КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

ФАКУЛЬТЕТ РАДІОФІЗИКИ, ЕЛЕКТРОНІКИ ТА КОМП'ЮТЕРНИХ СИСТЕМ

Кафедра радіотехніки та радіоелектронних систем

«На правах рукопису»

Робота допущена до захисту в ЕК
рішенням кафедри радіотехніки та радіоелектронних систем
від 20 травня 2024 року, протокол № 111.
Завідувач кафедри доктор фіз.-мат. наук, професор
_____ Ігор АНІСІМОВ

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА
на тему:

«Реалізація на Python телемедичної консультативної системи»

Виконав:
студент 4-го курсу
денної форми навчання
спеціальності 172 - Телекомунікації та радіотехніка
ОПП «Інформаційна безпека телекомунікаційних систем і мереж»
Хлоп'янець Михайло Олександрович _____

Науковий керівник:
канд. фіз.-мат. наук, доцент
Кононов Михайло Володимирович _____

Рецензент:
канд. фіз.-мат. наук, доцент
Єфіменко Світлана Володимирівна _____

Засвідчую, що у цій бакалаврській роботі
немає запозичень з праць інших авторів без
відповідних посилань
Студент Михайло Хлоп'янець

Київ - 2024

РЕФЕРАТ

Дипломна робота: 51 с., 29 рис., 1 дод. (3с.), 12 джерел.

Ключові слова: телемедична система, Python, консультативна система, військова медицина, кросплатформеність, медичні документи, безпека даних.

Об’єкт розроблення– телемедична консультативна системи.

Мета роботи –створення консультативної телемедичної системи на основі мови програмування Python.

Дипломна робота виконує функції розробки телемедичної консультативної системи з використанням декількох мов програмування python,sql. Основним фронт робіт був покладений в вирішенні потреб військовослужбовців, тобто створення застосунку, реалізованого програмним чином. Застосунок був розроблений з урахуванням кроссплатформеності та містить ряд унікальних функцій, які будуть явно демонструвати його користь як на передовій так й в тилу. Висока функціональність, інтуїтивно зрозумілий інтерфейс та можливість подальшого розширення функціоналу роблять цей програмний продукт цінним інструментом для військових медиків і військовослужбовців.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

ВООЗ - Всесвітня організація охорони здоров'я.

SMS - Служба коротких повідомлень Short Message Service.

MVC - Model-View-Controller.

MD - material design.

ТЦК - Територіальний центр комплектування.

Sql - Structured Query Language.

ЗСУ - Збройні сили України.

1.Програмні технології в телемедицині

1.1.Сучасний стан телемедицини

Сучасна гіпербола навколо телемедицини/телездоров'я передбачає, що це революція у наданні медичної допомоги. Телемедицина/телездоров'я часто асоціюється з технологіями, такими як телебачення, комп'ютери, радіо, Інтернет, відеозаписи та факси. Це пояснюється тим, що ці засоби комунікації зараз широко використовуються. Визначення телездоров'я як синоніму сучасних інформаційних технологій упускає той факт, що люди могли обмінюватися інформацією про стан здоров'я на відстані задовго до появи цих технологій. Прості засоби, такі як дзвони, прапори і знаки, використовувалися для цієї мети в минулому.

Покращення в сфері охорони здоров'я населення, такі як нинішня тривалість життя в Сполучених Штатах і Великій Британії, більше пов'язані з громадськими заходами охорони здоров'я для боротьби з інфекційними хворобами, ніж з новими медичними технологіями. До середини XX століття інфекційні захворювання були основною причиною смертності та захворюваності в США та Західній Європі. До недавнього часу системи охорони здоров'я були зайняті переважно боротьбою з інфекційними хворобами і приділяли менше уваги серцевим захворюванням, діабету та інсульту. Дистанційне надання медичної допомоги, або телездоров'я, використовувалося з давніх часів, застосовуючи примітивні комунікаційні технології для запобігання та контролю поширення інфекційних захворювань.

Телемедицина чи телездоров'я?

Слово "телемедицина" стало популярним у сфері охорони здоров'я на початку 1990-х років. У XXI столітті ми передбачаємо, що телемедицина розшириться далеко за межі надання медичної допомоги, тому віддаємо

перевагу терміну "телездоров'я" для опису цих ширших застосувань. Деякі сучасні визначення телемедицини та телездоров'я включають наступне[1]:

Визначення телемедицини 1: Телемедицина включає використання сучасних інформаційних технологій, зокрема двосторонніх інтерактивних аудіо/відео комунікацій, комп'ютерів та телеметрії, для надання медичних послуг віддаленим пацієнтам і для полегшення обміну інформацією між лікарями первинної медичної допомоги та спеціалістами на відстані.

Визначення телемедицини 2: Телемедицина – це медична допомога, яка надається на відстані.

Визначення телемедицини 3: Телемедицина – це використання передових телекомунікаційних технологій для обміну медичною інформацією і надання медичних послуг через географічні, часові, соціальні та культурні бар'єри.

Визначення телемедицини 4: Всесвітня організація охорони здоров'я (ВООЗ) робить розмежування між телемедициною і телездоров'ям. Якщо телездоров'я розуміється як інтеграція телекомунікаційних систем у практику захисту та сприяння здоров'ю, тоді як телемедицина є включенням цих систем у лікувальну медицину, слід визнати, що телездоров'я більше відповідає міжнародній діяльності ВООЗ у сфері громадського здоров'я. Вона охоплює освіту для здоров'я, громадське та комунальне здоров'я, розвиток систем охорони здоров'я та епідеміологію, тоді як телемедицина більше орієнтована на клінічний аспект.

Розвиток телемедичних консультаційних систем становить значний науковий і практичний інтерес, особливо у військовій сфері. Цей огляд має на меті з'ясувати сучасний стан цих систем, висвітлити їхні можливості, обмеження і технології, що лежать в їх основі.

Існуючі системи і технології

Було розроблено кілька систем телемедицини для задоволення різних потреб охорони здоров'я, включаючи реагування на надзвичайні ситуації, планові огляди і спеціалізовані консультації. Ці системи використовують різноманітні комунікаційні технології, такі як відеоконференції, обмін миттєвими повідомленнями та пристрої дистанційного моніторингу для полегшення взаємодії між пацієнтами та медичними працівниками. Використання цих технологій має вирішальне значення в ситуаціях, коли негайний доступ до медичної допомоги неможливий, наприклад, у зонах бойових дій або у віддалених районах.

Яскравим прикладом є система телемедицини, що використовується американськими військовими, яка дозволяє проводити консультації та діагностику в режимі реального часу за допомогою захищеного супутникового зв'язку. Надійність цієї системи забезпечується її здатністю функціонувати в несприятливих умовах та інтеграцією з різними медичними пристроями, які передають дані про пацієнта віддаленим фахівцям. Основною перевагою таких систем є значне скорочення часу реагування та надання кваліфікованої медичної консультації без необхідності фізичної присутності, що підвищує шанси на сприятливий результат лікування.

Це дає змогу забезпечити фізичну присутність, тим самим підвищуючи шанси на сприятливі медичні результати.

Переваги та недоліки

Основними перевагами систем телемедицини є підвищення доступності медичної допомоги, особливо у віддалених районах, а також скорочення затримок і витрат, пов'язаних з поїздками. Ці системи також сприяють постійному моніторингу та спостереженню, що має вирішальне значення для лікування хронічних захворювань і забезпечення дотримання планів лікування.

Однак системи телемедицини стикаються з низкою проблем. Одним з основних обмежень є залежність від надійних і безпечних мереж зв'язку, які можуть бути порушені в умовах військових дій або стихійних лих. Крім того, питання, пов'язані з конфіденційністю і безпекою даних, мають першорядне значення, оскільки конфіденційна інформація про пацієнтів повинна бути захищена від несанкціонованого доступу. Інтеграція телемедицини в існуючі системи охорони здоров'я також вимагає значних інвестицій в технології та навчання, що може бути непосильним для деяких організацій.

Технічні специфікації та реалізація

Технічна реалізація телемедичних систем часто передбачає використання складних програмних платформ, які підтримують різні функціональні можливості, такі як планування прийому пацієнтів, інтеграція з електронними медичними картками та спілкування в режимі реального часу. В контексті телемедичної консультаційної системи, розробленої для військового використання, мовою програмування було обрано Python завдяки його універсальності та потужній бібліотечній підтримці.

Архітектура системи включає такі функції, як безпечна генерація токенів для автентифікації, що гарантує, що тільки авторизований персонал може отримати доступ до системи. Процес генерації токенів криптографічно захищений, що запобігає несанкціонованому передбаченню або реплікації

токенів. Це підвищує загальну безпеку системи, особливо у ворожому середовищі, де перехоплення даних є значним ризиком.

Система також має зручний інтерфейс, який дозволяє військовослужбовцям ефективно вводити свої медичні дані та керувати ними. Це включає в себе поля для особистої інформації, історії хвороби та поточного стану здоров'я, які є важливими для надання точних і своєчасних медичних консультацій.

Розвиток телемедичних консультаційних систем є значним кроком вперед у наданні доступної і своєчасної медичної допомоги в різних умовах, особливо у військових цілях. Незважаючи на те, що існують проблеми, які необхідно вирішити, такі як забезпечення надійного зв'язку і безпеки даних, переваги цих систем у поліпшенні надання медичної допомоги і результатів є значними. Майбутні дослідження і розробки повинні бути спрямовані на підвищення надійності і безпеки цих систем, а також на їх інтеграцію в більш широкі системи охорони здоров'я для максимізації їхнього впливу.

Шляхи розвитку телемедицини.

Сценарій 1: Плановий огляд за допомогою відеоконсультації[2]

Опис: Пацієнт організовує плановий огляд з лікарем первинної ланки через телемедичну платформу. **Процес:**

1. **Планування:** Пацієнт записується на прийом через телемедичний додаток.
2. **Попередня консультація:** Пацієнт заповнює анкету перед візитом.
3. **Відеоконсультація:** Лікар проводить відеодзвінок, аналізуючи історію хвороби та поточні симптоми.
4. **Подальше спостереження:** Лікар виписує електронні рецепти, призначає контрольні візити та надає освітні матеріали через додаток.

Сценарій 2: Підтримка психічного здоров’я за допомогою телефонії

Опис: Пацієнт отримує підтримку психічного здоров’я через телефонну консультацію з ліцензованим терапевтом. **Процес:**

1. **Початковий контакт:** Пацієнт зв’язується зі службою телемедицини та обирає відповідного терапевта.
2. **Бронювання сеансу:** Призначається зустріч для телефонного сеансу.
3. **Телефонна консультація:** Терапевт надає консультації та підтримку по телефону.
4. **Після сеансу:** Терапевт надсилає стратегії подолання стресу та подальші рекомендації електронною поштою або SMS.

Сценарій 3: Пропозиція щодо впровадження телемедичного програмного забезпечення

Мета: Розробити та впровадити інтегроване телемедичне програмне забезпечення для медичної мережі. **Компоненти:**

1. **Інтерфейс користувача:** Створити інтуїтивно зрозумілий інтерфейс для пацієнтів і медичних працівників.
2. **Інтеграція відео та аудіо:** Забезпечити високоякісні відео- та аудіоможливості для безперервних консультацій.
3. **Електронні медичні записи (EHR):** Надати безпечний доступ до записів пацієнтів.
4. **Планування зустрічей:** Розробити систему планування з автоматичними нагадуваннями та підтвердженнями.
5. **Рецептурні послуги:** Впровадити електронні рецепти з інтеграцією аптек.
6. **Безпека даних:** Забезпечити надійний захист даних пацієнтів і відповідність нормативним вимогам.
7. **Аналітика та звітність:** Включити інструменти для аналізу даних і створення звітів для моніторингу використання та результатів.

Ці сценарії демонструють, як телемедицина може покращити медичне обслуговування і оптимізувати процеси надання медичних послуг за допомогою сучасних технологічних рішень, мною був обраний 3 сценарій розвитку й адаптації для військових потреб.

1.2. Використання Python для швидкої розробки програмних засобів

Використання Python у поєднанні з бібліотекою KivyMD представляє чудовий підхід до розробки програм. Python, відомий своєю повсюдністю та можливістю адаптації в сучасному програмуванні, цінується своєю чіткістю, зрозумілістю та великим набором бібліотек, що сприяє швидкій та ефективній

розробці програм. KivyMD, розширення Kivy, створене для сучасного дизайну інтерфейсу користувача з принципами матеріального дизайну, стає ключовим інструментом у цій справі.

KivyMD: огляд і його переваги

KivyMD дає змогу створювати естетично привабливі та інтуїтивно зрозумілі інтерфейси користувача, використовуючи компоненти Material Design, щоб надати додаткам сучасної візуальної привабливості відповідно до переважаючих парадигм дизайну UI/UX [11]. Крім того, програми, розроблені з використанням KivyMD, можуть похвалитися сумісністю з кількома платформами, бездоганно працюють у різноманітних операційних системах, охоплюючи середовища Windows, macOS, Linux, iOS та Android, тим самим демонструючи придатність Python і KivyMD для розробки міжплатформних рішень[12].

Простота та оперативність у розробці

Об’єднання Python і KivyMD оптимізує життєвий цикл розробки, сприяючи створенню надійних програм із зменшеними витратами коду. Репутація Python завдяки простоті та легкості оволодіння робить його доступним навіть для розробників-початківців, що ще більше підвищує ефективність процесів розробки програм.

Застосування в телемедицині

Використання Python і KivyMD для створення телемедичних додатків обіцяє суттєве підвищення доступності охорони здоров’я. Ці програми можуть включати спектр функцій, включаючи, але не обмежуючись:

1. Інтуїтивно зрозумілі інтерфейси, що сприяють бездоганній взаємодії як для пацієнтів, так і для медичних працівників.

2. Інтеграція можливостей відео- та аудіоконференцій високої точності для полегшення дистанційних консультацій.
3. Створення механізмів безпечного доступу до електронних медичних записів пацієнтів із забезпеченням конфіденційності та цілісності.
4. Функції запису до певного лікаря, не витрачаючи надмірного часу для з’ясування графіків та іншого.
5. Можливість використати довідникову частину програми, де розміщено багато корисних функцій:
   - Калькулятор калорій.
   - Вправи для підтримання фізичної форми та реабілітації після поранень.
   - Корисні статті, де описані додаткові бади для здоров’я.
6. Впровадження надійних заходів безпеки даних для захисту конфіденційної інформації пацієнтів від несанкціонованого доступу чи злому.

Переваги сучасних технологій Python

Поширення сучасних технологій Python, втіленням яких є бібліотека KivyMD, надає розробникам чудові інструменти для розробки складних і зручних програмних рішень. Ці технології дають змогу розробникам швидко реагувати на зміну потреб користувачів і бездоганно інтегрувати нові функції. Використання Python для розробки телемедичних додатків підвищує ефективність і доступність медичних послуг, вирішуючи вимоги, пов’язані з сучасними ландшафтами охорони здоров’я.

2. Архітектура та розробка програми

В роботі використано Visual Studio Code, мови програмування: python, sql, бібліотеки: kivy_MD, kivy, google_auth, email, uix, webbrowser. Розробка послідовності роботи програми зайняла найбільше часу, тому що для нашої програми важливість кросплатформеність, оптимізація, легкість сприйняття.

Архітектура з чіткими рамками забезпечить схему, структурність, поведінку, взаємодію компонентів програми. Також великим плюсом буде можливість маштабованості, тому варто використати розбиття на блоки, в такому випадку при видаленні або додаванні функціоналу ми не будемо втрачати здібність до роботи всієї програми. Таким чином, архітектура програмного забезпечення життєво важлива для створення масштабованих, придатних для обслуговування та високопродуктивних програм, гарантуючи, що вони можуть розвиватися та адаптуватися з часом відповідно до мінливих вимог і викликів.

2.1 Створення віртуального середовища

Віртуальне середовище(рисунок 2.1), перше що створюється при розробці нового програмного продукту. За допомогою цього інструменту ми ізолюємо залежність проекту від версій бібліотек, ми встановлюємо статичні залежності певних версій, також ми забезпечуємо безпеку проекту, уникаючи нових версій не надійних бібліотек.

1. Встановлення `virtualenv`:

```sh
pip install virtualenv
```

2. Створення віртуального середовища:

```sh
python -m virtualenv kivy_venv
```

3. Активізація віртуального середовища:

- На Windows:

```sh
kivy_venv\Scripts\activate
```

- На MacOS/Linux:

```sh
source kivy_venv/bin/activate
```

4. Встановлення Kivy:

```sh
pip install kivy[base] kivy_examples
```

5. Перевірка встановлення:

```sh
python -m kivy.examples.demo.touch
```

Рисунок 2.1 - Створення віртуального середовища.

2.2 Поділ на блоки

Проект був поділений на 7 блоків, кожен блок має свій спектр функцій. Кожен файл має відповідну назву для зручності.

1)input.py

Наданий код демонструє, як створюється базовий інтерфейс автентифікації користувача за допомогою бібліотеки Pythons KivyMD. Основна увага приділяється класу TestScreen, який діє як екран програми та успадковує Screen.

```python
class TestScreen(Screen):
    def __init__(self, **kwargs):
        super(TestScreen, self).__init__(**kwargs)
        self.orientation = 'vertical'

        # Adding labels and text fields
        self.label = MDLabel(text="Вітаю вас!", halign='center')
        self.email_input = MDTextField(
            hint_text="Введіть вашу пошту",
            pos_hint={'center_x': 0.5, 'center_y': 0.7},)
        self.password_input = MDTextField(hint_text="Пароль",
            pos_hint={'center_x': 0.5, 'center_y': 0.6},
            password=True,)

        # Load saved credentials
        saved_credentials = load_credentials()
        if saved_credentials:
            self.email_input.text = saved_credentials.get('email', '')
            self.password_input.text = saved_credentials.get('password', '')

        # Adding a vertical BoxLayout for buttons
        buttons_layout = BoxLayout(orientation='vertical')
```

Рисунок 2.2 – Демонстраціє класу TestScreen input.py.

У класі TestScreen(рисунок 2.2) налаштовано різні елементи інтерфейсу користувача, наприклад MDLabel для показу вітального повідомлення та MDTextField для введення даних користувача, таких як електронна адреса та пароль. Для покращення взаємодії з користувачем використовується функція load_credentials для автоматичного заповнення цих полів збереженою інформацією після ініціалізації.

Для навігації програмою включено дві кнопки за допомогою MDRaisedButton; один для переходу до вікна (метод go_to_next_window), а інший для збереження даних користувача (метод save_credentials). Ці кнопки розташовані вертикально в BoxLayout для макета.

Клас MyApp, який розширює MDApp, обробляє створення та запуск програми, повертаючи примірник TestScreen.

Стек технологій включає KivyMD для створення інтерфейсу користувача з компонентами матеріального дизайну та спеціальний модуль авторизації для керування сховищем облікових даних. Результатом цього налаштування є інтерфейс із функціями автентифікації користувача. Архітектура зосереджена на простоті, ясності та гнучкості, щоб гарантувати, що програму легко підтримувати та розширювати.

2) auth.py

В фрагменті коду(рисунок 2.3) було використано переваги мови python, а саме керування даними користувачів відбувається шляхом обробки JSON. Файл який окремо зберігає ці дані має назву: “credentials.json” від зберігає дані користувачів, точніше: пошту та паролі. Маємо 2 варіації використання: “save_credentials” і “load_credentials”

“save_credentials” – приймає дані, інкапсулює їх в структурований словник, потім викликає метод “json.dump”, щоб словник перейшов в JSON, який стає константою в «CREDENTIALS_FILE». Цей процес забезпечує постійне зберігання даних автентифікації користувача в структурованому форматі.

«load_credentials» - виконує завдання отримання збережених облікових даних із визначеного файлу JSON. Спочатку робиться спроба відкрити файл credentials.json у режимі читання. У разі успішного відкриття файлу його вміст десеріалізується(Цей процес, при якому дані, збережені у форматі, так само як JSON, XML або бінарний, перетворюються навпаки в об’єкти пам’яті комп’ютера. Це зворотна сторона серіалізації, яка змінює стан об'єкта в потоці байтів або рядку для зберігання або передачі.) за допомогою методу «json.load», у результаті чого відбувається реконструкція словника «облікових даних». Згодом цей словник, що містить дані автентифікації користувача, повертається абоненту для подальшої обробки. Однак у випадку виняткової

ситуації «FileNotFoundError», яка вказує на відсутність зазначеного файлу, повертається значення «Немає», що вказує на відсутність збережених облікових даних.

```
import json

CREDENTIALS_FILE = 'credentials.json'

def save_credentials(email, password):
    credentials = {
        'email': email,
        'password': password,
    }

    with open(CREDENTIALS_FILE, 'w') as file:
        json.dump(credentials, file)

def load_credentials():
    try:
        with open(CREDENTIALS_FILE, 'r') as file:
            credentials = json.load(file)
        return credentials
    except FileNotFoundError:
        return None
```

Рисунок 2.3 – вміст auth.py.

3) foto.py

Цей блок активно використовує компоненти MD(material design), створюється інтуїтивно-зрозумілий інтерфейс користувача. Більша частина коду інкапсульована в клас PhotoScreen,який бере свої властивості через успадкування від класу Screeen Kivy. В цьому класі реалізовано графічний

інтерфейс, який об’єднує різні поля введення та кнопки для взаємодії з користувачем.

Інтерфейс налічує декілька віджетів MDTextField, для збору особистих та медичних даних, такої як ім’я, прізвище, по батькові, зріст, вага, вік, військовий статус, самопочуття, група крові та коментарі. Два віджети MDRaisedButton полегшують завантаження фотографії та надсилання зібраних даних.

```
if not credentials:
    Snackbar(text="Failed to authenticate. Please check your creden
    return

subject = "Photo from KivyMD App"
body = "Attached is the photo submitted from the KivyMD App."

try:
    msg = MIMEMultipart()
    msg.attach(MIMEText(body, 'plain'))

    # Attach the photo
    with open(self.image_path, 'rb') as file:
        img = MIMEImage(file.read(), name='photo.jpg')
    msg.attach(img)

    # Use Gmail's SMTP server
    server = smtplib.SMTP('smtp.gmail.com', 587)
    server.starttls()
```

Рисунок 2.4 – вміст foto.py.

Наступним включається MDFileManager, дозволяє користувачу вибрати файлові зображення з бібліотеки свого пристрою, відображення відбувається за допомогою Snackbar, для підтвердження. Процес подання передбачає надсилання електронного листа з прикріпленою вибраною фотографією. Ця операція покладається на Google OAuth2 для автентифікації користувача та використовує бібліотеку smtplib для надсилання електронних листів через SMTP-сервер Gmail. Керування маркерами автентифікації здійснюється через

модулі google_auth_oauthlib.flow і google.oauth2.credentials, що забезпечує безпечний доступ.

Метод submit_photo проводить автентифікацію користувача за допомогою гугл акаунту, створюється електроний лист із вкладеним зображенням, який надсилає на відповідну адресу. Все це має гарантовано спрацювати за допомогою сповіщення SnackBar. (рисунок 2.4)

В цілому отримуємо архітектуру, яка підкреслює, що ми використали різні технології для створення додатку. Універсальність Python у поєднанні з надійними компонентами інтерфейсу KivyMD і безпечними механізмами автентифікації демонструє ефективний підхід до розробки сучасних програм.

4)Icons.py

В цьому файлі створюємо інтерфейс користувача, що містить піктограми MD. Використовується бібліотека Kivy, для декларативної побудови інтерфейсу. Архітектура побудована через шаблон MVC(Model-View-Controller), де логіка розписується в окремих класах.

Ключовим є імпортований модуль Builder, для завантаження визначень інтерфейсу користувача з формату рядка, «StringProperty» для визначення властивостей рядка в компонентах інтерфейсу користувача та класи «Screen» і «MDApp» для керування екраном і ініціалізація програми відповідно. Також спеціальним елементов в цьому випадку є OneLineIconListItem» від KivyMD, як окремий елемент піктограми.

метод 'Builder.load_string' використовується для завантаження визначення інтерфейсу користувача з формату рядка, що визначає макет і поведінку екрана 'PreviousMDIcons', включаючи організацію функції пошуку значків і відображення значків матеріального дизайну в RecycleView.

Взаємо пов’язані класи «CustomOneLineIconListItem» та «OneLineIconListItem», включають властивість «icon» для вибраного значка. «PreviousMDIcons» відпвідаж за заповнення списків піктограм, вибраних користувачем.

```
class CustomOneLineIconListItem(OneLineIconListItem):
    icon = StringProperty()


class PreviousMDIcons(Screen):

    def set_list_md_icons(self, text="", search=False):
        '''Builds a list of icons for the screen MDIcons.'''

        def add_icon_item(name_icon):
            self.ids.rv.data.append(
                {
                    "viewclass": "CustomOneLineIconListItem",
                    "icon": name_icon,
                    "text": name_icon,
                    "callback": lambda x: x,
                }
            )
```

Рисунок 2.5 – вміст Icons.py.

Клас «MainApp» служить точкою входу для програми, успадковуючи клас «MDApp», наданий фреймворком KivyMD. У цьому класі головний екран програми створюється як екземпляр класу «PreviousMDIcons», який згодом повертається методом «build» для візуалізації інтерфейсу користувача. Крім того, метод 'on_start' перевизначено, щоб запускати ініціалізацію списку значків під час запуску програми.

Загалом, сценарій демонструє інтеграцію фреймворків Kivy та KivyMD для розробки динамічних і візуально привабливих інтерфейсів користувача,

підкреслюючи модульний і декларативний підхід, який сприяють цим фреймворкам у розробці додатків Python.

5) navigator.py

Клас YourApp, успадкований від MDApp, використовує мову Kivy для визначення інтерфейсу користувача в рядку KV. Тут через віджет MDbottomNavigation, створюється навігаційна панель знизу з декількома вкладками, у кожної свій функціонал, поділені вони на екстрена допомога, запис до лікаря, довідник, калькулятор калорій.

```
KV = '''

    MDBottomNavigation:

        MDBottomNavigationItem:
            name: 'screen 1'
            text: 'Невідкладна допомога'
            icon: 'account-tie-voice'
            badge_icon: "numeric-10"

            MDRaisedButton:
                text: 'Підключитися до конференції'
                halign: 'center'
                size_hint_x: 0.5  # Встановлюємо ширину кнопки в 80% від ширин
                size_hint_y: 0.5
                font_size: '24sp'  # Змінюємо розмір шрифту на 24sp
                md_bg_color: (1, 0, 0, 1)  # Червоний колір фону (обведення) к
                line_color: (0, 0, 0, 1) # Чорний колір обведення шрифту
                pos_hint: {'center_x': 0.5, 'center_y': 0.5}
                on_release: app.join_google_meet()
```

Рисунок 2.6 – вміст navigator.py.

На першій вкладці з написом «Невідкладна допомога» є кнопка для підключення до конференції Google Meet. Ця функція досягається за допомогою методу join_google_meet, який відкриває вказану URL-адресу у веб-браузері за допомогою модуля веб-браузера.

Друга вкладка «Запис до лікаря» містить кнопки для різних медичних спеціальностей (хірург, офтальмолог, терапевт). Метод show_doctor_schedule

підключається до бази даних SQLite для отримання та відображення розкладу вибраного лікаря. Цей метод встановлює з’єднання з базою даних, виконує SQL-запит для отримання розкладу лікаря на основі спеціалізації та друкує результат.

Третя вкладка «Довідник» представляє посібник із мітками та зображеннями, завантаженими з URL-адрес. Остання вкладка, «Калькулятор калорій», зарезервована для калькулятора калорій, наразі відображається мітка-заповнювач.

6) shudle.py

Ця вкладка відповідає за роботу з sql таблицею, детальніше про це в розділі SQL.

7) main.py

Це головний файл, в якому прописаний класс “MyApp”, в ній реалізовано функцію Build, через яку реалізовану виклик 2 під файлів, які мають назву input_screen, photo_screen.

2.3 Використання SQL

В проекті SQL, використовується для можливості запису до лікаря.

Файл в проекті, який за це відповідає несе назву shudle.py.

```python
import sqlite3

# Підключення до бази даних SQLite
conn = sqlite3.connect('doctors.db')
conn = sqlite3.connect('schedules.db')
c = conn.cursor()

# Створення таблиці лікарів
c.execute('''CREATE TABLE doctors
             (id INTEGER PRIMARY KEY, name TEXT, specialization TEXT)''')

# Створення таблиці розкладів лікарів
c.execute('''CREATE TABLE schedules
             (doctor_id INTEGER, day TEXT, time TEXT)''')

# Додавання даних про лікарів
c.execute("INSERT INTO doctors (name, specialization) VALUES ('SMIT', 'Терапевт')")
c.execute("INSERT INTO doctors (name, specialization) VALUES ('DJONTH', 'Хірург')")
c.execute("INSERT INTO doctors (name, specialization) VALUES ('Vika', 'Окуліст')")

# Додавання розкладу лікарів
c.execute("INSERT INTO schedules (doctor_id, day, time) VALUES (1, 'Понеділок', '10:00')")
c.execute("INSERT INTO schedules (doctor_id, day, time) VALUES (1, 'Вівторок', '12:00')")
c.execute("INSERT INTO schedules (doctor_id, day, time) VALUES (2, 'Середа', '14:00')")
c.execute("INSERT INTO schedules (doctor_id, day, time) VALUES (2, 'Середа', '14:00')")
c.execute("INSERT INTO schedules (doctor_id, day, time) VALUES (2, 'Середа', '16:00')")
c.execute("INSERT INTO schedules (doctor_id, day, time) VALUES (2, 'Середа', '20:00')")
```

Рисунок 2.7 - shudle.py.

Файл містить імпорт самої бібліотеки SQL, використовується SQLlite3.

Далі відбувається під’єднання до таблиць: “doctors.db”, “schedules.db”, саа структура таблиць має наступний вигляд:

1) doctors.db -> name(ім’я), specialization(спеціалізація).
2) schedules -> doctor_id(айді), day(день роботи на тижні), time(робочі години).

За допомогою команди введення, insert, ми додаємо данні в таблиці, які далі буде використано в програмі. В файлі navigator.py, ми приєднуємося до таблиці(рисунок 2.8)

```python
def show_doctor_schedule(self, doctor):
    # Підключення до бази даних MySQL
    conn = sqlite3.connect('doctors.db')
    conn = sqlite3.connect('schedules.db')
```

Рисунок 2.8 – приєднання до таблиць navigator.py.

```
# Створення курсора для виконання запитів
cursor = conn.cursor()

# Виконання SQL-запиту для отримання розкладу роботи лікаря
cursor.execute("""SELECT time FROM schedules INNER JOIN doctors ON schedules.doctor_id = doctors.id WHERE doctors.specialization = ?""", (doctor,))
```

Рисунок 2.9 – виконуємо об’єднаний запит до обох таблиць одночасно navigator.py.

На рисунку 2.9 зображено, запит до таблиць, який виконується під час натискання на кнопку виклику запису по часу до певного лікаря рисунок 2.10

```
MDRaisedButton:
    text: 'Окуліст'
    size_hint: None, None
    size_hint_x: 0.2
    size_hint_y: 0.2
    pos_hint: {'center_x': 0.5, 'center_y': 0.5}

    on_release: app.show_doctor_schedule("Окуліст")
```

Рисунок 2.10 – виклик функції показу розкладу певного лікаря.

2.4 Інтерфейс

Перед розробкою інтерфейсу необхідно визначити критерії, за якими буде збиратися інформація про користувача (Рисунок 2.11):

- Ім’я
- Прізвище
- По-батькові
- Зріст
- Вага
- Вік
- Статус
- Група крові
- Коментарі

Статус відображає поточний стан військовослужбовця, зокрема участь у бойових діях, перебування на ротації, навчання чи реабілітацію. Ця інформація буде корисною для подальшого надання медичної допомоги, оцінки калорійності раціону та загальних медичних консультацій.

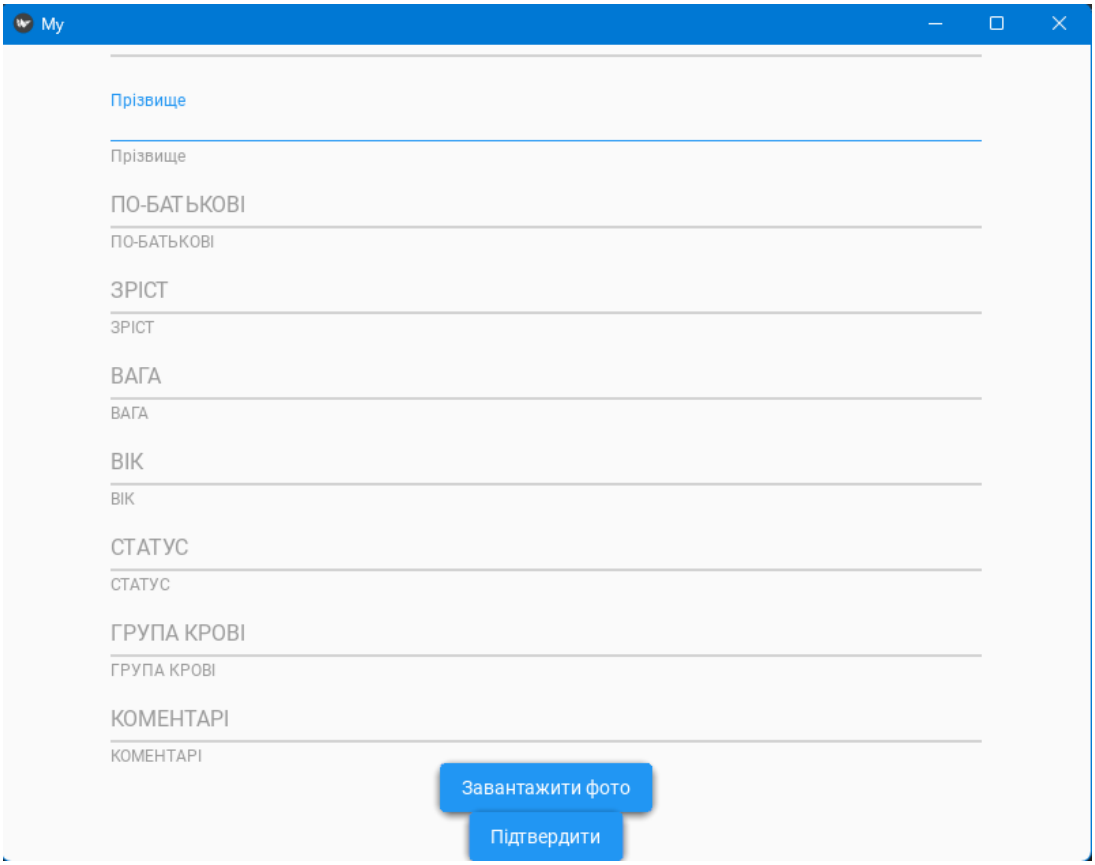

Рисунок 2.11 – Заповнення анкети, інтерфейс користувача.

Важливою частиною є кнопка – завантажити фото, вона потрібна, щоб військовий мій відправити всі фотографії діагнозів, документів, медичних висновків на сервер та не носити з собою в паперовому варіанті(рисунок 2.12).

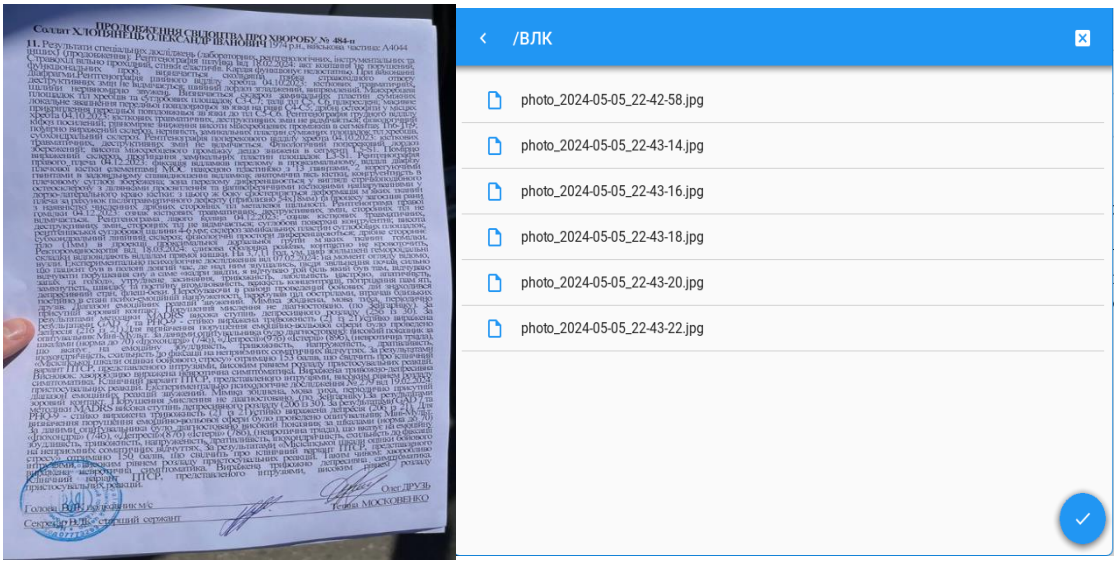

Рисунок 2.12 – Реалізація завантаження фото.

Програма частина (рисунок 2.13):

```
        layout = BoxLayout(orientation='vertical')
        layout.add_widget(self.name_input)
        layout.add_widget(self.surname_input)
        layout.add_widget(self.height_input)
        layout.add_widget(self.weight_input)
        layout.add_widget(self.age_input)
        layout.add_widget(self.militarystatus_input)
        layout.add_widget(self.wellbeing_input)
        layout.add_widget(self.blood_input)
        layout.add_widget(self.comments_input)
        layout.add_widget(self.button_upload)
        layout.add_widget(self.button_submit)

        self.add_widget(layout)

    def show_file_chooser(self, *args):
        self.file_manager.show('/')  # Starting path for the file manager

    def select_path(self, path):
        self.file_manager.close()
        Snackbar(text=f"Selected file: {path}").open()
        self.image_path = path
```

Рисунок 2.13 – Реалізація анкети та підвантаження фото мовою python.

Наведений фрагмент коду ініціалізує макет інтерфейсу користувача з використанням BoxLayout з вертикальною орієнтацією. У цьому макеті послідовно додаються різні віджети введення, включаючи поля для імені, прізвища, зросту, ваги, віку, військового статусу, самопочуття, групи крові та коментарів. Крім того, до макету додаються дві кнопки button_upload та button_submit. Створений макет додається до основної структури віджету за допомогою self.add_widget(layout).

Метод show_file_chooser призначений для відкриття файлового менеджера, ініціюючи процес перегляду файлів з кореневого каталогу ('/'). Цей метод полегшує взаємодію користувача з файловою системою для вибору файлів.

Метод select_path керує процесом вибору файлу. Після вибору шляху до файлу файловий менеджер закривається, а на панелі швидкого доступу з'являється сповіщення, у якому відображається шлях до вибраного файлу. Вибраний шлях до файлу зберігається в атрибуті екземпляра self.image_path для подальшого використання.

Цей код ефективно налаштовує зручний інтерфейс для введення даних і вибору файлів, гарантуючи, що вся необхідна інформація про користувача може бути зібрана, а файли можуть бути завантажені за потреби. Дизайн підкреслює модульність і ясність, полегшуючи як взаємодію з користувачем, так і подальше обслуговування або розширення коду.

2.4.1 Невідкладна допомога

В деяких сценаріях може виникнути ситуація, коли військова особа має необхідність у медичній допомозі для колеги, але власних медичних навичок їй недостатньо. Для таких ситуацій реалізована програмна екстрена допомога, яка забезпечує можливість зв'язку військового з черговим військовим лікарем через онлайн-зустріч. Відмінність цього підходу від звичайного телефонного зв'язку полягає в тому, що учасники можуть спілкуватися через відеоконференцію, що дозволяє уточнити проблему та прискорити процес її вирішення(рисунок 2.14). Після натискання на відповідну кнопку військового автоматично перенаправляється до відеоконференції, яка відбувається через Google Meet. Цей сервіс відмінний за своєю зручністю, простотою використання та ефективним використанням мережевого трафіку.

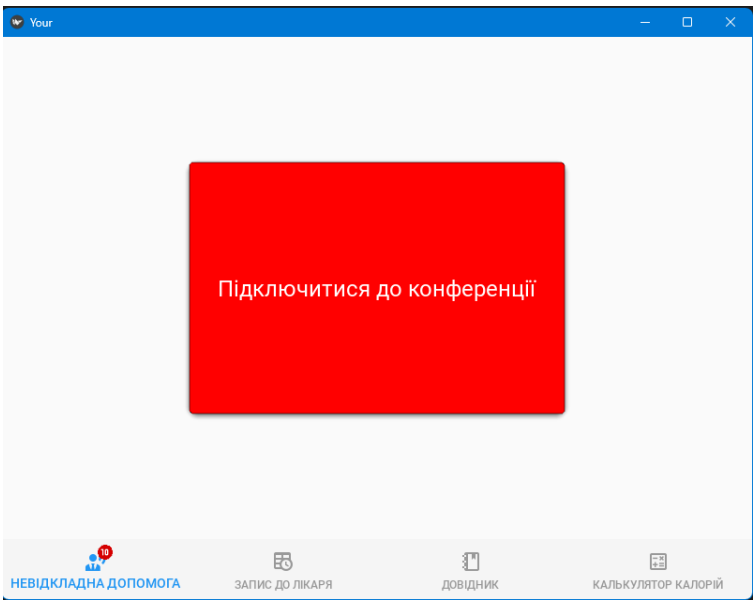

Рисунок 2.14 – Невідкладна допомога

2.4.2 Запис до лікаря.

Дуже важливим аспектом для військових є своєчасна допомога, як на “нулі”, так й на інших лініях оборони та навіть в тилу. Тому додатково реалізовано запис до потрібного лікаря через додаток.

Блок схема реалізації виглядає таким чином(рисунок 2.15):

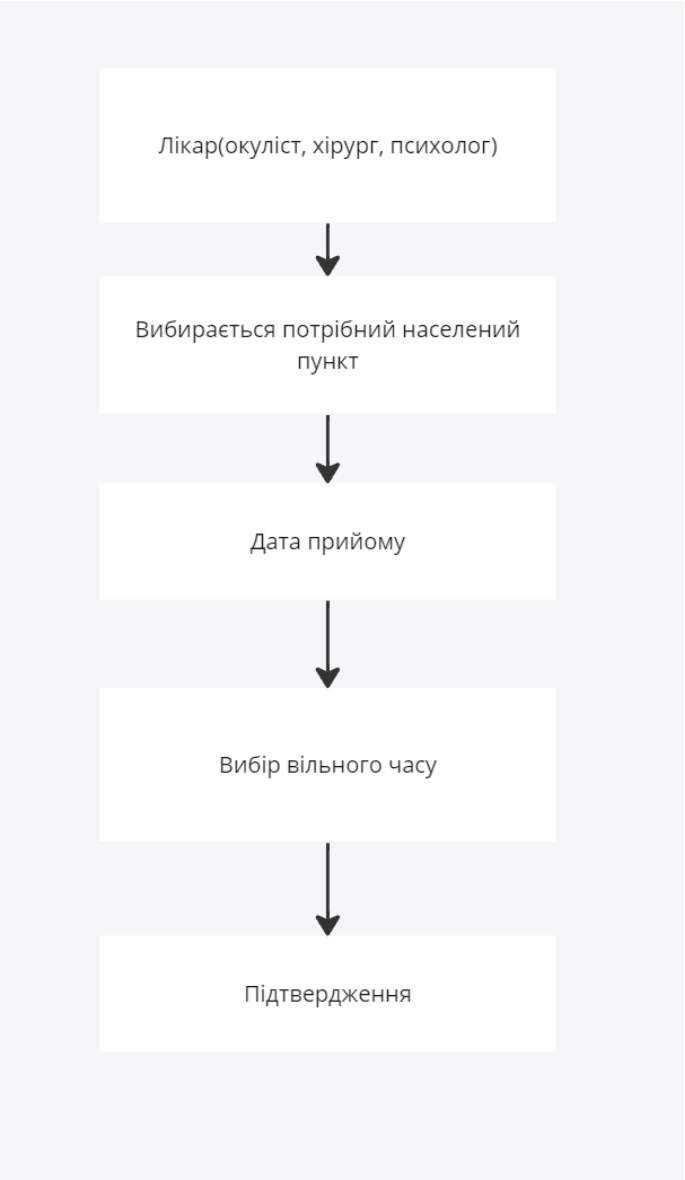

Рисунок 2.15 – Блок схема звернення до лікаря.

Після звернення запускається механізм, підготовки до прийому(рисунок 2.16):

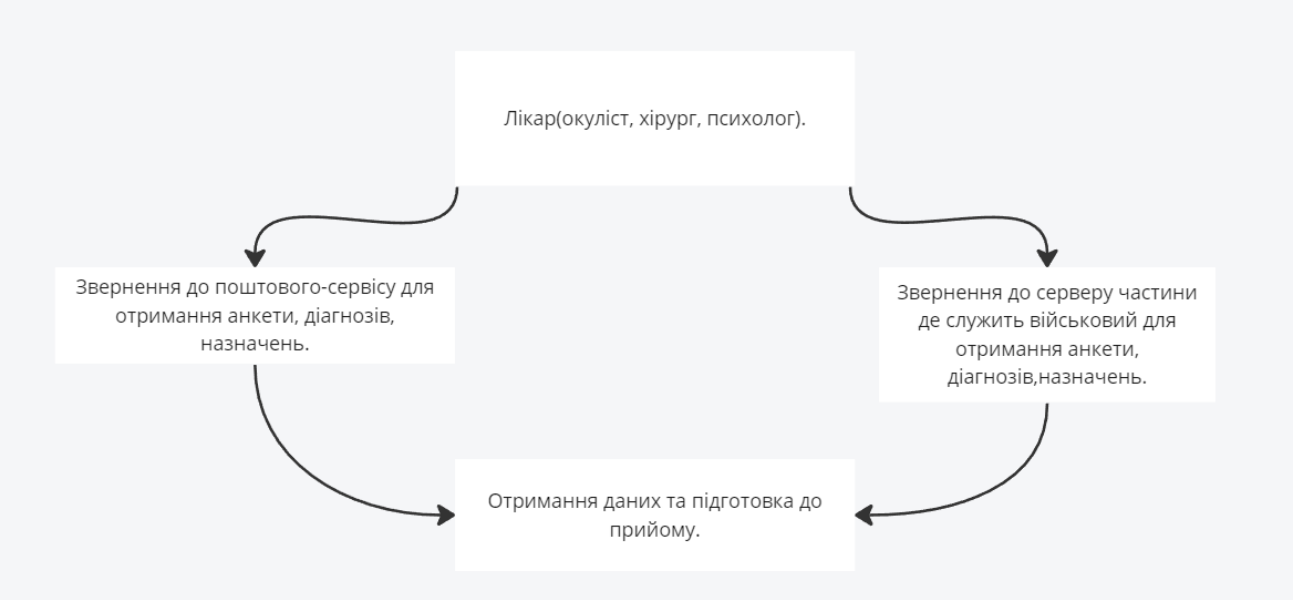

Рисунок 2.16 – Блок схема роботи лікаря після звернення.

Запис до лікаря через додаток, візуальне представлення(рисунок 2.17):

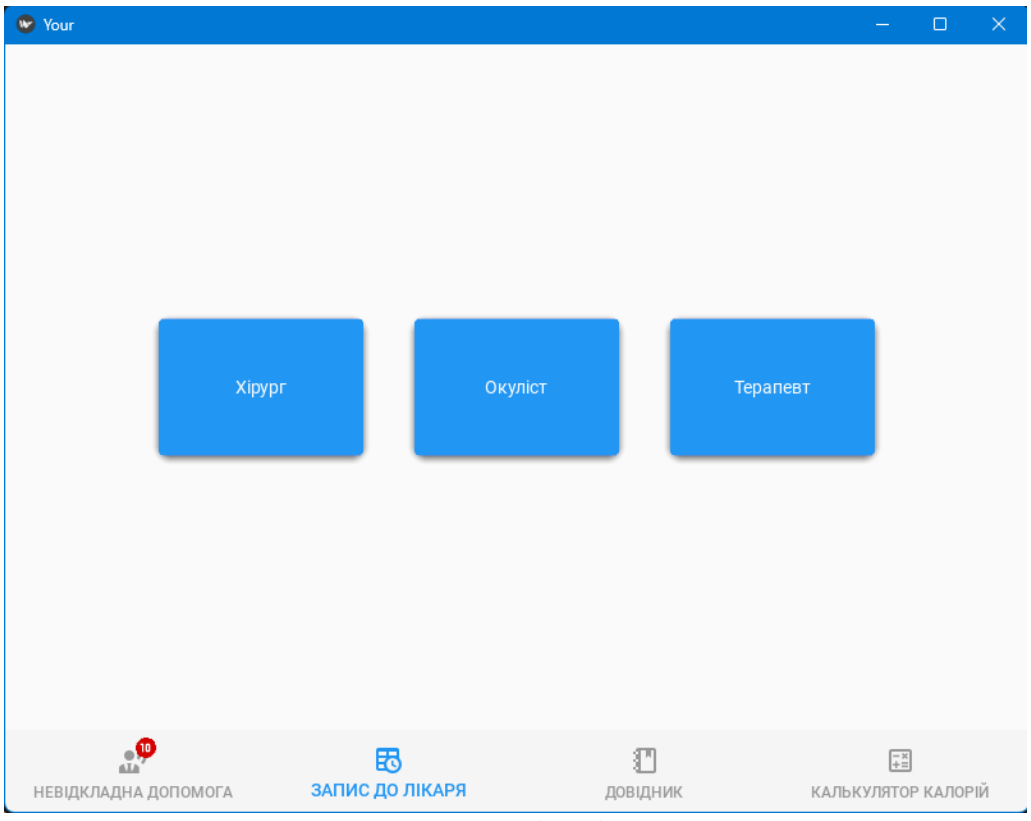

Рисунок 2.17 – Реалізація запису в додатку.

Запис реалізовано за рахунок створення 2 таблиць:

1)doctors.db – в цій таблиці реалізовано спеціальності та їх ПІБ.

2)schedules.db – це динамічна таблиця, графіку роботи лікарів.

Якщо би графік було реалізовано в спільній таблиці то ми би мали дизбаланс в таблиці, тобто стовчик №1 пункт-1, а стовпчик №2 мав би мінімум 3 пукти, тобто місцями таблиця була би не оптимізована й мала так звані переповнення.

Сам запит виконується об’єднанням 2 запитів до sql таблиці, тому нам не потрібно реалізовувати 2 окремих запити для отримання даних(рисунок 2.18):

```
def show_doctor_schedule(self, doctor):
    # Підключення до бази даних MySQL
    conn = sqlite3.connect('doctors.db')
    conn = sqlite3.connect('schedules.db')

    # Створення курсора для виконання запитів
    cursor = conn.cursor()

    # Виконання SQL-запиту для отримання розкладу роботи лікаря
    cursor.execute("""SELECT time FROM schedules INNER JOIN doctors ON schedules.doctor_id = doctors.id WHERE doctors.specialization = ?""", (doctor,))
```

Рисунок 2.18 – Реалізація звернення до таблиць.

2.4.3 Калькулятор калорій + БЖВ:

Сам розрахунок відбувається за рахунок таких параметрів, які військовий вказує у формі рисунки 2.19,2.20,2.21:

- Ріст.
- Вага.
- Вік.
- Військовий статус– включає в себе вибір: бойові дії, на ротації, підготування, на реабілітації.
- Загальний стан - включає в себе вибір: погано, добре, нормально.

Формула 17[6] включає власні напрацювання. Тому що містяться не стандартні параметри, такі як:

- Військовий статус – включає в себе вибір: in combat, on rotation, training and preparation, rehabilitation.
- Загальний стан - включає в себе вибір: погано, добре, нормально.

При виборі підпунктів, число калорій та БЖВ, також підлаштовується.

```
def calculate_calories(self):
    base_calories = self.calculate_base_calories()
    military_bonus = self.calculate_military_bonus()
    well_being_adjustment = self.calculate_well_being_adjustment()
    self.total_calories = base_calories + military_bonus + well_being_adjustment
    self.bju = self.calculate_bju_from_calories()
    return self.total_calories
```

Рисунок 2.19 – Формула розрахунку, яка включає важливі аспекти: military_bonus, well_being_adjustment.

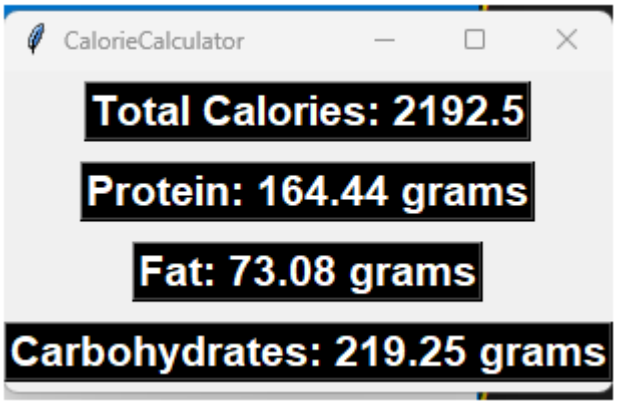

Рисунок 2.20 – Приклад калькулятора калорій.

```
class CalorieCalculator:


    def __init__(self, height, weight, age, military_status, well_being):
        self.root = tk.Tk()
        self.root.title("CalorieCalculator")
        self.height = height
        self.weight = weight
        self.age = age
        self.military_status = military_status
        self.well_being = well_being
        self.total_calories_label = tk.Label(self.root, text=f"Total Calories: {self.calculate_calories()}")
        protein, fat, carbohydrates = self.calculate_bju_from_calories()
        self.protein_label = tk.Label(self.root, text=f"Protein: {protein:.2f} grams")
        self.fat_label = tk.Label(self.root, text=f"Fat: {fat:.2f} grams")
        self.carb_label = tk.Label(self.root, text=f"Carbohydrates: {carbohydrates:.2f} grams")
```

Рисунок 2.21– Опис калькулятора калорій.

2.4.4 Довідник:

В останньому розділі описано довідник, який містить в собі інформацію для військових(рисунок 2.22).

Довідник включає підсписок тем[5],[6],[7],[8]:

1. Розділи за тематикою: Довідник розділений на категорії за тематикою, що дозволяє військовим швидко знаходити необхідну інформацію.
2. Інструкції та процедури: В довіднику містяться інструкції та процедури з різних аспектів військової служби, такі як екстрені ситуації, безпека, збройна тактика, медична допомога тощо.
3. Військові стандарти та нормативи: Програма надає інформацію про військові стандарти, вимоги та нормативи, які військовослужбовці повинні виконувати.
4. Тактичні рекомендації: Довідник містить тактичні рекомендації та поради щодо ефективного виконання різних військових завдань та операцій.
5. Медична допомога та перша допомога: Інформація про надання медичної допомоги та першої допомоги у військових умовах, включаючи методи надання допомоги пораненим та вразливим особам.
6. Комунікація та зв'язок: Опис різних методів комунікації та зв'язку у військових умовах, включаючи використання радіо, сигнальних пристроїв тощо.
7. Екіпірування та зброя: Інформація про екіпірування, військову форму, зброю та бойову техніку, включаючи правила використання та обслуговування.

8. Експертні поради та досвід: Довідник може містити експертні поради та досвід військових фахівців, які допомагають у вирішенні практичних завдань та ситуацій.

9. Посилання на додаткові ресурси: Програма може містити посилання на додаткові ресурси, такі як військові стандарти, навчальні матеріали, відео-уроки тощо, для отримання додаткової інформації та навчання.

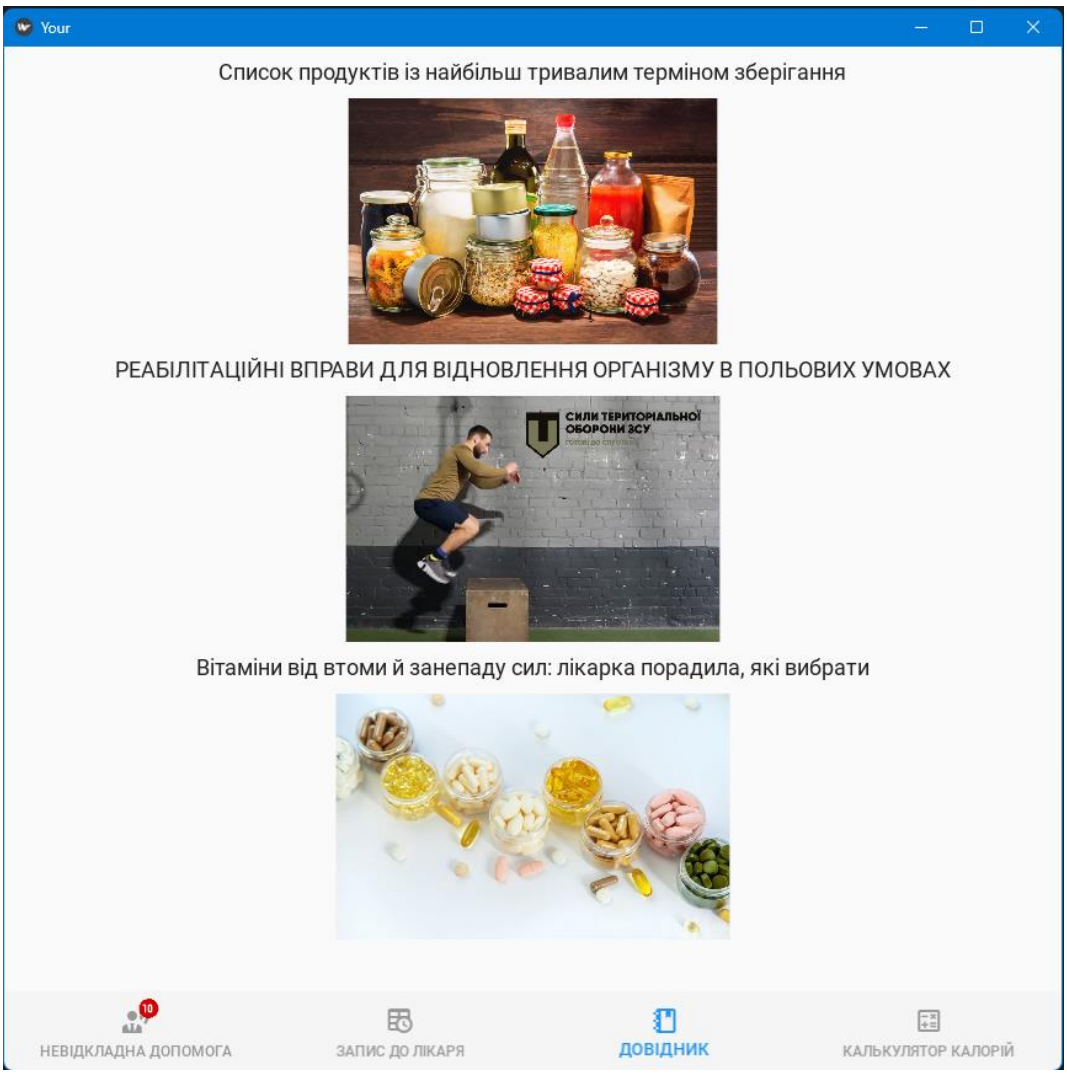

Рисунок 2.22 – Реалізація вкладки довідника.

2.4.5 Передача даних

Передача даних матиме наступний вигляд(рисунок 2.23):

1) Підготовка, збір інформації.
2) Підготовка відправлення, збір фото та анкетних даних в папку, яка буде архівуватися, сам архів буде запаролено, тобто використати інформацію зможуть тільки відповідальні люди з частини або лікарі, в яких буде доступ.
3) Відправка на пошту-сервіс, для чого це робиться?
    - Комфортно потім з пошти перевантажувати файл на хмарне сховище.
    - Також лікарі можуть використовувати напряму пошту для пришвидшення відкриття особових справ військових.
4) З хмарного сховища буде відбуватися підвантаження на сервер частини та резервний сервер частини.

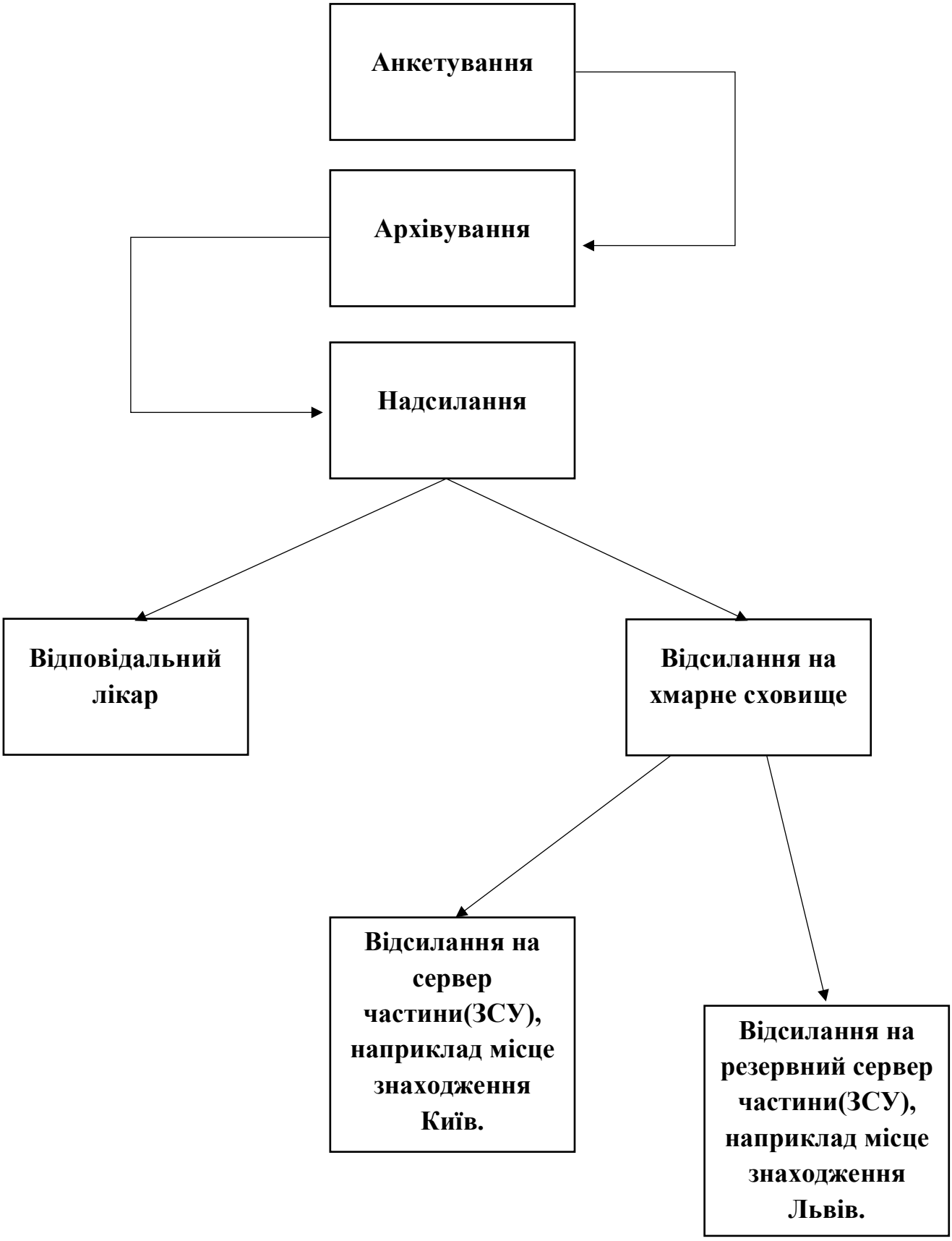

Рисунок 2.23 – Візуалізовано схематично, передачу даних в програмі, від анкети до серверів частини.

Схема роботи серверів:

1. Реплікація:
   - Використання схеми майстер-майстер реплікації: тобто обидва сервери є майстрами, вони роблять дублікати та обмінюються даними один з одним. Такий варіант є дуже надійним, бо навіть при повному знищені 1 із об’єктів, дані залишаються в повному обсязі.
   - Використання таких баз даних як: MySQL, PostgreSQL в яких закладено такі технології як: MySQL Replication, PostgreSQL Streaming Replication.
2. Постійний моніторинг серверів:
- Моніторинг стану серверів, використання таких систем як: Prometheus, Nagios.

Налаштування сповіщень про порушення роботи.

3. Механізм переключення:

Використання таких алгоритмів як: “Heartbeat”, “Keepalived”, вони моніторять стан серверів та відповідають за автоматичне переключення, у випадку недоступності одного з серверів.

4. Перенаправлення трафіку:
   - Використання технології Anycast, перенаправлення трафіку на доступний сервер.
5. Відновлення роботи після відновлення основого сервера:
   - Відкат налаштувань сервера до робочих показників.
   - Механізм повернення трафіку до відновленого робочого серверу.

3. Безпека

3.1 Метод розповсюдження програми, який виключає використання третіми особами та збереження токену.

Для визначення моменту надання доступу, потрібно виділи стадії, які людина проходить перед початком служби в ЗСУ рисунок 3.1.

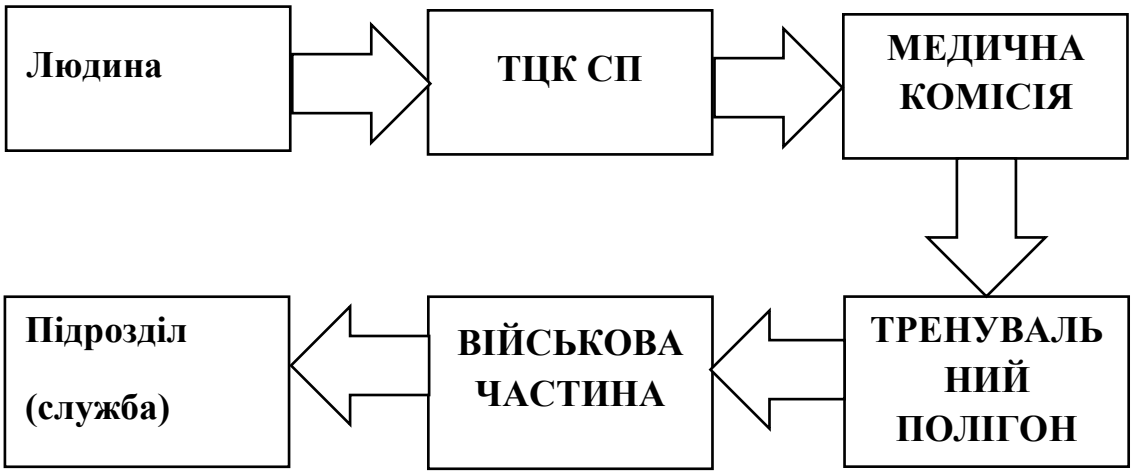

Рисунок 3.1 – Описує шлях початківця у військовій справі.

При вступі людини до Збройних Сил України вона проходить шість етапів(діаграма 1)початкова стадія, ТЦК (Територіальний центр комплектування), медична комісія, тренувальний полігон, військова частина та розподіл. Важливо визначити найкращий момент для надання токену для використання програми, враховуючи можливі ризики на кожному етапі.

Початкова стадія є першим кроком, де особа може ще не мати повної інформації про подальші процедури і вимоги. Надання токену на цьому етапі може бути передчасним, оскільки особа ще не пройшла необхідні перевірки і відбори. На етапі ТЦК також можуть виникнути певні ризики, оскільки тут відбувається первинна реєстрація та збір документів, і можливий доступ до токенів може створити додаткові ризики витоку інформації.

Медична комісія є важливим етапом, де визначається фізичний і психічний стан особи. Надання токену на цьому етапі може бути недоцільним, оскільки особа ще не отримала остаточного допуску до служби. Тренувальний полігон також не є оптимальним місцем для надання токену, оскільки особа проходить інтенсивну підготовку і може не мати постійного доступу до засобів зв'язку та пристроїв.

Військова частина є більш підходящим етапом, оскільки тут особа вже інтегрована у військову структуру і має необхідні засоби для використання програми. Однак, остаточний розподіл є найкращим моментом для надання токену. На цьому етапі особа вже повністю інтегрована в систему, призначена до конкретного підрозділу і має доступ до всіх необхідних ресурсів для ефективного використання програми. Це мінімізує ризики неправильного використання або втрати токену, оскільки військовослужбовець вже знаходиться під контролем і наглядом відповідних військових структур.

Збереження токену, закріпленим за певним солдатом рисунок 3.2.

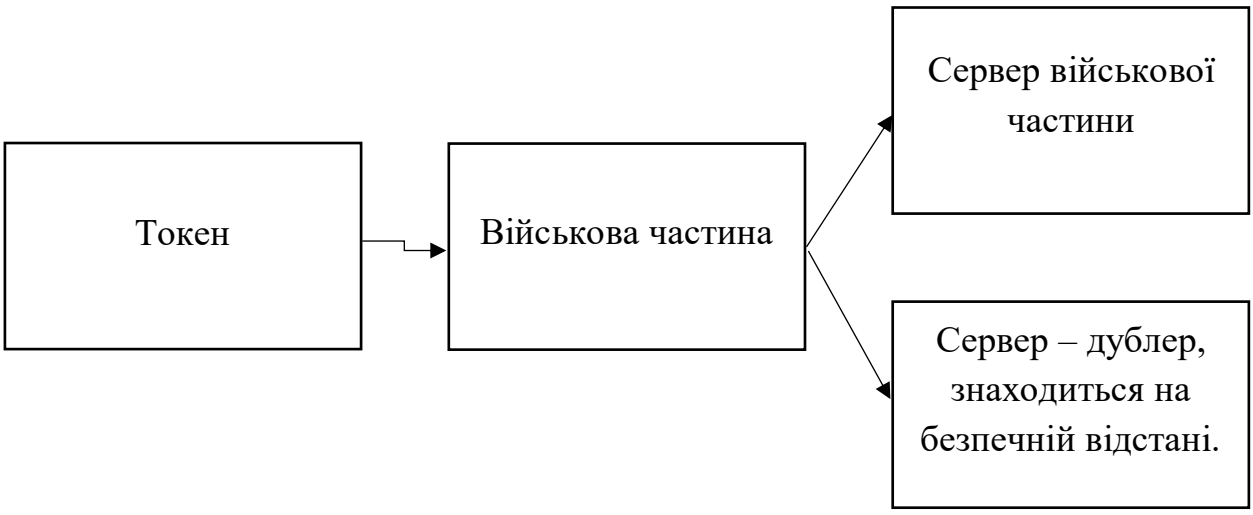

Рисунок 3.2 - Збереження токену.

Щоб виключати випадки коли сервер на якому зберігається токен та який може контролювати його, буде знищено, введемо сервери дублери, бажана кількість від 1 до 3 одиниць. Це виключить ймовірність втрати контролю через фізичні пошкодження та надасть можливість постійного контакту з сервером навіть у моменти загострення бойових дій.

3.2 Автентифікація

Автентифікація - це процес перевірки особи користувача перед наданням йому доступу до конфіденційної інформації, систем та програм. Існують різні методи здійснення автентифікації (Рисунок 3.3), основними з яких є наступні:

- **Ім'я користувача та пароль:** цей метод вважається непридатним через ризик компрометації пароля.
- **Двофакторна автентифікація:** Цей підхід також вважається неприйнятним, оскільки він стає неефективним, якщо пов'язаний з ним телефон втрачено.
- **Біометрична автентифікація:** Цей метод не рекомендується, оскільки він може бути використаний у разі захоплення солдата в полон.
- **Доступ на основі токенів:** Цей варіант є дуже зручним, оскільки токенами можна ефективно керувати, що дозволяє негайно відкликати і повторно авторизувати доступ.

Ці міркування підкреслюють важливість вибору надійного механізму автентифікації, який відповідає специфічним вимогам безпеки військових операцій.

Рисунок 3.3 – Приклади ідентифікації.

Після тривалих досліджень та аналізу було зроблено висновок, що генерація токенів є оптимальним рішенням з наступних причин:

- **Непередбачуваність:** Процес генерації за своєю суттю є випадковим і часто включає в себе криптографічно стійкі алгоритми. Це гарантує, що передбачити наступні токени неможливо.
- **Довжина токена:** Використання довжини 128 біт й більше більше створює величезний простір можливих значень, що робить вибір грубої сили практично неможливим.
- **Універсальність:** Токени можуть бути як одноразовими, так і багаторазовими, що забезпечує гнучкість у розповсюдженні та використанні програми.

- **Безпека:** Навіть якщо токен перехоплено, додаткові методи автентифікації запобігають несанкціонованому використанню, забезпечуючи надійну безпеку.

Програмна реалізація генерації токену.

```python
import random
import string


def generate_token(length=10):
    # Включення в генерацію токену спеціальних символів
    characters = (
        string.ascii_letters +
        string.digits +
        "!@#$%^&*()_+=-[]{}|:;<>?,./"
    )
    # Генерація з визначеного діапазону
    token = ''.join(random.choice(characters) for _ in range(length))
    return token


# Приклад використання:
generated_token = generate_token()
print("Generated Token:", generated_token)
```

Рисунок 3.4- Реалізація генерації токена мовою python.

Наведений код на Python визначає функцію generate_token, яка генерує випадковий токен заданої довжини, за замовчуванням 10 символів. Ця функція використовує модулі random та string для створення токену. Усередині функції ініціалізується змінна з іменем characters, яка містить повний набір символів, включаючи великі і малі літери, цифри і різні спеціальні символи. Цей різноманітний набір символів підвищує складність і непередбачуваність згенерованого токена(рисунок 3.4).

В основі процесу генерації токенів лежить перебір списку, який повторюється в діапазоні, визначеному заданою довжиною токена. На кожній ітерації використовується функція random.choice для вибору випадкового

символу з пулу символів. Потім ці вибрані символи об'єднуються в один рядок за допомогою методу join. Цей рядок представляє згенерований токен.

Функція завершується поверненням згенерованого токену. Код також містить приклад використання, де функція generate_token викликається без аргументів, таким чином генеруючи маркер стандартної довжини. Згенерований токен згодом виводиться на консоль.

Ця реалізація використовує стандартні бібліотеки Python для створення надійного та гнучкого механізму генерації токенів. Включення широкого спектру символів забезпечує високий рівень ентропії, що робить токени складно передбачуваними або піддаються грубому підбору. Крім того, параметр функції за замовчуванням дозволяє легко налаштовувати довжину токенів, забезпечуючи універсальність для різних вимог додатків. В цілому, код демонструє простий, але ефективний підхід до генерації безпечних випадкових токенів.

Реалізація додаткової автентифікації.

Для посилення захисту та розширення функціоналу також було вирішено додавати пошту та пароль, після того як було використано токен(рисунок 3.5).

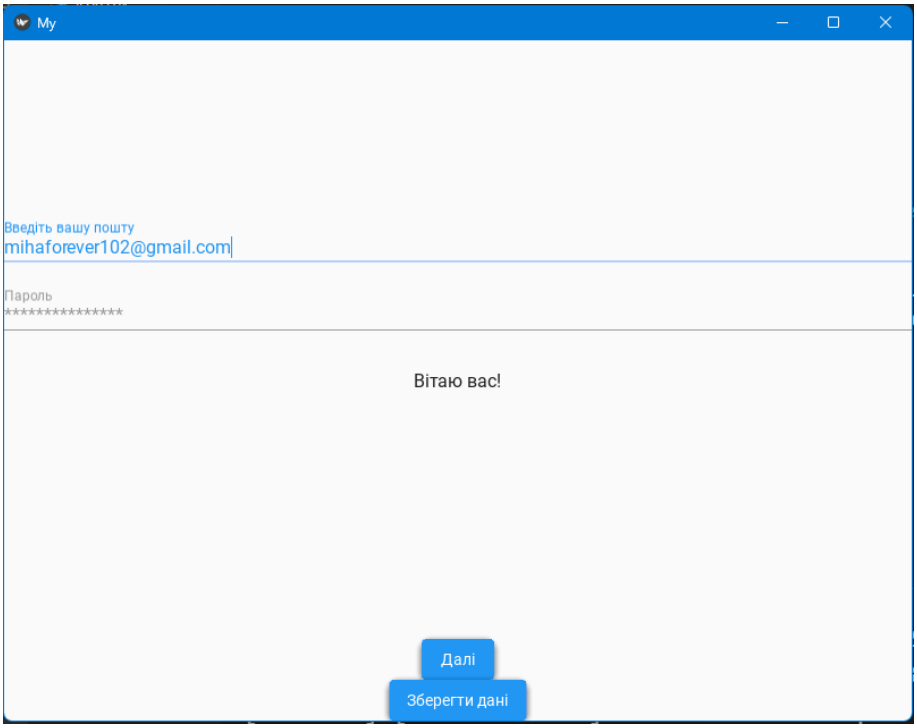

Рисунок 3.5 – Сторінка додаткової авторизації.

Максимальна простота надасть можливість швидко адаптуватися нашим воїнам до застосунку, також на самій сторінці присутні такі кнопки, як:

- Далі – переключення на наступну сторінку.
- Зберегти дані – для подальшого використання на самому пристрої під час нових сесій відкриття програми, без введення, дані зберігаються в окремий файл(рисунок 3.6).

```
    self.button_save_credentials = MDRaisedButton(text="Зберегти дані",
        on_release=self.save_credentials,
        pos_hint={'center_x': 0.5, 'center_y': 0.2},
        )

# Load saved credentials
saved_credentials = load_credentials()
if saved_credentials:
    self.email_input.text = saved_credentials.get('email', '')
    self.password_input.text = saved_credentials.get('password', '')

def save_credentials(self, instance):
    email = self.email_input.text
    password = self.password_input.text

    save_credentials(email, password)
    print("Credentials saved!")
```

Рисунок 3.6 – Реалізація кнопки збереження даних.

Методом self – передаються дані, які записані в 2 полях, далі вони конвертуються й зберігаються в форматі .json, це зроблено для зручності, адже сама мова python включає в себе роботу з цим форматом. Методом get – ми отримуємо записані дані з фалу credentials.json(рисунок 3.7).

```
{} credentials.json > ...
1  {"email": "mihaforever102@gmail.com", "password": "213123123123213"}
```

Рисунок 3.7 – Приклад запису в файлі credentials.json.

Висновок

У результаті виконаної роботи досягнуто мети – розробка та створення ефективної та надійної телемедичної консультативної системи для ЗСУ. Базовою мовою програмування став Python, він став вибором через кроссплатформеність, зручність у використанні, багатофункціональність, та мав значні переваги над конкурентами.

Важливою перевагою запропонованої системи є її здатність об'єднувати різні аспекти медичної допомоги в одному інтерфейсі, що сприяє оперативному та зручному доступу до необхідної інформації та консультацій.

Розроблений програмний засіб передбачає розширення функціональності

Показано, що телемедицина має хороші перспективи для збільшення ефективності в умовах ведення воєнних дій.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Telemedicine and Telehealth: Principles, Policies, Performances and Pitfalls Автор: Adam William Darkins, Margaret Ann Cary https://books.google.com.ua/books?id=0GY5lZBY0J8C&pg=PA1&hl=uk&source=gbs_toc_r&cad=2#v=onepage&q&f=false сторінки 5-20 (дата звернення: 12.11.2023)
2. Telemedicine A.B. Владзимірський https://books.google.com.ua/books?id=7Mhz9bkk2gkC&printsec=frontcover&hl=ru#v=onepage&q&f=false сторінки 160-170 (дата звернення: 1.05.2024).
3. Фізична підготовка і спорт. https://dshv.mil.gov.ua/fizichna-pidgotovka-i-sport/ (дата звернення: 12.11.2023)
4. ОЦІНЮВАННЯ РІВНЯ ФІЗИЧНОЇ ПІДГОТОВЛЕНОСТІ. https://nuou.org.ua/assets/documents/kfv-otsiniuvannia-fiz-pid-2022.pdf (дата звернення: 21.11.2023)
5. Калорійність харчування військових, які виконують бойові завдання. https://zakon.rada.gov.ua/laws/show/426-2002-%D0%BF#Text (дата звернення: 21.11.2023)
6. Калорійність харчування військових, які виконують бойові завдання. https://interfax.com.ua/news/general/342012.html
7. (дата звернення: 21.11.2023)
8. СИЛОВІ ТА ГІМНАСТИЧНІ ВПРАВИ З ВАГОЮ ВЛАСНОГО ТІЛА. https://sprotyvg7.com.ua/lesson/silovi-ta-gimnastichni-vpravi-z-vagoyu-vlasnogo-tila. (дата звернення: 24.11.2023)
9. РЕАБІЛІТАЦІЙНІ ВПРАВИ ДЛЯ ВІДНОВЛЕННЯ ОРГАНІЗМУ В ПОЛЬОВИХ УМОВАХ. https://sprotyvg7.com.ua/lesson/fizichna-pidgotovka (дата звернення: 1.05.2024)
10. Вітаміни для ЗСУ: Biotus відправляє на фронт вітаміни. https://blog.biotus.lv/lv/blog/vitaminy-dlya-zsu-biotus-vidpravlyaye-na-front-vitaminy.html (дата звернення: 1.05.2024)
11. Kivy. The Open Source Python App Development Framework. https://kivy.org/ (дата звернення: 1.05.2024)

12. KivyMD. https://kivymd.readthedocs.io/en/latest/ (дата звернення: 1.05.2024)

ДОДАТОК А

```
from kivy.lang import Builder
from kivy.uix.boxlayout import BoxLayout
from kivymd.app import MDApp
from kivymd.uix.button import MDRaisedButton
from kivymd.uix.label import MDLabel
import webbrowser
import sqlite3

KV = '''
MDScreen:

    MDBottomNavigation:

        MDBottomNavigationItem:
            name: 'screen 1'
            text: 'Невідкладна допомога'
            icon: 'account-tie-voice'
            badge_icon: "numeric-10"

            MDRaisedButton:
                text: 'Підключитися до конференції'
                halign: 'center'
                size_hint_x: 0.5  # Встановлюємо ширину кнопки в 80% від ширини вкладки
                size_hint_y: 0.5
                font_size: '24sp'  # Змінюємо розмір шрифту на 24sp
                md_bg_color: (1, 0, 0, 1)  # Червоний колір фону (обведення) кнопки
                line_color: (0, 0, 0, 1) # Чорний колір обведення шрифту
                pos_hint: {'center_x': 0.5, 'center_y': 0.5}
                on_release: app.join_google_meet()

        MDBottomNavigationItem:
            name: 'screen 2'
            text: 'Запис до лікаря'
            icon: 'timetable'

            MDFloatLayout:
                orientation: 'horizontal'
                padding: dp(30)  # Додати відступи на кожному боці
```

```
import sqlite3

# Підключення до бази даних SQLite
conn = sqlite3.connect('doctors.db')
conn = sqlite3.connect('schedules.db')
c = conn.cursor()

# Створення таблиці лікарів
c.execute('''CREATE TABLE doctors
             (id INTEGER PRIMARY KEY, name TEXT, specialization TEXT)''')

# Створення таблиці розкладів лікарів
c.execute('''CREATE TABLE schedules
             (doctor_id INTEGER, day TEXT, time TEXT)''')

# Додавання даних про лікарів
c.execute("INSERT INTO doctors (name, specialization) VALUES ('SMIT', 'Терапевт')")
c.execute("INSERT INTO doctors (name, specialization) VALUES ('DJONTH', 'Хірург')")
c.execute("INSERT INTO doctors (name, specialization) VALUES ('Vika', 'Окуліст')")

# Додавання розкладу лікарів
c.execute("INSERT INTO schedules (doctor_id, day, time) VALUES (1, 'Понеділок', '10:00')")
c.execute("INSERT INTO schedules (doctor_id, day, time) VALUES (1, 'Вівторок', '12:00')")
c.execute("INSERT INTO schedules (doctor_id, day, time) VALUES (2, 'Середа', '14:00')")
c.execute("INSERT INTO schedules (doctor_id, day, time) VALUES (2, 'Середа', '14:00')")
c.execute("INSERT INTO schedules (doctor_id, day, time) VALUES (2, 'Середа', '16:00')")
c.execute("INSERT INTO schedules (doctor_id, day, time) VALUES (2, 'Середа', '20:00')")

```

```
class TestScreen(Screen):
    def __init__(self, **kwargs):
        super(TestScreen, self).__init__(**kwargs)
        self.orientation = 'vertical'

        # Adding labels and text fields
        self.label = MDLabel(text="Вітаю вас!", halign='center')
        self.email_input = MDTextField(
            hint_text="Введіть вашу пошту",
            pos_hint={'center_x': 0.5, 'center_y': 0.7},)
        self.password_input = MDTextField(hint_text="Пароль",
            pos_hint={'center_x': 0.5, 'center_y': 0.6},
            password=True,)

        # Load saved credentials
        saved_credentials = load_credentials()
        if saved_credentials:
            self.email_input.text = saved_credentials.get('email', '')
            self.password_input.text = saved_credentials.get('password', '')

        # Adding a vertical BoxLayout for buttons
        buttons_layout = BoxLayout(orientation='vertical')

        # Adding buttons to the layout
        self.button_click_me = MDRaisedButton(text="Далі",
            on_release=self.go_to_next_window,
            pos_hint={'center_x': 0.5, 'center_y': 0.3},
            )
        self.button_save_credentials = MDRaisedButton(text="Зберегти дані",
            on_release=self.save_credentials,
            pos_hint={'center_x': 0.5, 'center_y': 0.2},
            )
```

```
class MyApp(MDApp):
    def build(self):
        screen_manager = ScreenManager()

        test_screen = TestScreen(name='test')
        photo_screen = PhotoScreen(name='foto')

        screen_manager.add_widget(test_screen)
        screen_manager.add_widget(photo_screen)

        return screen_manager
```

```
        self.height_input = MDTextField(
            hint_text="ПО-БАТЬКОВІ",
            helper_text="ПО-БАТЬКОВІ",
            helper_text_mode="on_focus",
            pos_hint={'center_x': 0.5, 'center_y': 0.8},
            size_hint=(0.8, None),
        )

        self.weight_input = MDTextField(
            hint_text="ЗРІСТ",
            helper_text="ЗРІСТ",
            helper_text_mode="on_focus",
            pos_hint={'center_x': 0.5, 'center_y': 0.7},
            size_hint=(0.8, None),
        )

        self.age_input = MDTextField(
            hint_text="ВАГА",
            helper_text="ВАГА",
            helper_text_mode="on_focus",
            pos_hint={'center_x': 0.5, 'center_y': 0.6},
            size_hint=(0.8, None),
        )

        self.militarystatus_input = MDTextField(
            hint_text="ВІК",
            helper_text="ВІК",
            helper_text_mode="on_focus",
            pos_hint={'center_x': 0.5, 'center_y': 0.5},
            size_hint=(0.8, None),
        )


        if not credentials:
            Snackbar(text="Failed to authenticate. Please check your credentials.").open()
            return

        subject = "Photo from KivyMD App"
        body = "Attached is the photo submitted from the KivyMD App."

        try:
            msg = MIMEMultipart()
            msg.attach(MIMEText(body, 'plain'))

            # Attach the photo
            with open(self.image_path, 'rb') as file:
                img = MIMEImage(file.read(), name='photo.jpg')
            msg.attach(img)

            # Use Gmail's SMTP server
            server = smtplib.SMTP('smtp.gmail.com', 587)
            server.starttls()

            # Send email using the authenticated user's credentials
            server.login(credentials.email, credentials.token)
            server.sendmail(credentials.email, email, msg.as_string())
            server.quit()

            Snackbar(text="Photo submitted successfully").open()
        except Exception as e:
            print(f"Error: {e}")
            Snackbar(text="Failed to submit photo. Check your email and credentials.").open()
    else:
        Snackbar(text="Please select a photo first").open()
```