Вінницький національний технічний університет

Факультет менеджменту та інформаційної безпеки

Кафедра менеджменту та безпеки інформаційних систем

БАКАЛАВРСЬКА ДИПЛОМНА РОБОТА

на тему:

Розробка програми комплексної автентифікації до системи захисту звикористанням ідентифікації за допомогою зліпка обличчя та голосу.

Виконав: студент 4 курсу, гр.2КІТС-20б
спеціальності 125– Кібербезпека
Освітня програма – Кібербезпека
інформаційних технологій та систем
(шифр і назва напряму підготовки, спеціальності)

Геник О.І.
(прізвище та ініціали)

Керівник: к.т.н., доц., каф. МБІС
Грицак А.В.
(прізвище та ініціали)
«6» червня 2024 р.

Рецензент: к.т.н., доц., доцент каф. ОТ
Томчук М.А.
(прізвище та ініціали)
«6» червня 2024 р.

Допущено до захисту
Голова секції УБ кафедри МБІС
д.т.н., проф. Яремчук Ю.Є.
“6” червня 2024р.

Вінниця ВНТУ – 2024

Вінницький національний технічний університет
Факультет менеджменту та інформаційної безпеки
Кафедра менеджменту та безпеки інформаційних систем

Рівень вищої освіти І-й (бакалаврський)
Галузь знань 12 – Інформаційні технології
Спеціальність 125 – Кібербезпека
Освітньо-професійна програма - Кібербезпека інформаційних технологій та систем

ЗАТВЕРДЖУЮ
Голова секції УБ, кафедра МБІС
д.т.н., проф. Яремчук Ю.Є.
" 22 " березня 2024 р.

З А В Д А Н Н Я
на бакалаврську дипломну роботу студенту
Геник Олексій Ігорович
(прізвище, ім`я, по-батькові)

1. Тема роботи Розробка програми комплексної автентифікації до системи захисту звикористанням ідентифікації за допомогою зліпка обличчя та голосу
Керівник роботи Грицак Анатолій Васильович к.т.н., доц. каф. МБІС
(прізвище, ім`я, по-батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від " 11 " березня 2024 року № 80
2. Термін подання студентом роботи 6 червня 2024 р
3. Вихідні дані до роботи Метою роботи є розробка програми для комплексної автентифікації в системі захисту з використанням ідентифікації за допомогою зліпка обличчя та голосу.
4. Зміст текстової частини: Для виконання мети дипломної роботи було проведено глибокий аналіз методів автентифікації користувачів за допомогою біометричних даних, зокрема розпізнавання обличчя та голосу. Було розглянуто актуальність використання таких методів у сучасних умовах кібербезпеки, їх переваги та недоліки, а також проаналізовано існуючі системи-аналоги. Також було розроблено програмний продукт, який здійснює комплексну автентифікацію для системи захисту з використанням ідентифікації за допомогою зліпка обличчя та голосу.
5. Перелік ілюстративного матеріалу (з точним зазначенням обов`язкових креслень)
У першому розділі бакалаврської дипломної роботи міститься 8 р другому розділі – 4 рисунки, а в третьому розділі – 17 рисунків.
6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Розділ I	Грицак Анатолій Васильович к.т.н., доц. каф. МБІС		
Розділ II	Грицак Анатолій Васильович к.т.н., доц. каф. МБІС		
Розділ III	Грицак Анатолій Васильович к.т.н., доц. каф. МБІС		

7. Дата видачі завдання 22 березня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Аналіз предметної області	23.03.2024	30.03.2024	Виконано
2	Аналіз методів розпізнавання обличчя та голосу	01.04.2024	10.04.2024	Виконано
3	Аналіз варіантів роботи модулів	11.04.2024	17.04.2024	Виконано
4	Дослідження інформаційних джерел	18.04.2024	23.04.2024	Виконано
5	Постановка задач розробки програмного продукту	24.04.2024	30.04.2024	Виконано
6	Розробка алгоритму роботи	30.04.2024	05.05.2024	Виконано
7	Реалізація програмного продукту	06.05.2024	17.05.2024	Виконано
8	Тестування програмного продукту	18.05.2024	24.05.2024	Виконано
9	Оформлення матеріалів до захисту БДР	25.05.2024	31.05.2024	Виконано

Студент (підпис) Геник О.І. (прізвище та ініціали)

Керівник роботи (підпис) Грицак А.В. (прізвище та ініціали)

Дипломна робота: 87 с., 25 рисунків, 7 таблиць, 18 джерел, 4 додатки.

Об'єкт дослідження: програмне забезпечення для комплексної автентифікації до системи захисту з використанням ідентифікації за допомогою зліпка обличчя та голосу.

Предмет дослідження: процес автентифікації користувачів за допомогою ідентифікації зліпка обличчя та голосу.

Мета дипломного проекту: розробка програмного забезпечення для автоматизації процесу автентифікації користувачів до системи захисту з використанням ідентифікації за допомогою зліпка обличчя та голосу.

У першому розділі проведено аналіз інформації щодо автентифікації користувачів за допомогою зліпка обличчя та голосу. Наведено аналіз процесів розпізнавання людини у задачах кібербезпеки, аналіз існуючих типів автентифікації, цілі автентифікації користувачів комп’ютерних систем, аналіз біометричного розпізнавання за зліпком обличчя та голосу.

У другому розділі проведено аналіз методів розпізнавання за зліпком обличчя та голосу, аналіз існуючих механізмів розпізнавання. На основі проведеного аналізу та мети роботи для виконання практичної частини роботи обрано найбільш задовольняючий потребам завдання метод, наведено аналіз систем-аналогів.

У третьому розділі проведено проектування та практичну реалізацію програмного продукту. Наведено опис процесу попередньої обробки зображень та аудіо, схематично зображено та наведено алгоритми роботи проектованого ПЗ, наведено та описано використані при розробці бібліотеки та модулі, докладно описано процес програмної реалізації проектованого ПЗ, проведено тестування розробленого програмного забезпечення.

Список ключових слів: АВТЕНТИФІКАЦІЯ, РОЗПІЗНАВАННЯ ОБЛИЧЧЯ, РОЗПІЗНАВАННЯ ГОЛОСУ, PYTHON, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ.

ABSTRACT

Thesis: 87 pp., 25 figures, 7 tables, 18 sources, 4 applications.

The object of the study: software for complex authentication to a security system using identification using a cast of a face and voice.

The subject of the study: the process of user authentication using face and voice cast identification.

The aim of the diploma project: development of software for automating the process of user authentication to the security system using identification using a cast of face and voice.

In the first section, the analysis of information on user authentication using facial and voice casts is carried out. The analysis of human recognition processes in cyber security tasks, the analysis of existing authentication types, the purposes of authentication of computer system users, the analysis of biometric recognition based on the geometry of the face and voice is provided.

In the second chapter, an analysis of methods of recognition based on a cast of a face and voice, an analysis of existing recognition mechanisms is carried out. On the basis of the conducted analysis and the purpose of the work, the method most satisfying the needs of the task was chosen for the implementation of the practical part of the work, an analysis of analog systems is given.

In the third section, the design and practical implementation of the software product was carried out. A description of the image and audio preprocessing process is given, the algorithms of the designed software are schematically depicted and given, the libraries and modules used in the development are given and described, the software implementation process of the designed software is described in detail, and the developed software is tested.

List of keywords: AUTHENTICATION, FACE RECOGNITION, VOICE RECOGNITION, PYTHON, SOFTWARE.

СПИСОК УМОВНИХ ПОЗНАЧЕНЬ

AAM (англ. Active appearance model) - моделі зовнішнього вигляду.

ASM (англ. Active shape model) - Активні моделі форми.

AUC (англ. Area Under Curve) - Площа під кривою на графіку.

OpenCV - бібліотека обробки зображень.

FAR (англ. False Acceptance Rate) - Відсоток помилкового збігу біометричних характеристик двох людей.

FERET (англ. Face Recognition Technology) - База даних зображень осіб.

FPR (англ. False Positives Rate) - Помилково позитивні випадки.

FRR (англ. False Rejection Rate) - Відсоток відмови у доступі людині

GUI (англ. Graphical user interface) - Графічний інтерфейс програми.

JPEG (англ. Joint Photographic Experts Group) - Формат цифрового зображення, застосовується до зображення більше 256 кольорів.

LBP-TOP (англ. Local Binary Patterns from Three Orthogonal Planes) - локальні бінарні шаблони трьох взаємно перпендикулярних площин

PGM (Portable Grey Map) - Формат цифрового зображення, що використовується як проміжний для зберігання у відтінках сірого.

ROC (англ. Receiver Operator Characteristic) - Крива, що використовується для оцінки результатів роботи алгоритму класифікації

TPR (англ. True Positives Rate) - Істинно позитивні випадки.

БД - База даних.

ПЗ - Програмне забезпечення.

СКД - Системи контролю доступу.

СППРО - Спосіб підтвердження справжності об'єкта, що розпізнається.

ЗМІСТ

ВСТУП

Актуальність - Актуальність даної роботи обумовлена тим, що складається тенденція до збільшення кількості атак та зламів приватних даних користувачів інтернету. Методи та технології злому особистих кабінетів користувачів регулярно удосконалюються, та існуючі алгоритми та системи автентифікації не завжди повністю виконують поставлену задачу. Однак розвиток технологій розпізнавання користувача за біометричними даними, такими як розпізнавання за зліпком обличчя та голосу, надає нові можливості для підвищення рівня безпеки та надійності систем автентифікації.

Розпізнавання обличчя є одним із найбільш перспективних напрямків у галузі біометричної ідентифікації завдяки своїй високій точності та зручності використання. Застосування голосової автентифікації додатково підвищує рівень безпеки, оскільки поєднання двох біометричних факторів значно ускладнює можливість несанкціонованого доступу. Це робить розробку та впровадження комплексної системи автентифікації на основі розпізнавання обличчя та голосу дуже актуальною у наш час.

В сучасному світі, де інформаційна безпека є пріоритетом, використання біометричних даних для автентифікації користувачів є необхідним кроком для забезпечення захисту особистої інформації та попередження кіберзлочинів. Розробка програмного забезпечення для комплексної автентифікації користувачів, яке об'єднує ідентифікацію за допомогою зліпка обличчя та голосу, сприяє підвищенню надійності та ефективності систем захисту.

Це обумовило **мету дослідження**, яка полягає в розробці та удосконаленні процесу автентифікації користувачів інформаційних систем шляхом розробки та впровадження програмного забезпечення для комплексної автентифікації з використанням зліпка обличчя та голосу.

Для досягнення поставленої мети було поставлено такі **завдання**:

- Проаналізувати типи автентифікації, існуючі методи, інструменти, механізми авторизації користувачів комп'ютерних систем за зліпком обличчя та голосу.
- Провести аналіз та вибір інструментів реалізації проектованого програмного забезпечення.
- Провести проектування та програмну реалізацію програми автентифікації користувачів комп'ютерних систем з використанням зліпка обличчя та голосу.
- Провести тестування розробленого програмного забезпечення.

Об’єкт дослідження: програмне забезпечення для комплексної автентифікації до системи захисту з використанням ідентифікації за допомогою зліпка обличчя та голосу.

Предмет дослідження: процес автентифікації користувачів за допомогою ідентифікації зліпка обличчя та голосу.

Методологічна основа дослідження: загальнонаукові та спеціальні методи, які дозволили вивчити предмет та об’єкт дослідження, дослідити напрями та шляхи оптимізації процесу автентифікації користувачів комп'ютерних систем за допомогою зліпка обличчя та голосу.

Практичне значення отриманих результатів: використання розробленого програмного забезпечення дозволить підвищити якість автентифікації користувачів комп'ютерних систем та покращити систему розпізнавання поточного користувача системи.

1. АНАЛІЗ МЕТОДІВ РОЗПІЗНАВАННЯ ГОЛОСУ І ОБЛИЧЧЯ

1.1 Аналіз процесів розпізнавання людини у задачах кібербезпеки

У зв'язку з тим, що у всьому світі почастішали випадки тероризму, для забезпечення кібербезпеки користувачів у різних сферах все частіше використовується ідентифікація людини. Процес розпізнавання, особливо за допомогою розпізнавання обличчя та голосу з кожним днем набуває все більш великого значення. Розв'язання задачі розпізнавання особи включає в себе етапи отримання зображення, попередньої обробки, виявлення осіб та ідентифікації з урахуванням виявлених особливостей.

Вирішення завдання виявлення осіб особливо важливе при використанні систем відеоспостереження (таких, як CCTV) та в охоронних комплексах. У зв'язку із зростанням обчислювальної потужності персональних комп'ютерів та мобільних пристроїв, виявлення осіб набирає популярність як засіб організації людино-машинної взаємодії. Соціальні мережі, такі як Facebook, виявляють особи в завантажених користувачем фотографіях і пропонують асоціювати їх з обліковим записом користувача в мережі.

Також, існують програми з використанням «доповненої реальності», такі як відеоігри, де гравець може взаємодіяти з об'єктами віртуального світу за допомогою рухів та жестів, що фіксуються камерою [1].

На сьогоднішній день сфера застосування алгоритмів виявлення осіб динамічно розвивається. Дані алгоритми знаходять застосування в різних вбудованих (embedded) системах, а умови застосування даних систем зумовлюють суттєві відмінності в якості зображень. Вимоги обробки в режимі реального часу унеможливлюють пост-обробку зображень або залучення оператора для виконання цього процесу, тому важливо розробляти стійкі до дефектів зображень алгоритми, що мають обчислювальну ефективність.

Таким чином, завдання виявлення осіб є одним із найприорітетніших напрямів розвитку алгоритмів машинного навчання та комп'ютерного зору.

1.2 Аналіз методів розпізнавання за зліпком обличчя та голосу

Дуже важлива частина систем автоматичного розпізнавання обличчя - безпосередньо самі алгоритми розпізнавання людини. На сьогоднішній день таких алгоритмів багато, і кожен із них має свою специфіку, свою швидкість роботи та свою надійність розпізнавання.

Алгоритми розпізнавання поділяються на дві категорії, залежно від застосовуваної технології розпізнавання – двомірні, у яких розпізнавання відбувається за зліпком обличчя (2D-технології) та тривимірними, в яких розпізнавання відбувається за будовою черепа (3D-технології).

Системи 2D-розпізнавання працюють із «плоскими», двомірними зображеннями і розпізнають обличчя, аналізуючи його текстуру та ділянки особи з високою контрастністю, тому при порушенні освітлення чи становища особи його розпізнавання дуже ускладнено.

Системи 3D-розпізнавання більш стійкі до таких змін, так як під час створення моделі особи враховуються особливості будови черепа. Але тим не менш, хоч системи 3D і мають такий великий плюс, через відсутність можливості обслуговувати велику кількість користувачів у режимі ідентифікації та через невисоку швидкість, 3D-технології поки не отримали широкого застосування. Також, 3D-технології розпізнавання вимагають великих обчислювальних ресурсів, та вартість обладнання для таких систем набагато вище, ніж у 2D.

На сьогоднішній день часто використовується кілька ефективних алгоритмів розпізнавання, всі вони засновані або на значенні пікселів, або на характерних точках. Нижче наведено короткий опис кожної з цих підгруп.

Методи, що ґрунтуються на значеннях пікселів, використовують для розпізнавання виявленого обличчя колір або яскравість. Найпростішими подібними методами є:

Eigenfaces – алгоритм, запропонований у 1991 році Метью Терком та Алексом Пентландом і здобув широку популярність як перший успішний

методу розпізнавання осіб. Основною ідеєю алгоритму Eigenfaces є представлення відмітних характеристик зображення обличчя за допомогою двовимірних зображень у сірих градаціях. Для ідентифікації особистості алгоритм порівнює отриману особу з кадру, з уже зареєстрованим шаблоном у базі, та визначає коефіцієнт відмінності. Цей коефіцієнт відмінності між

шаблонів і показує ступінь схожості.

Алгоритм EigenFaces показує високі результати при використанні в

добре освітлених приміщеннях і коли є можливість сканування обличчя в

фас щодо змін умов використання даного алгоритму, показники його ефективності різко знижуються.

Fisherfaces – нащадок Eigenface, що забезпечує більш високу точність

розпізнавання при змінах висвітлення чи виразу обличчя. На відміну від

методу eigenfaces, в основі Fisherfaces лежить лінійний дискримінантний аналіз LDA, а саме лінійний дискримінант Фішера.

Дія алгоритму Fisherfaces заснована на пошуку проекції даних, при

якої класи зображень осіб максимально розділяються. Ця відмінність з

Eigenface дозволяє вирішити проблему високої чутливості до змін освітлення.

Локальні бінарні шаблони (Local Binary Pattern, LBP) – простий у

реалізації та ефективний метод розпізнавання осіб. При розпізнаванні методом

LBP, кожному пікселю зображення за допомогою функції надається

значення яскравості, що описує його околицю. Отримане зображення

поділяється на області, для кожної з яких розраховується гістограма.

Далі гістограми конкатенуються та порівнюються за допомогою методів

машинного навчання. У класичному варіанті використовується метод найближчого сусіда [12].

На відміну від EigenFaces, алгоритм стійкий до монотонних змін освітлення, що робить його придатним для розпізнавання осіб у системах

обробки у реальному часі.

Методи, засновані на характерних точках, на відміну від попередніх, не

оцінюють яскравості пікселів, а використовують координати характерних точок на зображенні. Такими характерними точками можуть бути, наприклад, центри очей, становище носа, лінія брів, рота тощо. До цього класу методів відносяться активні моделі зовнішнього вигляду та активні моделі форми.

Активні моделі зовнішнього вигляду (Active Appearance Models, AAM) – це статистичні моделі зображень, які шляхом різного роду деформацій можуть бути підігнані під реальне зображення. Даний тип моделей в двовимірному варіанті був запропонований Тімом Кутсом та Крісом Тейлором у 1998 році. Активна модель зовнішнього вигляду містить два типи параметрів: параметри, пов'язані з формою (параметри форми), та параметри, пов'язані з статистичною моделлю пікселів зображення або текстурою (параметри зовнішнього вигляду). Перед використанням модель має бути навчена на безлічі заздалегідь розмічених зображень. Розмітка зображень виготовляється вручну.

Активні моделі форми (Active Shape Models, ASM) враховують статистичні зв'язки у взаємному розташуванні антропометричних точок. на

кожному зображенні вибірки експерт розмічає розташування антропометричних точок. Для того, щоб привести координати на всіх

зображення до єдиної системи зазвичай виконується так званий узагальнений прокрустів аналіз, в результаті якого всі точки наводяться до одного масштабу та центруються. Далі для всього набору образів обчислюється середня форма та матриця коваріації[13].

1.3 Аналіз існуючих механізмів розпізнавання за зліпком обличчя

1.3.1 Аналіз сукупностей антропометричних точок особи (Geometrical Features)

Той факт, що люди істотно відрізняються своєю зовнішністю і, зокрема, рисами обличчя очевидний. Так, наприклад, розташування очей та їх дрібні характеристики різняться навіть у близнюків. Тому не дивно, що історично перший похід до вирішення проблеми автоматичної ідентифікації людини за зображенням її обличчя було засновано на виділенні та порівнянні деяких антропометричних характеристик особи. Цей підхід давно використовується в практичній криміналістиці, проте виміри та порівняння виконувались вручну.

Основна проблема, з якою доводиться стикатися розробникам систем розпізнавання при використанні даного підходу, - вибір сукупності характерних точок, однозначно описують конкретне людське обличчя. При цьому необхідно враховувати наступні вимоги: точки на обличчі або риси особи, на яких ґрунтується ідентифікація, не повинні закриватися зачіскою, бородою, маскою і т.п.; для забезпечення незалежності процесу розпізнавання від мас-штабу зображення доцільно описувати систему ідентифікаційних точок у відносинах між ними; обрана система точок повинна забезпечувати відносну стійкість процесу розпізнавання при незначній зміні ракурсу зйомки (легкий поворот голови, нахил, зміна виразу обличчя і т.д.); кількість характерних точок системи, що задовольняє вищевикладеним вимогам, має бути мінімальною, тому що обчислювальна вартість алгоритмів зазвичай пропорційна їх кількості.

До теперішнього часу є багато робіт, присвячених дослідженням розпізнавання за допомогою різних сукупностей характерних точок та аналізу ефективності роботи систем, побудованих на їх основі.

1.3.2. Метод власних векторів (EigenVectors, EigenFaces)

Наступним, найбільш опрацьованим після методу аналізу антропометричних характеристик особи можна назвати метод власних векторів (іноді його називають методом головних компонент осіб). Він є прикладом того, як математичні методи (метод аналізу основних компонентів), що успішно застосовувалися в інших областях, виявилися ефективно адаптованими до розпізнавання людей за їх портретами.

Будь-яке цифрове зображення може бути представлене у вигляді вектора у просторі ознак. Якщо зображення описується w×h пікселями, розмірність найпростішого векторного простору, до якого даний вектор належить, дорівнюватиме добутку w на h і, відповідно, базис подібного векторного простору складатиметься з w×h векторів. Однак у зв'язку з тим, що всі людські особи схожі між собою (овальна форма з носом, ротом, очима і т.д.), всі вектори, що описують зображення облич, будуть розміщуватися у вузько обмеженій області даного векторного простору. Тому при рішенні завдання розпізнавання людей портретом опис і зберігання всього векторного простору не раціонально.

Таким чином, постає питання побудови простору меншої розмірності, в якому зображення людських осіб описуються більш компактно. Одним з варіантів є простір, базисними векторами якого служать головні компоненти всіх зображень осіб, що містяться в ньому. Розмірність такого простору заздалегідь визначити неможливо, але вона набагато менша за розмірність векторного простору всіх зображень.З вищесказаного випливає, що головною метою методу аналізу принципових компонентів є значне зменшення розмірності простору ознак таким чином, щоб воно якнайкраще описувало «типові» образи, що належать безлічі портретів.

У разі застосування даного методу для ідентифікації осіб такими образами будуть служити навчальні зображення. Іншими словами, за допомогою аналізу основних компонентів вдається виявити всілякі мінливості в навчальному

наборі зображень осіб і описати цю мінливість за допомогою декількох змінних. Ці змінні є w×h -розмірні вектори, які називаються власними. Якщо перетворити подібні вектори на зображення, то одержувані картинки будуть відображати головні компоненти представленої навчальної множини (також звані власними особами). Таким чином, за рахунок зниження розмірності простору базових векторів, в якому знаходяться зображення, домагаються хороших показників як у швидкості, так і в точності розпізнавання [14]. Наведемо схему роботи даного методу на рисунку 1.1.

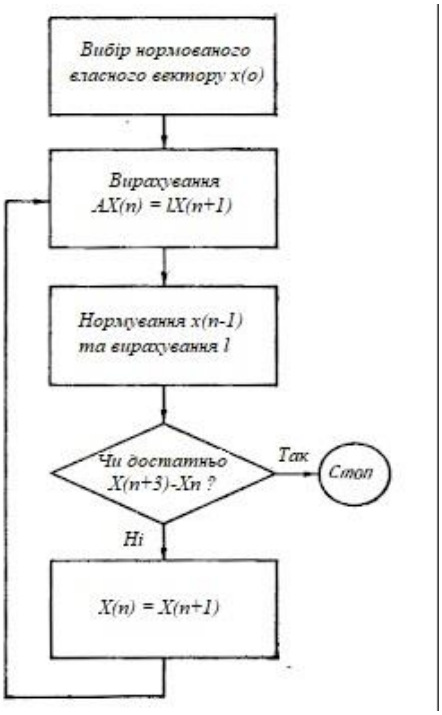


Рисунок 1.1 – Блок схема роботи методу власних векторів

1.3.3 Імовірнісний підхід (Probability estimation)

Подібно до попереднього методу в імовірнісних моделях також використовується навчальний набір. У цьому формуються два класи із усіх варіантів уявлення об'єктів: внутрішньооб’єктної і до зовнішньої мінливості, тобто відбираються ознаки, якими всі портрети поділяються на два класи: 1) портрет даної людини, 2) всі інші портрети. Функції щільності ймовірності для кожного класу оцінюються за допомогою згаданої вище навчальної множини

і згодом використовуються для обчислення міри схожості, яка ґрунтується на отриманих досвідченим шляхом ймовірностях. Крім того, для отримання більш точних результатів іноді використовується модель ймовірності деякого фізичного процесу, за допомогою якої і формується остаточна міра схожості двох зображень [15]. Наведемо схему роботи даного методу на рисунку 1.2.

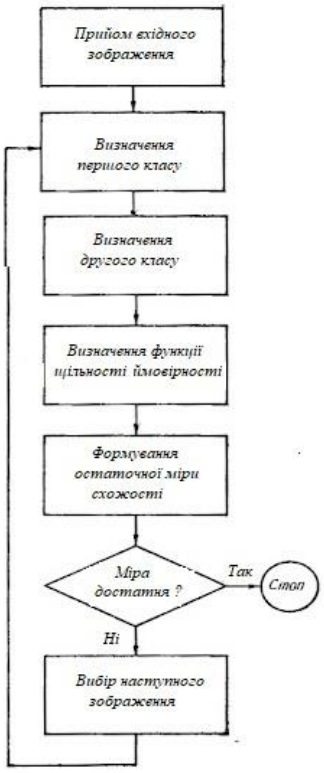


Рисунок 1.2 – Блок схема роботи методу ймовірнісного підходу

1.3.4 Метод зіставлення з еталоном (Template Matching)

У цьому підході процес розпізнавання розбивається на частини, відповідні окремим характеристикам особи. Кожна фотографія, що надходить на вхід системи, що розпізнає, повинна являти собою фронтальне зображення особи людини з певною для конкретної бази даних кількістю масок, що представляють основні для ідентифікації регіони особи (наприклад, очі, ніс, рот і нижня частина особи). Крім того, розташування даних масок повинні бути однаково нормалізовані (наприклад, щодо положення очей) для всіх зображень у базі даних. Під час процесу розпізнавання, коли частини вхідного зображення по черзі порівнюються з частинами зображення, що зберігається в

базі, використовується вектор, що відображає результат порівняння в балах (один бал за кожну рису обличчя, що збіглася) і обчислюється шляхом нормалізованої взаємної кореляції (втім, методи порівняння можуть бути різними). Після чого вхідне зображення класифікується відповідно до максимально набраних балів. Є також деякі різновиди даного підходу, наприклад, з еталонами, що змінюються в процесі порівняння [16]. Наведемо схему роботи даного методу на рисунку 1.3.

Рисунок 1.3 – Блок схема роботи методу зіставлення з еталоном

1.4 Аналіз існуючих механізмів розпізнавання за голосом

У сучасному світі технології розпізнавання голосу набули широкого поширення завдяки їх високій точності та зручності використання. Розпізнавання голосу використовується в різних сферах, включаючи безпеку, комунікації та персоналізовані послуги. Розглянемо три основні методи розпізнавання голосу: метод мел-частотних кепстральних коефіцієнтів (MFCC), метод глибоких нейронних мереж (DNN) та метод.

1.4.1 Метод мел-частотних кепстральних коефіцієнтів (MFCC)

Мел-частотні кепстральні коефіцієнти (MFCC) є одним із найбільш поширених методів для розпізнавання голосу завдяки своїй здатності точно моделювати фізіологічні особливості людського слуху. Цей метод базується на перетворенні звукового сигналу у набір коефіцієнтів, які представляють короткочасні спектральні властивості звуку. Основна ідея полягає у тому, що людське вухо сприймає частоти нелінійно, і MFCC враховує цю особливість, використовуючи мел-шкалу для аналізу частотного спектру сигналу.

Процес розпізнавання голосу за допомогою MFCC починається з попередньої обробки сигналу, яка включає нормалізацію амплітуди та видалення шуму. Це дозволяє підвищити якість сигналу та забезпечити більш точне витягнення характеристик. Далі сигнал розбивається на невеликі відрізки часу, звані вікнами, оскільки звуковий сигнал змінюється з часом, і аналіз невеликих відрізків дозволяє захопити ці зміни.

Після розбиття сигналу на вікна до кожного з них застосовується швидке перетворення Фур'є (FFT), яке перетворює часовий сигнал у частотну область. Потім до отриманого спектру застосовується мел-шкала, яка моделює нелінійну чутливість людського вуха до різних частот. Це означає, що низькі частоти аналізуються з високою роздільною здатністю, тоді як високі частоти об'єднуються у ширші смуги.

Наступним кроком є обчислення кепстральних коефіцієнтів. Це досягається шляхом застосування дискретного косинусного перетворення (DCT) до логарифмів енергій, отриманих з мел-фільтрів. Результат цього перетворення - набір коефіцієнтів, які називаються мел-частотними кепстральними коефіцієнтами (MFCC). Ці коефіцієнти є компактним представленням спектральних характеристик голосу і можуть бути використані для розпізнавання мовця.

Переваги методу MFCC включають високу точність і стійкість до змін у вимовлянні звуків, а також його здатність працювати в реальному часі. Проте,

метод може бути чутливим до шуму та змін у навколишньому середовищі, що може знизити його точність. У таких випадках можуть застосовуватися додаткові методи попередньої обробки сигналу та фільтрації шуму.

1.4.2 Метод глибоких нейронних мереж (DNN)

Глибокі нейронні мережі (DNN) є сучасним підходом до розпізнавання голосу, який демонструє високу точність завдяки здатності автоматично виділяти складні характеристики з даних. DNN використовують багатошарову архітектуру для навчання на великій кількості даних, що дозволяє досягти високої точності розпізнавання.

Процес розпізнавання голосу за допомогою DNN починається зі збору та підготовки великого обсягу даних для навчання мережі. Архітектура нейронної мережі складається з кількох шарів: вхідного шару, одного або більше прихованих шарів та вихідного шару.

Навчання мережі здійснюється за допомогою алгоритмів оптимізації, таких як градієнтний спуск, які налаштовують ваги зв'язків між нейронами на основі помилки прогнозування. Під час прогнозування DNN використовує навчені ваги для класифікації вхідних звукових сигналів. Вхідні дані проходять через всі шари мережі, і кожен шар виділяє все більш абстрактні та складні характеристики звуку. Це дозволяє мережі адаптуватися до різних умов і забезпечує високу точність розпізнавання навіть у складних середовищах.

Основні переваги DNN включають здатність адаптуватися до різних умов, високу точність і здатність автоматично виділяти релевантні характеристики з даних. Однак, DNN вимагають значних обчислювальних ресурсів та великого обсягу даних для навчання, що може бути недоліком у деяких застосуваннях.

1.4.3 Метод динамічного часового вирівнювання (DTW)

Метод динамічного часового вирівнювання (DTW) є потужним інструментом для порівняння часових послідовностей, таких як голосові сигнали, навіть якщо ці послідовності мають різну тривалість або вимовлені з

різною швидкістю. Цей метод знаходить широке застосування в розпізнаванні мови, аналізі рухів, біометричних дослідженнях та інших областях, де важливо зіставити дві часові послідовності.

DTW був розроблений для вирішення проблеми вирівнювання часових рядів, де необхідно враховувати можливі нелінійні зміни в часі. Наприклад, два зразки одного і того ж слова можуть мати різну тривалість або вимовлені з різною інтонацією, і метод DTW дозволяє знайти оптимальне вирівнювання між цими зразками. Процес розпізнавання голосу за допомогою DTW починається з розбиття записаного голосового сигналу на короткі сегменти (фрейми). Кожен фрейм може бути описаний набором характеристик, таких як мел-частотні кепстральні коефіцієнти (MFCC), які представляють спектральні властивості сигналу в кожному фреймі. Далі для кожного фрейму зразка голосу обчислюється відстань до кожного фрейму еталонного сигналу. Найчастіше використовується евклідова відстань, яка вимірює різницю між двома векторами характеристик.

Використовуючи матрицю відстаней, метод DTW обчислює кумулятивну матрицю відстаней, яка представляє найкоротший шлях вирівнювання між двома послідовностями. Кожна клітинка цієї матриці містить мінімальну відстань від початку до поточної точки. Після цього DTW знаходить оптимальний шлях через матрицю кумулятивних відстаней, який мінімізує загальну відстань між двома послідовностями. Цей шлях визначає, які фрейми зразка відповідають яким фреймам еталонного сигналу. Після знаходження оптимального шляху система обчислює загальну відстань вирівнювання, яка використовується для оцінки схожості між двома голосовими зразками. Якщо ця відстань менша за певний поріг, голоси вважаються схожими, і автентифікація вважається успішною.

DTW має кілька ключових переваг. По-перше, він може ефективно обробляти часові спотворення, що робить його стійким до змін у швидкості вимови та інтонації. Це особливо важливо в реальних умовах, де користувачі

можуть говорити з різною швидкістю та виразом. По-друге, DTW дозволяє точніше зіставити сегменти сигналів, що підвищує точність розпізнавання.

Однак DTW також має деякі недоліки. Він може бути чутливим до шуму, оскільки кожна помилка у фреймі може вплинути на загальне вирівнювання. Крім того, обчислення матриці кумулятивних відстаней може бути обчислювально інтенсивним, що вимагає значних ресурсів для великих наборів даних.

1.5 Детальний аналіз обраного методу розпізнавання за зліпком обличчя

Для розробки програмного забезпечення під час виконання дипломної роботи було обрано метод власних векторів.

Як уже пояснювалося вище, суть методу полягає в тому, що отриманий один раз на основі представницької навчальної вибірки набір власних векторів або осіб (рис.1.7) використовується при кодуванні решти зображень, які подаються для зберігання в базі виваженою комбінацією цих власних векторів (рис.1.8). Іншими словами, використовуючи обмежену кількість власних векторів, можна отримати поліпшену апроксимацію до вхідного зображення, яка потім зберігається в базі даних у вигляді вектора ваг, що служить одночасно ключем пошуку.

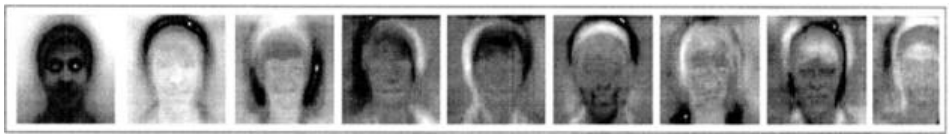


Рисунок 1.7 – Основні перші компоненти особи людини

Обгрунтування вибору однієї з кількох можливих наборів власних векторів наводиться в [8]. На відносно великій базі в 1316 зображень 504 чоловік продемонстровано перевагу систематизованого підходу до вибору простору власних векторів – отримана точність розпізнавання – 95-99% залежно від виду дослідів.

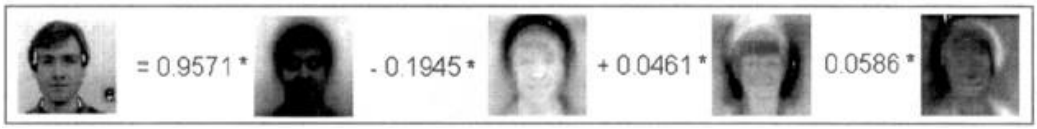


Рисунок 1.8 - Портрет, представлений у базисі головних компонентів

Наведемо принцип роботи методу з формулами розрахунку:

Нехай є набір зображень, кожен з яких описується вектором x^k_i , де i – номер зображень (i = 1, 2, 3, ..., Mk , k = 1, 2, ..., K). Розмірність вектора x^k_i дорівнює числу пікселів зображення (N). Таким чином, весь набір зображень можна подати у вигляді матриці X, стовпцями якої є вектори x^k_i . Розмірність простору зображень визначається результатом *N* x *M*.

Позначимо матрицю відцентрованих зображень як X^0. Стовпцями X^0 є вектори

$$\overline{x}_i^k = x_i^k - m \text{ , де}$$

$$m = \frac{1}{M}\sum_k^K \sum_i^{M_k} x_i^k \_$$

середньоарифметичний вектор зображень,

$$M = \sum_{k=1}^{K} M_k.$$

Підхід МГК+ЛДА складається з двох етапів: на першому етапі застосовується МГК для зменшення розмірності від *N* до *p* і отримання матриці V_{pca} , яка формується із власних векторів рівняння:

$$(\mathbf{A} - \lambda \mathbf{I}) v_0^{pca} = 0,$$

де A - коваріаційна матриця розмірністю N N, індекс «T» означає транспонування матриці, I – одинична матриця, $v^{pca}{}_0$ - Власний вектор, **λ** - власне значення. У зв'язку з тим, що матриця A має високий порядок, обчислення власних векторів становить значні труднощі. Тому ефективніше обчислювати основні компоненти V_{pca} за матрицею власних векторів U_{pca} [9], які визначаються шляхом вирішення рівняння:

$$(\mathbf{A} - \lambda \mathbf{I}) u_0^{pca} = 0,$$

де A - матриця Грама розмірністю *M x M, u* $^{pca}{}_0$ – власний вектор.

Слід зазначити, що власні значення збігаються. В зв'язку з тим що M значно менше N, можна істотно знизити трудомісткість обчислення матриці V_{pca}. На другому етапі, застосовується ЛДА з метою знаходження таких лінійних комбінацій ознак, які найкраще поділяють класи зображень облич. Метою ЛДА є отримання матриці перетворення W_{lda}, яка мінімізує внутрішньокласову та максимізує міжкласову відстань у просторі ознак

$$\mathbf{W}_{lda} = \arg\max_{\mathrm{w}} \frac{\left|\mathbf{W}^T \mathbf{V}_{pca}^T \mathbf{A}_b \mathbf{V}_{pca} \mathbf{W}\right|}{\left|\mathbf{W}^T \mathbf{V}_{pca}^T \mathbf{A}_\omega \mathbf{V}_{pca} \mathbf{W}\right|} = \arg\max_{\mathrm{w}} \frac{\left|\mathbf{W}^T \mathbf{A}_b' \mathbf{W}\right|}{\left|\mathbf{W}^T \mathbf{A}_\omega' \mathbf{W}\right|}.$$

Тут A_b - коварійна матриця міжкласових відмінностей, Aω - коварійна матриця внутрішньокласових відмінностей. Стовпцями матриці W_{lda} є власні вектори , які виходять в результаті розв'язування рівняння:

$$(\mathbf{A}_b' - \lambda \mathbf{A}_\omega') w_0^{lda} = 0.$$

У результаті рішення цього рівняння визначається матриця дискримінантних компонентів, стовпцями якої є власні вектори рівняння з найбільшими власними значеннями. Тюрк і Пентланд [9] провели комплексне дослідження даного методу на базі даних, що складається з портретів 16 осіб, зображення яких були отримані за різних умов освітлення, при зйомці з різних відстаней, при різних поворотах голови - всього 2500 фотографій. Однак отримані зображення були однакові за такими параметрами як міміка, деталі обличчя (борода, окуляри і т.д.). При зміні освітлення, ракурсу зйомки і масштабу точність розпізнавання склала 95, 85 і 64% відповідно. Крім того, середина особи виділялася для зменшення негативного ефекту від можливих змін у зачісці та фоні. За швидкістю роботи реалізована авторами на робочій станції SUN 3/160 система наближалася до режиму реального часу.

Найбільш представницькі за обсягом експерименти були описані Пентландом з колегами в [10]. Вони досліджували ефективність застосування

методу при роботі з великими базами даних, одна з яких складалася з 7562 зображень, що належать майже трьом тисячам осіб. На даний момент це найбільша база зображень осіб, про роботу з якою було оголошено. У ході експериментів два десятки власних векторів були отримані з 128 випадково обраних навчальних зображень. На додаток до векторів в базі зберігалася інформація про поле, расу, приблизний вік і вираз обличчя. На відміну від поліцейських баз даних, де використовуються лише два зображення одного об'єкта - в фас і профіль, в експериментальній базі було багато зображень, що належать одній людині, але відрізняються різними виразами обличчя, зачіскою і т.д.

Однак експерименти, що проводяться з базою, можна назвати обмеженими - перевірялася лише можливість інтерактивного пошуку по всій базі даних. Експерт запитував систему представити зображення осіб певного типу (наприклад, «негритянка 50 років»). Зображення, які задовольняли даним запитам, видавалися на екран групами по 21. Користувач вибирав серед них одне і система видавала особи з бази, які були найбільш схожі на необхідне, в порядку зменшення подібності. Під час експериментів із 200 обраними зображеннями коефіцієнт розпізнавання становив 95%, тобто. для 180 зображень найбільш схожою була обрана фотографія тієї ж персони. Таким чином, враховуючи той факт, що обране зображення спочатку розкладалося на власні вектори, а потім по них проводився пошук, помилка розпізнавання прямо залежить від похибки розкладання як обраного, так і що зберігаються в базі зображень.

Для оцінки точності розпізнавання як функції від раси були проведені досліди із зображеннями біло- та чорношкірих, а також азіатських осіб чоловічої статі. Для білих та чорних точність розпізнавання була 90 та 95% відповідно, і лише 80% – для азіатів. У роботі [12] Могхаддам і Пентланд оголосили про отримання хороших результатів при експериментах з базою даних FERET-тільки одне неправильне визначення на 150 фронтальних зображень.

Система проводила розширену передобробку за положенням голови, визначенням рис обличчя, а також нормалізацію по зліпку обличчя, освітленості, контрастності, масштабу і повороту.На основі вищесказаного можна відзначити, що хоча алгоритм розпізнавання зображень, представлених комбінацією власних осіб, і є відносно швидким, простим і практичним, при його роботі з дуже великими базами даних можуть виникнути проблеми з точністю [18].

1.6 Аналіз існуючих програмних засобів

Перш ніж почати розробляти певну систему, потрібно виконати пошук її аналогів, визначити їх переваги та недоліки, щоб не допустити ті ж помилки та використовувати переваги проаналізованих систем.

Проаналізовано три веб-ресурси, що реалізують функції предметної області авторизації за допомогою розпізнавання людини.

1. Інформаційна система «ZkTeco».
2. Інформаційна система «Privatbank».
3. Інформаційна система «IIdX».

Розглянемо докладніше подані вище аналоги. Першим розглянемо сервіс «ZkTeco».

На рисунку 1.6 зображено інтерфейс головної сторінки сервісу.

Рисунок 1.6 - Інтерфейс головної сторінки сервісу

Даний сервіс представлено у вигляді веб-додатку для продажу комерційних послуг з розпізнавання людини, який можна використувати для процесів автентицікації користувача чи мониторингу за ним. Сервіс надає

можливість технічної підтримки користувачів, має можливість авторизації та реєстрації. Для використання сервісу користувач має придбати послугу на сайті сервісу, після чого співробітники інтегрують процеси розпізнавання у бізнес користувача Наведений сервіс гарантує якісний аналіз фото та захист від помилкового ідентифікування. З мінусів можна виділити дороговизну придбання даного програмного забезпечення та неліквідність використання в малих некорпоративних проектах.

Наступною розглянемо систему «PrivatBank».

На рисунку 1.7 зображено вигляд головної сторінки інформаційної системи.

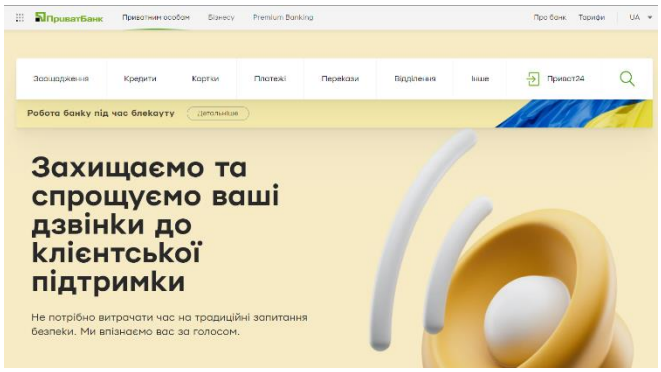


Рисунок 1.7 – Інтерфейс головної сторінки інформаційної системи

Даний сервіс є можливістю авторизуватися у особистому кабінеті банку «Privatbank» за допомогою розпізнавання людини. Ця функція була додана нещодавно і зараз активно тестується. При розпізнаванні алгоритми банку проводять розпізнавання особистості користувача на фото та порівнюють його з існуючим у базі даних користувачем. Якщо особу користувача підтверджено – йому буде надано доступ до особистого кабінету користувача. У зворотньому випадку – користувача буде знову перенаправлено на сторінку автентифікації. Методи, якими проводиться розпізнавання не розголошуються політикою банку.

Останньою розглянемо інформаційну систему «IIdX».

На рисунку 1.8 зображено інтерфейс головної сторінки інформаційної системи.

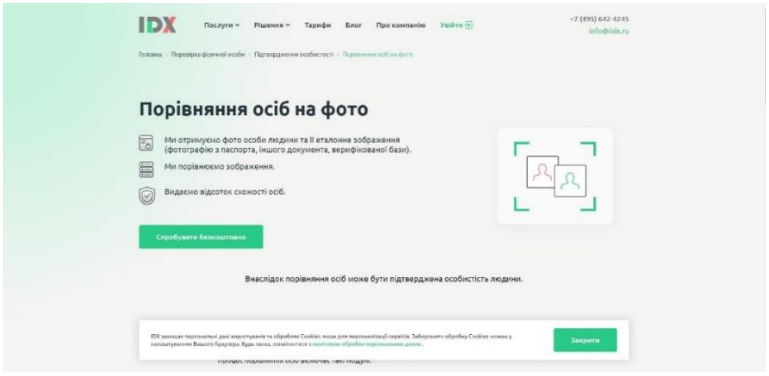


Рисунок 1.8 – Інтерфейс головної сторінки інформаційної системи

Даний сервіс представлено у вигляді API сервісу для автоматизації процесу розпізнавання користувача для бізнесу. Сервіс надає можливість технічної підтримки користувачів, має можливість авторизації та реєстрації. Для використання сервісу користувач має придбати послугу на сайті сервісу та вподальшому інтегрувати її до свого проекту. Сервіс приймає на вхід фото користувача, виконує розпізнавання та повертає результат. З мінусів можна виділити дороговизну придбання даного програмного забезпечення.

1.7 Порівняльний аналіз розглянутих програмних засобів

Для успішної розробки програми комплексної автентифікації до системи захисту з використанням ідентифікації за допомогою зліпка обличчя та голосу необхідно провести порівняльний аналіз існуючих програмних засобів. Такий аналіз дозволяє оцінити сучасні підходи та технології, що використовуються в області біометричної автентифікації, виявити їхні переваги та недоліки, а також визначити найкращі практики, які можна застосувати у нашій розробці.

Основною метою цього аналізу є ідентифікація ключових функціональних можливостей, рівня безпеки, зручності використання та ефективності різних систем. Це дозволить розробити більш надійну та конкурентоспроможну програму автентифікації, яка відповідатиме вимогам сучасних користувачів та стандартам безпеки. У таблиці 1.1 наведено детальний аналіз розглянутих систем, що використовують ідентифікацію людини за допомогою зліпка обличчя та голосу.

Таблиця 1.1 - Порівняльна характеристика систем – аналогів

Назва ПЗ	ZkTeco	Privatbank	IIdX
Тип системи	Веб-додаток для комерційного використання	Веб-додаток для авторизації у банківському кабінеті	API сервіс для автоматизації розпізнавання
Основна функція	Розпізнавання людини для автентифікації та моніторингу	Авторизація за допомогою розпізнавання особистості	Розпізнавання людини через API
Точність ідентифікації	Висока	Висока	Висока
Швидкість обробки даних	Висока	Середня	Висока
Надійність системи	Високий рівень аналізу фото та захист від помилок	Високий рівень (алгоритми не розголошуються)	Високий рівень розпізнавання
Користувацький досвід	Зручний інтерфейс	Зручний інтерфейс, інтегрований з банківським кабінетом	Зручний для розробників, потребує технічних знань для інтеграції

Продовження таблиці 1.1

Переваги	Якісний аналіз фото, захист від невалідної автентифікації	Висока ступінь захисту, зручність використання.	Зручний спосіб впровадження у проект у вигляді API, обширна документація

			щодо взаємодії з сервісом.
Недоліки	Висока вартість, неліквідність для малих проектів	Обмежена прозорість алгоритмів	Висока вартість
Застосування	Підходить для великих корпоративних проектів	Використовується для банківських сервісів	Підходить для бізнес-проектів, що потребують автоматизації розпізнавання

З проведеного аналізу видно, що жодна з наведених систем, що надає в повній мірі проектований функціонал автентифікації має свої недоліки основні з яких це дороговизна придбання для українського ринку, складність використання, не зручний спосіб впровадження сервісу у розроблюване ПЗ, зашифровані алгоритми розпізнавання.

Функції представлених веб – сервісів в більшості зорієнтовані на використання у високозавантажених проектах, де їх використання буде виправданим, та де бюджет використання дозволяє взаємодію з таким дороговартісним ПЗ. Після огляду систем-аналогів було проаналізовано всі переваги та недоліки цих ресурсів. До переваг відносяться такі функції, як автентифікація користувача за допомогою розпізнавання людини, зручний засіб впровадження у проекти у вигляді API а також зручний та інтуітивно зрозумілий середньостатистичному користувачеві інтерфейс користувача.

Основним недоліком є відсутність перерахованих вище функцій, а також не інтуїтивний та складний до розуміння інтерфейс користувача, що ускладнює роботу з системою. Всі вище наведені переваги та недоліки були враховані під час розробки веб-сервісу для автоматизації процесу

автентифікації користувачів компютерних систем за зліпком обличчя та голосом.

1.8 Висновок до розділу 1

У цьому розділі проведено детальний аналіз різних методів автентифікації користувачів за допомогою біометричних даних, зокрема, розпізнавання обличчя та голосу. Було наведено переваги та недоліки кожного методу, що дозволяє вибрати найбільш оптимальний для забезпечення кібербезпеки.

Проаналізовано існуючі механізми розпізнавання обличчя та голосу, включаючи методи сукупностей антропометричних точок, власних векторів, імовірнісні підходи та зіставлення з еталоном для обличчя, а також MFCC, DNN і DTW для голосу. Це дозволило визначити найбільш ефективні та надійні методи для різних сценаріїв використання.

І також було розглянуто актуальність використання методів біометричної автентифікації у сучасних умовах кібербезпеки. Результати аналізу підтвердили, що розпізнавання обличчя та голосу є перспективними напрямками для підвищення безпеки систем та можуть бути ефективно впроваджені у різні галузі.

2. ПРОЕКТУВАННЯ ПРОЦЕСУ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Обгрунтування вдосконалення існуючих методів

Розглянуті у 1 розділі сучасні методи автентифікації користувачів стикаються з багатьма викликами, включаючи зростання кількості кібератак та складність підтримання високого рівня безпеки без зниження зручності для користувачів. Відсутність надійних та одночасно зручних методів автентифікації спонукає до пошуку нових рішень, які поєднують у собі декілька біометричних факторів. У цій роботі пропонується удосконалення існуючих методів автентифікації шляхом поєднання розпізнавання обличчя та голосу, що забезпечує додатковий рівень безпеки та надійності.

Розпізнавання обличчя є однією з найбільш досліджуваних та ефективних технологій біометричної автентифікації. Однак традиційні методи, такі як метод власних векторів (Eigenfaces), мають обмеження у випадках зміни освітлення, кута зйомки та виразу обличчя. Для вирішення цих проблем, у даній роботі використовується бібліотека face_recognition, яка базується на сучасних методах машинного навчання та забезпечує високу точність навіть у складних умовах.

Автентифікація за голосом доповнює розпізнавання обличчя, оскільки голос є унікальним біометричним параметром для кожної людини. Використання бібліотеки speech_recognition для запису голосу та бібліотеки librosa для витягнення характеристик голосу (MFCC) дозволяє забезпечити високу точність розпізнавання. Порівняння евклідової відстані між характеристиками записаного голосу та збереженими даними у базі дозволяє точно ідентифікувати користувача.

Поєднання двох біометричних факторів у системі автентифікації дозволяє значно підвищити рівень безпеки. У той час як зловмисники можуть спробувати обійти систему за допомогою фотографії або запису голосу, комбінація обох методів робить таку атаку практично неможливою.

Використання бібліотеки face_recognition для розпізнавання обличчя та бібліотек speech_recognition і librosa для обробки голосу забезпечує високу точність та швидкість роботи системи. Ці бібліотеки використовують сучасні алгоритми машинного навчання та цифрової обробки сигналів, що дозволяє досягти високої ефективності навіть у складних умовах.

Розроблене програмне забезпечення автоматизує процес автентифікації, роблячи його зручним та швидким для користувачів. Користувачеві достатньо просто показати обличчя та вимовити фразу для входу в систему, що значно підвищує зручність використання без зниження рівня безпеки. Поєднання двох незалежних біометричних параметрів значно підвищує стійкість системи до атак. Навіть якщо один з параметрів буде скомпрометовано, зловмисник не зможе пройти автентифікацію без другого параметра. Це робить систему більш надійною та безпечною у порівнянні з традиційними методами.

Таким чином, удосконалення існуючих методів автентифікації користувачів шляхом поєднання розпізнавання обличчя та голосу дозволяє досягти високого рівня безпеки та зручності. Використання сучасних бібліотек та алгоритмів забезпечує точність та надійність системи, роблячи її ефективним засобом захисту інформації у сучасних умовах. Розроблене програмне забезпечення може бути використане у різних сферах для підвищення рівня безпеки та надійності систем автентифікації.

2.2 Обґрунтування та детальний опис алгоритму роботи проектованого програмного забезпечення

Визначимо та побудуємо алгоритми основних процесів роботи проектованого програмного забезпечення, а саме: процесу визначення користувача за зліпком обличчя, та загального процесу авторизації за допомогою зліпку обличчя та голосу.

Алгоритм процесу визначення користувача за зліпком обличчя є першим важливим етапом у системі автентифікації. Цей процес забезпечує ідентифікацію користувача за допомогою аналізу зображень обличчя, які

порівнюються з базою даних збережених зліпків. Алгоритм цього процесу наведено на блок-схемі (Рисунок 2.1) та описано нижче.

1. Старт: Процес розпочинається з отримання команди на автентифікацію користувача за допомогою зображення обличчя. На цьому етапі система отримує сигнал для запуску процедури автентифікації. Це може бути ініційовано користувачем або автоматичною системою моніторингу безпеки.
2. Отримання зображень: Система отримує набір зображень обличчя для аналізу. Це можуть бути зображення, отримані з камери або завантажені користувачем. Зображення можуть бути зроблені в реальному часі за допомогою камери або завантажені з файлу. Система обробляє ці зображення, готуючи їх до аналізу.
3. Для кожного зображення: Алгоритм проходить через кожне отримане зображення для подальшого аналізу. Кожне зображення обробляється окремо. Система проходить по черзі через всі отримані зображення, щоб визначити наявність облич.
4. Визначення обличчя на зображенні: За допомогою бібліотеки face_recognition система визначає області, які містять обличчя на кожному зображенні. Це досягається шляхом виявлення ключових точок обличчя, таких як очі, ніс та рот. Використовуючи алгоритми машинного навчання, система аналізує зображення та визначає області, які містять обличчя. Бібліотека face_recognition забезпечує високу точність виявлення обличчя навіть у складних умовах.
5. Для кожного визначеного обличчя: Алгоритм аналізує кожне виявлене обличчя на зображенні окремо. Якщо на зображенні знайдено декілька облич, кожне з них обробляється окремо для подальшого аналізу.
6. Отримання вектору обличчя: Використовуючи методи машинного навчання, система обчислює вектор характеристик (ембеддінг) для кожного визначеного обличчя. Цей вектор є унікальним представленням обличчя в багатовимірному просторі. Ембеддінг

обличчя являє собою набір числових значень, що описують унікальні особливості обличчя. Це дозволяє порівнювати обличчя між собою з високою точністю.

7. Пошук у масиві векторів: Вектор обличчя порівнюється з векторами, збереженими у базі даних. Це здійснюється шляхом обчислення відстані між векторами. Система порівнює вектор обличчя з векторами у базі даних, щоб знайти найбільш схожий вектор. Для цього використовується евклідова відстань або інші метрики схожості.
8. Відстань від виявленого вектору менша за визначений поріг?: Система перевіряє, чи є відстань між виявленим вектором та збереженими векторами меншою за встановлений поріг. Якщо відстань менша за поріг, це означає, що виявлене обличчя співпадає з одним із збережених зліпків. Поріг визначається на основі експериментальних даних і вказує на прийнятний рівень схожості між обличчями.

- Так: Якщо відстань менша за поріг, система ідентифікує користувача та підтверджує його автентифікацію.
- Ні: Якщо відстань більша за поріг, система не може ідентифікувати користувача і процес автентифікації завершується без успіху.

9. Визначення користувача: Після успішної ідентифікації користувача система підтверджує його автентифікацію та надає доступ до системи. Система повідомляє користувача про успішну автентифікацію та дозволяє доступ до захищених ресурсів.
10. Кінець: Процес завершується після успішної або неуспішної автентифікації. Система повертається у початковий стан і готова до наступної сесії автентифікації.

Цей алгоритм забезпечує надійний та ефективний спосіб автентифікації користувачів за допомогою розпізнавання обличчя. Використання сучасних методів машинного навчання та бібліотеки face_recognition дозволяє досягти високої точності та швидкості роботи системи.

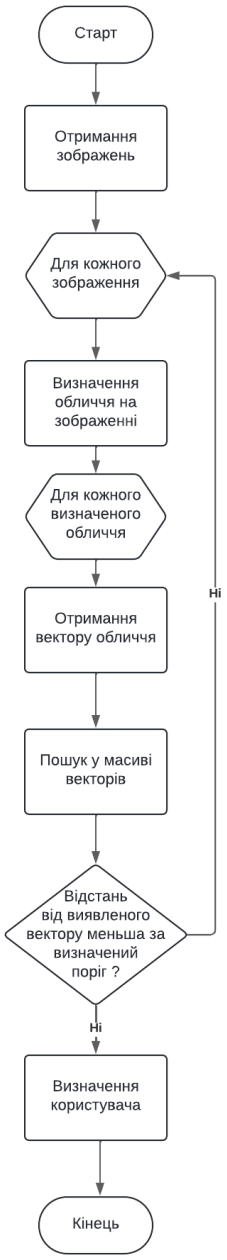


Рисунок 2.1 – Алгоритм процесу визначення користувача за зліпком обличчя

Алгоритм авторизації за допомогою зліпку обличчя та голосу об'єднує два біометричних параметри для забезпечення високого рівня безпеки. Цей процес включає введення логіну та паролю, перевірку голосу користувача та ідентифікацію за допомогою зображення обличчя. Алгоритм загального процесу роботи проектованого програмного забезпечення наведено на блок-схемі (Рисунок 2.2) та описано нижче.

1. Старт: Процес розпочинається з отримання команди на авторизацію користувача. На цьому етапі система отримує сигнал для запуску

процедури авторизації. Це може бути ініційовано користувачем або автоматичною системою моніторингу безпеки.

2. Введення логіну та паролю: Користувач вводить свої логін та пароль. Користувач надає свої облікові дані для початкової перевірки. Це стандартний етап автентифікації, який забезпечує базову перевірку.
3. Введені дані наявні у базі даних?: Система перевіряє, чи існують введені логін та пароль у базі даних. Якщо введені дані не відповідають жодному запису в базі даних, система видає повідомлення про помилку та завершує процес авторизації. Якщо введені дані правильні, процес переходить до наступного кроку. Якщо введені дані неправильні, система повідомляє користувача про помилку. Користувач отримує повідомлення про невірність введених даних і процес завершується.
4. Витягнення характеристик голосу користувача з БД: Система витягує збережені характеристики голосу користувача з бази даних. Для подальшої перевірки голосу система отримує збережені векторні характеристики голосу користувача.
5. Вмикання мікрофону користувача: Система вмикає мікрофон для запису голосу користувача. Користувач отримує інструкцію вимовити певну фразу або слово для запису голосу.
6. Отримання зразку голосу користувача: Система записує зразок голосу користувача через мікрофон. Записаний зразок голосу використовується для порівняння з збереженими характеристиками.
7. Витягнення характеристик зі зразку голосу: Система витягує характеристики з записаного зразку голосу користувача. Використовуючи бібліотеку librosa, система витягує мел-частотні кепстральні коефіцієнти (MFCC) з записаного голосу.
8. Порівняння характеристик: Система порівнює витягнуті характеристики голосу зі збереженими в базі даних. Використовується метод порівняння евклідової для визначення схожості між голосами.

9. Характеристики співпадають?: Система перевіряє, чи співпадають характеристики голосу з збереженими даними. Якщо характеристики голосу не співпадають, система видає повідомлення про помилку та завершує процес авторизації. Якщо характеристики голосу співпадають, процес переходить до наступного кроку.
10. Отримання зображення користувача: Система отримує зображення обличчя користувача. Це може бути зображення, зроблене камерою в реальному часі, або завантажене з файлу. Визначення обличчя на зображенні: Система визначає обличчя на зображенні за допомогою бібліотеки face_recognition. Використовуються алгоритми машинного навчання для виявлення обличчя на зображенні. Для кожного визначеного обличчя: Алгоритм аналізує кожне виявлене обличчя на зображенні окремо. Якщо на зображенні знайдено декілька облич, кожне з них обробляється окремо.
11. Отримання вектору обличчя: Система обчислює вектор характеристик (ембеддінг) для кожного визначеного обличчя. Вектор обличчя представляє унікальні характеристики обличчя в багатовимірному просторі
12. Пошук у масиві векторів: Вектор обличчя порівнюється з векторами, збереженими у базі даних. Система порівнює вектор обличчя з векторами у базі даних для визначення схожості.
13. Відстань від виявленого вектору менша за визначений поріг?: Система перевіряє, чи є відстань між виявленим вектором та збереженими векторами меншою за встановлений поріг. Якщо відстань менша за поріг, це означає, що виявлене обличчя співпадає з одним із збережених зліпків. Якщо відстань більша за поріг, система не може ідентифікувати користувача і процес завершується без успіху.
14. Визначення користувача: Після успішної ідентифікації користувача система підтверджує його автентифікацію. Користувач отримує

доступ до системи після успішної ідентифікації за обличчям та голосом.

15. Виведення повідомлення про успішну авторизацію: Система повідомляє користувача про успішну авторизацію. Користувач отримує підтвердження про успішний вхід у систему.
16. Кінець: Процес завершується після успішної або неуспішної авторизації. Система повертається у початковий стан і готова до наступної сесії авторизації.

Цей алгоритм забезпечує комплексну автентифікацію користувачів, поєднуючи розпізнавання обличчя та голосу Використання сучасних методів машинного навчання та бібліотек face_recognition, speech_recognition, і librosa дозволяє досягти високої точності та надійності системи.

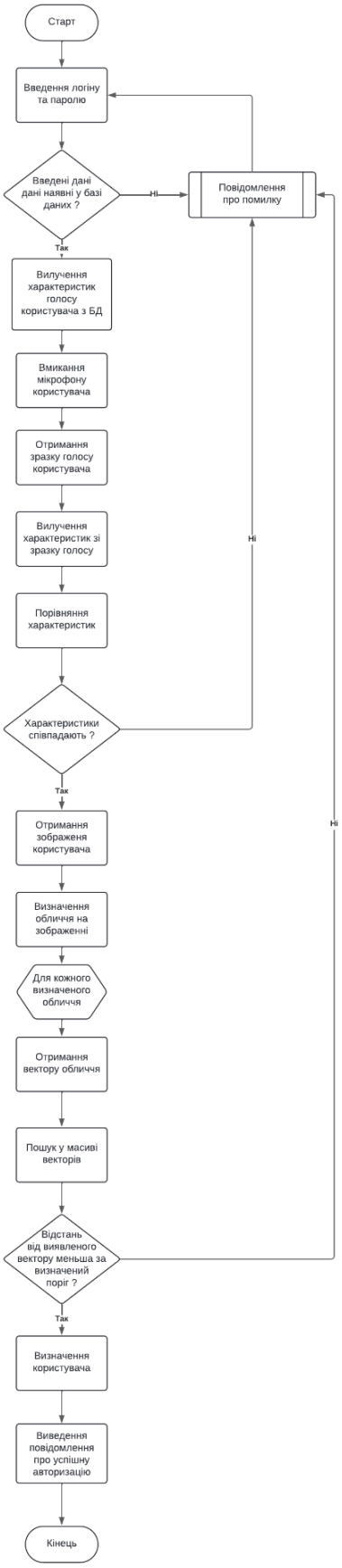


Рисунок 2.2 – Загальний алгоритм роботи проектованого програмного забезпечення

2.3 Вибір системи керування базами даних

Незалежно від того, яка інформаційна система та з використанням яких мов програмування буде проектуватися, для повного її функціонування будуть використані бази даних. База даних представляє собою систематизований набір даних. Вони підтримують електронне зберігання та маніпулювання даними. Бази даних значно спрощують управління даними.

Отже, розглянемо найпопулярніші системи управління базами даних та визначимо, яка з них найкраще підійде для вирішення нашого завдання.

SQLite - одна з найпопулярніших систем реляційних баз даних. Реалізована спочатку як рішення з відкритим вихідним кодом, SQLite тепер належить корпорації Oracle. Сьогодні SQLite є основою прикладного програмного забезпечення LAMP. Це означає, що він є частиною стека Linux, Apache, SQLite та Perl/PHP/Python. Маючи під капотом C і C++, SQLite добре працює з системними платформами, як Windows, Linux, MacOS, IRIX та іншими.

Плюси SQLite:

- Безкоштовне встановлення. Версія SQLite для спільноти доступна для безкоштовного завантаження. Версія SQLite Community Edition з базовим набором інструментів для індивідуального використання є гарним варіантом для початку.
- Простий синтаксис та невелика складність. Структура та стиль SQLite дуже прості. Розробники навіть вважають SQLite базою даних із людською мовою. SQLite часто використовується у тандемі з мовою програмування PHP.
- Хмарна сумісність. Орієнтована на бізнес за своєю природою та спочатку розроблена для Інтернету, SQLite підтримується найпопулярнішими хмарними провайдерами. Він доступний на таких провідних платформах, як Amazon, Microsoft та інші [11].

MariaDB – наступна система керування базами даних, що представляє собою форк MySQL з відкритим вихідним кодом та має комерційну підтримку. Він працює під Стандартною громадською ліцензією GNU і має ті ж команди, API та бібліотеки, що й MySQL.

Плюси MariaDB:

- Шифрування. Для MariaDB відкритий вихідний код означає небезпечний. Крім внутрішньої безпеки та перевірки паролів, MariaDB надає такі функції, як автентифікація PAM та LDAP, Kerberos та ролі користувачів. У поєднанні із зашифрованими табличними просторами, таблицями та журналами він створює надійний рівень захисту даних.
- Широкий функціонал. За останні кілька років MariaDB представила багато нових функцій. Наприклад, підтримка ГІС передбачає плавне зберігання координат та запити даних про місцезнаходження. Динамічні стовпці дозволяють одній СУБД забезпечувати обробку даних SQL та NoSQL для різноманітних потреб.
- Висока продуктивність. Розширені функції оптимізації покращують управління пулом потоків та обробку даних. Таким чином, при видаленні рядків з таблиці операційна система відразу звертається до вільного простору, усуваючи прогалини в табличному просторі [12].

Oracle – наступна система управління реляційними базами даних, створена та керована корпорацією Oracle. В даний час вона підтримує кілька моделей даних, таких як документ, граф, реляційна модель та модель "ключ-значення" в рамках однієї бази даних. У останніх випусках він переорієнтувався на хмарні обчислення.

Плюси Oracle:

- Інновації для щоденного робочого процесу. Починаючи з версії Oracle 12c, коли програмне забезпечення набуло ери гібридної хмари, регулярно з'являлися нові технології хмарних обчислень. З кожним новим випуском Oracle намагається не відставати від темпів інновацій, приділяючи особливу увагу інформаційній безпеці, включаючи

активний захист даних, поділ, покращене резервне копіювання та відновлення.

- Потужна техпідтримка та документація. Oracle забезпечує гідну підтримку клієнтів та надає вичерпну технічну документацію щодо кількох ресурсів.
- Велика ємність. Мультімодельне рішення Oracle дозволяє розміщувати та обробляти величезну кількість даних. Завдяки нещодавно випущеній функції мультиарендності архітектура бази даних тепер спрощує упаковку багатьох баз даних та забезпечує плавне керування ними [13].

PostgreSQL - наступна система управління базами даних, що поділяє свою популярність із MySQL. Це об'єктно-реляційна СУБД, в якій об'єкти користувача та табличні підходи об'єднуються для створення більш складних структур даних. Крім того, PostgreSQL має багато спільного з MySQL. Він спрямований на зміцнення стандартів відповідності та розширюваності. Отже, він може обробляти будь-яке робоче навантаження як для одномашинних продуктів, так і для складних додатків.

Плюси PostgreSQL:

- Відмінна масштабованість. Вертикальна масштабованість є відмінністю PostgreSQL, на відміну від СУБД MySQL. Враховуючи, що майже будь-яке спеціалізоване програмне рішення має тенденцію до зростання, що призводить до розширення бази даних, цей конкретний варіант, безумовно, сприяє зростанню та розвитку бізнесу.
- Підтримка типів даних користувача. PostgreSQL спочатку підтримує велику кількість типів даних за промовчанням, таких як JSON, XML, H-Store та інші. PostgreSQL використовує цю перевагу, будучи однією з небагатьох реляційних баз даних із сильною підтримкою NoSQL.
- Підтримка з відкритим вихідним кодом та спільнотою. Postgres має повністю відкритий вихідний код та підтримується спільнотою, що зміцнює його як повноцінну екосистему [14].

Для аналізу зазначених СУБД складемо порівняльну таблицю для прооведення порівняння за їх перевагами та недоліками (Таблиця 2.1).

Таблиця 2.1 - Порівняльна характеристика розглянутих СУБД

Назва	Переваги	Недоліки
SQLite	Пропонує багато функцій, навіть у безкоштовній версії Пакет SQLite включений у стандартні репозиторії дистрибутивів Linux. Може працювати з іншими базами даних, включаючи DB2 та Oracle.	Невеликий функціонал Відсутня підтримка XML або OLAP. Для безкоштовної версії доступна лише платна підтримка.
Oracle	Найсвіжіші інновації та вражаючий функціонал уже впроваджено в цьому продукті, оскільки Oracle прагне тримати планку навіть на тлі інших розробників СУБД. Oracle є вкрай надійною, фактично це еталон надійності серед таких систем.	Вартість Oracle може бути непомірно високою, особливо для невеликих організацій. Система може вимагати значних ресурсів відразу після установки, тому можливо потрібно модернізувати обладнання для впровадження Oracle.

Продовження таблиці 2.1

MariaDB	Продуктивність Індикатори дадуть вам знати, як обробляється запит. Розширювана архітектура та плагіни дозволяють налаштовувати інструмент відповідно до ваших потреб. Шифрування доступне в мережі, сервері та рівні програми.	На сьогодні стабільність нижча, ніж у MySQL, тому навіть на нових проектах можна рекомендувати встановлювати mysql. Двигун досить новий, тому поки що немає жодних гарантій подальших оновлень. Як і в інших безкоштовних базах даних, вам доведеться платити за підтримку.
PostgreSQL	Є масштабованим рішенням та дозволяє обробляти терабайти даних. Підтримує формат json. Існує безліч визначених функцій. Доступний ряд інтерфейсів.	Документація туманна, тому, можливо, відповіді на деякі запитання доведеться шукати в Інтернеті. Конфігурація може збентежити непідготовленого користувача. Швидкість роботи може зменшуватися під час пакетних операцій або виконання запитів читання.

На основі проведеного аналізу було обрано систему керування базами даних (СКБД) SQLite. Це рішення обумовлено кількома важливими факторами, що роблять SQLite оптимальним вибором для реалізації проекту.

Перш за все, SQLite є вбудованою системою керування базами даних, яка не вимагає налаштування окремого серверного програмного забезпечення. Це значно спрощує процес розробки, розгортання та підтримки системи. У контексті проекту, де ключову роль відіграє ефективність та швидкість

доступу до даних, SQLite забезпечує необхідну продуктивність завдяки своїй легковажності та високій швидкості роботи.

Крім того, SQLite підтримує основні функціональні можливості реляційних баз даних, включаючи транзакції, обмеження цілісності даних та складні SQL-запити. Це дозволяє реалізувати всі необхідні вимоги до обробки та зберігання даних у системі автентифікації.

Ще одним важливим фактором є крос-платформенність SQLite, що дозволяє використовувати її в різних операційних системах без необхідності додаткових налаштувань. Це забезпечує гнучкість та зручність при розробці та тестуванні програмного забезпечення.

Оскільки даний проект потребує зберігання та швидкого доступу до біометричних даних користувачів, включаючи зображення облич та аудіозаписи голосу, використання SQLite дозволяє ефективно працювати з великими обсягами даних. Зважаючи на все вищевикладене, вибір SQLite є найбільш доцільним та виправданим для реалізації поставлених задач у рамках даного проекту.

2.4 Обгрунтування вибору мови програмування

В рамках виконання поставленого в дипломній роботі завдання є потреба написання програмного продукту, який буде програмно реалізовувати процес комплексної автентифікації до системи захисту з використанням ідентифікації за допомогою зліпка обличчя та голосу. Для написання обумовленного програмного продукту необхідно вибрати основний інструмент реалізації, а саме мову програмування від якої залежатиме швидкість написання та роботи проектованого програмного продукту.

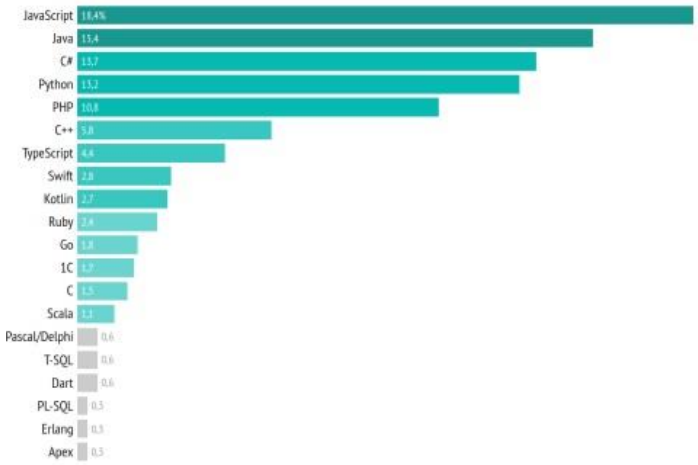


Рисунок 2.3 – Рейтинг мов програмування у комерційних проектах, %

На сьогоднішній день існує велика кількість різних мов програмування і в кожної з них своя сфера застосування, але все ж для проведення аналізу на вибір кращої мови для написання проектованого програмного продукту, необхідно вибрати кілька найпопулярніших мов, щоб між ними проводити аналіз, тож у цьому розділі звернемося до статистики за популярністю мов. На малюнку 3.1 показаний рейтинг мов 2022 у комерційних розробках [11].

За цими даними можна зробити висновок, що JavaScript значно випереджає Java і є найпопулярнішою мовою програмування. У п'ятірку найкращих мов увійшли також: C#, Python, PHP. На малюнку 3.2 можна подивитися, як змінювалися дані на протязі 2012-2022 років.

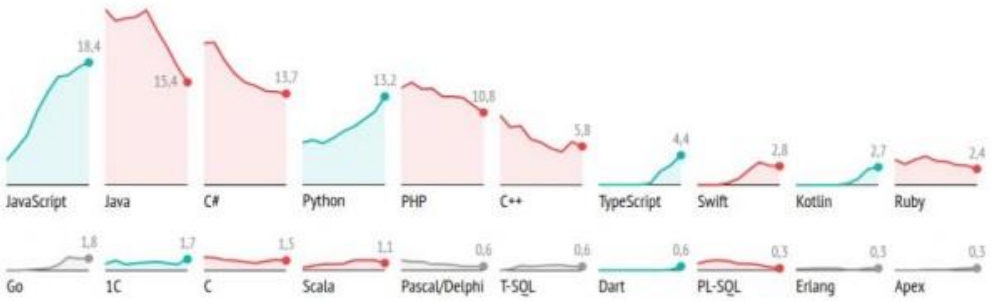


Рисунок 2.4 – Графік зміни популярності мов програмування, %

На малюнку 2.4 можна побачити, що популярність мови Java і C# сильно падає, а популярність таких мов як JavaScript, TypeScript та Python продовжує зростати.

Python — це високорівнева мова загального програмування призначення, яке також використовується для розробки веб-додатків. Мова націлена на підвищення продуктивності праці розробників та читабельності коду. Python підтримує різні парадигми програмування: структурну, об'єктно-орієнтовану, функціональну, імперативну та аспектно-орієнтовану. Мова включає динамічну типізацію, автоматичне керування пам'яттю, повне самоспостереження, механізм обробки винятків, підтримку багатопоточних обчислень та практичні структури даних високого рівня.

Переваги Python :

- Відкрита розробка;
- Досить простий у вивченні, особливо на початковому етапі;
- особливості синтаксису стимулюють програміста писати код, що легко читається;
- Надає засоби швидкого прототипування та динамічної семантики;
- має велику спільноту, позитивно налаштовану по відношенню до новачкам;
- Безліч корисних бібліотек та розширень мови можна легко використовувати у своїх проектах завдяки гранично уніфікованим механізмом імпорту та програмним інтерфейсів;
- Механізми модульності добре продумані та можуть бути легко використані;
- Абсолютно все в Python є об'єктами в сенсі ООП, але при цьому об'єктний підхід не нав'язується програмісту.
- Недоліки Python:
- Не надто вдала підтримка багатопоточності;
- На Python створено не так вже й багато якісних програмних проектів у порівнянні з іншими універсальними мовами програмування, наприклад, з Java;
- Початкова обмеженість коштів для роботи з базами даних;

- Бенчмарки показують меншу продуктивність Python за порівняно з основними Java VM, що створює цій мові репутацію повільної [12].

JavaScript – мультипарадигменна мова програмування. Підтримує об'єктно-орієнтований, імперативний та функціональний стилі.

Переваги JavaScript:

- Жоден сучасний браузер не обходиться без підтримки JavaScript.
- З використанням написаних на JavaScript плагінів та скриптів впорається навіть не фахівець.
- Корисні функціональні налаштування.
- Мова, що постійно вдосконалюється – зараз розробляється бета-варіація проекту, JavaScript2.
- Взаємодія з додатком може здійснюватися навіть через текстові редактори – Microsoft Office та Open Office.
- Недоліки JavaScript:
- Знижений рівень безпеки через повсюдний та вільного доступу до вихідних кодів найпопулярніших скриптів.
- Безліч дрібних дратівливих помилок на кожному етапі роботи. Більшість їх легко виправляється, але їх наявність дозволяє вважати цю мову менш професійною, порівняно з іншими.
- Повсюдне поширення. Своєрідним недоліком можна вважати той факт, що частина програм, що активно використовуються (особливо додатків) перестануть існувати за відсутності мови, оскільки повністю базуються на ній [13].

C# - мова програмування, що поєднує об'єктно-орієнтовані та контекстно-орієнтовані концепції.

Переваги:

- Для малих підприємств та деяких окремих розробників безкоштовні інструменти включають Visual Studio, Azure Cloud, Windows Server, Parallels Desktop для Mac Pro та багато інших;

- Велика кількість синтаксичних конструкцій, розроблених для кращого розуміння написання коду;
- Дуже простий у вивченні;
- Ціла спільнота з досвідчених програмістів.
- Недоліки:
- Пріоритетна орієнтованість на Windows платформу;
- Мова безкоштовна тільки для невеликих фірм, індивідуальних програмістів, стартапів та учнів. Великої компанії покупка ліцензійної версії цієї мови коштуватиме круглу суму;
- У мові залишилася можливість використання оператора безперечного переходу [14].

Для аналізу характеристик можливих для використання технологій, складемо порівняльні таблиці для проведення порівняння за такими характеристиками як: Парадигми, Типізація, Керування пам'яттю, Об'єктно - орієнтованими можливостями та частотою використання.

Таблиця 2.2 - Порівняльна характеристика за парадигмами

Можливість	C#	Java	JavaScript	PHP	Python	Ruby
Імперативна	+	+	+	+	+	+
ООП	+	+	+	+	+	+
Функціональна	+/-	+/-	+/-	+/-	+	+
Рефлексивна	+/-	+/-	+	+	+	+
Узагальнена	+	+	+	+	+	+
Логічна	-	-	-	-	-	-
Декларативна	+/-	-	+/-	+	+	+
Розподілена	+/-	+	-	+	+/-	+/-

Таблиця 2.3 - Порівняльна характеристика за типізацією

Можливість	C#	Java	JavaScript	PHP	Python	Ruby
Статична типізація	+	+	-	-	-	-
Динамічна типізація	+	-	+	+	+	+
Явна типізація	+/-	+	-	+/-	+/-	-

Неявна типізація	+	-	+	+	+	+
Неявне приведення	-	+	+	+	+	+

Таблиця 2.4 - Характеристика за можливостями керування пам’яттю

Можливість	C#	Java	JavaScript	PHP	Python	Ruby
Створення об’єктів на стеку	+	-	-	-	-	-
Некеровані вказівники	+	-	-	-	-	-
Ручне керування пам’яттю	+	-	-	-	-	-
Збирання сміття	+	+	+	+	+	+

Таблиця 2.5 - Порівняльна характеристика за функціональними можливостями

Можливість	C#	Java	JavaScript	PHP	Python	Ruby
Декларації чистоти функцій	-	-	-	-	-	-
First class функції	+	-	+	+	+	+
Анонімні функції	+	+	+	+	+	+
Лексичні замкнення	+	+	+	+	+	+
Часткове застосування	-	-	+	-	+	+
Карування	+	-	+	+	+	+

Таблиця 2.6 - Порівняльна характеристика за об’єктно – орієнтованими можливостями

Можливість	C#	Java	JavaScript	PHP	Python	Ruby
Інтерфейси	+	+	+	+	+	-
Мультиметоди	+/-	-	-	-	-	-
Міксини	-	+	+	+	+	+
Спадкування	-	-	-	-	-	-
Множинне спадкування	-	-	-	-	+	-

Щоб зробити вибір мови програмування для створення програмного продукту для реалізації процесу комплексної автентифікації до системи захисту з використанням ідентифікації за допомогою зліпка обличчя та голосу, потрібно зрозуміти, можливості якої мови можуть задовольнити поточну

потребу. У виконанні цього завдання принциповими є швидкість виконання алгоритму роботи, наявність необхідного інструментарію та зручність реалізації.

На основі проведеного аналізу було обрано мову програмування Python для реалізації програми комплексної автентифікації до системи захисту з використанням ідентифікації за допомогою зліпка обличчя та голосу. Python був обраний завдяки своїй високій продуктивності, зручності використання та широкому спектру бібліотек, які забезпечують необхідний функціонал для обробки та аналізу зображень і голосових даних. Додатковою перевагою є велика спільнота розробників, що сприяє швидкому вирішенню можливих проблем та доступності навчальних матеріалів. Таким чином, Python забезпечує ефективну реалізацію та інтеграцію необхідних компонентів для побудови надійної системи автентифікації.

2.5 Висновок до розділу 2

Уданому розділі проведено обґрунтування вибору вдосконалених методів автентифікації та детальний опис алгоритму роботи програмного забезпечення. Це дозволяє забезпечити високу точність та надійність роботи системи, що розробляється.

Було визначено та обґрунтовано вибір системи керування базами даних та мови програмування для розробки програмного забезпечення. Використання сучасних технологій та інструментів сприяє підвищенню ефективності розробки та експлуатації системи.

І наведено схематичне зображення алгоритмів роботи програмного забезпечення, що розробляється. Це включає попередню обробку зображень та аудіо, що забезпечує структурованість та логічність процесу розробки.

3. ПРАКТИЧНА РЕАЛІЗАЦІЯ ПРОГРАМНОГО ПРОДУКТУ

3.1 Опис процесу попередньої обробки зображень обличчя у відеопотоці

Для того, щоб реалізувати функціонал ідентифікації за допомогою зліпка обличчя, необхідно детально проаналізувати алгоритм взаємодії з зображенням цього обличчя.

Першим кроком необхідно виконати перетворення, або, іншими словами, ембеддинг (embedding), зображення обличчя в числовій вектор. Це також називається глибоким метричним навчанням. Наше перше завдання – це виявлення осіб на зображенні або відеопотоці. Далі, коли ми знаємо точне розташування або координати особи, ми беремо цю особу для подальшої обробки.

Вирізавши зліпок обличчя із зображення, ми маємо витягти з нього характерні риси. Для цього ми використовуватимемо попередньо описану процедуру під назвою ембеддинг.

Нейронна мережа приймає на вхід зображення, а на виході повертає числовий вектор, що характеризує основні ознаки цієї особи. У машинному навчанні цей вектор і називається ембеддингом.

Тепер наведемо на прикладі, як це допомагає у розпізнаванні осіб різних людей.

$$f\left(\text{  }\right) = \begin{pmatrix} 0.112 \\ 0.067 \\ 0.091 \\ 0.129 \\ 0.002 \\ 0.012 \\ 0.175 \\ \vdots \\ 0.023 \end{pmatrix}$$

Рисунок 3.1 – Вилучення векторів з зображення

Під час навчання нейронна мережа вчиться видавати близькі вектори для осіб, які виглядають схожими один на одного.

Наприклад, якщо у нас є кілька зображень особи в різні моменти часу, то природно, що деякі риси обличчя можуть змінюватися, але все ж таки незначно. Таким чином, вектори цих зображень будуть дуже близькими у векторному просторі. Щоб отримати загальне уявлення про це, створимо графік (Рисунок 3.2):

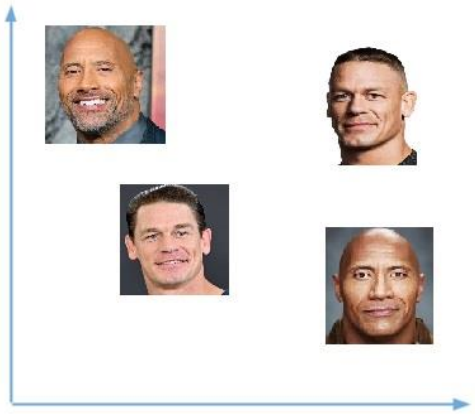


Рисунок 3.2 – Приклад вхідних зображень

Щоб визначати особи однієї людини, мережа вчитиметься виводити вектори, що знаходяться поруч у векторному просторі. Після навчання ці вектори трансформуються так (Рисунок 3.3):

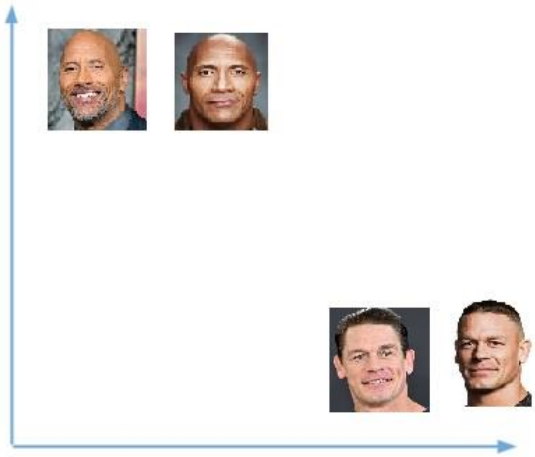


Рисунок 3.3 – Результат навчання моделі

Тепер, коли система має вектор (ембеддинг) для кожної особи з бази даних, нам потрібно навчитися розпізнавати особи з нових зображень.

Таким чином, нам потрібно, як і раніше, обчислити вектор для нової особи, а потім порівняти його з існуючими векторами. Ми зможемо розпізнати зліпок обличчя, якщо воно схоже на одну з осіб, які вже є в нашій базі даних. Це означає, що їх вектори будуть розташовані поблизу один від одного, як показано на рисунку 3.4.

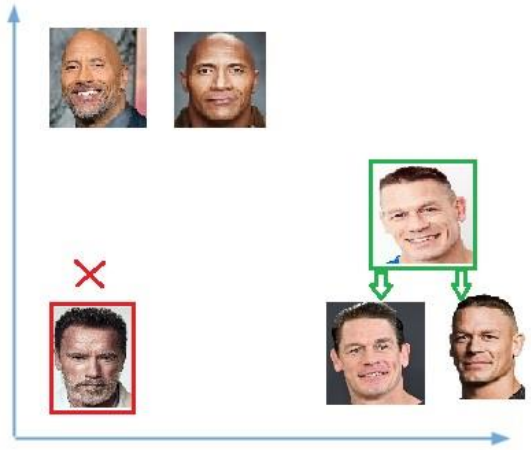


Рисунок 3.4 – Процес розпізнавання

Отже, ми передали до мережі дві фотографії (Рисунок 3.6), одна Джона Сіни, інша Арнольда Шварценегера. Для зображень Джона у нас були вектори (ембеддінги), а для Арнольда нічого не було. Таким чином, коли ми порівняли ембеддинг нового зображення Джона, він був близький до вже наявних векторів, і ми розпізнали його. А ось зображень Арнольда в нашій базі не було, тож розпізнати його не вдалося.

3.2 Використані при розробці бібліотеки та модулі

При розробці проектованого програмного продукту було використано бібліотеки та модулі Python для написання логіки процесу розпізнавання користувача за зліпком обличчя та голосом, та розробки інтерфейсу для взаємодії з програмним продуктом. Наведемо короткий опис цих модулів та бібліотек :

Dlib - Просунута бібліотека машинного навчання, створена для вирішення складних завдань розпізнавання. Ця бібліотека була створена з використанням мови програмування C++ та працює з C/C++, Python та Java. В реалізованому

програмному забезпеченні dlib використовується для розпізнавання зліпків облич на зображеннях та виділення рис обличчя, що дозволяє ідентифікувати знайдених людей.

Face recognition - API інструмент, який встановлюється з бібліотекою dlib. Сам інструмент виділяє риси обличчя на зображеннях і, якщо нам потрібен такий функціонал, намагається по них ідентифікувати знайдену людину. В реалізованому програмному забезпеченні використовується для виявлення та розпізнавання облич на зображеннях, що дозволяє ідентифікувати користувачів за їхніми обличчями.

OpenCV - Бібліотека алгоритмів комп'ютерного зору, обробки зображень та чисельних алгоритмів загального призначення з відкритим кодом. Реалізована C/C++, також розробляється для Python, Java, Ruby, Matlab, Lua та інших мов. В реалізованому програмному забезпеченні використовується для завантаження та обробки зображень, зокрема для перетворення зображень у формати, зручні для аналізу бібліотекою dlib.

SQLite - Бібліотека , що реалізує легковажну дискову базу даних (БД), що не вимагає окремого серверного процесу і дозволяє отримати доступ до БД з використанням мови запитів SQL. Деякі програми можуть використовувати SQLite для внутрішнього зберігання даних. В реалізованому програмному забезпеченні використовується для зберігання даних користувачів, логів авторизацій та реєстрацій у локальній базі даних.

Tkinter - Крос-платформна подієво-орієнтована графічна бібліотека на основі засобів Tk, написана Стіном Лумхольтом та Гвідо ван Россумом. Входить до стандартної бібліотеки Python. Tkinter – це вільне програмне забезпечення, яке розповсюджується під Python-ліцензією. В реалізованому програмному забезпеченні використовується для створення графічного інтерфейсу користувача (GUI), через який користувачі можуть реєструватися, авторизуватися та підтверджувати свою особу.

Os - Вбудована бібліотека мови програмування, що надає інтерфейс для взаємодії з базовою операційною системою, під керуванням якої працює

Python. Цей модуль є портативним способом використання функціональності, яка залежить від операційної системи. В реалізованому програмному забезпеченні використовується для роботи з файловою системою, наприклад, для створення директорій та переміщення файлів.

Shutil - Модуль, який містить набір функцій високого рівня обробки файлів, груп файлів і папок. Зокрема, доступні тут функції дозволяють копіювати, переміщати та видаляти файли та папки. Часто використовується разом із модулем os. В реалізованому програмному забезпеченні використовується для копіювання та переміщення зображень користувачів у відповідні директорії.

SpeechRecognition - Бібліотека для розпізнавання голосу, що підтримує кілька різних API для розпізнавання голосу. В реалізованому програмному забезпеченні використовується для запису голосових даних користувачів та подальшого їх збереження у базі даних для перевірки голосової ідентифікації.

PyAudio - Бібліотека для роботи з аудіо, що надає можливості для запису та відтворення звуку. В реалізованому програмному забезпеченні використовується для запису голосу користувача з мікрофона, який зберігається та використовується для голосової ідентифікації.

Wave - Вбудована бібліотека Python для роботи з WAV файлами. В реалізованому програмному забезпеченні використовується для збереження записаних голосових даних у форматі WAV.

NumPy - Бібліотека для наукових обчислень у Python, що забезпечує підтримку великих багатовимірних масивів і матриць, разом з великою колекцією високорівневих математичних функцій для роботи з цими масивами. В реалізованому програмному забезпеченні використовується для обробки та порівняння голосових даних.

PyAudioAnalysis - Бібліотека для аналізу аудіо сигналів і екстракції характеристик. В реалізованому програмному забезпеченні використовується для екстракції характеристик з голосових записів, що дозволяє порівнювати голосові дані для ідентифікації користувачів.

3.3 Програмна реалізація проектованого ПЗ

3.3.1 Програмна реалізація аналізу наявних зображень та створення датасету

Для початку роботи з розпізнавання необхідно знайти датасет з особами, з якими власно наше програмне забезпечення буде порівнювати вхідне зображення, або створити власний. Переконаємось, що всі зображення знаходяться в папках, причому в кожній папці повинні бути фотографії однієї людини.

Далі розмістимо датасет у робочій директорії проекту де ми створюватимо наше програмне забезпечення та задамо шлях до неї.

```
imagePaths = list(paths.list_images('Images'))
knownEncodings = []
knownNames = []
```

Наступним кроком реалізуємо перебір усіх папок з фотографіями людин для аналізу, вирахуємо їх ембендінги та додамо ці значення у відповідний словник:

```
for (i, imagePath) in enumerate(imagePaths):
    name = imagePath.split(os.path.sep)[-2]
    image = cv2.imread(imagePath)
    rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    boxes = face_recognition.face_locations(rgb, model='hog')
    encodings = face_recognition.face_encodings(rgb, boxes)
    for encoding in encodings:
        knownEncodings.append(encoding)
        knownNames.append(name)
```

Наступним кроком збережемо усі отримані ембендінги та їх імена у форматі словника та збережемо отриманий код у функцію analytic(), щоб система мала можливість робити повний аналіз датасету зображень при кожному запуску програми.

```
data = {"encodings": knownEncodings, "names": knownNames}
f = open("face_enc", "wb")
f.write(pickle.dumps(data))
f.close()
```

На цьому етапі ми зберегли усі ембеддінги у файл під назвою face_enc. Тепер ми можемо використовувати створений датасет для розпізнавання

зліпку обличчя на зображеннях або під час відеостриму з веб-камери під час роботи системи.

3.3.2 Програмна реалізація логіки розпізнавання зліпку обличчя

Виведемо логіку розпізнавання обличчя у окремий модуль для зручності. При зверненні до цього модулю він повинен отримувати на вхід 2 параметра , а саме зображення людини для аналізу та його ім’я для перевірки, бо наша головна задача ідентифікувати, що саме цей користувач в даний момент намагається авторизуватися на даному пристрої.

```
            def check(imgpath,login):
    cascPathface = os.path.dirname(
    cv2.__file__) + "/data/haarcascade_frontalface_alt2.xml"
  faceCascade = cv2.CascadeClassifier(cascPathface)
  data = pickle.loads(open('face_enc', "rb").read())
  image = cv2.imread(imgpath)
  gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
  faces = faceCascade.detectMultiScale(gray,
                       scaleFactor=1.1,
                       minNeighbors=5,
                       minSize=(60, 60),
                       flags=cv2.CASCADE_SCALE_IMAGE)
  encodings = face_recognition.face_encodings(rgb)
  names = []
```

У наведеній частині коду ми отримуємо на вхід параметри зображення людини, яка намагається авторизуватися та його логіну у системі та декодуємо його до формату вектору, зручного до порівняння з наявними у датасеті.

```
for encoding in encodings:
  matches = face_recognition.compare_faces(data["encodings"],
                       encoding)
  name = "Unknown"
  if True in matches:
    matchedIdxs = [i for (i, b) in enumerate(matches) if b]
    counts = {}
    for i in matchedIdxs:
      name = data["names"][i]
      counts[name] = counts.get(name, 0) + 1
      name = max(counts, key=counts.get)
    names.append(name)
    for ((x, y, w, h), name) in zip(faces, names):
      cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0), 2)
      cv2.putText(image, name, (x, y), cv2.FONT_HERSHEY_SIMPLEX,
          0.75, (0, 255, 0), 2)
  cv2.imshow("Frame", image)
  cv2.waitKey(0)
  if name == login:
    print('yes')
    return True
  else:
```

```
    print('nope')
    return False
```

У даній частині коду ми порівнюємо ембендінг отриманого зображення людини з наявними ембендінгами у датасеті та повертаємо на виклик функції True, якщо визначена людина співпадає з логіном під яким вона намагається авторизуватися, або False, якщо не співпадає.

Відповідно логіка основного виконуючого файлу зможе у будь який момент звернутися до написаної функції, передавши їй фото та логін та отримавши назад відповідь в якості простої змінної, ідентифіковано користувача чи ні.

3.3.3 Програмна реалізація логіки взаємодії з голосом користувача

Для можливості використання голосу користувача в процесі його авторизації, реалізуємо функції, що будуть надавати функціонал отримання характеристик голосу та взаємодію з ними.

```
def record_voice(filename):
  r = sr.Recognizer()
  with sr.Microphone() as source:
    print("Будь ласка, скажіть щось...")
    audio = r.listen(source)
    with open(filename, "wb") as f:
      f.write(audio.get_wav_data())
```

Реалізована функція використовується для запису голосу користувача за допомогою мікрофона. Записаний звук зберігається у вказаний файл, щоб його можна було використовувати для подальшої обробки та аналізувати та порівнювати голосові дані для ідентифікації користувачів.

```
def extract_features(filename):
  y, sr = librosa.load(filename)
  mfccs = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=13)
  return np.mean(mfccs.T, axis=0)
def compare_voices(features1, features2):
  return np.linalg.norm(features1 - features2)
```

Перша з наведених функцій реалізує витягування мел – частотних кепстральних коефіцієнтів (MFCC) з аудіофайлу. MFCC є важливими характеристиками, які використовуються для аналізу та порівняння голосових сигналів.

Друга з наведених функцій обчислює евклідову відстань між двома наборами характеристик голосу (MFCC). Чим менше значення евклідової

відстані, тим більш схожі голоси. Це використовується для перевірки, чи співпадає записаний голос з голосом, збереженим в базі даних.

3.3.4 Програмна реалізація логіки отримання зліпку обличчя користувача з його веб – камери

Наступним кроком для можливості використання зліпку обличчя користувача в процесі його авторизації, реалізуємо функції, що будуть надавати функціонал взаємодії з його веб – камерою та визначення його обличчя на ній. Для цієї мети проведемо реалізацію функції capture_image(), яка забезпечує захоплення зображення обличчя користувача з веб-камери та збереження цього зображення у файл. Функція capture_image реалізована з використанням бібліотеки OpenCV. Перейдемо до програмної реалізації.

```
cap = cv2.VideoCapture(0)
```

На початку функції виконується ініціалізація об'єкта VideoCapture, який дозволяє працювати з веб-камерою. Параметр 0 вказує на першу камеру в системі. Якщо в системі є кілька камер, цей параметр можна змінити для вибору відповідної камери.

```
if not cap.isOpened():
  print("Не вдалося відкрити камеру")
  return None
```

Після ініціалізації виконується перевірка, чи вдалося відкрити камеру. Якщо камера не була успішно відкрита, функція виводить повідомлення про помилку та повертає None.

```
while True:
    ret, frame = cap.read()
    if not ret:
        print("Не вдалося захопити зображення")
        continue

    cv2.imshow('Press Space to Capture', frame)
    if cv2.waitKey(1) & 0xFF == ord(' '):  # Press Space to capture
        img_path = f'images/{username}.jpg'
        cv2.imwrite(img_path, frame)
        break
```

Основна частина функції виконує безперервне захоплення зображень з камери у циклі while True. cap.read() захоплює зображення з камери. Змінна ret вказує на успішність захоплення, а frame містить саме зображення. Якщо захоплення зображення не вдалося (ret == False), виводиться повідомлення про

помилку, і цикл продовжується. cv2.imshow('Press Space to Capture', frame) відображає захоплене зображення у вікні з повідомленням "Press Space to Capture". cv2.waitKey(1) & 0xFF == ord(' ') чекає на натискання пробілу користувачем. Якщо пробіл натиснуто, захоплене зображення зберігається у файл з ім'ям, що включає ім'я користувача, і цикл завершується.

```
cap.release()
cv2.destroyAllWindows()
return img_path
```

Після захоплення та збереження зображення веб-камера звільняється, а всі вікна OpenCV закриваються. Функція повертає шлях до збереженого зображення. Реалізована функція capture_image забезпечує інтерактивний спосіб захоплення зображення обличчя користувача з веб-камери та зберігання його у файл для подальшої обробки та автентифікації.

3.3.5 Програмна реалізація загальної логіки авторизації

Перед проектованою системою стоїть 2 основні задачі моніторингу користувача – авторизувати користувача при вході у систему та періодично перевіряти, чи саме цей користувач у даний час використовує обліковий запис. Тому основний виконуваний файл буде спрацьовувати при першому запуску програми та з періодичністю, яку буде встановлено.

Для цього занесемо весь основний виконуваний код до функції та будемо викликати її при запуску та з періодичністю в 10 хвилин за допомогою методу time.sleep().

Першим кроком почнемо роботу з базою даних. Ініціюємо підключення та створимо таблиці для зберігання логінів, паролів, характеристик голосу користувачів та зберігання логів авторизації.

```
db = sqlite3.connect("data.db")
sql = db.cursor()
sql.execute("""CREATE TABLE IF NOT EXISTS "users" (
   "username"  TEXT,
   "password"  TEXT,
   "voice"  BLOB)""")
sql.execute("""CREATE TABLE IF NOT EXISTS "logs" (
   "username"  TEXT,
   "type"  TEXT,
   "time"  TEXT)""")
db.commit()
```

Наступним кроком реалізуємо логіку процесу реєстрації, оскільки для подальшого процесу авторизації за допомогою зліпку зображення та голосу, користувача спочатку треба зареєструвати. Для реєстрації користувачу потрібно буде вказати свій логін, пароль, за допомогою мікрофону зробити запис голосу та захопити зображення з веб-камери для створення його векторного ембендінгу для подальшого проведення авторизації. Також при виконанні реєстрації ми заносимо до бази даних лог з інформацією, що такого користувача було зареєстровано у такий час.

```
def reg(login, password, imgpath, voicepath):
    username = login
    password = password
    imgpath = imgpath
    voicepath = voicepath

    sql.execute(f"SELECT username, password FROM users WHERE username = '{username}' AND password =
'{password}'")
    if sql.fetchone() is None:
        current_datetime = datetime.now()
        sql.execute(f"INSERT INTO logs VALUES (?,?,?)", (username, "регистрация", str(current_datetime)))
        voice_features = extract_features(voicepath)
        with open(voicepath, "rb") as f:
            voice_data = f.read()
        sql.execute(f"INSERT INTO users VALUES (?,?,?)", (username, password, voice_data))
        db.commit()
        path_s = imgpath
        path_d = 'images/' + login
        os.mkdir(path_d)
        shutil.move(imgpath, path_d)
        return True
    else:
        return False
```

Також за допомогою бібліотеки tkinter створимо тестовий інтерфейс для зручності взаємодії з проектованим ПЗ.

```
def registration():
    window3 = Tk()
    window3.title('Реестрація')
    window3.geometry('450x270')
    window3.resizable(False, False)

    def clicked():
        username = username_entry.get()
        password = password_entry.get()
        voicepath = 'voices/' + username + '.wav'
        record_voice(voicepath)
        imgpath = capture_image(username)
        if reg(username, password, voicepath):
            messagebox.showinfo('Успішно зареєстровано', 'Успішно зареєстровано')
            window3.withdraw()
        else:
```

```
            messagebox.showinfo('Відмовлено', 'Такий користувач вже існує')

    main_label = Label(window3, text='Реєстрація', font=font_header, justify=CENTER, **header_padding)
    main_label.pack()
    username_label = Label(window3, text='Ім\'я користувача', font=label_font, **base_padding)
    username_label.pack()
    username_entry = Entry(window3, bg='#fff', fg='#444', font=font_entry)
    username_entry.pack()
    password_label = Label(window3, text='Пароль', font=label_font, **base_padding)
    password_label.pack()
    password_entry = Entry(window3, bg='#fff', fg='#444', font=font_entry)
    password_entry.pack()
    send_btn = Button(window3, text='Зареєструватися', command=clicked)
    send_btn.pack(**base_padding)
    window2.withdraw()
```

Цей інтерфейс буде звертатися до наведеної вище функції, яка відповідно при успішному процесі реєстрації буде вертати True та давати користувачу зрозуміти, що процес реєстрації пройдено успішно.

Далі напишемо логіку безпосередньо авторизації користувача. Почнемо з функції auth(), задачою якої буде первинна перевірка на логін та пароль користувача та перевірка співпадання характеристик голосу зі збереженими у базі даних.

```
def auth():
    window = Tk()
    window.title('Авторизація')
    window.geometry('450x230')
    window.resizable(False, False)

    def clicked():
        username = username_entry.get()
        password = password_entry.get()
        voicepath = 'voices/' + username + '_auth.wav'
        record_voice(voicepath)
        if login(username, password, voicepath):
            imgpath = capture_image(username)
            window.withdraw()
            approove(username, imgpath)
        else:
            messagebox.showinfo('Відмовлено', 'Будь ласка, перевірте правильність введення')
    main_label = Label(window, text='Авторизація', font=font_header, justify=CENTER, **header_padding)
    main_label.pack()
    username_label = Label(window, text='Ім\'я користувача', font=label_font, **base_padding)
    username_label.pack()
    username_entry = Entry(window, bg='#fff', fg='#444', font=font_entry)
    username_entry.pack()
    password_label = Label(window, text='Пароль', font=label_font, **base_padding)
    password_label.pack()
    password_entry = Entry(window, bg='#fff', fg='#444', font=font_entry)
    password_entry.pack()

    send_btn = Button(window, text='Увійти', command=clicked)
```

```
  send_btn.pack(**base_padding)
  window2.withdraw()
```

Наведена функція відкриває нове вікно з заголовком "Авторизація". Коли користувач натискає кнопку "Увійти", зчитуються значення, введені в поля для імені користувача та пароля та записується голос користувача у файл з ім'ям користувача та позначкою "auth".

Наступним кроком викликається функція login, яка перевіряє відповідність введених даних та голосу з тими, що збережені в базі даних. Якщо дані правильні і голос співпадає, вікно авторизації закривається, і викликається функція approove для подальшої перевірки. Якщо дані неправильні або голос не співпадає, відображається повідомлення про помилку з проханням перевірити введені дані.

Для перевірки наявності такого користувача у базі даних напишемо безпосередньо функцію login для взаємодії з базою даних.

```
def login(login, password, voicepath):
  username = login
  password = password
  sql.execute(
    f"SELECT username, password, voice FROM users WHERE username = '{username}' AND password =
'{password}'")
  user_data = sql.fetchone()
  if not user_data:
    return False
  else:
    recorded_voice_features = extract_features(voicepath)
    stored_voice_data = user_data[2]
    stored_voice_path = 'stored_voice.wav'
    with open(stored_voice_path, 'wb') as f:
      f.write(stored_voice_data)
    stored_voice_features = extract_features(stored_voice_path)
    if compare_voices(recorded_voice_features, stored_voice_features) < 10:  # Порогове значення можна
налаштувати
      current_datetime = datetime.now()
      sql.execute(f"INSERT INTO logs VALUES (?,?,?)", (username, "авторизація", str(current_datetime)))
      db.commit()
      return True
    else:
      return False
```

Це доволі проста функція, метою якої є прийняти на вхід логін та пароль, перевірити наявність такого користувача у базі даних та надати на вихід True якщо логін та пароль співпадають та відповідно False, якщо не співпадають.

Також при виконанні авторизації ми заносимо до бази даних лог з інформацією, що такого користувача було авторизовано у такий час.

Якщо цей етап авторизації пройдено користувача буде перенаправлено на безпосередньо авторизацію за допомогою розпізнавання зліпку зображення його обличчя. Реалізуємо цей функціонал у функції approve().

```
def approove(username, imgpath):
    window4 = Tk()
    window4.title('Підтвердження входу')
    window4.geometry('450x270')
    window4.resizable(False, False)

    def clicked():
        if process.check(imgpath, username):
            messagebox.showinfo('Успішна авторизація', 'Успішна авторизація')
            window4.withdraw()
        else:
            messagebox.showinfo('Відмовлено', 'Перевірка не пройдена')
            window4.withdraw()

    main_label = Label(window4, text='Підтвердження входу', font=font_header,
justify=CENTER, **header_padding)
    main_label.pack()

    send_btn = Button(window4, text='Перевірити', command=clicked)
  send_btn.pack(**base_padding3)
```

Ця функція приймає на вхід логін користувача та шлях до зображення, яке було захоплено раніше. Функція створює вікно підтвердження входу, в якому користувач повинен підтвердити свою особу.

При натисканні кнопки "Перевірити" викликається функція clicked, яка використовує модуль розпізнавання для перевірки відповідності зображення з веб-камери та зареєстрованого користувача. Якщо розпізнавання успішне, виводиться повідомлення про успішну авторизацію та вікно закривається. В іншому випадку виводиться повідомлення про невдалу перевірку та вікно закривається.

Наступним кроком реалізуємо саме поведінку процесу моніторингу за користувачем. Для цього реалізуємо функцію monitoring(), яка буде з встановленою нами періодичністю проводити моніторинг користувача за допомогою зліпку його обличчя. Окрім цього в базу даних треба безпосередньо заносити лог з інформацією про те, що користувач пройшов моніторинг чи ні.

```
def monitoring():
    sql.execute(f"SELECT * FROM logs")
    answer = sql.fetchall()
    username = 1
    for i in answer:
        if i[1] == 'авторизація' or i[1] == 'реєстрація':
            username = i[0]
    db.commit()
    window6 = Tk()
    window6.title('Система моніторингу')
    window6.geometry('450x230')
    window6.resizable(False, False)

    def clicked():
        imgpath = capture_image(username)
        voicepath = 'voices/' + username + '_monitoring.wav'
        record_voice(voicepath)
        if process.check(imgpath, username) and login(username, password, voicepath):
            messagebox.showinfo('Успіх', 'Ви успішно підтвердили свою сесію')
            current_datetime = datetime.now()
            sql.execute(f"INSERT INTO logs VALUES (?,?,?)", (username, "успішна сесія моніторингу",
str(current_datetime)))
            window6.withdraw()
        else:
            messagebox.showinfo('Відмовлено', 'Користувач змінений, будь ласка, пройдіть авторизацію знову')
            current_datetime = datetime.now()
            sql.execute(f"INSERT INTO logs VALUES (?,?,?)", (username, "помилка моніторингу",
str(current_datetime)))
            window6.withdraw()
            window2.mainloop()

    main_label = Label(window6, text='Будь ласка, пройдіть моніторинг системи', font=font_header,
justify=CENTER, **header_padding)
    main_label.pack()

    send_btn = Button(window6, text='Перевірити', command=clicked)
    send_btn.pack(**base_padding3)
    window2.withdraw()
    window6.mainloop()
```

Реалізована функція monitoring при виклику робить запит до бази даних логів, щоб отримати логін останнього авторизованого у системі на даному пристрої користувача, після чого зберігає його у змінній. Наступним кроком система формує екран моніторингу, де запитує у користувача його зображення за допомогою веб-камери. Після проведення процесу розпізнавання, система перевірить, чи співпадає поточний авторизований користувач з користувачем, який на даний момент використовує пристрій. Якщо користувача буде підтверджено, система сповістить його про це та додасть запис у базу даних про успішний моніторинг. У зворотньому випадку користувач отримає повідомлення про помилку та буде перенаправлений до екрану автентифікації.

3.4 Тестування розробленого програмного забезпечення

Для перевірки працездатності розробленого програмного забезпечення проведемо тестування основних його функцій , а саме авторизації користувача, реєстрації користувача, результату порівняння за зліпком обличчя та голосу.

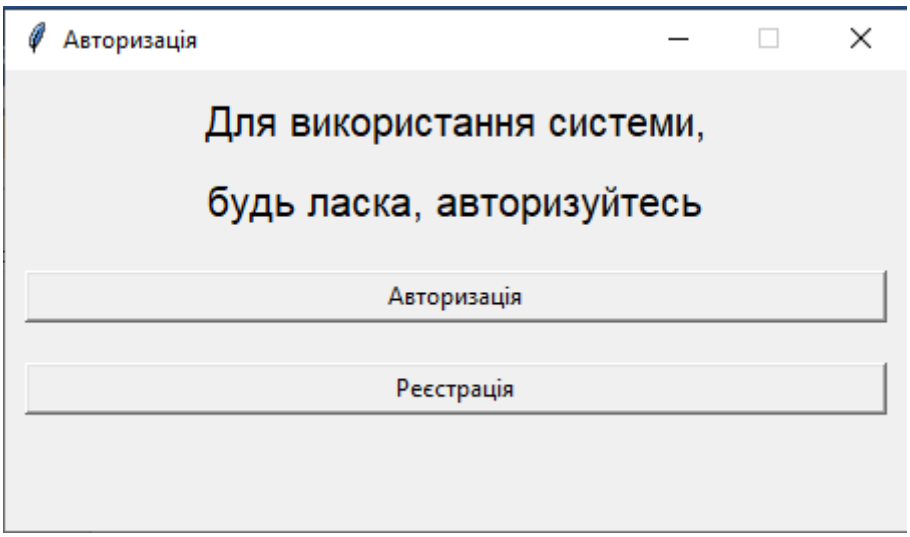


Рисунок 3.11 – Інтерфейс вибору першої дії

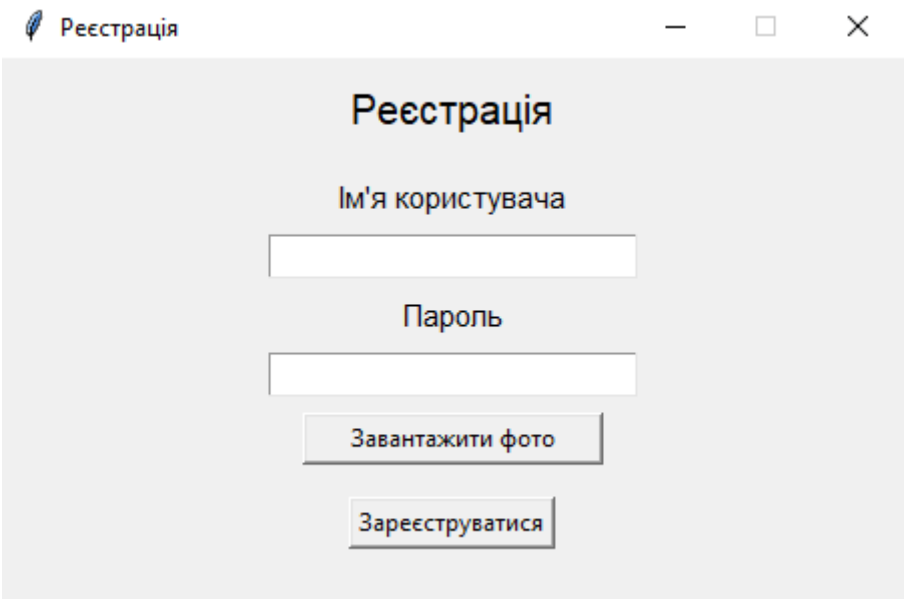


Рисунок 3.12 – Інтерфейс реєстрації користувача для подальшої авторизації

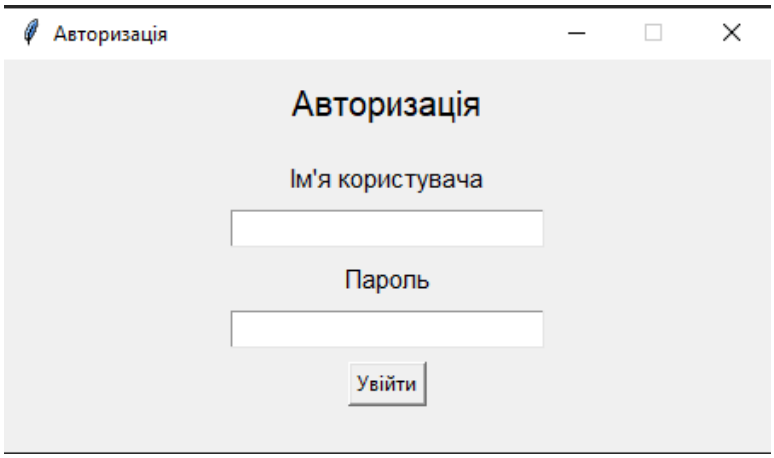


Рисунок 3.13 – Інтерфейс першого етапу авторизації користувача та перевірки його голосу при натисканні «Увійти»

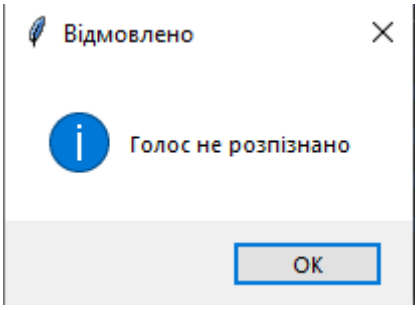


Рисунок 3.14 – Помилка розпізнавання при невідповідності голосу

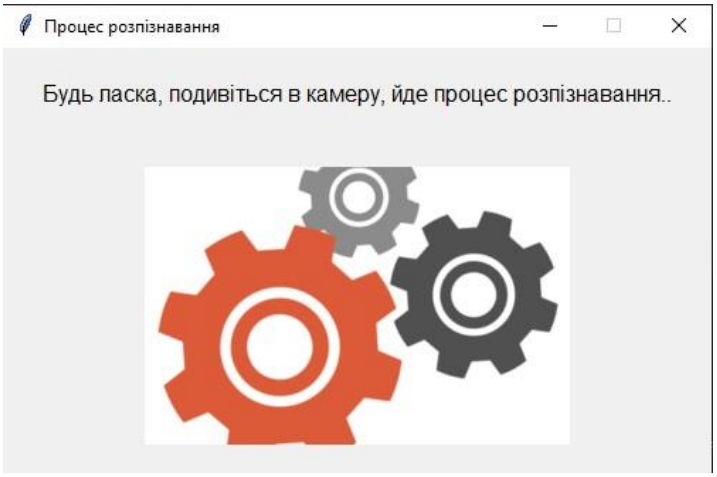


Рисунок 3.15 – Інтерфейс процесу розпізнавання обличчя

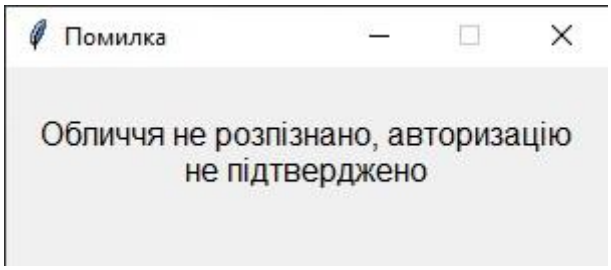


Рисунок 3.16 – Помилка розпізнавання при розпізнаванні обличчя

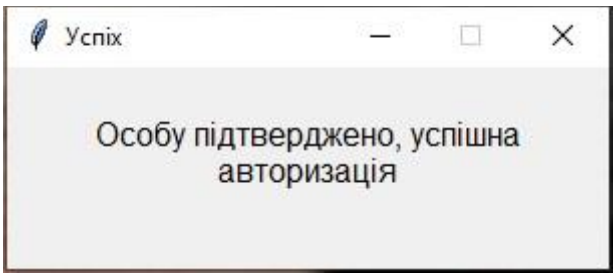


Рисунок 3.17 – Результат успішної авторизації за розпізнаванням обличчя та голосу користувача

3.5 Порівняння розробленого ПЗ з аналогами

Останнім кроком проведемо порівняння розробленого програмного забезпечення з аналогами, які було проаналізовано у розділі 1.5. Порівняльну характеристику наведено у таблиці 3.1.

Таблиця 3.1 - Порівняльна характеристика програмного забезпечення

Розробник	Aiwe.pl	Reklama.LVC	Webera	Виконання дипломної роботи
Назва ПЗ	ZkTeco	Privatbank	IIdX	-
Функціонал	Даний сервіс представлено у вигляді веб-додатку для продажу комерційних послуг з розпізнавання обличчя, який	Даний сервіс є можливістю авторизуватися у особистому кабінеті банку «Privatbank» за допомогою завантаження фото,	Даний сервіс представлено у вигляді API сервісу для автоматизації процесу розпізнавання користувача для бізнесу. Сервіс	Даний сервіс представлено у вигляді exe програми, яка може виконуватись при запуску ПК користувачем та з встановленою в налаштуваннях

	можна використовувати для процесів автентицікації користувача чи мониторингу за ним Сервіс надає можливість технічної підтримки користувачів, має можливість авторизації та реєстрації. Для використання сервісу користувач має придбати послугу на сайті сервісу, після чого співробітники інтегрують процеси розпізнавання у бізнес користувача	зробленого на телефон. Ця функція була додана нещодавно і зараз активно тестується. При завантаженні фото алгоритми банку проводять розпізнавання особистості користувача на фото та порівнюють його з існуючим у базі даних користувачем. Якщо особу користувача підтверджено – йому буде надано доступ до особистого кабінету користувача.	надає можливість технічної підтримки користувачів, має можливість авторизації та реєстрації. Для використання сервісу користувач має придбати послугу на сайті сервісу та вподальшому інтегрувати її до свого проекту. Сервіс приймає на вхід фото користувача, виконує розпізнавання та повертає результат.	періодичністю виконувати функціонал авторизації поточного користувача за допомогою розпізнавання зліпку його обличчя та голосу та проводити моніторинг подальшої присутності саме цього користувача на робочому місці. Сервіс має максимально просту взаємодію, розпізнає користувача за допомогою методу власних векторів та порівняння мел – частотних кепстральних коефіцієнтів.
Інтерфейс користувача	Є інтуітивно незрозумілим та перегруженим.	Є інтуїтивно зрозумілим, колірна гама підібрана вдало.	Є інтуїтивно зрозумілим, колірна гама підібрана вдало.	Є мінімалістичним та максиально зрозумілим.
Переваги	Якісний аналіз фото, захист	Висока ступінь захисту,	Зручний спосіб впровадження у	Повністю безкоштовний,простий

	від невалідної автентифікації	зручність використання.	проект у вигляді API, обширна документація щодо взаємодії з сервісом.	інтерфейс взаємодії, можливість моніторингу користувача.
Недоліки	Складний для звичайного користувача спосіб впровадження, дороговизна придбання ПЗ.	Зашифровані методи ідентифікування користувача, немає можливості моніторингу.	Дороговизна придбання даного програмного забезпечення, неліквідність для малих проектів.	Недостатня маштабованість, відсутність документації, відсутність технічної підтримки користувачів.

3.5 Висновок до розділу 3

В заключному розділі було здійснено докладний опис процесу програмної реалізації проектованого програмного забезпечення, включаючи використані бібліотеки та модулі. Це дозволяє відтворити процес розробки та забезпечити якісну реалізацію проекту.

Проведено тестування розробленого програмного забезпечення, що підтвердило його ефективність та надійність у виконанні завдань автентифікації користувачів. Це важливий етап для забезпечення відповідності програмного продукту вимогам безпеки та надійності.

ВИСНОВКИ

У результаті виконання дипломної роботи було проведено глибокий аналіз методів автентифікації користувачів за допомогою біометричних даних, зокрема розпізнавання обличчя та голосу. Було розглянуто актуальність використання таких методів у сучасних умовах кібербезпеки, їх переваги та недоліки, а також проаналізовано існуючі системи-аналоги.

У ході роботи було обрано метод розпізнавання обличчя за допомогою власних векторів (Eigenfaces) як найбільш оптимальний для реалізації поставлених задач. Проведено детальний аналіз цього методу та продемонстровано його ефективність у порівнянні з іншими методами розпізнавання. Також було розглянуто та впроваджено додатковий механізм автентифікації за допомогою розпізнавання голосу. Для цього було використано бібліотеки speech_recognition та librosa, які дозволяють ефективно обробляти аудіо дані та здійснювати їх порівняння. Було реалізовано функції запису голосу, витягнення аудіо характеристик та їх порівняння, що забезпечило додатковий рівень безпеки при автентифікації користувачів.

Розроблений програмний продукт було реалізовано на мові програмування Python, що дозволило скористатися багатим набором бібліотек для роботи з обробкою зображень та аудіо. Для створення графічного інтерфейсу було використано бібліотеку Tkinter, яка забезпечує простоту та зручність у використанні.

Загалом, результати виконаної роботи демонструють ефективність використання біометричних даних для автентифікації користувачів та підтверджують доцільність впровадження таких методів у сучасних системах захисту інформації. Розроблений програмний продукт має широкий спектр застосування та може бути використаний у різних сферах для підвищення рівня безпеки та надійності систем автентифікації.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Wang, M., & Deng, W. (2018). Deep Face Recognition: A Survey. arXiv preprint arXiv:1804.06655.

2. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). MobileNetV2: Inverted Residuals and Linear Bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 45-46).

3. Jiang, Y., Wu, Y., Feng, X., Mao, J., & Liu, W. (2017). Facial Landmark Detection via Spatially Weighted Ensemble of Regression Trees. IEEE Transactions on Pattern Analysis and Machine Intelligence, 39(8), 151-153.

4. Parkhi, O. M., Vedaldi, A., & Zisserman, A. (2015). Deep Face Recognition. In British Machine Vision Conference (BMVC) (Vol. 1, No. 3, p. 6).

5. Taigman, Y., Yang, M., Ranzato, M. A., & Wolf, L. (2014). DeepFace: Closing the Gap to Human-Level Performance in Face Verification. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 170-178).

6. Schroff, F., Kalenichenko, D., & Philbin, J. (2015). FaceNet: A Unified Embedding for Face Recognition and Clustering. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 81-83).

7. Sun, Y., Wang, X., & Tang, X. (2014). Deep Learning Face Representation by Joint Identification-Verification. In Advances in Neural Information Processing Systems (pp. 18-19).

8. Cao, Q., Shen, L., Xie, W., Parkhi, O. M., & Zisserman, A. (2018). VGGFace2: A Dataset for Recognising Faces across Pose and Age. In 2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018) (pp. 67-74).

9. Masi, I., Tran, A. T., Hassner, T., Leksut, J. T., & Medioni, G. (2016). Do We Really Need to Collect Millions of Faces for Effective Face Recognition? In European Conference on Computer Vision (pp. 57-59).

10. Zhao, W., Chellappa, R., Phillips, P. J., & Rosenfeld, A. (2003). Face Recognition: A Literature Survey. ACM Computing Surveys (CSUR), 35(4), 39-45.

11. Zhou, Z. H., & Zhang, C. (2017). A Comprehensive Review on Deep Learning-Based Methods for Face Recognition. International Journal of Automation and Computing, 14(4), 40-42.

12. Ahonen, T., Hadid, A., & Pietikäinen, M. (2006). Face Description with Local Binary Patterns: Application to Face Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 28(12), 203-204.

13. Turk, M., & Pentland, A. (1991). Eigenfaces for Recognition. Journal of Cognitive Neuroscience, 3(1), 71-86.

14. Phillips, P. J., Moon, H., Rizvi, S. A., & Rauss, P. J. (2000). The FERET Evaluation Methodology for Face-Recognition Algorithms. IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(10), 109-110.

15. Grother, P., Quinn, G. W., & Phillips, P. J. (2010). Report on the Evaluation of 2D Still-Image Face Recognition Algorithms. National Institute of Standards and Technology, NISTIR 7709.

16. Blanz, V., & Vetter, T. (2003). Face Recognition Based on Fitting a 3D Morphable Model. IEEE Transactions on Pattern Analysis and Machine Intelligence, 25(9), 106-107.

17. Li, S. Z., & Jain, A. K. (Eds.). (2011). Handbook of Face Recognition. Springer.

18. Dantcheva, A., Elia, P., & Ross, A. (2016). What Else Does Your Biometric Data Reveal? A Survey on Soft Biometrics. IEEE Transactions on Information Forensics and Security, 11(3), 44-46.

ДОДАТКИ

Додаток А. Технічне завдання

Вінницький національний технічний університет
Факультет менеджменту та інформаційної безпеки
Кафедра менеджменту та безпеки інформаційних систем

ЗАТВЕРДЖУЮ
Голова секції
“Управління інформаційною
безпекою” кафедри МБІС
д.т.н., професор
_____Юрій ЯРЕМЧУК
“ 22 ” березня 2024 р.

ТЕХНІЧНЕ ЗАВДАННЯ
до бакалаврської дипломної роботи на тему:

Розробка програми комплексної автентифікації до системи захисту звикористанням ідентифікації за допомогою зліпка обличчя та голосу.
08-72.БДР.004.00.100 ТЗ

Керівник бакалаврської дипломної роботи
к.т.н., доц. каф. МБІС
_____ Грицак А. В.
“____”_____

Вінниця – 2024 р.

1. Найменування та область застосування.

Розробка програми комплексної автентифікації до системи захисту з використанням ідентифікації за допомогою зліпка обличчя та голосу. Область застосування: програмне забезпечення для комплексної автентифікації до системи захисту з використанням ідентифікації за допомогою зліпка обличчя та голосу.

2.Підстава для розробки

Розробка виконується на основі наказу ректора ВНТУ № 80 від 11.03.2024 р.

3. Мета та призначення розробки

3.1 Мету дослідження, яка полягає в розробці та удосконаленні процесу автентифікації користувачів інформаційних систем шляхом розробки та впровадження програмного забезпечення для комплексної автентифікації з використанням зліпка обличчя та голосу.

3.2 Розробка програми комплексної автентифікації до системи захисту звикористанням ідентифікації за допомогою зліпка обличчя та голосу.

4. Джерела розробки

4.1. Cao, Q., Shen, L., Xie, W., Parkhi, O. M., & Zisserman, A. (2018). VGGFace2: A Dataset for Recognising Faces across Pose and Age. In 2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018) (pp. 67-74).

4.2. Dantcheva, A., Elia, P., & Ross, A. (2016). What Else Does Your Biometric Data Reveal? A Survey on Soft Biometrics. IEEE Transactions on Information Forensics and Security, 11(3), 44-46.

4.3. Taigman, Y., Yang, M., Ranzato, M. A., & Wolf, L. (2014). DeepFace: Closing the Gap to Human-Level Performance in Face Verification. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 170-178).

4.4. Wang, M., & Deng, W. (2018). Deep Face Recognition: A Survey. arXiv preprint arXiv:1804.06655.

5. Вимоги до програми

5.1 Вимоги до функціональних характеристик:

5.1.1 Програмний засіб повинен мати зручний, легкий у використанні інтерфейс користувача;

5.1.2 Реалізація методу не повинна вимагати спеціальних ліцензійних програмних додатків;

5.1.3 Програмний засіб повинен виконувати процес автентифікації користувачів у системі.

5.2 Вимоги до надійності:

5.2.1 Програмний засіб повинен працювати без помилок, у випадку виникнення критичних ситуацій необхідно передбачити виведення відповідних повідомлень;

5.2.2 Програмний засіб повинен виконувати свої функції.

5.3 Вимоги до складу і параметрів технічних засобів:

Даний проект реалізований мовою програмування Python.

– процесор – Pentium 1500 МГц і подібні до них;

– оперативна пам’ять – не менше 1024 Mb;

– веб-камера – Logitech не менше 360p;

– мікрофон – діапазон яких становить від 20 - 20000 Гц;

6. Вимоги до програмної документації

6.1 Обов’язкова поетапна інструкція для майбутніх користувачів, наведена у пункті 2.2

7. Вимоги до технічного захисту інформації

7.1 Необхідно забезпечити захист розроблюваного програмного засобу від несанкціонованого використання.

7.2 Неможливість отримання доступу незареєстрованих користувачів до інформаційних ресурсів.

8. Техніко-економічні показники

8.1 Цінність результатів використання даного проекту повинна перевищувати витрати на його реалізацію.

8.2 Має бути реалізований таким чином, щоб підходити для використання широкого загалу.

9. Стадії та етапи розробки

№ з/п	Назва етапів бакалаврської дипломної роботи	Початок	Закінчення
1.	Визначення напрямку бакалаврської роботи, формулювання теми	21.03.2024	22.03.2024
2.	Аналіз предметної області обраної теми	23.03.2024	21.04.2024
3.	Розробка алгоритму роботи	10.05.2024	18.05.2024
4.	Написання бакалаврської роботи на основі розробленої теми	18.05.2024	25.05.2024
5.	Передзахист бакалаврської дипломної роботи	26.05.2024	27.05.2024
6.	Виправлення, уточнення, корегування бакалаврської дипломної роботи	28.05.2024	06.06.2024
7.	Захист бакалаврської дипломної роботи	12.06.2024	17.06.2024

10. Порядок контролю та прийому

10.1 До приймання бакалаврської дипломної роботи надаються:

- ПЗ до бакалаврської дипломної роботи;
- демонстрація результату бакалаврської дипломної роботи;
- презентація;
- відзив керівника роботи;
- відзив рецензента

Технічне завдання до виконання прийняв _____ Генюк О.І.

Додаток Б. Лістинг програми

Лістинг основного виконуваного файлу

```
from tkinter import *
import tkinter as tk
from tkinter import messagebox
import sqlite3
import os
import shutil
import process
import time
from datetime import datetime
import speech_recognition as sr
import numpy as np
import librosa
import cv2

db = sqlite3.connect("data.db")
sql = db.cursor()
sql.execute("""CREATE TABLE IF NOT EXISTS "users" (
   "username"  TEXT,
   "password"  TEXT,
   "voice"  BLOB)""")
sql.execute("""CREATE TABLE IF NOT EXISTS "logs" (
   "username"  TEXT,
   "type"  TEXT,
   "time"  TEXT)""")
db.commit()

def record_voice(filename):
   r = sr.Recognizer()
   with sr.Microphone() as source:
      print("Будь ласка, скажіть щось...")
      audio = r.listen(source)
      with open(filename, "wb") as f:
         f.write(audio.get_wav_data())

def extract_features(filename):
   y, sr = librosa.load(filename)
   mfccs = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=13)
   return np.mean(mfccs.T, axis=0)

def compare_voices(features1, features2):
   return np.linalg.norm(features1 - features2)

def reg(login, password, voicepath):
   username = login
   password = password
   voicepath = voicepath

   sql.execute(f"SELECT username, password FROM users WHERE username = '{username}' AND password =
'{password}'")
   if sql.fetchone() is None:
      current_datetime = datetime.now()
      sql.execute(f"INSERT INTO logs VALUES (?,?,?)", (username, "реєстрація", str(current_datetime)))
      voice_features = extract_features(voicepath)
      with open(voicepath, "rb") as f:
```

```
            voice_data = f.read()
        sql.execute(f"INSERT INTO users VALUES (?,?,?,?)", (username, password, voice_data, ""))
        db.commit()
        return True
    else:
        return False

def login(login, password, voicepath):
    username = login
    password = password
    sql.execute(f"SELECT username, password, voice FROM users WHERE username = '{username}' AND password =
'{password}'")
    user_data = sql.fetchone()
    if not user_data:
        return False
    else:
        recorded_voice_features = extract_features(voicepath)
        stored_voice_data = user_data[2]
        stored_voice_path = 'stored_voice.wav'
        with open(stored_voice_path, 'wb') as f:
            f.write(stored_voice_data)
        stored_voice_features = extract_features(stored_voice_path)
        if compare_voices(recorded_voice_features, stored_voice_features) < 10:  # Порогове значення можна
налаштувати
            current_datetime = datetime.now()
            sql.execute(f"INSERT INTO logs VALUES (?,?,?)", (username, "авторизація", str(current_datetime)))
            db.commit()
            return True
        else:
            return False

def capture_image(username):
    cap = cv2.VideoCapture(0)
    while True:
        ret, frame = cap.read()
        cv2.imshow('Press Space to Capture', frame)
        if cv2.waitKey(1) & 0xFF == ord(' '):  # Press Space to capture
            img_path = f'images/{username}.jpg'
            cv2.imwrite(img_path, frame)
            break
    cap.release()
    cv2.destroyAllWindows()
    return img_path

# Головне вікно програми
window2 = Tk()
window2.title('Авторизація')
window2.geometry('450x230')
window2.resizable(False, False)

# Настройки шрифтів і відступів
font_header = ('Arial', 15)
font_entry = ('Arial', 12)
label_font = ('Arial', 11)
base_padding = {'padx': 10, 'pady': 8}
base_padding2 = {'padx': 10, 'pady': 20}
base_padding3 = {'padx': 10, 'pady': 60}
header_padding = {'padx': 10, 'pady': 12}
```

```
base_padding4 = {'padx': 10, 'pady': 0}

def auth():
    window = Tk()
    window.title('Авторизація')
    window.geometry('450x230')
    window.resizable(False, False)

    def clicked():
        username = username_entry.get()
        password = password_entry.get()
        voicepath = 'voices/' + username + '_auth.wav'
        record_voice(voicepath)
        if login(username, password, voicepath):
            imgpath = capture_image(username)
            window.withdraw()
            approove(username, imgpath)
        else:
            messagebox.showinfo('Відмовлено', 'Будь ласка, перевірте правильність введення')

    main_label = Label(window, text='Авторизація', font=font_header, justify=CENTER, **header_padding)
    main_label.pack()
    username_label = Label(window, text='Ім\'я користувача', font=label_font, **base_padding)
    username_label.pack()
    username_entry = Entry(window, bg='#fff', fg='#444', font=font_entry)
    username_entry.pack()
    password_label = Label(window, text='Пароль', font=label_font, **base_padding)
    password_label.pack()
    password_entry = Entry(window, bg='#fff', fg='#444', font=font_entry)
    password_entry.pack()

    send_btn = Button(window, text='Увійти', command=clicked)
    send_btn.pack(**base_padding)

    window2.withdraw()

def registration():
    window3 = Tk()
    window3.title('Реєстрація')
    window3.geometry('450x270')
    window3.resizable(False, False)

    def clicked():
        username = username_entry.get()
        password = password_entry.get()
        voicepath = 'voices/' + username + '.wav'
        record_voice(voicepath)
        imgpath = capture_image(username)
        if reg(username, password, voicepath):
            messagebox.showinfo('Успішно зареєстровано', 'Успішно зареєстровано')
            window3.withdraw()
        else:
            messagebox.showinfo('Відмовлено', 'Такий користувач вже існує')

    main_label = Label(window3, text='Реєстрація', font=font_header, justify=CENTER, **header_padding)
    main_label.pack()
    username_label = Label(window3, text='Ім\'я користувача', font=label_font, **base_padding)
    username_label.pack()
```

```
    username_entry = Entry(window3, bg='#fff', fg='#444', font=font_entry)
    username_entry.pack()
    password_label = Label(window3, text='Пароль', font=label_font, **base_padding)
    password_label.pack()
    password_entry = Entry(window3, bg='#fff', fg='#444', font=font_entry)
    password_entry.pack()

    send_btn = Button(window3, text='Зареєструватися', command=clicked)
    send_btn.pack(**base_padding)

    window2.withdraw()

def approove(username, imgpath):
    window4 = Tk()
    window4.title('Підтвердження входу')
    window4.geometry('450x270')
    window4.resizable(False, False)

    def clicked():
        if process.check(imgpath, username):
            messagebox.showinfo('Успішна авторизація', 'Успішна авторизація')
            window4.withdraw()
        else:
            messagebox.showinfo('Відмовлено', 'Перевірка не пройдена')
            window4.withdraw()

    main_label = Label(window4, text='Підтвердження входу', font=font_header, justify=CENTER,
**header_padding)
    main_label.pack()

    send_btn = Button(window4, text='Перевірити', command=clicked)
    send_btn.pack(**base_padding3)

def monitoring():
    sql.execute(f"SELECT * FROM logs")
    answer = sql.fetchall()
    username = 1
    for i in answer:
        if i[1] == 'авторизація' or i[1] == 'реєстрація':
            username = i[0]
    db.commit()
    window6 = Tk()
    window6.title('Система моніторингу')
    window6.geometry('450x230')
    window6.resizable(False, False)

    def clicked():
        imgpath = capture_image(username)
        voicepath = 'voices/' + username + '_monitoring.wav'
        record_voice(voicepath)
        if process.check(imgpath, username) and login(username, password, voicepath):
            messagebox.showinfo('Успіх', 'Ви успішно підтвердили свою сесію')
            current_datetime = datetime.now()
            sql.execute(f"INSERT INTO logs VALUES (?,?,?)", (username, "успішна сесія моніторингу",
str(current_datetime)))
            window6.withdraw()
        else:
            messagebox.showinfo('Відмовлено', 'Користувач змінений, будь ласка, пройдіть авторизацію знову')
```

```
        current_datetime = datetime.now()
        sql.execute(f"INSERT INTO logs VALUES (?,?,?)", (username, "помилка моніторингу",
str(current_datetime)))
        window6.withdraw()
        window2.mainloop()

    main_label = Label(window6, text='Будь ласка, пройдіть моніторинг системи', font=font_header,
justify=CENTER, **header_padding)
    main_label.pack()

    send_btn = Button(window6, text='Перевірити', command=clicked)
    send_btn.pack(**base_padding3)
    window2.withdraw()
    window6.mainloop()

main_label = Label(window2, text='Для використання системи,', font=font_header, justify=CENTER,
**header_padding)
main_label.pack()
main_label2 = Label(window2, text='будь ласка, авторизуйтесь', font=font_header, justify=CENTER)
main_label2.pack()

send_btn2 = Button(window2, text='Авторизація', width=100, command=auth)
send_btn3 = Button(window2, text='Реєстрація', width=100, command=registration)
send_btn2.pack(**base_padding2)
send_btn3.pack(**base_padding4)

window2.mainloop()

while True:
    time.sleep(600)
    monitoring()
```

Лістинг модулю аналізу наявних зображень та створення датасету

```
from imutils import paths
import face_recognition
import pickle
import cv2
import os

def analytic():
    # В директорії Images зберігаються папки з усіма зображеннями
    imagePaths = list(paths.list_images('Images'))
    knownEncodings = []
    knownNames = []

    # Створюємо пустий файл, якщо він не існує
    if not os.path.exists("face_enc"):
        with open("face_enc", "wb") as f:
            pass

    # Перебираємо всі папки з зображеннями
    for (i, imagePath) in enumerate(imagePaths):
        # Витягуємо ім'я людини з назви папки
        name = imagePath.split(os.path.sep)[-2]
        # Завантажуємо зображення і конвертуємо його з BGR (OpenCV ordering)
        # в dlib ordering (RGB)
        image = cv2.imread(imagePath)
        rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
```

```
        # Використовуємо бібліотеку Face_recognition для виявлення облич
        boxes = face_recognition.face_locations(rgb, model='hog')
        # Обчислюємо ембеддінги для кожного обличчя
        encodings = face_recognition.face_encodings(rgb, boxes)
        # Перебираємо всі ембеддінги
        for encoding in encodings:
            knownEncodings.append(encoding)
            knownNames.append(name)

    # Збережемо ембеддінги разом з їх іменами в форматі словника
    data = {"encodings": knownEncodings, "names": knownNames}

    # Для збереження даних у файл використовуємо метод pickle
    with open("face_enc", "wb") as f:
        f.write(pickle.dumps(data))

# Викликаємо функцію для створення файлу face_enc
analytic()
```

Лістинг модулю аналізу зліпків обличь

```
import face_recognition
import imutils
import pickle
import time
import cv2
import os
import sqlite3


def check(imgpath, login):
    cascPathface = os.path.dirname(
        cv2.__file__) + "/data/haarcascade_frontalface_alt2.xml"
    faceCascade = cv2.CascadeClassifier(cascPathface)
    data = pickle.loads(open('face_enc', "rb").read())
    image = cv2.imread(imgpath)
    rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    faces = faceCascade.detectMultiScale(gray,
                                         scaleFactor=1.1,
                                         minNeighbors=5,
                                         minSize=(60, 60),
                                         flags=cv2.CASCADE_SCALE_IMAGE)

    encodings = face_recognition.face_encodings(rgb)
    names = []
    for encoding in encodings:
        matches = face_recognition.compare_faces(data["encodings"],
                                                 encoding)
        name = "Unknown"
        if True in matches:
            matchedIdxs = [i for (i, b) in enumerate(matches) if b]
            counts = {}
            for i in matchedIdxs:
                name = data["names"][i]
                counts[name] = counts.get(name, 0) + 1
                name = max(counts, key=counts.get)
            names.append(name)
            for ((x, y, w, h), name) in zip(faces, names):
```

```
        cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0), 2)
        cv2.putText(image, name, (x, y), cv2.FONT_HERSHEY_SIMPLEX,
                    0.75, (0, 255, 0), 2)
cv2.imshow("Frame", image)
cv2.waitKey(0)
if name == login:
    print('yes')
    return True
else:
    print('nope')
    return False
```

Додаток В. Ілюстративний матеріал

Вінницький національний технічний університет

Факультет менеджменту та інформаційної безпеки

АТЕСТАЦІЙНА РОБОТА БАКАЛАВРА

«Розробка програми комплексної автентифікації до системи захисту з використанням ідентифікації за допомогою зліпка обличчя та голосу»

Виконав
ст. гр.
Геник О.І.

Керівник
Грицак А.В.

Мета та об’єкт дослідження

- ***Метою роботи*** *є розробка програмного забезпечення для автоматизації процесу автентифікації користувачів до системи захисту з використанням ідентифікації за допомогою зліпка обличчя та голосу.*
- ***Об’єктом дослідження*** *є програмне забезпечення для комплексної автентифікації до системи захисту з використанням ідентифікації за допомогою зліпка обличчя та голосу.*
- ***Практичне значення*** *- використання розробленого програмного забезпечення дозволить підвищити якість автентифікації користувачів комп'ютерних систем та покращити систему розпізнавання поточного користувача системи.*

Постановка задачі

- *Проаналізувати типи автентифікації, існуючі методи, інструменти, механізми авторизації користувачів комп'ютерних систем за зліпком обличчя та голосу.*
- *Провести аналіз та вибір інструментів реалізації проектованого програмного забезпечення.*
- *Провести проектування та програмну реалізацію програми автентифікації користувачів комп'ютерних систем з використанням зліпка обличчя та голосу.*
- *Провести тестування розробленого програмного забезпечення.*

Використані для розробки програмні засоби

Використані бібліотеки

OpenCV

Tkinter

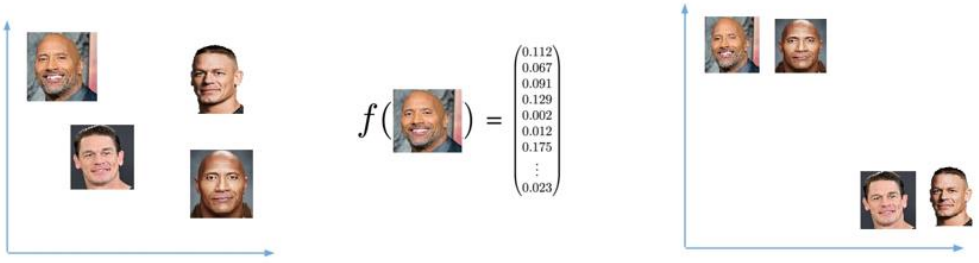
Dlib

SQLite

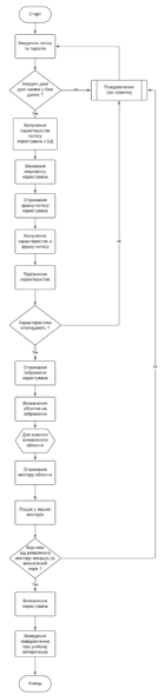
pyAudio Analysis

NumPy

Принцип розпізнавання за зліпком обличчя



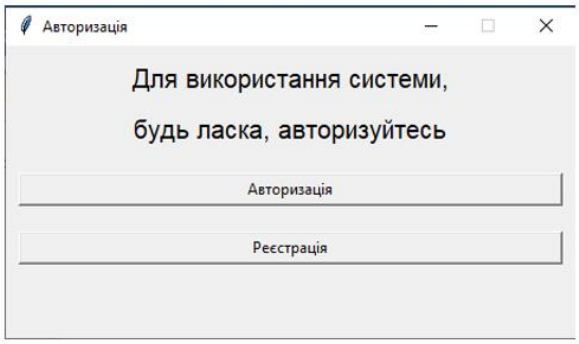
РЕАЛІЗОВАНИЙ АЛГОРИТМ ПРОЦЕСУ ВИЗНАЧЕННЯ ЗА ЗЛІПКОМ ОБЛИЧЧЯ



ЗАГАЛЬНИЙ АЛГОРИТМ АВТОРИЗАЦІЇ ЗА ДОПОМОГОЮ ОБЛИЧЧЯ ТА ГОЛОСУ

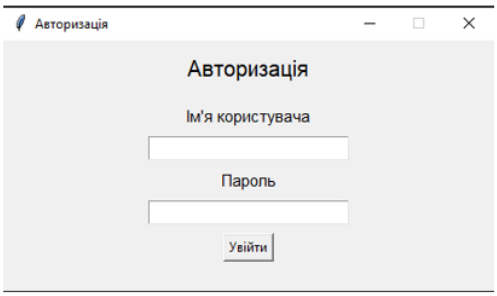
Тестування системи

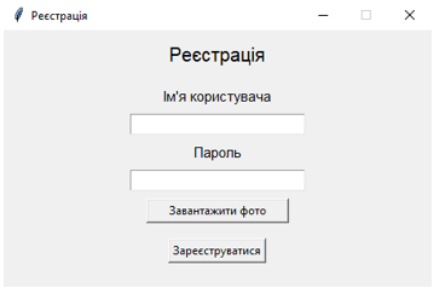
Інтерфейс вибору першої дії



Тестування системи

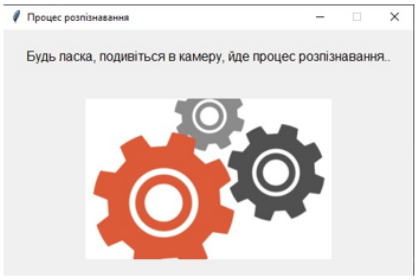
Інтерфейси авторизації та реєстрації

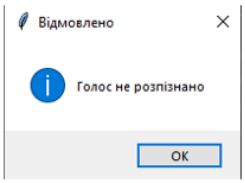




Тестування системи

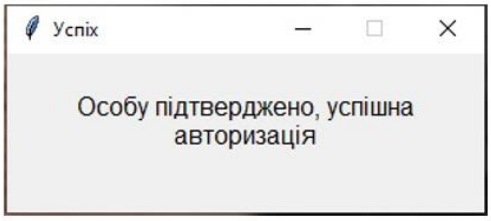
Приклад вдалого та невдалого проходження першого фактору авторизації за допомогою розпізнавання голосу

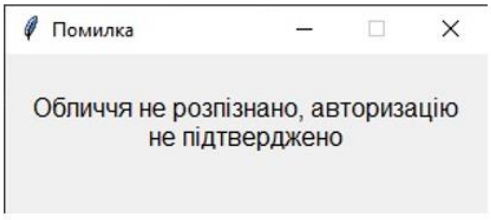




Тестування системи

Приклад вдало та невдало пройденої авторизації





Висновки

- *У результаті виконання дипломної роботи було проведено глибокий аналіз методів автентифікації користувачів за допомогою біометричних даних, зокрема розпізнавання обличчя та голосу. Було розглянуто актуальність використання таких методів у сучасних умовах кібербезпеки, їх переваги та недоліки, а також проаналізовано існуючі системи-аналоги.*
- *У ході роботи було обрано метод розпізнавання обличчя за допомогою власних векторів (Eigenfaces) як найбільш оптимальний для реалізації поставлених задач. Проведено детальний аналіз цього методу та продемонстровано його ефективність у порівнянні з іншими методами розпізнавання. Також було розглянуто та впроваджено додатковий механізм автентифікації за допомогою розпізнавання голосу. Для цього було використано бібліотеки speech_recognition та librosa, які дозволяють ефективно обробляти аудіо дані та здійснювати їх порівняння. Було реалізовано функції запису голосу, витягнення аудіо характеристик та їх порівняння, що забезпечило додатковий рівень безпеки при автентифікації користувачів.*
- *Розроблений програмний продукт було реалізовано на мові програмування Python, що дозволило скористатися багатим набором бібліотек для роботи з обробкою зображень та аудіо. Для створення графічного інтерфейсу було використано бібліотеку Tkinter, яка забезпечує простоту та зручність у використанні.*
- *Загалом, результати виконаної роботи демонструють ефективність використання біометричних даних для автентифікації користувачів та підтверджують доцільність впровадження таких методів у сучасних системах захисту інформації. Розроблений програмний продукт має широкий спектр застосування та може бути використаний у різних сферах для підвищення рівня безпеки та надійності систем автентифікації.*

ДЯКУЮ ЗА УВАГУ!

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ
ЗАПОЗИЧЕНЬ

Назва роботи: Розробка програми комплексної автентифікації до системи захисту з використанням ідентифікації за допомогою зліпка обличчя та голосу

Тип роботи: бакалаврська дипломна робота
(БДР, МКР)

Підрозділ: Кафедра менеджменту та безпеки інформаційних систем
Факультет менеджменту та інформаційної безпеки
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 70 % Схожість 30 %

Аналіз звіту подібності (відмітити потрібне):

1. Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

2. Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

3. Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Коваль Н.П.
(підпис) (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Геник О.І.
(підпис) (прізвище, ініціали)

Керівник роботи _____ Грицак А.В.
(підпис) (прізвище, ініціали)