Міністерство освіти і науки України
Тернопільський національний педагогічний університет
імені Володимира Гнатюка

Фізико-математичний факультет
Кафедра інформатики та методики її навчання

Кваліфікаційна робота

ПРОЄКТУВАННЯ ТА РОЗРОБЛЕННЯ МОДУЛЯ АВТЕНТИФІКАЦІЇ КОРИСТУВАЧІВ НА ОСНОВІ ТЕХНОЛОГІЇ МАШИННОГО ЗОРУ

Спеціальність 122 Комп'ютерні науки
Освітня програма «Комп'ютерні науки»

Здобувача вищої освіти освітньо-кваліфікаційного рівня «магістр»
Грицая Івана Андрійовича

НАУКОВИЙ КЕРІВНИК:
доктор педагогічних наук, професор
Олексюк Василь Петрович

РЕЦЕНЗЕНТ:
кандидат технічних наук, доцент кафедри комп'ютерних наук Тернопільського національного технічного університету ім. Івана Пулюя
Дмитроца Леся Павлівна

Тернопіль – 2025

АНОТАЦІЯ

Грицай І. А. Проєктування та розроблення модуля автентифікації користувачів на основі технології машинного зору. Кваліфікаційна робота на здобуття освітнього ступеня «магістр» зі спеціальності 122 Комп’ютерні науки. ТНПУ ім. В. Гнатюка. Тернопіль, 2025. 82 с.

У кваліфікаційній роботі розглянуто принципи функціонування біометричних систем автентифікації та сучасні підходи до розпізнавання облич на основі технологій комп’ютерного зору й машинного навчання. Проаналізовано методи обробки зображень і формування векторних ознак облич. Спроєктовано та реалізовано прототип програмного модуля біометричної автентифікації, визначено архітектуру системи, структуру бази даних і механізми взаємодії її складових. Проведено тестування розробленого рішення та оцінено його ефективність у реальних умовах використання.

Ключові слова: біометрична автентифікація, комп’ютерний зір, машинне навчання, розпізнавання облич, інформаційна безпека.

ABSTRACT

Hrytsai I. A. Design and Development of a user authentication module based on machine vision technology. The qualification work for obtaining a master's degree in the specialty Specialty 122 Computer Science.Ternopil Volodymyr Hnatiuk National Pedagogical University. Ternopil, 2025. 82 p.

The thesis examines the principles of biometric user authentication systems and modern approaches to face recognition based on computer vision and machine learning technologies. Image processing methods and facial feature vector generation techniques are analyzed. A prototype of a biometric authentication software module was designed and implemented, including system architecture, database structure, and interaction between system components. The developed solution was tested, and its effectiveness under practical usage conditions was evaluated.

Keywords: biometric authentication, computer vision, machine learning, face recognition, information security.

ЗМІСТ

ВСТУП

Розвиток цифрових технологій, зростання кількості онлайн-сервісів та обсягів персональних даних, що обробляються, висувають нові вимоги до систем інформаційної безпеки. Одним із ключових аспектів захисту інформації є достовірна автентифікація користувачів, яка забезпечує контроль доступу до ресурсів та зменшує ризики порушення цілісності чи конфіденційності даних. Традиційні методи автентифікації, зокрема паролі, графічні ключі чи PIN-коди, демонструють низку недоліків: вони вразливі до перехоплення, підбору, соціальної інженерії та крадіжки облікових даних. Це зумовлює зростання інтересу до більш стійких і технологічних механізмів автентифікації користувачів.

Значну увагу привертають біометричні методи автентифікації, що базуються на унікальних фізіологічних або поведінкових характеристиках людини. Сучасні алгоритми машинного навчання та комп'ютерного зору створюють передумови для формування високоточних систем розпізнавання, здатних автоматично аналізувати візуальні дані та адаптуватися до варіативних умов функціонування. Серед біометричних технологій особливої популярності набирають методи розпізнавання обличчя, які відзначаються зручністю використання, доступністю апаратного забезпечення, високою швидкістю обробки та ефективністю в реальних сценаріях застосування - від мобільних пристроїв до корпоративних платформ контролю доступу.

Проєктування та впровадження таких модулів потребує ґрунтовного розуміння принципів обробки зображень, машинного навчання та сучасних підходів до архітектури програмних систем. Це зумовлює потребу у створенні прототипів, що поєднують у собі актуальні алгоритмічні та інженерні рішення, забезпечуючи можливість їх подальшого впровадження в існуючі інформаційні середовища.

Актуальність даного дослідження визначається необхідністю розроблення надійного, адаптивного та інтелектуального прототипу модуля автентифікації користувачів, здатного гарантувати високий рівень безпеки, оперативність

функціонування та зручність для кінцевого користувача. В епоху розвитку штучного інтелекту важливим є дослідження можливостей застосування алгоритмів глибокого навчання та технологій комп'ютерного зору для підвищення точності та стійкості процесів верифікації персональних даних.

Метою роботи є дослідження теоретичних засад процесів верифікації достовірності користувачів, а також проєктування та програмна реалізація прототипу модуля автентифікації, що використовує технології розпізнавання зображення та машинного навчання.

Для досягнення мети визначено такі **завдання** дослідження:

1. Вивчити засади функціонування технологій біометричної автентифікації та алгоритмів машинного навчання, що застосовуються для розпізнавання зображень користувачів.

2. Спроєктувати модель програмного модуля біометричної автентифікації.

3. Розробити прототип модуля автентифікації на основі технологій комп’ютерного зору та машинного навчання.

4. Провести тестування розробленого прототипу та оцінити його ефективність у реальних умовах.

Об’єкт дослідження – процес автентифікації користувачів у цифрових системах.

Предмет дослідження – алгоритми машинного навчання та методи комп’ютерного зору, що використовуються для розпізнавання зображень користувачів під час автентифікації.

Наукова новизна дослідження полягає у аналізі й систематизації технологій біометричної автентифікації, створенні прототипу модуля автентифікації на основі технологій комп’ютерного зору та машинного навчання.

Практичне значення роботи полягає у розробленні й тестуванні вказаного прототипу.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ЗАСАДИ БІОМЕТРИЧНОЇ АВТЕНТИФІКАЦІЇ ТА ТЕХНОЛОГІЙ КОМП'ЮТЕРНОГО ЗОРУ

1.1 Сучасні підходи до автентифікації користувачів у цифрових системах

Захист персональних даних є одним із ключових завдань сучасних інформаційних систем, оскільки саме він забезпечує довіру користувачів та стабільність цифрового середовища. Автентифікація користувачів - процес підтвердження їхньої особи - традиційно базується на використанні паролів, токенів або двофакторної перевірки. Проте класичні знаннєві фактори, зокрема паролі, які десятиліттями залишалися основним засобом доступу до цифрових систем, дедалі частіше стають слабкою ланкою безпеки. Згідно зі звітом Verizon Data Breach Investigations Report, 74% успішних порушень інформаційної безпеки пов’язані з людським фактором [12],зокрема використанням викрадених облікових даних, помилками або неправомірним використанням привілеїв. Це свідчить про системну вразливість традиційних підходів до автентифікації.

У зв’язку зі зростанням кількості кіберзагроз спостерігається чітка тенденція переходу до багатофакторних (MFA) та біометричних систем [27]. Багатофакторна автентифікація поєднує два й більше незалежних факторів - знання, володіння та біометрію, що суттєво ускладнює можливість несанкціонованого доступу. Найпоширенішими прикладами є комбінації «пароль + одноразовий код», «токен + біометрія». Особливе місце останніми роками займають біометричні методи автентифікації, що базуються на унікальних фізіологічних та поведінкових ознаках людини [6].

Біометричні характеристики складно підробити або передати іншій особі. Вони не можуть бути забуті або випадково розголошені. Саме тому ця група методів забезпечує вищий рівень захисту порівняно з класичними підходами. Дослідження доводять ефективність біометричних рішень: біометрична автентифікація має значно більшу стійкість до атак, оскільки використовує незмінні та унікальні характеристики користувача [2]. Висока точність,

швидкість та безконтактність роблять її одним із домінуючих способів автентифікації у сучасних цифрових системах. За оцінкою ринку Gartner (2024), до 2027 року понад 60% систем доступу використовуватимуть біометричні фактори як основний або додатковий механізм підтвердження особи [15].

Розвиток технологій штучного інтелекту призвів до революції у сфері біометричної автентифікації. Алгоритми машинного навчання, а особливо глибокі нейронні мережі (Deep Learning), дозволили досягти точності, недосяжної традиційними методами комп'ютерного зору. Наприклад, такі відомі моделі, як FaceNet (Google) [24] та ArcFace [4], уже десять років тому демонстрували точність розпізнавання облич понад 99%. Це ставить їх на рівень, близький до людського сприйняття.

Однак одночасно з підвищенням точності алгоритмів виникають нові виклики. Серед найважливіших - захист від спуфінг-атак (пред'явлення фотографій, відео та 3D-масок), а також адверсаріальні атаки, що можуть вводити нейронну мережу в оману за допомогою спеціально модифікованих зображень. Тому сучасні системи біометричної автентифікації передбачають використання додаткових механізмів визначення «природності» (liveness detection) та оцінювання довіри до вхідних даних [31].

Сучасні тенденції вказують на поступовий перехід до passwordless-автентифікації, де ключову роль відіграють криптографічні протоколи, зокрема FIDO2/WebAuthn [32]. Значну увагу приділено і появі динамічної та контекстної автентифікації, яка враховує не лише особу користувача, а й ситуаційні ознаки: середовище, пристрій, ризиковий профіль сесії. У 2023–2024 рр. інтенсивно розвивалися системи risk-based authentication, які автоматично вибирають рівень перевірки залежно від ризику входу [10].

Отже, сучасні підходи до автентифікації користувачів зміщуються у бік більш інтелектуальних, стійких та зручних рішень, які викоистовують біометричні методи, багатофакторні моделі та machine learning–орієнтовані техніки аналізу поведінкових характеристик. Це формує основу для побудови

нових підходів до захисту інформації, де ключову роль відіграють безпечність, адаптивність і стійкість до сучасних кіберзагроз.

1.2 Традиційні та біометричні методи автентифікації користувачів

Автентифікаційні механізми, що застосовуються у цифрових системах, формують основу захисту доступу до ресурсів і персональних даних. Сучасні системи використовують різні підходи до підтвердження особи, проте їх ефективність значною мірою залежить від характеристик даних, на яких ґрунтується автентифікація, а також від способу взаємодії користувача із системою. Традиційно всі методи поділяють на дві великі групи – класичні (традиційні) та біометричні, кожна з яких має власний набір переваг та обмежень.

Традиційні методи автентифікації базуються на даних, які користувач знає або має. До першої групи належать паролі, секретні фрази та PIN-коди. Їх головною перевагою є технологічна простота впровадження та низькі вимоги до апаратної інфраструктури [35]. Проте дослідження з кібербезпеки підтверджують, що саме знаннєві фактори залишаються найбільш уразливими через залежність від людського чинника: користувачі часто обирають слабкі або повторювані паролі, використовують однакові облікові дані на різних платформах, а також стають жертвами фішингу чи інших соціотехнічних атак [3]. Саме це значно знижує надійність знаннєвих методів і вимагає додаткових механізмів контролю. Інші традиційні методи ґрунтується на факторі володіння. Вони передбачають використання фізичного пристрою або цифрового токена, наприклад, апаратних ключів, смарт-карток чи генераторів одноразових паролів (OTP). У порівнянні зі знаннєвими підходами такі методи забезпечують вищу стійкість до компрометації, оскільки потребують фізичної присутності засобу автентифікації. Разом із тим ризики втрати, крадіжки або перехоплення одноразових кодів зберігаються, а експлуатація таких систем вимагає підтримки додаткової інфраструктури [33].

Попри широке застосування, традиційні методи автентифікації часто виявляють недостатню ефективність у складних цифрових середовищах, що

сприяє переходу до більш надійних механізмів підтвердження особи. Біометричні методи, на відміну від класичних, базуються на фізіологічних або поведінкових характеристиках користувача, які не можуть бути передані або забуті. До фізіологічних ознак належать відбитки пальців, форма обличчя, структура райдужної оболонки ока, геометрія кисті. Поведінкові характеристики охоплюють динаміку клавіатурного набору, ходу, голос, звички користування пристроєм тощо [6].

Біометричні технології істотно підвищують рівень довіри до процесу автентифікації, оскільки використовують унікальні властивості людського організму, відносно стабільні у часі та складні для підробки. Саме тому розпізнавання облич, відбитків пальців та голосова автентифікація активно використовуються у мобільних платформах, банківських сервісах тощо. Додаткову надійність забезпечує можливість поєднання біометричних даних з іншими факторами у межах багатофакторної автентифікації (MFA – Multi Factor Authentication)), що підвищує загальний рівень безпеки системи [27].

Водночас точність біометричних методів залежить від якості сенсорів, умов збору даних та стійкості алгоритмів обробки зображень. У разі витоку біометрична інформація не може бути змінена, що потребує застосування спеціальних підходів до шифрування та зберігання даних [31]. Крім того, біометричні системи мають підвищену вразливість до маніпуляцій та підміни облич або інших параметрів,що потребує впровадження механізмів виявлення «природності» (liveness detection). Нище подана порівняльна таблиця 1.1. методів автентифікації з точки зору їх переваг, недоліків, сфер застосування та рівня безпеки.

Таблиця 1.1

Порівняльна характеристика основних методів автентифікації

Метод автентифікації	**Тип даних**	**Переваги**	**Недоліки**	**Рівень безпеки**	**Галузь застосування**

Паролі (knowledge-based)	Те, що знає користувач	Простота використання; низька вартість; не потребує додаткового обладнання	Схильність до вгадування; фішинг; слабкі паролі; потреба в запам’ятовуванні	Низький – середній	Веб-сайти, корпоративні системи, поштові сервіси
PIN-коди	Числовий секрет	Швидкий ввід; легко реалізувати; знайомий користувачам	Легко підглянути; легко підбирається; обмежений простір комбінацій	Низький	Банківські картки, мобільні пристрої
Електронні токени / одноразові паролі (OTP)	Одноразовий код	Висока стійкість до атак; не потребує запам’ятовування	Потребує фізичного пристрою або додатку; можливість крадіжки токена	Середній– високий	Банківські системи, корпоративні VPN
Смарт-картки / криптографічні ключі	Те, що має користувач	Висока криптостійкість; складно підробити	Потребує читача; ризик втрати пристрою	Високий	Державні установи, корпоративні мережі
Автентифікація за допомогою SMS/e-mail	Одноразовий код через канал зв’язку	Простота; не потребує спеціального обладнання	Уразливість SIM-swap; затримки доставки; можливий перехват	Середній	Двофакторна автентифікація в онлайн-сервісах
Біометрія (відбитки пальців)	Фізичні характеристики	Висока унікальність; зручність; швидкість	Можливість підробки слідів; потреба у сенсорах	Високий	Мобільні пристрої, системи доступу
Розпізнавання обличчя	Фізичні характеристики	Безконтактність; швидкість; інтуїтивність	Вразливість до підробок фото/відео без захисту Liveness; залежність від освітлення	Середній– високий	Смартфони, системи контролю доступу
Розпізнавання райдужної оболонки ока	Фізичні характеристики	Дуже висока унікальність і точність	Висока вартість обладнання; дискомфорт для	Високий	Високобезпечні об’єкти, державні системи

			користувача		
Голосова автентифікація	Біометрія голосу	Зручність; безконтактність	Залежність від шуму; можливість підробки аудіо	Середній	Контакт-центри, телеком-сервіси
Аналіз поведінки (behavioral biometrics)	Динамічні патерни (клавіатурний почерк, хода, навігація)	Важко підробити; працює у фоновому режимі	Потребує тривалого збору даних; залежність від контексту	Середній–високий	Мобільні додатки, банківські сервіси

Порівняльний аналіз демонструє, що традиційні методи автентифікації у наш час уже не забезпечують належного рівня безпеки у складних цифрових середовищах. У той час як біометричні технології надають більшої надійності, зручності та захисту від різних типів атак. Саме тому сучасні системи дедалі частіше використовують біометричні фактори як основний або додатковий компонент автентифікації.

1.3 Технології комп’ютерного зору та машинного навчання у біометричній автентифікації

Сучасні методи біометричної автентифікації користувачів ґрунтуються на швидкому розвитку технологій комп’ютерного зору та машинного навчання, які значно підвищують точність і надійність систем розпізнавання облич. Комп’ютерний зір забезпечує автоматичний аналіз та інтерпретацію візуальних даних, тоді як машинне навчання дозволяє моделі самостійно виявляти закономірності, формувати ознаки та приймати рішення про ідентичність користувача. У поєднанні ці технології створюють інтелектуальні системи, які здатні автентифікувати людину за мілісекунди навіть у складних умовах зйомки та при наявності зовнішніх перешкод.

Основою більшості сучасних біометричних систем є процес обробки зображень, що передбачає кілька послідовних етапів: виявлення обличчя у вхідному кадрі, нормалізацію та попередню обробку, виділення ознак за допомогою нейронних мереж, формування векторного представлення

(ембеддингу) особи та порівняння цього вектора з еталонними шаблонами [13]. На кожному з цих етапів застосовуються спеціалізовані алгоритми комп'ютерного зору, які забезпечують точність і стійкість системи до впливу зовнішніх факторів, таких як зміна освітлення, ракурсу, міміки, часткове перекриття обличчя чи наявність аксесуарів.

Однією з ключових технологій у сучасних біометричних системах є глибоке навчання [25]. Нейронні мережі, зокрема згорткові нейронні мережі (Convolutional Neural Networks, CNN), стали стандартом для задач виявлення та розпізнавання облич. Наприклад, моделі *VGGFace, FaceNet, ArcFace* та *ResNet50* [6] забезпечують автоматичне виділення високорівневих ознак, що зумовлюють здатність системи розрізняти мільйони різних осіб навіть за мінімальних відмінностей у зовнішності. Ці моделі навчаються на масштабних наборах даних, що налічують мільйони зображень, і тому здатні узагальнювати інформацію та працювати стабільно у реальних умовах. Формування ембедингів – компактних векторних описів особи – дозволяє виконувати верифікацію шляхом обчислення косинусної подібності або евклідової відстані, що значно пришвидшує процес автентифікації [8].

Окремої уваги потребують технології Presentation Attack Detection (PAD), спрямовані на виявлення спроб підробки обличчя за допомогою фотографій, відео чи 3D-масок. PAD-моделі аналізують такі ознаки, як глибина сцени, відбиття світла, мікрорухи обличчя, текстурні характеристики шкіри. Сучасні системи використовують мультимодальні дані – RGB, інфрачервоні (IR) та глибокі (depth) зображення – що забезпечує високий рівень захисту від спуфінг-атак. Використання нейронних мереж у PAD-процесі дозволяє системі самостійно виявляти нехарактерні патерни та аномалії, які притаманні підробленим зображенням [11].

У біометричній автентифікації все частіше використовуються трансформерні архітектури (Vision Transformers, Swin Transformer), які дозволяють обробляти зображення не через фільтри згортки, а через механізм самоуваги [21]. Це підвищує стійкість до варіацій у зовнішності та значно

покращує якість порівняння ознак. Такі моделі часто поєднуються з CNN у гібридних архітектурах, що забезпечує баланс між високою продуктивністю та точністю.

Важливою тенденцією є використання оновлюваних моделей та персоналізованого навчання, які дозволяють підлаштовувати ембеддинги під конкретного користувача, підвищуючи точність системи при мінімальних змінах зовнішності. Також набирають популярності методи федеративного навчання [34], які дають змогу навчати моделі без передачі біометричних даних на центральний сервер, що підвищує рівень приватності користувачів.

Ще один ключовий аспект - оптимізація моделей для мобільних та вбудованих систем. Використовуються методи квантизації параметрів, стиснення нейронних мереж, апаратного прискорення (Neural Processing Units), що робить біометричну автентифікацію можливою в реальному часі на недорогих пристроях [10].

Отже, технології комп’ютерного зору та машинного навчання є фундаментом сучасних систем біометричної автентифікації. Вони забезпечують точне та швидке розпізнавання облич, стійкість до атак, адаптивність до зовнішніх умов та можливість масштабування у різних цифрових середовищах. Сукупність цих технологій дозволяє створювати надійні, зручні й ефективні модулі автентифікації, що відповідають актуальним вимогам безпеки та є ключовими компонентами сучасних інформаційних систем

1.4 Алгоритми розпізнавання облич

Ефективність біометричної автентифікації значною мірою залежить від алгоритмів виявлення обличчя, що забезпечують виділення області зображення, придатної для подальшої обробки. У традиційних системах розпізнавання облич застосовувалися класичні алгоритми, що працюють на основі лінійних перетворень та ознак, отриманих вручну. Одним із перших і найбільш відомих є метод головних компонент (PCA), який дозволяє зменшувати розмір зображень облич і представляти їх у вигляді набору так званих «eigenfaces»[20]. Проте цей підхід виявився надто чутливим до змін освітлення, положення голови та

мімічних варіацій. Модифікацією PCA став метод лінійного дискримінантного аналізу (LDA). Його головна ідея полягає в тому, щоб знайти таке лінійне перетворення простору ознак, яке робить об’єкти різних класів більш віддаленими один від одного, а об’єкти одного класу – більш близькими [23]. Однак LDA добре працює лише тоді, коли зображення різних об'єктів отримані в максимально схожих умовах. Мали успіх алгоритми, що використовували локальні бінарні шаблони (LBP). Вони виділяли текстурні ознаки обличчя шляхом порівняння сусідніх пікселів і відзначалися кращою стійкістю до змін освітлення, це дозволило їм знайти широке використання у системах реального часу. Класичні методи в цілому не могли повноцінно моделювати нелінійні залежності, які властиві зображенням облич, і втрачали ефективність у випадку зміни у зовнішньому вигляді або якості зображення.

Використання глибокого навчання змінило точність і принципи роботи систем розпізнавання облич . Згорткові нейронні мережі (CNN) дозволили відмовитися від ручного конструювання ознак і перейти до автоматичного витягнення багаторівневих дескрипторів, що значно краще відображають структуру візуальних даних [25]. Однією з перших масштабних моделей стала VGG-Face, яка навчалася на мільйонах зображень і продемонструвала близьку до людської точність у задачах верифікації. Наступний прорив відбувся після появи архітектури FaceNet, яка запровадила триплетну функцію втрат. Навчання на основі порівняння трьох зображень – «опорного», позитивного та негативного – дозволило формувати компактні векторні представлення, у яких відстань між векторами відображає ступінь подібності облич. Такий підхід забезпечив новий рівень універсальності та точності, що сприяло швидкому поширенню FaceNet у біометричних системах різного типу.

Подальший розвиток глибоких алгоритмів зосередився на оптимізації метрик простору ознак. Моделі ArcFace, CosFace та SphereFace використовують спеціальні математичні прийоми, які збільшують різницю між ознаками різних людей. Завдяки цьому система легше розрізняє обличчя та працює точніше [4].

Завдяки цим моделям сучасні системи розпізнавання облич досягають точності понад 99,8 % на стандартних наборах даних. Важливою складовою систем розпізнавання є етап детекції та попередньої обробки зображення. Алгоритми MTCNN або RetinaFace забезпечують точне знаходження облич та визначення ключових точок, що дозволяє виконати геометричне вирівнювання, зменшити вплив поворотів голови й забезпечити стабільність подальшої обробки [1]. Без цього етапу навіть найточніші глибокі моделі можуть втрачати ефективність.

Значний прогрес у галузі став можливим завдяки використанню великих анотованих наборів даних, таких як LFW, VGGFace2 або MS-Celeb-1M. Саме вони забезпечили базу для навчання моделей з мільйонами параметрів. Крім того, оцінювання якості алгоритмів проводиться за допомогою метрик FAR, FRR, EER та ROC-кривих, що дозволяє об’єктивно порівнювати різні підходи та аналізувати їх придатність у реальних умовах [21].

Попри високий рівень розвитку, сучасні системи розпізнавання облич стикаються з низкою викликів. Одним із ключових є загроза атак із використанням підроблених зображень або відео (спуфінг). Для протидії цьому впроваджуються алгоритми оцінювання «природності» користувача, які аналізують мікрорухи, структуру шкіри, глибину сцени або часові зміни у відеопотоці. Іншою проблемою залишаються етичні та правові аспекти зберігання біометричних даних, які потребують розроблення відповідних механізмів захисту та нормативного регулювання [2].

Підсумовуючи, можна зазначити, що перехід від класичних методів до глибоких нейронних мереж кардинально змінив можливості систем розпізнавання облич. Сучасні алгоритми забезпечують високу точність у різноманітних умовах, демонструють стійкість до зовнішніх факторів і здатні працювати в режимі реального часу. Це робить їх оптимальною основою для впровадження в сучасні модулі біометричної автентифікації, які потребують надійності, швидкодії та здатності до масштабування. Технології глибокого навчання постійно вдосконалюються, відкриваючи нові перспективи для

створення інтелектуальних систем контролю доступу та підвищення рівня інформаційної безпеки.

1.5 Технології комп’ютерного зору для верифікації користувачів

Комп’ютерний зір у системах верифікації користувачів охоплює комплекс технологій, які забезпечують перетворення візуальної інформації з камер або інших сенсорів у структуровані дані, придатні для аналізу й прийняття рішень щодо автентичності особи. На відміну від загальних алгоритмів розпізнавання, для систем автентифікації важливими є не лише точність визначення облич, а й можливість стабільної роботи у реальному середовищі, де присутні змінне освітлення, рух, фон і варіативність пози користувача [20].

Одним із ключових аспектів технологій комп’ютерного зору є процедури захоплення та нормалізації зображення, спрямовані на отримання якісного вхідного сигналу. На цьому етапі використовуються методи автоматичного визначення експозиції, балансування білого, компенсації шумів та стабілізації відеопотоку. Стабільність вхідних даних безпосередньо впливає на точність подальших операцій, а отже, визначає надійність усієї системи верифікації.

Не менш важливою складовою є геометрична та фотометрична нормалізація обличчя, що передбачає корекцію перспективи, симетризацію зображення та приведення його до стандартного формату [11]. Застосування таких технологій дає змогу усунути вплив зовнішніх факторів і підготувати зображення до подальшого аналізу. Нормалізація передбачає обробку ключових точок, фіксацію позиції очей, центрів рота та інших орієнтирів, що дозволяє стандартизувати геометрію обличчя навіть у разі неідеального ракурсу.

Для систем верифікації особливе значення має аналіз якості кадру, який оцінює придатність зображення для біометричної обробки. Механізми контролю якості визначають чіткість, рівень контрасту, наявність артефактів руху та ступінь закриття обличчя (окулярами, масками, волоссям). Якщо зображення не відповідає встановленим критеріям, система може запросити повторний кадр, тим самим мінімізуючи ризик помилок у процесі автентифікації.

Окрему категорію становлять технології детекції та відсіювання підробок. Вони спрямовані на протидію атакам із використанням фотографій, відео, масок чи цифрових підробок [31]. На цьому рівні застосовуються методи аналізу текстур, виявлення глибини сцени, оцінювання мікрорухів або аналіз відблисків та тіней. Пристрої з інфрачервоними або 3D-сенсорами забезпечують додаткову інформацію про об'ємну структуру обличчя, що унеможливлює використання двовимірних підробок.

Важливе місце у комп'ютерному зорі для верифікації відіграє технологічна інфраструктура, яка забезпечує обробку та передавання візуальних даних. У сучасних системах застосовуються два основні підходи – локальна (on-device) обробка та хмарна обробка. Локальна обробка дає змогу виконувати аналіз без надсилання даних на сервер, що підвищує конфіденційність і швидкість роботи, тоді як хмарні сервіси забезпечують високу обчислювальну потужність і можливість використання складніших моделей. Вибір підходу залежить від вимог до безпеки, затримок та доступності ресурсів.

Використання комп'ютерного зору у верифікації потребує не лише інтелектуальних алгоритмів, а й ефективної інтеграції з програмними системами, у яких він функціонує. Тому важливими є стандартизовані бібліотеки та фреймворки, такі як OpenCV, MediaPipe, ONNX Runtime або TensorRT, що забезпечують апаратне прискорення обробки та оптимізацію моделей для різних пристроїв. Завдяки їм системи автентифікації можуть працювати у реальному часі, забезпечуючи низькі затримки й високу пропускну здатність навіть на недорогих камерах або мобільних пристроях.

Окрему увагу привертають технології безпечного зберігання та передавання біометричних даних, що є важливим компонентом комп'ютерного зору в контексті автентифікації. Застосовуються методи шифрування, токенізації, генерації біометричних шаблонів та їхнього перетворення у формат, не придатний для зворотного відтворення [21]. Це дозволяє мінімізувати ризики витоку персональних даних та зменшити вплив потенційних атак.

Отже, технології комп’ютерного зору у верифікації користувачів охоплюють широкий спектр інструментів та процедур - від захоплення й нормалізації зображень до перевірки їхньої автентичності, оптимізації обробки й забезпечення безпеки біометричних шаблонів. Їхній комплексний характер робить комп’ютерний зір невід’ємною частиною сучасних біометричних систем, які повинні працювати швидко, точно і надійно в умовах реального середовища.

1.6 Огляд сучасних систем і прототипів біометричної автентифікації на основі розпізнавання облич

Сучасні системи біометричної автентифікації, що базуються на технології розпізнавання облич, відіграють важливу роль у забезпеченні безпеки доступу до цифрових сервісів, мобільних пристроїв і інформаційних систем різного рівня критичності. Розвиток алгоритмів комп’ютерного зору, глибокого навчання та апаратних засобів призвів до появи рішень, здатних забезпечувати високу точність розпізнавання навіть у складних умовах, таких як низька якість зображення, динамічні сцени або презентаційні атаки. Попри значну варіативність доступних технологій, більшість сучасних систем побудовані на поєднанні CNN- або трансформерних моделей, модулів попередньої обробки та захищених механізмів зберігання шаблонів.

Одним із найвідоміших прикладів є Apple Face ID [1], що використовує проєкцію структурованого світла та інфрачервоне сканування для формування тривимірної карти обличчя. Комплексна архітектура системи мінімізує ризики спуфінгу та забезпечує високий рівень точності. Принциповою особливістю Face ID є локальне зберігання біометричних шаблонів у модулі Secure Enclave, що унеможливлює їх витік та використання сторонніми особами.

Іншим поширеним рішенням є Microsoft Windows Hello [26], де автентифікація на основі обличчя реалізується за допомогою IR-камери та глибоких моделей розпізнавання, оптимізованих під роботу в операційній системі Windows 10/11. Система підтримує стандарти FIDO2 і використовується для passwordless-автентифікації як у персональних, так і в корпоративних

середовищах. На відміну від низки хмарних сервісів, Windows Hello так само дотримується концепції локальної обробки, що зменшує ризики витоку даних.

Хмарні сервіси, такі як Amazon Rekognition та Microsoft Azure Face API, пропонують інструменти для масштабного розпізнавання облич у потоковому відео чи великих базах даних. Завдяки використанню інфраструктури AWS та Azure ці системи легко інтегруються у корпоративні рішення для контролю доступу, аналітики поведінки та моніторингу. Основними перевагами таких систем є висока масштабованість, можливість автоматичного оновлення моделей та широкі API-можливості. Водночас їх використання супроводжується важливими питаннями приватності, оскільки шаблони облич часто зберігаються у хмарних середовищах.

Окреме місце займають рішення, орієнтовані на масові комерційні та державні системи відеоспостереження, такі як Face++ (Megvii). Ця система поєднує алгоритми вирівнювання, виявлення ознак, 2D- і 3D-реконструкцію облич та модулі PAD. Вона використовується у сфері банківських операцій, кібербезпеки та гаджетів на ринку Азійсько-Тихоокеанського регіону. Face++ демонструє високі результати в міжнародних тестах NIST FRVT, що підтверджує її ефективність при обробці великих масивів облич.

Важливим напрямом є відкриті системи та дослідницькі прототипи. Зокрема, InsightFace, який включає ArcFace, CosFace, RetinaFace та інші компоненти, є одним із провідних open-source проєктів у сфері розпізнавання облич [17]. Завдяки доступності коду та можливості адаптації моделей під конкретні задачі InsightFace широко використовується в академічних дослідженнях та у створенні кастомних модулів автентифікації. Інші відкриті рішення, такі як OpenFace або системи на основі MediaPipe, орієнтовані на легковагові застосунки та мобільні системи.

Таблиця 1.2.

Порівняльна характеристика сучасних систем біометричної автентифікації на основі розпізнавання облич

Система / Прототип	Тип даних	Архітектура / Технологія	Місце зберігання шаблонів	Стійкість до спуфінгу	Особливості
Apple Face ID	3D + IR	Структуроване світло, Neural Engine	Secure Enclave (локально)	Дуже висока	Тривимірна карта обличчя, локальна обробка
Windows Hello	2D + IR	CNN + IR depth-sensing	TPM / локально	Висока	Підтримка FIDO2, passwordless
Amazon Rekognition	2D/відео	Хмарні CNN-моделі	Хмарні сховища AWS	Середня	Пошук у великих базах, відеоаналітика
Azure Face API	2D	CNN + хмарна інфраструктура	Azure Cloud	Середня	Інтеграція з корпоративними сервісами
Face++ (Megvii)	2D/3D	Вирівнювання + CNN + PAD	Хмарно / локально	Висока	Масштабованість, використання в державних системах
InsightFace (ArcFace)	2D	ArcFace, RetinaFace	Локально / серверно	Залежить від імплементації	Найточніша open-source модель
OpenFace	2D	Triplet-loss CNN	Локально	Низька–середня	Легковагова, дослідницькі застосування
MediaPipe Face Recognition	2D	Lightweight CNN	Локально	Середня	Мобільні пристрої, low-resource середовища

Загалом сучасні системи розпізнавання облич демонструють значний прогрес у підвищенні точності, швидкодії та захищеності, однак залишаються актуальними питання боротьби зі спуфінгом (табл.1.2), забезпечення приватності та мінімізації демографічних упереджень.

Висновки до першого розділу

У першому розділі проаналізовано теоретичні підходи до автентифікації користувачів у цифрових системах і обґрунтовано доцільність переходу від традиційних методів підтвердження особи до біометричних і багатофакторних рішень в умовах зростання кіберзагроз.

Показано, що біометрична автентифікація, зокрема на основі розпізнавання облич, забезпечує підвищену зручність і потенційно вищий рівень безпеки завдяки використанню унікальних фізіологічних і поведінкових

характеристик користувача. Водночас виокремлено специфічні ризики таких систем, пов’язані з якістю вхідних даних, незворотністю наслідків у разі компрометації біометричних шаблонів та вразливістю до презентаційних і адверсаріальних атак, що зумовлює потребу в механізмах контролю якості та перевірки «природності».

Визначено, що сучасні системи розпізнавання облич ґрунтуються на поєднанні методів комп’ютерного зору та машинного навчання, які реалізують послідовність етапів обробки: детекцію, нормалізацію, виділення ознак, формування векторних представлень і порівняння за метриками подібності. Наголошено, що застосування глибоких нейронних мереж істотно підвищує точність, однак ефективність системи залежить від коректної попередньої обробки та узгодженої організації всіх компонентів.

Обґрунтовано, що результативність біометричних рішень визначається не лише якістю моделей, а й інженерними, організаційними та правовими чинниками, зокрема стійкістю до варіативних умов знімання, захищеним зберіганням і передаванням даних та дотриманням принципів захисту персональної інформації. Сформовані положення створюють наукове підґрунтя для подальшого проєктування та реалізації прототипу модуля біометричної автентифікації, орієнтованого на точність, стійкість до атак і відповідність сучасним вимогам інформаційної безпеки.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ТА ТЕХНОЛОГІЧНЕ ЗАБЕЗПЕЧЕННЯ МОДУЛЯ БІОМЕТРИЧНОЇ АВТЕНТИФІКАЦІЇ

2.1 Основні підходи до проєктування прототипу модуля біометричної автентифікації

Проєктування модуля біометричної автентифікації користувачів ґрунтується на поєднанні принципів інформаційної безпеки, вимог до надійності та точності розпізнавання, а також сучасних підходів до обробки візуальних даних. У центрі такого підходу лежить ідея використання зображення обличчя як унікальної біометричної характеристики, яку важко підробити, що дозволяє надійно підтвердити особу користувача. Це забезпечує вищий рівень довіри порівняно з традиційними методами автентифікації на основі паролів або токенів. Сучасні стандарти розроблення інформаційних систем визначають, що архітектура модулів автентифікації повинна відповідати принципам конфіденційності, цілісності та доступності (CIA-triad), доповнюючись вимогами до стійкості алгоритмів та зменшення впливу людського чинника [24].

При проєктуванні прототипу модуля біометричної автентифікації основну увагу звертаємо на здатність системи до однозначного встановлення відповідності між вхідним зображенням користувача та збереженим біометричним шаблоном, а також забезпечення коректності процесів ідентифікації й верифікації. Ці процедури становлять основні механізми автентифікації: ідентифікація забезпечує визначення особи серед множини зареєстрованих користувачів, тоді як верифікація підтверджує, що надане зображення належить конкретному користувачу. Ідентифікація і верифікація є невід’ємними складовими процесу автентифікації, що забезпечують її повноту та надійність, незалежно від того, який із цих двох сценаріїв буде пріоритетним на етапі реалізації.

Проєктування модуля має спиратися на такі концептуальні засади:

1. Орієнтація на точність та стійкість біометричної ознаки. Біометричні характеристики користувача піддаються впливу освітлення, поворотів голови,

міміки та часткових перекриттів. Тому в основу проєктування мають бути покладені технології, здатні компенсувати такі варіації, забезпечуючи стабільність роботи у реальних умовах експлуатації. Сучасні дослідження показують, що підвищення точності систем розпізнавання досягається не лише за рахунок попередньої обробки зображень, а й завдяки комплексному використанню таких механізмів, як геометричне вирівнювання обличчя, нормалізація тональних та колірних характеристик, генерація векторних ознак із використанням спеціалізованих архітектур глибокого навчання (наприклад, ArcFace, FaceNet, MobileFaceNet) та оптимізація метрик порівняння векторних ознак. Саме точність формування та стійкість цих векторних репрезентацій визначають здатність системи коректно розрізняти схожі обличчя та мінімізувати ймовірність помилкової ідентифікації або верифікації [6].

2. Адаптивність та масштабованість програмної архітектури. Система має підтримувати розширення функціоналу, інтеграцію нових моделей машинного навчання, зміну алгоритмів або оптимізацію компонентів без втрати сумісності. Модульна структура застосунку та чітке розмежування функціональності між компонентами забезпечують можливість незалежного оновлення клієнтської, серверної та ML-частин. Подібну архітектурну гнучкість рекомендують сучасні підходи до побудови безпечних систем автентифікації [5].

3. Мінімізація збережених даних і захист біометричних шаблонів. Оскільки біометричні дані є незмінними та особливо чутливими, система біометричної автентифікації повинна зберігати лише векторизовані шаблони, які не дозволяють відтворити первинне зображення. Застосування процедур шифрування, контроль доступу та аудит операцій є базовими вимогами сучасних протоколів безпеки для біометричних систем [33].

4. Надійність і стабільність роботи алгоритмів автентифікації. Концепція проєктування передбачає, що система має коректно працювати не лише за ідеальних умов, але й за наявності шумів, слабкого освітлення, технічних обмежень веб-камери. Для цього на концептуальному рівні визначається

механізм контролю якості кадрів, повторного захоплення зображення та використання порогових значень достовірності.

5. Узгодженість з інтерфейсними та серверними компонентами.Оскільки модуль автентифікації взаємодіє з користувачем через веб-інтерфейс, його концептуальна модель повинна забезпечувати: сумісність із браузерами та можливість доступу до камери; асинхронну обробку запитів; клієнтську валідацію якості зображення до відправлення на сервер; логічне розмежування: інтерфейс користувача - збір даних, бекенд; обробка запитів; виконання верифікації.

6. Використання стандартизованого процесу проєктування. Проєктування повинно базуватися на життєвому циклі ML-систем впродовж таких етапів: визначення вимог, збирання даних, моделювання, оцінювання точності, інтеграція, тестування та моніторинг. Численні дослідження вказують, що системи комп'ютерного зору потребують постійного донавчання моделей та стабільного контролю продуктивності [7].

На рис.2.1 подано основні етапи розробки прототипу модуля від вибору алгоритму до тестування

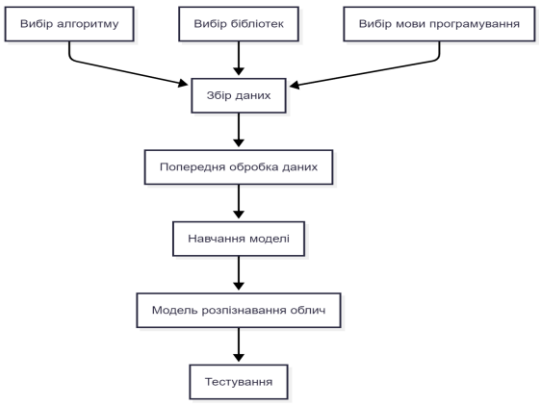

Рис.2.1. Етапи розробки модуля

Отже, концептуальні засади проєктування прототипу модуля біометричної автентифікації визначають загальний напрям формування архітектури, структурних компонентів та алгоритмічної логіки. Вони забезпечують основу для подальшого вибору технологій, способів взаємодії компонентів, побудови модуля обробки зображень та реалізації механізмів безпечної й точної верифікації користувачів.

2.2 Модель даних та структура бази даних MySQL

Проєктування модуля біометричної автентифікації потребує формування чіткої, структурованої та безпечної моделі даних, яка забезпечує коректне збереження, обробку та використання біометричної інформації користувачів [29]. Оскільки верифікація на основі машинного зору передбачає роботу з векторами ознак обличчя, історією автентифікацій та метаданими. До таких метаданих належать службові та контекстуальні параметри, що описують умови проходження автентифікації та характеристики біометричного шаблону, зокрема дата й час створення та останнього оновлення запису, на основі яких система ухвалює позитивний чи негативний результат автентифікації. Така деталізація дозволяє не лише підвищити точність розпізнавання, але й забезпечити можливість контролю за безпекою даних упродовж життєвого циклу системи. Обрана архітектура бази даних має відповідати вимогам масштабованості, швидкодії та надійності. Враховуючи ці потреби, для реалізації збереження даних нами використовується реляційна система керування базами даних MySQL, що відзначається стабільністю, розвиненою підтримкою транзакцій, широкою сумісністю та можливістю оптимізованого зберігання великих структурованих записів.

Розглянемо сутності MySQL та їх призначення. У структурі MySQL передбачено три основні сутності, кожна з яких виконує окрему функціональну роль у процесі автентифікації.

1. Сутність users. Сутність зберігає облікові та основні дані користувача, необхідні для початкової ідентифікації у системі керування доступом. Поля цієї сутності зображені у вигляді таблиці 2.1.

Таблиця 2.1

Поля сутності *users*

Поле	Тип	Призначення
id	INT, PK, AUTO_INCREMENT	Унікальний ідентифікатор користувача
name	VARCHAR	Ім’я або псевдонім
email	VARCHAR, UNIQUE	Прив’язка до облікового запису

created_at	DATETIME	Дата реєстрації
status	ENUM(active, disabled)	Стан доступу

2. Сутність biometric_templates. Містить у собі найважливіший елемент - вектор ознак обличчя, який формується Python-модулем за допомогою алгоритмів комп'ютерного зору (MediaPipe, OpenCV) та моделей кодування облич (наприклад, ResNet-based embeddings) (табл.2.2).

Таблиця **2.2**

Поля сутності *biometric_templates*

Поле	Тип	Призначення
id	INT, PK	Ідентифікатор шаблону
user_id	INT, FK	Посилання на користувача
face_vector	JSON / BLOB	Збереження 128- або 512-вимірного вектора ознак
created_at	DATETIME	Дата створення шаблону
quality_score	FLOAT	Оцінка якості зображення (за потреби)

Зберігання вектора у форматі JSON дає можливість легко індексувати та швидко передавати масиви чисел. Формат BLOB підходить, якщо потрібно використовувати власний бінарний формат.

3. Сутність *auth_logs.* Виконує функцію журналювання подій автентифікації. Його поля подано у таблиці 2.3

Таблиця 2.3.

Поля сутності *auth_logs*

Поле	Тип	Призначення
id	INT, PK	Ідентифікатор запису журналу
user_id	INT, FK	Хто проходив автентифікацію
timestamp	DATETIME	Час події
similarity_score	FLOAT	Обчислена схожість між вектором системи і новим вектором
result	ENUM(success, fail)	Підсумок автентифікації

client_ip	VARCHAR	IP-адреса запиту
device_info	VARCHAR	Інформація про пристрій

2.3 Логічна модель та зв’язки (ER-модель)

Модель «сутність–зв’язок» (Entity–Relationship Model, ER-модель) є однією з базових концептуальних моделей даних, яка використовується для формального опису структури інформаційної системи, її об’єктів та логічних взаємозв’язків між ними. Її основна мета – створення абстрактного представлення предметної області, яке дозволяє систематизувати об’єкти, визначити їх властивості та встановити зв’язки між ними перед фізичним моделюванням [10].

У межах розроблення інформаційних систем ER-модель виконує функцію концептуального рівня опису даних. Вона дає змогу проєктувальникам та розробникам узгодити логіку зберігання інформації, визначити повноту набору сутностей, мінімізувати дублювання даних і забезпечити цілісність системи. На відміну від фізичних моделей, які описують конкретну структуру таблиць у базі даних, ER-модель зосереджена на логічних взаємозв’язках, що робить її універсальною та придатною для використання як у реляційних, так і в нереляційних системах управління базами даних.

Необхідність використання ER-моделі зумовлена складністю сучасних інформаційних систем та вимогами до їх гнучкості, масштабованості й безпеки. Вона забезпечує можливість візуалізації структури даних, що спрощує комунікацію між розробниками, аналітиками та дослідниками, дозволяє формалізувати вимоги до системи та забезпечити узгодженість між програмними компонентами. Крім того, ER-модель виступає основою для подальшого проєктування фізичної бази даних, визначення типів зв’язків, механізмів запитів і оптимізації доступу до інформаційних ресурсів. , логічна модель даних є ключовим елементом методології розроблення складних програмних систем, що забезпечує коректне та передбачуване функціонування підсистем, пов’язаних із зберіганням та обробкою даних.

Між основними сутностями встановлюється наступна система зв’язків

- users – biometric_templates: тип зв’язку один-до-одного, оскільки для кожного користувача формується та підтримується лише один актуальний біометричний шаблон. У разі розширення системи та впровадження механізмів адаптації до змінних умов середовища цей зв’язок може бути трансформовано в модель один-до-багатьох, що дозволить зберігати декілька версій шаблонів для різних умов освітлення, ракурсів або пристроїв захоплення зображення.
- users – auth_logs: зв’язок один-до-багатьох, оскільки один користувач може ініціювати процес автентифікації необмежену кількість разів, а кожна спроба входу фіксується як окремий запис у журналі подій. Така структура забезпечує можливість детального аналізу історії автентифікацій, виявлення аномалій, спроб несанкціонованого доступу та формування аналітичних звітів щодо поведінки користувачів і роботи системи загалом.
- biometric_templates – auth_logs: опційний зв’язок, який використовується у випадках, коли під час автентифікації система зберігає інформацію про конкретний біометричний шаблон, що застосовувався для порівняння. Такий зв’язок не є обов’язковим, оскільки лог події може містити лише результат верифікації без прямої прив’язки до шаблону. Однак у системах, що передбачають аналіз еволюції шаблонів, аудит алгоритмічних рішень або оцінювання точності моделі, фіксація використаного шаблону дозволяє забезпечити прозорість процесів, відтворюваність експериментів та можливість виявлення дефектів у вже протестованих модулях додатка.

2.4 Формати збереження векторів облич

Вектор ознак – це числове представлення обличчя у багатовимірному просторі. У практиці зазвичай використовуються такі формати: JSON (масив цілих чисел) - зручний для читання і використання Python → PHP → MySQL; BLOB (бінарний набір даних) - оптимальний для швидкісного доступу та економії пам’яті; FLOAT ARRAY (масив чисел з плаваючою комою) -формат, у якому зберігаються вектор зображення обличчя, необхідні для верифікації користувачів, застосовується через окрему таблицю, але рідше через складність реалізації. Найбільш оптимальним для веборієнтованої системи є JSON, оскільки

він забезпечує прозорість, простоту перенесення між сервісами та інтеграцію з API.

Забезпечення цілісності та безпеки даних у модулі біометричної автентифікації є важливою умовою його надійного функціонування, оскільки система працює з особливо чутливою інформацією, яка не може бути змінена або відкликана користувачем, на відміну від традиційних паролів [33]. Будь-яке порушення цілісності даних, спотворення біометричних шаблонів або несанкціонований доступ до них може спричинити незворотні наслідки, включно з компрометацією особистості користувача та підміною автентифікаційних даних.

Досягнення цілісності даних забезпечується через сувору регламентацію механізмів запису, оновлення та видалення інформації у базі даних. На кожному етапі роботи з біометричним шаблоном здійснюється контроль цілісності та коректності даних, що передбачає перевірку структури записів, узгодженість форматів. Це унеможливлює появу частково збережених або логічно некоректних даних у сховищі. Для додаткового захисту застосовуються механізми контролю змін – хеш-функції. Вони дозволяють оперативно виявляти будь-які несанкціоновані модифікації та гарантують, що біометричні шаблони залишаються незмінними як під час зберігання, так і в процесі передавання між компонентами системи [31].

Безпека даних у системі розглядається комплексно та включає декілька взаємопов’язаних аспектів. По-перше, біометричні шаблони зберігаються у формі, яка не дозволяє відновити первинне зображення обличчя, що мінімізує ризик приватності. По-друге, канали передавання даних захищаються криптографічними протоколами, що унеможливлює перехоплення або підміну інформації під час комунікації між клієнтськими та серверними компонентами системи. По-третє, доступ до інформаційних ресурсів жорстко обмежується завдяки системі ролей, мультифакторним політикам доступу та журналу аудиту, який дозволяє відстежувати всі операції з даними [3].

Отже, інтегрований підхід до забезпечення цілісності й безпеки даних не лише гарантує надійність процесів автентифікації, але й формує довіру користувачів до системи, сприяє відповідності сучасним стандартам інформаційної безпеки та створює фундамент для подальшого масштабування й розширення функціональності модуля.

Модель даних повинна гарантувати коректність і захищеність біометричних шаблонів. Вбачаємо такі підходи забезпечення цих вимог:

1. Зовнішні ключі використовуються для підтримання структурної цілісності між таблицями модуля біометричної автентифікації. Основними є три основні зв'язки. Перша – зовнішній ключ у таблиці biometric_templates, що посилається на users і гарантує належність кожного шаблону конкретному користувачеві. Друга – зовнішній ключ у таблиці auth_logs, який пов'язує кожен запис автентифікації з відповідним користувачем, забезпечуючи повну історію входів. Третя – опційний зовнішній ключ в auth_logs, що може посилатися на biometric_templates, коли система фіксує шаблон, використаний під час порівняння. Такі зв'язки гарантують логічну узгодженість усіх даних у системі.

2. Шифрування чутливої інформації здійснюється за допомогою алгоритму AES або вбудованих механізмів MySQL Encryption Functions(функцій шифрування), що гарантує захист даних навіть у разі несанкціонованого доступу до сховища. Шифруванню підлягають поля, які містять критичні ідентифікаційні та службові відомості, зокрема у таблиці users шифруються email, а в таблиці biometric_templates – біометричний шаблон (вектор ознак), його хеш та допоміжні параметри, пов'язані з ідентифікацією користувача. Це унеможливлює відтворення або підміну персональних та біометричних даних, зберігаючи їх конфіденційність у межах системи.

3. Обмеження доступу реалізується через розмежування ролей у базі даних. Роль Python-модуля має мінімально необхідні дозволи: читання та запис біометричних шаблонів, додавання нових записів у журнал автентифікацій і доступ лише до технічних полів, потрібних для роботи алгоритмів. Адміністратор не має права змінювати або переглядати біометричні шаблони, а

отримує доступ лише до журналу автентифікацій для моніторингу роботи системи та аналізу інцидентів. Такий підхід мінімізує ризики несанкціонованого доступу й забезпечує принцип найменших привілеїв.

4. Регулярне резервне копіювання бази даних.

5. Хешування застосовується виключно до технічних метаданих, а не до векторів ознак обличчя. Біометричні вектори повинні залишатися у «сирому» числовому вигляді, оскільки саме вони використовуються для обчислення евклідової відстані та прийняття рішення про автентифікацію. Хешуються дані, що не впливають на роботу алгоритму розпізнавання, але є важливими для цілісності систем(електронна пошта користувача). Хешування пошти необхідне для того, щоб уникнути прямої ідентифікації особи у разі компрометації бази даних, зберігаючи можливість порівняння та пошуку записів без розкриття реальної адреси. Це унеможливлює підробку метаданих, дозволяє перевіряти достовірність запитів і суттєво знижує ризики витоку конфіденційної інформації.

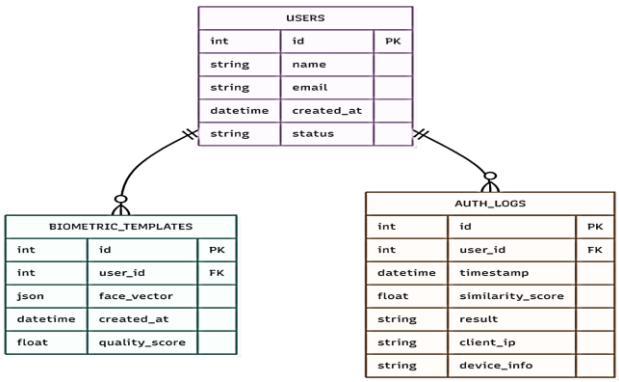

Рис.2.2. Графічне подання моделі даних

Модель даних (рис.2.2), реалізована на основі MySQL, забезпечує структуроване та захищене зберігання інформації, необхідної для повноцінної роботи біометричного модуля автентифікації. Її архітектура передбачає чітке розмежування сутностей – користувачів, біометричних шаблонів та журналів автентифікацій – що дає можливість масштабувати систему, розширювати функціональність і інтегрувати нові алгоритми машинного навчання без необхідності радикальної зміни структури бази даних. Це зумовлено тим, що в БД зберігається не сама ML-модель чи її параметри, а уніфікований результат її

роботи – числовий вектор ознак (embedding). Незалежно від того, яким алгоритмом він створений (FaceNet, ArcFace, InsightFace, Vision Transformer тощо), структура таблиць не змінюється, оскільки система працює з абстрактним типом даних, а не з конкретною реалізацією моделі. Саме ця логіка проєктування й демонструє можливість легкої інтеграції нових алгоритмів без потреби перебудови схеми даних.

MySQL обрано нами не випадково. На відміну від PostgreSQL чи інших СУБД, MySQL забезпечує стабільне поєднання продуктивності, простоти конфігурації та низького порогу входження для розробників вебсистем. PostgreSQL, безперечно, має розширені можливості типізації, складніші індексні структури та ширшу підтримку аналітичних операцій, однак ці переваги є надлишковими для розв’язання завдання зберігання векторних описів особи та журналів автентифікації. Використання PostgreSQL у цьому випадку збільшує складність налаштування, вимагає більше ресурсів і не дає суттєвої переваги при роботі з числовими векторами, які обробляються за межами БД, у Python-модулі. Альтернативні NoSQL-системи, такі як MongoDB або Cassandra, могли б забезпечити гнучкішу модель зберігання, проте вони ускладнюють підтримку транзакційної цілісності, яка є критичною для біометричної автентифікації. Крім того, MySQL має широке поширення в екосистемі PHP-додатків, добре інтегрується зі стандартними ORM-інструментами та витримує виробничі навантаження у веборієнтованих сервісах.

Отже, MySQL у цьому контексті виступає оптимальним компромісом між функціональністю, продуктивністю, простотою інтеграції та витратами на підтримку, що робить її раціональним вибором для впровадження модуля біометричної автентифікації.

2.5 Обґрунтування вибору технологій та інструментів

Проєктування модуля біометричної автентифікації потребує обгрунтованого вибору технологічного стеку, оскільки від цього залежить точність розпізнавання, швидкість роботи, масштабованість системи та можливість її інтеграції у вебсередовище. Різноманітність існуючих

інструментів комп’ютерного зору та машинного навчання вимагає порівняння їх можливостей, обмежень та сумісності з архітектурою розроблюваного рішення. Порівняємо популярні фреймворки та бібліотеки, що застосовуються для задач виявлення та верифікації облич: MediaPipe, OpenCV, dlib, face_recognition, TensorFlow, PyTorch, а також альтернативні рішення – такі як DeepFace, InsightFace, YOLO-моделі, які використовуються для детекції об’єктів.

Одним із основних критеріїв вибору є точність розпізнавання та стійкість до варіацій, таких як зміна освітлення, ракурс обличчя чи часткові перекриття. Згідно з дослідженнями останніх років, моделі на основі глибоких згорткових мереж (CNN) демонструють значно вищі показники точності у порівнянні з класичними алгоритмами (Haar, HOG) [24]. Однак використання складних нейронних архітектур у веб-модулі автентифікації значних обчислювальних ресурсів, що є недоцільним для розроблюваного прототипу модуля, у якому частина обробки здійснюється на боці користувача, а також на малопотужних веб-серверах.

Важливою перевагою бібліотеки MediaPipe FaceMesh є її здатність виконувати надшвидкі обчислення без графічного прискорювача, що дозволяє стабільно працювати у браузері та забезпечує детальну трекінгову модель обличчя з понад 468 ключових точок. Ця технологія створена для реального часу та оптимізована для обмежених середовищ, що робить її надзвичайно ефективною саме для автентифікації користувачів у веб-інтерфейсі [30]. Крім того, MediaPipe має пряму підтримку JavaScript, що значно спрощує її інтеграцію з клієнтською частиною модуля та забезпечує виконання обчислень безпосередньо у браузері, що потребує використання фреймворку здатного забезпечити впорядковану структуру компонентів і стабільну взаємодію з потоковими даними. До таких фреймворків належить Vue.js – це прогресивний JavaScript-фреймворк для розроблення інтерфейсів користувача та односторінкових вебзастосунків. Він побудований за компонентним підходом: інтерфейс поділяється на незалежні логічні блоки (компоненти), кожен з яких поєднує розмітку, стилі та поведінку. Архітектурно Vue орієнтований на модель

уявного (віртуального) DOM (об'єктна модель документа) реактивне оновлення стану та декларативний опис інтерфейсу, завдяки чому зміни даних автоматично відображаються в UI без прямого маніпулювання DOM [27].

Як фреймворк, Vue забезпечує повноцінну екосистему для побудови клієнтської частини застосунку: він підтримує зв’язування даних (data binding), компоненти, маршрутизацію (через Vue Router), керування станом (через Vuex або Pinia), інтеграцію з інструментами збірки (Vite, Webpack), а також типізацію (TypeScript). Це дозволяє використовувати його як для невеликих віджетів, так і для складних багатомодульних систем.

Переваги Vue полягають насамперед у низькому порозі входження та зрозумілій декларативній моделі. Синтаксис фреймворку є інтуїтивно читабельним, що спрощує навчання й подальшу підтримку коду. Реактивна система даних автоматично відстежує залежності між станом та відображенням, тому розробник зосереджується на логіці, а не на технічних деталях оновлення інтерфейсу. Другою важливою перевагою є гнучкість: Vue можна інтегрувати як «тонкий шар» до вже наявних серверних застосунків (наприклад, на PHP або Python), або використовувати як основу повноцінного SPA. До позитивних характеристик також належать невеликий розмір ядра, висока продуктивність при роботі з віртуальним DOM, активна спільнота, наявність офіційних бібліотек для маршрутизації та керування станом, а також якісна документація.

Водночас Vue має й певні недоліки. Порівняно з більш «корпоративно орієнтованими» рішеннями, такими як Angular, він менш жорстко нав’язує архітектурні шаблони, що в умовах великої команди може призводити до різнотипних стилів написання коду та потребує внутрішніх стандартів. Іншим обмеженням є фрагментація екосистеми: перехід від Vue 2 до Vue 3 супроводжувався зміною підходів (Options API → Composition API), що створює додаткові вимоги до підтримки старих проєктів. Крім того, у порівнянні з React, частина бібліотек та готових UI-рішень може бути менш численною, що інколи потребує додаткової інтеграції з універсальними або «нейтральними» інструментами [22].

У контексті побудови модулів біометричної автентифікації Vue є доцільним вибором завдяки підтримці реактивних інтерфейсів у реальному часі, простій інтеграції з веб-API браузера (камера, відеопотік) та можливості організувати чітку компонентну структуру: окремі компоненти для захоплення зображення, візуалізації ключових точок, керування сценаріями «природності» та взаємодії з бекендом. Це дозволяє поєднати складну внутрішню логіку обробки біометричних даних із прозорою та керованою презентаційною частиною.

Для серверної обробки та побудови векторних представлень облич ми обрали OpenCV у поєднанні з Python-бібліотеками dlib або face_recognition. OpenCV забезпечує необхідний комплекс інструментів для попередньої обробки зображень, що передбачає їх нормалізацію, масштабування, фільтрацію тощо [16]. При цьому dlib дозволяє будувати 128-вимірні вектори ознак на основі CNN-моделі, що забезпечує хороше співвідношення точності та швидкодії. У порівнянні з фреймворками TensorFlow та PyTorch, використання dlib є енергетично та обчислювально ефективнішим і не потребує додаткової інфраструктури.

Під час аналізу альтернатив розглядалися також такі засоби, як DeepFace та InsightFace. Вони демонструють високу точність і підтримують сучасні моделі (ArcFace, CosFace), проте їх інтеграція потребує розгортання окремих серверів або Docker-контейнерів та значно більших ресурсів, що не відповідає концепції прототипу. Вибір мови програмування також було здійснено обґрунтовано. Нами був обраний Python як основа обчислювального блоку ML через його домінування у галузі машинного навчання, наявність великої кількості бібліотек, активну спільноту та можливість швидкого прототипування. Нами були розглянуті й альтернативні мови – C++ та Java. Проте вони мають недоліки в контексті дослідження, зокрема обмежену кількість готових високорівневих ML-інструментів (C++), складність у розгортанні та повільніший цикл розробки (Java) [30]. Отож, Python вважаємо компромісом між продуктивністю та гнучкістю. Для узагальнення проведеного аналізу інструментів та обґрунтування

вибору оптимальної технологічної платформи було систематизовано основні характеристики розглянутих рішень у структурованому вигляді. Це дозволило наочно порівняти їх можливості, обмеження та відповідність вимогам модуля біометричної автентифікації. Підсумкове порівняння наведено в таблиці 2.4.

Таблиця 2.4

Порівняльний аналіз технологій

Технологія / бібліотека	**Призначення**	**Переваги**	**Недоліки**	**Доцільність використання**
OpenCV	Обробка зображень, детекція облич	Висока швидкість, великий набір функцій, кросплатформність	Обмежена точність у складних умовах	Використовується для попередньої обробки зображення
MediaPipe FaceMesh	Трекінг ключових точок	Дуже висока точність, працює в реальному часі, низькі вимоги до ресурсів	Не виконує повну векторизацію облич	Найкраще рішення для визначення геометрії та антиспуфінгу
Dlib	Побудова 128-вимірних векторів ознак облич	Висока точність, стійкість до кутів та освітлення	Високе навантаження на CPU	Доцільний для векторизації та збереження шаблонів облич
Face_recognition	Інструментарій для розпізнавання облич	Простота, висока точність, готові моделі	Повільна робота на великих масивах даних	Підходить для інтеграції в невеликі системи
InsightFace	Сучасні високоточні моделі ArcFace	Найвища точність, стійкість	Вимогливість до GPU	Не використовується через відсутність апаратної підтримки
Python	Мова для ML-модуля	Великий набір бібліотек, простота інтеграції	Нижча продуктивність за C++	Обрана як оптимальна для ML-частини
JavaScript (Web API)	Отримання відеопотоку	Простота, нативна підтримка браузерами	Не підходить для важких задач ML	Використовується для фронтенду, але не для обчислень

Порівняння технологій засвідчує, що вдалим рішенням для розроблюваного модуля є комбіноване використання технологій MediaPipe FaceMesh, бібліотеки OpenCV та додаткових Python-бібліотек для векторизації

облич (dlib / face_recognition). Така конфігурація забезпечує баланс між точністю, продуктивністю та простотою інтеграції у веб-орієнтовану архітектуру

MediaPipe забезпечує високоточне відстеження обличчя в режимі реального часу та підтримує механізми виявлення ознак «природності» користувача, що дає змогу відрізнити реальну біометричну взаємодію від спроби підміни статичним зображенням або відеозаписом. OpenCV виконує швидку попередню обробку; Python дає можливість легко реалізувати логіку векторизації та порівняння шаблонів, тоді як інші технології (наприклад InsightFace чи власне моделі CNN) виявилися надмірно вимогливими до апаратних ресурсів або складними в інтеграції, що суперечить вимогам до веб-орієнтованого модуля.

Отож, використання технологій MediaPipe + OpenCV + dlib має забезпечити реалізацію поставлених завдань, за умов дотримання балансу між точністю, обчислювальною ефективністю, простотою інтеграції та можливістю роботи в реальному часі.

2.6 Архітектурно-алгоритмічні засади розроблення модуля біометричної автентифікації

Архітектура розроблюваного модуля біометричної автентифікації ґрунтується на багаторівневій моделі , яка передбачає чітке розмежування функціональних обов’язків між клієнтською частиною, серверним компонентом та модулем машинного навчання. Такий підхід забезпечує масштабованість системи, підвищує її продуктивність та дозволяє незалежно модифікувати окремі підсистеми без порушення загальної логіки роботи. Обрана архітектура відповідає сучасним підходам до побудови модулів комп’ютерного зору, про що свідчать результати наукових досліджень, присвячених розробленню модулів автентифікації на основі машинного навчання та OpenCV [16]

У загальному вигляді система складається з чотирьох основних компонентів: Клієнтського веб-інтерфейсу; PHP-сервера; Python-модуля верифікації; Бази даних MySQL.

Клієнтська частина реалізується у вигляді вебінтерфейсу, який здійснює взаємодію з камерою користувача для отримання зображення обличчя. Для первинної обробки та виявлення ключових точок обличчя застосовується бібліотека MediaPipe FaceMesh, яка працює безпосередньо у браузері, що знижує навантаження на сервер і дає змогу швидко перевіряти коректність позиціювання та видимості обличчя. Застосування такої технології відповідає сучасним тенденціям локальної попередньої обробки даних перед передачею на сервер, що підвищує швидкодію системи.

PHP-контролер відіграє роль посередника між інтерфейсом користувача та модулем машинного навчання. Він приймає HTTP-запити, відповідає за кодування отриманого зображення у формат Base64, передає його Python-компоненту та обробляє відповідь від нього. Окреме виділення веб-сервера дає змогу організувати чіткий контроль потоків даних, реалізувати авторизацію запитів, а також інтегрувати модуль у вже наявні веб-системи. На подібну багаторівневу модель вказують сучасні дослідження, де серверна частина виділяється як окремий логічний блок, що взаємодіє з ML-компонентами через API [14].

Найважливішим елементом системи є Python-модуль, який реалізує алгоритми верифікації. Він отримує захоплене зображення, виконує виявлення обличчя за допомогою OpenCV та визначає вектор ознак, використовуючи модель машинного навчання (наприклад, ResNet-архітектуру, яка застосовується у бібліотеці face_recognition). Подібний підхід, коли векторизація обличчя виконується в окремому Python-процесі, відповідає рекомендаціям наукової літератури щодо використання глибоких моделей CNN/ResNet для біометричного розпізнавання обличчя, що забезпечує високу точність і стійкість до зовнішніх чинників [25].Результати роботи Python-модуля зберігаються та зіставляються з даними, які знаходяться у базі MySQL. У ній зберігаються вектори ознак зареєстрованих користувачів, їхні персональні дані та журнал подій автентифікації

Взаємодія між компонентами модуля організована у вигляді послідовного потоку даних: клієнтська частина передає зображення → PHP-сервер приймає та обробляє запит → Python-модуль виконує верифікацію → результати передаються назад на PHP-сервер → користувач отримує відповідь про успішність автентифікації. Такий підхід дає змогу чітко відокремити логіку обробки даних та забезпечити модульність системи.

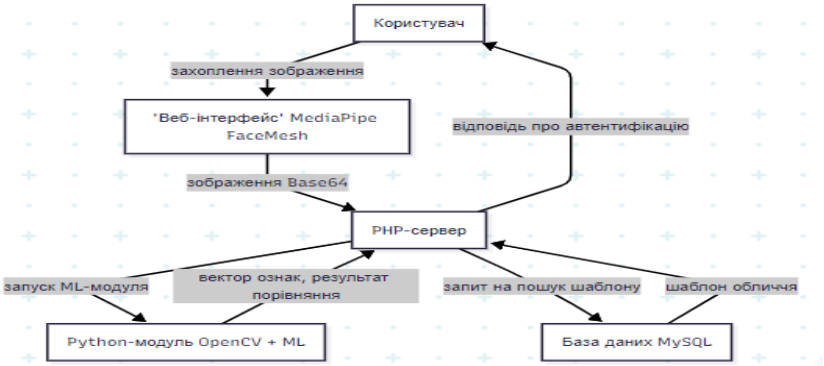

Рис.2.3 Схема передавання даних

Наведена схема (рис.2.3) відображає логічну послідовність обробки біометричних даних у розроблюваному модулі автентифікації та демонструє взаємозв’язок між його основними компонентами. Процес ініціюється користувачем, який активує механізм захоплення зображення обличчя через клієнтський веб-інтерфейс, реалізований на основі технології MediaPipe FaceMesh. Отримане зображення кодується у формат Base64 та передається на серверну частину системи.

PHP-сервер виконує функцію координуючої ланки, що забезпечує маршрутизацію даних та взаємодію між підсистемами. Він передає отримане зображення до Python-модуля, де за допомогою бібліотеки OpenCV та алгоритмів машинного навчання формується вектор біометричних ознак обличчя. Після обчислення вектор порівнюється з відповідними шаблонами, що зберігаються у базі даних MySQL.

За результатами порівняння PHP-сервер здійснює запит до бази даних з метою встановлення відповідності між отриманим вектором та наявним біометричним шаблоном. У випадку виявлення збігу система приймає рішення про успішну автентифікацію, після чого на клієнтську сторону повертається відповідний результат.

Отже, схема є формальним відображенням архітектурної логіки модуля та демонструє послідовний рух даних між клієнтським середовищем, серверною частиною та підсистемою машинного навчання, що забезпечує цілісність, узгодженість і відтворюваність процесів біометричної автентифікації.

Алгоритмічне забезпечення модуля біометричної автентифікації визначає логіку обробки вхідних даних, механізми виявлення та аналізу облич, способи формування біометричних ознак, а також підходи до порівняння отриманих характеристик із наявними шаблонами у базі даних. Оскільки модель, яка застосовується в модулі, на момент проєктування не є попередньо навченою, процес її навчання є ключовим етапом розроблення. Це дозволяє адаптувати алгоритм до особливостей цільового набору даних, а також забезпечити високий рівень точності у специфічних умовах експлуатації. Як наслідок алгоритмічне забезпечення передбачає виконання завдань щодо навчання моделі та безпосереднього виконання автентифікації в реальному часі.

На етапі навчання системи формується репрезентативний набір даних, що містить зображення облич користувачів, отримані у різноманітних умовах освітлення, під різними ракурсами та з різними мімічними варіаціями. Така варіативність необхідна для забезпечення стійкості моделі до реальних сценаріїв використання, оскільки алгоритми біометричної автентифікації повинні коректно працювати навіть за умов неповного, затемненого або частково спотвореного відображення обличчя.

Перед формуванням ознак кожне зображення проходить процедури попередньої обробки, які містять нормалізацію освітлення, масштабування та усунення шумів. Після цього виконується детекція області обличчя за допомогою моделей MediaPipe FaceMesh або вбудованих детекторів OpenCV [16]. Використання MediaPipe FaceMesh є обґрунтованим, оскільки ця модель забезпечує високу точність локалізації контурів обличчя та здатна визначати понад 400 тривимірних ключових точок. Це дозволяє отримати детальні геометричні характеристики структури обличчя, які є менш чутливими до змін міміки, кута повороту та варіацій освітлення.

На основі виділених ключових точок формується вектор ознак, який виступає компактним математичним представленням обличчя користувача. Саме цей вектор надалі використовується як вхід для моделей машинного навчання, що реалізують процедури порівняння та ідентифікації. Такий підхід забезпечує високу стійкість системи до неконтрольованих зовнішніх факторів та дозволяє отримати векторні подання, які є придатними для подальшого аналізу, класифікації та збереження у базі даних без прив’язки до конкретного зображення.

Отже, процес навчання моделі є багатоетапним і передбачає підготовку набору даних, детекцію облич, формування ключових точок та побудову векторів ознак, що у сукупності забезпечує основу для функціонування алгоритмів біометричної автентифікації.

Після визначення області обличчя виконується нормалізація, яка використовує масштабування, усунення шумів, вирівнювання по ключових точках та коригування контрасту. Такі операції забезпечують зменшення впливу варіативності вхідних даних на точність подальших алгоритмів машинного навчання. Наступним кроком є кодування обличчя у вигляді вектора ознак. Залежно від обраного інструментарію це 128- або 512-вимірний вектор, отриманий за допомогою моделей dlib (ResNet-based), FaceNet або інших нейромережевих архітектур. Вектор ознак слугує компактним і стійким представленням обличчя, яке може бути використане як для навчання, так і для верифікації (рис.2.4).

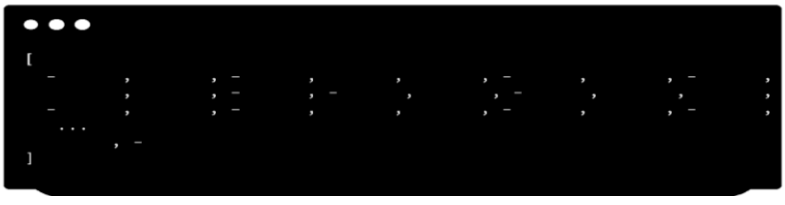

Рис.2.4 128-вимірний вектор

Для моделі, яка поки не навчена, формується векторний простір, у якому векторні подання різних користувачів підлягають класифікації або кластеризації з використанням алгоритмів метричного навчання (наприклад, triplet loss або cosine-margin loss). На цьому етапі відбувається оптимізація параметрів моделі з

метою визначення максимальної відстані між векторами різних користувачів та мінімізації відстані між векторами зображень одного користувача. Важливо забезпечити достатню кількість варіативних прикладів для кожного класу, що дозволяє підвищити узагальнюючу здатність моделі.

У режимі верифікації у реальному часі модуль працює за аналогічним принципом, проте без етапу навчання. Система отримує зображення з вебкамери, виконує детекцію обличчя, визначає ключові точки та будує їхнє векторне представлення. Далі здійснюється порівняння векторних ознак з тими, що зберігаються у базі даних MySQL як еталонний шаблон. Порівняння виконується за евклідовою «формула (2.5)» або косинусною «формула (2.6)» відстанню, причому поріг прийняття рішення обирається на основі результатів попереднього експериментального тестування. Поріг прийняття рішення в модулі біометричної автентифікації визначається не довільно, а встановлюється на основі емпіричних спостережень. Його значення отримано шляхом аналізу розподілу евклідових відстаней між векторами, сформованими для однойменних та різних користувачів. Під час експериментального тестування було встановлено, що поріг 0.6 забезпечує оптимальне співвідношення між показниками хибного допуску (FAR) та хибної відмови (FRR). Зменшення порогового значення призводило до зростання частки некоректних відмов автентичним користувачам, тоді як його збільшення спричиняло підвищення ймовірності неправомірного доступу.

$$d_{E(a,b)} = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2} \qquad (2.5)$$

де: $d_{E(a,b)}$ — евклідова відстань між векторами ознак a та b, a – вектор ознак зареєстрованого користувача, b – вектор ознак, отриманий під час автентифікації, a_i — i – та компонента вектора a, b_i — i та компонента вектора b, n – розмірність векторів ознак:

$$d_{cos}(a,b) = 1 - sim_{cos}(a,b) \qquad (2.6)$$

де $d_{cos}(a,b)$ — косинусна відстань між векторами a та b, $sim_{cos}(a,b)$ – косинусна подібність між векторами, a – вектор ознак зареєстрованого користувача, b – вектор ознак, отриманий під час автентифікації.

Окремим пунктом є реалізаціяж захисту від атак «підміни» (liveness detection). У модулі передбачено методики виявлення ознак реальності, наприклад аналіз руху голови, мікродинаміку міміки, реакцію зіниць або зміни глибини кадру. Такі методи суттєво знижують ризик автентифікації за допомогою статичних зображень або відеозаписів.

Для структуризації процесу автентифікації та фіксації послідовності його етапів подано схематичне зображення алгоритму модуля. На рисунку 2.7 відображено основні компоненти системи та їх взаємодію: від захоплення зображення обличчя до обчислення вектора ознак і порівняння його з еталонним шаблоном у базі даних. Схема дозволяє чітко простежити логіку обробки даних та визначити функціональне призначення кожного елемента, що забезпечує цілісність та відтворюваність процесу біометричної автентифікації.

Рис.2.7 Алгоритм модуля автентифікації.

Проєктування програмних компонентів прототипу модуля біометричної автентифікації передбачає чітке визначення ролей кожного елемента системи та організацію їхньої взаємодії так, щоб забезпечити стабільну, точну та швидку роботу модуля в умовах реального часу. Архітектурні рішення базуються на принципах модульності, слабкого зв’язування та багаторівневої обробки даних, що дає можливість масштабувати систему й окремо оптимізувати кожен її компонент [7].

Система складається з кількох логічних рівнів: клієнтської частини , яка відповідає за захоплення та попередню обробку зображення; серверної частини, що координує передавання даних та виконує бізнес-логіку; модуля машинного навчання, який формує та порівнює вектори ознак; а також підсистеми зберігання, де фіксуються біометричні шаблони та результати автентифікацій. Кожен із цих рівнів виконує окрему функцію, але разом вони забезпечують цілісність та ефективність роботи системи.

Front-end (веб-інтерфейс). Клієнтська частина відповідає за організацію взаємодії користувача з модулем автентифікації та забезпечує доступ до камери пристрою. Тут реалізується збирання відеопотоку, попередня валідація даних, а також формування запитів до PHP-сервера. Використання JavaScript та стандартів WebRTC дозволяє виконувати захоплення кадрів у режимі реального часу з мінімальною затримкою.

PHP виконує функцію контролера, який координує взаємодію між веб-інтерфейсом, Python-модулем та базою даних MySQL. Сервер приймає дані із клієнтської частини, здійснює попередню обробку запитів, передає дані до Python-модуля здійснюється у двох режимах. Якщо користувач проходить первинну реєстрацію, PHP надсилає безпосередньо зображення обличчя, щоб Python сформував відповідний вектор ознак. Під час автентифікації, коли вектор уже збережено в базі даних, PHP передає Python лише нове зображення для побудови поточного векторного опису та порівняння його з еталонним шаблоном. Крім цього, PHP відповідає за доступ до MySQL, виконання CRUD-операцій та формування API-відповідей у форматі JSON або XML [14].

Python компонент є обчислювальним ядром системи. Він виконує: детекцію облич на зображенні за допомогою OpenCV; побудову лендмарків за допомогою MediaPipe FaceMesh; кодування облич у векторні ознаки; порівняння отриманих векторів із шаблонами, що зберігаються в MySQL; повернення PHP-серверу результату у вигляді ймовірності. Взаємодія між компонентами

Процес верифікації відбувається впродовж таких етапів:

1. Користувач активує веб-камеру у браузері та надсилає кадр до PHP.

2. PHP перенаправляє зображення до Python-модуля через локальний або віддалений API.

3. Python обробляє кадр, визначає обличчя, обчислює векторні ознаки та порівнює їх зі збереженим шаблоном.

4. Результат повертається на PHP-сервер, який зберігає лог у MySQL і надсилає відповідь у браузер.

5. На фронтенді користувач отримує рішення «успішно» «відмовлено».

Модульність такого підходу спрощує тестування, дає змогу замінювати алгоритми ML без зміни всієї системи та забезпечує гнучку інтеграцію з будь-якими веб-платформами.

Під час реєстрації відбувається первинне формування біометричного шаблону, який надалі використовується для автентифікації. Цей процес складається з таких етапів:

Автентифікація також передбачає узгоджену взаємодію всіх архітектурних компонентів системи, проте саме на фінальному етапі виконується порівняння нових біометричних даних користувача з еталонним шаблоном, що зберігається у базі даних MySQL. До таких компонентів належать клієнтський модуль захоплення зображення, серверна частина, яка забезпечує маршрутизацію запитів, модуль обробки та побудови ембедингів (Python/OpenCV/ML-модель), а також підсистема зберігання біометричних шаблонів у MySQL [13]. Кожен із цих компонентів виконує визначену функцію, але саме база даних виступає джерелом референсного шаблону, на основі якого ухвалюється рішення щодо успішності або відмови у доступі.

1. Захоплення поточного зображення у фронтенді. Як і під час реєстрації, користувач за допомогою веб-камери робить фото обличчя в інтерфейсі входу.

2. Формування запиту на PHP-сервер. Отримане зображення надсилається до PHP-модуля, який визначає, який обліковий запис автентифікується, та передає дані Python-модулю для подальшої обробки.

3. Обробка та побудова векторного представлення у Python. Python-модуль, отримавши зображення, повторює попередні кроки:

a. детекція облич;

b. знаходження ключових точок MediaPipe;

c. формування векторного подання.

4. Отримання шаблону з MySQL та порівняння. PHP отримує векторний опис від Python, запитує з MySQL відповідний шаблон користувача та викликає Python-модуль повторно для здійснення порівняння. Порівняння виконується за метрикою подібності (евклідова відстань або cosine similarity). Якщо отримане значення менше за встановлений поріг, система вважає, що обличчя належить зареєстрованому користувачу.

5. Повернення результату до фронтенду. PHP надсилає у веб-інтерфейс відповідь: «автентифікація успішна» або «відмова». Користувач отримує візуальний та текстовий результат у браузері.

Виконане нами проєктування забезпечує цілісну основу для розроблення функціонального модуля біометричної автентифікації. Обґрунтовані вибір технологій, структурні рішення та алгоритмічні моделі гарантують можливість реалізації прототипу, здатного забезпечити достатню точність, продуктивність та інтегрованість із сучасними веб-системами. Подальші етапи роботи будуть пов’язані з реалізацією програмного забезпечення та оцінюванням його ефективності на реальних даних.

Отже, нами було створено цілісну концепцію розроблення модуля біометричної автентифікації, що поєднує вимоги безпеки, точність розпізнавання та стійкість до реальних умов використання. Визначено основні принципи: надійність біометричної ознаки, модульність та масштабованість архітектури, мінімізацію збережених даних, шифрування критичної інформації та узгоджену взаємодію між клієнтською, серверною та ML-частинами системи.

Побудовано модель даних на основі MySQL із чіткою структурою сутностей users, biometric_templates і auth_logs. Обґрунтовано використання форматів JSON/BLOB для зберігання ембедингів, а також застосування механізмів шифрування, контролю доступу й аудиту для забезпечення цілісності та конфіденційності біометричних шаблонів.

Обрано набір технологій, у якому MediaPipe FaceMesh відповідає за визначення облич, OpenCV і dlib/face_recognition – за створення векторів ознак, Python виконує основні обчислення, Vue.js працює як інтерфейс користувача, а PHP – як серверна частина. Створена архітектура має кілька рівнів, забезпечує чітку передачу даних між компонентами та дає змогу легко оновлювати або змінювати окремі частини системи.

Узгоджене поєднання архітектурних рішень, моделей даних, алгоритмів та інструментів створює основу для подальшої реалізації прототипу біометричної автентифікації .

У другому розділі сформовано концептуальні та інженерні засади проєктування прототипу модуля біометричної автентифікації, орієнтованого на верифікацію й ідентифікацію користувачів за ознаками обличчя в умовах веб-орієнтованого середовища. Визначено, що ключовими вимогами до такого модуля є забезпечення конфіденційності, цілісності та доступності даних, а також досягнення високої точності й стабільності алгоритмів у реальних умовах експлуатації (варіативність освітлення, ракурсу, якості камери та наявність часткових перекриттів). Обґрунтовано необхідність поєднання механізмів алгоритмічної стійкості з організаційними та технічними заходами безпеки, зокрема принципом мінімізації збережених біометричних даних.

Висновки до другого розділу

У розділі ІІ виконано проєктування та обґрунтовано технологічне забезпечення прототипу модуля біометричної автентифікації, орієнтованого на ідентифікацію та верифікацію користувачів за ознаками обличчя у веб-орієнтованому середовищі. Визначено ключові вимоги до системи: забезпечення конфіденційності, цілісності та доступності даних, досягнення високої точності розпізнавання та стабільної роботи в реальних умовах експлуатації (зміна освітлення, ракурсів, якості камери, часткові перекриття).

Сформовано концептуальні засади проєктування, що включають орієнтацію на стійкість біометричної ознаки, модульність і масштабованість архітектури, мінімізацію збережених біометричних даних, використання

порогових значень достовірності для ухвалення рішень та узгоджену інтеграцію клієнтських і серверних компонентів. Розроблено модель даних на базі MySQL із виокремленням сутностей users**,** biometric_templates та auth_logs, визначено їх призначення і зв’язки, а також обґрунтовано формати збереження векторних ознак (переважно JSON/BLOB) з урахуванням вимог до швидкодії та сумісності між компонентами.

Обґрунтовано вибір технологічного стеку: MediaPipe FaceMesh для локальної детекції та аналізу геометрії обличчя у браузері, OpenCV для попередньої обробки зображень, dlib/face_recognition для формування векторних представлень, Python як обчислювальне ядро, PHP як координуючий серверний компонент та Vue.js для реалізації інтерфейсу користувача. Описано багаторівневу архітектуру та алгоритмічна послідовність (захоплення кадру → попередня обробка → векторизація → порівняння за метриками евклідової або косинусної відстані → прийняття рішення), а також визначено підхід до встановлення порога рішення на основі експериментальних спостережень.

Отже, отримані архітектурні рішення, модель даних, механізми захисту та обрані інструменти формують цілісну основу для подальшої практичної реалізації прототипу модуля біометричної автентифікації та його експериментального оцінювання на реальних даних.

РОЗДІЛ 3
РОЗРОБЛЕННЯ ТА РЕАЛІЗАЦІЯ МОДУЛЯ

3.1 Ініціалізація структури модуля та базової сторінки входу

Розроблення модуля біометричної автентифікації розпочалося зі створення програмної структури, здатної забезпечити інтеграцію вебінтерфейсу, серверної логіки та модулів машинного зору. Початковим кроком у процесі розроблення модуля біометричної автентифікації було створення та налаштування базового застосунку на PHP, який став каркасом усієї майбутньої системи. На цьому етапі було важливо не лише отримати функціональне середовище, а й створити просту архітектуру, яка забезпечувала б гнучкість, розширюваність і можливість інтеграції зовнішніх компонентів, зокрема Python-модулів комп'ютерного зору. Як основу було обрано фреймворк Laravel, котрий вирізняється чіткою MVC-архітектурою, розвиненою системою маршрутизації. Після ініціалізації проєкту було створено таку структуру каталогів: -app/ - містить основну бізнес-логіку, контролери та моделі; -routes/ - зберігає файли з маршрутами, які визначають поведінку HTTP-запитів; -resources/ - містить шаблони інтерфейсу, компоненти JavaScript, стилі; -public/ - кореневий каталог веб-сервера, де зберігаються згенеровані скрипти та стилі.

Така структуризація дозволяє логічно розподіляти відповідальність між частинами системи: клієнтська частина, серверна частина та ресурси оброблення даних не перетинаються і можуть розвиватися незалежно.

Після створення каркасу ми визначили, що користувач взаємодіятиме з системою через єдину початкову сторінку. Саме цей підхід було обрано з міркувань модульності: замість створення окремих HTML-документів для кожного сценарію (вхід, реєстрація, перегляд облич), було вирішено використати принцип динамічного завантаження компонентів у межах однієї SPA-архітектури (Single Page Application). Ця концепція передбачала, що під час переходу між сторінками користувач фактично не залишає меж однієї веб-сторінки,а змінюється лише її внутрішній вміст, який підвантажується через JavaScript. Додатковими перевагами такої реалізації є: базова сторінка

завантажується лише один раз; подальші компоненти (реєстрація обличчя, авторизація, перегляд списку користувачів) підключаються асинхронно; зменшується навантаження на сервер і час очікування користувача; створюється відчуття окремого застосунку, а не традиційного сайту.

Такий підхід дозволив модулю залишатися швидким, інтуїтивним і здатним до масштабування. Якщо в майбутньому виникне потреба додати новий функціональний блок (наприклад, редагування профілю чи аналіз журналів автентифікації), це можливо зробити без зміни базової структури, шляхом додавання нового Vue-компонента.

Отже, етап ініціалізації не обмежувався лише створенням порожнього каркасу проєкту - він сформував архітектурну основу, яка забезпечила логічну ізоляцію частин програми, дозволила використовувати SPA-підхід, інтегрувати Python-скрипти та створити єдину точку входу, що уможливлює централізоване керування всіма процесами біометричної автентифікації..

3.2 Розроблення та реалізація сценаріїв для маршрутизації

Другим етапом розроблення модуля стало формування маршрутизації на стороні інтерфейсу користувача, яка пов’язує логічні сценарії роботи системи з конкретними візуальними компонентами інтерфейсу. У межах односторінкової архітектури саме маршрутизатор Vue Router відповідає за те, який компонент (форма автентифікації, реєстрація та захоплення зображення, отримання кадру та перевірка «природності», акаунт користувача) буде відтворено у кореневому контейнері застосунку в залежності від поточного шляху в адресному рядку браузера. Завдяки цьому реалізується поведінка, близька до класичної багатосторінкової веб-системи, але без фізичного переходу між окремими HTML-документами. Вихідний код маршрутизатора зосереджений у файлі `router.js`. На початку відбувається імпорт бібліотек та компонент, що відповідають за різні розділи модуля біометричної автентифікації: список користувачів, реєстрацію, ідентифікацію, сторінки акаунту та помилки.

```
import Vue from 'vue';
import VueRouter from 'vue-router';
import List from './views/List.vue';
import Identify from './views/Identify.vue';
import Register from './views/Register.vue';
import Account from './views/Account.vue';
import NotFound from './views/NotFound.vue';

Vue.use(VueRouter);

```

Рис. 3.1 Ініціалізація маршрутизації клієнтської частини

У цьому фрагменті Vue Router реєструється як плагін для основного екземпляра Vue, що надає можливість описувати маршрути у вигляді декларативної конфігурації. Кожен із компонентів `List`, `Register`, `Identify`, `Account` та `NotFound` є окремим представленням, яке реалізує самостійний підсценарій роботи користувача з модулем (рис.3.1). Така побудова відповідає компонентно-орієнтованому підходу: логіка, розмітка та стилі кожного сценарію ізольовані в межах відповідного Vue-файла. Наведемо приклад масиву `routes`, у якому описано відповідність між шляхами (URL) та компонентами модуля (рис.3.2.).

```
const routes = [
    {
        path: '/',
        name: 'List',
        component: List
    },
    {
        path: '/register',
        name: 'Register',
        component: Register
    },
    {
        path: '/identify',
        name: 'Identify',
        component: Identify
    },
    {
        path: '/account',
        name: 'Account',
        component: Account
    },
    {
        path: '*',
        redirect: '/404'
    },
    {
        path: '/404',
        component: NotFound
    }
];
```

Рис. 3.2 Конфігурація маршрутів вебзастосунку

Узгодженість структури маршрутів із функціональністю модуля є важливою з погляду як інженерії, так і користувацького досвіду. Кореневий маршрут `/` пов’язано з компонентом `List`, який відображає перелік зареєстрованих користувачів. Це дає змогу сприймати головну сторінку як «панель огляду» системи, де адміністратор або оператор бачить загальний стан бази облич. Маршрут `/register` асоціюється з компонентом `Register`, у якому реалізовано повний процес реєстрації: від введення імені та електронної пошти до виклику камери та захоплення зображення. Перехід за шляхом `/identify` активує компонент `Identify`, який відповідає за сценарій розпізнавання:

користувач підносить обличчя до камери, система аналізує зображення, викликає Python-скрипт та повертає результат автентифікації.

Маршрут `/account` вказує на розширення системи автентифікації від суто алгоритмічного розпізнавання до повноцінної роботи з профілем користувача. Цей компонент може містити, наприклад, дані про поточного користувача, історію входів, додаткові налаштування або прив’язки до інших сервісів. Наявність окремого маршруту для акаунта свідчить про те, що модуль позиціонується не як ізольований приклад технології розпізнавання, а як складова повноцінної програмної системи, у якій біометрична автентифікація виконує лише одну зі спеціалізованих функцій. Це означає, що архітектура передбачає подальший розвиток – інтеграцію ролей користувачів, персоналізованих налаштувань, систем сповіщень, додаткових методів доступу та інших сервісних механізмів, які виходять за межі простого порівняння обличчя. Отже, біометричні дані перетворюються не на кінцеву мету, а на ядро, яке забезпечує безпечний доступ до розширених можливостей системи, що відповідає концепції масштабованих та модульних вебплатформ.

Дуже важливим елементом конфігурації є обробка помилкових шляхів. Запис з `path: '*'` визначає поведінку системи для будь-якої адреси, що не збігається з жодним із явно оголошених маршрутів. Такі запити автоматично перенаправляються на `/404`, де відображається компонент `NotFound`. Це дозволяє уникнути ситуацій, у яких користувач потрапляє у «порожній» інтерфейс або бачить неінформативне повідомлення браузера. Натомість йому демонструється повідомлення про помилку, що є важливою складовою ергономіки та надійності модуля.

Завершальним етапом конфігурації маршрутизації є ініціалізація екземпляра Vue Router та визначення режиму його функціонування. На цьому кроці формується об’єкт маршрутизатора, у межах якого фіксується повний перелік доступних маршрутів, їх відповідність конкретним компонентам інтерфейсу та правила переходу між ними (рис.3.3).

```
const router = new VueRouter({
    mode: 'history',
    base: process.env.BASE_URL,
    routes
});

export default router;

```

Рис. 3.3 Створення об’єкта маршрутизатора з використанням режиму history

Використання history означає, що маршрутизація реалізується за допомогою History API браузера. Замість URL із хеш-частиною (`/#/identify`) застосовуються «чисті» адреси (`/identify`), які сприймаються як повноцінні маршрути. Це особливо важливо в умовах, коли фронтенд розгортається разом з бекендом на одному домені: сервер може бути налаштований у такий спосіб, щоб усі невідомі запити, що не відповідають статичним ресурсам, перенаправлялися на базовий HTML-файл, а далі вже Vue Router бере на себе визначення фактичного компонента, який буде завантаувати відповідний JavaScript.

Такий спосіб організації навігації виконує роль контролера, який акумулює всю логіку переходів між розділами в одному місці. Це дає нам точку керування сценаріями: додавання нового розділу системи зводиться до імпорту нового компонента, опису нового маршруту у масиві `routes` та, за потреби, конфігурування доступу чи додаткових параметрів. Для користувача це проявляється в плавних переходах, відсутності перезавантажень сторінки, стабільній структурі адрес і передбачуваній поведінці інтерфейсу.

У підсумку маршрутизатор на основі Vue Router формує каркас навігації модуля біометричної автентифікації. Він узгоджує між собою сценарії перегляду списку користувачів, реєстрації обличчя, ідентифікації на основі знятого зображення та роботи з акаунтом, а також забезпечує коректне опрацювання помилкових маршрутів.

3.3 Реалізація сторінки реєстрації обличчя та компонента захоплення зображення

На цьому етапі було реалізовано функціональну сторінку реєстрації користувача, яка поєднує традиційну форму введення даних із інтерактивним компонентом захоплення зображення обличчя через вебкамеру. Основною метою створення такої сторінки є забезпечення можливості формування

якісного біометричного зразка, що використовується для побудови векторних представлень та подальшої ідентифікації користувача в системі.

Головним елементом розроблюваного інтерфейсу виступає Vue-компонент Register, який організовує логіку сторінки та забезпечує узгоджену взаємодію між усіма елементами процесу: формою введення даних, інструктивними елементами, камерою та модулем FaceMesh. Компонент відповідає за ініціалізацію доступу до вебкамери, налаштування параметрів відображення відеопотоку, підключення до бібліотеки MediaPipe та запуск алгоритмів аналізу обличчя. Завдяки цьому забезпечується коректне захоплення кадру, який у подальшому передається на сервер у форматі Base64 для оброблення Python-модулем.

Інтерфейс сторінки реєстрації побудований так, щоб користувач отримував чітку структуру взаємодії. У верхній частині розміщено блок для відображення повідомлень про помилки. Він використовується для інформування про некоректно заповнені поля, неможливість доступу до камери або інші технічні відхилення. Такий механізм дозволяє покращити користувацький досвід та забезпечити контроль правильності виконання кожного етапу.

Під блоком повідомлень розташовано форму введення персональних даних – імені та електронної пошти. Ці поля використовують реактивну валідацію, яка прив’язує значення полів до відповідних моделей ($v.name.$model та $v.email.$model). На рівні логіки компоненту визначено правила обов’язкового заповнення полів, що унеможливлює створення неповних або некоректних записів у базі даних. Кнопка переходу до наступного етапу («Продовжити») стає активною лише тоді, коли всі поля заповнено правильно.

```
openInstructionModal() {
    this.$v.$touch();
    if (this.$v.$invalid) return;
    const actions = ["smile", "blink", "nod"];
    this.currentAction = actions[Math.floor(Math.random() *
actions.length)];
    this.$bvModal.show("instruction-modal");
}

```

Рис. 3.4 Метод ініціалізації інструкцій користувача

Натискання кнопки не спричиняє негайного відправлення форми. Замість цього викликається метод openInstructionModal (рис3.4), який виконує повторну

перевірку коректності введених даних та переводить користувача до наступного інтерфейсного кроку. На цьому етапі користувач отримує інструкцію щодо подальших дій та переходить до процесу увімкнення камери. Вміст інструктивного вікна формується геттером actionText, що забезпечує динамічне оновлення тексту.

Після зчитування інструкції користувач активує камеру, а компонент Register ініціалізує модуль MediaPipe FaceMesh, який займається аналізом основних геометричних характеристик обличчя. FaceMesh в реальному часі відслідковує ключові ознаки обличчя та визначає, чи присутнє обличчя у кадрі, чи воно достатньо освітлене, чи займає відповідну частину області знімання дані алгоритми дозволяють переконатися у тому, що захоплений кадр є придатним для подальшої обробки та побудови векторних представлень.

Після успішного визначення обличчя користувач може виконати знімання кадру. Отримане зображення конвертується у формат Base64 та надсилається разом із текстовими полями на сервер для формування початкового біометричного шаблону. Так реалізовано послідовний процес створення нового облікового запису, у якому користувач передає як структуровані текстові дані, так і біометричний зразок, що є необхідним для подальших процедур ідентифікації.

Створена архітектура сторінки реєстрації забезпечує чітку логіку взаємодії та модульність (рис.3.5). Компонент Register виконує функцію центрального керуючого елемента, а інтерфейсні блоки, форма введення даних, інструктивне вікно та модулі роботи з камерою формують цілісний механізм зручної та коректної реєстрації користувача в системі біометричної автентифікації.

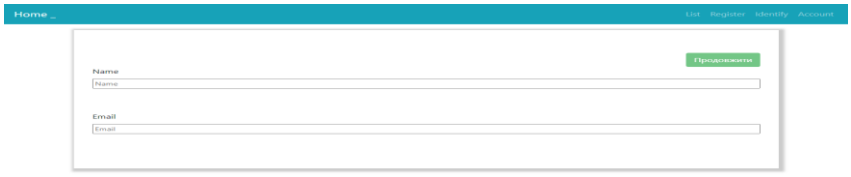

Рис. 3.5 Користувацький інтерфейс сторінки реєстрації

3.4 Обробка зображення на стороні PHP

Після реєстрації користувача фронтенд-компонент надсилає на сервер POST-запит, у якому передаються дані користувача. Обробка цього запиту в Laravel винесена в окремий сервісний клас Train, що відповідає за збереження даних у базу, декодування зображення, формування файлового датасету та ініціацію скрипта навчання моделі на Python (рис.3.6). Такий підхід розвантажує контролер і концентрує всю логіку, пов’язану з «тренуванням», в одному структурному елементі.

```php
public function execute($request)
{
    $user = User::create($request->toArray());
    $filename = $user->id . '.png';
    //var_dump(explode(',', $request->encode_img));
    $content = base64_decode(explode(',', $request->encode_img)[1]);
    Storage::put('faces/' . $filename, $content);
    $this->runScript();

    return response()->json([
        'status' => 'success',
    ], 201)
}
```

Рис.3.6 Обробка даних та збереження біометричного зображення на сервері

У вказаному класі розглянемо детальніше метод execute($request). На його початку створюється новий запис користувача у моделі User на основі даних HTTP-запиту: User::create($request->toArray()). Це дозволяє отримати унікальний ідентифікатор щойно зареєстрованого користувача, який надалі використовується як частина імені файла зображення. Файл формально ідентифікується як <id>.png, де <id> - це числовий первинний ключ створеного облікового запису користувача. Як наслідок між записом у базі та файлом із його обличчям встановлюється однозначний зв’язок.

Після цього з рядка encode_img, що містить зображення у форматі виділяється власне закодований фрагмент. Це реалізується через операцію explode(',', $request->encode_img), яка відділяє префікс із MIME-типом від тіла закодованого зображення. Далі цей фрагмент декодується стандартною функцією base64_decode, у результаті чого отримується бінарне подання JPEG-зображення. Отриманий вміст записується до файлового сховища Laravel через фасад Storage. Виклик методу Storage::put('faces/' . $filename, $content) зберігає файл у каталозі storage/app/faces, де кожному користувачу відповідає окремий файл із назвою, що співпадає з його ID. Така організація каталогу робить датасет

структурованим, а самі файли легкодоступними для подальшої обробки Python-модулем.

Після успішного збереження зображення викликається приватний метод runScript(), який відповідає за запуск скрипта навчання моделі розпізнавання (рис.3.7).

```
private function runScript()
{
    $storagePath = storage_path('app' . self::DS . 'faces');
    $scriptPath = app_path('Console' . self::DS . 'Scripts');
    $pathScript = 'Console' . self::DS . 'Scripts' .
        self::DS . 'face_train.py';
    $scriptAppPath = app_path($pathScript);
    $process = new Process([
        env('PYTHON_PATH'),
        $scriptAppPath,
        $storagePath,
        $scriptPath
    ]);
    $process->run();
    if (!$process->isSuccessful()) {
        throw new ProcessFailedException($process);
    }
}
```

Рис. 3.7 Реалізація запуску Python-скрипту для тренування моделі

Усередині цього методу формується два шляхи: storagePath із посиланням на каталог storage/app/faces, де зберігаються всі зображення облич, та scriptPath із посиланням на каталог, у якому розташовано Python-скрипти (наприклад, app/Console/Scripts). Далі конструюється повний шлях до файлу скрипта face_train.py. Для виконання скрипта використовується компонент Symfony\Component\Process\Process, що дозволяє безпечно запускати зовнішні процеси з PHP. Екземпляру Process передається масив параметрів: шлях до інтерпретатора Python (визначений у змінній середовища PYTHON_PATH), шлях до скрипта, а також аргументи, серед яких каталог зі збереженими обличчями та шлях до каталогу скриптів.

Запуск відбувається через $process->run(). Після завершення перевіряється статус виконання: якщо процес завершився невдало (!$process->isSuccessful()), опрацьовується виняток ProcessFailedException, що сигналізує про помилку у роботі Python-частини. У випадку успішного виконання скрипт навчання оновлює модель розпізнавання, враховуючи нове зображення, додане до датасету. Отож, кожна реєстрація користувача не лише поповнює базу даних, але й одразу додає відповідне обличчя до тренувального набору OpenCV-моделі.

На виході метод execute() формує JSON-відповідь зі статусом success та кодом 201, що сигналізує про успішне створення нового ресурсу (рис.3.8).

```
return response()->json([
    'status' => 'success',
], 201);
```

Рис. 3.8 Формування JSON-відповіді сервера про успішне виконання запиту

Уся описана послідовність від декодування Base64-зображення і збереження у файловому сховищі до запуску зовнішнього Python-процесу забезпечує безперервний ланцюг між подією реєстрації на фронтенді та оновленням внутрішньої моделі розпізнавання облич, яка надалі використовується у процедурі автентифікації.

3.5 Реалізація Python-модуля тренування моделі розпізнавання облич

На наступному етапі реалізації модуля було розроблено окремий Python-скрипт face_train.py, який виконує роль тренувального модуля системи розпізнавання облич. На відміну від класичних рішень на базі OpenCV-розпізнавачів (на кшталт LBPHFaceRecognizer), у цьому проєкті використано стек бібліотек dlib який містить попередньо натреновану ResNet-модель. Такий підхід забезпечує формування додаткових векторних подань обличчя (embeddings, які надалі використовуються для процедури ідентифікації користувачів.

Скрипт отримує два аргументи командного рядка: шлях до каталогу з зображеннями облич (path) та шлях до каталогу скриптів (scriptPath), де розміщено файли моделей та де зберігатиметься результат тренування. Відповідні змінні ініціалізуються на початку скрипта.

```
path = sys.argv[1]      # директорія з фото
scriptPath = sys.argv[2] # де зберігати trainer.pkl
```

Далі завантажуються необхідні моделі dlib: фронтальний детектор облич, предиктор ключових точок та модель побудови дескрипторів облич:

```
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor(os.path.join(scriptPath, "shape_predictor_68_face_landmarks.dat"))
facerec = dlib.face_recognition_model_v1(
   os.path.join(scriptPath, "dlib_face_recognition_resnet_model_v1.dat")
)
```

Отож, модуль має три основні компоненти: детектор, що визначає положення обличчя на зображенні; предиктор, який обчислює 68 опорних точок обличчя; та ResNet-модель, що на основі цих точок формує числовий вектор

фіксованої довжини, який є унікальним «відбитком» обличчя в ознаковому просторі.

Основна логіка обробки зображень інкапсульована у функції getEmbeddingsAndLabels(path). На вхід вона отримує шлях до каталогу зі збереженими у файловому сховищі обличчями, попередньо доданими на етапі реєстрації. Усередині функції формується список шляхів до файлів, а для кожного з них виконується послідовність кроків. Спочатку зображення відкривається за допомогою бібліотеки Pillow та конвертується у формат RGB:

```
img = Image.open(imagePath).convert('RGB')
img_numpy = np.array(img)
```

Далі з назви файлу регулярним виразом виділяється числовий ідентифікатор користувача. Це дозволяє не залежати від конкретної структури імені файлу, достатньо, щоб у ньому містився ID:

```
id_match = re.search(r'\d+', imagePath)
if not id_match:
   continue
id = int(id_match[0])
```

Після цього задіюється dlib-детектор для пошуку облич на зображенні. Для кожного знайденого обличчя обчислюються точки за допомогою методу predictor та отримується вектор ознак обличчя (face descriptor) за допомогою методу facerec.compute_face_descriptor:

```
dets = detector(img_numpy, 1)
for face in dets:
   shape = predictor(img_numpy, face)
   face_descriptor = np.array(
      facerec.compute_face_descriptor(img_numpy, shape)
   )
```

Отриманий результат організовується у словник embeddings, де ключем є ID користувача, а значенням – список векторів, отриманих з усіх зображень, що відповідають його обліковому запису. Це дозволяє усереднити дані з кількох зразків, якщо для одного ID є декілька фото. Зазначене завдання виконуєтсья за допомогою такого фрагменту коду:

```
if id not in embeddings:
   embeddings[id] = []
embeddings[id].append(face_descriptor)
```

Після проходження по всіх зображеннях для кожного користувача за допомогою наступного фрагмету обчислюється середнє значення векторів, щоб

отримати один узагальнений ембединг, який представляє «типове» обличчя даного користувача:

```
for id in embeddings:
   embeddings[id] = np.mean(embeddings[id], axis=0)
```

Отриманий словник embeddings містить для кожного користувача один усереднений вектор, підготовлений для подальшого використання в модулі ідентифікації. Якщо словник не порожній, він серіалізується та зберігається у файл trainer.pkl у каталозі scriptPath з використанням модуля pickle:

```
with open(os.path.join(scriptPath, "trainer.pkl"), "wb") as f:
   pickle.dump(embeddings, f)
```

У завершальній частині скрипт формує JSON-відповідь зі статусом success або error і виводить її в стандартний потік, щоб PHP-код, який запускає цей процес, міг за потреби інтерпретувати результат. Тобто Python-модуль тренування моделі реалізує повний цикл обробки: завантаження фотозразків, детекцію облич, побудову векторних представлень, їх агрегацію за користувачами та формування структурованого файлу з embeddings, який використовується під час розпізнавання облич на етапі автентифікації.

3.6 Реалізація Python-скрипта ідентифікації облич

Модуль ідентифікації завершує цикл біометричної автентифікації, реалізуючи логіку порівняння нового зображення користувача з існуючими біометричними відбитками, що були сформовані на етапі тренування. На цьому рівні система вже не виконує навчання, а працює як розпізнавач, застосовуючи попередньо сформовану базу векторних описів особи, де кожне обличчя користувача представлено у вигляді вектора ознак.

Скрипт запускається сервером Laravel через системну команду та отримує два параметри. Перший є шляхом до зображення, яке було зроблено на стороні клієнта при спробі автентифікації. Другий – вказує на каталог, у якому зберігаються всі необхідні файли для розпізнавання: моделі dlib, дані ембедингів та конфігураційні параметри.

```
imgPath = sys.argv[1]
scriptPath = sys.argv[2]
```

Після обробки переданих параметрів скрипт завантажує попередньо сформовані ембеддинги, що є основою для порівняння з новими даними.На

наступному етапі завантажуються три моделі dlib. Це фронтальний детектор для виявлення облич, предиктор для побудови ключових точок та ResNet-модель для перетворення обличчя у 128-вимірний числовий вектор.

Використання цих компонентів дозволяє виділити з обличчя фрагменти (патерни), які є стабільними, незалежно від освітлення, повороту голови чи незначних змін міміки (рис. 3.9).

```
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor(os.path.join(scriptPath,
  "shape_predictor_68_face_landmarks.dat"))
facerec = dlib.face_recognition_model_v1(
    os.path.join(scriptPath, "dlib_face_recognition_resnet_model_v1.dat")
)
```

Рис.3.9. Завантаження моделей детекції облич

Обробка зображення починається із його завантаження та запуску детектора. У випадку, якщо обличчя не було знайдено, система повертає відповідь із позначкою про неможливість ідентифікації. Це дозволяє бекенду не помилково інтерпретувати відсутність обличчя як неправильний пароль.

Коли детектор визначає координати обличчя (рис.3.10.), предиктор формує карту ключових точок, що використовується моделлю ResNet для генерації ембедингу. Кожний векторний опис є числовою проєкцією обличчя у багатовимірний простір, де подібні обличчя розташовані близько одне до одного.

```
shape = predictor(image, face)
face_descriptor = np.array(
    facerec.compute_face_descriptor(image, shape)
).reshape(1, -1)
```

Рис.3.10. Формування векторного представлення обличчя

Новий вектор ознак порівнюється з усіма векторними поданнями у базі. Критерій відповідності визначається через евклідову відстань, що є числовою мірою схожості між двома векторами. Чим менша ця відстань, тим більш імовірно, що обличчя належить певному користувачеві. Вона обчислюється за допомогою виклику наступного методу:

```
dist = euclidean_distances(face_descriptor, embedding.reshape(1, -1))[0][0]
```

Скрипт обирає найменшу відстань, яка інтерпретується як показник впевненості. Значення confidence у даній реалізації не є відсотковим показником,

а розглядається як числова помилка. Її низьке значення означає високу точність збігу, а велике – невідповідність.

```
response = {"confidence": confidence, "id": id}
print(json.JSONEncoder().encode(response))
```

Рис.3.11. Формування JSON-відповіді з результатами біометричної ідентифікації

Як наслідок бекенд отримує структуру з ідентифікатором користувача та значенням впевненості (рис.3.11). Надалі PHP-сервер ухвалює остаточне рішення щодо автентифікації, враховуючи порогове значення. Якщо відстань менша або дорівнює порогу, користувача вважають автентифікованим. Якщо ж вона перевищує поріг, система реєструє подію як спробу входу невідомого користувача.

Така архітектура дозволяє масштабувати систему без необхідності повного перенавчання моделі, оскільки додавання нового користувача означає лише розширення файлу ембедингів, а не перерахунок параметрів всієї моделі.

3.7 Інтеграція Python із PHP-сервером через REST API і фронтенд-клієнт

На фінальному етапі розробки було реалізовано інтеграцію між клієнтським інтерфейсом, PHP-бекендом та Python-модулями розпізнавання через API, що відповідає архітектурі REST. Така інтеграція побудована навколо трьох основних HTTP-маршрутів: list, register та recog, які реалізовані у вигляді методів контролера UserController на стороні сервера:

```
Route::get('list', 'UserController@list');
Route::post('register', 'UserController@register');
Route::post('recog', 'UserController@recog');
```

Кожен із цих маршрутів відповідає окремому етапу життєвого циклу роботи з біометричними даними. Перший маршрут забезпечує отримання списку всіх зареєстрованих користувачів разом із закодованими зображеннями, другий-реєстрацію нового користувача та запуск тренування моделі на Python-стороні, третій – процедуру розпізнавання обличчя та повернення результатів ідентифікації. Важливо, що для фронтенду ці маршрути виглядають як звичайні REST-ендпоінти(кінцеві точки доступу).

На клієнтському рівні взаємодія з API реалізована за допомогою спеціального сервісу api (обгортка над HTTP-клієнтом, наприклад Axios). Для відображення списку зареєстрованих користувачів Vue-компонент викликає метод list(), який відправляє GET-запит на ендпоінт /list:

```
list: function () {
  api.get("/list").then((response) => {
    let data = response.data;
    this.items = response.data;
    this.search = true;
  });},
```

Отримані дані записуються до масиву items, який використовується для побудови таблиці облікових записів користувачів із відображенням їх ідентифікаторів, імен, email-адрес та закодованих зображень. Отже, бекенд можна вважати джерелом даних, а фронтенд лише відтворює те, що повертає сервер, не маючи доступу до внутрішніх деталей зберігання чи тренування моделі.

```
const data = {
    name: this.name,
    email: this.email,
    encode_img: image,
};
if (!this.$v.$invalid) {
    try {
        await api.post("register", data);
        // ...
    } catch (err) {
        // обробка помилок
    }
}
```

Рис. 3.12 Надсилання даних для реєстрації

Під час реєстрації на ендпоінт register передає дані у UserController@register (рис.3.12), де вони спрямовуються до спеціального сервісу (наприклад, Train), що реалізує збереження користувача, декодування зображення, запис файлу у файлову систему та виклик Python-скрипта навчання. У результаті вектор нового обличчя додається до бази даних, і система одразу готова розпізнавати цього користувача при подальших спробах входу. Для фронтенда цей процес зведений до простої операції api.post("register", data), хоча за нею стоїть повний ланцюг дій на бекенді та Python-рівні.

Процес розпізнавання (автентифікації) реалізовано через метод recog() (рис.3.13) у Vue-компоненті, який викликає ендпоінт recog.

```
recog() {
    this.found = false;
    this.searching = true;
    const snapshotCanvas = this.$refs.snapshot;
    const imgEncoded = snapshotCanvas.toDataURL();
    api
        .post("recog", { img: imgEncoded })
        .then((response) => {
            const data = response.data;
            if (data.status == "success") {
                this.name = data.user.name;
                this.email = data.user.email;
                this.confidence = data.confidence.toFixed(2) + "%";
                this.found = true;
            } else {
                this.errorMessage = data.status == "unknown"
                    ? "No user was found."
                    : "No face was found.";
            }
        })
        .catch((err) => {
            this.errorServer = err.response
                ? err.response.status + " | " + err.response.statusText
                : err.message;
        })
        .finally(() => {
            this.searching = false;
        });
}

```

Рис.3.13 Метод recog()

На бекенді метод UserController@recog приймає зображення у полі img, декодує його та зберігає як тимчасовий файл, після чого викликає Python-скрипт ідентифікації, передаючи йому шлях до цього файлу та шлях до каталогу з моделями. Скрипт обчислює ембединг (векторне представлення), порівнює його з наявною базою і повертає JSON-результат з ідентифікатором користувача та числовим показником впевненості. PHP-код інтерпретує відповідь: при успішному зіставленні завантажує з бази даних інформацію про користувача та формує відповідь із полями status, user, confidence; у разі відсутності збігу або обличчя повертаються відповідні статуси unknown чи no_face.

Отже, інтеграція Python із PHP-сервером реалізована не прямо, а через внутрішні виклики й обробку результатів у контролерах та сервісних класах, тоді як для клієнтської частини взаємодія зводиться до роботи з трьома REST-ендпоінтами(кінцевими точками). Такий поділ дозволяє чітко розмежувати відповідальність: Vue-компоненти відповідають за збір і відображення даних, Laravel - за маршрутизацію, валідацію й координацію, а Python - за алгоритмічно складні обчислення у сфері комп’ютерного зору та розпізнавання обличчя.

3.8 Сторінка «списку облич» (List)

Сторінка «списку облич» є однією з ключових частин модуля біометричної автентифікації, оскільки забезпечує відображення та контроль над зареєстрованими у системі користувачами. Вона дає змогу наочно переглядати базу осіб, що вже були додані, а також використовує збережені зображення як підтвердження коректності роботи механізмів реєстрації, тренування та взаємодії з файловою підсистемою. У ширшому контексті сторінка списку

виступає контролем стану датасету, оскільки саме ці дані слугують навчальною множиною для моделі розпізнавання облич.

У початковій реалізації репозиторію для цієї сторінки використовувався Blade-шаблон, що відтворював табличний список зареєстрованих облич. Метод index() контролера отримував усі записи з моделі Face та передавав їх у вигляд:

```
public function index()
{
   $faces = Face::all();
   return view('face.list', compact('faces'));
}
```

Шаблон, відповідальний за рендеринг таблиці, виглядав наступним чином:

```
{{-- resources/views/face/list.blade.php --}}
@extends('layouts.app')

@section('content')
   <h2>Зареєстровані обличчя</h2>
   <table>
      <thead>
         <tr>
            <th>ID</th>
            <th>Ім'я</th>
         </tr>
      </thead>
      <tbody>
         @foreach($faces as $face)
            <tr>
               <td>{{ $face->id }}</td>
               <td>{{ $face->name }}</td>
            </tr>
         @endforeach
      </tbody>
   </table>
@endsection
```

Ця реалізація демонструвала основну функцію - відображення зареєстрованих користувачів, однак у межах SPA-архітектури вона була модернізована і переведена у формат Vue-компонента, що отримує дані з REST API. Такий перехід дозволив додати інтерактивні можливості, зокрема візуалізацію фотографій, перегляд кожного зображення у збільшеному вигляді та потенційне видалення або редагування запису.

У новій реалізації метод list() на клієнті здійснює запит до бекенд-маршруту GET /list:

```
list: function () {
   api.get("/list").then((response) => {
```

```
      let data = response.data;
      this.items = data;
      this.search = true;
    });
},
```

Система зберігає отримані результати у масиві items, що використовується для заповнення таблиці у компоненті List.vue. Таблиця тепер містить як текстові поля, так і закодоване зображення обличчя, яке можна переглянути у збільшеному форматі за допомогою модального вікна. Приклад відповідного фрагменту:

```
<b-table :items="items" :fields="fields">
  <template v-slot:cell(picture)="row">
    <img
      v-b-modal="'modal-'+ row.item.id"
      class="img-picture"
      :src="'' + row.item.base64"
    />
    <b-modal :id="'modal-'+ row.item.id" :title="row.item.name" ok-only>
      <img class="img-modal" :src="'' + row.item.base64" />
    </b-modal>
  </template>
</b-table>
```

Сторінка отримала додаткову функціональність, яка суттєво розширила її первинне призначення та перетворила її на повноцінний інструмент взаємодії з біометричним репозиторієм користувачів. Одним із ключових нововведень стала можливість прямої візуалізації зображень, що зберігаються у файловій системі разом із записами в базі даних. Завдяки цьому можна одразу переконатися, що обличчя, яке було зафіксоване під час реєстрації, коректно прив’язане до відповідного користувача й успішно інтегроване до датасету, який надалі бере участь у тренуванні моделі розпізнавання.

Не менш важливою складовою є розширений механізм відображення великої кількості користувачів, реалізований за допомогою пагінації(нумерації сторінок). Це рішення забезпечує масштабованість системи та дозволяє працювати з великими наборами даних без істотних затримок або перевантаження інтерфейсу. Завдяки доданому контекстному перегляду зображень користувач має змогу відкривати зображення у збільшеному форматі, що є особливо зручним під час діагностики помилок або перевірки відповідності вхідного зображення після спроби автентифікації. Така функція робить

можливим швидке порівняння фактичної візуальної інформації з результатами роботи моделі.

Ще однією важливою характеристикою оновленої сторінки є швидка взаємодія з бекендом. Усі зміни, внесені до бази даних, як додавання нових облич, так і видалення чи редагування існуючих записів, відображаються у фронтенді без необхідності перезавантаження сторінки. Це забезпечує відчуття безперервної роботи системи та відповідає сучасним вимогам до інтерактивності вебзастосунків.

У результаті сторінка List перетворилася з простого механізму перегляду текстових даних на багатофункціональний інструмент управління біометричним контентом. Вона відіграє ключову роль у забезпеченні контролю за якістю датасету, дає змогу переконатися у його цілісності, відстежувати появу нових користувачів, оперативно виявляти некоректні записи та аналізувати результати роботи алгоритму розпізнавання облич на практиці. Саме через цю сторінку модуль отримує зворотний зв’язок із даними, на яких він навчається та працює, а користувач упевненість у прозорості та коректності процесів, що лежать в основі біометричної автентифікації.

3.9 Реалізація механізму для запобігання спуфінгу

На етапі проєктування клієнтської частини модуля реєстрації було поставлене окреме завдання – мінімізувати ризики спуфінгу, тобто спроб автентифікації за допомогою фотографій, скріншотів або заздалегідь підготовлених відеозаписів. Для цього було реалізовано механізм перевірки «природності» користувача, який поєднує випадковий вибір дії, аналіз мікрорухів обличчя за даними MediaPipe FaceMesh та просторовий контроль коректності позиціонування обличчя в кадрі. У результаті система не лише фіксує візуальну подібність, але й вимагає від користувача виконання в реальному часі динамічної дії, яку складно відтворити статичним зображенням. Після формування логіки вибору випадкової дії та її передачі у компонент інтерфейсу наступним кроком стало створення механізму генерації текстових інструкцій, які відображаються користувачеві відповідно до обраного сценарію.

Це забезпечує узгодженість між внутрішнім станом системи та змістом повідомлення. Нижче наведено фрагмент коду, відповідальний за формування тексту інструкції для вибраної дії:

```
  computed: {
 actionText() {
    switch (this.currentAction) {
      case "smile":
        return "Будь ласка, посміхніться ☺";
      case "blink":
        return "Будь ласка, кліпніть очима 😉";
      case "nod":
        return "Будь ласка, нахиліть голову 🙃";
      default:
        return "Будь ласка, виконайте дію";
    }
  },
},
```

Як було зазначено вище, була використана ідея випадково обраної інструкції, яку користувач повинен виконати безпосередньо перед захопленням кадру для реєстрації. Для цього у компоненті реєстрації визначено обчислювану властивість actionText, що повертає текст інструкції залежно від значення поля currentAction. Якщо поточна дія дорівнює «smile», інтерфейс виводить повідомлення «Будь ласка, посміхніться 😊», для «blink» користувачеві пропонується «Будь ласка, кліпніть очима 😉», а для «nod» – «Будь ласка, нахиліть голову 🙃». Усі інші значення призводять до виведення тексту помилки. Значення змінної currentAction є логічним тригером, який визначає очікувану поведінку користувача у межах одного сеансу реєстрації. Він виконується за допомогою такого коду

В масиві actions зібрані три можливі команди – «smile», «blink» та «nod». За допомогою генератора псевдовипадкових чисел вибирається одна з них, після чого встановлюється значення currentAction і відкривається модальне вікно з інструкцією. Такий підхід забезпечує непередбачуваність поведінки системи для потенційного зловмисника: неможливо заздалегідь підготувати статичний чи зациклений відеоматеріал, який би гарантовано відповідав потрібному жесту. Саме на цьому етапі відбувається підготовка інструментів комп’ютерного зору,

необхідних для аналізу відеопотоку в реальному часі. Нижче наведено фрагмент коду, що демонструє початок процедури ініціалізації FaceMesh:

```
async initFaceMesh() {
  this.faceMesh = new FaceMesh({
    locateFile: (file) =>
      `https://cdn.jsdelivr.net/npm/@mediapipe/face_mesh/${file}`,
  });

  await this.faceMesh.setOptions({
    maxNumFaces: 1,
    refineLandmarks: true,
    minDetectionConfidence: 0.5,
    minTrackingConfidence: 0.5,
  });
  this.faceMesh.onResults(this.onResults);
}
```

Для аналізу рухів обличчя застосовано бібліотеку MediaPipe, зокрема її модуль FaceMesh. Її ініціалізація виконується у методі initFaceMesh, де створюється об’єкт FaceMesh з вказанням функції locateFile, яка завантажує необхідні ресурси з CDN. Далі задаються параметри детекції: система налаштована на обробку одного обличчя, з уточненими опорними точками та визначеними порогами впевненості для детекції та трекінгу. Обробник результатів onResults реєструється як колбек, що дозволяє при кожному кадрі отримувати масив ключових точок обличчя та передавати їх у методи drawLandmarks і detectAction. Тобто FaceMesh працює як високорівневий сенсор, який трансформує відеопотік у послідовність координат анатомічно значущих точок.

Візуальний контроль положення обличчя реалізовано так: на початку він очищає полотно, заповнює його напівпрозорим затемненням, після чого у центрі створюється «вирізаний» овальний отвір, який позначає область, де має розташовуватися обличчя. Додатково овал обводиться зеленим контуром (рис3.14), що виконує роль візуального маркера для користувача. Поверх цього контурного каркасу відображаються точки обличчя, що надходять із FaceMesh.

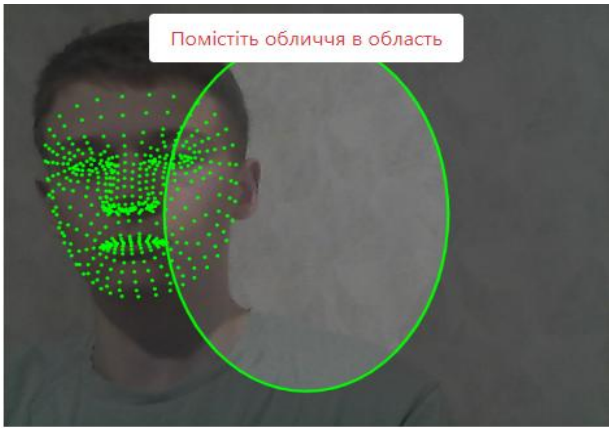

Рис.3.14 Контроль коректного розміщення обличчя

Така організація інтерфейсу не лише покращує зручність взаємодії з користувачем, але й виконує функцію початкової фільтрації даних: якщо обличчя виходить за межі контрольної зони, трекінг вважається недостовірним. Це дозволяє відсіювати кадри зі сторонніми об’єктами, некоректним положенням голови або надто великою відстанню до камери, забезпечуючи якість подальшої обробки. Для реалізації цієї перевірки використовується спеціальна логіка аналізу координат ключових точок, що визначає, чи перебуває обличчя всередині допустимої області. Нижче наведено фрагмент коду, який формує геометрію контрольного овалу та слугує основою для подальшої оцінки правильності позиціонування обличчя:

```
detectAction(landmarks) {
  const width = this.$refs.overlayCanvas.width;
  const height = this.$refs.overlayCanvas.height;
  const centerX = width / 2;
  const centerY = height / 2;
  const ovalWidth = 300;
  const ovalHeight = 400;

  const isInsideOval = landmarks.every((point) => {
    const x = point.x * width;
    const y = point.y * height;
    const dx = x - centerX;
    const dy = y - centerY;
    return ((dx * dx) / ((ovalWidth / 2) ** 2) + (dy * dy) / ((ovalHeight / 2) ** 2)) <= 1;
  });
  if (!isInsideOval) {
    this.error = "Помістіть обличчя в область";
    return;
  }
  this.error = "";
  const mouthWidth = Math.abs(landmarks[291].x - landmarks[61].x);
  const mouthHeight = Math.abs(landmarks[14].y - landmarks[13].y);
  const leftEye = Math.abs(landmarks[159].y - landmarks[145].y);
  const rightEye = Math.abs(landmarks[386].y - landmarks[374].y);
  const eyeOpen = (leftEye + rightEye) / 2;
```

```
    const noseTipY = landmarks[1].y;
    const foreheadY = landmarks[10].y;
    const headTilt = noseTipY - foreheadY;
    switch (this.currentAction) {
      case "smile":
        if (mouthWidth > 0.06 && mouthHeight > 0.02) {
          this.smileDetected = true;
          this.captureAndSubmit();
        }
        break;
      case "blink":
        const isBlinking = eyeOpen < 0.015;
        if (isBlinking && !this.previousBlinkState) {
          this.previousBlinkState = true;
          this.smileDetected = true;
          this.captureAndSubmit();
        }
        if (!isBlinking) {
          this.previousBlinkState = false;
        }
        break;
      case "nod":
        if (headTilt > 0.05) {
          this.smileDetected = true;
          this.captureAndSubmit();
        }
        break;
    }
}
```

Логіка виявлення дії користувача реалізована у методі detectAction. Спершу він повторно обчислює геометрію овалу, що використовується як область інтересу. Кожна точка з масиву landmarks масштабується до розмірів полотна, після чого перевіряється, чи потрапляє вона всередину еліпса. Важливим аспектом є те, що після успішного спрацювання будь-якої дії встановлюється прапорець smileDetected, який блокує подальшу обробку, аби уникнути дублювання запитів до сервера. Надалі метод працює лише доти, доки ця змінна має значення false. Для аналізу посмішки використовуються відносні відстані між ключовими точками рота: ширина обчислюється як різниця між координатами точок 291 та 61, а висота – між 14 та 13. Якщо обидва показники перевищують емпірично підібрані порогові значення, інтерпретується, що користувач посміхнувся. Для визначення кліпання очима розраховується середнє значення відкритості лівого та правого ока за різницею вертикальних координат відповідних точок. Якщо це значення короткочасно падає нижче певного порогу, а попередній стан був «очі відкриті», фіксується факт кліпання, і викликається захоплення кадру. Для жесту нахилу голови порівнюються

вертикальні координати кінчика носа та ділянки лоба; перевищення їх різниці над заданим порогом інтерпретується як нахил уперед (рис.3.15).

```
drawLandmarks(landmarks) {
    const canvas = this.$refs.overlayCanvas;
    const ctx = canvas.getContext("2d");
    ctx.fillStyle = "rgba(0, 0, 0, 0.5)";
    ctx.fillRect(0, 0, canvas.width, canvas.height);
    ctx.save();
    ctx.globalCompositeOperation = "destination-out";
    ctx.beginPath();
    ctx.ellipse(canvas.width / 2, canvas.height / 2, 150, 200, 0, 0,
Math.PI * 2);
    ctx.fill();
    ctx.restore();
}

```

Рис. 3.15 Формування області позиціонування обличчя

У всіх трьох випадках, після виконання потрібної дії, встановлюється істиненне значення змінної smileDetected, і викликається метод captureAndSubmit, який зупиняє логіку природності й переходить до етапу формування зображення для відправлення на сервер. Як наслідок кадр захоплюється не довільно, а в момент, коли система з високою ймовірністю зафіксувала виконання користувачем наперед невідомої дії. Це суттєво ускладнює використання статичних та записаних матеріалів для обходу модуля.

Висновки до третього розділу

У розділі було реалізовано повний цикл роботи модуля біометричної автентифікації – від захоплення зображення обличчя на стороні клієнта до прийняття рішення про автентифікацію на основі векторного подання. Побудовано SPA-архітектуру на базі Laravel та Vue з єдиною стартовою сторінкою, чіткою маршрутизацією («список – реєстрація – ідентифікація – акаунт») та обробкою помилкових маршрутів. Це забезпечило логічне розділення клієнтської, серверної та ML-частини, спростило розширення функціональності та надало змогу централізовано керувати сценаріями взаємодії користувача із системою. Розроблено інтерактивну сторінку реєстрації з компонентом захоплення зображення, сервісом Train для декодування та збереження фото, а також Python-модулем тренування, який формує усереднені ембедінги користувачів і зберігає їх у структурованому вигляді. Окремий скрипт ідентифікації реалізує порівняння нового зображення з базою векторів за евклідовою відстанню та повертає результат через REST-інтеграцію з PHP-бекендом. Сторінка «списку облич» забезпечує візуальний контроль датасету, а

механізм перевірки «природності» з використанням MediaPipe FaceMesh та випадкових дій знижує ризики спуфінгу.

У сукупності реалізоване рішення формує цілісний та масштабований модуль біометричної автентифікації, який поєднує зручний користувацький інтерфейс, ефективну обробку даних і підвищений рівень захисту від несанкціонованого доступу. Реалізовано повноцінний програмний прототип модуля біометричної автентифікації, що забезпечує завершений цикл роботи з біометричними даними: від захоплення зображення обличчя на стороні клієнта до ухвалення рішення про автентифікацію на сервері. Побудована SPA-архітектура на основі Laravel та Vue із чітко визначеними сценаріями навігації («список – реєстрація – ідентифікація – акаунт») забезпечує модульність, керованість і розширюваність системи. Реалізовано сторінку реєстрації користувачів з інтегрованим компонентом захоплення зображення та клієнтською валідацією даних, що дозволяє формувати якісні біометричні зразки. На серверному рівні впроваджено механізм декодування та збереження зображень, а також автоматизований запуск Python-модуля тренування, який формує усереднені векторні представлення облич і зберігає їх у структурованому вигляді для подальшого використання.

Розроблено Python-скрипт ідентифікації, який виконує побудову векторів нового зображення, порівняння їх із наявною базою за евклідовою відстанню та повертає результат для ухвалення рішення на PHP-рівні. Інтеграція між фронтендом, бекендом і ML-компонентами реалізована через REST API, що забезпечує чітке розмежування відповідальності між рівнями системи та прозору взаємодію компонентів. Впроваджено сторінку «списку облич» для візуального контролю датасету, перегляду збережених користувачів і аналізу коректності реєстрації. Додатково реалізовано механізм запобігання спуфінгу на основі MediaPipe FaceMesh і випадкових динамічних дій користувача, що підвищує стійкість модуля до атак із використанням статичних або записаних зображень. Узгоджене поєднання клієнтських, серверних і алгоритмічних рішень забезпечує створення цілісного, масштабованого та практично придатного модуля

біометричної автентифікації, готового до подальшого експериментального оцінювання та інтеграції у реальні вебсистеми.

РОЗДІЛ 4
ТЕСТУВАННЯ МОДУЛЯ БІОМЕТРИЧНОЇ АВТЕНТИФІКАЦІЇ

4.1 Підходи до тестування прототипу

Тестування прототипу модуля біометричної автентифікації мало на меті об’єктивно підтвердити працездатність, точність, часові затримки та надійність запропонованої архітектури в умовах реального використання. Важливість цього етапу зумовлена специфікою біометричних систем, де будь-який програмний дефект може призвести до або необґрунтованого доступу сторонньої особи, або, навпаки, відмови легітимному користувачу. Саме тому тестування є не лише механізмом пошуку помилок, а й формальним методом доведення коректності вибраної моделі обробки та порівняння векторних представлень облич.

Особливістю тестування біометричних систем є необхідність оперування не точними символьними значеннями, а ймовірнісними величинами, що визначають схожість двох біометричних шаблонів. На відміну від традиційних застосунків, у яких результат визначається однозначно, модуль розпізнавання повинен працювати з невизначеністю, зумовленою варіативністю облич, змінами освітлення, наявністю шумових ефектів веб-камери, кутом повороту голови та мікровиразами. Тестові процедури були спрямовані не на перевірку фіксованого результату, а на встановлення меж надійності та визначення допустимих рівнів похибки, які не заважають коректній роботі алгоритму.

Вказані вимоги дали змогу побудувати методику тестування як сукупність ізоляційних та комбінованих підходів. Окремо оцінювалася точність обчислення векторних ознак, коректність передавання оброблених даних між модулями, а також витривалість системи до навантаження, що імітує реальне середовище функціонування багатокористувацького сервісу.

4.2 Тестування функціональної коректності

Першочерговим завданням було підтвердження правильності ідентифікації користувача за допомогою моделі, сформованої Python-скриптами. Для цього використовувався сценарій, у якому кожну спробу автентифікації супроводжувало створення спеціального журналу подій (log-файлу рис 4.1.). У

ньому зберігалися позначка часу, ідентифікатор користувача, отримана дистанція між вектором введеного зображення та збереженою в базі ознакою, а також логічний висновок про збіг чи його відсутність. Така система дозволила не лише об’єктивно відстежувати результат кожної операції, а й аналізувати динаміку похибок на окремих етапах розпізнавання.

```
[2025-07-18 08:21:43] ID: 001 | Distance: 37.84 | Match: TRUE
[2025-07-18 14:32:10] ID: 002 | Distance: 66.11 | Match: FALSE
[2025-07-19 09:05:27] ID: 003 | Distance: 41.25 | Match: TRUE
[2025-07-19 18:45:03] ID: 001 | Distance: 39.77 | Match: TRUE
[2025-07-20 07:59:50] ID: 004 | Distance: 79.93 | Match: FALSE
[2025-07-20 13:22:34] ID: 002 | Distance: 44.08 | Match: TRUE
[2025-07-21 10:13:17] ID: 005 | Distance: 91.34 | Match: FALSE
[2025-07-21 16:40:52] ID: 003 | Distance: 38.56 | Match: TRUE
[2025-07-22 11:55:05] ID: 006 | Distance: 68.12 | Match: FALSE
[2025-07-23 08:08:19] ID: 001 | Distance: 36.90 | Match: TRUE
```

Рис. 4.1 Журнал подій.

Побудова підсистеми логування дала змогу зафіксувати поведінку алгоритму за різних умов. У випадках, коли розпізнавання завершувалося невдачею, можна було ретроспективно відтворити ситуацію і оцінити її причини. Аналіз таких записів показав, що основними джерелами помилкової автентифікації ставали надмірне затемнення кадру, накладання сторонніх об’єктів на обличчя, відхилення голови під кутом, більшим ніж 25°, та наявність розмиття, спричиненого низькою частотою кадрів веб-камери або рухом користувача.

Для формальної оцінки точності використовувалися відкриті датасети. Ці набори зображень містять обличчя людей у природних умовах із різними виразами, позами, аксесуарами, ступенем освітлення та фоном. Тестування модуля здійснювалося в реальних сценах, які моделюють повсякденну поведінку користувачів. Завдяки цьому вдалося підтвердити параметри розпізнавання, які є прийнятними для сучасних систем біометричної автентифікації.

Результати тестування засвідчили, що при пороговому значенні 0.6 система інтерпретує обличчя як належне одному й тому самому користувачу, тобто автентифікація вважається успішною. Якщо відстань не перевищує 0.6, ідентифікаційна точність перевищила 96.8%, що відповідає середнім значенням комерційних моделей та підтверджує валідність обраного алгоритмічного підходу. Додаткова верифікація показала, що за умов нормального освітлення та статичного положення голови точність наближається до 99%, проте у випадку

рухів користувача або обмеженого доступу до камери вона очікувано знижується.

4.3 Тестування інтеграційної взаємодії

Оскільки модуль використовує Python як обчислювальний бекенд, а PHP як сервер логіки, важливим стало тестування взаємодії між ними. Внутрішня взаємодія клієнтської та серверної частин реалізується через системний виклик який передає параметри у процес Python у вигляді аргументів командного рядка. Саме на цьому етапі було виявлено низку проблем, характерних для гібридних систем цього типу. Перша група збоїв стосувалася обмежень операційної системи на розмір параметра командного рядка. Зображення, закодоване у форматі base64, може містити до кількох мільйонів символів, і якщо його передано без попередньої нормалізації, виконання скрипта припиняється. Для вирішення цієї проблеми була розроблена передпроцедура валідації, яка перевіряє коректність формату даних, відповідність MIME-типу, розмір масиву та наявність сигнатури зображення. Такий підхід запобігав аварійним завершенням процесу й унеможливлював передавання некоректних об’єктів. Другою проблемою стала відсутність попередньої перевірки наявності обличчя до створення ембедінгів. Без цієї перевірки Python-модуль намагався обчислити вектор ознак навіть у випадку, коли жодного обличчя не було знайдено. Це призводило до помилкових результатів, зниження точності та непередбачуваної поведінки системи. Для запобігання цьому у модулі було використано функцію face_locations(), яка передбачає обов’язковий пошук облич на зображенні перед виконанням обчислень. Якщо обличчя не знайдене, обробка припиняється, що забезпечує коректність вхідних даних для наступних етапів.

4.4 Тестування навантаження

Для оцінки продуктивності модуля було проведено навантажувальні випробування. Модуль тестувався у сценаріях, що моделювали одночасні запити до сервера на виконання операції автентифікації. Результати досліджень показали, що при навантаженні до 50 запитів на хвилину система працювала у штатному режимі, однак при перевищенні цього порогу час відповіді починав

стрімко зростати. Причиною виявився синхронний характер виклику Python-процесу, коли кожен запит змушений чекати завершення попереднього. Наслідком цього стало формування вимоги щодо подальшої оптимізації модуля та перенесення Python-скриптів у незалежний сервіс із підтримкою асинхронних операцій. Серед рекомендованих фреймворків було визначено FastAPI та Flask, які підтримують багатопоточність та можуть обробляти запити без блокування черги. Це дозволяє розподіляти навантаження та забезпечувати майже лінійне масштабування модуля у разі зростання кількості користувачів.

4.5 Валідаційне тестування та захист від спотворених даних

Окремим завданням тестування модуля автентифікації було тестування оцінювання механізмів валідації та захисту від спуфінгу. Біометричні системи, які не містять захисних перевірок, вразливі до атак підміни, коли замість реального обличчя демонструється фотографія або зображення на екрані смартфона. Саме тому модуль було доповнено перевіркою динамічних дій користувача (миготіння, поворот голови, посмішка), що унеможливлює використання статичних зображень для обходу системи. Перевірка динамічних атрибутів реалізована на фронтенді засобами MediaPipe FaceMesh, що аналізує зміщення ключових точок обличчя у часі. Тестування цих методів здійснювалося поетапно. Спочатку перевірялася коректність детекції базових жестів шляхом послідовного виконання рухів очей і голови. Для кожної дії фіксувалися стабільність відстеження ключових точок, швидкість реагування та точність визначення руху. На наступному етапі виконавець відтворював набір визначених жестів – дворазове миготіння, нахил голови, поворот у заданий бік та посмішку. Порівнювалися очікуваний жест і той, що був зафіксований алгоритмом.

Тестування цього механізму показало, що найефективнішим є поєднання трьох умов: визначення позиції очей, вимірювання співвідношення висоти рота до його ширини та оцінка різниці координат носових точок між двома сусідніми кадрами. Власне така реалізація підтвердила свою прагматичність і дозволила

повністю усунути можливість успішного входу за допомогою статичного зображення.

4.6 Аналіз результатів та типових помилок

Аналіз експериментів виявив два основні типи помилок. Перший стосувався помилкового відхилення легітимного користувача, що траплялося у випадках надмірного затемнення або фрагментарного потрапляння обличчя в область розпізнавання. Другий тип полягав у хибному прийнятті доступу, що виникав у рідкісних випадках перекриття об’єктів або у середовищах з некоректним балансом білого. Подальший аналіз підтвердив, що переважна більшість таких збоїв пов’язана не з алгоритмом, а з якістю вхідних даних.

Комплексне тестування здійснювалося у кілька етапів, що охоплювали як функціональні властивості модуля, так і його стійкість до зовнішніх факторів та спроб обходу системи, даний підхід підтвердило коректність реалізованого рішення та його відповідність сучасним вимогам до систем біометричної автентифікації. Модуль демонструє стабільну роботу, високий рівень точності, передбачувану поведінку в умовах змін освітлення та успішно протистоїть атакам спуфінгу середнього рівня. Попри деякі недоліки архітектури, пов’язані із синхронним запуском Python-процесів, модуль може бути масштабований до промислового рівня шляхом використання асинхронних фреймворків.

Висновки до четвертого розділу

Отримані результати підтверджують доцільність використання векторного представлення облич як основи для автентифікації та демонструють можливість застосування запропонованої реалізації у реальних інформаційних системах, зокрема у у програмних рішеннях компанії «Мейджфен» після проходження додаткового тестування та адаптації до внутрішніх технічних вимог. Проведене тестування підтвердило працездатність прототипу модуля біометричної автентифікації та коректність обраної архітектури взаємодії між клієнтською частиною, PHP-сервером і Python-обчислювальним компонентом. Методика випробувань була зорієнтована на специфіку біометричних систем, де рішення формується на основі ймовірного порогу схожості, а тому ключовими

критеріями оцінювання стали межі надійності, допустимі рівні похибки та стабільність роботи за змінних умов знімання.

Функціональне тестування засвідчило, що за порогового значення 0,6 модуль забезпечує ідентифікаційну точність понад 96,8%, а за сприятливих умов (достатнє освітлення та статичне положення голови) показники наближаються до 99%. Аналіз журналів подій дозволив установити типові причини помилок: недостатня освітленість, розмиття кадру, часткові перекриття обличчя та значні відхилення ракурсу, що підтверджує домінуючий вплив якості вхідних даних на результат розпізнавання. Інтеграційне тестування виявило критичні обмеження синхронної взаємодії PHP–Python через аргументи командного рядка, зокрема чутливість до обсягу Base64-представлення та необхідність попередньої валідації формату даних і наявності обличчя у кадрі. Упровадження процедур перевірки MIME-типу, розміру та детекції обличчя до побудови векторів зменшило кількість аварійних збоїв і підвищило передбачуваність поведінки системи.

Навантажувальні випробування продемонстрували, що за інтенсивності до 50 запитів на хвилину модуль функціонує стабільно, тоді як подальше зростання навантаження призводить до збільшення часу відповіді через блокувальний характер запуску Python-процесів. Це обґрунтовує доцільність подальшої оптимізації шляхом винесення ML-частини в окремий сервіс із підтримкою асинхронної обробки запитів, що створює передумови для масштабування. Валідаційне тестування засвідчило ефективність механізмів протидії спуфінгу на основі динамічних дій користувача та аналізу ключових точок MediaPipe FaceMesh, що унеможливлює успішну автентифікацію за статичними зображеннями. Узагальнений аналіз результатів підтверджує доцільність використання векторного подання облич як бази для автентифікації та демонструє практичну придатність запропонованого рішення для інтеграції в реальні інформаційні системи за умови подальшої оптимізації продуктивності й адаптації до внутрішніх технічних вимог.

ЗАГАЛЬНІ ВИСНОВКИ

Кваліфікаційна робота була виконана відповідно до поставленої мети та визначених завдань, що дозволило отримати обґрунтовані теоретичні положення й практичні результати. Усі завдання дослідження були виконані, що дає підстави зробити такі висновки:

1. У ході виконання роботи було досліджено теоретичні засади біометричної автентифікації користувачів та проаналізовано сучасні алгоритми машинного навчання і технології комп'ютерного зору, що застосовуються для розпізнавання облич. Розглянуто класичні методи виявлення облич та підходи до їх ідентифікації на основі згорткових нейронних мереж. Визначено, що перевагами сучасних моделей глибокого навчання є висока точність розпізнавання, стійкість до варіацій освітлення та ракурсу, а також здатність до узагальнення на нових даних. Водночас до обмежень таких підходів належать підвищені вимоги до обчислювальних ресурсів, залежність від якості вибірок даних для навчання та чутливість до часткового перекриття обличчя. Установлено, що ефективне застосування алгоритмів розпізнавання облич у системах автентифікації можливе за умови достатньої якості вхідних зображень, контрольованих умов зйомки та використання попередньо навчених моделей, оптимізованих для конкретного сценарію використання.

2. На основі результатів теоретичного аналізу було спроєктовано модель програмного модуля біометричної автентифікації користувачів, що містить взаємопов'язані функціональні складники, зокрема модуль захоплення та попередньої обробки зображень, компонент детекції та нормалізації облич, модуль формування векторних біометричних ознак, механізм порівняння шаблонів і підсистему прийняття рішення про автентифікацію. Окрему увагу приділено структурі зберігання біометричних даних у вигляді векторних представлень, що мінімізує ризики витоку персональної інформації. Запропонована модель забезпечує чіткий розподіл відповідальності між компонентами системи та відповідає принципам конфіденційності, цілісності та доступності (CIA).

3. У межах практичної частини роботи було розроблено прототип модуля автентифікації користувачів на основі технологій комп’ютерного зору та машинного навчання. Реалізовано повний цикл обробки візуальних біометричних даних, який передбачає захоплення зображення з камери, його попередню обробку, формування векторних представлень облич, їх порівняння з еталонними шаблонами та прийняття рішення щодо автентифікації користувача. Використовуючи архітектурний стиль REST API, реалізовано взаємодію Python-модуля розпізнавання з серверною частиною застосунку. Додатково впроваджено базові механізми захисту від спуфінг-атак. Отримані результати підтвердили можливість практичного використання запропонованого підходу в інформаційних системах різного призначення.

4. Проведене тестування розробленого прототипу дозволило оцінити його функціональну коректність, стабільність роботи та ефективність у реальних умовах використання. Отримані результати засвідчили, що система забезпечує достатньо високий рівень точності автентифікації та здатна працювати в режимі, наближеному до реального часу. Встановлено, що важливими чинниками, які впливають на якість розпізнавання, є роздільна здатність і чіткість зображень, умови освітлення та апаратні характеристики пристроїв захоплення даних, що є типовими обмеженнями для біометричних систем на основі комп’ютерного зору.

Подальший розвиток дослідження доцільно спрямувати на вдосконалення алгоритмів розпізнавання облич шляхом використання новітніх архітектур глибокого навчання, зокрема трансформерних і гібридних моделей, що поєднують згорткові нейронні мережі та механізми самоуваги. Окремим напрямом подальших досліджень є впровадження та оптимізація функціоналу розпізнавання користувачів в окулярах, що передбачає аналіз впливу часткового перекриття ключових зон обличчя на якість формування векторних ознак. Перспективним також є поєднання біометричної автентифікації з багатофакторними та ризик-орієнтованими моделями контролю доступу, що дозволить підвищити загальний рівень безпеки сучасних інформаційних систем.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. About Face ID advanced technology – Apple Support. Apple Support. URL: https://support.apple.com/en-us/102381 (дата звернення: 14.12.2025).

2. A Face Spoofing Detection Method Based on Domain Adaptation and Lossless Size Adaptation / W. Sun та ін. *IEEE Access*. 2020. Т. 8. С. 66553–66563. URL: https://doi.org/10.1109/access.2020.2985453 (дата звернення: 14.12.2025).

3. Analysis of authentication methods for full-stack applications and implementation of a web application with an integrated authentication system / T. Radivilova та ін. *Innovative Technologies and Scientific Solutions for Industries*. 2024. № 3 (29). С. 76–90. URL: https://doi.org/10.30837/2522-9818.2024.3.076 (дата звернення: 14.12.2025).

4. ArcFace: Additive Angular Margin Loss for Deep Face Recognition / J. Deng та ін. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2021. С. 1. URL: https://doi.org/10.1109/tpami.2021.3087709 (дата звернення: 14.12.2025).

5. Balaban S. Deep learning and face recognition: the state of the art. *SPIE Defense + Security*, м. Baltimore, Maryland, United States / ред.: I. A. Kakadiaris, A. Kumar, W. J. Scheirer. 2015. URL: https://doi.org/10.1117/12.2181526 (дата звернення: 14.12.2025).

6. Behavioral biometrics & continuous user authentication on mobile devices: A survey / I. Stylios та ін. *Information Fusion*. 2021. Т. 66. С. 76–99. URL: https://doi.org/10.1016/j.inffus.2020.08.021 (дата звернення: 14.12.2025).

7. Boyko N., Basystiuk O., Shakhovska N. Performance Evaluation and Comparison of Software for Face Recognition, Based on Dlib and Opencv Library. *2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP)*, м. Lviv, 21–25 серп. 2018 р. 2018. URL: https://doi.org/10.1109/dsmp.2018.8478556 (дата звернення: 14.12.2025).

8. CR-GAN: Automatic craniofacial reconstruction for personal identification / Y. Li та ін. *Pattern Recognition*. 2022. Т. 124. С. 108400. URL: https://doi.org/10.1016/j.patcog.2021.108400 (дата звернення: 14.12.2025).

9. Deng J., Zafeririou S. ArcFace for Disguised Face Recognition. *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, м. Seoul, Korea (South), 27–28 жовт. 2019 р. 2019. URL: https://doi.org/10.1109/iccvw.2019.00061 (дата звернення: 14.12.2025).

10. Divya S., T J., Geo A. V. A. Innovative Approaches to Criminal Identification Using Real Time Facial Recognition. *2025 6th International Conference on Mobile Computing and Sustainable Informatics (ICMCSI)*, м. Goathgaun, Nepal, 7–8 січ. 2025 р. 2025. С. 1694–1700. URL: https://doi.org/10.1109/icmcsi64620.2025.10883092 (дата звернення: 14.12.2025).

11. Dubovečak M., Dumić E., Bernik A. Face Detection and Recognition Using Raspberry PI Computer. *Tehnički glasnik*. 2023. Т. 17, № 3. С. 346–352. URL: https://doi.org/10.31803/tg-20220321232047 (дата звернення: 14.12.2025).

12. *DBIR 2023 Data Breach Investigations Report*. URL: https://www.verizon.com/business/resources/reports/2023-data-breach-investigations-report-dbir.pdf (дата звернення: 14.12.2025).

13. Efficient Face Detection And Identification In Networked Video Surveillance Systems / M. N. Saadat та ін. *2020 14th International Conference on Ubiquitous Information Management and Communication (IMCOM)*, м. Taichung, Taiwan, 3–5 січ. 2020 р. 2020. URL: https://doi.org/10.1109/imcom48794.2020.9001697 (дата звернення: 14.12.2025).

14. FastAPI. FastAPI. URL: https://fastapi.tiangolo.com (дата звернення: 14.12.2025).

15. Gartner. Market Guide for Identity Proofing and Affirmation. 2023. URL: https://www.gartner.com/en/documents/4718531 (дата звернення: 08.10.2025).

16. Get Started. OpenCV. OpenCV. URL: https://opencv.org/get-started/ (дата звернення: 14.12.2025).

17. GitHub – deepinsight/insightface: State-of-the-art 2D and 3D Face Analysis Project. GitHub. URL: https://github.com/deepinsight/insightface (дата звернення: 14.12.2025).

18. GitHub - hukkelas/DSFD-Pytorch-Inference: A High-Performance Pytorch Implementation of face detection models, including RetinaFace and DSFD. GitHub. URL: https://github.com/hukkelas/DSFD-Pytorch-Inference (дата звернення: 14.12.2025).

19. GitHub – Star-Clouds/CenterFace: face detection. GitHub. URL: https://github.com/Star-Clouds/CenterFace (дата звернення: 14.12.2025).

20. Golla M. R., Sharma P. Performance Evaluation of Facenet on Low Resolution Face Images. *Communications in Computer and Information Science*. Singapore, 2018. С. 317–325. URL: https://doi.org/10.1007/978-981-13-2372-0_28 (дата звернення: 14.12.2025).

21. Hybrid Token Transformer for Deep Face Recognition / W. Su та ін. *Pattern Recognition*. 2023. С. 109443. URL: https://doi.org/10.1016/j.patcog.2023.109443 (дата звернення: 14.12.2025).

22. JavaScript With Syntax For Types. TypeScript: JavaScript With Syntax For Types. URL: https://www.typescriptlang.org/ (дата звернення: 14.12.2025).

23. Karpiuk N., Klym H., Vasylchychyn I. Facial recognition system based on the Haar cascade classifier method. *2023 24th International Conference on Computational Problems of Electrical Engineering (CPEE)*, м. Grybów, Poland, 10–13 верес. 2023 р. 2023. URL: https://doi.org/10.1109/cpee59623.2023.10285310 (дата звернення: 14.12.2025).

24. Kumar A., Jain S., Kumar M. Face and gait biometrics authentication system based on simplified deep neural networks. *International Journal of Information Technology*. 2022. URL: https://doi.org/10.1007/s41870-022-01087-5 (дата звернення: 14.12.2025).

25. Liu R. Face Recognition Based on Convolutional Neural Networks. *Highlights in Science, Engineering and Technology*. 2022. Т. 16. С. 32–39. URL: https://doi.org/10.54097/hset.v16i.2225 (дата звернення: 14.12.2025).

26. Microsoft Learn: Build skills that open doors in your career. Microsoft Learn. URL: https://learn.microsoft.com/ (дата звернення: 14.12.2025).

27. Multi-Factor Authentication: A Survey / A. Ometov та ін. *Cryptography*. 2018. Т. 2, № 1. С. 1. URL: https://doi.org/10.3390/cryptography2010001 (дата звернення: 14.12.2025).

28. MUI: The React component library you always wanted. MUI. URL: https://mui.com/ (дата звернення: 14.12.2025).

29. MySQL :: MySQL Documentation. MySQL :: Developer Zone. URL: https://dev.mysql.com/doc/ (дата звернення: 14.12.2025).

30. React – JavaScript-бібліотека для створення користувацьких інтерфейсів. React. URL: https://uk.reactjs.org/ (дата звернення: 14.12.2025).

31. Security and Privacy Controls for Information Systems and Organizations. National Institute of Standards and Technology, 2020. URL: https://doi.org/10.6028/nist.sp.800-53r5 (дата звернення: 14.12.2025).

32. Sitzmann T., Mayer M., Zibuschka J. І все ж ви використовуєте пароль – дослідження ключів безпеки FIDO2 у малій компанії. URL: https://www.researchgate.net/publication/358140546_You_still_use_the_password_after_all_-_Exploring_FIDO2_Security_Keys_in_a_Small_Company (дата звернення: 09.09.2025).

33. The Quest to Replace Passwords: A Framework for Comparative Evaluation of Web Authentication Schemes / J. Bonneau та ін. *2012 IEEE Symposium on Security and Privacy (SP)*, м. San Francisco, CA, USA, 20–23 трав. 2012 р. 2012. URL: https://doi.org/10.1109/sp.2012.44 (дата звернення: 14.12.2025).

34. Toth P. NIST MEP cybersecurity self-assessment handbook for assessing NIST SP 800-171 security requirements in response to DFARS cybersecurity requirements. Gaithersburg, MD : National Institute of Standards and Technology, 2017. URL: https://doi.org/10.6028/nist.hb.162 (дата звернення: 14.12.2025).

35. Вступ до Machine Learning: знайомство з моделями. DOU. URL: https://dou.ua/lenta/articles/introduction-machine-learning-1/ (дата звернення: 01.12.2025).

36. Олексюк В. П. Єдина система автентифікації як крок до створення освітнього простору загальноосвітнього навчального закладу. Науковий часопис

Національного педагогічного університету імені М. П. Драгоманова. Серія 2: Комп’ютерно-орієнтовані системи навчання. 2012. № 13 (20). С. 187-192.