НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії_____
Кафедра комп'ютеризованих систем управління_____
Спеціальність 126 «Інформаційні системи та технології»_____
(шифр, найменування)

Освітньо-професійна програма «Інформаційні системи та технології»_____
Форма навчання денна_____

ЗАТВЕРДЖУЮ
Завідувач кафедри

___Литвиненко О.Є_____
«_____»_____202 __ р.

ДИПЛОМНИЙ ПРОЄКТ
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ЗДОБУВАЧА ОСВІТНЬОГО СТУПЕНЯ

"БАКАЛАВР"

Тема: "Програмно-технологічне рішення для прискорення медичних перевірок пасажирів в аеропортах під час карантинних заходів."

Виконавець: Сковпень Данило Васильович

Керівник: Проф. Віталій Юрійович Зубок

Консультанти з окремих розділів пояснювальної записки:
_____
_____
_____

Нормоконтролер: Тупота Є.В. _____

Київ 2022

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії

Кафедра інженерії програмного забезпечення

Освітній ступінь бакалавр

Спеціальність 126 Інформаційні системи та технології

Освітньо-професійна програма Інормаційні системи та технології

ЗАТВЕРДЖУЮ

Завідувач кафедри

Литвиненко О.Є.

"___" _____ 2022 р

ЗАВДАННЯ

на виконання дипломного проєкта студента

Сковпня Данила Васильовича

1. Тема проєкту: «Програмно-технологічне рішення для прискорення медичних перевірок пасажирів в аеропортах під час карантинних заходів.»
затверджена наказом ректора від «04» квітня 2022 р. № 340
2. Термін виконання проєкту: з 16.05.2022 р. до 06.06.2022 р.
3. Вихідні данні до проєкту: програмний застосунок розроблений за допомогою середовища програмування PyCharm Community Edition.
4. Зміст пояснювальної записки:

1) Аналіз предметної області.
2) Вимоги до застосунку програмно-технологічне рішення для прискорення медичних перевірок пасажирів в аеропортах під час карантинних заходів.
3) Структура застосунку програмно-технологічне рішення для прискорення медичних перевірок пасажирів в аеропортах під час карантинних заходів.
4) Прототип застосунку програмно-технологічного рішення для прискорення медичних перевірок пасажирів в аеропортах під час карантинних заходів.

5. Перелік обов'язкових слайдів презентації:

1) Тема, виконавець, керівник.
2) Вимоги до застосунку.
3) Структура засобу.
4) Діаграма класів проекту.

5) Інтерфейс засобу.
6) Висновок.

6. Календарний план-графік.

№ пор	Завдання	Термін виконання	Відмітка про виконання
1.	Ознайомлення з постановкою задачі та вивчення літератури Написання 1 розділу, представлення керівнику	23.05.2022 – 24.05.2022	
2.	Написання 2 розділу, представлення керівнику	25.05.2021 – 26.05.2021	
3.	Написання 3 розділу, представлення керівнику	26.05.2021– 27.05.2021	
4.	Написання 4 розділу, представлення керівнику	27.05.22 – 02.06.22	
5.	Загальне редагування та друк пояснювальної записки, графічного матеріалу	04.06.22	
6.	Проходження нормо-контролю, перепліт пояснювальної записки.	06.06.22	
7.	Захист дипломного проекту	16.06.22	

7. Консультанти з окремих розділів

Розділ	Консультант (посада, П.І.Б.)	Дата, підпис	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання____12.05.2022 *початок проектування*
Керівник: _____Проф. Віталій Юрійович Зубок
Завдання прийняв до виконання: _____Сковпень Данило Васильович
Дата:_____12.05.2022

РЕФЕРАТ

Пояснювальна записка до дипломної роботи “Застосунок“ Програмно-технологічне рішення для прискорення медичних перевірок пасажирів в аеропортах під час карантинних заходів”.

ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ, ПРОГРАМУВАННЯ, ПРОГРАМНИЙ ЗАСІБ.

Об’єкт розробки – застосунок програмно-технологічного рішення для прискорення медичних перевірок пасажирів в аеропортах під час карантинних заходів.

Мета роботи – розробити програмно-технологічне рішення, яке допоможе прискорити медичні перевірки пасажирів в аеропортах під час карантинних заходів.

ABSTRACT

Explanatory note to the thesis “Application “Software and technological solution to speed up medical inspections of passengers at airports during quarantine activities”.

INFORMATION TECHNOLOGIES, PROGRAMMING, SOFTWARE.

Property development – application of software and technological solutions to speed up medical inspections of passengers at airports during quarantine activities.

Purpose – to develop a software and technology solution that will help speed up medical examinations of passengers at airports during quarantine activities.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНИ

ІТ – Інформаційні Технології

ОС – Операційна Система

ПЗ – Програмне Забезпечення

АЗ – Апаратне Забезпечення

UI – User Interface – Інтерфейс Користувача

БД – База Даних

IDE – Integrated Development Environment – Інтегроване Середовище Розробки

CPU – Central processing unit – Центральний процесор(ЦП)

GPU – Graphics Processing Unit – Графічний процесор(ГП)

ERP – Enterprise Resources Planning – система планування ресурсів підприємства

CRM – Customer Relationship Management – система управління взаємовідносинами з клієнтами

SCM – Supply Chain Management – система управління логістичним ланцюгом

PLM – Product Lifecycle Management – система управління життєвим циклом продукту

SRM – Supplier Relationship Management – система управління взаємовідносинами з постачальниками

BI – Business Intelligence – інтелектуальні системи підтримки стратегічного менеджменту

SQL(від англ. *Structured query language* — мова структурованих запитів) — декларативна мова програмування для взаємодії користувача з базами даних,

що застосовується формуванням запитів, оновленням та керуванням реляційними БД, створенням схем бази даних та їх модифікаціями, системами контролю за доступом до баз даних.

ВСТУП

Проживаючи у вік інформаційних технологій, коли інформаційні системи проникли практично в усі, чи вже повністю проникли в усі області життєдіяльності людини, ті, хто прагнуть залишатися в курсі подій з усього світу та на зв’язку з сім’єю та друзями, є лише один варіант - користування смартфонами, комп'ютерами та іншими електронними гаджетами, які мають доступ до мережі Інтернет. Так склалося, що для роботи ці пристрої потребують власне програмне забезпечення. Дякуючи швидкому розвиненню ІТ у XXI столітті, значна кількість дій, які придивилися лише тільки вручну, та об'єктів, які існували лише у фізичній формі, людство крок за кроком перейшло у цифровий простір. Сьогодні людині не потрібно тримати у себе величезну колекцію громіздких паперових книг, які збирають пил чи стояти у величезних чергах до банків та державних послуг. На даний час потрібно мати лише електронний пристрій у якого буде доступ Інтернету, і всі можливі книжки, пісні та витвори мистецтва у будь який час та у будь якій точці планети будуть доступні вам.

Попри розвиток інформаційних технологій в наш час, за останні три роки також стрімко росте кількість захворювань з приходом пандемії Covid-19 в усьому світі. Це в свою чергу, створює попит на автоматизацію послуг де є контакт людини з людиною. По всьому світу різні фармацевтичні та ІТ компанії борються з цим створюючи нові ліки, вакцини та технології.

Пандемія коронавірусної хвороби 2019 спричинена SARS-CoV-2. Спалах захворювання розпочався у грудні 2019 року у м. Ухань, Хубей, КНР, та визнаний ВООЗ пандемією 11 березня 2020 року. Станом на 20 березня 2022 року по світу захворіло 470 млн людей. Понад 6 млн померли. Одужали 400 млн.

13 липня 2020 року глава Всесвітньої організації охорони здоров'я Тедрос Адан Гебрейесус назвав чотири основні сценарії поширення коронавірусу в різних регіонах світу: перший сценарій реалізувався в

країнах, які були попереджені та обізнані про спалах хвороби — в результаті їм вдалося уникнути великих спалахів (деякі країни Південно-Східної Азії, Карибського басейну, Африки та Тихоокеанського регіону), другий сценарій спостерігався в багатьох країнах Європи (масштабні спалахи захворювання, проте їх вдалося взяти під контроль завдяки сильному керівництву), третій сценарій розвертався в країнах, яким вдалося подолати перший пік спалаху хвороби, проте вони послабили обмеження і тепер змушені боротися з новою хвилею захворювання (глава ВООЗ не навів приклади), четвертий сценарій — фаза інтенсивної передачі інфекції (Америка, Південна Азія і кілька країн Африки).

Усвідомлюючи масштаб трагедії у світі, було вирішено створити технологічне рішення, яке допоможе спростити та оптимізувати проходження різних зон аеропорту під час карантинних заходів і таким чином запобігти уникненню нових спалахів пандемії.

Метою дипломного проєкту є вдосконалення процедур аеропортової перевірки пасажирів під час карантинних заходів шляхом розробки програмно-технологічного рішення.

Для досягнення мети в ході виконання роботи були виконані наступні завдання:

1) Аналіз предметної області;

2) Визначення вимог до програмно-технологічного рішення;

3) Розроблення програмних модулів для програмно-технологічного рішення для прискорення медичних перевірок пасажирів в аеропортах під час карантинних заходів;

4) Тестування розроблених програмних модулів застосунку.

РОЗДІЛ 1. Аналіз предметної області.

1.1. Програмно-технологічні рішення, їх різновид та класи.

Програмно-технологічні рішення можна віднести до інформаційних систем та технологій, які в свою чергу складаються з програмного та апаратного забезпечення. А отже:

Інформаційні технології — це така система методів, процесів і способів використання обчислювальної техніки та систем зв'язку для створення, збору, передачі, пошуку, оброблення та поширення інформації з метою ефективної організації діяльності людей.

Види сучасних інформаційних технологій:

- Інформаційна технологія опрацювання
- Інформаційна технологія керування
- Інформаційна технологія підтримки прийняття рішень
- Інформаційна технологія експертних систем

Також є декілька властивостей які притаманні інформаційним технологіям, це:

- високий ступінь розчленованості процесу на стадії, що відкриває нові можливості для його раціоналізації і перекладу на виконання за допомогою машин. Це — найважливіша характеристика машинного технологічного процесу;
- системна повнота (цілісність) процесу, що повинний включати весь набір елементів, що забезпечують необхідну завершеність дій людини при досягненні поставленої мети;
- регулярність процесу й однозначність його фаз, що дозволяють застосовувати середні величини при їхній характеристиці, і, отже, що допускають їхню стандартизацію й уніфікацію. У результаті з'являється можливість обліку, планування, диспетчеризації інформаційних процесів.

Програмне забезпечення (програмні засоби) (*ПЗ*; англ. *software*) — сукупність програм системи оброблення інформації та програмних документів, необхідних для експлуатації цих програм. Це набір інструкцій, які розповідають комп'ютеру, як працювати. Це на відміну від апаратного забезпечення, з якого побудована система і фактично виконує роботу.

На найнижчому рівні програмування виконуваний код складається з інструкцій машинної мови, які підтримуються окремим процесором — як правило, центральним процесором (CPU) або графічним процесором (GPU). Машинна мова складається з груп двійкових значень, що означають інструкції процесора, які змінюють стан комп'ютера з його попереднього стану. Наприклад, інструкція може змінити значення, що зберігається в певному місці зберігання на комп'ютері — ефект, який користувач не може спостерігати безпосередньо. Інструкція також може викликати одну з багатьох операцій введення або виведення, наприклад, відображення тексту на екрані комп'ютера; спричиняє зміни стану, які мають бути видимі для користувача.

Процесор виконує інструкції в тому порядку, в якому вони надані, якщо йому не вказано «перейти» до іншої інструкції або він не переривається операційною системою. Станом на 2015 рік більшість персональних комп'ютерів, смартфонів і серверів мають процесори з кількома виконавчими блоками або кількома процесорами, які виконують обчислення разом, і обчислення стали набагато більш одночасною діяльністю, ніж у минулому.

Більшість програмного забезпечення написано на мовах програмування високого рівня. Вони легші та ефективніші для програмістів, оскільки вони ближчі до природних мов, ніж до машинних. Мови високого рівня перекладаються на машинну мову за допомогою компілятора або інтерпретатора або їх комбінації. Програмне забезпечення також може бути написане на мові ассемблера низького рівня, яка повністю відповідає інструкціям машинної мови комп'ютера і перекладається на машинну мову за допомогою асемблера.

Розрізняють системне програмне забезпечення (зокрема, операційна система, транслятори, редактори, графічний інтерфейс користувача); прикладне програмне забезпечення, що використовується для виконання конкретних завдань, наприклад, статистичне програмне забезпечення; інструментальне програмне забезпечення (комп'ютерні програми, призначені для проектування, розробки, адміністрування і супроводження системного та прикладного програмного забезпечення).

Виконання програмного забезпечення комп'ютером полягає у маніпулюванні інформацією та керуванні апаратними компонентами комп'ютера. Наприклад, типовим для персональних комп'ютерів є відтворення інформації на екран та отримання її з клавіатури.

Програмне забезпечення (*software*) та апаратне забезпечення (*hardware*) — це два комплементарні компоненти комп'ютера, причому межа між ними нечітка: деякі фрагменти програмного забезпечення на практиці реалізуються суто

апаратурою мікросхем комп'ютера, а програмне забезпечення, своєю чергою, здатне виконувати (емулювати) функції електронної апаратури. По суті, призначення програмного забезпечення полягає в керуванні як самим комп'ютером, так і іншими програмами та маніпулюванні інформацією.

Комплекс програм, які забезпечують управління компонентами комп'ютерної системи, такими як процесор, оперативна пам'ять, пристрої введення-виведення, мережеве обладнання, виступаючи як «міжшаровий інтерфейс», з одного боку якого — апаратура, а з іншого — додатки користувача. На відміну від прикладного програмного забезпечення, системне не вирішує конкретні практичні завдання, а лише забезпечує роботу інших програм, надаючи їм сервісні функції, абстрагуючи деталі апаратної та мікропрограмної реалізації обчислювальної системи, керує апаратними ресурсами обчислювальної системи. Віднесення того чи іншого програмного забезпечення до системного є умовним, і залежить від угод, використовуваних у конкретному контексті. Як правило, до системного програмного забезпечення відносяться операційні системи, широкий клас сполучного програмного забезпечення.

Програмне забезпечення для підприємств поділяється на класи:

- ERP (Enterprise Resources Planning) — система планування ресурсів підприємства;
- CRM (Customer Relationship Management) — система управління взаємовідносинами з клієнтами;
- SCM (Supply Chain Management) — система управління логістичним ланцюгом;
- PLM (Product Lifecycle Management) — система управління життєвим циклом продукту;
- SRM (Supplier Relationship Management) — система управління взаємовідносинами з постачальниками;

- BI (Business Intelligence) — інтелектуальні системи підтримки стратегічного менеджменту.

За ступенем тиражованості все програмне забезпечення ділиться на три категорії:

- програмне забезпечення, що розробляється на замовлення;
- програмне забезпечення для великих корпорацій і організацій;
- програмне забезпечення для масового споживача.

За ступенями переносності програми діляться на:

- Платформозалежні.
- Кросплатформові.

За способами розповсюдження і використання ПЗ поділяється на:

- невільні;
- відкриті;
- вільні.

За призначенням програми діляться на:

- системні;
- прикладні.

За видами програми ділять на:

- компонент — це програма, що розглядається як те єдине ціле, що виконує закінчену функцію і застосовується самостійно або в складі комплексу;
- комплекс — це програма, що складається з двох або більше компонентів і (або) комплексів, що виконують взаємозв'язані функції, і застосовується самостійно або в складі іншого комплексу.

Окрім того, додатково ще виділяють безкоштовні програми.

Також не від’ємною частиною ПЗ є – тестування. Будь-який програмний продукт має бути протестованим для виявлення дефектів і помилок, припущених на стадії інженерії ПЗ. Тестування програмного забезпечення — це перевірка того, чи відповідають фактичні результати очікуваним. Процес передбачає запуск та виконання компонента програмного забезпечення або компонента системи для оцінки однієї або декількох властивостей.

Виділяють три основних види програмного забезпечення: системне програмне забезпечення, пакети прикладних програм та інструментарій технології програмування.

Системне програмне забезпечення являє собою сукупність програм і програмних комплексів, що забезпечують роботу комп'ютера і комп'ютерних мереж. Системне програмне забезпечення направлено:

- на створення операційного середовища функціонування інших програм;

- забезпечення надійної та ефективної роботи самого комп'ютера та комп'ютерної мережі;
- проведення діагностики і профілактики апаратури комп'ютера та комп'ютерної мережі;
- виконання допоміжних технологічних процесів (копіювання, архівування, відновлення файлів програм і баз даних і т. д.).

Цей клас програмних продуктів тісно пов'язаний з комп'ютером і є його невіддільною частиною.

Пакети прикладних програм є комплекс взаємопов'язаних програм для вирішення функціональних завдань певного класу в конкретній предметній області. *Прикладне* програмне забезпечення, або додатки, відноситься до найширшого класу програмних продуктів, призначених безпосередньо для користувача.

Інструментарій технології програмування являє собою сукупність програм і програмних засобів, що забезпечують технологію розробки, налагодження і впровадження створюваних програмних продуктів.

1.2. Огляд програмно-технологічних рішень з автоматизації перевірок пасажирів.

Інформаційна система грає значну роль у виробництві і технологіях виконуваних робіт, слугує фундаментом організації аеропортової діяльності, у тому числі й обслуговування пасажирів. Відомо, що найсучасніші системи автоматизації управління ресурсами і пропускною спроможністю аеропорту базуються на ІТ.

Наявність в аеровокзалі інформаційної системи, в якій враховується все, що відбувається у виробничому циклі обслуговування пасажирів і клієнтури, безперервно забезпечує персонал і пасажирів необхідною візуальною інформацією, визначає ритмічність і послідовність їх дій.

Налагодження системою інформації, високого ступеня надійності її функціювання, доступності та простоті сприйняття візуальної інформації в приміщеннях аеровокзалів повинна приділятися особлива увага.

Автоматизація реєстрації та перевірок пасажирів в аеровокзалах здійснюється за допомогою впровадження електронних систем, які постійно вдосконалюються.

В ДМА «Бориспіль», на долю якого припадає до 40% транспортної роботи авіаційного транспорту, використовуються наступні автоматизовані системи управління:

– Система загального використання ресурсів аеропорту

– CUPSS – Система авіакомпанії для обробки пасажирів і багажу

– DCS(Departure Control System) – Центральна база даних, яка зберігає всю інформацію, пов'язану з діяльністю аеропорту

– AODB – Система управління будівлею аеропорту

– BMS – Принтер бирок для багажу

– BTP – Система відображення пасажирської інформації на моніторах

– FIDS – Система для зчитування багажних бирок на стійках реєстрації – BRS

Лідерство міжнародної системи реєстрації пасажирів і контролю посадки DCS, виробником якої є американська компанія SITA, надає змогу здійснювати реєстрацію пасажирів незалежно від її типу реєстрації: реєстрація на власні рейси авіакомпанії, на стикувальні рейси або рейси інтерлайн-партнерів (сервер знаходиться у постачальника послуг).

Цю систему використовують більше 200 авіакомпаній по всьому світу. Всього на базі SITA у світі працює 235 аеропортів. Це більше 30 тисяч робочих станцій і понад 60 мільйонів пасажирів на рік, тобто кожен третій комерційний пасажир у світі. Вона сумісна з будь-якою системою бронювання, в автоматичному режимі формує всі необхідні супровідні документи про рейси авіакомпаній, згідно з вимогами міжнародних

галузевих стандартів, здатна суттєво спростити процедуру реєстрації пасажирів.

Автоматизована система контролю відправок пасажирів і багажу SITA DCS дозволяє прискорити реєстрацію пасажирів, а також оперативно обробляти і пересилати всю необхідну інформацію про пасажирів та їх багажу, підвищуючи якість обслуговування різних категорій пасажирів. На основі отриманих SITA DCS даних про кількість пасажирів і багажу автоматично визначається ступінь завантаження центрування повітряного судна. Використання такого алгоритму є більш зручнішим, тому що раніше цю функцію виконували диспетчера центрування. Крім цього, система дозволить автоматично закріплювати за пасажиром найбільш зручніше, на його думку, місце в літаку, друкувати посадочні талони і багажні бирки.

Важливішою перевагою SITA DCS є можливість наскрізної реєстрації трансферних пасажирів, які прямують через інші аеропорти, зменшити час стикування рейсів, підвищити рівень регулярності польотів. Така реєстрація з використанням комп'ютеризованих інформаційних систем відіграє одну з ключових ролей в існуванні і розвитку хабів, оскільки дозволяє пасажирам швидко пересісти з одного рейсу на інший.

Програмний комплекс SITA DCS дозволяє працювати з кіосками самостійної реєстрації, які вже зараз широко поширені в Америці, Канаді, Європі, в тому числі і в ДМА «Бориспіль».

Вже зараз велика кількість аеропортів світу зазначають, що час реєстрації при самостійному обслуговуванні пасажирів вдвічі зменшує проміжки часу на аеропортові процедури. Враховуючи це, знижується потреба у висококваліфікованому персоналі, скорочуються вимоги щодо наявності великих площ в аеропортах для обслуговування пасажирів, знижуються витрати на прибирання та утримання великих залів реєстрації.

Сьогодні перевізники впроваджують рішення щодо друкування багажних бирок на домашньому принтері, що дозволить збільшити економічний ефект по обслуговуванню пасажирів в аеропорту. Ініціатива по використанню процедури самостійного випуску багажної бирки сьогодні є метою авіакомпаній в рамках реалізації стратегії "стати компанією, з якою літати найлегше".

Безумовно, наступним кроком автоматизації роботи аеропортів і авіакомпаній буде організація самостійної посадки пасажира на певни рейс. Розповсюджене поширення нових технологій позитивно відобразиться на зменшенні вартості цього процесу, подібно зниженню витрат на інші схожі процедури при впровадженні сучасних підходів. Аеропорти "майбутнього" будуть все більше переходити під управління інформаційних систем і впроваджувати самостійне обслуговування.

1.3. Огляд програмно-технологічних рішень з перевірки сертифікатів вакцинації.

Одним з таких рішень є введення Мінцифрою налагодженого ефективного процесу між держустановами для швидкого створення та впровадження ковідних сертифікатів у Дії, які вже допомогли всім вакцинованим українцям повернутися до звичного життя. Ковідний сертифікат — це документ у Дії, який буде підтверджувати ваш статус про вакцинацію в Україні, на території ЄС та інших країнах.

Ковідні сертифікати працюють так, як й інші документи в Дії. Перевірити дійсність сертифіката можна за QR-кодом. Це зручно, адже не потрібно додатково впроваджувати спеціальне обладнання для перевірки. Також ковідний сертифікат можна згенерувати після негативного результату ПЛР-тестів або якщо ви нещодавно перехворіли на COVID-19.

Документ розробили з урахуванням вимог EU Digital COVID Certificate Єврокомісії та Smart Vaccination Certificate ВООЗ. Технічною реалізацією спільно займаються команди Мінцифри, МОЗ, НСЗУ, МЗС, ДПСУ.

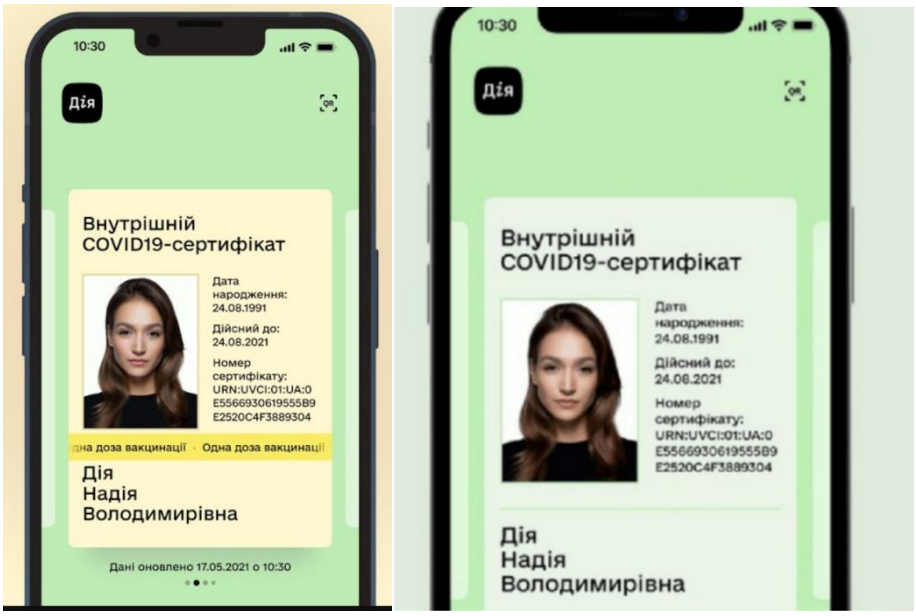


Рис. 1.3.1-1.3.2 - Приклад жовтого та зеленого сертифікатів вакцинацій(одна та дві дози).

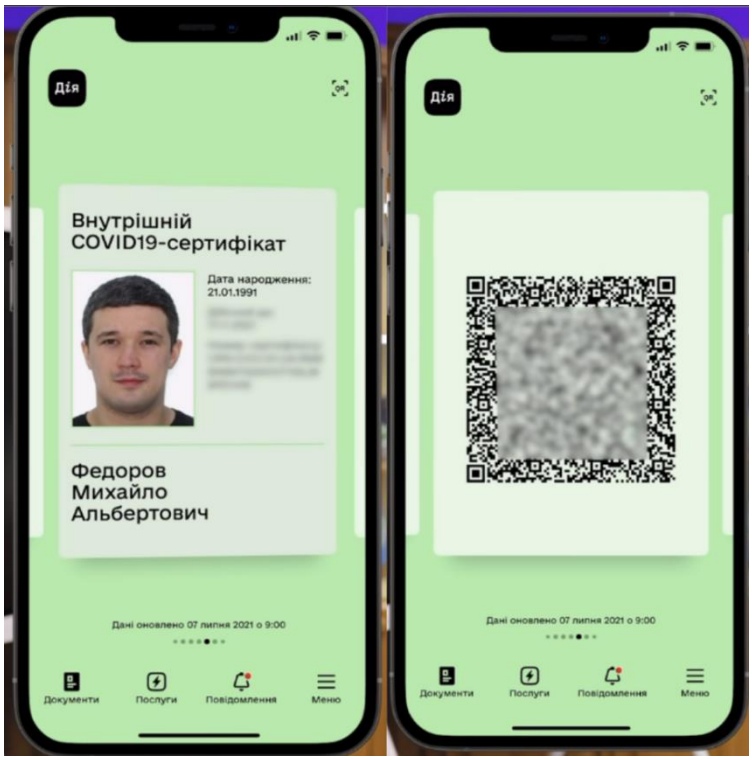


Рис. 1.3.3 - Приклад сертифікату вакцинації з QR-кодом.

Висновок

У цьому розділі була проведена робота по аналізу предметної області. Було розглянуто що таке програмно-технологічне рішення, що таке ІТ, та властивості які притаманні ІТ, також було розглянуто що таке ПЗ та їх класи. Розглянуто програмно-технологічні рішення з автоматизації перевірок пасажирів. А також розглянуто програмно-технологічне рішення з перевірки сертифікатів вакцинації від МОЗ.

РОЗДІЛ 2. Розроблення вимог до програмно-технологічного рішення для прискорення медичних перевірок пасажирів в аеропортах під час карантинних заходів.

2.1. Створення інформаційних потоків.

Такими потоками будуть корисні відео на Ютубі, які підходять під тему проєкту. Також усі новинні сайти про автоматизацію ковіних сертифікатів та автоматизацію перевірок пасажирів в аеропортах.

А також поговоримо про принцип дії програмного забезпечення. Принципом дії буде те що вже вакцинований пасажир(не вакциновані не розглядаються) прибуває в аеропорт з електронним посадковим квитком, тож пред тим як від дістанеться літака треба буде пройти через декілька перевірочних точок, де треба буде показувати білет та стан вакцинації. Але завдяки цьому програмному рішенню значну кількість перевірок можна буде уникнути. Отже, пасажир вводить свої дані(ім’я, прізвище, місто, куди він летить, вакцина та кількість доз цієї вакцини), але від кількості доз(одна чи дві) будуть накладені обмеження по відвідуванню деяких приміщень. Завдяки цьому можна значно скоротити скупчення людей в одному місці, а отже оптимізувати роботу аеропорту, уникаючи великої кількості захворювань серед населення.

2.2. Структурно-функціональна схема програмно-технологічного рішення.

Розробляючи програму, було вирішено розробити систему класів програми та зобразити їх взаємодію через діаграму класів. Отже розглянемо її на рис. 2.2.1. В цій схемі показано який клас від якого наслідується та залежність класів від бази даних. Також зображено виклики функцій з класу Main у функціях init_child та init_edit. Зображено також виклики інших класів у класі Main.

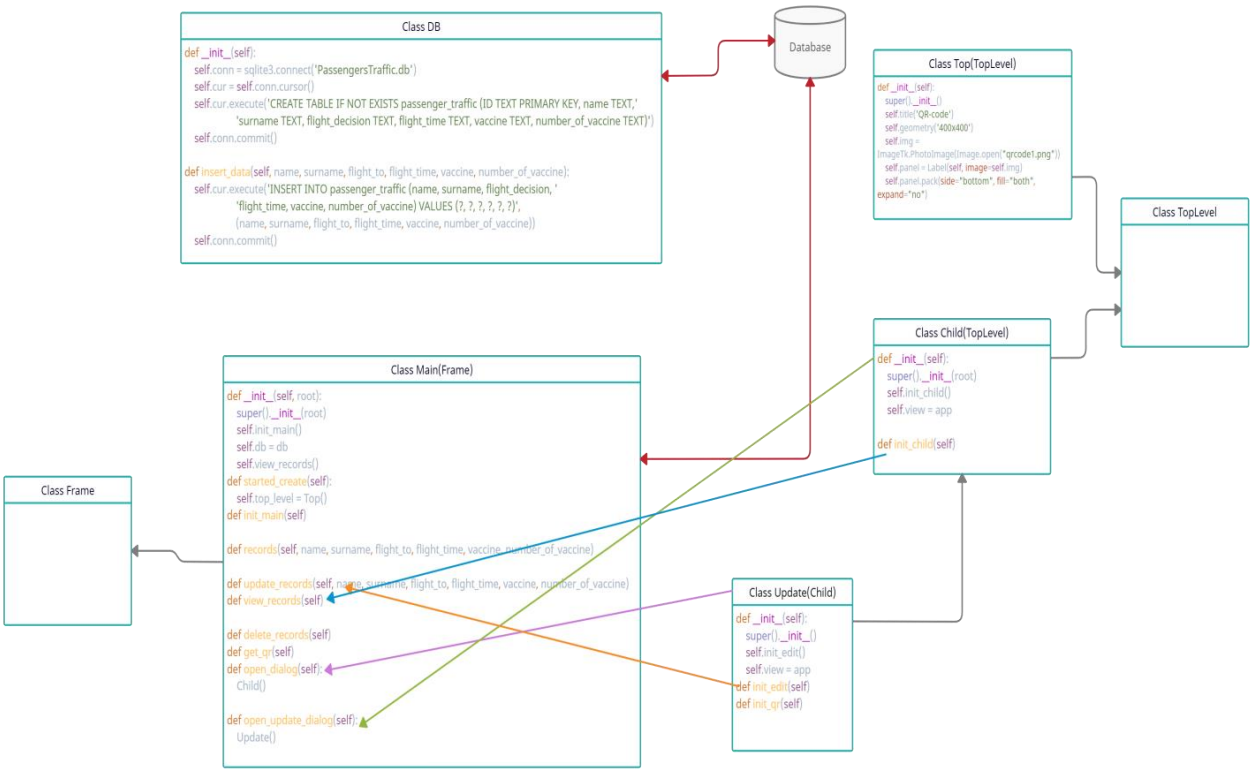


Рис. 2.2.1 - Діаграма класів.

2.3. Вимоги до програмного забезпечення.

Інженерія вимог – це розділ інженерії програмного забезпечення, що включає в себе збір та аналіз вимог до програмної системи що знаходиться в розробці, визначення важливих цілей функцій, характеристик та алгоритмів, які властиві програмному продукту. Крім цього інженерія вимог також займається документуванням та валідацією поставлених вимог. Підсумовуючи, можна сказати що вимоги – це твердження стосовно

характеристик та властивостей, які повинен мати програмний продукт, який знадходять як з боку замовника системи, так і може виникнути в процесі аналізу командної розробки.

Вимоги проходять чотири фази розробки:

1. Виявлення вимог та збір всієї необхідної інформації щодо них;
2. Аналіз вимог, проведення уточнення в формулюванні забраних вимог та встановлення для них рамок та пріоритетів;
3. Документування поставлених вимог;
4. Перевірка специфікації вимог.

Вимогами до даного програмного продукту будуть невеличкі але не більш важливі. Тож, при плануванні та реалізації продукту потрібно дотримуватися чіткого плану побудови класів та функцій системи.

Застосунок буде мати 3 вимоги: функціональні, нефункціональні та системні.

Функціональні вимоги:

Функціональні вимоги – це вимоги, що відповідають за функціональну наповненість програмного забезпечення, визначають задачі та функції, які має реалізувати розробник програмної системи. Опис функціональних вимог полягає в структурованому списку функцій та поведінку системи в процесі реакції на них.

Програмно-технологічне рішення для прискорення медичних перевірок пасажирів в аеропортах під час карантинних заходів повинно мати наступні функції:

- Можливість ведення певних даних від користувачів.
- Створення та відображення даних.
- Надійне збереження даних у БД, які вводять користувачі.

- Редагування особистої інформації.
- Генерація QR-коду ґрунтуючись на введених даних.
- Відображення QR-коду.

Нефункціональні вимоги:

Нефункціональні вимоги – це поставлені вимоги, які спрямовані на опис інтерфейсу, атрибутів, характеристик та властивостей, які повинна мати програмна система.

Для розроблюваного застосунку найважливіше дотримуватись наступних вимог:

- Простота та зручність інтерфейсу програмного продукту.
- Ергономічне розташування елементів інтерфейсу.
- Пропорціональне співвідношення розмірів кнопок, тексту та іконок між собою.

Системні вимоги:

Вони визначають характеристики, яким має відповідати комп’ютер, смартфон чи інший електронний пристрій, на якому можливо буде встановити розроблювальний програмний продукт.

Вимоги до ОС: актуальна LTS версія Microsoft Windows.

Вимоги до АЗ: процесор з тактовою частотою від 2,5 ГГц, 1 ГБ ОЗП, відеоадаптер з 1 ГБ відеопам’яті, >1 ГБ вільного місця на жорсткому диску.

Дотримуючись цих вимог буде розроблено просту в розумінні програму у короткий строк, інтерфейс якої буде інтуїтивно зрозумілим кожній людині яка вміє користуватися комп’ютером чи смартфоном.

Висновок

У цьому розділі було створено інформаційні потоки, діаграму залежності класів один від одного. Також було створено вимоги до ПЗ, такі як: функціональні вимоги, нефункціональні вимоги та вимоги до системи.

РОЗДІЛ 3. Розроблення програмних модулів для програмно-технологічного рішення для прискорення медичних перевірок пасажирів в аеропортах під час карантинних заходів.

3.1. Вибір мови програмування.

Для розробки програмного застосунку обрано мову Python та компилятор який влаштований у програмне середовище PyCharm. Вибір обумовлений тим що ця мова легка в розумінні та засвоюванні, а також активно підтримується спільнотою, завдяки чому в інтернеті можна знайти багато документацій та обговорень на різні теми мови та все що з нею пов’язано. Це високорівнева мова програмування заснована на декількох парадигмах: імперативна, функціональна та об'єктно-орієнтована, також ця мова є строго типізованою.

Також мною була використана мова SQL - це діалогова мова програмування за допомоги якої можна здійснити запит і внести змін до бази даних, а також керувати ними. Користуючись нею я створив базу до якої будуть занесені дані про пасажира: його ім’я, прізвище, місто до якого він летить, час відправки рейсу, назва вакцини та кількість доз цієї вакцини.

Для розгляду вже створеної бази даних та коригування процесом створення самої БД, була взята СУБД під назвою SQLite. Рішення використовувати цю СУБД було прийнято виходячи з мови програмування та її вмонтованих функціях та підтримки саме SQLite. Отже, SQLite — полегшена реляційна система керування базами даних. Втілена у вигляді бібліотеки, де реалізовано багато зі стандарту SQL-92. Сирцевий код SQLite поширюється як суспільне надбання (англ. *public domain*), тобто може використовуватися без обмежень та безоплатно з будь-якою метою.

3.2. Розроблення програмного модуля для занесення даних в БД, їх зміна та видалення з бази даних.

Для створення модулів я використав вмонтовані в Python бібліотеки під назвою “tkinter” та “sqlite3” використавши синтаксис import tkinter та import sqlite3 відповідно.

```
import tkinter as tk
from tkinter import ttk
import sqlite3
```

Рис. 3.2.1 - Імпорт бібліотек для майбутнього використання.

Основним кодом цієї частини програми буде клас “Main”, який відповідає за всі важливі функції, створені у ньому. До таких, наприклад, буде відноситися функція “init_main”, у якій створюється головне вікно, до якого додано кнопки та таблицю, яка відображає внесенні та зміненні дані. Але повернемося до функції, за допомоги якої створюються дані та заносяться до БД.

Для того щоб дані створювалися та заносилися до БД, відповідно спочатку треба її створити, отже, створення відбувається у класі під назвою “DB”. У ній є головна функція та функція, яка відповідає за отримання даних у відповідні колонки БД. Головна функція називається “__init__” та відповідає за створення самої бази даних(її назва, колонки та їх тип). Інша ж функція відповідає за саму відповідність змінних до назв колонок.

```
class DB:
    def __init__(self):
        self.conn = sqlite3.connect('PassengersTraffic.db')
        self.cur = self.conn.cursor()
        self.cur.execute('CREATE TABLE IF NOT EXISTS passenger_traffic (ID TEXT PRIMARY KEY, name TEXT,'
                         'surname TEXT, flight_decision TEXT, flight_time TEXT, vaccine TEXT, number_of_vaccine TEXT)')
        self.conn.commit()

    def insert_data(self, name, surname, flight_to, flight_time, vaccine, number_of_vaccine):
        self.cur.execute('INSERT INTO passenger_traffic (name, surname, flight_decision, '
                         'flight_time, vaccine, number_of_vaccine) VALUES (?, ?, ?, ?, ?, ?)',
                         (name, surname, flight_to, flight_time, vaccine, number_of_vaccine))
        self.conn.commit()
```

Рис. 3.2.2 - Клас “DB”, в якому створюється база даних.

Тепер коли ми створили БД, можемо почати заносити в неї дані. Це буде відбуватися у головному класі, функції “view_records” та “records”. Перша відповідає за приймання внесених даних у базу даних та викликає “view_records”, “view_records” в свою чергу відповідає за відображення їх у головному вікні програми.

```
def records(self, name, surname, flight_to, flight_time, vaccine, number_of_vaccine):
    self.db.insert_data(name, surname, flight_to, flight_time, vaccine, number_of_vaccine)
    self.view_records()
```

Рис. 3.2.3 - Функція “records” головного класу “Main”.

```
def view_records(self):
    self.db.cur.execute('SELECT * FROM passenger_traffic')
    [self.tree.delete(i) for i in self.tree.get_children()]
    [self.tree.insert('', '0', values=row) for row in self.db.cur.fetchall()]
```

Рис. 3.2.4 - Функція “view_records” головного класу “Main”, для відображення даних в головному вікні.

Відповідно, майже таким же чином виконана функція зміни даних. Це відбувається в середині функції “update_records”. Таким же чином приймаючи дані та вносячи зміни, завдяки виклику функції “view_records” будуть відображатись змінені чи оновленні дані.

```
def update_records(self, name, surname, flight_to, flight_time, vaccine, number_of_vaccine):
    self.db.cur.execute('UPDATE passenger_traffic SET name=?, surname=?, flight_decision=?,'
                        'flight_time=?, vaccine=?, number_of_vaccine=? WHERE ID=?',
                        (name, surname, flight_to, flight_time, vaccine, number_of_vaccine,
                         self.tree.set(self.tree.selection()[0], '#1'),))
    self.db.conn.commit()
    self.view_records()
```

Рис. 3.2.5 - Функція “update_records” головного класу “Main”, для внесення змін.

Тепер поговоримо про видалення даних з БД. Саме видалення відбувається у функції “delete_records”, та використовує синтаксис мови SQL для видалення. Завдяки строчці коду self.db.cur.execute('DELETE FROM passenger_traffic WHERE id=?', (self.tree.set(selection_item, '#1'),)) видаляються тільки обрані строки.

```
def delete_records(self):
    for selection_item in self.tree.selection():
        self.db.cur.execute('DELETE FROM passenger_traffic WHERE id=?', (self.tree.set(selection_item, '#1'),))
    self.db.conn.commit()
    self.view_records()
```

Рис. 3.2.6 - Функція “delete_records” головного класу “Main”, для видалення непотрібних та неактуальних даних.

Але всі ці дії не виконуються просто так, вони запрограмовані на виконання по натиску на певну кнопку. Тож розглянемо і цю частину коду. Для початку створимо змінну на яку повісимо картинку майбутньої кнопку, наприклад: self.add_img, self.update_img, self.delete_img. Потім ми даємо назву змінної яка буде зберігати в собі весь функціонал кнопки (btn_open_dialog, btn_edit_dialog, btn_delete). Створюючи кнопку, першим атрибутом надаємо їй, те, до якого фрейму буде належати кнопка, потім даємо їй назву, прописуємо яку команду вона буде виконувати, де вона буде розташована, її задній фон, назначаємо їй картинку, створену раніше, та за допомоги методу pack(), розташовуємо її по ліву сторону.

```
def init_main(self):
    toolbar = tk.Frame(bg='#d7d8e0', bd=2)
    toolbar.pack(side=tk.TOP, fill=tk.X)
    self.add_img = tk.PhotoImage(file='add.gif')
    btn_open_dialog = tk.Button(toolbar, text='Додавання даних', command=self.open_dialog,
                                bg='#d7d8e0', bd=0, compound=tk.TOP, image=self.add_img)
    btn_open_dialog.pack(side=tk.LEFT)

    self.update_img = tk.PhotoImage(file='update.gif')
    btn_edit_dialog = tk.Button(toolbar, text='Редагувати дані', bg='#d7d8e0', bd=0, image=self.update_img,
                                compound=tk.TOP, command=self.open_update_dialog)
    btn_edit_dialog.pack(side=tk.LEFT)

    self.delete_img = tk.PhotoImage(file='delete.gif')
    btn_delete = tk.Button(toolbar, text='Видалити дані', bg='#d7d8e0', bd=0, image=self.delete_img,
                           compound=tk.TOP, command=self.delete_records)
    btn_delete.pack(side=tk.LEFT)
```

Рис. 3.2.7 - Функція “init_main” в якій також реалізовані кнопки для занесення, зміни та видалення даних з БД.

3.3. Розроблення програмного модуля для отримання QR-коду.

Для отримання QR-коду також був використаний модуль “pyqrcode”, який я попередньо встановив. Встановлення модуля відбулося завдяки

команді pip install pyqrcode. Виклик бібліотеки - за допомоги команди import pyqrcode. Головним кодом створення QR-коду є функція:

```
def get_qr(self):
    with sqlite3.connect("PassengersTraffic.db") as connect:
        items = connect.execute("SELECT name, surname, flight_decision, flight_time,"
                                "vaccine, number_of_vaccine FROM passenger_traffic").fetchone()
        print(items)

    text = "\n".join(items)
    qr_code = pyqrcode.create(text)
    qr_code.png('qrcode1.png', scale=8)
```

Рис. 3.3.1 - Функція створення QR-коду.

Також щоб згенерувати та відобразити QR-код, назначаємо ці дій на певні кнопки, яким даємо назви “Згенерувати QR-код” та “Отримати QR-код”. Кнопки реалізовані таким же чином, як було описано у пункті 3.3. Створюємо картинку, яка буде кнопкою, далі до якого фрейму будуть відноситися кнопки, даємо назви кнопкам, їх дії, положення, задній фон. Все, таким чином було реалізовано генерування та отримання QR-коду з даними пасажира.

```
self.get_qr_img = tk.PhotoImage(file='qr.gif')
btn_get_qr = tk.Button(toolbar, text='Згенерувати QR-код', bg='#d7d8e0', bd=0, image=self.get_qr_img,
                       compound=tk.TOP, command=self.get_qr)
btn_get_qr.pack(side=tk.LEFT)

self.get_qr_img1 = tk.PhotoImage(file='qr.gif')
btn_get_qr1 = tk.Button(toolbar, text='Отримати QR-код', bg='#d7d8e0', bd=0, image=self.get_qr_img1,
                        compound=tk.TOP, command=self.started_create)
btn_get_qr1.pack(side=tk.LEFT)
```

Рис. 3.3.2 - Реалізація кнопок для генерування та отримання QR-коду з даними про пасажира.

Висновок

Пишучи цей розділ було розписано які мови програмування було вибрано та чому саме вони. Також було розроблено програмні модулі для занесення даних в БД, їх зміну та видалення з бази даних. Другим розробленим модулем був модуль для генерації та отримання QR-коду.

РОЗДІЛ 4. Тестування розроблених програмних модулів.

4.1. Підготовка вхідних даних.

Так як інтерфейс був розроблений завдяки вмонтованому модулю tkinter, отримаємо вікно, в якому реалізований весь функціонал.

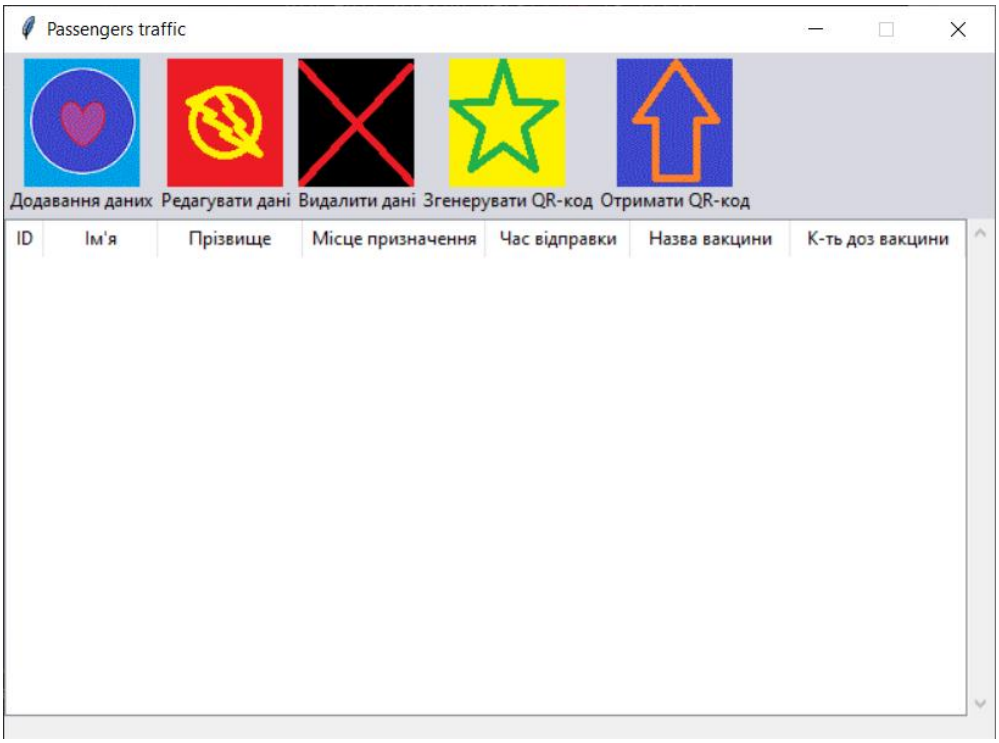


Рис. 4.1.1 - Головне меню програми.

Щоб додати дані, потрібно натиснути “Додавання даних”(Важливо: данні повинні бути записані англійською мовою).

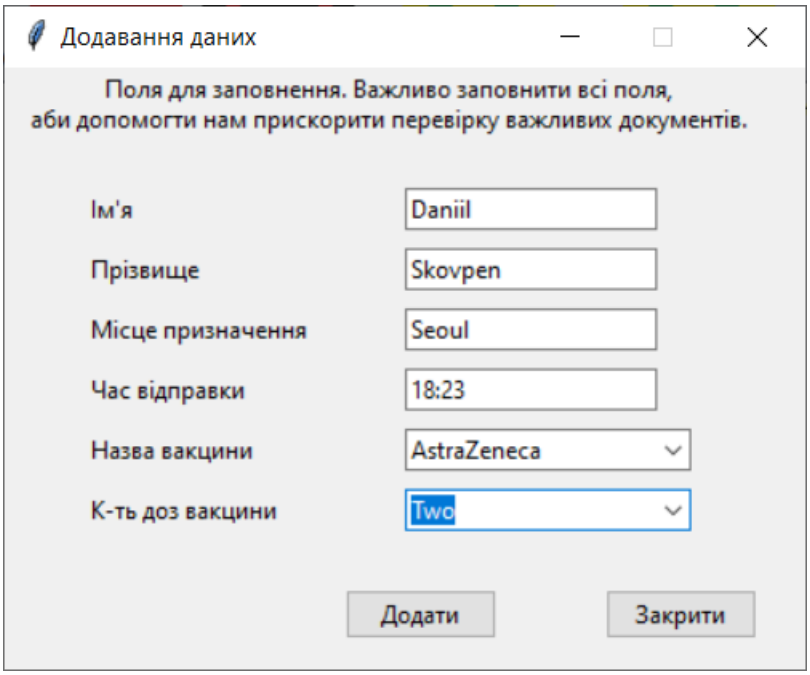


Рис. 4.1.2 - Допоміжне вікно внесення даних.

Натискаємо “Додати” та “Закрити”. Як бачимо дані успішно внесено(Рис. 4.1.3). Тепер перевіримо чи додалися данні до БД.

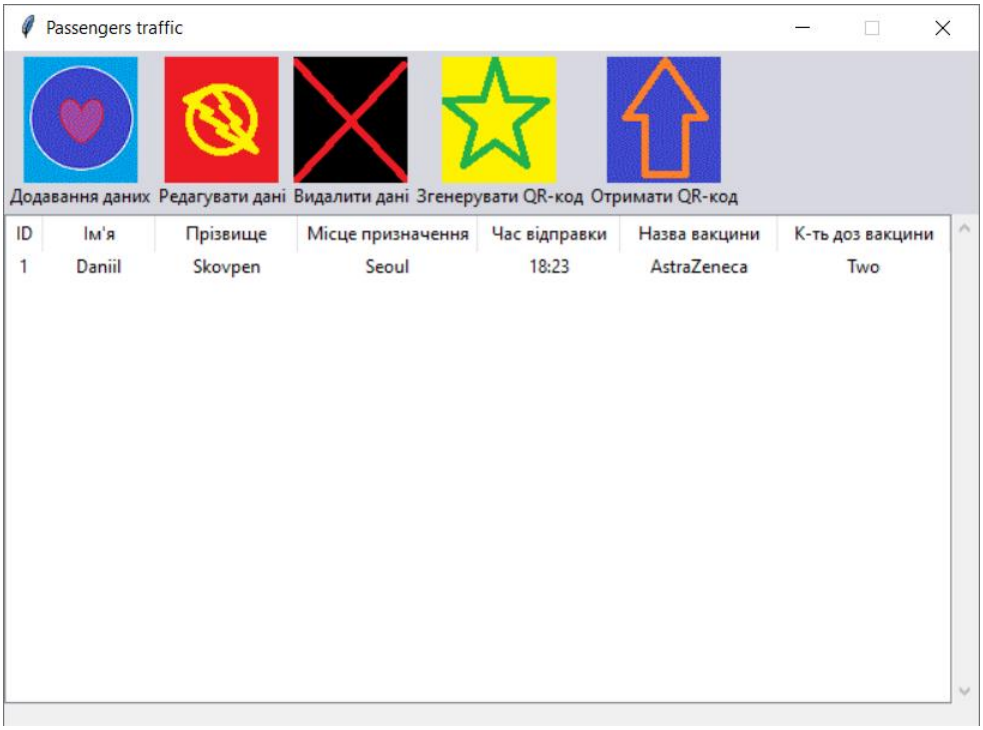
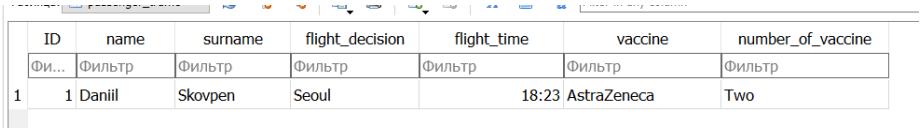


Рис. 4.1.3 - Занесенні дані.

Як бачимо дані в БД було занесено(Рис. 4.1.4). Далі спробуємо згенерувати та отримати QR-код. Щоб це зробити натискаємо спочатку “Згенерувати QR-код” і сразу після цього - “Отримати QR-код”. Для цього важливо спочатку потрібну строку як показано на рис. 4.1.5.



	ID	name	surname	flight_decision	flight_time	vaccine	number_of_vaccine
	Фи...	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр
1	1	Daniil	Skovpen	Seoul	18:23	AstraZeneca	Two

Рис. 4.1.4 - Внесені дані у БД.

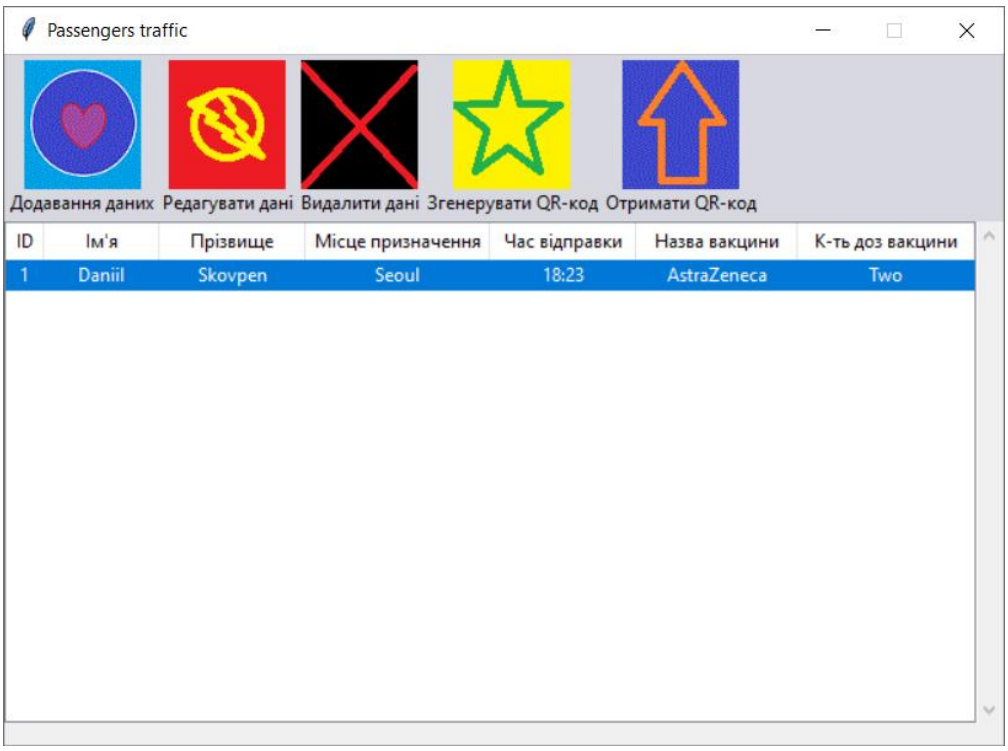


Рис. 4.1.5 - Вибір строки з якої буде згенеровано QR-код.

Щоб зрозуміти що все вийшло, було вирішено разом із генерацією відображати дані які потраплять до QR-коду(Рис. 4.1.6).

```
('Daniil', 'Skovpen', 'Seoul', '18:23', 'AstraZeneca', 'Two')
```

Рис. 4.1.6 - Відображені дані, занесені до QR-коду.

Рис. 4.1.7 - Готовий QR-код з усіма важливими даними про пасажира.

Тепер перевіримо редагування та видалення даних. Спочатку відредагуємо дані які було використано вище. Натискаємо “Редагувати дані”, попередньо вибравши строку яку ми хочемо відредагувати. Вискочить ще одне вікно, в якому і буде відбуватися редагування.

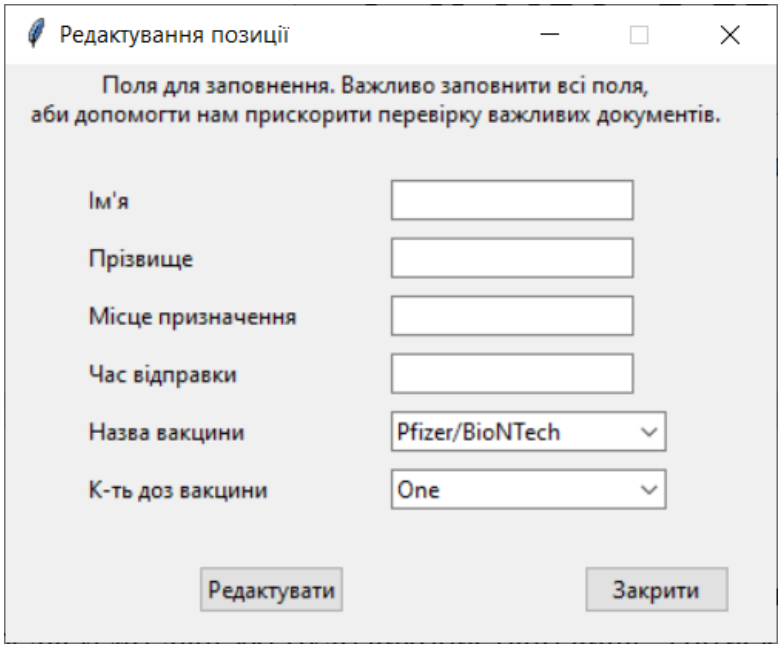


Рис. 4.1.8 - Вікно редагування.

Змінимо наприклад тільки ім’я та назву вакцини. Як бачимо по рис. 4.1.9 та рис. 4.1.10, дані успішно відредаговано.

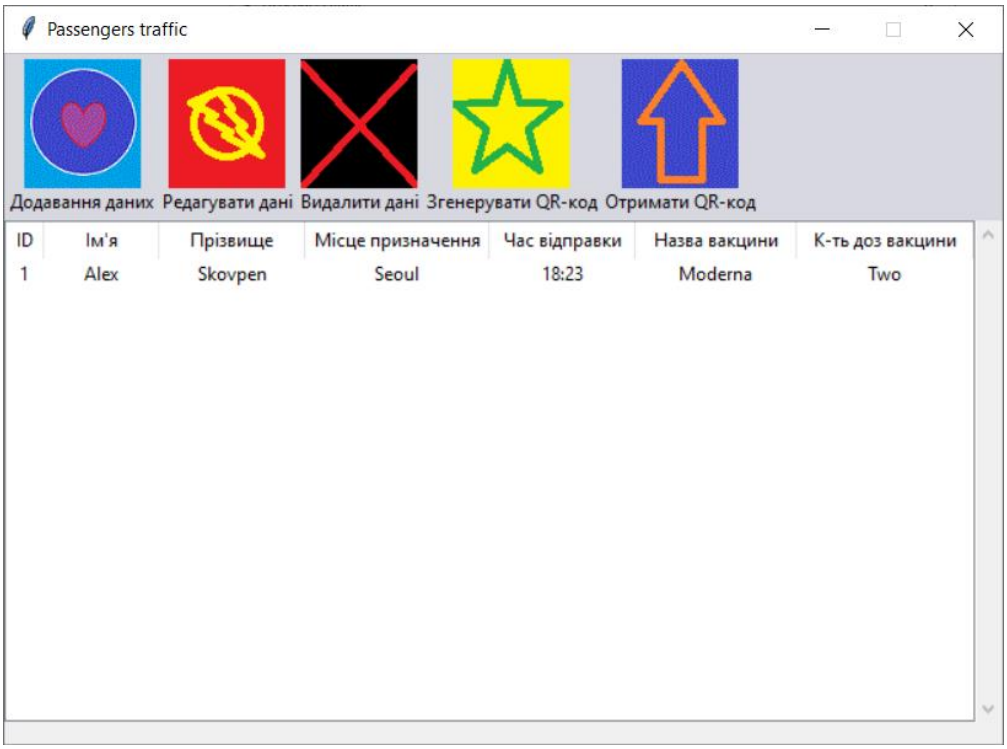


Рис. 4.1.9 - Зміни на головном екрані.

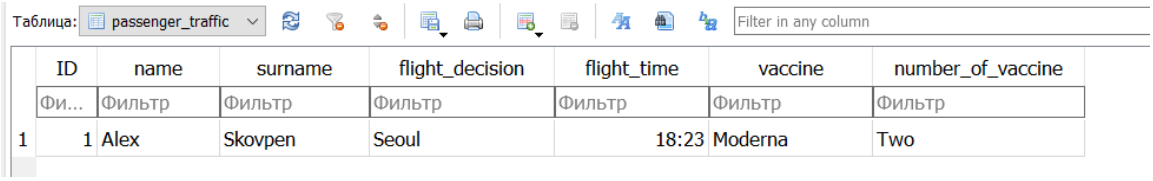


Рис. 4.1.10 - Зміни у самій БД.

Тепер видаляємо, виділивши строку, яку ми хочемо видалити та натикаємо кнопку “Видалити дані”, як бачимо по рис. 4.1.11 та рис. 4.1.12 - все вийшло.

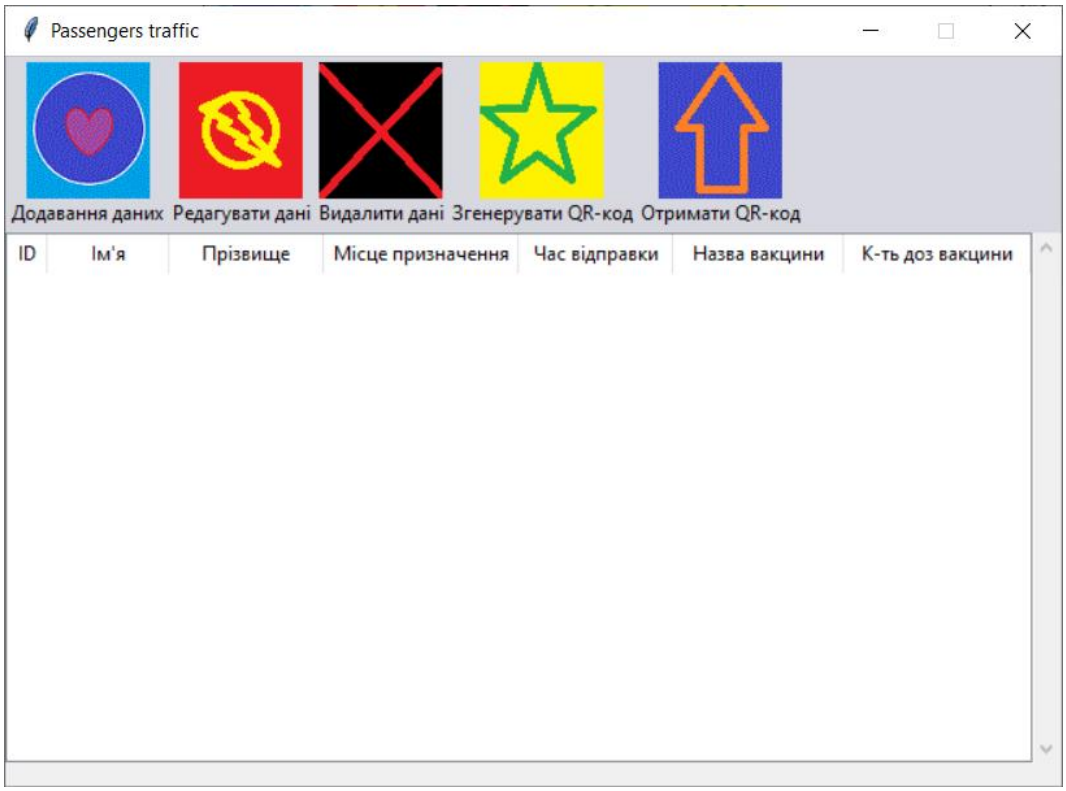


Рис. 4.1.11 - Дані успішно видалені.

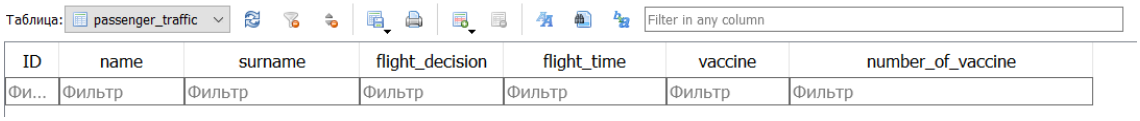


Рис. 4.1.12 - Даних в БД теж відсутні.

4.2. Аналіз результатів роботи.

Програмний застосунок забезпечує виконання таких функцій: Можливість ведення певних даних від користувачів; Створення та відображення даних; Надійне збереження даних у БД, які вводять користувачі; Редагування особистої інформації; Генерація QR-коду ґрунтуючись на введених даних; Відображення QR-коду.

Інтерфейс користувача забезпечує легке внесення потрібних даних та вилучення їх із бази даних у вигляді QR-коду, за допомоги якого і пришвидшиться проходження між різними зонами аеропорту.

Підводячи підсумки виконаної роботи, у цьому розділі було проведенно роботу по основному функціоналу прототипу програмного застосунку.

Описано функції кнопок та взаємозв'язок між ними. В розробці було виконано всі описані в попередніх розділах вимоги. Функціонал реалізовано повністю як зазначено у специфікації та вимогах до застосунку.

Висновок

В останньому розділі було продемонстровано роботу програми. Кожен етап роботи всіх розроблених модулів. Були підведені підсумки роботи програми.

ВИСНОВКИ

Наявність в аеровокзалі інформаційної системи, в якій враховується все, що відбувається у виробничому циклі обслуговування пасажирів, безперервно забезпечує персонал і пасажирів необхідною візуальною інформацією, визначає ритмічність і послідовність їх дій.

Протягом останніх років через пандемію COVID-19 та необхідність дотримання карантинних обмежень існує великий попит на автоматизацію процесів, які раніше передбачали контакт людини з людиною.

В дипломному проєкті проведено аналіз програмно-технологічного рішення, аналіз програмно-технологічних рішень з автоматизації перевірок пасажирів та аналіз програмно-технологічних рішень з перевірок сертифікатів вакцинації. Була проведена розробка спеціального програмного застосунку для прискорення медичних перевірок пасажирів в аеропортах під час карантинних заходів для полегшення та прискорення переходу між різними зонами аеропорту.

Для створення цього додатку було проведено аналіз рограмно-технологічного рішення, аналіз програмно-технологічних рішень з автоматизації перевірок пасажирів та аналіз програмно-технологічних рішень з перевірок сертифікатів вакцинації, сформульовано створення інформаційних потоків та вимог до програмного забезпечення, і розроблено програмний застосунок, що забезпечує прискорення медичних перевірок пасажирів в аеропортах під час карантинних заходів для полегшення та прискорення переходу між різними зонами аеропорту і відповідає сформульованим вимогам.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. A Byte of Python(Укус пітона) / Swaroop C. H. - 164 ст.
2. Пришвидшений курс Python / Маттес Ерік; Видавництво старого лева, 2021. — 600 с. — ISBN 978-617-679-853-8.
3. Об'єктно-орієнтоване програмування: лабораторний практикум/ МОН; Національний авіаційний університет; Гамаюн В. П., Зудов О. М., уклад. – Київ: НАУ-друк, 2010. – 48 с.– CD
4. Навчально-методичний комплекс навчальної дисципліни «об'єктно-орієнтоване програмування» / Серебрякова Світлана Вікторівна – 2020.
5. Python documentation [Електронний ресурс]. – Режим доступу: https://www.python.org/doc/
6. PyQRcode documentation [Електронний ресурс]. – Режим доступу: https://pythonhosted.org/PyQRCode/
7. Python SQLite3 documentation [Електронний ресурс]. – Режим доступу: https://docs.python.org/3/library/sqlite3.html
8. Інструкція по Python та модулю tkinter – Metanit [Електронний ресурс]. – Режим доступу: https://metanit.com/python/tutorial/