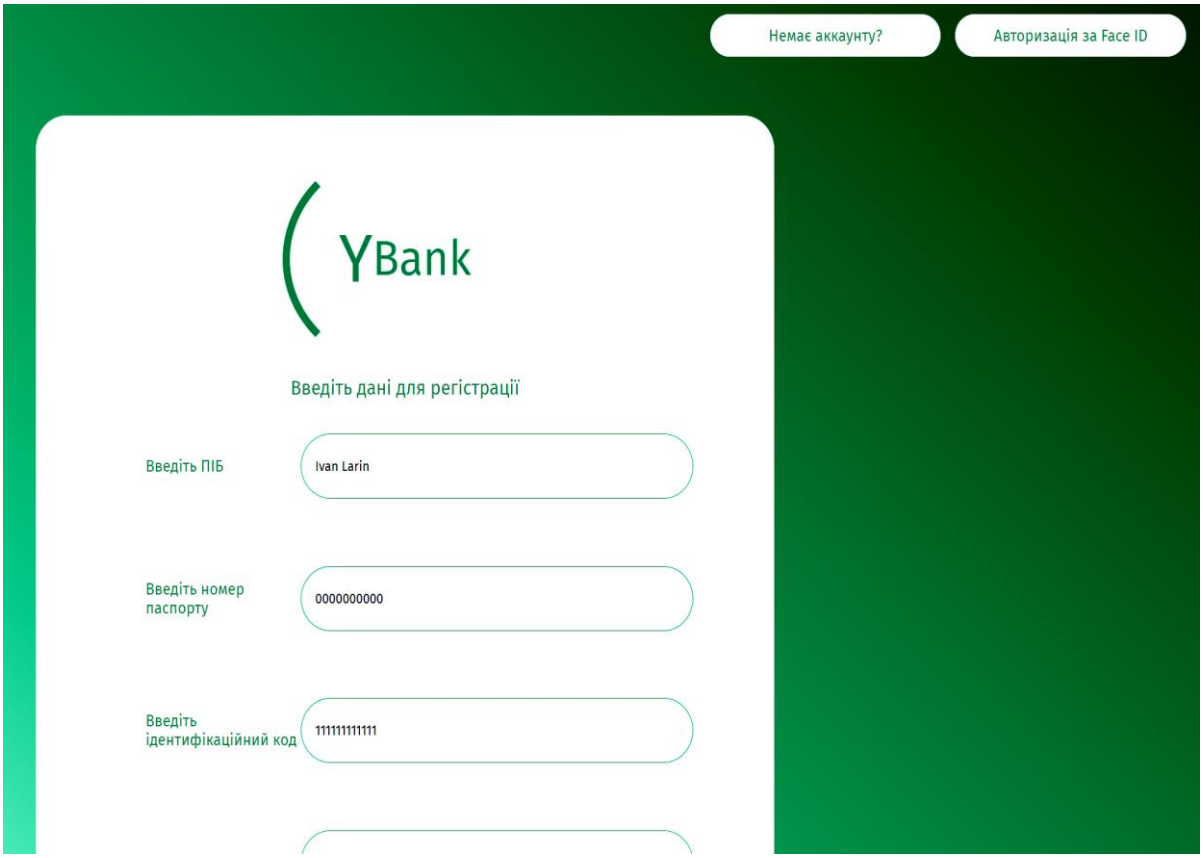


Рисунок 3.4 – Сторінка реєстрації

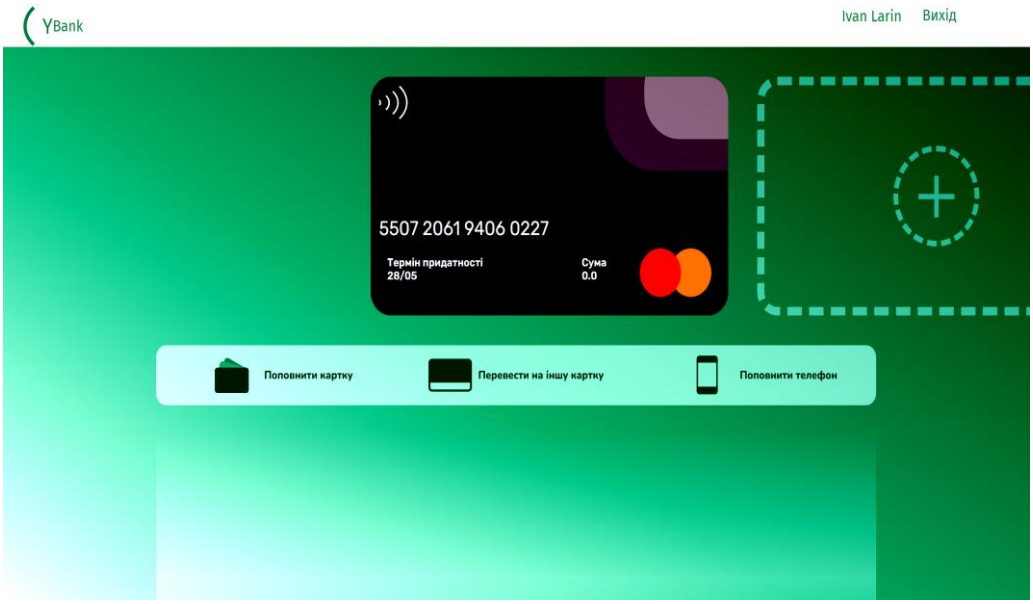


Рисунок 3.5 – Головна сторінка застосунку

Коли користувач на головній сторінці обирає пункт «Авторизація за Face ID», він попадає на іншу сторінку (рис. 3.6), де має можливість ввести власну адресу електронної пошти. Цю функцію реалізовано для порівняння зображень конкретного користувача, з метою уникнення пошуку по всій базі даних.

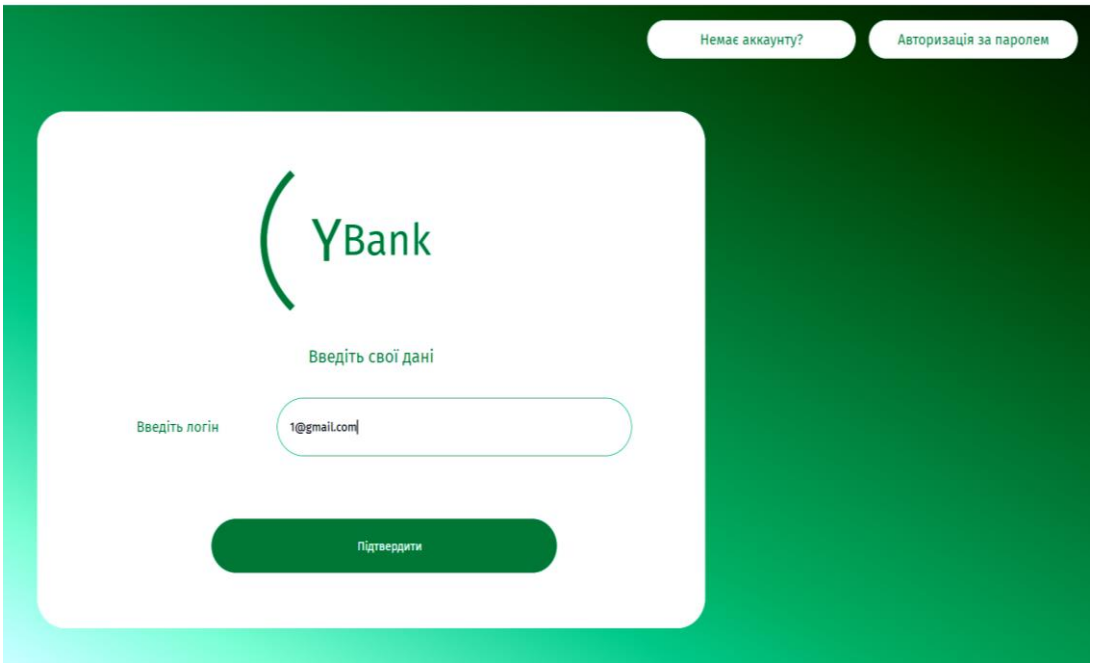


Рисунок 3.6 – Сторінка авторизації за Face ID
(етап введення електронної пошти)

Якщо електронна адреса існує, то користувач попадає на сторінку розпізнавання обличчя, яку зображено на рисунку 3.7.

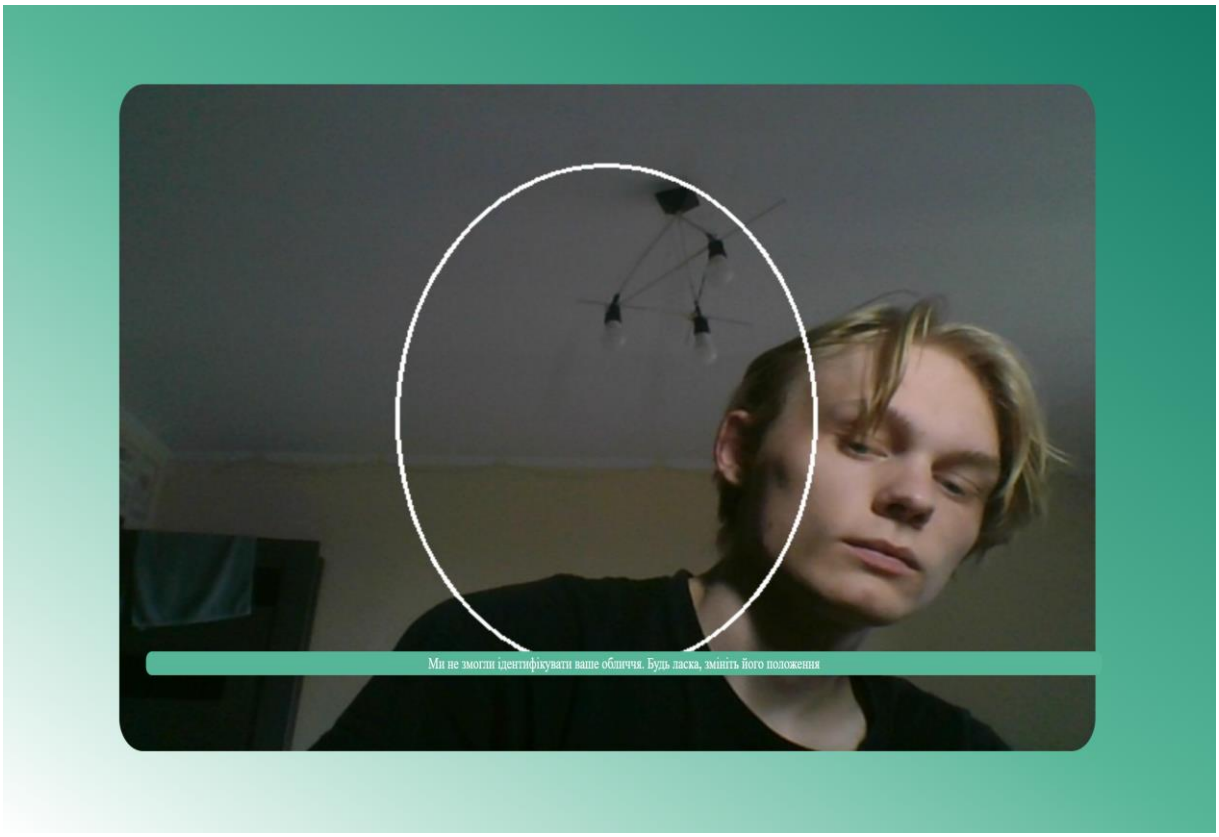


Рисунок 3.7 – Сторінка розпізнавання обличчя

Якщо обличчя на зображенні розпізнано, то з’явиться червоне або зелене коло. Червоне коло повідомляє, що обличчя розпізнано, але знаходиться не в області для створення фото, зображеної у вигляді білого овалу. Зелене показує, що обличчя розташоване правильно.

Якщо для створення фото не виконані якісь із умов, то буде показаний відповідний коментар. Приклади коментарів показані на рисунках 3.8 – 3.11.

Якщо обличчя розпізнано коректно, знаходиться у межах області фотографування, яка не є затемною чи засвітлою, то користувачу буде показано повідомлення зі зворотнім відліком до фотографування. Приклад демонструється на рисунку 3.12. У випадку, коли фото зроблено і коректно опрацьовано (користувач не зник з кадру у момент фіксування фото), користувача буде авторизовано.

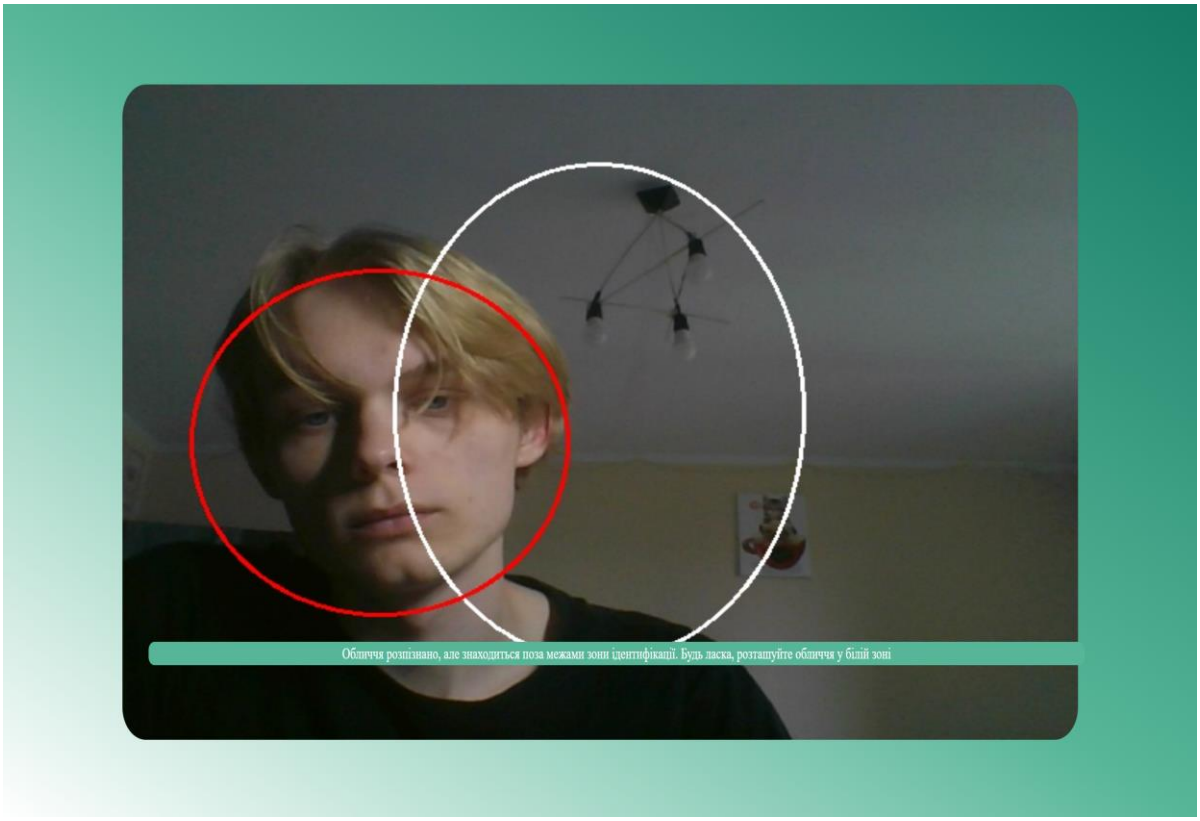


Рисунок 3.8 – Приклад обробки фото, за умови розпізнаного обличчя, що не знаходиться у межах області фотографування

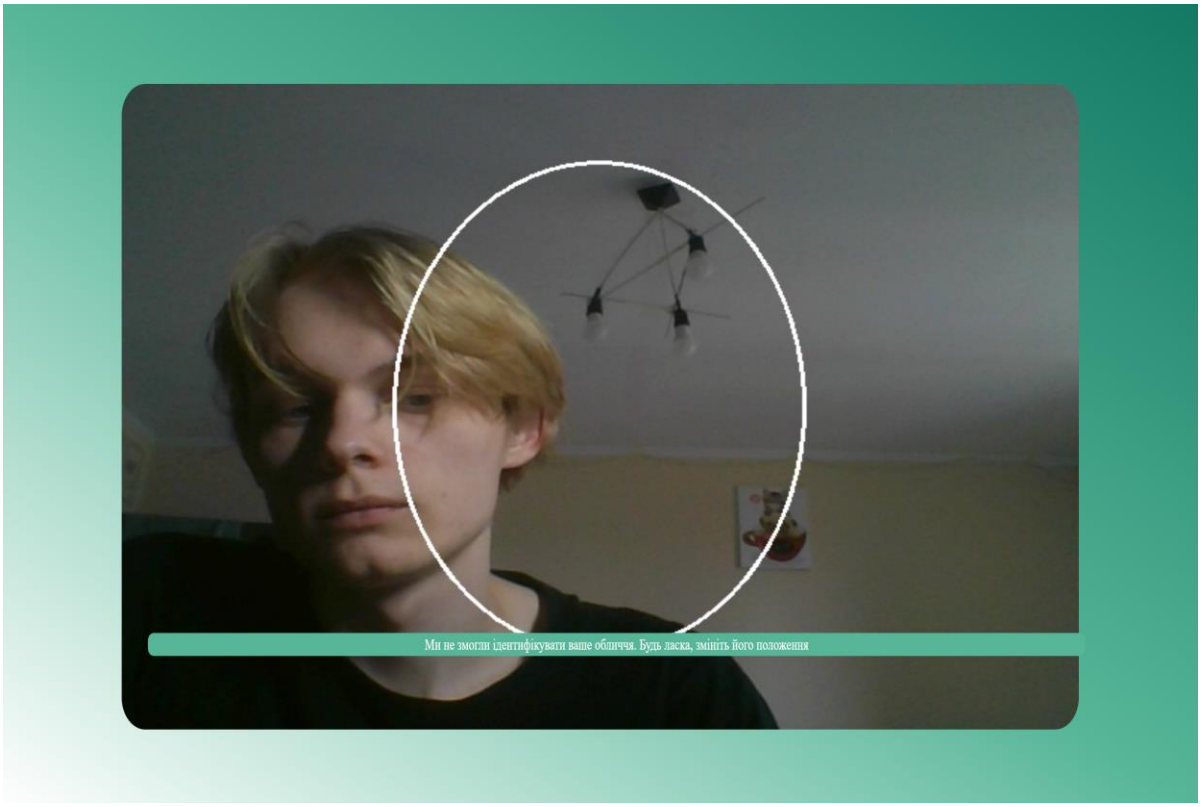


Рисунок 3.9 – Приклад обробки фото, за умови нерозпізнаного обличчя

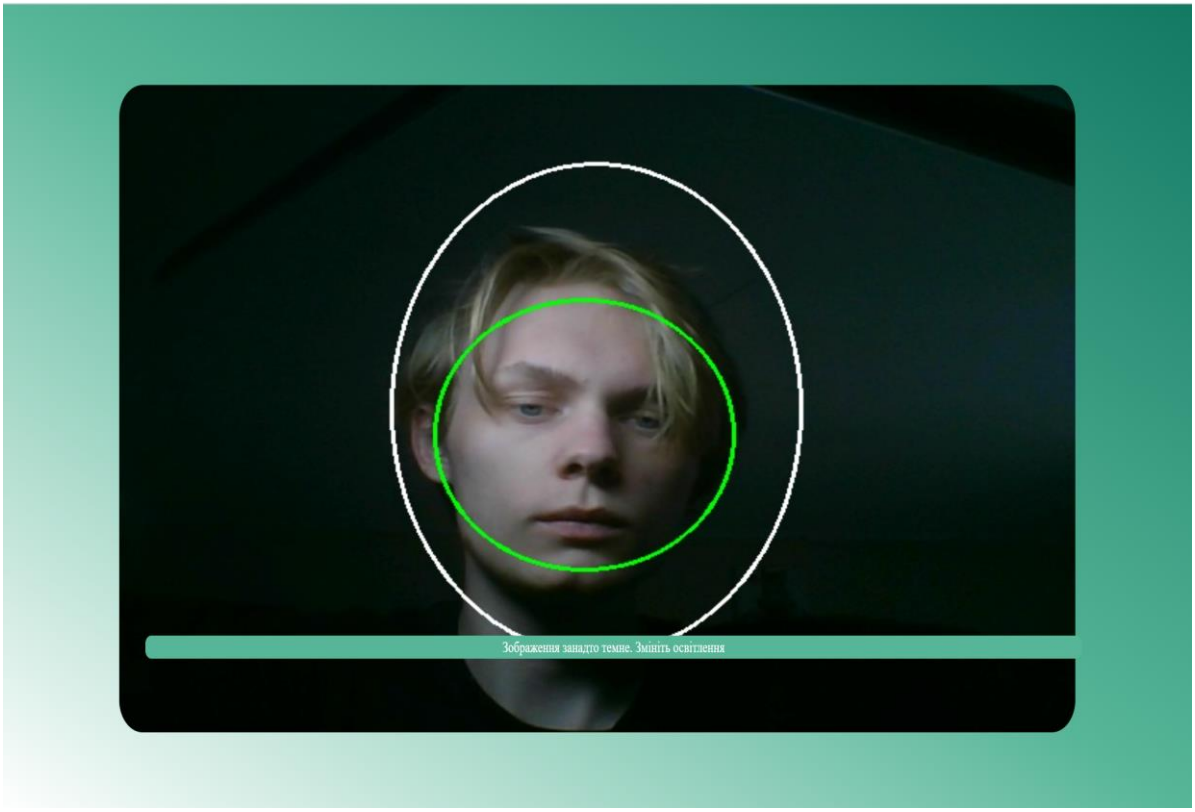


Рисунок 3.10 – Приклад обробки фото, за умови розпізнаного обличчя та занадто темного зображення

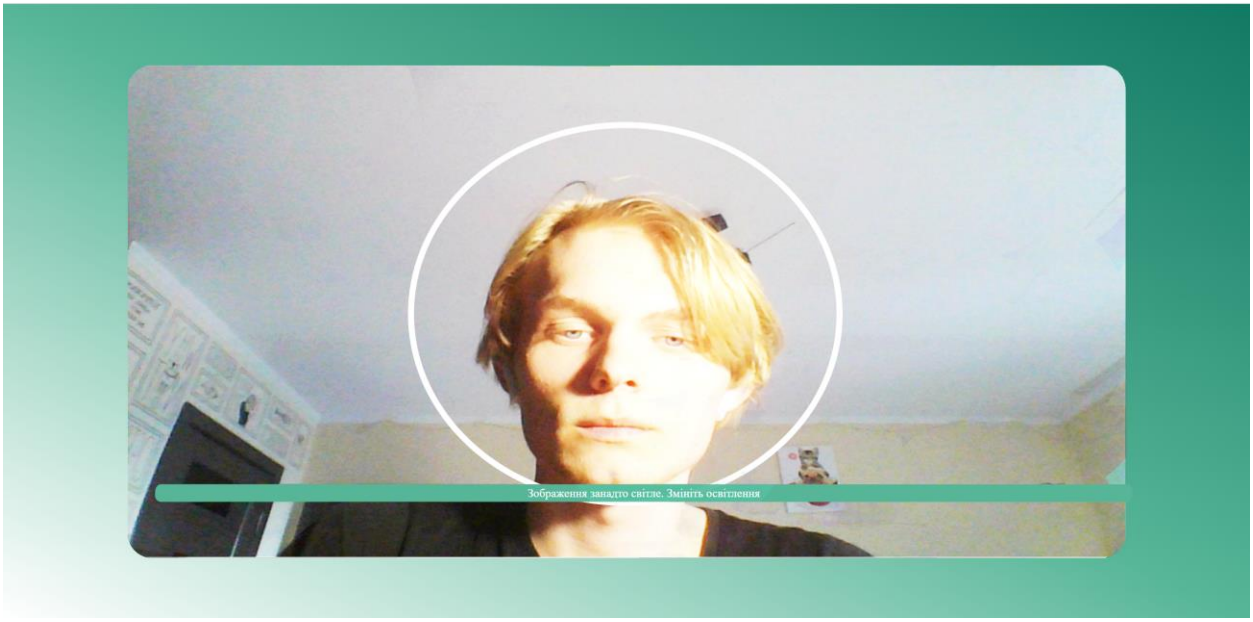


Рисунок 3.11 – Приклад обробки фото, за умови розпізнаного обличчя та занадто світлого зображення

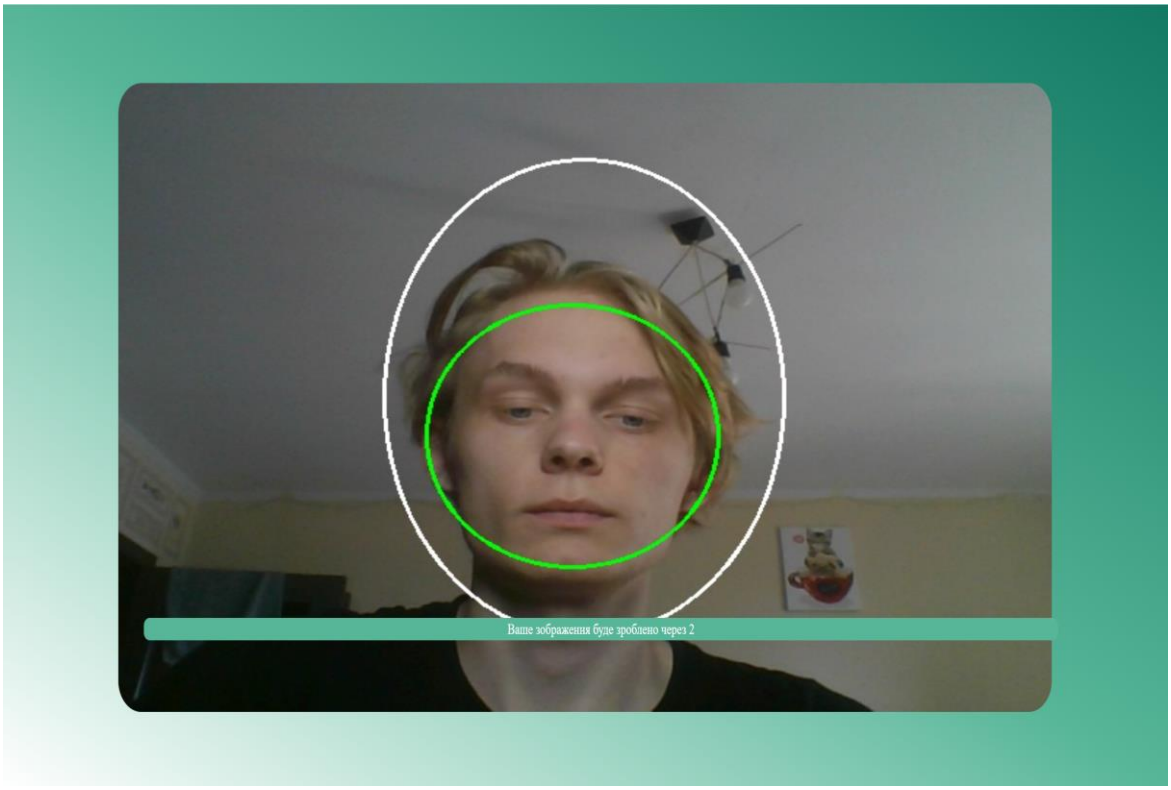


Рисунок 3.12 – Зворотній відлік до фотографування

Під час авторизації показується вікно з повідомленням про прохання очікувати. Приклад вікна показано на рисунку 3.13.

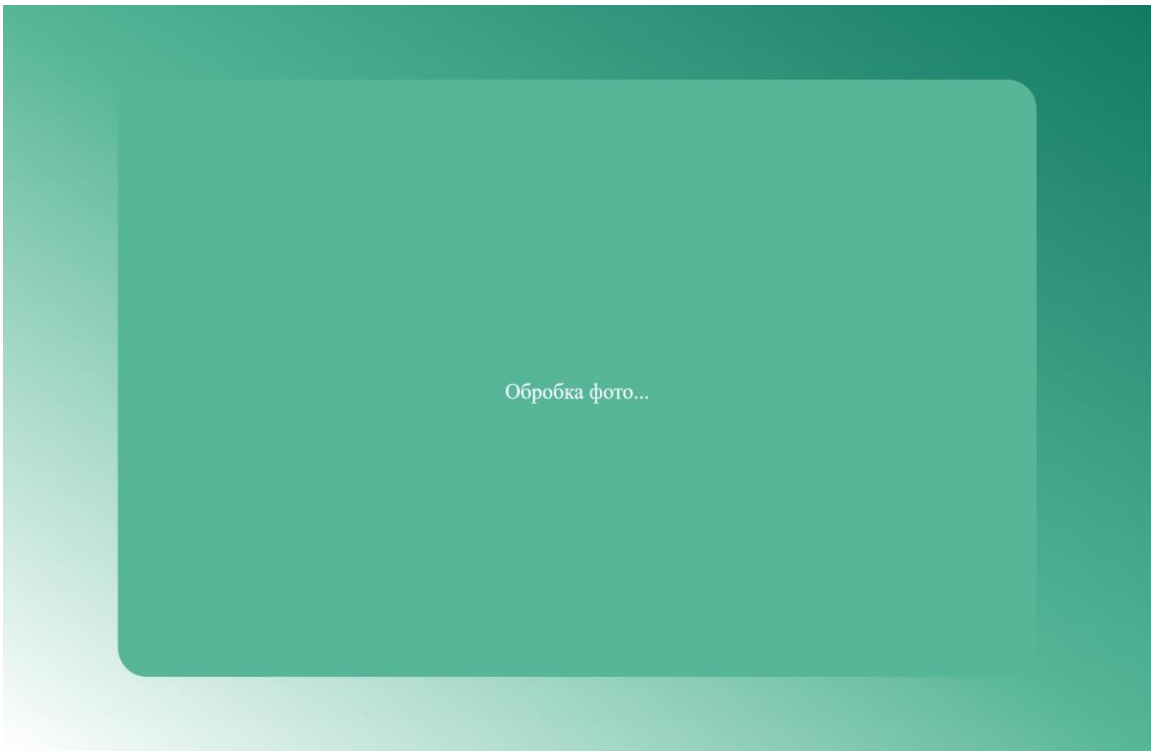


Рисунок 3.13 – Вікно очікування під час обробки фото при авторизації

У випадку успішної авторизації користувачу буде показано головну сторінку особистого кабінету (рис. 3.5).

Під час порівняння відбувається порівняння зображення, яке було зроблено під час процесу, описаного вище, а також зображення у базі даних. У даній реалізації це відбувається трьома методами, Eigenfaces, FaceNet, OpenFace відповідно, які повертають наступні ймовірності (рис. 3.14) під час порівняння облич на рисунках 3.15 та 3.16.

Розрахунок середньої відсоткової схожості облич дозволяє авторизувати користувача.

У випадку невдалої авторизації користувачеві надається повідомлення про помилку (рис. 3.17).

```
result [86.53, 98.94, 99.66]
similarity in percents 95.04
```

Рисунок 3.14 – Відсоток схожості облич при оцінці методами Eigenfaces, FaceNet, OpenFace відповідно та середній відсоток схожості

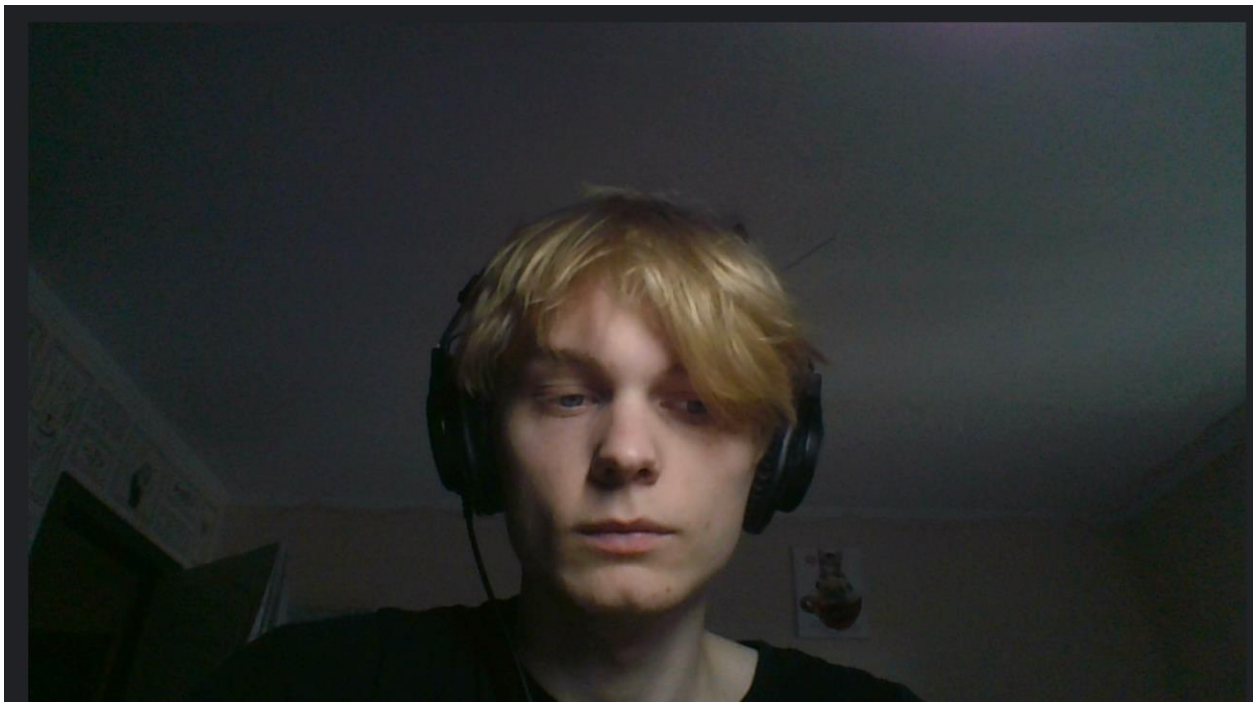


Рисунок 3.15 – Приклад обличчя, яке порівнюється під час коректної авторизації (зображення зроблене системою при авторизації)

Рисунок 3.16 – Приклад обличчя, яке порівнюється під час коректної авторизації (зображення зроблене системою при реєстрації)

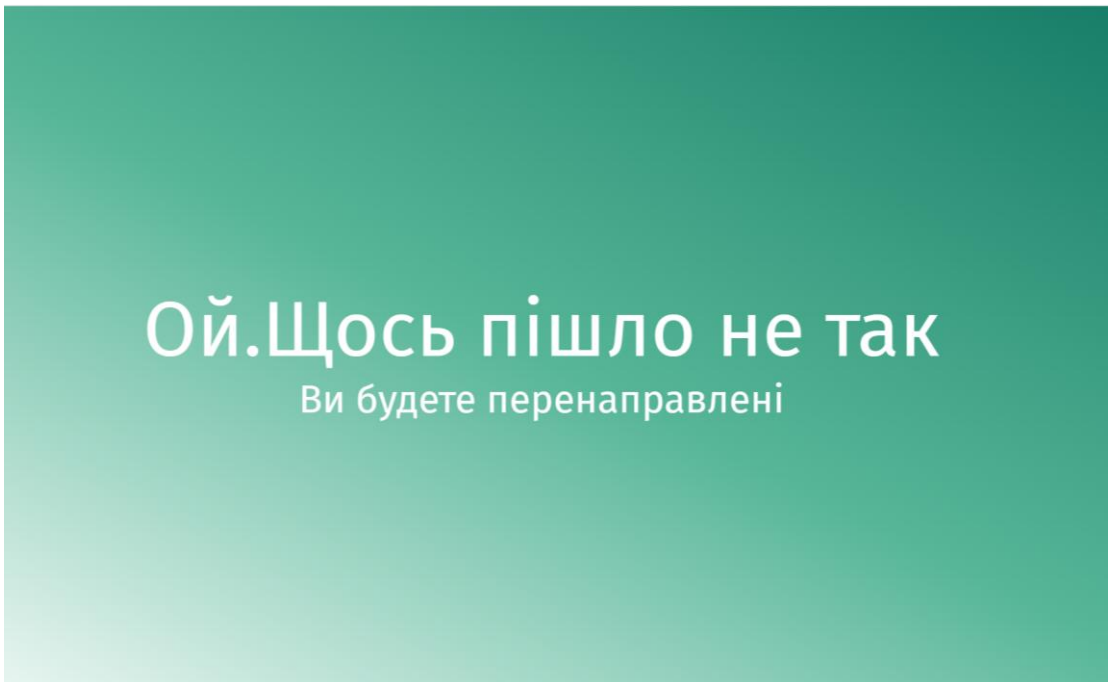


Рисунок 3.17 – Повідомлення про помилку

Під час ручного тестування системи можна побачити, що основні функції реєєстрації та авторизації працюють коректно і дають можливість здійснити відповідні функції.

3.2.5 Тестування методів оцінки схожості зображень та аналіз результатів

Для розуміння показника якості обраних методів Eigenfaces, FaceNet та OpenFace проведено тестування на наборі даних із 32 зображень для 8 людей.

Зображення зберігаються у папках по 4 зображення на кожну людину. Порівняння відбувається в межах однієї папки за принципом «кожна до кожної».

Приклад зображення із набору даних наведено на рисунку 3.18 [21].

Рисунок 3.18 – Приклад зображення із набору даних

Результати тестування наведені у додатку Б. Середній відсоток схожісті становить близько 85%.

Для розуміння, який ступінь схожості у двох зображень з різними людьми, проведено додаткове тестування, де використано зображення 3.19, 3.20 [21].

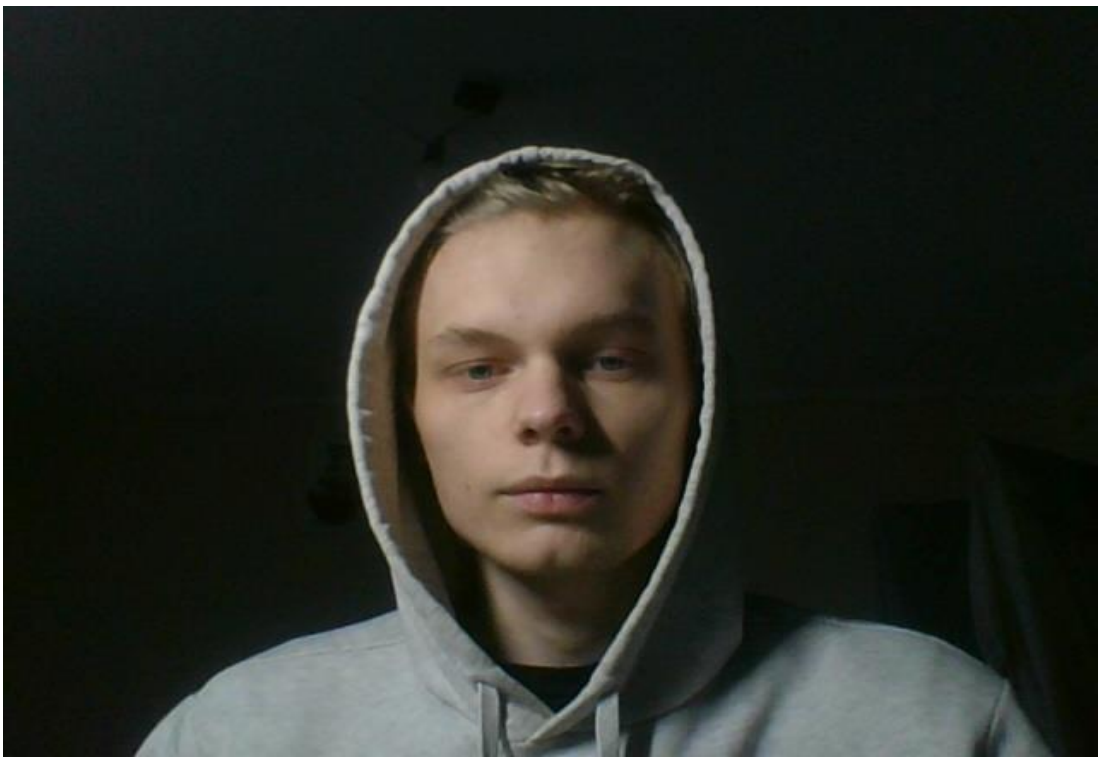


Рисунок 3.19 – Приклад обличчя № 1,
яке порівнюється під час некоректної авторизації

Рисунок 3.20 – Приклад обличчя № 2,
яке порівнюється під час некоректної авторизації

Під час порівняння отримано такий результат (рис. 3.21).

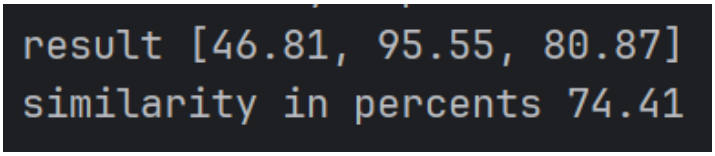


Рисунок 3.21 – Ступінь схожості зображення з двома різними людьми

Найчастіше програма розпізнає схожість обличчя людей з вірогідністю 83% – 88%. У випадках, коли люди різні, відсоток є нижчим – ~70%.

Рівень, який може бути вставновлений для поточної реалізації системи є 86%. За умови, якщо відсоток схожості є нижчим, рекомендовано не авторизувати користувача. Метод Eigenfaces зазвичай видає найнижчий показник схожості 70% – 90%, а FaceNet та OpenFace – вище 90%. Eigenfaces найкраще спрацював при порівнянні зображень різних людей, результат – 46%, FaceNet показав схожість у 95%.

Таким чином, рекомендовано використовувати декілька методів для перешкоджання невірної авторизації через низьку точність [22 – 29].

3.3 Перспективи подальшої роботи

Під час розробки вебзастосунку розпізнавання та ідентифікації користувачів банківської системи були враховані поставлені завданням умови і реалізовані функції реєстрації та авторизації, які включали в себе розробку інших функцій. Таким чином, виконано великий обсяг роботи, але все ще залишаються напрямки, які потребують подальшої доробки [22 – 30].

Першим з таких напрямків є вдосконалення функції розпізнавання та ідентифікації облич. Зокрема, дослідження нових методів, використання підходів та способів ідентифікації облич для підвищення відсотку схожості, за яким дозволено авторизацію. Окрім існуючих вимог при фотографуванні користувача можна додати відслідковування дальності користувача від

вебкамери, додати необхідність виконувати певні дії для авторизації (кивок, кліпання очима), робити декілька зображень і вимагати при авторизації, щоб обличчя співпадало з різних боків, а не тільки в анфас. Іншою можливістю є захист від спуфінгу, коли одна людина, наприклад, використовує фотографію іншої особи для отримання доступу замість неї.

Другим напрямком треба зазначити підвищення відказостійкості системи. Необхідно провести більш глибоке мануальне тестування, виявлення помилок та їх виправлення. Зокрема, зосередитись на тестуванні розпізнавання обличчя, а також врахувати можливість одночасної авторизації декількома користувачами.

Наступним напрямком, який може покращити систему, є збільшення тестування за допомогою різних автоматизованих підходів: проведення JUnit- та Cucumber-тестів, використання JMeter. Це може забезпечити розуміння надійності системи.

Четвертий напрямок це розширення системи. Можна додати ролі адміністратора та робітника банку, який буде мати можливість працювати з акаунтами клієнтів. У цьому напрямі також можна додати можливість поповнювати картки, переводити між ними кошти.

Є можливість покращити візуальну складову. Можна включити як виправлення візуальних помилок у вигляді невірного вирівнювання, так і додавання нового: додати можливість адміністратору чи керівнику банку додавати, вмикати та вимикати певні методи розпізнавання обличчя.

Останнім напрямком є підвищення захисту даних. Наразі паролі зберігаються у базі даних, як і зображення користувачів. До того ж обмін між мікросервісами використовує закодовані дані, однак, вони можуть бути розкодовані. Подальше впровадження нових технологій захисту може покращити захист системи.

Таким чином, перспективи розвитку системи є досить значними. Їх реалізація може підвищити загальну якість IT-продукту, а також створити можливість виведення його на інформаційний ринок.

ВИСНОВКИ

У даній кваліфікаційній роботі продемонстровано процес розробки вебзастосунку з можливістю розпізнавання та ідентифікації користувачів банківської системи. Робота включала в себе дослідження застосунків, які вже використовують схожі можливості, аналіз робіт, розробку архітектури системи, а також її реалізацію за допомогою сучасних технологій розробки.

Розроблено вебзастосунок з можливістю розпізнавання та ідентифікації користувачів банківської системи.

Для роботи обрано мови програмування Python та Java.

Вебзастосунок використовує мікросервісну модель архітектури, яка дозволяє мікросервісам на зазначених мовах, комунікувати між собою. Для роботи із функціями розпізнавання обличчя використано бібліотеки OpenCV та DeepFace. Під час розробки використано середовище розробки PyCharm та IntelliJ IDEA.

Тестування передбачало два етапи: вручну для всієї системи, а також тестування на наборі даних для методів розпізнавання обличчя. Визначено рекомендований поріг схожості облич, який може використовуватися для дозволу авторизації. Під час роботи встановлено, що використання одного методу може вплинути на якість авторизації за Face ID через те, що він буде завищувати чи занижувати схожість. Через це рекомендується використання декількох методів.

Розглянуті подальші перспективи розвитку застосунку, які включають у себе покращення методів розпізнавання обличчя, стабілізація системи, автоматизація тестування, покращення візуальної складової, розширення системи, підвищення захисту даних.

Результати роботи мають значення для розвитку вебзастосунків з використанням авторизації у різних сферах. Результати роботи апробовано у вигляді тез доповіді під час Міжнародного молодіжного форуму «РАДІОЕЛЕКТРОНІКА ТА МОЛОДЬ У XXI СТОЛІТТІ» [30].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Alshingiti, Z., Alaqel, R., Al-Muhtadi, J., Haq, Q. E. U., Saleem, K., & Faheem, M. H. (2023). A deep learning-based phishing detection system using CNN, LSTM, and LSTM-CNN. *Electronics*, *12*(1), 232.

2. Mytnyk, B., Tkachyk, O., Shakhovska, N., Fedushko, S., & Syerov, Y. (2023). Application of artificial intelligence for fraudulent banking operations recognition. *Big Data and Cognitive Computing*, *7*(2), 93.

3. AL-Dosari, K., Fetais, N., & Kucukvar, M. (2024). Artificial intelligence and cyber defense system for banking industry: A qualitative study of AI applications and challenges. *Cybernetics and systems*, *55*(2), 302–330.

4. Sai, G. H., Tyagi, A. K., & Sreenath, N. (2023, January). Biometric security in Internet of Things based system against identity theft attacks. In *2023 International Conference on Computer Communication and Informatics (ICCCI)* (pp. 1–7). IEEE.

5. Hangaragi, S., Singh, T., & Neelima, N. (2023). Face detection and Recognition using Face Mesh and deep neural network. *Procedia Computer Science*, *218*, 741–749.

6. Pomazan V., Tvoroshenko I., and Gorokhovatskyi V. (2023) Handwritten character recognition models based on convolutional neural networks, *International Journal of Academic Engineering Research*, 7(9), pp. 64–72.

7. Tvoroshenko I., Gorokhovatskyi V., Kobylin O., and Tvoroshenko A. (2023) Application of deep learning methods for recognizing and classifying culinary dishes in images, *International Journal of Academic and Applied Research*, 7(9), pp. 57–70.

8. Гороховатський В., Передрій О., Творошенко І., Марков Т. (2023) Матриця відстаней для множини компонентів структурного опису як інструмент для створення класифікатора зображень, *Сучасні інформаційні системи*, 7(1), С. 5–13.

9. Tvoroshenko I., and Gorokhovatskyi V. (2022) The Application of Hybrid Intelligence Systems for Dynamic Data Analysis, *International Journal of Engineering and Information Systems*, 6(2), pp. 40–48.

10. Tvoroshenko I., Pomazan V., Gorokhovatskyi V., and Kobylin O. (2023) Application of video data classification models using convolutional neural networks, *International Journal of Academic and Applied Research*, 7(11), pp. 134–145.

11. Гороховатський В.О., Творошенко І.С., Чмутов Ю.В. (2022) Застосування систем ортогональних функцій для формування простору ознак у методах класифікації зображень, *Сучасні інформаційні системи*, 6(3), С. 5–12.

12. Pomazan, V., Tvoroshenko, I., & Gorokhovatskyi, V. (2023). Development of an application for recognizing emotions using convolutional neural networks, *International Journal of Academic Information Systems Research*, 7(7), pp. 25–36.

13. Gorokhovatskyi, V., Tvoroshenko, I., Kobylin, O., & Vlasenko, N. (2023). Search for visual objects by request in the form of a cluster representation for the structural image description, *Advances in Electrical and Electronic Engineering*, 21(1), pp. 19–27.

14. Гороховатський В., Творошенко І., Сидоренко Д. (2021) Класифікація зображень із використанням кластерного подання, *Міжн. наук. симпозіум Інтелектуальні рішення-С. Обчислювальний інтелект. Теорія прийняття рішень: праці міжн. наук. симп. (Вересень 29, 2021)*. Київ-Ужгород, С. 44–45.

15. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., and Zeghid M. (2022) Cluster representation of the structural description of images for effective classification, *Computers, Materials & Continua*, 73(3), pp. 6069–6084.

16. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., and Al-Dhaifallah M. (2022) Classification of Images Based on a System of Hierarchical Features, *Computers, Materials & Continua*, 72(1), pp. 1785–1797.

17. Gorokhovatskyi V., Tvoroshenko I. (2023) Identification of visual objects by the search request. *International scientific symposium «INTELLIGENT SOLUTIONS-S». Computational intelligence (results, problems and perspectives). Decision making theory: proceedings of the international symposium*, September 28, 2023, Kyiv-Uzhorod, Ukraine, pp. 25–27.

18. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., and Zeghid M. (2022) Tools for fast metric data search in structural methods for image classification, *IEEE Access*, 10, pp. 124738-124746.

19. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., Gadetska S., and Al-Dhaifallah M. (2023) Statistical data analysis models for determining the relevance of structural image descriptions, *IEEE Access*, vol. 11, pp. 126938-126949.

20. Gorokhovatskyi V., Tvoroshenko I., and Yakovleva O. (2024) Transforming image descriptions as a set of descriptors to construct classification features, *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 33, no. 1, pp. 113-125.

21. Selfies and Videos Dataset. URL: https://www.kaggle.com/datasets/tapakah68/selfies-and-video-dataset-4-000-people (дата звернення 12.05.2024).

22. Etim, G. S., Ada, J. A., Eyo, I. E., Ndem, S. E., & James, E. E. (2023). Electronic banking and customers' access to banking services in rural settlements. RES MILITARIS, 13(3), 1161-1177.

23. Gorokhovatskyi V., Tvoroshenko I., Yakovleva O., Hudáková M., and Gorokhovatskyi O. (2024) Application a committee of Kohonen neural networks to training of image classifier based on description of descriptors set, *IEEE Access*, vol. 12, pp. 73376-73385.

24. Srokosz, M., Bobyk, A., Ksiezopolski, B., & Wydra, M. (2023). Machine-Learning-Based Scoring System for Antifraud CISIRTs in Banking Environment. Electronics, 12(1), 251.

25. Khan, H. U., Malik, M. Z., Nazir, S., & Khan, F. (2023). Utilizing bio metric system for enhancing cyber security in banking sector: a systematic analysis. IEEE Access.

26. AL-Aadamy, M. M. J., & AL-Dulaimi, M. K. H. (2023). Enhancement of E-Banking System in Iraq by web application-based authentication system using face recognition. Wasit Journal for Pure Sciences, 2(4).

27. Manonmani, M. S. P., Abirami, G., & Sri, V. N. SPOOF PREVENTION FOR E-BANKING USING LIVE FACE RECOGNITION.

28. Afaneh, M. (2023). New Trends in the Banking Sector and the Development of E-Banking. In Artificial Intelligence and Transforming Digital Marketing (pp. 1107-1115). Cham: Springer Nature Switzerland.

29. Akhisar, I., Tunay, K. B., & Tunay, N. (2015). The effects of innovations on bank performance: The case of electronic banking services. Procedia-Social and Behavioral Sciences, 195, 369-375.

30. Ларін І.П. Аналіз сучасного стану розвитку вебзастосунків розпізнавання та ідентифікації користувачів інформаційних систем. *Радіоелектроніка та молодь у XXI столітті: тези доповідей 28-го Міжнародного молодіжного форуму (Харків, 16–18 квітня 2024 р.). Харків: ХНУРЕ*, 2024. Т. 7. С. 69-70.