Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

Навчально-науковий інститут деревообробних та
комп'ютерних технологій і дизайну
(повне найменування інституту, назва факультету (відділення))

Кафедра інформаційних технологій
(повна назва кафедри (предметної, циклової комісії))

# Пояснювальна записка
до дипломної роботи

другий (магістерський)
(рівень вищої освіти)

на тему: Інформаційна технологія оцінки процесу поширення інфекційних збудників у міській забудові

Виконав: студент 6 курсу групи КН-61м
спеціальності
122 "Комп'ютерні науки"
(шифр і назва напряму підготовки, спеціальності)

Терешків С.М.
(прізвище та ініціали)

Керівник ШабатураЮ.В.
(прізвище та ініціали)

Рецензент
(прізвище та ініціали)

Львів – 2021

Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

ННІ деревообробних та комп'ютерних технологій і дизайну
Кафедра інформаційних технологій

Рівень вищої освіти другий (магістерський)
Спеціальність 122 "Комп'ютерні науки"
(шифр і назва)

**ЗАТВЕРДЖУЮ**
**В.о. завідувача кафедри**

_____Крошний І. М.
"____"_____20___року

# З А В Д А Н Н Я
## НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

**Терешків Сергій Миколайович**
(прізвище, ім'я, по батькові)

1. Тема роботи *Інформаційна технологія оцінки процесу поширення інфекційних збудників у міській забудові*
**керівник роботи** ***Шабатура Юрій Васильович, професор кафедри ІТ доктор технічних наук, професор*** **,**
( прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від "_30_" грудня 2020 року № С-593_____
2. Термін подання студентом роботи "10" грудня 2021 р.
3. Вихідні дані до роботи Постановка задачі та її формалізація
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

СТАН ПРЕДМЕТНОЇ ОБЛАСТІ
ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ
Короткий огляд задач на виявлення та розпізнавання облич
Сфери застосування розпізнавання облич
Інформаційне забезпечення оцінки
Реалізація збору даних для системи оцінки процесу поширення інфекційних збудників
Схема бази даних
МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ
Аналіз і вибір методів, алгоритмів та засобів розв'язання задачі
Обчислення оцінки процесу поширення інфекційних збудників
ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ
Загальна структура програмного коду
Опис використаних сторонніх бібліотек та модулів

Інструкція щодо проектування бази даних

Розробка та опис інтерфейсу користувач

РОЗРОБЛЕННЯ СТАРТАП ПРОЕКТУ

Опис ідеї проекту

Організаційний та фінансовий плани

Ризики проекту

Висновки до розділу

5. Дата видачі завдання “18” грудня 2020 року.

# КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів дипломної роботи | Строк виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Вивчення стану проблемної області для поставленої задачі | 18.12.2020 – 18.01.2021 | виконано |
| 2 | Огляд літературних та інших джерел згідно досліджуваної теми | 19.01.2021-20.02.2021 | виконано |
| 3 | Постановка задачі та її формалізація | 21.02.2020-12.03.2021 | виконано |
| 4 | Вибір та обґрунтування методів і алгоритмів | 13.03.2020-31.03.2020 | виконано |
| 5 | Аналіз наявних машинного навчання методів та їх переваг | 01.04.2020-24.09.2021 | виконано |
| 6 | Програмна реалізація завдання | 25.09.2021-26.10.2021 | виконано |
| 7 | Оцінка похибки отриманих результатів | 27.10.2021-30.11.2021 | виконано |
| 8 | Оформлення пояснювальної записки до дипломного проекту | 01.12.2021-09.12.2021 | виконано |
| 9 | Подання пояснювальної записки на кафедру інформаційних технологій НЛТУ України | 10.12.2021 | виконано |

**Студент** _____ **Терешків С.М.**
( підпис ) (прізвище та ініціали)

**Керівник роботи** _____ **Шабатура Ю.В.**
( підпис ) (прізвище та ініціали)

## РЕФЕРАТ

Магістерська дипломна робота: пояснювальна записка: 73 стор., 18 рис., 8 додаток, 19 джерел.

В даній роботі розроблено програмне та алгоритмічне забезпечення системи оцінки поширення інфекційних збудників серед відвідувачів підприємства. Програмна система, що створена на базі мови програмування Python, може використовуватися як керівниками закладу чи підприємства так і лікарями чи медпрацівниками певного підприємства.

Ключові слова: інфекція, комп’ютерне бачення, Python, OpenCv.

## ABSTRACT

Diploma paper contains 73 pages of explanatory note, 18 pictures, 8 application, 19 used literary sources.

In this work, the algorithmic and software for movie rating prediction system is developed using machine learning. The system that is based on the Python programming language, can be used both by people who plan to make a film to assess its popularity, and by viewers who plan to watch a future film.

In this work, software and algorithmic software for the evaluating of common infections spreading among visitors or workers. The software system, which is based on Python programming language, can be used both by the heads of the institution and by doctors or nurses of a certain factory.

Keywords: Infections, computer vision, Python, OpenCv.

# ТЕХНІЧНЕ ЗАВДАННЯ

Необхідно розробити програмне та алгоритмічне забезпечення системи оцінювання процесу поширення інфекційних збудників у міській забудові, а саме:

- реалізувати засоби збору інформації про відвідувачів, працівників, лікарів та приміщення у яких вони працюють;
- провести попередній аналіз та обробку даних;
- вибрати алгоритми комп’ютерного бачення та оцінки поширення інфекційних збудників;
- реалізувати програмне рішення;
- провести тестові запуски поведінки розробленої системи;
- описати результати тестування.

# АНОТАЦІЯ

Метою виконання даної роботи є зменшення ризиків швидкого поширення інфекційних захворювань за рахунок застосування розроблюваної інформаційної технології, яка дозволяє в автоматичному режимі визначати коло і тривалість контактів інфікованої особи з іншими завдяки їх ідентифікації на основі розпізнавання обличчя, що дозволяє своєчасно застосовувати ефективні медичні заходи. Об’єктом дослідження в даній роботі є процес поширення збудників інфекційних захворювань у міській забудові.

Предметом дослідження є інформаційні системи для надання оцінки процесу поширення інфекційних збудників у міській забудові, приміщенні тощо, а також технології розпізнавання та ідентифікації облич.

# ЗМІСТ

## ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

| | |
|---|---|
| LBPH | – Local Binary Patterns Histogram |
| БД | – База даних |
| ОС | – Операційна система |
| ПЗ | – Програмне забезпечення |
| KFC | – Kentucky Fastfood Center |
| SNoW | – Sparse Network of Winnows |
| SMQT | – Successive Mean Quantization Transform |
| НМ | – Нейронна мережа |
| БНМ | – Багатошарова нейронна мережа |
| AAM | – Active Appearance Models |
| ASM | – Active Shape Models |
| BSD | – Berkeley Software Distribution |
| ПЗП | – Постійний запам'ятовувальний пристрій |
| SVG | – Scalable Vector Graphics |
| GUI | – Graphical User Interface |

# ВСТУП

Інфекційні захворювання існують вже багато років, навіть можна сказати з часів зародження людства. І на благо у нинішньому столітті є безліч праць та знань, які сповіщають людство про способи передачі, важкість протікання захворювання та профілактику захворювань. Потрібно лише вміло користуватися цією інформацією. Інформаційні технології теж не стоять на місці і щодня вдосконалюються, спрощують людям життя. За останні десятиліття вони активно інтегруються і в медицину.

**Актуальність проблеми**. Контроль здоров'я соціальних груп: студентів, учнів, працівників, тощо – має надзвичайно важливе значення у сучасному суспільстві. Автоматизація цього процесу надає неабиякі переваги у веденні господарської, соціальної, фінансової та інших діяльностей. Відсутність систем контролю здоров'я ускладнює або й унеможливлює роботу великих компаній-гігантів та підприємств. Потреба в таких системах виникла особливо зараз у нашому повсякденному житті. З процесом поширення спалаху коронавірусу, що перетворився у епідемію стало надзвичайно важливо контролювати процес передачі вірусу починаючи від маленьких спільнот і закінчуючи цілими континентами. Здійснити такий контроль можна на будь-якому соціальному рівні, використовуючи технології автоматизації систем доступу, за допомогою, наприклад, обличчя.

**Об'єктом дослідження** в даній роботі є процес поширення збудників інфекційних захворювань у міській забудові.

**Предметом дослідження** є інформаційні системи для надання оцінки процесу поширення інфекційних збудників у міській забудові, приміщенні тощо, а також технології розпізнавання та ідентифікації облич.

**Метою роботи** є зменшення ризиків швидкого поширення інфекційних захворювань за рахунок застосування розроблюваної

інформаційної технології, яка дозволяє в автоматичному режимі визначати коло і тривалість контактів інфікованої особи з іншими завдяки їх ідентифікації на основі розпізнавання облич, що дозволить своєчасно застосовувати ефективні медичні заходи.

Розроблений застосунок дасть змогу фіксувати входження людей у приміщення, заносити і зберігати дані про осіб, які перебували в певному приміщенні, отримувати необхідні довідкові дані тощо.

**Основні завдання**, що мають бути розроблені у роботі:

1. Охарактеризувати об’єкт дослідження і сформулювати задачі, що мають бути розроблені;
2. Провести огляд літературних джерел щодо алгоритмів комп’ютерного зору;
3. Дослідити алгоритми розпізнавання облич та ідентифікації осіб, вибрати оптимальні для програмної реалізації;
4. Спроектувати структуру бази даних системи;
5. Розробити програмний продукт, що буде реалізовувати поставлені задачі, використавши відповідний інструментарій та зовнішні рішення і бібліотеки;
6. Провести тестування розробленого програмного забезпечення з метою перевірки працездатності системи реєстрації;
7. Розробити модуль, який буде надавати оцінку процесу поширення інфекцій.

**Новизна роботи.** ***Новизна роботи*** полягає в тому, що запропонована інформаційна технологія, яка дозволила реалізувати єдиний програмний продукт, який автоматично ідентифікує та слідкує за переміщенням людей по приміщеннях забудови, підприємства, громадського транспортного засобу, тощо, та відповідно до певних запитів дозволяє миттєво визначати

коло осіб, які могли бути інфіковані в результаті контактів з носієм інфекційного збудника.

**Практичне значення одержаних результатів.**

Розроблено програмний продукт для автоматичної фіксації полягатиме у можливості використовувати розроблену систему для вирішення проблеми автоматичної ідентифікації осіб та, в подальшому, для надання їм доступу до певних робіт (приміщень), що високоефективно у запобіганню процесів поширення інфекцій.

# РОЗДІЛ 1. СТАН ПРОБЛЕМНОЇ ОБЛАСТІ

## 1.1. Актуальність теми дослідження

Хоча людство сильно просунулося в лікуванні, виявленні та профілактиці захворювань, нам від них не позбутися раз і назавжди. Основна задача зараз стоїть у проведенні досліджень та використання результатів задля мінімізації поширень. Допомагають нам у цьому питанні інформаційні технології.

Дуже добре видно на прикладі інфекційного захворювання, яке з'явилося у 2019 році – Covid-19. З його появою актуальність теми надзвичайно зросла, адже стало необхідно якось контролювати рух інфекційних збудників та сприяти зниженню їх поширення.

Та як хвороба зупинила звичайний людський процес життя, тому людство буде інтенсивно боротися з нею та наступними вірусними викликами, в чому їм допомагатимуть програмні засоби у тій чи іншій області від контролю та статистики поширення інфекційних збудників до сервісів, які даватимуть приблизний прогноз поширення і рекомендації як його знизити.

Інформаційні технології застосовують нині майже у всіх сферах життя суспільства. За останні десятиліття вони активно інтегруються і в медицину.

З кожним роком інформаційні технології все активніше впроваджуються у всі сфери діяльності людини, зокрема й у галузь охорони здоров'я. У Європейському Союзі, до прикладу, протягом останніх 15 років близько 500 мільйонів євро було спрямовано на наукові дослідження у сфері медичної інформатики.

Медичні інформаційні технології — це сукупність методів та засобів, що дають змогу обробляти медичні дані у цілісних технологічних системах

для створення, використання, зберігання, передавання і захисту інформаційного продукту. Застосування медичних інформаційних технологій відбувається при розв'язанні поставлених завдань у полі медичних інформаційних систем.

Медична інформаційна система — це інструмент, який дає змогу визначати і планувати всі ресурси закладу охорони здоров'я шляхом застосування спеціалізованого програмного забезпечення, засобів обчислювальної техніки, необхідного медичного обладнання, засобів зв'язку, і підтримує лікувально-діагностичну, фінансову, адміністративно-господарську, облікову та сервісну діяльність установи для надання якісних медичних послуг пацієнтам.

Медична інформатика істотно удосконалює роботу системи охорони здоров'я, робить медицину доступною для населення, а процес надання медичних послуг — ефективнішим. Це можливо завдяки змінам в організаційних питаннях системи, які підвищують якість надання медичних послуг з одночасним зменшенням фінансових видатків на їх проведення. При цьому інформаційні технології сприяють удосконаленню роботи усіх складових галузі охорони здоров'я: дають змогу спростити облік пацієнтів, організувати та скоротити робочий час спеціалістів, вести автоматичний облік ліжкофонду, контролювати призначення препаратів, спростити ведення та отримання статистичних даних.

Світовий досвід розробки та застосування інформаційних технологій свідчить про необмежений потенціал цієї галузі щодо розв'язання соціальних завдань суспільства.

У директиві А58/21 Всесвітньої організації охорони здоров'я йдеться про те, що охорона здоров'я в онлайні «…відкриває унікальну можливість для розвитку суспільної охорони здоров'я. Зміцнення охорони здоров'я за допомогою системи електронної охорони здоров'я може сприяти

здійсненню основних прав людини в результаті підвищення рівня справедливості, солідарності, якості життя і якості медико санітарної допомоги».

Так, США вже здійснює масштабну реформу системи охорони здоров’я, спрямовану на застосування електронних медичних карт і впровадження телемедицини.

### 1.2. Аналіз предметної області

Інфекційні хвороби - розлади здоров'я людей, що спричинені збудниками (вірусами, бактеріями, рикетсіями, найпростішими, грибками, гельмінтами, кліщами, іншими патогенними паразитами), продуктами їх життєдіяльності (токсинами), патогенними білками (пріонами), які передаються від заражених осіб здоровим і здатні до масового поширення.

Небезпечні інфекційні хвороби ‑ інфекційні хвороби, що характеризуються тяжкими та (або) стійкими розладами здоров'я в окремих хворих і становлять небезпеку для їх життя та здоров'я.

Небезпечні інфекційні хвороби ‑ інфекційні хвороби, що характеризуються тяжкими та (або) стійкими розладами здоров'я в окремих хворих і становлять небезпеку для їх життя та здоров'я.

Особливо небезпечні інфекційні хвороби ‑ інфекційні хвороби (у тому числі карантинні: чума, холера, жовта лихоманка), що характеризуються тяжкими та (або) стійкими розладами здоров'я у значної кількості хворих, високим рівнем смертності, швидким поширенням цих хвороб серед населення.

Джерело інфекційної хвороби ‑ людина або тварина, заражені збудниками інфекційної хвороби.

Контактні особи - люди, які перебували в контакті з джерелом інфекції, внаслідок чого вони вважаються зараженими інфекційною хворобою.

Чинники передачі збудників інфекційних хвороб - забруднені збудниками інфекційних хвороб об'єкти середовища життєдіяльності людини (повітря, ґрунт, вода, харчові продукти, продовольча сировина, кров та інші біологічні препарати, медичні інструменти, предмети побуту тощо), а також заражені збудниками інфекційних хвороб живі організми, за участю яких відбувається перенесення збудників інфекційних хвороб від джерела інфекції до інших осіб.

**Шляхи передавання**

Інфекційні заразні хвороби займають особливе місце серед інших захворювань людини. Найважливішою особливістю інфекційних хвороб є їх заразливість, тобто можливість передачі від хворої людини або тварини здоровій.

Багато з цих хвороб, наприклад чума, холера, грип, здатні до масового (епідеміологічного) розповсюдження, що охоплює при наявності відповідних умов цілком село, місто, область, країни і континенти.

Інфекційні хвороби викликаються мікробами (мікроорганізмами). Мікроорганізми перебувають у великій кількості в ґрунті, воді, повітрі. Численні види мікробів населяють кишечник людини і тварин, що мешкають на шкірі і в ротовій порожнині. З допомогою деяких мікробів отримують багато продукти і лікувальні засоби. На ряду з корисними існують і шкідливі мікроорганізми. Деякі з них є збудниками заразних (інфекційних) захворювань людини. Ці мікроорганізми є патогенними.

Безпосередньою причиною виникнення хвороби служить впровадження в організм людини патогенного мікроорганізму або отруєння токсином. Збудники інфекційних хвороб передаються від хворих здоровим різним шляхом. Поширення інфекційних хвороб в людському колективі отримало назву епідемічного процесу. У процесі поширення інфекційних

хвороб розрізняють три ланки: 1) джерело збудника інфекцій; 2) механізм передачі; 3) сприйнятливість населення.

Є такі види механізмів передачі інфекційних збудників:

- фекально-оральний — коли патогени, які містяться у випорожненнях джерела інфекції, потрапляють до травної системи здорової людини чи тварини (сприйнятливий контингент) через рот.
- трансмісивний (кров'яний) — передача інфекції від джерела відбувається через укус комах (блохи, комарі, кліщі тощо) або втирання їхніх решток чи фекалій (воші).
- повітряно-крапельний (респіраторний, дихальний) — передача відбувається від джерела інфекції через активне виділення збудника з секретами респіраторної системи.
- контактний (включає статевий шлях) — передача інфекції відбувається від різних джерел через макро- і мікро пошкодження шкіри або слизових, зокрема через укус тварин, травматизацію тощо.
- вертикальний — специфічна передача, яка відбувається від матері до плода впродовж вагітності, під час пологів та першого тижня від народження дитини.
- артифіціальний — рідкий механізм передачі, якій відбувається за допомогою технічних засобів (зокрема, як при легіонельозі, коли зараження відбувається через кондиціонери). На сьогодні цей механізм не визнаний усіма науковцями.
- гемоконтактний — механізм передачі, при якому відбувається передача від людини — джерела інфекції через застосування медичних (переливання препаратів крові, ін'єкції, оперативні втручання тощо) або немедичних (внутрішньовенне введення наркотичних речовин, татуаж тощо) маніпуляцій, при яких

відбувається пошкодження шкіри або / та слизових оболонок і потрапляння частинок крові у кровоток сприятливої здорової людини.

Звернемо увагу на повітряно-крапельний спосіб передачі інфекційних збудників, адже зараз він найбільш актуальний і вражає найбільше, так як багато гострих вірусних інфекцій передаються в основному саме через дихальний шлях. Не є винятком і коронавірусна хвороба, в якої переважає контактний шлях, коли вірус потрапляє на слизові оболонки після торкання до об'єктів, що контаміновані виділеннями із дихальним шляхів хворого чи інфікованого. Також краплинним шляхом передається від людини до людини під час чхання та кашлю. Або ж шлях повітряний, коли вірус передається від людини до людини через аерозольні частинки, які безперешкодно можуть потрапляти в альвеоли і бронхіоли. Тому надалі в роботі сконцентруємось на повітряно-крапельному шляху передачі вірусів.

**Колективний імунітет**

Існує період коли та чи інша інфекція перестає нести загрозу для людства в цілому або відокремлених соціальних груп. Для цього існує термін Колективний імунітет - це явище несприйнятливості окремої спільноти до інфекційних захворювань, а також метод захисту від інфекційних хвороб, що полягає в порушенні доступу до джерел інфекції та можливих механізмів і шляхів передачі інфекції, якщо великий відсоток спільноти отримали індивідуальний імунітет до інфекції, та забезпечує ефективно захист особам, які ще не захищені.

Чим більше осіб у спільноті, що мають індивідуальний імунітет, тим менша ймовірність того, що незахищені вступатимуть в контакт з носієм хвороби.

Індивідуальний імунітет особи отримують шляхом природного відновлення після інфекції або через штучні методи, такі як вакцинація.

Деякі особи не можуть бути вакциновані з медичних показань чи взагалі набути імунітет, тому в спільнотах колективний імунітет є важливим та ефективним методом захисту від інфекційних хвороб.

Якщо $R_0$ — так званий коефіцієнт відтворення інфекційної хвороби (кількість ефективних інфікувань осіб в середньому на одну особу-носія інфекції), то критичний рівень колективного імунітету обчислюємо як

$$q_c = 1 - \frac{1}{R_0} \tag{1.1}$$

Таким чином $q_c$ дає відсоток осіб з імунітетом, тобто з повною несприйнятливістю до певної хвороби, що необхідний, щоб зупинити поширення певної інфекційної хвороби.

У реальних умовах, вакцинація не дає в ефекті імунітет на 100%, тому з допомогою коефіцієнта ефективності вакцини E, тобто відсотка осіб, що отримали вакцину та набули імунітет, отримуємо необхідний відсоток осіб $V_c$, які повинні бути вакцинованими, щоб досягти колективного імунітету в певній спільноті та зупинити поширення хвороби.

$$V_C = \frac{q_c}{E} \tag{1.2}$$

Тобто $V_C$ показує скільки відсотків потрібно вакцинувати або заразити, щоб зупинити поширення інфекції.

## 1.3. Огляд існуючих методів і засобів оцінювання поширення інфекційних збудників.

**Огляд існуючих методів.**

Прогнозування захворюваності належить до 4 групи епідеміологічних методик і при цьому використовують такі підходи: статистичні методи чи математичні формули або ж комбінований підхід.

Математичне моделювання епідемічного процесу інфекцій – один з найкращих засобів отримати першочергову інформацію про швидкість розповсюдження інфекційних хвороб та ширину розповсюдження.

Розглянемо існуючі моделі розповсюдження захворюваності.

Принципи, які покладені в основу сучасного епідемічного моделювання – засновані на диф рівняннях, і були визначені В. Хамером в 1906 році.

Нехай x(t) – число людей з популяції, яка приймає участь в моделюванні. Ці люди вважаються здоровими. І y(t) – це число вже хворих індивідів. В такому випадку зміну кількості інфікованих персонажів за час t можна описати так:

$$\frac{dy}{dt} = \beta xy \tag{1.3}$$

де β – параметр, що визначає інтенсивність передачі інфекції.

Сучасні моделі з базою диференційних рівнянь є моделі SIR і SEIR. Перша з них була винайдена В. Кермаком і А. МакКендриком. Суть цієї моделі полягає в тому що уся популяція поділяється на три категорії: «Сприйнятливий» (від англ. Susceptible) – вразливий до PBI, який може захворіти, але зараз є здоровий; «Інфікований» (від англ.Infectious) – люди, які є хворими і в даний час є епідемічно значущими; «Реконвалесценти» (від англ. «Recovered») – люди, які вже одужали і мають тимчасовий імунний захист і більше до нього не вразливі, або померли. Тож загальне число людей залишається незмінним, а тоді приріст числа людей в кожній із виділених груп можна описати такою системою рівнянь:

$$\frac{dS}{dt} = -\beta SI, \frac{dI}{dt} = \beta SI - \gamma I, \frac{dR}{dt} = \gamma I \tag{1.4}$$

,де $\beta$ – коефіцієнт контакту між людьми, а γ - інтенсивність переходу індивідів у стан R.

Є ще трішки просунута модель епідеміологічного моделювання. Для прикладу наслідок моделі SIR – модель SEIR.

У ній до перелічених вище груп індивідів, що моделюються в моделі SIR, додається ще одна: «Незахищений» (з англ. Exposed) – індивіди, інфекція в яких ще на стадії інкубаційного періоду. Тоді система рівнянь, що описує приріст числа хворих індивідів, буде

$$\frac{dS}{dt} = B - \beta SI - \mu S, \frac{dE}{dt} = \beta SI - (\varepsilon - \mu)E, \frac{dI}{dt} = \varepsilon E - (\gamma + \mu)I, \frac{dR}{dt} = \gamma I - \mu R \quad (1.5)$$

, де B – середній рівень народжуваності індивідів на території, що моделюється, μ – середній рівень смертності індивідів на території, що моделюється, $1/\varepsilon$ – середня тривалість інкубаційного періоду захворювання.

Перебіг поширення інфекції залежить від такого співвідношення:

$$R_0 = \frac{\beta}{\gamma} \quad (1.6)$$

Це значення є коефіцієнт поширення інфекції.

Як правило, епідемія поширюється серед сприятливої популяції, коли $R_0$ більше за одиницю, тобто $R_0 > 1$, і, таким чином, кількість випадків зараження буде зростати. Рідко все населення може бути сприйнятливим до інфікування у фізичному світі. Деякі інтромісії будуть імунними, можливо, через попередню інфекцію, яка виробила довічний імунітет, або, можливо, через попередню імунізацію. Таким чином, вся популяція не буде інфікована, а середнє число вторинних випадків на одну інфекційну людину буде меншим за $R_0$ , і це вимірюється через ефективний коефіцієнт відтворення, позначений R. $R_0$ залежить від захворювання та популяції хазяїна і різний для різних інфекційних захворювань, наприклад, $R_0$= 2,6 для туберкульозу великої рогатої худоби, $R_0$ = [3 4] для грипу у людей. $R_0$= [3,5 6] для віспи у людей і $R_0$ = [16 18] кору у людей.

Ефективне репродуктивне число - середня кількість вторинних інфекцій на інфекційний випадок у популяції, що розглядається, для чутливих і нечутливих носіїв. Кількість інфекційних випадків збільшується при R>1, тоді і починається епідемія, якщо R = 1, то хвороба вважається ендемічною. Це означає, що, в середньому, кожна інфікована особа заражає рівно одну особину (більше того, кількість інфікованих людей буде зростати експоненційно, що спричинило б епідемію, проте з часом хвороба спадатиме) а при R<1 кількість інфікованих буде зменшуватися. R можна оцінити за добутком $R_0$ і фракції чутливих ($x$) популяції хазяїна. Таким чином, R визначається як

$$R = R_0 x \tag{1.7}$$

Або ж це можна інтерпретувати так

$$R = R_0 cTt \tag{1.8}$$

де $c$ - це рівень контакту між сприйнятливими та інфікованими особами

$T$ – трансмісивність, тобто ймовірність зараження при контакті.

**Огляд існуючих засобів оцінювання поширення інфекційних збудників.**

На даний момент існує кілька застосунків для оцінки та контролю поширення інфекцій. Розглянемо додатки на прикладі поширення Covid-19, адже це супер актуально зараз і весь світ зосереджений в основному на цей напрям.

За даними Всесвітньої організації охорони здоров’я, діагностичне дослідження на COVID-19 є критичним для відстеження вірусу, розуміння епідеміології, інформування про ведення справ та придушення передачі.

У цій пандемії використовуються цифрові технології, такі як програми для смартфонів (програми) Технологія Bluetooth необхідна для відстеження заражених людей у прилеглих районах (ЮНЕСКО, 2020, en.unesco.org). Такі програми розробляються у всьому світі США, Сінгапуром, Індією,

Великобританією та США багатьох інших країн для відстеження та контролю коронавірусної хвороби.

Ці програми допомагають користувачам провести карантин правильно та іншу інформацію про безпеку. Автори стверджують, що якщо більша кількість людей завантажувала та використовувала подібні програми, це є потенціал для зменшення рівня поширення коронавірусної хвороби.

Отже перший додаток – це TraceTogether. Це популярна програма відстеження контактів для смартфонів, яка використовує Bluetooth для відстеження інфікованих людей та повідомляє людей, які були поруч із ними протягом останніх 15 днів. Будь-хто з мобільним номером Сінгапуру та смартфоном із підтримкою Bluetooth може завантажити цей додаток.

Aarogya Setu застосунок здійснює моніторинг через Bluetooth та графік, що генерується за місцем розташування, який записує близькість до будь-якої інфікованої людини. Ця програма була розроблена Міністерством Електроніки Індії і повідомляє користувачів, якщо вони перетиналися з кимось, хто мав діагноз - позитивний для Covid-19. Індія також використовує різні інші програми, такі як "Керала", "Карантин відстеження", "Більше, ніж просто відстеження" відстеження Covid-19. Ці програми взаємодіють між собою і можуть бути корисно для жителів усієї країни, які використовують ці програми.

COVID symptom tracker. Ця програма постійно відстежує новітні дослідження про симптоми вірусу, а також відстежує вірус серед тих, хто використовує додаток. Додаток відповідає загальне регулювання захисту даних, і дані використовуються лише для досліджень здоров'я, а не для комерційних цілей.

The corona dataSpende. Це німецька програма для розумних годинників, яка відстежує поширення коронавірусу шляхом збору життєво важливих ознак (частота серцевих скорочень, режим сну, температура тіла) у

добровольців, які використовують смарт-годинник або трекер фізичної. Ця програма може перевірити, чи у людини розвинулися симптоми Covid-19 чи ні. Результати відображаються на інтерактивній онлайн-карті, що дозволяє органам охорони здоров'я підвести підсумки ситуації та визначити гарячі точки.

NHS smartphone app. Додаток буде підтримувати контроль за рухами людей та попередити їх, якщо вони контактують із зараженими.

Додаток класифікує дані користувачів на основі демографічних даних, побутових структур та моделей мобільності. На основі аналізу даних може бути встановлена максимальна кількість людей, яким дозволено вільно пересуватися.

Kwarantana Dommowa. Польща була однією з перших західних країн, яка запустила програму для смартфонів, яка збирає велику кількість особистої інформації, включаючи місцезнаходження людей та цифрові фотографії. Під час використання цього додатка люди завантажують свої селфі-зображення за запитом з своїм точним місцезнаходженням. Цей додаток є обов'язковий для тих, хто отримує симптоми коронавірусу в Польщі.

Дій вдома. Застосунок для запобіганню поширенню COVID-19 в Україні. Функціонал застосунку дозволяє зареєструвати в місці проходження самоізоляції або обсервації, зробити фото підтвердження перебування за місцем проходження самоізоляція, зробити екстрений виклик на гарячу лінію МОЗ України.

### 1.4. Постановка завдання дослідження.

Завданням магістерської роботи є реалізувати систему для відеофіксації присутності зареєстрованих осіб і надання інтерфейсу для

зазначення, що особа є інфікованою, та для отримання кола осіб з якими, ця людина могла мати контакт.

Практичну реалізацію поставленого завдання можна розділити на такі основні етапи:

• проектування концептуальної моделі системи та визначення функцій її компонентів;

• розробка бізнес-логіки користувацького застосунку, що передбачає правильне реагування на дії користувачів;

• розробка користувацького інтерфейсу для зручної взаємодіяти із застосунком;

• проектування та розробка бази даних;

• інтеграція зовнішніх сервісів, які дають змогу зменшити час і витрати на розробку, а також дають можливість застосувати сучасні рішення й технології у процесі створення продукту.

До функціональних вимог розроблюваного продукту належать:

• забезпечити безпомилкову фіксацію та розпізнавання облич з відеопотоку;

• надати можливість занесення потрібної інформації про осіб у базу даних;

• забезпечити збереження всіх даних, в тому числі зафіксованих облич із вказанням дати, часу, приміщення тощо;

• надати зручну можливість створювати запити, щоб отримати потрібну інформації про зафіксованих осіб;

• відображати у зручній формі результати запиту.

Вимоги до процесу:

• використання простого графічного інтерфейсу.

Зовнішні вимоги:

• операційна система: Windows;

• веб-камера з чіткою роздільною здатністю (1280x720) ;

• добре освітлене приміщення.

# РОЗДІЛ 2. ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

Сучасне суспільство дійшло до етапу, коли раптове зникнення технологій означатиме крах цивілізації та нормального забезпечення життєдіяльності. Сучасні технології наскільки сильно вплелись в наше повсякденне життя, що ми не можемо представити життя без них.

## 2.1. Короткий огляд задач на виявлення та розпізнавання облич

Розпізнавання образів – виділення істотних ознак та їх віднесення до певного класу, що характеризують даний образ із загальної маси даних.

Розпізнавання облич – один із підрозділів більш широкої категорії розпізнавання образів. Алгоритми та методи розпізнавання мають схожість, проте відмінні за функціями розпізнавання, а точніше за їх параметрами.

Список задач, які вирішують розпізнавання образів значно зріс із підвищенням обчислювальних потужностей та вдосконаленням алгоритмів розпізнавання. До нього належать:

- комп'ютерні ігри та віртуальна реальність;
- персоналізація побутових пристроїв;
- електронна комерція;
- шифрування даних;
- взаємодія комп'ютер-людина;
- доступ до інформаційних баз;
- соціальні сервіси.

На сьогодні вже інтегровано алгоритми автоматичного розпізнавання облич на фотографіях в популярних соціальних сервісах Facebook та Google Picassa.

## 2.2. Сфери застосування розпізнавання облич

Щоб повноцінно зрозуміти та оцінити усі можливості даної технології розглянемо програмні проекти, які можуть виконувати різноманітні завдання, базуючись на системі розпізнавання облич.

**Face Unlock у смартфонах**

Face ID — це зчитувач об’ємної форми особи, який створений компанією Apple. Встановлено в смартфонах iPhone X, Iphone XR,Iphone XS,XS Max. Дана технологія заміняє Touch ID. Дозволяє розблокувати пристрій, здійснювати покупки в магазинах, онлайн-платформах.

Користувач реєструє своє обличчя. За принципом це робиться доволі просто: користувач дивиться у телефон і робить плавні колові рухи голови, щоб зареєструвати лице. На цьому реєстрація закінчується, а смартфон готовий для розблокування та подальшого використовування. Це рятує від небажаної активності на пристрої для прикладу якщо хтось інший взяв у руки телефон.

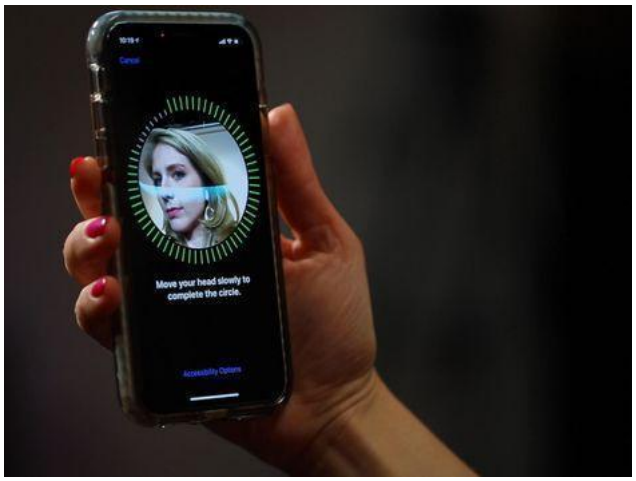


Рис. 2.1. Приклади роботи реєстрації обличчя у смартфонах Apple

**Розпізнавання облич у суміжних сферах**

У Китаї широко використовуються спеціальні сонячні окуляри з функцією розпізнавання облич для того, щоби поліцейські спостерігали за правопорядком у багатолюдних місцях таких як: метро, вокзали, аеропорти.

Прилад може ідентифікувати людину за 100 мілісекунд і вже не раз допомагав службам безпеки спіймати злочинців.

Також широке застосування мають камери фіксації порушень пішоходами. Ці камери встановлені на складних ділянках вулиць і слідкують аби люди не перебігали дорогу на заборонений сигнал світлофору. Для визначення особи порушника використовують систему розпізнавання облич.

На вступних екзаменах у деяких країнах використовуються розпізнаванн обличь і відбитків пальців щоб гарантувати, що абітурієнти є справжніми вступниками.

Після кількох зафіксованих випадків викрадення дітей деякі дитячі садки відкривають двері лише тим людям, чиї обличчя зареєстровані у системі.

У ресторанах KFC можна оплатити замовлення за допомогою сканера особи. Система може використовуватись зареєстрованими користувачами, вдосконалюючи покупки Alipay. Для того, щоб оплатити їжу клієнти дивляться у тримірну веб-камеру, яка робить фотознімок і відскановує його обличчя. Інформація, прив’язана до участі записів у Alipay, дозволяє перевірити користувача та вдосконалити покупку.

Китайський Airbnb встановлює в будинках «розумні» замки з розпізнаванням облич.

- В Кінці 2018 року Xiaozhu повідомив про початкове впровадження «розумних» замків із розпізнаванням облич у домах та квартирах, які пропонують орендувати з допомогою сервісу. Нова технологія, що дозволяє перевірити особу орендарів, збільшила свою потужність у значній мірі за рахунок вдосконалення охорони та безпеки на швидкодіючих ринках короткострокових оренд.

### 2.3. Інформаційне забезпечення оцінки

Щоб зрозуміти як поширюються віруси у міській забудові потрібно звернутися до декількох статей на цю тему та медичної термінології. На основі статті британського інтернет видання про можливість поширення вірусних інфекцій звичайною людиною та супер поширювачем із з фактором навколишнього середовища проаналізовано і визначено чинники, які впливають на швидкість та ширину поширення вірусів. Для того аби детальніше зрозуміти як це працює, потрібно ввести кілька понять.

Перше з них це - базове репродукційне число ($R_0$) - середня кількість осіб, що безпосередньо інфікуються хворим упродовж усього заразного періоду хвороби за умови потрапляння хворого до повністю здорової популяції. Користь цього показника полягає у змозі визначити можливість поширення інфекційної хвороби серед популяції. У нашому випадку популяцією може бути працівники чи відвідувачі міської забудови. $R_0$ – це безрозмірний параметр, що характеризує заразність інфекційного захворювання у медичній та ветеринарній епідеміології. Зазвичай визначається як кількість індивідуумів, які будуть заражені типовим хворим, що потрапили в повністю не імунізоване оточення за відсутності спеціальних епідеміологічних заходів, спрямованих на запобігання поширенню захворювання (наприклад, карантину). Якщо $R_0$>1то на початковому етапі кількість хворих буде зростати експоненційно.

Також потрібно підняти питання частоти контактів та інфекційний період. Припустимо, що інфіковані особини складають у середньому β інфекційні контакти в одиницю часу t, із середнім інфекційним періодом . Тоді основний номер відтворення:

$$R_0 = \beta * t \qquad (2.1)$$

Найчастіше використання $R_0$ - це визначення, чи може виникла інфекційна хвороба поширюватися в популяції, і визначити, яку частину

населення слід імунізувати шляхом вакцинації для ліквідації хвороби. У широко використовуваних моделях інфекції, коли $R_0 > 1$ інфекція може почати поширюватися в популяції, але якщо $R_0<1$, то навпаки. Як правило, чим більше значення $R_0$, тим важче контролювати епідемію.

Ця проста формула пропонує різні способи зменшення $R_0$ і, зрештою, поширення інфекції. За одиницю часу можна зменшити кількість контактів, що викликають інфекцію β шляхом зменшення кількості контактів за одиницю часу (наприклад, перебування вдома, якщо інфекція потребує контакту з іншими для поширення) або частки контактів, які викликають інфекцію (наприклад, носіння якогось захисного спорядження). Отже, цю формулу також можна записати так:

$$R_0 = \mathrm{c} * T * t \qquad (2.2)$$

де *c* - це рівень контакту між сприйнятливими та інфікованими особами

*T* – трансмісивність, тобто ймовірність зараження при контакті.

Тобто до с можна віднести такі показники як кількість людей в приміщенні, наскільки приміщення велике, яка провітрюваність у кімнаті і вологість та температура, а також носіння масок відвідувачами кімнати та тривалість знаходження у приміщенні, а до Т – хто з контактів вакцинований, вакциною з яким степенем захисту, чи контакт перехворів.

Ефективне репродуктивне число (R). Населення рідко буде повністю сприйнятливим до інфекції в реальному світі. Деякі контактні особи будуть несприйнятливі, наприклад, через попередню інфекцію, яка надала довічний імунітет, або в результаті попередньої імунізації. Таким чином, не всі контакти будуть інфіковані, і середня кількість вторинних випадків на один інфекційний випадок буде нижчою за основне число відтворення. Ефективне репродуктивне число (R) — це середня кількість вторинних випадків на інфекційний випадок у популяції, що складається як із чутливих, так і з нечутливих хазяїв. Якщо R>1, кількість випадків зросте,

наприклад, на початку епідемії. Якщо R=1, хвороба є ендемічною, а де R<1 буде зменшуватися кількість випадків.

Ефективне число розмноження можна оцінити за добутком основного репродуктивного числа та частки сприятливої популяції хазяїна (x). Так:

$$R = R_0 * x \quad (2.3)$$

Наприклад, якщо $R_0$ для грипу дорівнює 12 у популяції, де половина населення має імунітет, ефективне репродуктивне число для грипу становить 12 x 0,5 = 6. За цих обставин один випадок грипу спричинить у середньому 6 нових вторинних випадків.

### 2.4. Реалізація збору даних для системи оцінки процесу поширення інфекційних збудників

Для того аби інформаційна система була робоча та виконувала усі свої зобов’язання, потрібно аби вона оперувала з великою кількістю даних. Для цього на початковому етапі повинна існувати база даних для людей, тобто працівників міської забудови та база даних приміщень, доступ до яких контролюється системою реєстрації осіб на основі системи розпізнавання облич.

Система реєстрації осіб на основі розпізнавання облич буде існувати лише за початкового налаштування – внесення даних та фотографій про можливих відвідувачів забудови. За цей елемент відповідає адміністратор, який заповнює усю необхідну інформацію про суб’єкта реєстрації: прізвище, ім’я, далі, якщо це працівник, то бригаду, а якщо керівник чи відвідувач, то інші потрібні дані. Після того як усі дані введено, для завершення реєстрації система автоматично зробить фотографію особи, яка реєструється. Користувач заздалегідь повідомлений про те, що буде зроблена фотографія, щоб не виникало проблем у майбутньому з використанням особистих даних. Слід зазначити, що для правильної роботи

системи потрібно, щоб на фотографії було обличчя людини. Після цих дій, дані заносяться у відповідні таблиці бази даних, а для фотографій запускається процес тренування вибірки, завдяки чому зроблена фотографія класифікується. Ці функції повторюються, поки дані про усіх потрібних осіб не буде занесено у базу даних. Також для для початку потрібно зареєструвати приміщення, в яких можуть знаходитись робітники чи відвідувачі. Для цього адміністратор вказує номери приміщення, координати, площу загальну, кількість вікон. Крім цього в приміщенні повинні знаходитись датчики температури та вологості, які будуть посилати свої дані у таблиці, щоб підтримувати актуальну інформацію про приміщення у базі даних.

Безпосередньо під час роботи система повинна знати час входу і виходу кожного відвідувача у певне приміщення. Для цього буде створено додатково таблиця в базі даних, яка містить дату і час входження ідентифікованого персонажа у приміщення.

Також мусить бути окрема таблиця стану здоров’я працівників на певний момент часу. Отже до цієї таблиці буде мати відношення таблиця з відомими інфекційними хворобами та їхніми обчисленими основними репродуктивними числами або базовими репродуктивними числами.

### 2.5. Схема бази даних

Збереження всіх даних системи, як тих, що отримані під час початкової реєстрації осіб, так і тих, що надходять у процесі відео моніторингу, здійснюється у базі даних. Розглянемо структуру її таблиць. Початковими є три таблиці: Приміщення, Інфекції та Вакцина. Ці таблиці створюються з самого початку роботи системи адміністратором та лікарем. Приміщення описують границі фіксування пересування людей та ідентифікують при кожному русі з приміщення чи в приміщення. Таблиця Інфекцій описує

базові відомості про можливі інфекції, які передаються повітряно-крапельним шляхом і є об’єктом спостереження поширення, а також таблиця містить інформацію про базове поширення серед осіб. Таблиця Вакцина описує базову інформацію про вакцину, тобто проти якої інфекції використовується, бренд вакцини та її базова ефективність. Також є таблиці, які заповнюються, коли адміністратор здійснює реєстрацію осіб для системи (момент наповнення бази даних). Особи – це усі люди, яких система буде фіксувати. Серед них бувають працівники, лікарі та керівники. Приміщення – це кімнати, обмежені входом та виходом для яких встановлено засоби фіксування. Адміністратор також створює таблиці реєстрації присутності, а заповнюватися вони будуть в процесі фіксації входу чи виходу з приміщення зареєстрованих осіб. Таблиця Інфіковані працівники буде містити знання про людей, які були інфіковані тією чи іншою інфекцією, до того ж час та дату інфікування. Коли людина виздоровіє, її дані мають бути видалені з цієї таблиці. Саме на основі цих таблиць будуть реалізовуватися запити про присутність за введеними критеріями, та обчислюватися ймовірність та надаватися оцінка поширення вірусів. Будуть вибиратися особи, які у вказаний день і час були у заданому приміщенні.

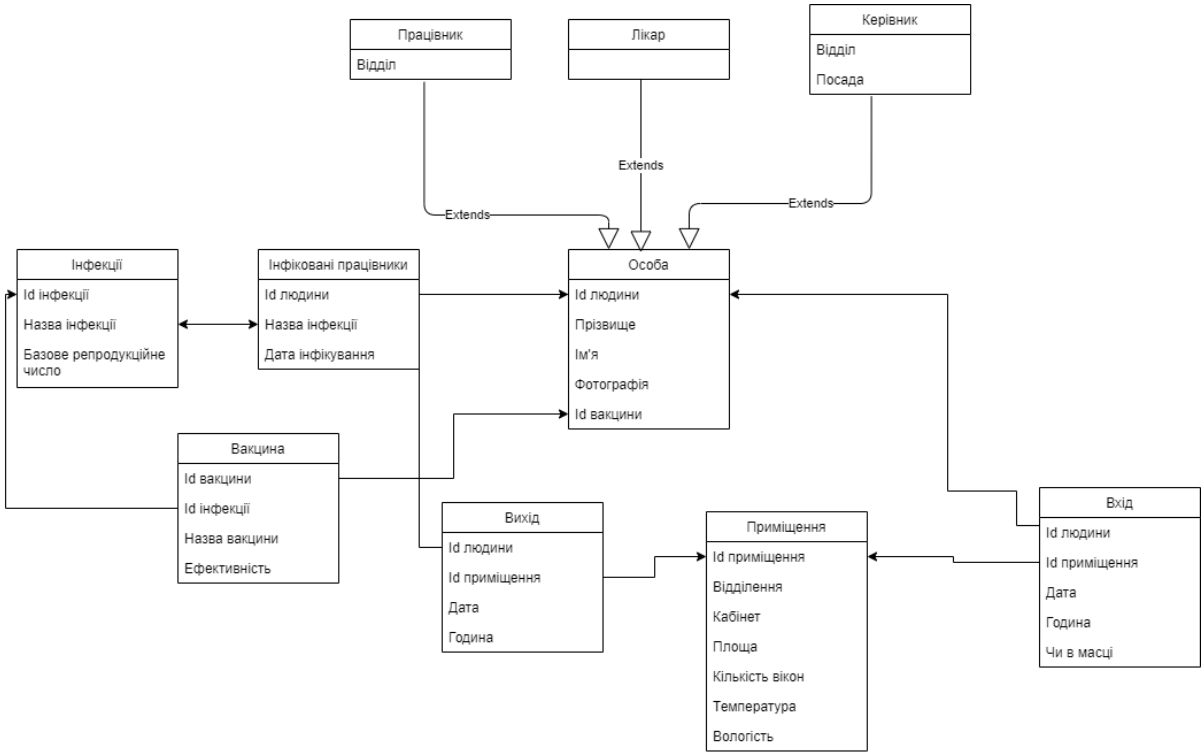


Рис. 2.2

Таблиця Особа – містить інформацію про працівників, лікарів та керівників:

- Id людини – код, первинний ключ
- Прізвище – прізвище особи
- Ім’я – ім’я особи
- Фотографія – фотознімок обличчя користувача
- Id вакцини – код з таблиці Вакцина.

Таблиця Приміщення – містить інформацію про кімнату:

- Id приміщення – код, первинний ключ
- Відділення – назва відділу
- Кабінет – номер кабінету
- Площа – подвійне значення площі кімнати
- Кількість вікон – номер кількості вікон у кімнаті
- Температура – число-значення температури у приміщенні

- Вологість – число-значення вологості кімнати

Таблиця Вхід – містить дані про входження:

- Id приміщення – код поміщення
- Id людини – код особи
- Дата – дата реєстрації
- Година – момент реєстрації у приміщенні
- Чи в масці – прапорець чи людина увійшла у масці

Таблиця Вихід – містить дані про вихід:

- Id приміщення – код аудиторії
- Id людини – код особи
- Дата – дата реєстрації
- Час – момент реєстрації у приміщенні

Таблиця Інфіковані працівники – тримає дані про хворих працівників:

- Id людини – код особи
- Назва інфекції - найменування інфекційного діагнозу
- Дата інфікування – дата, коли людина підхопила інфекцію

Таблиця Інфекція – охоплює інформацію про можливі інфекції, які передаються повітряно-крапельним шляхом:

- Id інфекції – код інфекції, первинний ключ
- Назва інфекції - найменування інфекційного діагнозу
- Базове репродукційне число – значення можливого коефіцієнту поширення інфекції

Таблиця Вакцина – містить інформацію про можливі вакцини від вірусних інфекцій:

- Id вакцини – код вакцини, первинний ключ
- Id інфекції – код інфекції
- Назва вакцини – брендове найменування вакцини від певної інфекції

- Ефективність – значення у відсотках, яке відповідає за захист від вірусів

На рисунку 2.1 подано схему запроектованої бази даних із вказанням таблиць, полів записів та зв'язків між таблицями і полями.

# РОЗДІЛ 3. МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

## 3.1. Аналіз і вибір методів, алгоритмів та засобів розв’язання задачі

Для розв’язання поставленої задачі потрібно обрати алгоритм виявлення обличчя на відео та алгоритм визначення особи людини на відео, а також базу даних, в якій будуть зберігатися дані про зареєстрованих користувачів.

**Алгоритм Віоли-Джонсона**

Метод Віоли-Джонса (англ. Viola-Jones object detection) – метод, що дає змогу виявляти об'єкти на зображеннях в реальному часі. Запропонований у 2001 році Paul Viola і Michael Jones. Хоча метод може розпізнавати різні класи зображень, основним завданням при його створенні було виявлення людей. Існує безліч реалізацій, в тому числі у складі бібліотеки комп'ютерного зору OpenCV (функція cvHaarDetectObjects()). Метод знаходить осіб з високою точністю і низькою кількістю помилкових спрацьовувань. [29]

Метод заснований на таких принципах:

- застосовуються зображення в інтегральному уявленні, що дає змогу швидко знаходити об'єкти для розпізнавання;
- за допомогою ознак Хаара здійснюється пошук необхідного об'єкта (у даному контексті, людини та її рис );
- використовується бустінг (від англ. boost - поліпшення, посилення ), який знаходить найбільш наближених ознак для шуканого об'єкта на даній області зображення;
- всі показники подаються на вхід класифікатора, який відповідає вірно або хибно;
- використовуються каскади ознак для швидкого пропуску кадрів, де немає обличчя.

Навчання класифікаторів йде дуже повільно, але результати пошуку особи дуже швидкі. Метод Віоли-Джонса є одним з кращих за співвідношенням показників ефективність розпізнавання та швидкість роботи. Також цей детектор вирізняється вкрай низькою ймовірністю помилкового виявлення особи. Метод добре працює і розпізнає риси обличчя навіть під невеликим кутом, приблизно до 30 градусів. При куті нахилу більше 30 градусів відсоток розпізнавань значно зменшується. І це не дає змоги в стандартній реалізації детектувати повернуте обличчя людини під довільним кутом, що значною мірою ускладнює або унеможливлює використання методу в сучасних виробничих системах з урахуванням їх зростаючих потреб.

Даний метод в загальному вигляді шукає осіб і риси обличчя за загальним принципом скануючого вікна. Основним завданням скануючого вікна є виявлення особи та рис обличчя людини на цифровому зображенні. Наприклад, на зображенні є шукані об'єкти. Зображення представлено двовимірної матрицею пікселів розміром w * h, в якій кожен піксель має значення:

- Від 0 до 255, якщо це чорно-біле зображення;
- Від 0 до 2553, якщо це кольорове зображення (компоненти R, G, B).

Як результат своєї роботи, метод повинен визначити осіб та їхні риси і позначити їх – пошук здійснюється в активній області зображення прямокутними ознаками, за допомогою яких і описується формулою 3.1 знайдена особа та її риси:

$$rectanglei = \{x, y, w, h, a\} \quad (3.1)$$

де $x$, $y$ – координати центру $i$-го прямокутника; $w$ – ширина; $h$ – висота; $a$ – кут нахилу прямокутника до вертикальної осі зображення.

Для того, щоб проводити будь-які дії з даними, використовується інтегральне представлення зображень в методі Віоли-Джонса. Таке уявлення використовується часто і в інших методах, наприклад, в вейвлет-перетвореннях. Інтегральне представлення дає змогу швидко розраховувати сумарну яскравість довільного прямокутника на даному зображенні, причому час розрахунку яскравості не залежить від розміру прямокутника.

Інтегральне представлення зображення – це матриця, що збігається за розмірами з вихідним зображенням. В кожному елементі її зберігається сума інтенсивностей всіх пікселів, що знаходяться лівіше і вище даного елемента. Елементи матриці розраховуються за формулою 3.2:

$$L(x, y) = \sum_{i=0,j=0}^{i\leq x, j\leq y} I(i, j) \quad (3.2)$$

де $I\,(i, j)$ – яскравість пікселя вихідного зображення.

Кожен елемент матриці L$[x, y]$ являє собою суму пікселів в прямокутнику від (0,0) до ($x$, $y$), тобто значення кожного пікселя ($x$, $y$) дорівнює сумі значень всіх пікселів лівіше і вище даного пікселя ($x$, $y$). Розрахунок матриці займає лінійний час, пропорційний до кількості пікселів у зображенні, тому інтегральне зображення розраховується за один прохід. Розрахунок матриці можливий за формулою 3.3:

$$L(x, y) = I(x, y) - L(x - 1, y - 1) + L(x, y - 1) + L(x - 1, y). \quad (3.3)$$

За такою інтегральною матрицею можна дуже швидко порахувати суму пікселів довільного прямокутника з довільною площею.

Ознака Хаара – відображення f: X => Df, де Df – множина допустимих значень ознаки. Якщо задані ознаки f1, ..., fn, то вектор ознак x = (f1 (x), ..., fn (x)) називається описом об'єкта за ознаками x ∈ X. Опис за ознаками допустимо ототожнювати з самими об'єктами. При цьому множину X = Df1 * ... * Dfn називають просторовою ознакою.

Ознаки поділяються на такі типи в залежності від множини Df:

- бінарна ознака, Df = {0,1};
- номінальна ознака: Df – кінцева множина ;
- порядкова ознака: Df – кінцева впорядкована множина;
- кількісна ознака: Df – множина дійсних чисел.

Природно, бувають прикладні завдання з різнотипними ознаками, для їх вирішення підходять далеко не всі методи.

У стандартному методі Віоли-Джонса використовуються прямокутні ознаки, які називаються примітивами Хаара. У розширеному методі Віоли-Джонса, що використовується в бібліотеці OpenCV, використовуються додаткові ознаки, які наведено на рис. 3.1.

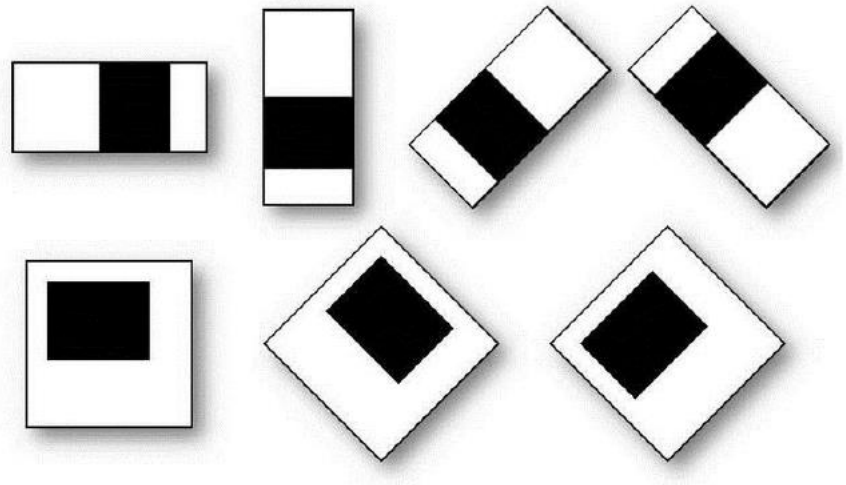

Рис. 3.1. Додаткові ознаки розширеного методу Віоли-Джонса

Обчислюваним значенням такої ознаки буде результат формули 3.4:

$$F = X - Y \tag{3.4}$$

де $X$ - сума значень яскравостей точок, які закриваються світлою частиною ознаки; $Y$ - сума значень яскравостей точок, які закриваються темною частиною ознаки. Для їх обчислення використовується поняття

інтегрального зображення. Ознаки Хаара дають точкове значення перепаду яскравості по осі *X* і *Y* відповідно.

У контексті алгоритму, є безліч об'єктів (зображень), розділених деяким чином на класи. Задано кінцеву множину зображень, для яких відомо, до якого класу вони відносяться (наприклад, це може бути клас «фронтальне положення носа»). Ця множина називається навчальною вибіркою. Класова приналежність решти об'єктів не відома. Потрібно побудувати алгоритм, здатний класифікувати довільний об'єкт з вихідної множини.

Класифікувати об'єкт – значить, вказати номер (або найменування класу), до якого відноситься даний об'єкт. Класифікатор (classifier) – в задачах класифікації це апроксимуюча функція, яка приймає рішення, до якого саме класу даний об'єкт належить. Навчальна вибірка – кінцеваа множина даних.

Розпізнавання образів по своїй суті саме є класифікацією зображень і сигналів. У разі методу Віоли-Джонса для ідентифікації та розпізнавання обличчя класифікація є двокласовою.

Постановка класифікації виглядає наступним чином:

Є X – множина, в якій зберігається опис об'єктів, Y – кінцеве множина номерів, що належать класам. Між ними є залежність - відображення Y *: X => Y. Навчальна вибірка представлена Xm = {(x1, y1), ..., (xm, ym)}. Конструюється функція f від вектора ознак X, яка видає відповідь для будь-якого можливого спостереження X і здатна класифікувати об'єкт x∈X. Дане просте правило повинне добре працювати і на нових даних.

Для вирішення проблеми такого достатньо складного навчання класифікаторів існує технологія бустінга.

Бустінг – комплекс методів, що сприяють підвищенню точності аналітичних моделей. Ефективна модель, що допускає мало помилок

класифікації, називається «сильною». «Слабка» ж, навпаки, не дає змоги надійно розділяти класи або давати точні прогнози, робить в роботі велику кількість помилок. Тому бустінг (від англ. Boosting - підвищення, поліпшення) означає дослівно «посилення» «слабких» моделей – це процедура послідовної побудови композиції алгоритмів машинного навчання, коли кожен наступний алгоритм прагне компенсувати недоліки композиції всіх попередніх алгоритмів.

Ідея бустінга була запропонована Робертом Шапіро (Schapire) наприкінці 90-х років, коли треба було знайти рішення питання про те, щоб маючи безліч поганих (таких, що незначно відрізняються від випадкових) алгоритмів навчання, отримати один хороший. В основі цієї ідеї лежить побудова ланцюжка (ансамблю) класифікаторів, який називається каскадом, кожен з яких (крім першого) навчається на помилках попереднього. Наприклад, один з перших алгоритмів бустінга Boost1 використовував каскад з 3-х моделей, перша з яких навчалася на всьому наборі даних, друга – на вибірці прикладів, в половині з яких перша дала правильні відповіді, а третя – на прикладах, де «відповіді» перших двох розійшлися. Таким чином, має місце послідовна обробка прикладів каскадом класифікаторів, причому так, що завдання для кожного наступного етапу стає важчим. Результат визначається шляхом простого голосування: екземпляр відноситься до того класу, який виданий більшістю моделей каскаду.

Бустінг над деревами рішень вважається одним з найбільш ефективних методів з точки зору якості класифікації. У багатьох експериментах спостерігалося практично необмежене зменшення частоти помилок на незалежній тестовій вибірці в міру нарощування композиції. Більше того, якість на тестовій вибірці часто продовжувала поліпшуватися навіть після досягнення безпомилкового розпізнавання на всій навчальній вибірці. Це

перевернуло існуючі довгий час уявлення про те, що для підвищення узагальнюючої здатності необхідно обмежувати складність алгоритмів [9]. На прикладі бустінга стало зрозуміло, що гарною якістю можуть володіти як завгодно складні композиції, якщо їх правильно налаштовувати.

**Алгоритм локальних бінарних шаблонів**

Він заснований на локальному бінарному операторі і є одним з найкращих дескрипторів текстури. Потреба в системах розпізнавання обличчя зростає з кожним днем. Вони використовуються в системах управління входом, системах спостереження, розблокуванні смартфона тощо. У цьому розділі ми детально опишемо LBPH.

Алгоритм побудови гістограми на основі локальних бінарних шаблонів був запропонований у 2006 році. Він заснований на локальному бінарному операторі. Він широко використовується для розпізнавання обличчя завдяки своїй обчислювальній простоті. [30]

Алгоритм побудови гістограми на основі локальних бінарних шаблонів включає такі етапи:

- Створення набору даних;
- Визначення обличчя;
- Виявлення особливостей;
- Класифікація.


Рис 3.2. Схема роботи алгоритму LBPH

Алгоритм LBPH є частиною OpenCV. [28]

Припустимо, у нас є зображення з розмірами $N \times M$. Ми ділимо його на сегменти однакової висоти та ширини, в результаті чого маємо розмір $m \times m$ для кожного сегмента.

Рис. 3.3. Поділ зображення на сегменти

Локальний двійковий оператор використовується для кожного сегмента. Оператор LBP визначений у вікні 3×3.

$$LBP(x_c, y_c) = \sum_{p=0}^{p-1} \quad 2^p s(i_p - i_c) \tag{3.5}$$

тут $(x_c, y_c)$ – центральний піксель з інтенсивністю $i_c$, а $i_n$ –інтенсивність сусіднього пікселя.

Використовуємо середнє значення пікселя як порогове значення, порівнюємо піксель з його 8 найближчими пікселями за допомогою цієї формули 3.6:

$$s(x) = \{1, x \geq 0 \;\; 0, x < 0 \tag{3.6}$$

Якщо значення сусіда більше або дорівнює центральному значенню, значення функції встановлюється як 1, інакше воно встановлюється як 0. Таким чином, ми отримуємо загалом 8 двійкових значень від 8 сусідів. Після поєднання цих значень ми отримуємо 8-бітове двійкове число, яке для нашої зручності переводиться у десяткове значення. Це десяткове число називається значенням LBP пікселя, а його діапазон - 0-255.

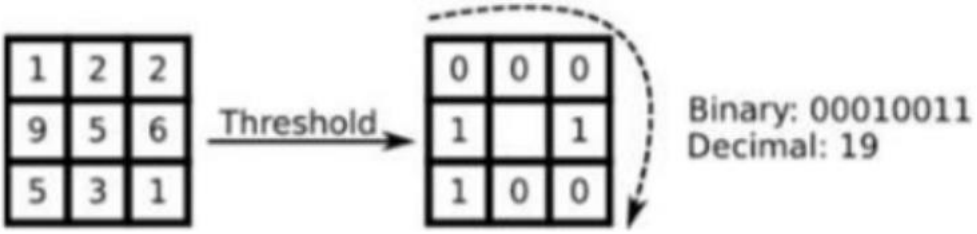


Рис. 3.4. Візуалізація обчислення інтенсивності пікселя

Після генерації значень LBP значення гістограми області створюється шляхом підрахунку кількості подібних значень LBP в сегменті.

Після створення гістограми для кожної області всі гістограми об'єднуються, утворюючи єдину гістограму, яка називається вектором ознак зображення.

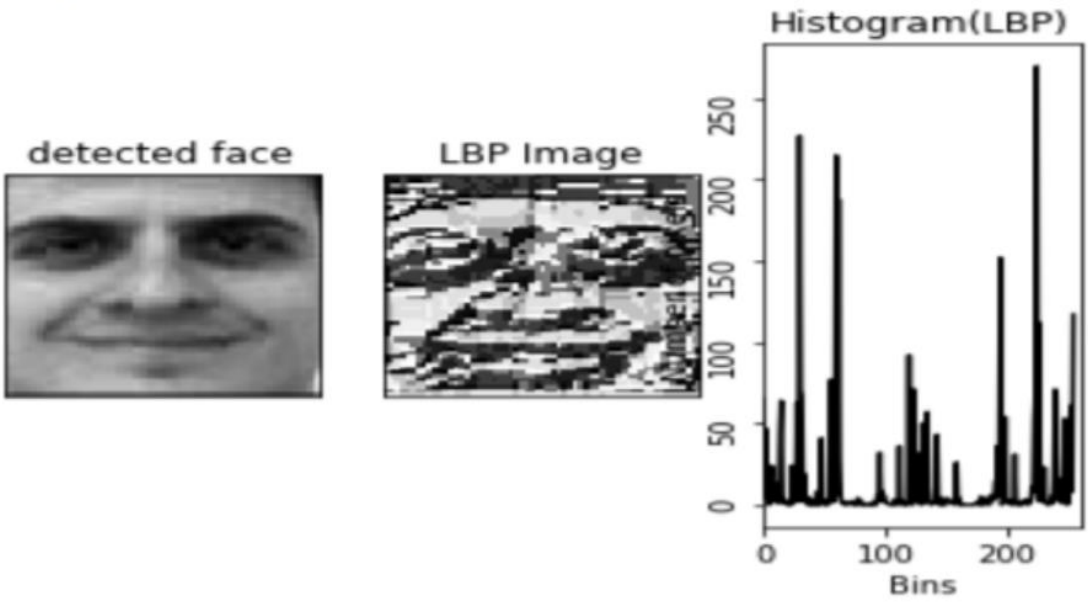


Рис. 3.5. Приклад вигляду зображень у LBP поданні
та у вигляді гістограми LBP

Тепер ми порівнюємо гістограми тестового зображення та зображення в базі даних, а потім повертаємо зображення з найближчою гістограмою. Порівняння можна зробити за допомогою різних методик, таких як

евклідова відстань, квадрат чи, абсолютне значення тощо. Виберемо методику обчислення евклідової відстані.

Евклідова відстань обчислюється шляхом порівняння характеристик тестового зображення з характеристиками, збереженими в наборі даних. Мінімальна відстань між тестовим та вихідним зображенням дає відсоток відповідності. [13]

На виході, ми отримуємо ідентифікатор зображення з бази даних, якщо тестове зображення розпізнано.

Переваги алгоритму LBPH:

- оператор LBP стійкий до монотонних перетворень сірого масштабу;
- потребує меншого зберігання особистих даних та менше часу на розпізнавання;
- він може представляти локальні особливості у зображеннях;
- LBPH може розпізнавати як бічну, так і передню грані.

### 3.2. Обчислення оцінки процесу поширення інфекційних збудників

Щоб відповісти на питання скільки людей можуть заразитися тією чи іншою інфекцією візьмемо за основу обчислення ефективного репродуктивного числа для прогнозування кількості заражень.

Є різні чинники і різні алгоритми знаходження кількості потенційних інфікованих, але більшість з них базуються на базовому репродукційному числі або ж на обчисленні базового репродукційного числа.

Введемо поняття ефективного репродуктивного числа. Це значення, подібне до базового репродукційного числа, але показує середню кількість осіб, що можуть інфікуватися від інфікованого упродовж заразного періоду хвороби за реальних умов навколишнього середовища. Тобто якраз те що

нам потрібно, а саме із врахуванням таких показників як температура, вологість та інші чинники, які впливають на ефективність передачі вірусу.

Ефективне число розмноження обчислюється за добутком основного репродуктивного числа на фактори поширення вірусних часток. Отже, цю формулу також можна записати так:

$$R_0 = \mathrm{c} * T * t \tag{3.7}$$

де *c* - це рівень контакту між сприйнятливими та інфікованими особами

*T* – трансмісивність, тобто ймовірність зараження при контакті.

Знаємо, що поширеність вірусів залежить від відстані між можливим розповсюджувачем на іншими суб'єктами. Тоді добавимо такий чинник, але так як у нас відстань між людьми буде залежати від кількості людей в приміщенні, тому одразу інтерпретуємо частку сприятливої популяції хазяїна під площа та кількість людей у приміщенні. Формуда буде виглядати так:

$$R = R_0 * \frac{k_1 * N}{S} \tag{3.8}$$

Де N – кількість людей у приміщення, а S - загальна площа приміщення, $k_1$ - це коефіцієнт, який показує скільки квадратних метрів людині потрібно щоб шанс інфікування не збільшувався.

Введемо до того ж чинники температури та вологості приміщення, які теж грають не малу роль. Отже формула розшириться до такої:

$$R = R_0 * \frac{k_1 * N}{S} * \frac{k_2}{T} * \frac{k_3}{\varphi} \tag{3.9}$$

T – температура повітря, а φ - відносна вологість приміщення. Якщо температура чи вологість приймають значення 0, то необхідно брати мінімальне значення вище нуля, у нашому випадку це 1. Якщо ж температура опускається нижче 0, то так само, ефективність розповсюдження при цьому не зростає значним чином, тому за граничне значення теж береться 1.

На швидкість розповсюдження вірусів ще може впливати провітрюваність приміщення. Так як провітрюваність оцінити якимись цифровими показниками досить складно, то було прийнято рішення спростити його до параметра, який вказує скільки приміщення має відкритих вікон. Розмір вікна при цьому до уваги не береться, вважається що вони однакові. Після усіх розрахунків формула буде виглядати наступним чином:

$$R = R_0 * \frac{k_1 * N}{S} * \frac{k_2}{T} * \frac{k_3}{\text{ф}} * \frac{k_4}{\text{C}^{-1/2}} \tag{3.10}$$

Де С – кількість відкритих вікон.

Залишилося врахувати факт вакцинації. Який теж грає роль у розповсюдженні вірусів. Наявність вакцини важливо впливає на значення ефективного числа розмноження. Так як вакцин є багато для кожного типу віруса, то для розрахунку береться саме ефективність конкретної вакцини. Кожна особа, яка розглядається може мати різні вакцини з різною ефективністю, тому цей показник потрібно вираховувати проходячи по усіх кількості людей у приміщенні:

$$R = R_0 * \frac{k_1 * N}{S} * \frac{k_2}{T} * \frac{k_3}{\text{ф}} * \frac{k_4}{\text{C}^{-\frac{1}{2}}} * \sum_{i=1}^{N} \quad \frac{k_5}{P} \tag{3.11}$$

де Р – заявлена ефективність вакцини.

Отже так буде виглядати кінцева формула. Результат показуватиме скільки людей у конкретній змодельованій ситуації може заразитися, якщо носій певного вірусу був у приміщенні з такими параметрами.

# РОЗДІЛ 4. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

## 4.1. Загальна структура програмного коду

Розробка програмного рішення для відео реєстрації користувачів включала в себе такі етапи:

• розробку бізнес-логіки;

• проектування і розробку користувацького інтерфейсу;

• інтеграцію зовнішніх засобів та технологій;

• проектування бази даних.

Всі етапи виконувалися у інтегрованому середовищі розробки для програмування мовою Python, а саме у PyCharm.

Програмне рішення складається з декількох файлів, які об'єднані в проект, але можуть бути запущені окремо. Це зроблено тому, що система передбачає різних користувачів, кожен з яких має власний набір доступних функцій. У моєму випадку це: Адміністратор, який має доступ до всіх функціональних можливостей системи, користувач – Медпрацівник, який крім реєстрації, може робити запити – отримувати інформацію про осіб, які перебували у певному приміщенні в заданий час, та Працівник, який може зафіксувати свою присутність за допомогою відео.

Умовно файли, з яких складається проект, можна розділити на три групи:

• файли, які реалізують бізнес-логіку;

• графічні інтерфейси;

• обробники графічних інтерфейсів.

До графічних інтерфейсів належать файли MainWindow.py, RegisterWindow.py, ShowListMenu.py.

MainWindow.py – клас графічного інтерфейсу, який зібрав усі можливості програми. З нього можна перейти до будь-якого іншого графічного інтерфейсу, до прикладу у меню формування бази даних.

RegisterWindow.py – клас графічного інтерфейсу, який містить форми для внесення даних у базу даних.

ShowListMenu.py – клас графічного інтерфейсу, що містить форми та віджети для введення критеріїв та відображення запитів на виведення результуючих списків.

До файлів, що виконують бізнес-логіку, належать DataBaseManager.py, OpenCvManager.py. Перший файл містить клас, який має імплементовані методи для взаємодії з базою даних: під’єднатися до бази даних, додати особу, вибрати особу, вибрати список осіб тощо. OpenCvManager - це клас у OpenCvManager.py файлі, що робить усі завдання, які можуть бути пов’язані з технологією OpenCv. Це може бути зробити фотографію, знайти обличчя на відео, розпізнати обличчя тощо. Тобто ці класи спрощують роботу з різними технологіями та запобігають повторному використанню коду, завдяки тому, що окремі логічні частини виділені у методи.

### 4.2. Опис використаних сторонніх бібліотек та модулів

Сьогодні існує багато бібліотек, модулів, проектів, які допомагають спростити написання програмних продуктів, за рахунок уже готових компонент. У цьому рішенні було використано бібліотеку OpenCV для python cv2. Це бібліотека комп'ютерного бачення та машинного навчання з відкритим кодом. У ній більше ніж 2500 алгоритмів, серед яких є як класичні, так і сучасні алгоритми для комп'ютерного бачення та машинного навчання. Ця бібліотека має інтерфейси для різних мов, серед яких зокрема Python. Підключається OpenCV бібліотека за допомогою команди import cv2.

Опишемо основні методи та їх використання.

Метод cv2.face.LBPHFaceRecognizer_create() створює об’єкт, який вміє розпізнавати обличчя на основі алгоритму локальних бінарних шаблонів. [30]

Метод cv2.VideoCapture().cap.read() – фіксує відео-стрім з камери, декодує його та повертає відео фрейм.

Метод recognizer.predict() – за вхідними параметрами облич прогнозує людину із списку.

Метод cv2.imshow() – показує відео-фрейм у новому вікні.

Метод cv2.rectangle() – вимальовує прямокутник всередині фрейма у заданих координатах та із заданим кольором.

Метод cv2.putText() – наносить текст всередині фрейма по заданих координатах.

Метод self.face_cascade.detectMultiScale() – виокремлює обличчя на вхідному відео-фреймі.

Метод recognizer.train() – здійснює тренування навчальної вибірки

Метод recognizer.save() – зберігає результати навчання у файлі.

Наступний модуль це pyqt. PyQt – бібліотека-оболонка, призначена для взаємодії Python з Qt – міжплатформним C++ фреймворком. PyQt розроблений і підтримується компанією Riverbank Computing Limited. Сам Qt розробляється як частина проекту Qt. PyQt забезпечує прив'язки для Qt 4 та Qt 5. PyQt поширюється за вибором ліцензій: GPL версії 3 або комерційної ліцензії. Використовувався даних засіб для створення графічної оболонки та додавання віджетів.

Також використовувався модель pickle. Модуль pickle реалізує потужний алгоритм серіалізації і десеріалізації об'єктів Python. "Pickling" - процес перетворення об'єкта Python в потік байтів, а "unpickling" - зворотна операція, в результаті якої потік байтів перетворюється назад в Python-

об'єкт. Так як потік байтів легко можна записати в файл, модуль pickle широко застосовується для збереження і завантаження складних об'єктів в Python.

Модуль os зі стандартної бібліотеки мови програмування Python використовується для роботи зі встановленою ОС, а також файлової системою ПК. Він містить масу корисних методів для взаємодії з файлами і папками на жорсткому диску. Програми, що працюють з модулем os, що не залежать від типу ОС і легко переносяться на іншу платформу. За допомогою модуля os створювалися папки, перейменовувалися, перевірялося чи існує певна директорія.

Модуль uuid використовувався для найменування файлів зображень. UUID (англ. Universally unique identifier, тобто «універсальний унікальний ідентифікатор») – це стандарт ідентифікації, який використовується для створення програмного забезпечення, стандартизований Open Software Foundation (OSF) як частина DCE – середовища розподілених обчислень. Основним призначенням UUID є дозволити розподіленим системам унікально ідентифікувати інформацію без центру координації. Таким чином, будь-хто може створити UUID і використовувати його для ідентифікації будь-чого з прийнятним рівнем впевненості, що даний ідентифікатор ненавмисно ніколи не буде використаний для чогось іншого.

Модуль PIL для роботи із зображеннями. Бібліотека зображень Python, або PIL (Python Imaging Library) потрібна для обробки графіки в Python. Використано щоб відкрити зображення і передати для подальшого опрацювання.

NumPy – це розширення мови Python, що надає підтримку опрацювання великих багатовимірних масивів і матриць, разом з великою бібліотекою високорівневих математичних функцій для операцій з цими

масивами. Щоб запустити процес навчання на вибірці облич потрібно конвертувати зображення за допомогою цього модуля.

Модуль pypyodbc використовувався для підключення і роботи з базою даних.

Щоб усі вище описані модулі підключити до середовища розробки потрібно скористатися утилітою pip. pip - це інсталятор пакетів для Python. Можна використовувати pip для встановлення пакетів з набору пакунків Python.

### 4.3. Інструкція щодо проектування бази даних

Для того, щоб продукт працював згідно зі своїми функціями, потрібно зробити кілька дій з боку адміністратора.

Для початку потрібно інсталювати SQL server. Це система управління базами даних. Як сервер даних виконує головну функцію по збереженню та наданню даних у відповідь на запити інших застосунків, які можуть виконуватися як на тому ж самому сервері, так і у мережі. Рекомендую встановити Microsoft Sql Server Management Studio 18, бо вона має простий зрозумілий інтерфейс для користування. Цей засіб дасть змогу створити базу даних, а потім програма матиме змогу записувати або зчитувати з неї інформацію.

На рисунку 4.1 можна побачити конфігураційне вікно Sql Server.

Адміністратор повинен змінити або запам’ятати Server name і задати це саме ім’я у файлі конфігурації у полі server name. Після цього натиснути кнопку “Connect”.

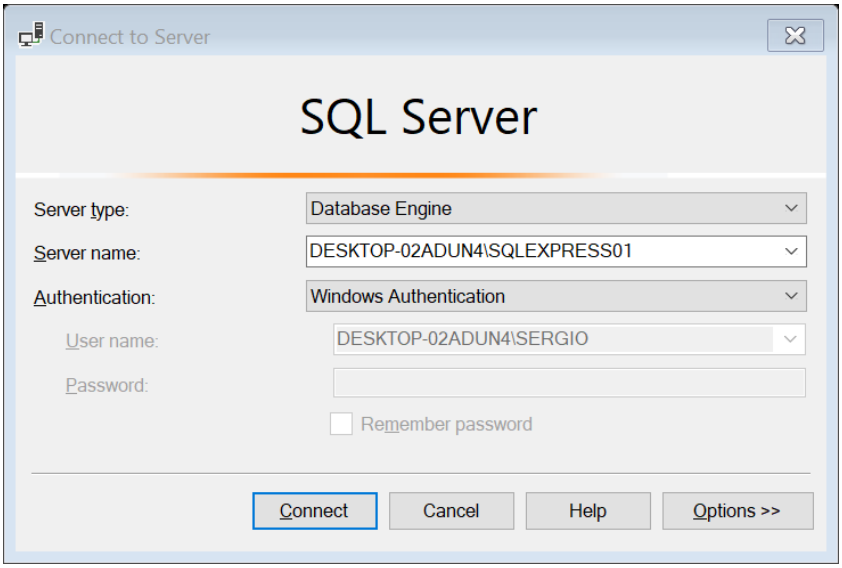


Рис. 4.1. Конфігураційне вікно Sql Server Management Studio

Тепер у нас є відкритий Sql сервер, на якому потрібно створити власну базу даних. Для цього у правій частині Management Studio у частині *Object Explorer* потрібно натиснути правою кнопкою миші по папці *Databases* та вибрати *New database…* як це зображено на рисунку 4.2. Далі вписати назву бази і натиснути кнопку *Create*.

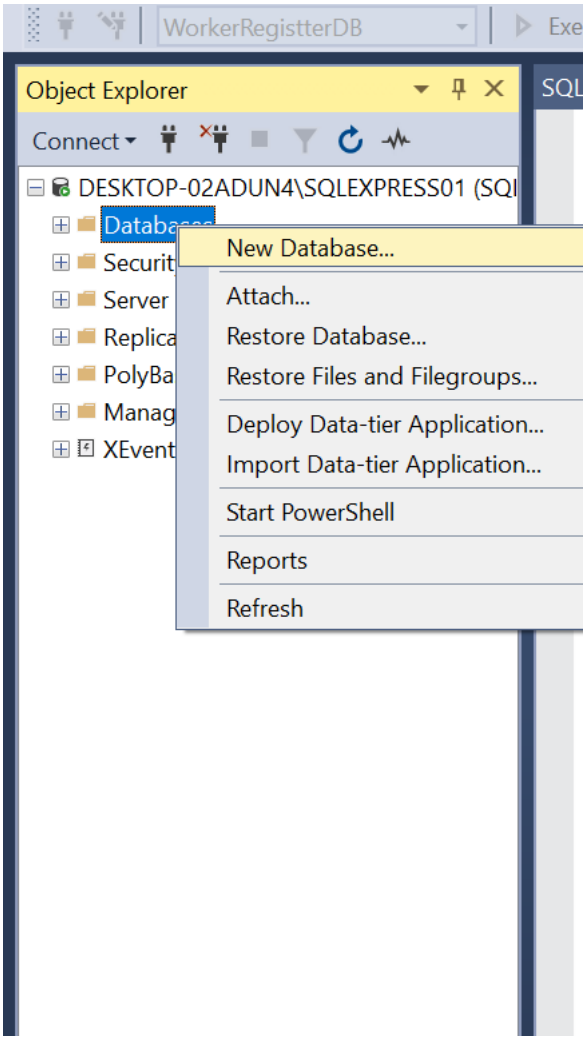


Рис. 4.2. Етап створення нової бази даних

Далі на у частині Toolbar вибрати щойно створену базу даних. Для прикладу в мене вона називається WorkerRegistterDB.

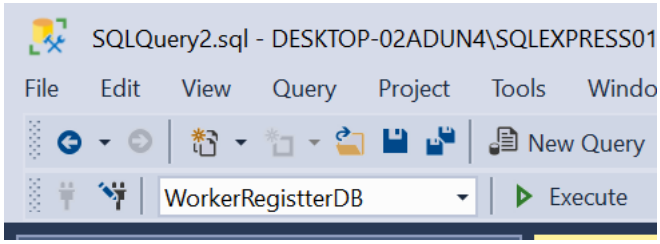


Рис. 4.3. Меню Toolbar SQL Server Management Studio

Наступний крок - це створити таблиці. Для цього можна відкрити файл запитів SqlQueries.sql, у якому описані всі необхідні таблиці та виконати

даний скрипт. Або можна просто скопіювати цей файл у робочу директорію бази даних та виконати його.

### 4.4. Розробка та опис інтерфейсу користувача

Програма має досить простий інтерфейс, без надлишкових інформаційних вікон, написів чи зображень.

При першому запуску програми відкривається вікно, яке доволі лаконічно описує, призначення розробленого програмного продукту.

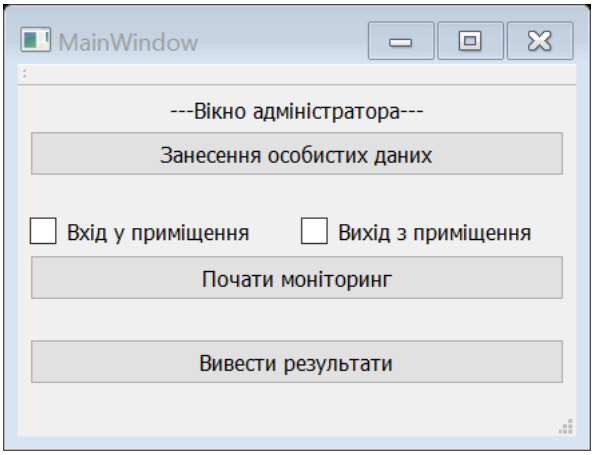


Рис. 4.4. Головне вікно програми відео фіксації

Головне вікно програми відео фіксації зображено на рисунку 5.1. Перше, що бачить користувач, це три кнопки, які описують функціонал продукту. Тож користувач вибирає щось із цього списку і відповідно до вибору виконується подальша логіка. Опишемо можливі варіанти вибору:

- Внесення особистих даних – реєстрація особи, занесення в базу даних інформації про людей, які в подальшому зможуть автоматично проходити фіксацію присутності (розпізнавання) через систему відеоспостереження.
- Почати моніторинг – запуск процесу, який виділяє з відео потоку і фіксує обличчя та встановлює відповідну особу із числа занесених

у внутрішню базу даних, або ж зазначає, що обличчя не вдалося розпізнати.

- Вивести результати – відкривається вікно, у якому потрібно заповнити форму для створення і обробки запиту на отримання присутніх осіб; у результаті виконання запиту буде отримано інформацію за вказаними критеріями.

Розглянемо детальніше ці меню.

На рисунку 4.5 показано графічну оболонку для меню Реєстрації. Угорі вікна є можливість вибрати, якого користувача реєструємо. Передбачено такі варіанти: працівник та керівник. Згідно того, який варіант обрано, додаються поля у форму реєстрації. Наприклад, якщо це працівник, то з'являється поле Посада. Далі – звичайні форми для запису даних. Вікно містить текст, що супроводжує користувача і дає підказки, що, де і які дані потрібно вводити. У цьому вікні потрібно обов'язково ввести таку відомість: прізвище та ім'я особи. Обов'язкові поля позначені *, та підписано, що це є необхідні поля для заповнення. Решта даних є опціональними. До них відносяться,  зокрема, поля посади, відділення. Усі поля вводяться у звичайному письмовому форматі. Після того, як усі дані введені, можна завершити реєстрацію, натиснувши кнопку «Занести у базу даних», яка також запустить наступний процес – Зробити фотографію. Слід зазначити, що для правильної роботи системи потрібно, щоб на фотографії було чітке зображення обличчя особи. Тому після натискання цієї кнопки програма очікує поки на відео-стрімі з'явиться лице людини, а тоді вже завершує процес реєстрації, робить і заносить у базу даних фотографію обличчя та повідомляє про успішне завершення реєстрації.

Вікно реєстрації призначене для адміністратора, тобто користувача, який володіє потрібними правами. Але для того аби закінчити процес реєстрації має бути зроблена фотографія. Тож заповнення даного меню без

самого працівника не може відбутися. Одночасно кожен зможе перевірити достовірність введеної інформації.

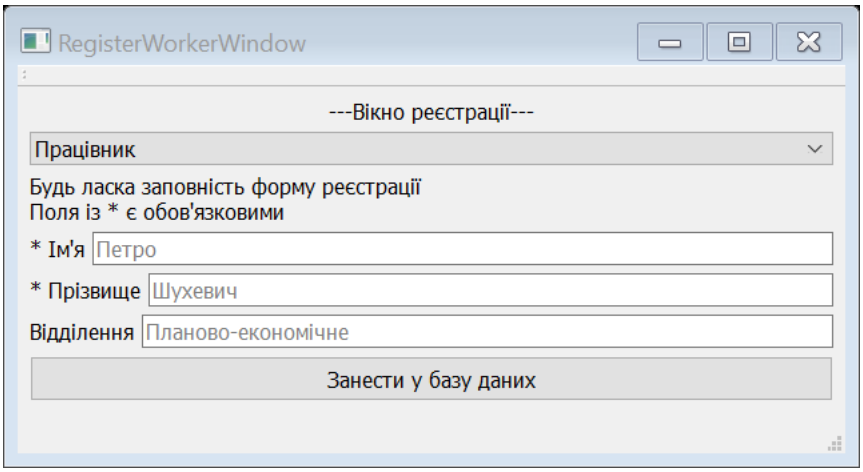


Рис. 4.5. Вікно підменю для реєстрації працівників

Перейдемо до другого розділу меню – це початок моніторингу.

Для того щоб запустити дану функцію, потрібно вибрати в якому режимі здійснюється фіксація присутності: чи коли людина входить у приміщення, чи коли виходить з нього. Вибраний режим визначає у яку таблицю буде внесено запис про реєстрацію та забезпечує правильний потік даних, що дуже важливо для створення у майбутньому результуючої таблиці.

При натисненні на кнопку “Почати моніторинг” здійснюється підключення до веб-камери та зчитування відео-стріму з неї до моменту, коли у ньому фіксується обличчя, яке відразу перевіряється на належність до облич, які занесені попередньо сформовану базу даних системи. При цьому відкривається вікно, яке видно на рисунку 3.3. Програма, яка реалізована у цьому модулі на базі алгоритмів, які описані у попередніх розділах, визначає обличчя з відеопотоку, розпізнає його, порівнюючи з фотографіями із бази даних. Якщо виявлене обличчя належить одному із

зареєстрованих працівників, то на екрані з’являється підпис з іменем відповідної особи. При фіксуванні особи, робиться запис у базі даних у відповідну таблицю входу чи виходу з приміщення та закривається вікно фіксації – програма переходить в режим очікування. Передбачено, що для того, аби повторно зайти, потрібно спочатку зафіксувати вихід і навпаки, крім першого входження.

Важливим і потрібним вдосконаленням цього кроку було б розпізнавання людини, яка знаходиться у захисній масці. Для прикладу це може бути чи сітківка ока чи розріз очей. Чи можливо відбиток чи відсканування пальця.

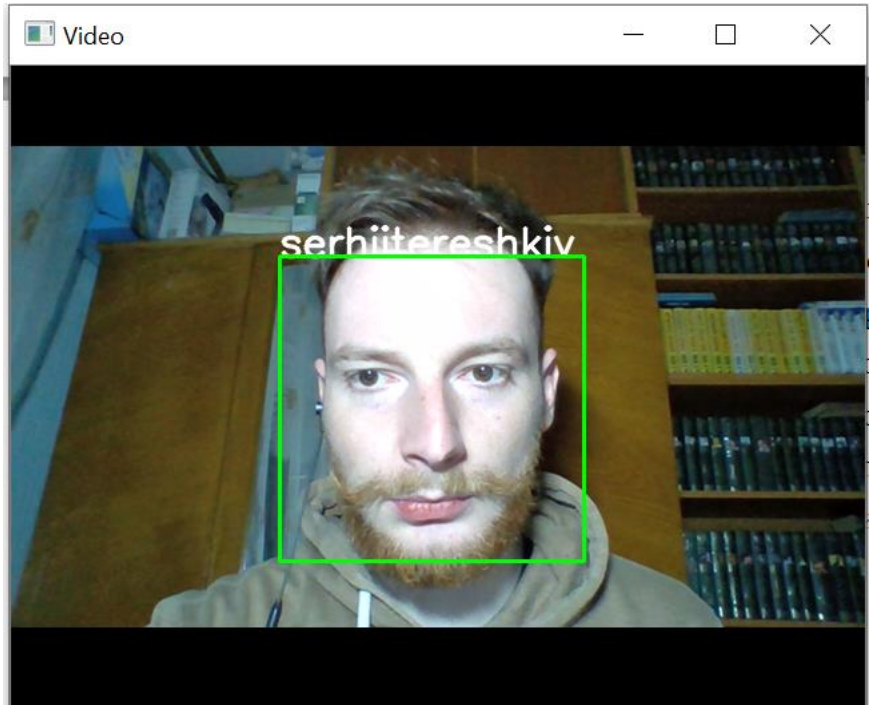


Рис. 4.6. Відображення результату моніторингу

Ще одним розділом є - виведення зведених даних з інформацією про зафіксованих осіб. Аби його отримати потрібно натиснути кнопку «Вивести

результати». Доступне адміністратору та керівнику. Вигляд вікна запиту присутності подано на рис. 4.7.

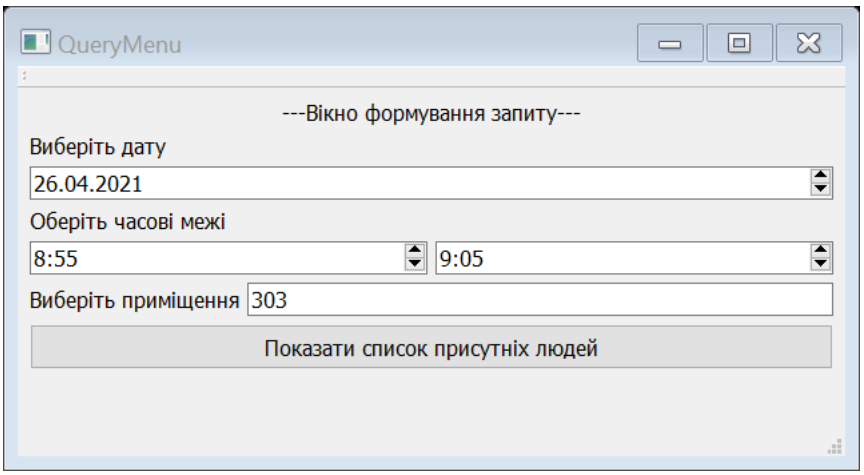


Рис. 4.7. Меню виведення результатів

Відкривається вікно для формування запитів. Для того щоб отримати результат потрібно ще декілька дій. А саме вказати дату фіксації та часовий проміжок з якого по який момент потрібно зафіксувати присутність осіб і аудиторію, в якій фіксувалися особи та натиснути кнопку «Показати список присутніх людей». Після цього виконується запит до бази даних і формується таблиця відвідувачів, як видно на рисунку 4.8.

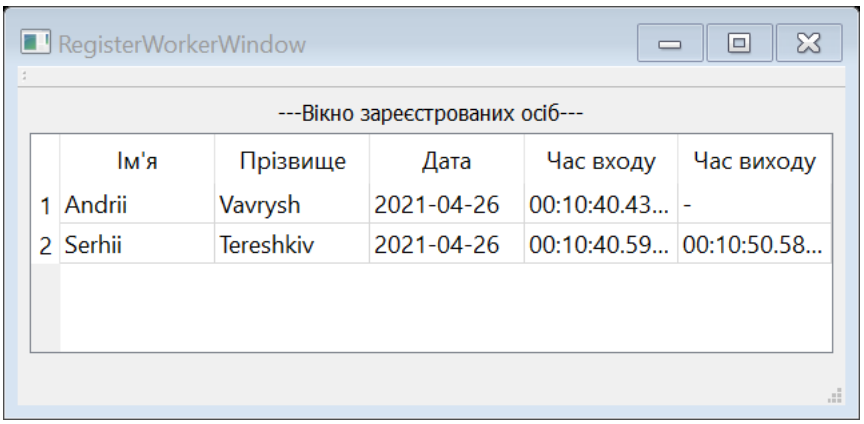


Рис. 4.8. Список зареєстрованих осіб.

Таблиця відкривається у окремому вікні, в якому без надлишкової інформації виведено ім’я та прізвище людини, яка була зареєстрована у певний день, час реєстрації особи при входження у приміщення та час виходу людини з приміщення. Слід зазначити, що людина могла за вибраний проміжок часу і не вийти з аудиторії, то в такому випадку у колонці «Час виходу» зазначається порожній рядок, який означає, що людина на вибраний момент часу знаходиться у приміщенні.

Керівник або працівник медичної сфери (медсестра, лікар, мед брат) підприємства має можливість, на основі проведених консультацій чи аналізів, вносити інформації до бази даних щодо певних осіб, які захворіли за допомогою наступного вікна.

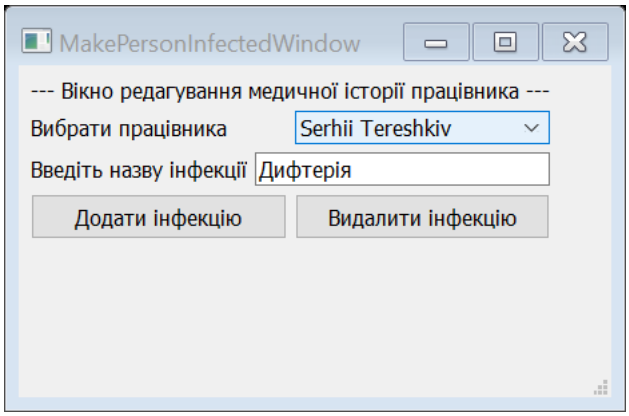


Рис. 4.9. Вікно редагування медичної історії працівника.

Користувачеві потрібно вибрати працівника зі списку, та ввести назву інфекції, якою захворів працівник і натиснути кнопку «Додати інфекцію». Також цим вікном можна скористатися у разі коли працівник виздоровів і в такому випадку оновити базу даних, що працівник виздоровів від певної інфекції.

Для перевірки стану здоров’я працівника існує вікно отримання інформації про здоров’я працівника. Яке дозволяє ввести дані працівника і знайти інформацію по ньому.

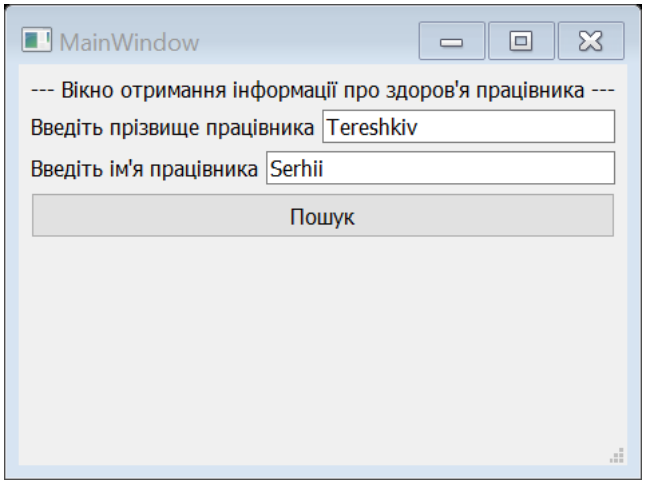


Рис. 4.10. Вікно пошуку медичної інформації про працівника.

При натисненні на кнопку пошуку з’являється нове вікно, яке описує стан здоров’я працівника та його контактних осіб. Тут можна дізнатися чи працівник здоровий чи інфікований, якщо інфікований то якою інфекцією, та скільки контактних осіб було зафіксовано з ним (людей які в певний час знаходилися з ним в певному приміщенні) та хто саме. На основі даних описаних у розділі Обчислення оцінки процесу поширення інфекційних збудників вираховується базове репродукційне число, яке показує скільки в середньому інфікований може заразити інших людей. Якщо це число є менше ніж одиниця значить не відбулося поширення інфекції, значить інші працівники скоріше за все є здорові. Якщо ж число більше за одиницю, то значить вірус міг передаватися іншим особам. Відповідно до результатів, начальник чи лікар вирішує, що робити у подальшому з інфікованим та його контактними особами.

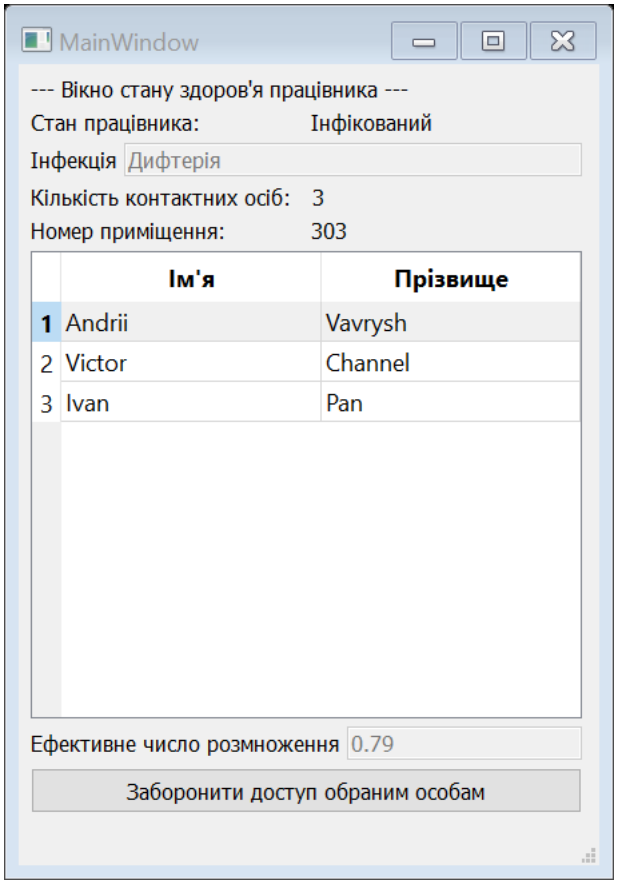


Рис. 4.11. Вікно пошуку медичної інформації про працівника.

Буває таке в бізнесі, що деяких контактних осіб потрібно також ізолювати від інших людей, все залежить від інфекції. Для цього керівники можуть скористатися функціональною можливістю і натиснути кнопку «Заборонити доступ обраним особам». Після цього особи які знаходяться у цьому списку не зможуть проходити фіксацію при вході у приміщення, що унеможливлює їх входження у приміщення і подальше можливе зараження інших працівників.

# РОЗДІЛ 5. РОЗРОБЛЕННЯ СТАРТАП-ПРОЕКТУ

## 5.1. Опис проекту

Перед тим як впроваджувати продукт у широкі маси, потрібно пройти декілька етапів.

Перший – це створення інформаційної мапи проєкту. Тут описано особливості проєкту та приблизний бюджет, який вміщує оренду обладнання, зарплату розробників, яка обчислюється на зазначені терміни розробки. В таблиці 1 ми можемо простежити сформовану інформаційну мапу проекту.

Таблиця 5.1. Інформаційна мапа проєкту

| | |
|---|---|
| Назва номінації | Python програма |
| Назва проєкту | «contagious predict» |
| Назва ВНЗ, спеціальності та факультету | НЛТУ,<br>кафедра інформаційних технологій,<br>Комп’ютерні науки |
| Прізвище, ім’я, по-батькові автора | Терешків Сергій Миколайович |
| Мета і завдання 2 проєкту | 1. Розроблена система має забезпечувати виконання таких функцій як:<br>- створити приблизне передбачення кількості заражених, при умові, якщо є збудник зарази<br>- мати усю історію пересування працівників, щоб відстежувати можливе поширення вірусу |

| | |
|---|---|
| Короткий зміст проєкту | Дана програма дає користувачам приблизно оцінити та шанс поширення певної інфекційної хвороби серед людей, які переміщуються між приміщеннями. |
| Терміни виконання проєкту | 4 місяці |
| Бюджет проєкту | 270 000 грн |

### 5.2 Стратегії проекту

На даний час існує велика кількість готових систем для реєстрації працівників на підприємствах, приміщеннях тощо. Переважно вони реалізовані у паперовому вигляді з паперовими посвідченнями та паперовими журналами, або ж електронні карти доступу та ідентифікаційні значки. Їхніми основними функціями є вести облік осіб, контроль часу на виробництві та обмеження доступу до певних регіонів.

Система реєструє осіб при вході та виході за допомогою обличчя та записує це у базу даних. Ці записи можуть бути використані для створення статистики по працівниках, їхній роботі, можна ввести електронний автоматизований журнал, що спрощує роботу керівникам.

Але головною функцією даної системи – це передбачення та контроль поширеності вірусів серед працівників та відвідувачів конкретного підприємства. Що є надзвичайно зручно мати єдиний журнал оцінки переміщення працівників та можливість обмежити їх пересування за необхідності. У наш час це дуже цінується, адже в епоху коли нові інфекції

намагаються боротись з людством, наша система дає перевагу стороні людства. І на державному рівні такі системи надзвичайно ціняться і знаходять своє існування в різних галузях.

До основних функцій продукту належать:

1) Занести дані про об’єкти реєстрації у базу даних, тобто сформувати базу даних.

2) Фіксування обличчя на місці дослідження та запис інформації у базу даних.

3) Відображення результатів реєстрації працівників за допомогою системи розпізнавання облич. Створення і обробка запитів на формування журналу з бази даних.

4) Обчислення репродукційного числа поширення інфекції для кожного приміщення і по кожній інфекції.

Насамперед ми зрозуміли, що зробить наш проєкт унікальним, що змусить клієнтів користуватися нашими послугами. Зараз ідеальний час для впровадження і розповсюдження подібних систем, а саме нашого проекту.

Щоб запустити успішний стартап і заявити про себе, необхідно уважно ставитися до кожної стадії розвитку проєкту, вміти оперативно контролювати усі ключові моменти.

Тому було визначено такі кроки:

1) Визначено сферу проекту.
2) Візуалізація ідей та цілей.

Було обрано цільову аудиторію, щоб зрозуміти чого хочуть клієнти. Було досліджено і вивчено потреби клієнта. Найчастіше у клієнтів є потреба, але не існує способів її задоволення.

3) Вивчено стратегію і рух конкурентів.

На основі досліджень та складеного плану було обрано відповідну стратегію розвитку нашого продукту, вдалось дослідити популяризацію даної сфери, виявлено переваги та недоліки та можливості покращити проєкт відповідно до потреб та бажань кінцевих споживачів.

### 5.3. Розроблення маркетингової програми стартап-проекту

Основною метою маркетингу є формування й стимулювання попиту. Не менш важливим є і розширення обсягу продажів, ринкової частки та прибутку.

На основі проаналізованих функцій маркетингу, ми можемо виділити наступне:

1. Першою є функція аналізу та синтезу. Мета цієї функції полягає у вивченні зовнішнього середовища й конкретних товарних ринків, а також споживачів та конкурентів.

На основі ринку систем реєстрації та програм обчислювачів ймовірності поширення інфекційних збудників вивчено зовнішні середовища багатьох проектів та було оцінено можливих споживачів та конкурентів.

2. Наступна функція - продуктово-виробнича.

Її мета - створити нові продукти, товари чи послуги. За допомогою продуктово-виробничій функції ми змогли створити абсолютно новий продукт для сфери оцінки поширення вірусів та організували процес його виробництва.

3. Функція реалізації - сформування асортиментів та проведення гнучкої цінової політики, а також організація системи руху товарів.
4. Функція переконання та стимулювання - у цій функції йдеться про формування попиту та стимулювання збуту.

Розміркувавши над поставленою проблемою та її реалізацією, було визначено, що попит буде сформовано на перших етапах розвитку продукту.

Останнім етапом та функцією маркетингу є планування, керування та контроль.

За допомогою аналізу, плануванню маркетингової діяльності та організації систем контролю ми змогли адаптувати наш проєкт.

Відомо, що принципи маркетингу ведуть до загальної спрямованості мети організації в області самого маркетингу й ринкових стратегій. Тому, до основних принципів нашого проєкту ми можемо віднести наступне:

1. Орієнтація на користувача
2. Спрямована на виконання основних завдань.
3. Адаптивність і гнучкість
4. Можливість удосконалення по проханню користувачів
5. Продуктово-цільовий підхід

Для найбільш повного розкриття можливостей маркетингу було виконано наступні вимоги:

- спланувано підготовчу роботу для успішного запуску маркетингу на проєкті;
- розподілено представлення у вищого керівництва проєкту про дійсне місце й роль служби маркетингу як знаряддя діючого підвищення ефективності всієї діяльності проєкту;
- дійшли до усвідомлення того, що принципове положення й мистецтво керування маркетингом, у кінцевому підсумку, вкладається в максимальному використанні проєктом внутрішніх факторів, що піддаються контролю й впливу, і в максимальному застосуванні своїх можливостей до зовнішніх факторів, що не мають вплив;

- чітко усвідомили керівниками, і рядовим персоналом можливостей маркетингу й чітких умов для їхнього здійснення;

Отже, зробивши висновок вдалось отримати спеціальні функції, які мають значний вплив на маркетинг стартапу. До того ж вдалось структуризувати процес та продукт відповідно до існуючих відомих та дієвих принципів маркетингу. Оцінено монетизацію та встановлено плинність цієї складової спираючись на відомі типи монетизації, а також оцінено переваги та недоліки кожного з цих типів.

### 5.4. Елементи фінансової моделі стартапу

Фінансова модель стартапу складається з декількох елементів. В таблиці 1, пункту 5.1 даного розділу, можна побачити, що бюджет проекту складає 270 000 гривень. Фінансову підтримку можна знайти на IT-ринку стартапів. Якщо проект являється новаторським або перспективним відносно існуючих аналогів, тоді знайти інвестора не буде великою проблемою, в принципі проект відповідає цим вимогам.

Проект, що розглядається, отримав такий бюджет виходячи з ключових елементів в які слід вкласти кошти. Основним з них є витрати на приміщення, технічне забезпечення та спеціалістів, що будуть розробляти продукт. А також на рекламу.

Додаткові витрати складають витрати на спеціалістів в області тестування програмного забезпечення та хостингу, де буде викладено готовий до роботи інструмент. В таблиці 4 показано всі складові формування фінансової моделі стартапу.

Таблиця 5.2. Складові формування фінансової моделі стартапу

| **Статті витрат** | **Одиниці виміру** | **Фактична кількість, шт.** | **Ціна одиниці, грн.** | **Разом, грн.** |
|---|---|---|---|---|
| Сировина і матеріали | шт. | - | - | 41299 |

| | | | | |
|---|---|---|---|---|
| Паливо та електроенергія на технологічні цілі | - | - | - | 1000 |
| Основна заробітна плата | грн. | 4 | - | 41750 |
| Відрахування на соціальне страхування | грн. | - | - | 9185 |
| Витрати на утримання й експлуатацію устаткування | грн. | - | - | 1000 |
| Загальновиробничі витрати, у т.ч.: | | | | |
| - змінні; | - | - | - | 7975 |
| - постійні; | - | - | - | 15000 |
| *Разом виробничих витрат:* | | | | 22975 |
| Адміністративні витрати | - | - | - | 20398.75 |
| Витрати на збут | - | - | - | 4785 |
| Інші операційні витрати | - | - | - | 5000 |
| *Разом виробничих і операційних витрат:* | | | | 170367.8 |

Результат калькуляції показує нам, що за 4 місяці активної розробки продукту необхідно затратити 270 00 грн.

### 5.5. Висновки

Обирається остаточна стратегія на підставі даних попередніх етапів. Також необхідно врахувати характеристики програмного забезпечення та обґрунтувати доцільність додаткових послуг.

Після детального аналізу доцільності розробки задуманого продукту на предмет актуальності системи, прибутку з її розробки та стратегії розвитку можна зробити деякі висновки. Перш за все, високу вартість

продукту визначає дороге апаратне забезпечення, але висока затребуваність і популярність впливають на чистий прибуток від розробки. Оскільки метою проекту є розробка програмного рішення для обчислення ймовірності поширення інфекційних збудників, тому більше робота направлена на самонавчання та дослідження. Відповідно, отримання відгуків від потенційних користувачів дасть змогу покращити систему і отримати кращі прибутки.

До того ж проект є новий, тому є шанс очікувати стрімкого росту продукту після певного періоду користування.

На нашу думку, перше на, що потрібно орієнтуватись на початковій стадії архітектури проєкту це стратегія. Стратегія - являє собою так звану модель узагальнення дій, які є необхідними для досягнення поставлених цілей завдяки координуванню і розподілу ресурсів фірми, тобто введення стратегії фірми полягає в складанні планів досягнення цілей, який передбачає усі можливості фірми. Вибір тої чи іншої стратегії насамперед залежить від специфіки та виду конкретного проєкту.

Можна стверджувати, що будь-яка стратегія є дієвою та правильною, тому ми можна рекомендувати застосовувати різні принципи, спиралися на ті функції, які якомога дієво працюють на проєкті.

До того ж потрібно розвивати маркетингові вимоги, бо проект має бути популярний серед користувачів та користуватись попитом.

# ВИСНОВКИ

У результаті виконання магістерської роботи було проведено ознайомлення з предметною областю та огляд методів оцінювання поширення інфекційних збудників, пошуку аналогів додатків, які використовують в медичних сферах, було обрано технології, за допомогою яких можна вирішити поставлені завдання, та запроектовано інформаційну систему для оцінки поширення інфекційних збудників, а саме імплементовано алгоритми для розпізнавання і визначення облич на відео, розроблено дії з базою даних, внесення осіб у базу даних, створення результуючих запитів щодо стану здоров'я працівників, та стану здоров'я контактних працівників, впроваджено логіку обчислення ймовірність зараження для контактних осіб. Як результат, отримані дані можна зберігати та на основі них визначити на скільки поширилася інфекцію, та на скільки людей могла поширитися за допомогою розробленого рішення. Даний продукт описаний, протестований та запущений в тестовому режимі. Для користувачів даної системи – інформаційна система представляє зручний інструмент у вигляді настільної програми, хоча розділена на декілька частин і залежно від призначення може виконуватися на міні-комп'ютерах так і на звичайних персональних пристроях.

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. V. Gupta, D. Sharma. A Study of Various Face Detection Methods // International Journal of Advanced Research in Computer and Communication Engineering Vol. 3, May 2014, №5. С. 6694–6697.
2. P. Viola, M. Jones. Rapid object detection using a boosted cascade of simple features / 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Vol. 1. 8–14 December 2001 / The Institute of Electrical and Electronics Engineers, Inc. С. 511–518.
3. Y. Freund, R. E. Schapire. A Short Introduction to Boosting // Journal of Japanese Society for Artificial Intillegence. 1999, №14(5), С. 771-780.
4. D. Roth, The SNoW Learning Architecture // Technical Report UIUCDCS-R-99-2102 . UIUC Computer Science Department, 1999.
5. M. Nilsson, M. Dahl, I. Claesson. The successive mean quantization transform / Proceedings of IEEE Int. Conf.ICASSP 2005, Vol. 4 / The Institute of Electrical and Electronics Engineers Signal Processing Society С. 429 – 432.
6. H. A. Rowley, S. Baluja, .T. Kanade. Neural Network-Based Face Detection // PAMI, January 1998.
7. E. Osuna, R. Freund, F. Girosi. Training support vector machines:an application to face detection // In Proceedings of Computer Vision and pattern Recognition 1997, С. 130-136.
8. Machine learning methods // Graphicon. [електронний ресурс]: [Веб-книжка] http://www.graphicon.ru/oldgr/ru/publications/text/gc2006avezh.pdf.
9. T. Rawlinson, A. Bhalerao, L. Wang. Principles and Methods for Face Recognition and Face Modelling // Handbook of research on computational forensics, digital crime and investigation: methods and solutions. IGI Global, С. 53-78.
10. M. Turk, A. Pentland. Eigenfaces for recognition. // Cognitive Neuroscience, 1991, №3(1), С.71–86.

11. Інфекційні захворювання [електронний ресурс] https://uk.wikipedia.org/wiki/Інфекційні_захворювання
12. Математичне моделювання інфекційних захворювань [електронний ресурс] Математичне моделювання інфекційних захворювань — Вікіпедія (wikipedia.org)
13. Трекінг поширення вірусу [електронний ресурс] https://www.sciencemag.org/news/2020/03/cellphone-tracking-could-help-stem-spread-coronavirus-privacy-price
14. OpenCV [електронний ресурс]: [Веб-сайт] https://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_tutorial.html
15. Detecting objects using Haar Cascade Classifier [електронний ресурс]: [Веб-сайт] https://towardsdatascience.com/computer-vision-detecting-objects-using-haar-cascade-classifier-4585472829a9
16. Face recognition using Local Binary Patterns (LBP) [електронний ресурс]: [Веб-сайт] https://globaljournals.org/GJCST_Volume13/1-Face-Recognition-using-Local.pdf
17. Python (programming language) [електронний ресурс]: [Веб-сайт] https://en.wikipedia.org/wiki/Python_(programming_language)
18. Qt [електронний ресурс]: [Веб-сайт] https://en.wikipedia.org/wiki/Qt_(software)
19. Математичне моделювання і прогнозування захворюваності на ротавірусну інфекцію, д.т.н, професор Молчанов О.А., аспірант Терещенко І.О.

## ДОДАТОК 1. ФАЙЛ MAINMENU.PY

```
    # -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'mainwindow.ui'
#
# Created by: PyQt5 UI code generator 5.5.1
#
# WARNING! All changes made in this file will be lost!

from PyQt5 import QtCore, QtGui, QtWidgets
```

```
import OpenCvManager
import StudentDataBaseProcessor
from RegisterMenu import Ui_RegisterWindow
import DataBase
from ShowListMenu import Ui_ShowListWindow


class Ui_MainWindow(object):

    def setupUi(self, MainWindow):
        #standart settings
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(418, 279)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.gridLayout = QtWidgets.QGridLayout(self.centralwidget)
        self.gridLayout.setObjectName("gridLayout")
        self.verticalLayout_2 = QtWidgets.QVBoxLayout()
        self.verticalLayout_2.setObjectName("verticalLayout_2")
        self.verticalLayout = QtWidgets.QVBoxLayout()
        self.verticalLayout.setObjectName("verticalLayout")

        self.titleLabel = QtWidgets.QLabel(self.centralwidget)
        self.titleLabel.setText("---Вікно адміністратора---")
        #self.titleLabel.setText("---Вікно працівника---")
        self.titleLabel.setAlignment(QtCore.Qt.AlignCenter)
        self.verticalLayout.addWidget(self.titleLabel)

        self.registerButton = QtWidgets.QPushButton(self.centralwidget)
        self.registerButton.setObjectName("registerButton")
        self.verticalLayout.addWidget(self.registerButton)

        spacerItem = QtWidgets.QSpacerItem(5, 10, QtWidgets.QSizePolicy.Minimum,
QtWidgets.QSizePolicy.Expanding)
        self.verticalLayout.addItem(spacerItem)

        self.horizontalLayoutForMonitoring = QtWidgets.QHBoxLayout()
        self.enterCheckBox = QtWidgets.QCheckBox("На вхід")
        self.exitCheckBox = QtWidgets.QCheckBox("На вихід")
        self.monitoringButton = QtWidgets.QPushButton(self.centralwidget)
        self.monitoringButton.setObjectName("monitoringButton")
        self.horizontalLayoutForMonitoring.addWidget(self.enterCheckBox)
        self.horizontalLayoutForMonitoring.addWidget(self.exitCheckBox)
        self.verticalLayout.addLayout(self.horizontalLayoutForMonitoring)

        self.verticalLayout.addWidget(self.monitoringButton)

        self.verticalLayout.addItem(spacerItem)

        self.listButton = QtWidgets.QPushButton(self.centralwidget)
        self.listButton.setObjectName("listButton")
        self.verticalLayout.addWidget(self.listButton)
        #self.cameraOnlineLabel = QtWidgets.QLabel(self.centralwidget)
        #self.cameraOnlineLabel.setObjectName("cameraOnlineLabel")
        #self.verticalLayout.addWidget(self.cameraOnlineLabel)
        self.verticalLayout_2.addLayout(self.verticalLayout)
        self.video = QtWidgets.QWidget(self.centralwidget)
        self.video.setObjectName("video")
        #self.verticalLayout_2.addWidget(self.video)
        self.gridLayout.addLayout(self.verticalLayout_2, 0, 0, 1, 1)
```

```
            MainWindow.setCentralWidget(self.centralwidget)
            self.menubar = QtWidgets.QMenuBar(MainWindow)
            self.menubar.setGeometry(QtCore.QRect(0, 0, 618, 22))
            self.menubar.setObjectName("menubar")
            MainWindow.setMenuBar(self.menubar)
            self.statusbar = QtWidgets.QStatusBar(MainWindow)
            self.statusbar.setObjectName("statusbar")
            MainWindow.setStatusBar(self.statusbar)
            self.toolBar = QtWidgets.QToolBar(MainWindow)
            self.toolBar.setObjectName("toolBar")
            MainWindow.addToolBar(QtCore.Qt.TopToolBarArea, self.toolBar)

            # connect to slots
            #self.enterCheckBox.toggled()

            self.monitoringButton.clicked.connect(self.StartMonitoring)
            self.registerButton.clicked.connect(self.OpenRegisterStudentWindow)
            self.listButton.clicked.connect(self.OpenResultListWindow)

            self.retranslateUi(MainWindow)
            QtCore.QMetaObject.connectSlotsByName(MainWindow)

            # create opencv stuff
            self.OpenCvManager = OpenCvManager.OpenCvManager()

            # create db stuff
            self.DBManager = DataBase.DataBase()


        def retranslateUi(self, MainWindow):
            _translate = QtCore.QCoreApplication.translate
            MainWindow.setWindowTitle(_translate("MainWindow", "MainWindow"))
            self.registerButton.setText(_translate("MainWindow", "Занесення особистих
даних"))
            self.monitoringButton.setText(_translate("MainWindow", "Почати
моніторинг"))
            self.listButton.setText(_translate("MainWindow", "Вивести результати"))
            #self.cameraOnlineLabel.setText(_translate("MainWindow", "Camera
online:"))
            self.toolBar.setWindowTitle(_translate("MainWindow", "toolBar"))

        # TODO add some code
        def StartMonitoring(self):
            isEnterMode = self.enterCheckBox.isChecked()
            self.OpenCvManager.StartMonitoring(self.DBManager, isEnterMode)

        def OpenRegisterStudentWindow(self):
            self.registerWindow = QtWidgets.QMainWindow()
            self.registerWindowUI = Ui_RegisterWindow()
            self.registerWindowUI.setupUi(self.registerWindow, self.OpenCvManager,
self.DBManager)
            self.registerWindow.show()
            #MainWindow.setDisabled(True)

        def OpenResultListWindow(self):
            self.listResultWindow = QtWidgets.QMainWindow()
            self.listResultWindowUI = Ui_ShowListWindow()
            self.listResultWindowUI.setupUi(self.listResultWindow, self.DBManager)
            self.listResultWindow.show()
```

```
class RegisterWindow():
    def __init__(self):
        super.__init__()

    #def initRegisterUI(self):



if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    sys.exit(app.exec_())
```

# ДОДАТОК 2. ФАЙЛ REGISTERMENU.PY

```
   # -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'newregisterwindow.ui'
#
# Created by: PyQt5 UI code generator 5.14.0
#
# WARNING! All changes made in this file will be lost!
import random

from PyQt5 import QtCore, QtGui, QtWidgets
from RegisterMenuManager import RegisterMenuManager
from Student import Student
import uuid
import DataBase

class Ui_RegisterWindow(object):
    def setupUi(self, MainWindow, OpenCvManager, DBManager):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(620, 230)

        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.verticalLayout_2 = QtWidgets.QVBoxLayout(self.centralwidget)
        self.verticalLayout_2.setObjectName("verticalLayout_2")

        self.titleLabel = QtWidgets.QLabel(self.centralwidget)
        self.titleLabel.setText("---Вікно реєстрації---")
        self.titleLabel.setAlignment(QtCore.Qt.AlignCenter)
        self.verticalLayout_2.addWidget(self.titleLabel)

        self.comboBox = QtWidgets.QComboBox(self.centralwidget)
        self.comboBox.setObjectName("comboBox")
        self.comboBox.addItem("Працівник")
        self.comboBox.addItem("Викладач")
        self.comboBox.addItem("Студент")
        self.verticalLayout_2.addWidget(self.comboBox)
        self.verticalLayout = QtWidgets.QVBoxLayout()
        self.verticalLayout.setObjectName("verticalLayout")
        self.label = QtWidgets.QLabel(self.centralwidget)
```

```
                self.label.setObjectName("label")
                self.verticalLayout.addWidget(self.label)
                self.horizontalLayout = QtWidgets.QHBoxLayout()
                self.horizontalLayout.setObjectName("horizontalLayout")
                self.label_2 = QtWidgets.QLabel(self.centralwidget)
                self.label_2.setObjectName("label_2")
                self.horizontalLayout.addWidget(self.label_2)
                self.firstNameLine = QtWidgets.QLineEdit(self.centralwidget)
                self.firstNameLine.setObjectName("firstNameLine")
                self.horizontalLayout.addWidget(self.firstNameLine)
                self.verticalLayout.addLayout(self.horizontalLayout)
                self.horizontalLayout_2 = QtWidgets.QHBoxLayout()
                self.horizontalLayout_2.setObjectName("horizontalLayout_2")
                self.label_3 = QtWidgets.QLabel(self.centralwidget)
                self.label_3.setObjectName("label_3")
                self.horizontalLayout_2.addWidget(self.label_3)
                self.secondNameLine = QtWidgets.QLineEdit(self.centralwidget)
                self.secondNameLine.setText("")
                self.secondNameLine.setObjectName("secondNameLine")
                self.horizontalLayout_2.addWidget(self.secondNameLine)
                self.verticalLayout.addLayout(self.horizontalLayout_2)
                self.horizontalLayout_3 = QtWidgets.QHBoxLayout()
                self.horizontalLayout_3.setObjectName("horizontalLayout_3")
                self.label_4 = QtWidgets.QLabel(self.centralwidget)
                self.label_4.setObjectName("label_4")
                self.horizontalLayout_3.addWidget(self.label_4)
                self.instituteLine = QtWidgets.QLineEdit(self.centralwidget)
                self.instituteLine.setText("")
                self.instituteLine.setObjectName("instituteLine")
                self.horizontalLayout_3.addWidget(self.instituteLine)
                self.verticalLayout.addLayout(self.horizontalLayout_3)
                self.verticalLayout_2.addLayout(self.verticalLayout)
                spacerItem = QtWidgets.QSpacerItem(20, 40, QtWidgets.QSizePolicy.Minimum,
QtWidgets.QSizePolicy.Expanding)
                self.verticalLayout_2.addItem(spacerItem)
                self.registerButton = QtWidgets.QPushButton(self.centralwidget)
                self.registerButton.setObjectName("registerButton")
                self.verticalLayout_2.addWidget(self.registerButton)
                MainWindow.setCentralWidget(self.centralwidget)
                self.menubar = QtWidgets.QMenuBar(MainWindow)
                self.menubar.setGeometry(QtCore.QRect(0, 0, 620, 17))
                self.menubar.setObjectName("menubar")
                MainWindow.setMenuBar(self.menubar)
                self.statusbar = QtWidgets.QStatusBar(MainWindow)
                self.statusbar.setObjectName("statusbar")
                MainWindow.setStatusBar(self.statusbar)
                self.toolBar = QtWidgets.QToolBar(MainWindow)
                self.toolBar.setObjectName("toolBar")
                MainWindow.addToolBar(QtCore.Qt.TopToolBarArea, self.toolBar)

                # connect to slots
                self.registerButton.clicked.connect(self.Register)

                self.comboBox.currentTextChanged.connect(self.ChangeUi)

                self.retranslateUi(MainWindow)
                QtCore.QMetaObject.connectSlotsByName(MainWindow)

                self.OpenCvManager = OpenCvManager
                self.manager = RegisterMenuManager()
```

```
            self.dataBaseManager = DBManager


        def retranslateUi(self, MainWindow):
            _translate = QtCore.QCoreApplication.translate
            MainWindow.setWindowTitle(_translate("MainWindow",
"RegisterStudentWindow"))
            self.label.setText(_translate("MainWindow", "Будь ласка заповніть форму
реєстрації\nПоля із * є обов\'язковими"))
            self.label_2.setText(_translate("MainWindow", "* Ім\'я"))
            self.firstNameLine.setPlaceholderText(_translate("MainWindow", "Петро"))
            self.label_3.setText(_translate("MainWindow", "* Прізвище"))
            self.secondNameLine.setPlaceholderText(_translate("MainWindow",
"Шухевич"))
            self.label_4.setText(_translate("MainWindow", "Інститут"))
            self.instituteLine.setPlaceholderText(_translate("MainWindow", "ІКНІ"))
            self.registerButton.setText(_translate("MainWindow", "Занести у базу
даних"))
            self.toolBar.setWindowTitle(_translate("MainWindow", "toolBar"))

        def Register(self):
            currentItem = self.comboBox.currentText()
            isSucessfullyRegistered = False
            if currentItem == "Викладач":
                while not isSucessfullyRegistered:
                    isSucessfullyRegistered = self.ReadDataFromTeacherField()
                    pass
            elif currentItem == "Працівник":
                while not isSucessfullyRegistered:
                    isSucessfullyRegistered=self.ReadDataFromWorkerField()
                    pass
            elif currentItem == "Студент":
                while not isSucessfullyRegistered:
                    isSucessfullyRegistered=self.ReadDataFromStudentField()
                    pass
            self.manager.CalculateFolerPath(self.firstNameLine.text(),
self.secondNameLine.text())
            self.OpenCvManager.TakeAPhoto(self.manager.GetFolderPath())
            self.OpenCvManager.TrainPhotos(self.dataBaseManager)


        def ReadDataFromWorkerField(self):
            try:
                self.dataBaseManager.AddWorkerToDB(pid=random.randint(0, 999999),
                                                   name=self.firstNameLine.text(),

surname=self.secondNameLine.text(),

department=self.instituteLine.text())
                return True
            except:
                print("Cannot register person in database. Please try again")
                return  False



        def ReadDataFromStudentField(self):
            try:
```

```
                self.dataBaseManager.AddStudentToDB(pid=random.randint(0, 999999),
                                                    name=self.firstNameLine.text(),
surname=self.secondNameLine.text(),
                                                    group=self.groupLine.text())
                self.instituteLine.text()
                return True
            except:
                print("Cannot register person in database. Please try again")
                return False

        def ReadDataFromTeacherField(self):
            try:
                self.dataBaseManager.AddTeacherToDB(pid=random.randint(0, 999999),
                                                    name=self.firstNameLine.text(),
surname=self.secondNameLine.text(),
department=self.instituteLine.text(),
profession=self.professionLine.text())
                return True
            except:
                print("Cannot register person in database. Please try again")
                return False

        def ChangeUi(self):
            currentItem = self.comboBox.currentText()
            if currentItem == "Викладач":
                try:
                    self.DeleteStudentUI()
                except:
                    print("There isnt student ui")
                self.SetTeacherUi()
            elif currentItem == "Працівник":
                try:
                    self.DeleteTeacherUI()
                except:
                    print("There isnt Teacher UI")
                try:
                    self.DeleteStudentUI()
                except:
                    print("There isnt student ui")
            elif currentItem == "Студент":
                try:
                    self.DeleteTeacherUI()
                except:
                    print("There isnt Teacher UI")
                self.SetStudentUi()

        def SetTeacherUi(self):
            self.horizontalLayout_4 = QtWidgets.QHBoxLayout()
            self.horizontalLayout_4.setObjectName("horizontalLayout_4")
            self.label_5 = QtWidgets.QLabel(self.centralwidget)
            self.label_5.setObjectName("label_5")
            self.horizontalLayout_4.addWidget(self.label_5)
            self.professionLine = QtWidgets.QLineEdit(self.centralwidget)
            self.professionLine.setText("")
            self.professionLine.setObjectName("professionLine")
            self.professionLine.setPlaceholderText("Професор")
```

```
        self.horizontalLayout_4.addWidget(self.professionLine)
        self.verticalLayout.addLayout(self.horizontalLayout_4)
        self.label_5.setText("Посада")

    def SetStudentUi(self):
        self.horizontalLayout_5 = QtWidgets.QHBoxLayout()
        self.horizontalLayout_5.setObjectName("horizontalLayout_5")
        self.label_6 = QtWidgets.QLabel(self.centralwidget)
        self.label_6.setObjectName("label_6")
        self.horizontalLayout_5.addWidget(self.label_6)
        self.groupLine = QtWidgets.QLineEdit(self.centralwidget)
        self.groupLine.setText("")
        self.groupLine.setObjectName("groupLine")
        self.groupLine.setPlaceholderText("KH-412")
        self.horizontalLayout_5.addWidget(self.groupLine)
        self.verticalLayout.addLayout(self.horizontalLayout_5)
        self.label_6.setText("Група")

    def DeleteStudentUI(self):
        try:
            self.horizontalLayout_5.deleteLater()
            self.label_6.deleteLater()
            self.groupLine.deleteLater()
            self.verticalLayout.removeItem(self.horizontalLayout_5)
        finally:
            pass

    def DeleteTeacherUI(self):
        try:
            self.horizontalLayout_4.deleteLater()
            self.label_5.deleteLater()
            self.professionLine.deleteLater()
            self.verticalLayout.removeItem(self.horizontalLayout_4)
        finally:
            pass

if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_RegisterWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    sys.exit(app.exec_())
```

## ДОДАТОК 3. ФАЙЛ SHOWLISTMENU.PY

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'newregisterwindow.ui'
#
# Created by: PyQt5 UI code generator 5.14.0
#
# WARNING! All changes made in this file will be lost!


from PyQt5 import QtCore, QtGui, QtWidgets
from ResultTableUi import Ui_TableResultWindow

class Ui_ShowListWindow(object):
```

```
        def setupUi(self, MainWindow, DataBaseManager):
            MainWindow.setObjectName("MainWindow")
            MainWindow.resize(620, 292)
            self.centralwidget = QtWidgets.QWidget(MainWindow)
            self.centralwidget.setObjectName("centralwidget")
            self.verticalLayout_2 = QtWidgets.QVBoxLayout(self.centralwidget)
            self.verticalLayout_2.setObjectName("verticalLayout_2")
            self.verticalLayout = QtWidgets.QVBoxLayout()
            self.verticalLayout.setObjectName("verticalLayout")

            self.titleLabel = QtWidgets.QLabel(self.centralwidget)
            self.titleLabel.setText("---Вікно формування запиту---")
            self.titleLabel.setAlignment(QtCore.Qt.AlignCenter)
            self.verticalLayout_2.addWidget(self.titleLabel)

            self.label = QtWidgets.QLabel(self.centralwidget)
            self.label.setObjectName("label")
            self.verticalLayout.addWidget(self.label)

            self.dateEdit = QtWidgets.QDateEdit(self.centralwidget)
            self.dateEdit.setObjectName("dateEdit")
            self.verticalLayout.addWidget(self.dateEdit)
            self.label_3 = QtWidgets.QLabel(self.centralwidget)
            self.label_3.setObjectName("label_3")
            self.verticalLayout.addWidget(self.label_3)
            self.horizontalLayout_2 = QtWidgets.QHBoxLayout()
            self.horizontalLayout_2.setObjectName("horizontalLayout_2")
            self.timeEditBefore = QtWidgets.QTimeEdit(self.centralwidget)
            self.timeEditBefore.setObjectName("timeEdit")
            self.horizontalLayout_2.addWidget(self.timeEditBefore)
            self.timeEditAfter = QtWidgets.QTimeEdit(self.centralwidget)
            self.timeEditAfter.setObjectName("timeEdit_2")
            self.horizontalLayout_2.addWidget(self.timeEditAfter)
            self.verticalLayout.addLayout(self.horizontalLayout_2)
            self.horizontalLayout_3 = QtWidgets.QHBoxLayout()
            self.horizontalLayout_3.setObjectName("horizontalLayout_3")
            self.auditory = QtWidgets.QLabel(self.centralwidget)
            self.auditory.setObjectName("auditory")
            self.horizontalLayout_3.addWidget(self.auditory)
            self.auditoryLine = QtWidgets.QLineEdit(self.centralwidget)
            self.auditoryLine.setText("")
            self.auditoryLine.setObjectName("auditoryLine")
            self.horizontalLayout_3.addWidget(self.auditoryLine)
            self.verticalLayout.addLayout(self.horizontalLayout_3)
            self.verticalLayout_2.addLayout(self.verticalLayout)
            self.showListButton = QtWidgets.QPushButton(self.centralwidget)
            self.showListButton.setObjectName("showListButton")
            self.verticalLayout_2.addWidget(self.showListButton)
            spacerItem = QtWidgets.QSpacerItem(20, 40, QtWidgets.QSizePolicy.Minimum,
QtWidgets.QSizePolicy.Expanding)
            self.verticalLayout_2.addItem(spacerItem)
            MainWindow.setCentralWidget(self.centralwidget)
            self.menubar = QtWidgets.QMenuBar(MainWindow)
            self.menubar.setGeometry(QtCore.QRect(0, 0, 620, 17))
            self.menubar.setObjectName("menubar")
            MainWindow.setMenuBar(self.menubar)
            self.statusbar = QtWidgets.QStatusBar(MainWindow)
            self.statusbar.setObjectName("statusbar")
            MainWindow.setStatusBar(self.statusbar)
            self.toolBar = QtWidgets.QToolBar(MainWindow)
```

```
            self.toolBar.setObjectName("toolBar")
            MainWindow.addToolBar(QtCore.Qt.TopToolBarArea, self.toolBar)

            # connect to buttons
            self.showListButton.clicked.connect(self.OpenTableWindow)
            self.retranslateUi(MainWindow)

            self.DBManager = DataBaseManager

            QtCore.QMetaObject.connectSlotsByName(MainWindow)

        def retranslateUi(self, MainWindow):
            _translate = QtCore.QCoreApplication.translate
            MainWindow.setWindowTitle(_translate("MainWindow", "QueryMenu"))
            self.label.setText(_translate("MainWindow",  "Виберіть  бажану  дату  та
час"))
            self.auditory.setText(_translate("MainWindow", "Аудиторія"))
            self.label_3.setText(_translate("MainWindow", "Виберіть часові рамки"))
            #self.auditoryLine.setPlaceholderText(_translate("MainWindow",      "e.g.
IKNI"))
            self.showListButton.setText(_translate("MainWindow",   "Показати   список
присутніх людей"))
            self.toolBar.setWindowTitle(_translate("MainWindow", "toolBar"))

        def PrintPersons(self):
            # delete
            try:
                self.tableWidget.deleteLater()
                self.verticalLayout.removeItem(self.tableWidget)
            except:
                print("No table to delete")

            listOfPids                                                          =
self.DBManager.GetCurrentVisitorsList(self.dateTimeEdit.date(),
self.dateTimeEdit.time())
            self.tableWidget = QtWidgets.QTableWidget()
            self.tableWidget.setRowCount(len(listOfPids))
            self.tableWidget.setColumnCount(2)
            self.tableWidget.setHorizontalHeaderLabels(("Ім\'я", "Прізвище"))
            header = self.tableWidget.horizontalHeader()
            header.setSectionResizeMode(QtWidgets.QHeaderView.ResizeToContents)
            header.setSectionResizeMode(0, QtWidgets.QHeaderView.Stretch)
            header.setSectionResizeMode(1, QtWidgets.QHeaderView.Stretch)
            #header.setSectionResizeMode(2, QtWidgets.QHeaderView.Stretch)
            print(listOfPids)
            for obj in listOfPids:
                print(obj)
                for j in range(len(obj)):
                    print(listOfPids.index(obj), j, obj[j])
                    self.tableWidget.setItem(listOfPids.index(obj),               j,
QtWidgets.QTableWidgetItem(obj[j]))
            self.verticalLayout.addWidget(self.tableWidget)


        def OpenTableWindow(self):
            resultTableList                                                      =
self.DBManager.GetCurrentVisitorsList(self.dateEdit.date(),

self.timeEditBefore.time(), self.timeEditAfter.time())
            self.tableResultWindow = QtWidgets.QMainWindow()
```

```
            self.tableResultWindowUI = Ui_TableResultWindow()
            self.tableResultWindowUI.setupUi(MainWindow=self.tableResultWindow,
listOfPersons=resultTableList)
            self.tableResultWindow.show()
```

# ДОДАТОК 4. ФАЙЛ RESULTTABLEUI.PY

```
    # -*- coding: utf-8 -*-

    # Form implementation generated from reading ui file 'ResultTable.ui'
    #
    # Created by: PyQt5 UI code generator 5.14.0
    #
    # WARNING! All changes made in this file will be lost!


    from PyQt5 import QtCore, QtGui, QtWidgets


    class Ui_TableResultWindow(object):
        def setupUi(self, MainWindow, listOfPersons):
            MainWindow.setObjectName("MainWindow")
            MainWindow.resize(620, 254)
            self.centralwidget = QtWidgets.QWidget(MainWindow)
            self.centralwidget.setObjectName("centralwidget")
            self.gridLayout = QtWidgets.QGridLayout(self.centralwidget)
            self.gridLayout.setObjectName("gridLayout")
            self.verticalLayout = QtWidgets.QVBoxLayout()
            self.verticalLayout.setObjectName("verticalLayout")
            self.label = QtWidgets.QLabel(self.centralwidget)
            self.label.setAlignment(QtCore.Qt.AlignCenter)
            self.label.setObjectName("label")
            self.verticalLayout.addWidget(self.label)
            self.tableWidget = QtWidgets.QTableWidget(self.centralwidget)
            self.tableWidget.setObjectName("tableWidget")
            self.tableWidget.setColumnCount(0)
            self.tableWidget.setRowCount(0)
            self.verticalLayout.addWidget(self.tableWidget)
            self.gridLayout.addLayout(self.verticalLayout, 0, 0, 1, 1)
            MainWindow.setCentralWidget(self.centralwidget)
            self.menubar = QtWidgets.QMenuBar(MainWindow)
            self.menubar.setGeometry(QtCore.QRect(0, 0, 620, 17))
            self.menubar.setObjectName("menubar")
            MainWindow.setMenuBar(self.menubar)
            self.statusbar = QtWidgets.QStatusBar(MainWindow)
            self.statusbar.setObjectName("statusbar")
            MainWindow.setStatusBar(self.statusbar)
            self.toolBar = QtWidgets.QToolBar(MainWindow)
            self.toolBar.setObjectName("toolBar")
            MainWindow.addToolBar(QtCore.Qt.TopToolBarArea, self.toolBar)

            self.tableWidget.setRowCount(len(listOfPersons))
            self.tableWidget.setColumnCount(5)
            self.tableWidget.setHorizontalHeaderLabels(("Ім\'я", "Прізвище", "Дата",
"Час входу", "Час виходу"))
            header = self.tableWidget.horizontalHeader()
            header.setSectionResizeMode(QtWidgets.QHeaderView.ResizeToContents)
            header.setSectionResizeMode(0, QtWidgets.QHeaderView.Stretch)
            header.setSectionResizeMode(1, QtWidgets.QHeaderView.Stretch)
            header.setSectionResizeMode(2, QtWidgets.QHeaderView.Stretch)
```

```
            header.setSectionResizeMode(3, QtWidgets.QHeaderView.Stretch)
            header.setSectionResizeMode(4, QtWidgets.QHeaderView.Stretch)
            print(listOfPersons)
            correctViewResultTable = list()
            for obj in listOfPersons:
                name = obj[1]
                surname = obj[2]
                date = obj[3].date()
                enterTime = obj[3].time()
                try:
                    exitTime = obj[4].time()
                except:
                    exitTime = '-'
                correctViewResultTable.append((name,   surname,   date,   enterTime,
exitTime))

            for obj in correctViewResultTable:
                for j in range(len(obj)):
                    self.tableWidget.setItem(correctViewResultTable.index(obj),   j,
QtWidgets.QTableWidgetItem(str(obj[j])))


            self.retranslateUi(MainWindow)
            QtCore.QMetaObject.connectSlotsByName(MainWindow)

        def retranslateUi(self, MainWindow):
            _translate = QtCore.QCoreApplication.translate
            MainWindow.setWindowTitle(_translate("MainWindow",
"RegisterStudentWindow"))
            self.label.setText(_translate("MainWindow",   "---Вікно   зареєстрованих
осіб---"))
            self.toolBar.setWindowTitle(_translate("MainWindow", "toolBar"))
```

# ДОДАТОК 5. ФАЙЛ DATABASE.PY

```
    import datetime
    import pypyodbc

    class DataBase():
        def __init__(self):
            self.sqlServerName = "DESKTOP-02ADUN4\\SQLEXPRESS01"
            self.dataBaseName = "WorkerRegistterDB"
            self.connection = pypyodbc.connect('Driver={SQL Server};'
                                               'Server=' + self.sqlServerName + ';'

'Database=' + self.dataBaseName + ';')
            self.cursor = self.connection.cursor()

        def AddPersonToDB(self, pid, name, surname):
            sqlQuery = ("""
                        insert Persons
                        (person_id, first_name, last_name)
                        values
                        (?, ?, ?)
                        """)
            self.cursor.execute(sqlQuery, (pid, name, surname))
            self.connection.commit()

        def AddStudentToDB(self, pid, name, surname, group):
            self.AddPersonToDB(pid, name, surname)
```

```
    sqlQuery = ("""
            insert Students
            (student_id, group_name)
            values
            (?, ?)
            """)
    self.cursor.execute(sqlQuery, (pid, group))
    self.connection.commit()

def AddTeacherToDB(self, pid, name, surname, department, profession):
    self.AddPersonToDB(pid, name, surname)
    sqlQuery = ("""
            insert Teachers
            (teacher_id, department, profession)
            values
            (?, ?, ?)
            """)
    self.cursor.execute(sqlQuery, (pid, department, profession))
    self.connection.commit()

def AddWorkerToDB(self, pid, name, surname, department):
    self.AddPersonToDB(pid, name, surname)
    sqlQuery = ("""
            insert Worker
            (worker_id, department)
            values
            (?, ?)
            """)
    self.cursor.execute(sqlQuery, (pid, department))
    self.connection.commit()

def AddPersonEnter(self, pid):
    sqlQuery = ("""
                insert Enters
                (person_id, room_id, enter_date)
                values
                (?, ?, GETDATE())
                """)
    self.cursor.execute(sqlQuery, (pid, 1))
    self.connection.commit()

def AddPersonExit(self, pid):
    sqlQuery = ("""
                insert Exits
                (person_id, room_id, exit_date)
                values
                (?, ?, GETDATE())
                """)
    self.cursor.execute(sqlQuery, (pid, 1))
    self.connection.commit()

def CheckIfPersonIsInEnters(self, pid):
    sqlQuery = ("""
                select * from Enters where person_id=?
                """)
    pidList = (pid,)
    self.cursor.execute(sqlQuery, pidList)
    results = self.cursor.fetchall()
    if results:
        return True
```

```
        else:
            return False

    def GetListOfPersons(self):
        sqlQuery = ("""
                    select * from Persons
                """)
        self.cursor.execute(sqlQuery)
        results = self.cursor.fetchall()
        print(results)
        return results

    def CheckIfPersonIsInExits(self, pid):
        sqlQuery = ("""
                            select * from Exits where person_id=?
                            """)
        pidList = (pid,)
        self.cursor.execute(sqlQuery, pidList)
        results = self.cursor.fetchall()
        if results:
            return True
        else:
            return False

    def IsValidEnter(self, pid):
        sqlQuery = ("""
                    select * from Exits where
                    exit_date>(select enter_date from Enters
                    where enter_id=(select max(enter_id) from Enters
                    where person_id=?))
                    """)
        pidList = (pid,)
        self.cursor.execute(sqlQuery, pidList)
        results = self.cursor.fetchall()
        print("Result table of IsValidEnter" + str(results))
        if results:
            return True
        else:
            return False

    def IsValidExit(self, pid):
        sqlQuery = ("""select * from Enters where
            enter_date>(select exit_date from Exits
            where exit_id=(select max(exit_id) from Exits
            where person_id=?))""")
        pidList = (pid,)
        self.cursor.execute(sqlQuery, pidList)
        results = self.cursor.fetchall()
        print("Result table of IsValidExit" + str(results))
        if results:
            return True
        else:
            return False

    def FixPersonsEnter(self, pid):
        if not self.CheckIfPersonIsInEnters(pid) or self.IsValidEnter(pid):
            self.AddPersonEnter(pid)
            nameAndSurname = self.GetPersonNames(pid)
            self.AddToResultEntersExits(nameAndSurname[0])
```

```
        def FixPersonsExit(self, pid):
            if not self.CheckIfPersonIsInExits(pid) or self.IsValidExit(pid):
                self.AddPersonExit(pid)
                nameAndSurname = self.GetPersonNames(pid)
                self.AppendToResultEntersExits(nameAndSurname[0])

        def GetCurrentVisitorsList(self, date, timeBefore, timeAfter):
            dateTimeStringBefore = str(date.year()) + "-" + str(date.month()) + \
                                   "-" + str(date.day()) + " " + str(timeBefore.hour()) \
                                   +    ":"    +    str(timeBefore.minute())    +    ":"    +
str(timeBefore.second())
            dateTimeStringAfter = str(date.year()) + "-" + str(date.month()) + \
                                    "-" + str(date.day()) + " " + str(timeAfter.hour())
\
                                    +   ":"   +   str(timeAfter.minute())   +   ":"   +
str(timeAfter.second())
            print("Before date = " + dateTimeStringBefore)
            print("After date = " + dateTimeStringAfter)
            #sqlQuery = ("""
            #select person_id from Enters where enter_date>?
            #""")
            sqlQuery = ("""
            select * from ResultEntersExits where (enter_date > ? and exit_date < ?)
or (enter_date > ? )""")
            self.cursor.execute(sqlQuery, (dateTimeStringBefore, dateTimeStringAfter,
dateTimeStringBefore))
            results = self.cursor.fetchall()
            print("Result query table")
            print(results)
            return results

        def GetPersonNames(self, pid):
            sqlQuery = ("""
                    select first_name, last_name
                    from Persons where person_id = ?
                            """)
            pidList = (pid,)
            self.cursor.execute(sqlQuery, pidList)
            results = self.cursor.fetchall()
            print(results)
            return results

        def AddToResultEntersExits(self, nameAndSurname):
            sqlQuery = ("""
            insert ResultEntersExits
            (first_name, last_name, enter_date)
            values
            (?, ?, GETDATE())""")
            self.cursor.execute(sqlQuery, (nameAndSurname[0], nameAndSurname[1]))
            self.connection.commit()

        def AppendToResultEntersExits(self, nameAndSurname):
            sqlQuery = ("""
            update ResultEntersExits
            set exit_date = GETDATE()
            where table_id=(select max(table_id) from ResultEntersExits
            where first_name=? and last_name=?)
            """)
            self.cursor.execute(sqlQuery, (nameAndSurname[0], nameAndSurname[1]))
            self.connection.commit()
```

```
if __name__ == "__main__":
    db = DataBase()
    db.GetListOfPersons()

    #db.GetPersonNames(676251)
```

## ДОДАТОК 6. ФАЙЛ OPENCVMANAGER.PY

```
from builtins import print

import cv2
from cv2 import *
import pickle
import os
import uuid

import numpy
from PIL import Image


# TODO add mutex to recognizer

class OpenCvManager:
    def __init__(self):
        self.face_cascade                                                    =
cv2.CascadeClassifier('cascades/data/haarcascade_frontalface_alt2.xml')
        self.BASE_DIR  =  os.path.dirname(os.path.abspath(__file__))    #  project
directory
        self.IMAGES_DIR = os.path.join(self.BASE_DIR, 'images')  # directory of
images

    def LoadNameLabels(self):
        try:
            with open("labels.pickle", 'rb') as f:
                og_labels = pickle.load(f)
                return og_labels
        except IOError:
            labels = {}
            return labels


    def ReverseLabels(self, og_labels):
        labels = {v: k for k, v in og_labels.items()}
        return labels

    def CreateRecognizer(self):
        recognizer = cv2.face.LBPHFaceRecognizer_create()
        return recognizer

    def ReadFromFileToRecognizer(self, recognizer):
        recognizer.read("train.yml")

    def StartMonitoring(self, DBManager, isEnterMode):
        cap = cv2.VideoCapture(0)
        recognizer = self.CreateRecognizer()
        self.ReadFromFileToRecognizer(recognizer)
        while True:
            ret, frame = cap.read()
            # convert frame to gray color
```

```
                gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
                faces  =  self.face_cascade.detectMultiScale(gray,  scaleFactor=1.5,
minNeighbors=5)
                for (x, y, w, h) in faces:
                    # contains faces on the screen
                    roi_gray = gray[y:y + h, x:x + w]
                    roi_color = frame[y:y + h, x:x + w]

                    # recognize
                    # TODO if TimeToLive is not new read from recognizer
                    id_, conf = recognizer.predict(roi_gray)
                    if conf >= 45 and conf <= 85:
                        print("COnf = " + str(conf))
                        labels = self.ReverseLabels(self.LoadNameLabels())

                        print(id_)
                        if isEnterMode:
                            DBManager.FixPersonsEnter(id_)
                        else:
                            DBManager.FixPersonsExit(id_)

                        label = labels[id_]
                        self.SetTextAroundFace(frame, x, y, id_, label)
                        self.DrawRectangleAroundFaces(frame, x, y, w, h)

                    # todo write roi into file

                cv2.imshow('Video', frame)
                if cv2.waitKey(20) & 0xFF == ord('q'):
                    cv2.destroyAllWindows()
                    break

        def DrawRectangleAroundFaces(self, frame, x, y, w, h):
            color = (0, 255, 0)     # green color
            stroke = 2
            end_cord_x = x + w
            end_cord_y = y + h
            cv2.rectangle(frame, (x, y), (end_cord_x, end_cord_y), color, stroke)

        def SetTextAroundFace(self, frame, x, y, id_, name):
            font = cv2.FONT_HERSHEY_SIMPLEX
            color = (255, 255, 255)
            stroke = 2
            cv2.putText(frame, name, (x,y), font, 1, color, stroke, cv2.LINE_AA)

        def TrainPhotos(self, dataBase):
            recognizer = self.CreateRecognizer()
            # dictionary " name : id "
            label_ids = self.LoadNameLabels()
            print(label_ids)
            listOfStudents = dataBase.GetListOfPersons()
            #listOfStudents = dataBase.DeserializeDataBase()
            # 2 dimential array of detected faces
            x_train = []
            y_labels = []
            for root, dirs, files in os.walk(self.IMAGES_DIR):
                for file in files:
                    if file.endswith("png") or file.endswith("jpg"):
                        path = os.path.join(root, file)
                        label = os.path.basename(root).replace(" ", "_").lower()
```

```
                        if not label in label_ids:
                            current_id = self.GetUuidByName(label, listOfStudents)
                            #current_id = int(clearstruui, base=16)
                            print(current_id)
                            label_ids[label] = current_id
                        # set current id
                        id_ = label_ids[label]
                        # some numbers of persons
                        pil_image = Image.open(path).convert("L")  # grayscale
                        # convert to array of numpy
                        image_array = numpy.array(pil_image, "uint8")
                        # detect faces on the frame
                        faces    =    self.face_cascade.detectMultiScale(image_array,
scaleFactor=1.5, minNeighbors=5)

                        for x, y, w, h in faces:
                            # region of interests
                            print("Faces was found")
                            roi = image_array[y:y + h, x:x + w]
                            # face
                            x_train.append(roi)
                            # ids
                            y_labels.append(id_)

            # save face names
            with open("labels.pickle", "wb") as f:
                pickle.dump(label_ids, f)

            # train data and save it to file
            recognizer.train(x_train, numpy.array(y_labels))
            recognizer.save("train.yml")

        def GetUuidByName(self, label, listofstudents):
            # todo hanle if is not matches
            # 0 - pid, 1 - name, 2 - surname
            for x in listofstudents:
                if x[1].lower() + x[2].lower() == label:
                    return x[0]
            # matches = next(item for item in listofstudents if item["name"].lower()
+ item["surname"].lower() == label)
            # print("Matches = " + matches["pid"])
            print("GetUuidByName crashed because it was no same label in list of
students")
            return 0

        def TakeAPhoto(self, folderPath):
            cam = VideoCapture(0, cv2.CAP_DSHOW)  # 0 -> index of camera
            amountOfFaces=0
            while True:
                s, img = cam.read()
                if s:  # frame captured without any errors
                    # check if person already exists
                    if not os.path.exists(folderPath):
                        os.mkdir(folderPath)
                    # TODO don't work with unicode ((
                    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
                    faces = self.face_cascade.detectMultiScale(gray, scaleFactor=1.5,
minNeighbors=5)
                    print(faces)
                    if len(faces) is not 0:
```

```
                        print("Photo is done")
                        imwrite(os.path.join(folderPath, str(uuid.uuid4()) + '.png'),
img)
                        amountOfFaces = amountOfFaces + 1
                        if amountOfFaces == 10:
                            print("All 10 photos are done")
                            break
            cam.release()
            cv2.destroyAllWindows()
```

## ДОДАТОК 7. ФАЙЛ REGISTERMENUMANAGER.PY

```
from cv2 import *
from Student import Student
import uuid
import StudentDataBaseProcessor

class RegisterMenuManager:
    def __init__(self):
        # TODO move in constants
        self.__images_path = os.path.join(os.getcwd(), 'images')
        self.__dbproccessor = StudentDataBaseProcessor.DataBaseProcessor()
        self.__folderPath = ''

    def RegisterInDataBase(self, OpenCvManager, student):
        # Add in list of students
        self.__dbproccessor.AppendStudent(student)
        #self.__dbproccessor.AppendDataBase()
        OpenCvManager.TrainPhotos(self.__dbproccessor)

    def GetFolderPath(self):
        return self.__folderPath

    def CalculateFolerPath(self, name, surname):
        folderName = name.lower()+surname.lower()
        self.__folderPath = os.path.join(self.__images_path, folderName)
```

## ДОДАТОК 8. ФАЙЛ SQLQUERIES.SQL

```
Create table Persons
(
person_id int primary key  NOT NULL,
first_name varchar(20) not null,
last_name varchar(20) not null
)
go

alter table Persons
add
primary key (person_id)
go

create table Students
(
student_id int primary key references Persons (person_id),
group_name varchar(10) not null,
)

create table Worker
```

```
(
worker_id int primary key references Persons (person_id),
department varchar(50) not null
)

create table Teachers
(
teacher_id int primary key references Persons (person_id),
department varchar(50) not null,
profession varchar(30) not null
)
go

create table Rooms
(
room_id int primary key NOT NULL,
building varchar(20) not null,
room varchar(20) not null
)
go


create table Enters
(
enter_id int primary key identity,
person_id int foreign key references Persons(person_id),
room_id int foreign key references Rooms(room_id),
enter_date datetime
)
go

create table Exits
(
exit_id int primary key identity,
person_id int foreign key references Persons(person_id),
room_id int foreign key references Rooms(room_id),
exit_date datetime
)
go

create table ResultEntersExits
(
table_id int primary key identity,
first_name varchar(20) not null,
last_name varchar(20) not null,
enter_date datetime null,
exit_date datetime null
)
```