ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра Інженерія програмного забезпечення

Пояснювальна записка

до бакалаврської роботи

на ступінь вищої освіти бакалавр

на тему: «**РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ПІДТРИМКИ ПРОЦЕСУ НАДАННЯ ПЕРШОЇ ДОПОМОГИ МОВОЮ PYTHON**»

Виконав: студент 4 курсу, групи ПД-41 спеціальності 121 Інженерії програмного забезпечення

(шифр і назва спеціальності)

Спіцин А.Я.

(прізвище та ініціали)

Керівник: Трінтіна Н.А.

(прізвище та ініціали)

Рецензент:

(прізвище та ініціали)

Київ – 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти – «Бакалавр»

Спеціальність - 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного
забезпечення
Негоденко О.В.
«____»_____2023 року

ЗАВДАННЯ

НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Спіцин Андрій Ярославович

(прізвище, ім’я, по батькові)

1. Тема роботи: «Розробка програмного забезпечення для підтримки процесу надання першої допомоги мовою Python»

Керівник роботи к.т.н., доц. Трінтіна Наталія Альбертівна

(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від «24» лютого 2023 року №26

2. Строк подання студентом роботи «1» червня 2023 року

3. Вихідні дані до роботи:

Науково-технічна література;

PyCharm;

Figma;

draw.io;

Офіційна документація мови програмування Python.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):

4.1. Визначення особливостей процесу надання першої допомоги

4.2. Огляд та аналіз існуючих додатків для підтримки процесу надання першої допомоги

4.3. Проектування додатку

4.4. Розробка додатку
4.5. Тестування додатку
4.6. Висновки
5. Перелік графічного матеріалу
5.1. Мета, об’єкт та предмет дослідження
5.2. Задачі дипломної роботи
5.2. Аналіз аналогів
5.3. Вимоги до програмного забезпечення
5.4. Програмні засоби реалізації
5.5. Діаграма варіантів використання
5.6. Діаграма класів
5.7. Схема роботи програми
5.8. Екранні форми
5.9. Апробація результатів дослідження
5.10. Висновки
6. Дата видачі завдання «25» лютого 2023 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Отримання завдання	25.02.2023	Виконано
2	Аналіз наукових джерел	28.02.2023	Виконано
3	Формування вимог	02.03.2023	Виконано
4	Проектування програмного забезпечення	13.03.2023	Виконано
5	Розробка програмного забезпечення	03.04.2023	Виконано
6	Тестування програмного забезпечення	10.04.2023	Виконано
7	Оформлення пояснювальної записки	17.05.2023	Виконано
8	Перевірка від керівника	19.05.2023	Виконано
9	Передзахист дипломної роботи	23.05.2023	Виконано
10	Перевірка на нормоконтроль та плагіат	25.05.2023	Виконано
11	Здача роботи	1.06.2023	

Студент _____ Спіцин А.Я.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Трінтіна Н.А.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Текстова частина бакалаврської роботи 67 с., 45 рис., 14 джерел.

Об'єкт дослідження – процес надання першої допомоги.

Предмет дослідження – програмне забезпечення для підтримки процесу надання першої допомоги

Мета дослідження – зменшити час надання першої допомоги за рахунок застосування мобільного додатку, створеного мовою Python.

У дипломній роботі:

1. Проведено аналіз процесу надання першої допомоги;
2. Сформовані вимоги до програмного забезпечення для підтримки процесу надання першої допомоги;
3. Реалізовано єдиний алгоритм першої допомоги;
4. Спроектовано програмне забезпечення;
5. Розроблено програмне забезпечення;
6. Протестовано програмне забезпечення згідно сформованих вимог.

Наукова новизна – повністю озвучений єдиний алгоритм першої допомоги.

Галузь використання – медицина.

ЗМІСТ

ВСТУП

У сучасному світі смартфони є одним з найпоширеніших засобів зв'язку та спілкування. Незалежно від того, чи використовуєте ви його для звичайних розмов або для ведення бізнесу, є безліч ситуацій, коли може знадобитися перша допомога. Статистика показує, що більшість людей не має достатніх знань про те, як надавати першу допомогу, що може призвести до непотрібної затримки в допомозі або навіть погіршення стану постраждалого.

Один зі способів розв'язання цієї проблеми полягає в розробці додатку, який надасть користувачам інструкції першої допомоги у режимі реального часу. Він може стати важливим допоміжним засобом в будь-якій ситуації, коли необхідна невідкладна медична допомога.

Щоб забезпечити максимальну ефективність, додаток буде озвучувати всі інструкції, щоб користувач міг сконцентруватися на наданні допомоги постраждалому. Проте, користувачеві все ще доведеться натискати на екран смартфону, щоб повідомити додаток про стан постраждалого для отримання наступних інструкцій.

На відміну від офіційного додатку Червоного Хреста для першої допомоги, розроблюваний додаток буде мати більш оперативний доступ до інструкцій. Це дасть можливість зекономити трохи часу та негайно перейти до допомоги потерпілому. Крім того представлення інформації в аудіо форматі звільнить користувача додатку необхідності постійно дивитися на екран, читаючи інструкції. Такі пункти вигідно відрізняють розроблюваний додаток.

Додаток стає особливо корисним у сучасних реаліях війни. У такий час значно зростає небезпека для громадян і крім того медична допомога потерпілому може бути недоступна, або надходити занадто довго. Але правильна та своєчасна перша допомога може виграти цей дорогоцінний час.

Загалом, розробка додатку, який би надавав інструкції першої допомоги у реальному часі, може стати важливим кроком у покращенні доступу до медичної допомоги в будь-яких умовах. Для цього необхідно внести значний вклад у

розробку та вдосконалення додатку, а також в його популяризацію серед широкої громадськості, щоб кожен зміг навчитися надавати першу допомогу та зберегти життя в критичний момент.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд предметної області

Перша допомога - це першочергова медична допомога, яка надається людям у випадку нагальних медичних ситуацій. Це може бути необхідно в разі травм, обморожень, опіків, укусів, отруєнь, серцевих нападів та інших надзвичайних ситуацій, коли кожна хвилина рахується.

Основною метою першої допомоги є збереження життя та запобігання подальшому ушкодженню доки не приїдуть медичні працівники. При цьому необхідно враховувати особливості конкретної ситуації та дотримуватися визначених процедур. Однак, незважаючи на це, важливо пам'ятати, що будь-яка допомога краще, ніж відсутність допомоги.

Основні принципи першої допомоги - це швидкість реакції, точність дій та максимальна безпека як для постраждалого, так і для допомагаючого. Наприклад, в разі порізу важливо зупинити кровотечу, накласти асептичний бинт та забезпечити стерильність рани. При серцевому нападі важливо швидко надати доступні ліки та зателефонувати на "швидку допомогу".

Важливо також пам'ятати, що перша допомога - це тимчасова допомога, яка повинна бути доповнена повноцінною медичною допомогою, наданою професійними медичними працівниками.

Перша допомога – це набір заходів, направлених на покращення стану та врятування життя постраждалої людини. Згідно статистики Британського Червоного Хреста 59% смертей, спричинених травмами можна було б уникнути, якби була вчасно та коректно надана перша допомога. Ця ж статистика показує, що більшість людей не мають достатнього рівня знань першої допомоги для її виконання. Це доволі великі показники, що вказує на достатню актуальність проблеми.

Через це незнаюча людина при першій допомозі витрачає багато часу на те, щоб знайти, прочитати та засвоїти інформацію, необхідну для надання допомоги.

Ця затримка може призвести як до погіршення стану потерпілого, так і до його смерті.

Отже, знання першої допомоги - це важливе уміння, яке може допомогти врятувати життя у критичний момент. Навчитися цьому умінню можна у спеціалізованих школах, медичних університетах, а також самостійно, з використанням літератури та Інтернет-ресурсів.

Оскільки у цього феномену дуже давня історія, існує безліч літератури, пов’язаної з наданням першої допомоги. Такої як наприклад «The Complete First Aid Pocket Guide», написаний кваліфікованим інструктором з надання першої допомоги та реанімації Джоном Фурстом. Або «Comprehensive Guide for First Aid & CPR», виданий Канадським Червоним Хрестом. Саме цей посібник став головним джерелом для написання більшості інструкцій у розробленому додатку.

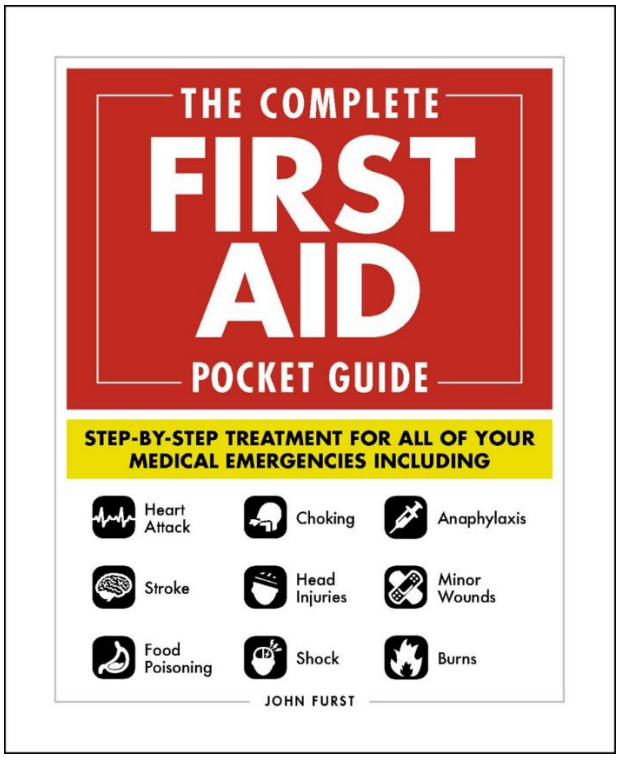


Рисунок 1.1 – The Complete First Aid Pocket Guide

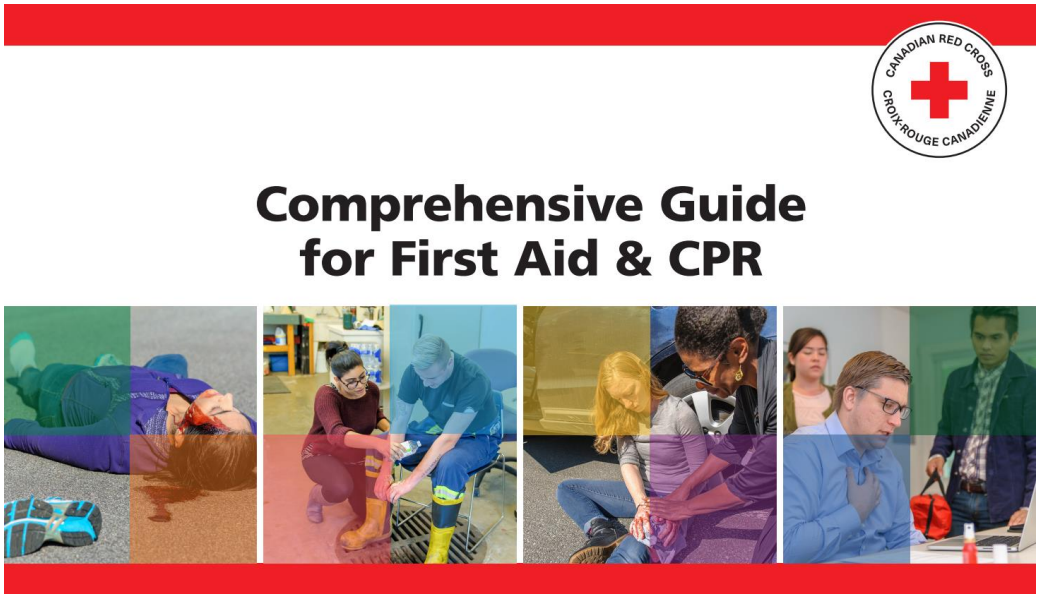


Рисунок 1.2 – Comprehensive Guide for First Aid & CPR

1.2 Вивчення сучасних підходів до надання першої допомоги

Надання першої допомоги – це комплексна задача з багатьма нюансами та деталями. Тому написання абсолютного, єдиного алгоритму на всі випадки, що можуть трапитися з людиною, є важким завданням, що потребує достатньої кваліфікації у медичній сфері. Крім того такий алгоритм зайняв би приблизно 200 – 250 сторінок тексту, тобто десь 7,86 – 9,8 годин аудіо інструкцій.

Через це було прийнято рішення обмежити інструкції для створення прототипу програмного продукту. Так у першу версію програми потрапляють інструкції тільки на ті випадки, які безпосередньо можуть містити загрозу життю потерпілого. Отже список інструкцій був сформований наступним чином:

1. Серцево-легенева реанімація
2. Кровотеча
3. Опік
4. Травма
5. Серцевий напад
6. Удушення
7. Анафілаксія

А також проміжні інструкції щодо огляду потерпілого та виходу вже на конкретну інструкцію допомоги.

Наступні інструкції були написані спираючись на посібник «Comprehensive Guide for First Aid & CPR», виданий товариством Червоного Хреста.

Перевірка на притомність:

Перед наближенням до потерпілого, перевірте навколишнє середовище на потенційні загрози. Лише потім підходьте та перевірте чи притомна людина.

Перевірка дихання:

Перевірте дихання потерпілого. Для цього нахиліть його голову назад та наблизьте свою щоку до обличчя потерпілого. Ви маєте відчути подих.

Перевірка серецебиття:

Перевірте наявність пульсу потерпілого на сонній артерії.

Огляд потерпілого:

Огляньте потерпілого з голови до ніг на наявність кровотеч: зовнішніх або внутрішніх, опіків або травм. Внутрішню кровотечу можна визначити за синцями у постраждалій області або затвердінням м’яких тканин, таких як живота.

Огляд притомного потерпілого:

Оцініть стан постраждалої людини: зверніть увагу на будь-які скарги, які може мати постраждалий. Якщо ви думаєте, що у постраждалої людини є ознаки серцевого нападу, такі як біль або стиснення у грудях, слабкість, нудота, перейдіть до кроку «серцевий напад». Якщо ви бачите, що людина давиться, не може дихати та хапається за горло, перейдіть до кроку «удушення». Якщо ви думаєте, що у постраждалої людини є ознаки анафілаксії, такі як висип, набряк обличчя, проблеми з диханням, перейдіть до кроку «анафілаксія». В інших варіантах, перейдіть до кроку «інше».

Серцево-легенева реанімація:

Попросіть когось викликати швидку. Сядьте з лівого боку потерпілого. Покладіть обидві руки на центр його грудей та зробіть 30 натисків хоча б на 5 см глибини, згідно ритму. Для дітей молодше одного року натискайте двома пальцями на третину товщини тіла. Після 30 натисків нахиліть голову потерпілого назад,

затисніть ніс потерпілого та зробіть 2 односекундні видихи у рот потерпілого, побачивши як підіймаються груди. Повторіть цикл. 30 натисків хоча б на 5 см глибини, згідно ритму та 2 односекундні видихи.

Допомога при кровотечі:

Якщо кровотеча серйозна, викличте швидку. Для зовнішньої кровотечі надайте тиск на рану, доки кровотеча не зупиниться. Якщо вона не зупиняється, за можливості використайте турнікет. Для внутрішньої просто викличте швидку, ви не зможете сильно допомогти.

Допомога при опіках:

Якщо опік серйозний, викличте швидку. Охолоджуйте опік чистою водою або компресом, але не дуже холодним, протягом 10 хвилин. Зніміть одяг та прикраси з місця опіку, але не намагайтесь відривати те, що прилипло до шкіри. Коли опік охолоне, нещільно накрийте його сухою стерильною пов’язкою.

Допомога при травмах:

Якщо травма серйозна, викличте швидку. При травмі голови, шиї, або хребта, зафіксуйте відповідні частини тіла потерпілого, як вони були, та чекайте на швидку допомогу. При звичайних переломах також зафіксуйте уражену частину тіла у тій позиції, у якій вона була. Охолоджуйте уражену ділянку протягом 20 хвилин кожної години, якщо це не спричиняє болю. При можливості, тримайте уражену ділянку вище рівня серця.

Допомога при серцевому нападі:

Викличте швидку допомогу. Дайте людині відпочити та заспокойте її. При наявності, одноразово дайте людині аспірин, якщо в неї немає не нього алергії. Також можна прийняти нітрогліцерин, для зняття болю у грудях.

Допомога при удушенні:

Нахиліть стоячу людину, доки корпус не буде хоча б паралельно землі. Зробіть 5 жорстких ударів по спині між лопаток. Якщо не допомагає, станьте позаду людини та охопіть людину руками, щоб одна рука була в кулаку, розташована одразу під ребрами, а інша її накриває. Зробіть 5 поштовхів всередину та наверх. Повторюйте цикл з цих двох технік, доки об’єкт не вийде з горла.

Допомога при анафілаксії:

Викличте швидку. Спитайте потерпілого про наявність епінефрину, якщо він мав приступи раніше. Якщо немає, просто очікуйте швидку. Допоможіть потерпілому застосувати епінефрин. Для цього потрібно зробити укол у стегно. Якщо симптоми не проходять та є друга доза, можна зробити укол у друге стегно. Використаний шприц епінефрину варто обов'язково відправити до лікарні разом з потерпілим. Якщо людина припинила дихати, перейдіть до відповідного розділу.

Загалом виходить 7 основних інструкцій на більшість інцидентів, які можуть мати загрозу життю потерпілому. А також 5 інструкцій з огляду потерпілого та визначення його стану.

1.3 Аналіз аналогів

1.3.1 Перша допомога — МФЧХ і ЧП

Офіційний мобільний додаток для першої допомоги, що був розроблений товариством Червоного Хреста. Містить прості покрокові інструкції та відео, з усією необхідною інформацією для надання першої допомоги в обраній ситуації. У додатку також можна проходити тести для перевірки своїх знань. Додаток безкоштовний та локалізований 40 мовами, що робить його доступним для будь-кого.

Добре підходить для вирішення поставленої задачі, адже надає усі інструкції у два кліки. Однак все ж таки має свої недоліки. Серед таких недоліків можна виділити недостатню чіткість деяких інструкцій. А також додаток все ще потребує час на ознайомлення з інструкцією, що може забрати якийсь час, що міг би бути використаний на допомогу потерпілому.

Екранні зображення:

Рисунок 1.3 – Приклади інструкцій

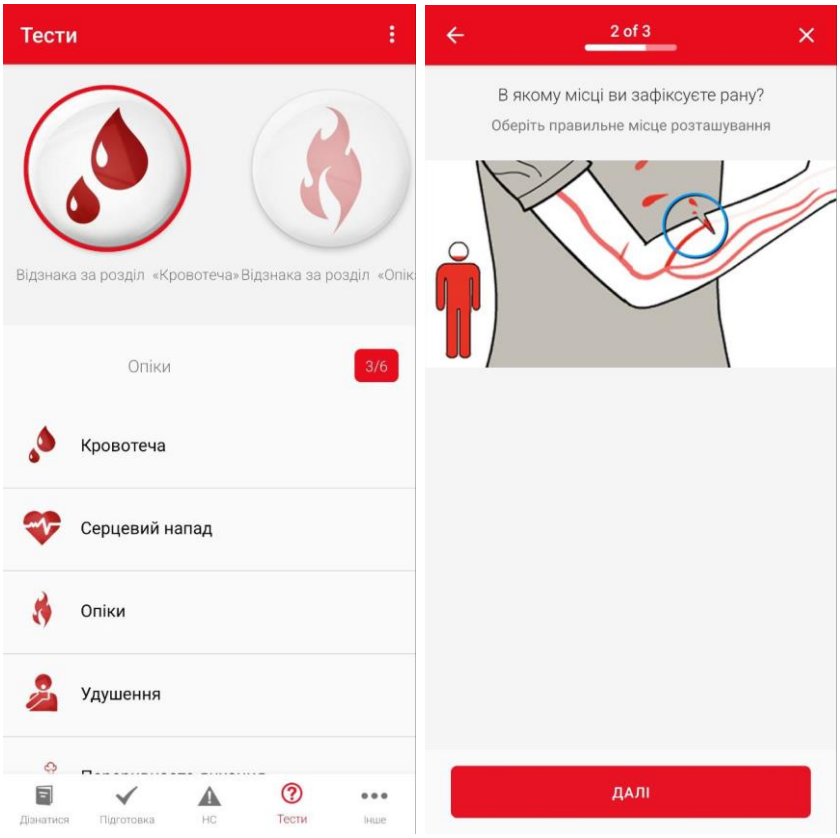


Рисунок 1.4 – Сторінка тестів та приклад питання

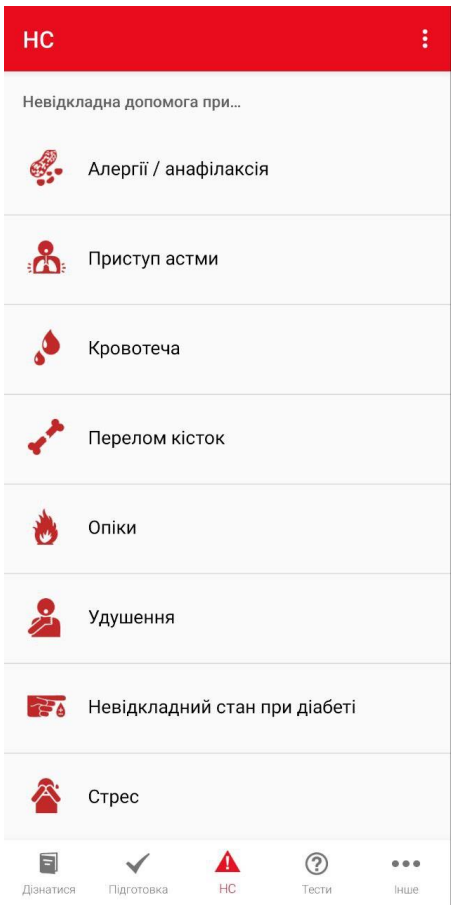


Рисунок 1.5 – Сторінка надзвичайних ситуацій

Переваги «Перша допомога — МФЧХ і ЧП»:

1. Простий та інтуїтивно зрозумілий інтерфейс
2. Корисні картинки

Недоліки «Перша допомога — МФЧХ і ЧП»:

1. Деякі інструкції написані недостатньо детально

1.3.2 First Aid Fast

First Aid Fast – це мобільний додаток, що надає швидкий доступ до усіх потрібних інструкцій першої допомоги. Користувач має можливість знайти інструкцію на будь-які випадки, що можуть трапитися у повсякденному житті, скориставшись списком тем на головному екрані. Інструкції містять демонстраційні відео, що частково допомагають зекономити час. Крім того є можливість відкрити список найближчих лікарень у окремій вкладці, якщо

потрібно. Однак для користування цим додатком потрібно підв’язати до нього електрону пошту.

Також є непоганим варіантом вирішення поставленої задачі. Однак для людини, якій терміново будуть потрібні інструкції першої допомоги, буде не дуже приємно застрягти на екрані реєстрації.

Екранні зображення:

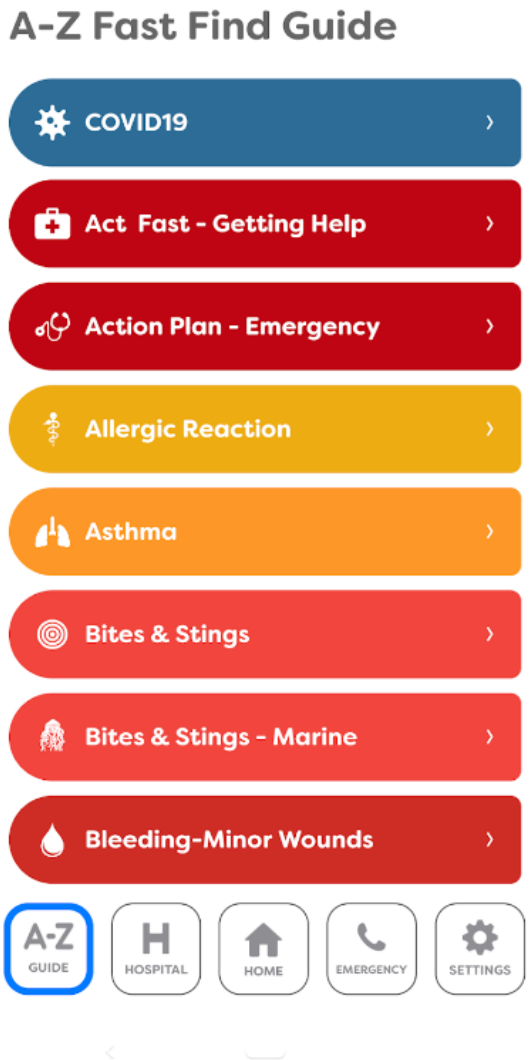


Рисунок 1.6 – Сторінка зі списком інструкцій

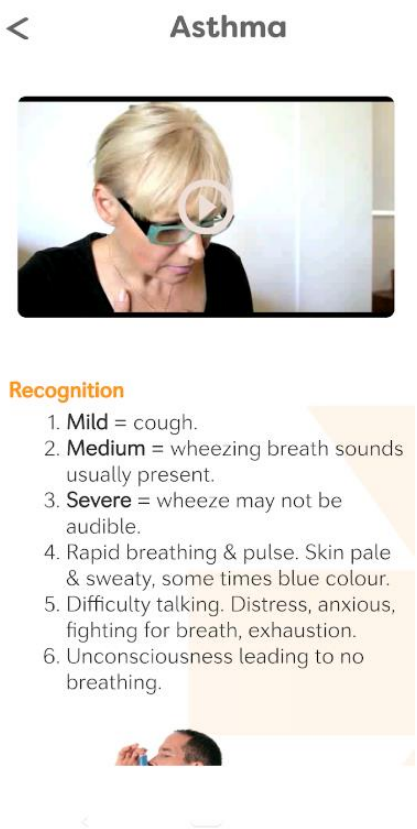


Рисунок 1.7 – Приклад інструкції

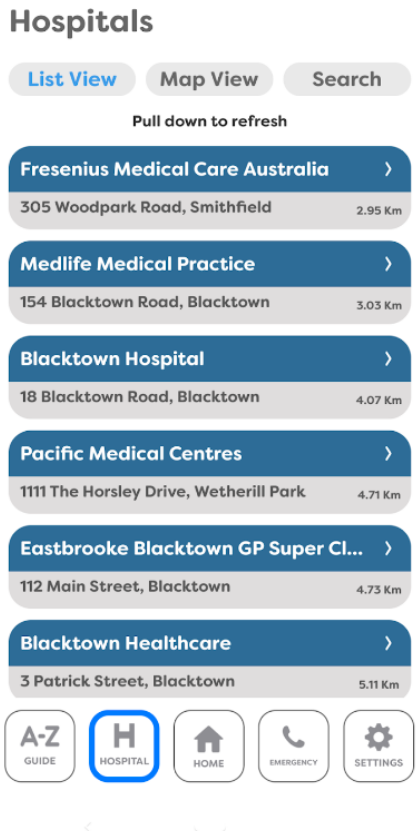


Рисунок 1.8 – Сторінка зі списком лікарень

Переваги «First Aid Fast»:

1. Гарні навчальні відео з кожної теми

2. Простий у використанні інтерфейс, який легко зрозуміти в надзвичайних ситуаціях

3. Можливість знайти лікарню в додатку

Недоліки «First Aid Fast»:

1. До нього має бути прикріплений обліковий запис електронної пошти.

1.3.3 First Aids and Emergency techniques

Мобільний додаток, що надає доступ до інструкцій першої допомоги. При відкритті додатку одразу відкривається список ситуацій з відповідними інструкціями, де користувач може обрати потрібну йому. Інструкції являють собою текст без зображень та відео.

Це програмне забезпечення не виконує поставленої задачі, адже великий суцільний текст, як єдине джерело інформації, забере багато часу на ознайомлення. Крім того пошук потрібної теми виконаний не найкращим чином. Усі теми розбиті на категорії «На дорозі», «На роботі», що не є інтуїтивно зрозумілим варіантом. Таким чином програма з великою кількістю інформації, але проблемами з її пошуком може краще використовуватись для нетермінових ситуацій, або для ознайомлення з інструкціями на випадок майбутньої потреби.

Екранні зображення:

Рисунок 1.9 – Головна сторінка

Рисунок 1.10 – Сторінка міфів про першу допомогу

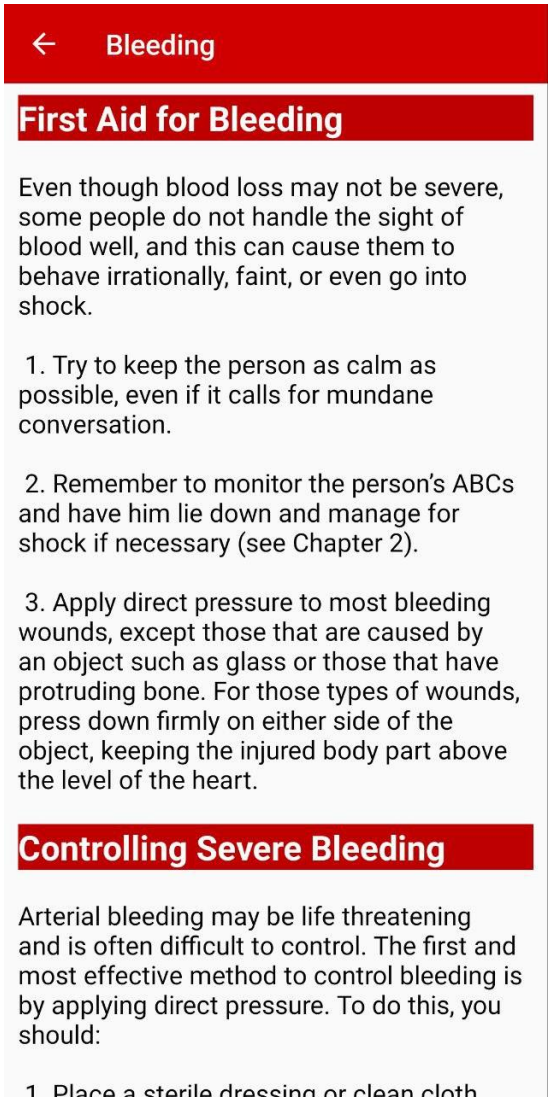


Рисунок 1.11 – Приклад інструкції

Переваги «First Aids and Emergency techniques»:

1. Підійде для ознайомлення та підготовки до екстреної ситуації

Недоліки «First Aids and Emergency techniques»:

1. Немає картинок, а лише суцільний текст
2. Важко знайти потрібну тему

1.3.4 Висновки по проведеному аналізу аналогів

Таблиця 1.1 – порівняння програм аналогів

	Перша допомога — МФЧХ і ЧП	First Aid Fast	First Aids and Emergency techniques	Розроблений додаток
Інструкції першої допомоги	+	+	+	+
Наявність картинок	+	+	–	+
Тести	+	–	–	–
Виклик швидкої	+	–	–	–
Доступність українською	+	–	–	+
Знайти лікарню	–	+	–	–
Повне озвучення	+/–	+/–	–	+

Після аналізу було виявлено, що більшість аналогічних програм має майже однаковий функціонал. А якість додатку загалом визначається зручністю його навігації та поданням інформації.

Аналогічні додатки можуть містити свої унікальні переваги, такі як наявність тестів для перевірки своїх знань, або можливість подивитися найближчі лікарні, але вони не мають такого прямого відношення до поставленої задачі, як повне озвучення інструкцій.

1.4 Постановка задачі

Беручи до уваги усе зазначене вище, можна сформувати таку задачу: зменшити витрату часу на пошук інформації та її читання при виконанні першої допомоги постраждалому.

Одним з варіантів вирішення цієї задачі є розробка програмного забезпечення для підтримки процесу надання першої допомоги. Ця програма буде надавати інструкції у реальному часі, що будуть цілком озвучені. Це зменшить потребу користувача дивитися на екран, та дасть можливість сконцентруватися на потерпілому, замість шматка тексту. Крім того, ці інструкції будуть відтворюватись одразу, як тільки додаток буде відкритий, що також зменшить витрату часу.

Відкривши додаток, користувач потрапляє на стартовий екран, де починає читатися інструкція по загальному алгоритму першої допомоги. Цей алгоритм дасть можливість користувачу оглянути потерпілого та визначити що з ним не так, після чого виведе на відповідну інструкцію допомоги. Однак у випадку, коли користувач одразу чітко розуміє ситуацію постраждалого, у додатку завжди буде доступна кнопка для відкриття усього списку інструкцій. Тут можна буде обрати потрібний випадок та, пропустивши усі огляди потерпілого, одразу отримати інструкції для самої допомоги.

А у випадку, коли людина пропустила аудіо інструкцію та простих зображень із текстом недостатньо, на кожному етапі буде доступна кнопка для повтору поточної інструкції.

Оскільки перша допомога може бути потрібна будь-кому, цей додаток буде підходити як і для звичайних людей, не знайомих з першою допомогою, так і для знаючих. Завдяки простому інтерфейсу, у стресовій ситуації, будь-хто зможе легко знайти потрібний йому розділ, та отримати термінові інструкції. У той час як люди знайомі з першою допомогою, можуть використати алгоритм в програмі як схематичне нагадування, що і як робити, для підстраховки.

2. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Вибір технологій та платформи розробки для програмного забезпечення

Програмне забезпечення для підтримки процесу надання першої допомоги буде розроблено для мобільних платформ, а саме пристроїв із системою Android. Таке рішення випливає з технічних складностей у реалізації додатків з іншими системами та значною статистичною перевагою у популярності цієї системи в Україні. І хоча останнім часом за цією статистикою Android девайси починають поступово втрачати популярність, кількість їх користувачів все ще перевищує усіх конкурентів.

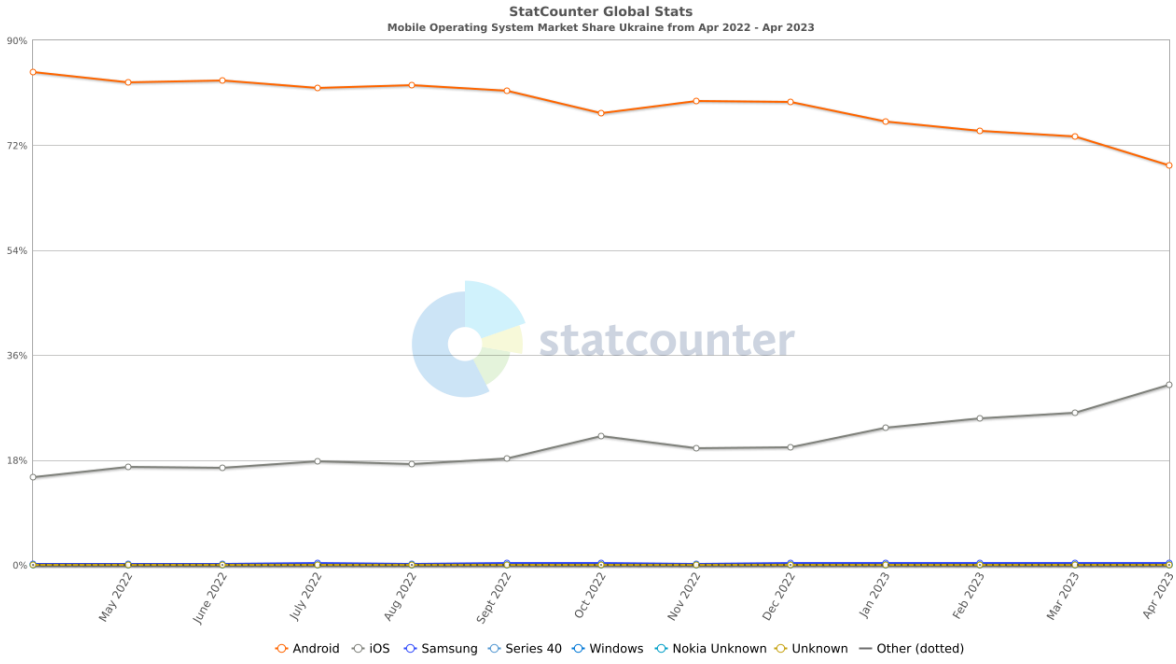


Рисунок 2.1 – Статистика використання різних мобільних систем в Україні

Для створення описаного програмного забезпечення було обрано мову програмування Python. Це потужна мова програмування з багатим набором бібліотек і фреймворків для розробки мобільних додатків, що спрощує процес розробки і гарантує швидку та ефективну роботу. Крім того Python також відомий

своєю простотою та зрозумілістю коду, що сприяє полегшенню розробки та підтримки програмного забезпечення.

Рисунок 2.2 – Логотип Python

В якості середовища розробки для Python був обраний PyCharm, розроблений компанією JetBrains. PyCharm є безкоштовним середовищем розробки, що спеціалізується саме на мові програмування Python. Воно містить усі потрібні інструменти та плагіни, а також це середовище є одним з кращих у питанні аналізу коду, що робить розробку значно простіше.

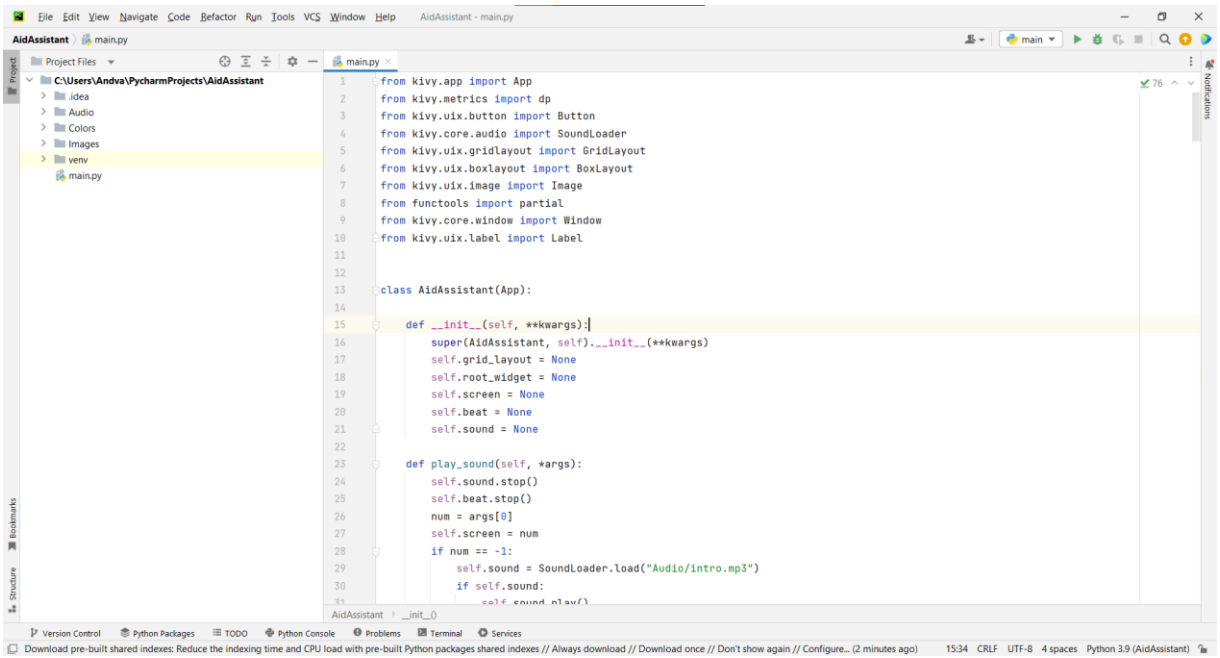


Рисунок 2.3 – Інтерфейс PyCharm

Для реалізації мобільного додатку на Python, потрібен також фреймворк Kivy. Цей фреймворк дозволяє використати мову Python для написання мобільних додатків, а також надає широкий набір віджетів, таких як кнопки, текст, зображення, що потрібні для функціонування додатку. Kivy підтримує різні платформи, що дозволить зручно розробляти додаток на Windows з пізнішим експортом програми на Android.

Рисунок 2.4 – Логотип Kivy

Для озвучування усіх потрібних інструкцій було вирішено застосувати технології штучного інтелекту, що так розвивається у цей час. Завдяки цій технології, написані інструкції першої допомоги можуть бути конвертовані у голос. В якості сервісу, що конвертує текст у голос, використовуючи ці технології був обраний онлайн сервіс Voicer, адже він містить опцію української мови з доволі чітким, нейтральним голосом. Для створення аудіо, потрібно обрати мову та голос, яким ви хочете озвучити текст та просто вписати ваш текст у поле вводу та натиснути «Voice Selected Lines». Після цього введений текст буде озвучений обраним голосом.

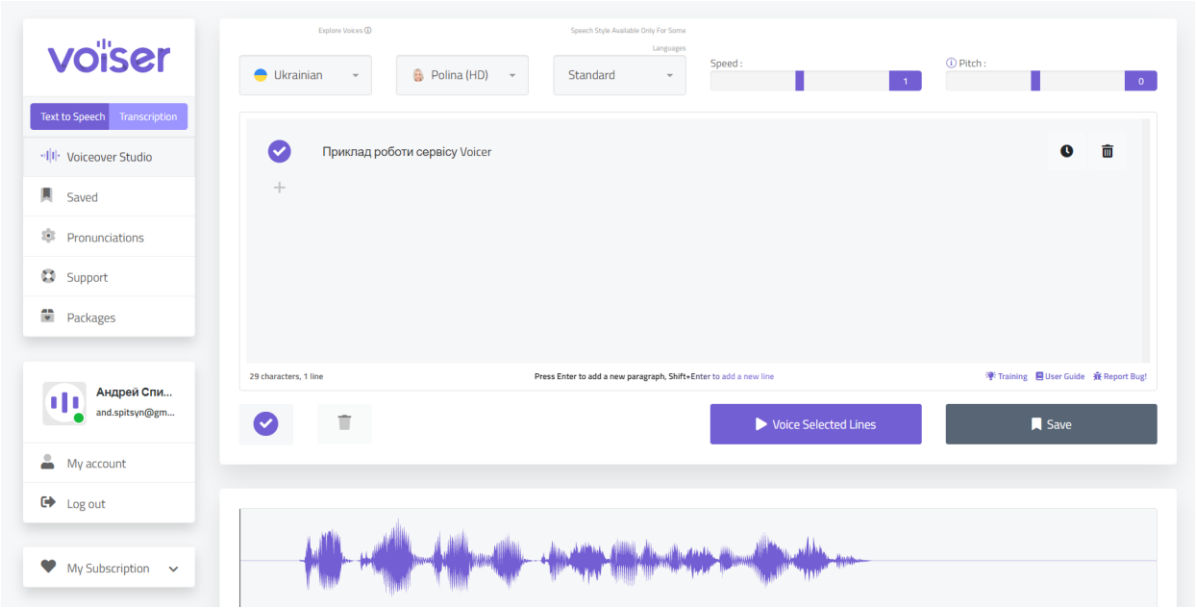


Рисунок 2.5 – Онлайн сервіс Voicer

Для розробки інтерфейсу було вирішено використати онлайн сервіс Figma. Оскільки це безкоштовний онлайн сервіс, що призначений безпосередньо для розробки дизайну. Крім того на Figma можна зберігати розроблені дизайни.

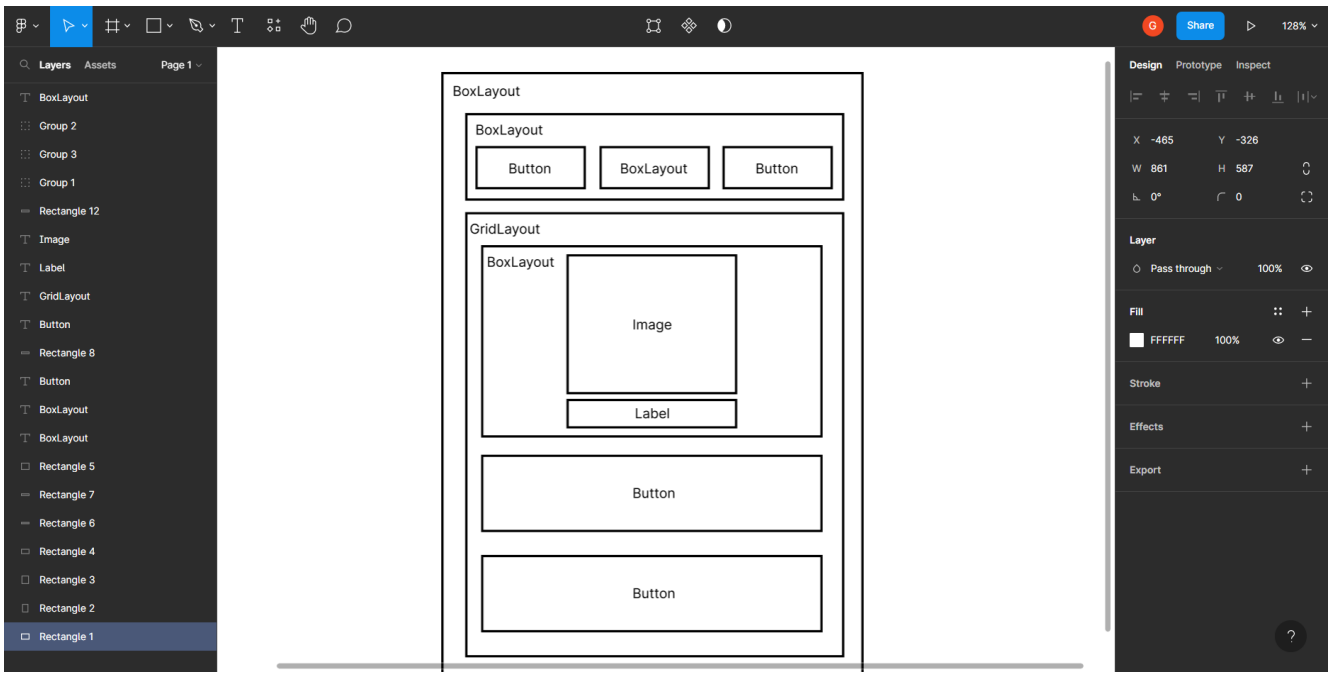


Рисунок 2.6 – Інтерфейс сервісу Figma

Для створення усіх потрібних діаграм було прийнято рішення використати онлайн сервіс draw.io. Адже він є безкоштовний та надає зручний функціонал створення різноманітних діаграм.

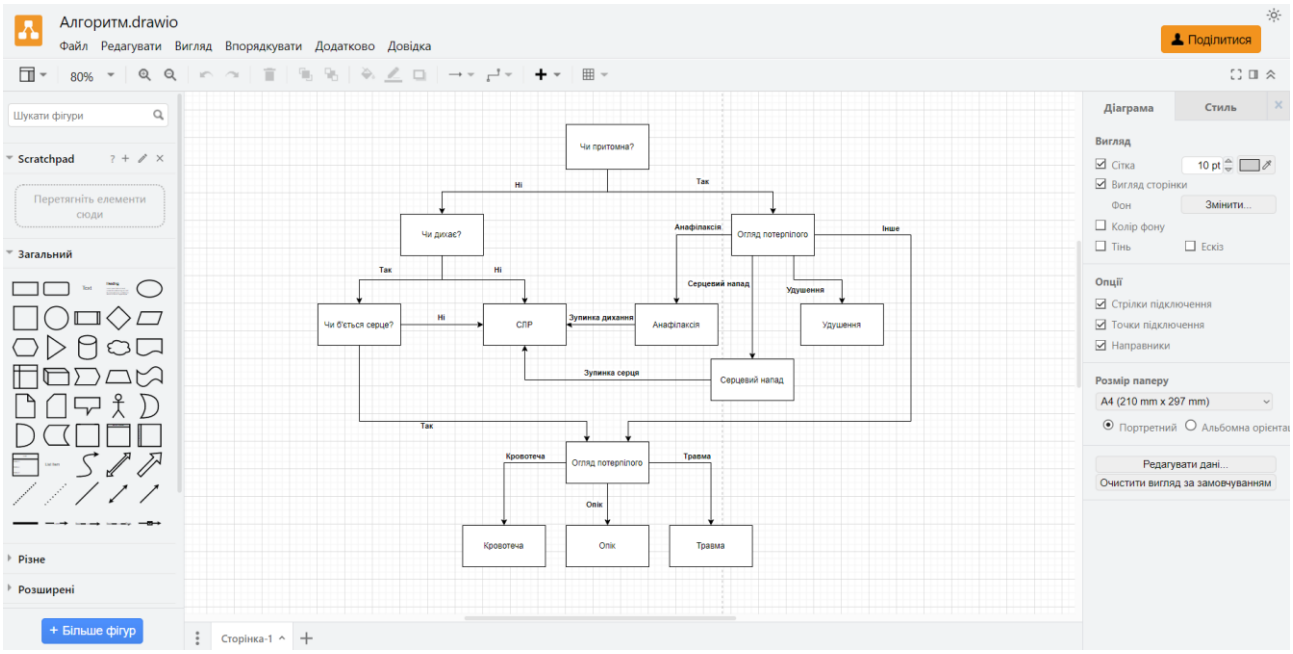


Рисунок 2.7 – Інтерфейс сервісу draw.io

2.2 Визначення функціональних та нефункціональних вимог до програми

Функціональні вимоги:

1. Інтерактивність

   Програма має забезпечити інтерактивність та можливість просуватись по алгоритму, виходячи на потрібну інструкцію шляхом обирання одного із запропонованих варіантів, а також запускати інші функції;

2. Прослуховування аудіо

   Програма має надавати аудіо інструкції щодо кожного етапу огляду потерпілого, а також його стабілізування;

3. Повтор аудіо

   Користувач повинен мати можливість повторювати усі інструкції натисканням відповідної кнопки;

4. Вибір зі списку інструкцій

   Програма має давати можливість обрати потрібний розділ зі списку на будь-якій стадії роботи програми;

5. Перезапуск алгоритму

   Програма має давати можливість починати алгоритм спочатку, натисканням відповідної кнопки.

Нефункціональні вимоги:

1. Інтерфейс

   Інтерфейс у програмі має бути вільний від зайвих деталей, а також мати просту для розуміння навігацію між екранами. Простота інтерфейсу є важливою вимогою через терміновіть у якій це програмне забезпечення буде використовуватись. Бо коли на екрані відсутня значна кількість елементів, це виділяє ті що лишилися. І погляд користувача інстинктивно буде спрямований на виділені елементи.

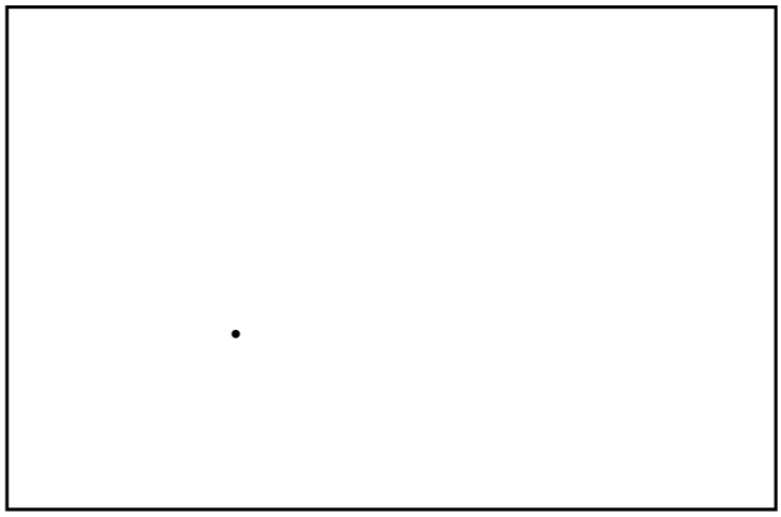


Рисунок 2.8 – На цьому зображенні погляд людини скоріш за все буде спрямований на єдиний елемент - точку

2. Мова

   Програма має бути повністю українською мовою. Враховуючи текст та аудіо інструкції.

2.3 Розробка користувацького інтерфейсу

На головних екранах додатку усі головні елементи знаходяться на одній вертикалі по центру екрану, що позбавляє користувача від необхідності бігати очами по екрану в пошуках потрібної інформації. В якості кольорової палітри біло вирішено використати білий, сірий та червоний кольори, де білий є домінуючим кольором, сірим виділяються кнопки, а червоним кнопки підсвічуються при натисканні.

Екран вибору варіанту є одним із самих розповсюджених, адже на ньому користувач відповідає на питання про стан потерпілого, тим самим веде програму до потрібної інструкції. На цьому екрані одразу написано головне запитання, тому за бажанням користувач може одразу натиснути на потрібну кнопку з відповіддю та перейти на наступне питання. На цьому екрані також знаходяться схематичне зображення, що доповнює поточне питання і кнопка повтору аудіо, на випадок якщо користувачу потрібно буде переслухати інструкцію. Останнє, що є на цьому екрані – це кнопка переходу до екрану зі списком усіх інструкцій.

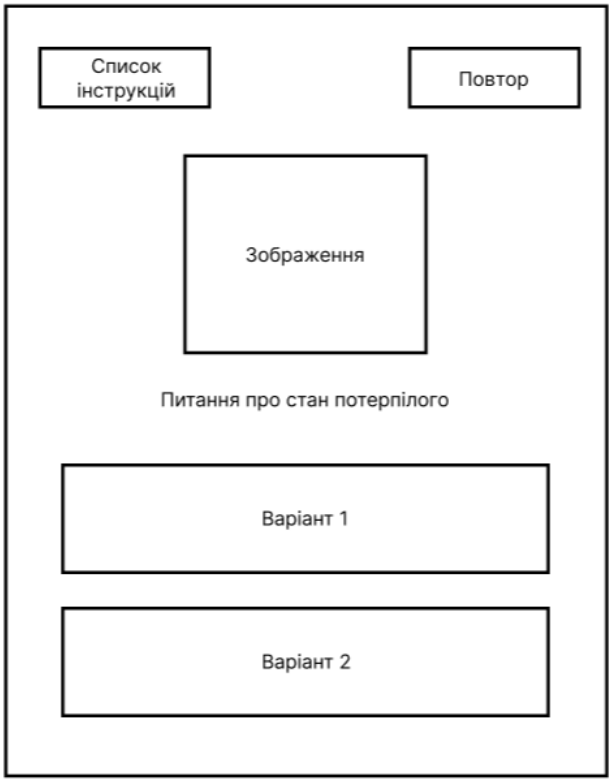


Рисунок 2.9 Екран вибору варіанту

Екран зі списком усіх інструкцій було вирішено створити як таблицю з двома стовпчиками. Якщо розмістити варіанти в один ряд, вони будуть або затонкі, або змусять додавати функцію прокрутки, і таким чином частина інструкцій будуть розміщені за межами екрану користувача. Тому для підвищення швидкості використання додатку та більшої його простоти було прийнято рішення відмовитись від цих варіантів та зробити усі елементи одночасно видимими. На цьому екрані, на відміну від усіх інших, відсутня кнопка повтору, адже тут не програвається ніяке аудіо. Також, оскільки це і є екран з усіма варіантами, кнопка переходу на нього замінюється на кнопку повернення до того кроку, на якому кнопка була натиснута.

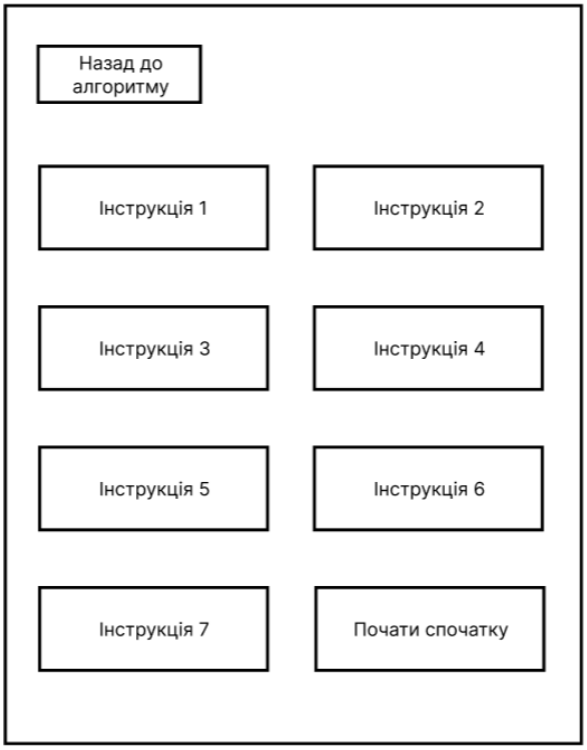


Рисунок 2.10 Екран усіх інструкцій

Екран інструкції являє собою набір схематичних зображень із підписами, необхідний для покращення розуміння інструкції. Він також, як і екран вибору варіанту містить кнопку повтору аудіо, разом із кнопкою переходу до екрану з усіма варіантами.

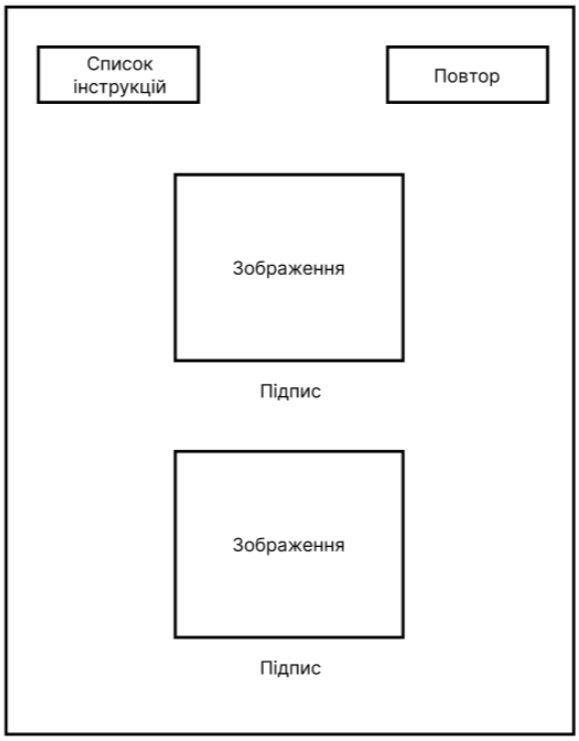


Рисунок 2.11 Екран інструкції

2.4 Проектування архітектури програмного забезпечення

Для роботи застосунку потрібно створити набір екранів на кожну інструкцію допомоги та огляду, що були описані у розділі 1.2. При натисканні на кнопки варіантів, програма буде завантажувати наступний потрібний екран. Порядок екранів буде визначатися згідно алгоритму програми.

Спираючись на проаналізовані наукові джерела, пов'язані з першою допомогою та на написані інструкції у розділі 1.2, був сформований такий алгоритм програми:

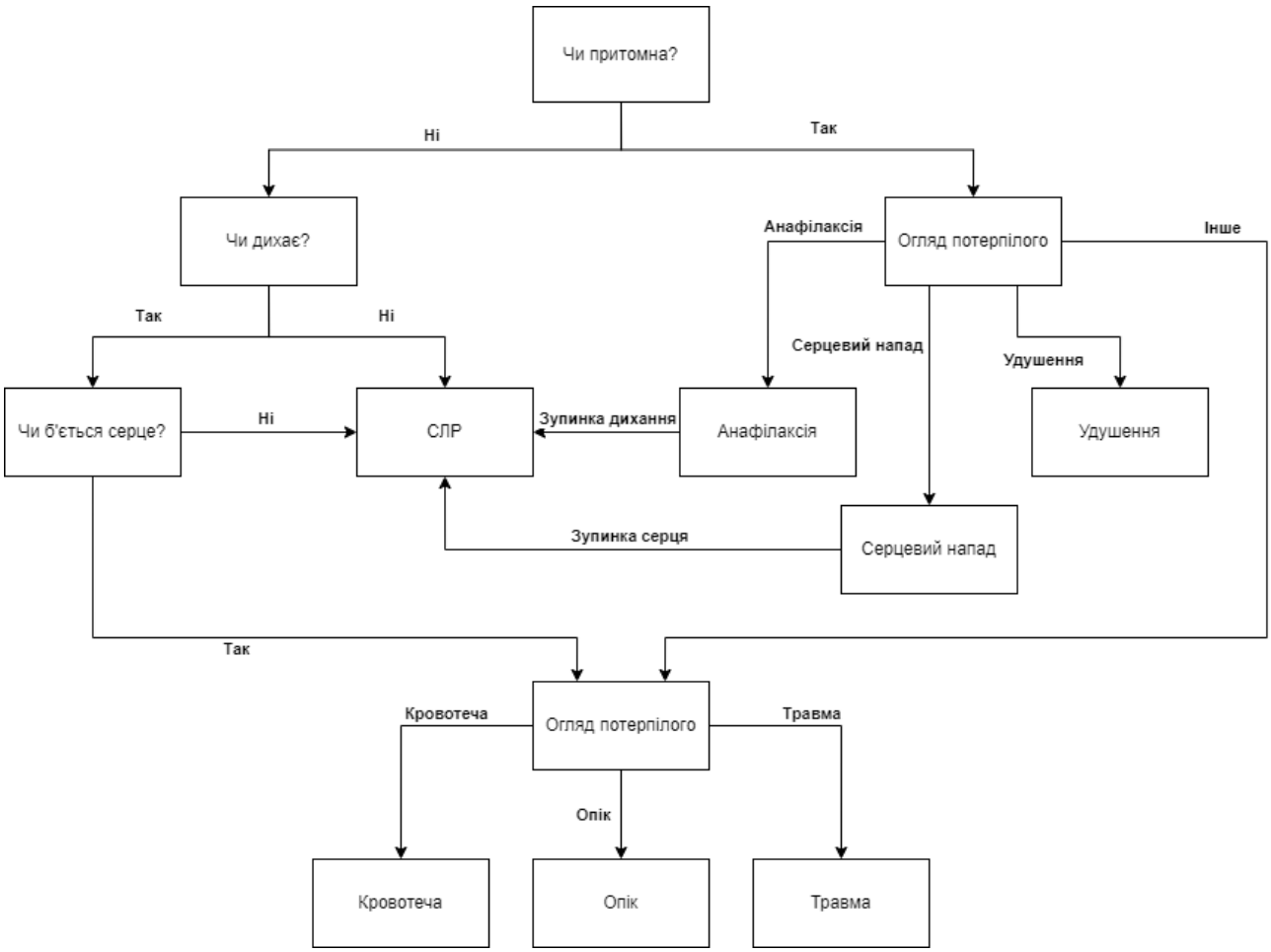


Рисунок 2.12 – Алгоритм програми

Цей алгоритм стане основою при створенні переходів між екранами у програмі. Але спочатку потрібно створити самі екрани. Оскільки програма призначена працювати на різних телефонах з різними розмірами, варто зробити весь інтерфейс масштабованим. Таким чином він підійде на більшість пристроїв.

Таким чином можна побудувати діаграму варіантів використання (use case diagram).

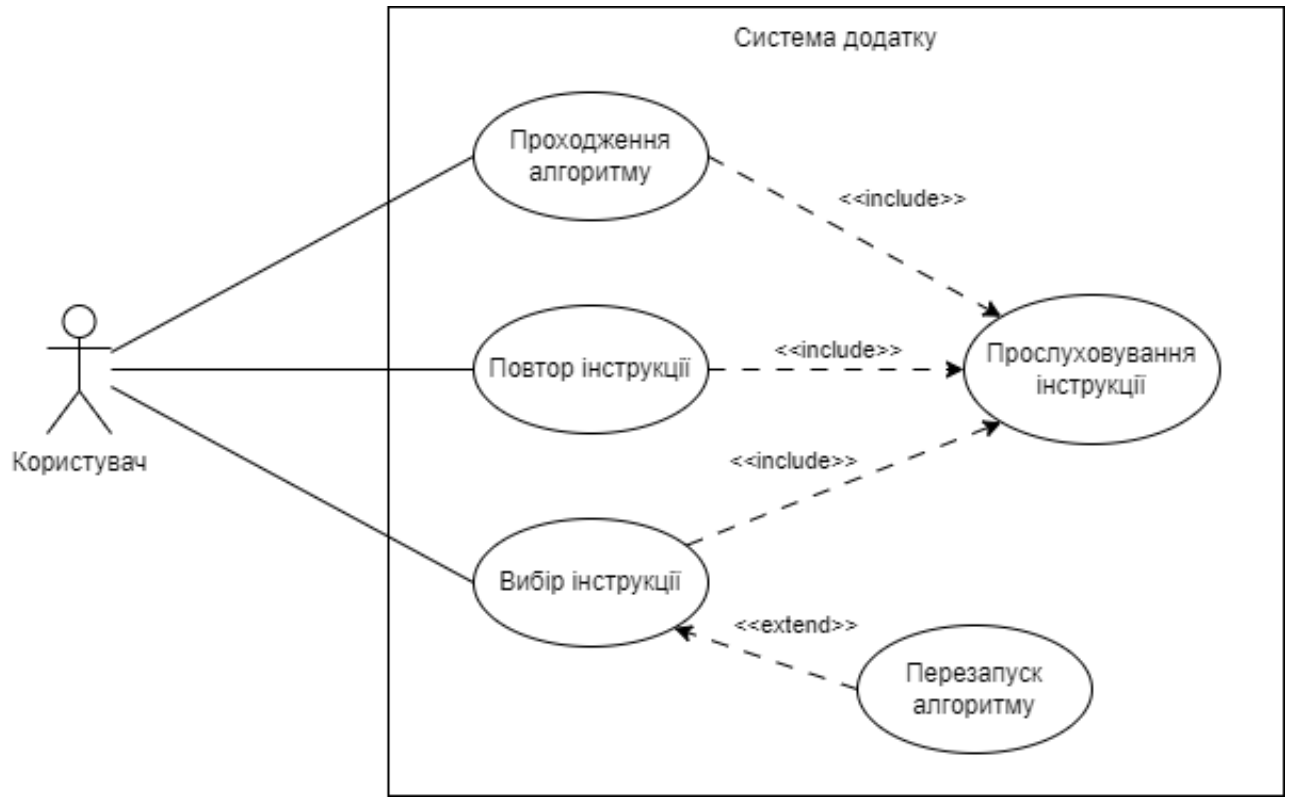


Рисунок 2.13 – Діаграма варіантів використання

Створення інтерфейсу у Kivy може місцями бути комплексним через його систему розташовування елементів. Для відображення будь-чого на екрані у Kivy, потрібно повернути Kivy елемент BoxLayout, що буде містити усі потрібні віджети. Якщо просто додавати ці віджети, без вказання ніяких інших параметрів, вони просто вишукуються в один ряд та розтягнуться на весь екран, тобто BoxLayout. Для потрібного розташування, при додаванні масштабованих віджетів, потрібно використовувати додаткові елементи, що будуть їх коректно розподіляти.

Так, на прикладі інтерфейсу з рисунку 2.11, ми створюємо головний BoxLayout, що містить 2 елементи: BoxLayout і GridLayout, які обидва містять інші елементи. У цьому BoxLayout знаходяться 2 кнопки, що розділені додатковим, пустим BoxLayout. У GridLayout також знаходяться 2 кнопки та BoxLayout, але на цей раз він потрібен не для розділення, а для об’єднання зображення з його підписом.

Таким чином виходять така архітектура інтерфейсу:

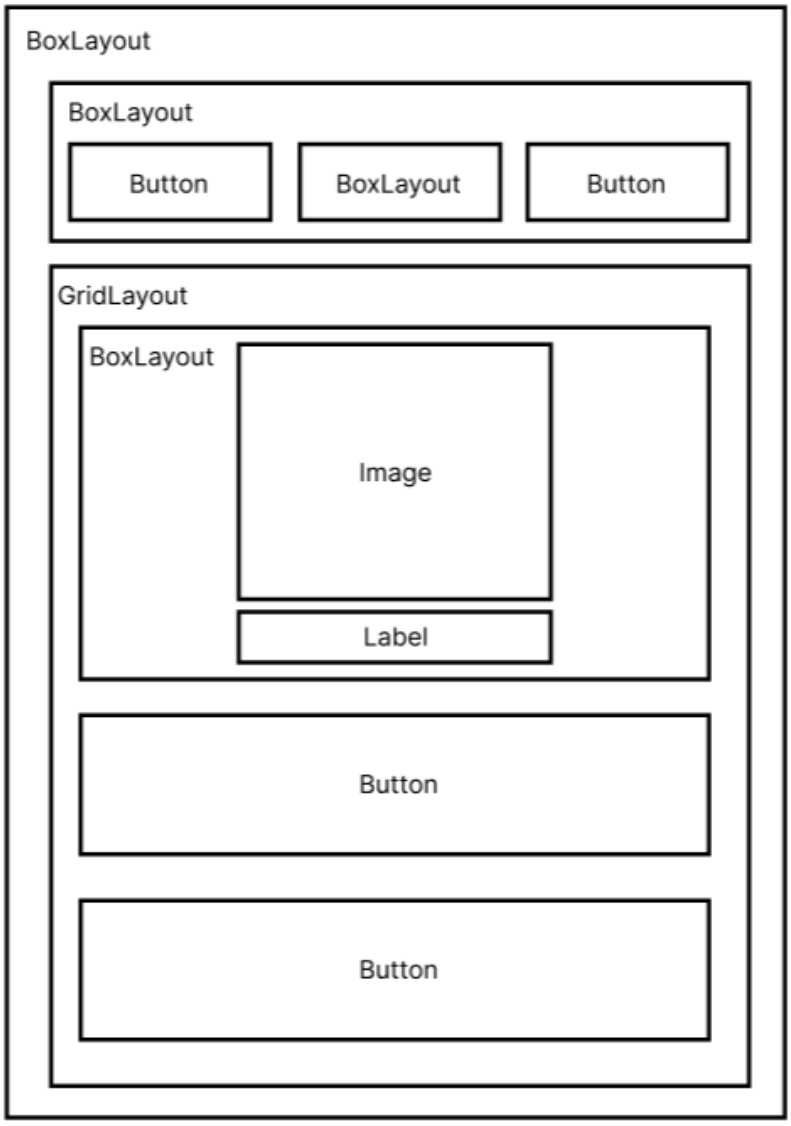


Рисунок 2.14 – Архітектура екрану вибору варіанта

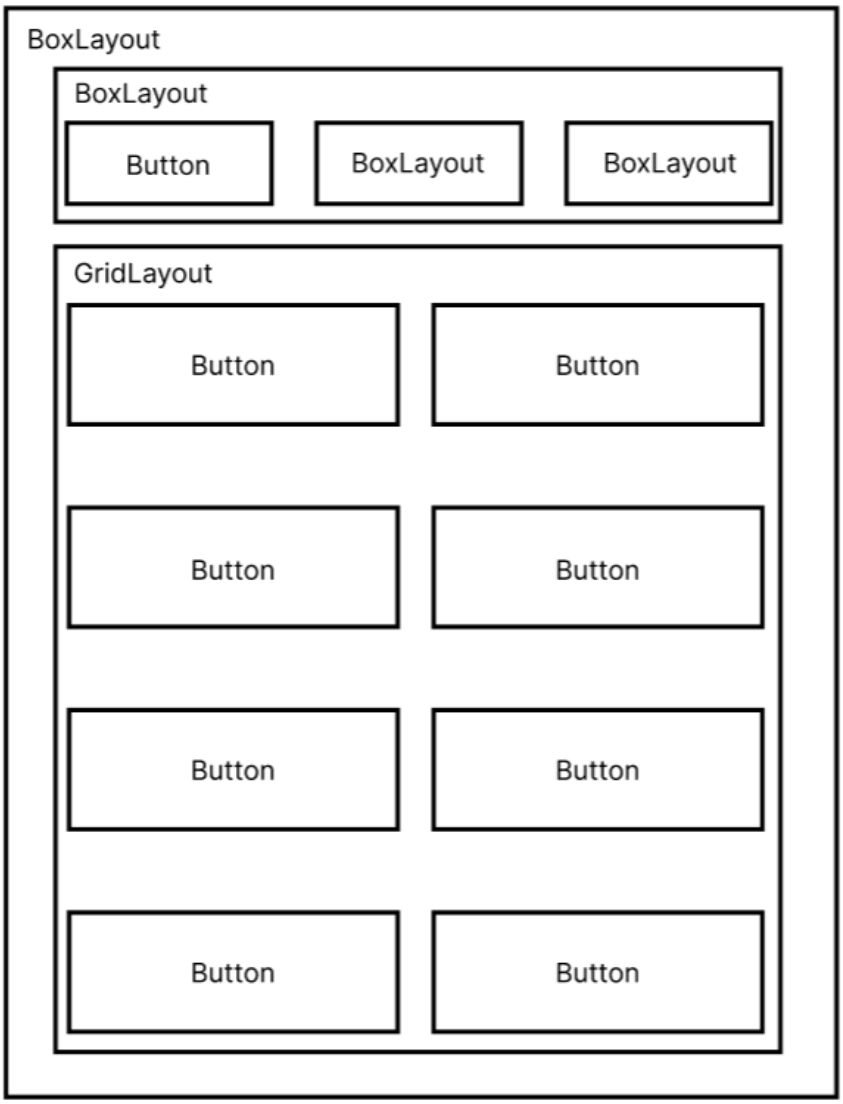


Рисунок 2.15 – Архітектура екрану зі списком інструкцій

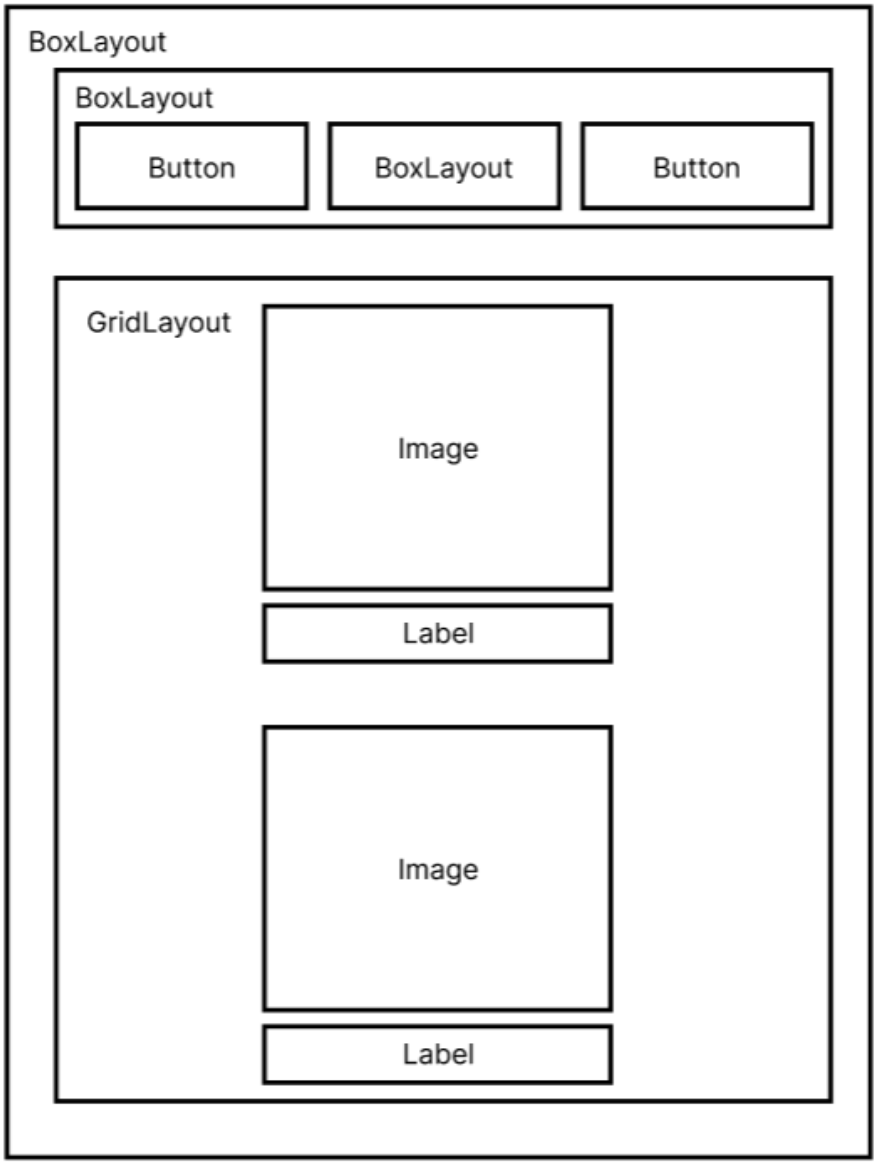


Рисунок 2.16 – Архітектура екрану інструкції

3. РОЗРОБКА ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Розробка системи програвання аудіо

Для відтворення звуку буде використовуватись такий Kivy елемент як SoundLoader. Він приймає аудіо файл та програє його. Використовуватись SoundLoader буде одразу при запуску кожного екрану, окрім екрану зі списком інструкцій. Це робиться завдяки таким командам:

```
self.sound = SoundLoader.load("Audio/intro.mp3")
if self.sound:
    self.sound.play()
```

Рисунок 3.1 – Програвання аудіо

Створюється SoundLoader змінна sound, якій передається файл intro.mp3. Далі йде перевірка, якщо змінна sound існує, то програти її файл. Таким чином ми запускаємо перший аудіо файл, який, згідно алгоритму програми, є перевіркою потерпілого на притомність.

Для відтворення усіх наступних файлів варто створити спеціальний метод.

```
def play_sound(self, *args):
    self.sound.stop()
    num = args[0]
    self.screen = num
    if num == -1:
        self.sound = SoundLoader.load("Audio/intro.mp3")
        if self.sound:
            self.sound.play()
```

Рисунок 3.2 – Метод play_sound

Цей метод приймає аргумент num, який відповідає номеру аудіо, яке необхідно програти. Для того, щоб аудіо не повторювалося, перед його програванням варто додати команду stop(). Після чого вже створити перевірку та запуск аудіо за вказаним номером.

За вимогами, програмі також потрібно програвати аудіо метроному. Для цього за аналогією потрібно створити змінну beat, якій надати потрібний файл. Після чого у методі play_sound, до запуску інструкції серцево-легеневої реанімації додати програвання файлу зі змінної beat.

```
if num == 2:
    self.sound = SoundLoader.load("Audio/cpr.mp3")
    if self.sound:
        self.sound.play()
    if self.beat:
        self.beat.play()
    self.assemble_cpr()
```

Рисунок 3.3 – Програвання метроному одночасно з інструкцією

Користуючись зазначеним у розділі 2.1 сервісом для конвертації тексту у голос Voicer, було створено набір потрібних інструкцій.

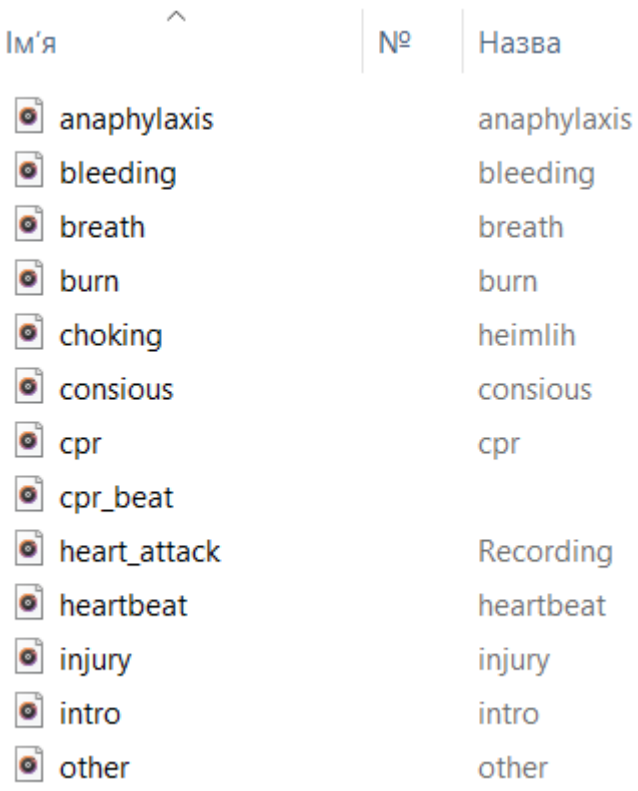


Рисунок 3.4 – Аудіо файли у папці

3.2 Розробка методів інтерактивності

Як було вже зазначено у розділі 2.4, для створення інтерфейсу Kivy потрібно повернути елемент BoxLayout. Тому ми маємо такий метод build:

```
def build(self):
    Window.clearcolor = (1, 1, 1, 1)
    self.root_widget = BoxLayout(orientation='vertical')
    return self.root_widget
```

Рисунок 3.5 – Початковий метод build

В цій частині коду був створений основний екран. Заданий білий колір фону вікна та повернений BoxLayout. Але поки він є пустим, тому при запуску відобразиться лише білий фон.

Для виправлення цього варто додати потрібні віджети. І оскільки у спроектованій програмі буде значна кількість елементів, що повторюються, буде доречним створити для кожного елементу окремий метод, щоб не копіювати його створення багато разів, порушуючи принцип SOLID. Спершу варто почати з найчисленнішого елементу – кнопки.

```
def create_button(self, text, num):
    button = Button(text=f'[b]{text}[/b]', markup=True, color=(0.15, 0.15, 0.15, 1), font_size='20sp',
                    size_hint=(1, 0.5), background_color=(0.85, 0.85, 0.85, 1),
                    background_normal='', background_down='Colors/Red.jpg')
    button.bind(on_release=partial(self.play_sound, num))
    self.grid_layout.add_widget(button)
```

Рисунок 3.6 – Метод create_button

Цей метод приймає такі змінні як text та num, що передають текст, який має бути на кнопці та номер, за яким викликається аудіо файл, відповідно. У самому методі створюється кнопка з однаковими для всіх параметрами, окрім переданого тексту. Кнопка прив’язується до методу playSound, що програє потрібний аудіо файл за вказаним номером. І зрештою використано команду add_widget для додання кнопки вже до layout.

```
def create_image(self, text, file):
    box_layout = BoxLayout(orientation='vertical', size_hint=(1, 1))
    image = Image(source=file, size_hint=(1, 0.8))
    label = Label(text=f'[b]{text}[/b]', markup=True, font_size='20sp',
                  size_hint=(1, 0.2), color=(0, 0, 0, 1))
    box_layout.add_widget(image)
    box_layout.add_widget(label)
    self.grid_layout.add_widget(box_layout)
```

Рисунок 3.7 – Метод create_image

Другий метод створює блок, що складається з зображення та підпису до нього, отже приймає він text для підпису та file для зображення. Створюється BoxLayout, у нього створюються та додаються image та label, після чого до

головного layout додається увесь блок, тобто BoxLayout. Варто зазначити, що для того щоб зображення та його підпис не зайняли по 50% від блоку, як зробив би Kivy за замовчуванням, було додано size_hint, у якому для зображення виставлено значення у 80% висоти від доступного місця та для підпису – 20%.

```
def create_top(self, num, screen):
    if num == 0:
        box_layout = BoxLayout(orientation='horizontal', padding=dp(10), spacing=dp(10), size_hint=(1, 0.11))
        button_all = Button(text='[b]СПИСОК[/b]', markup=True, color=(0.15, 0.15, 0.15, 1), font_size='20sp',
                            size_hint=(1, 1), background_color=(0.85, 0.85, 0.85, 1),
                            background_normal='', background_down='Colors/Red.jpg')
        button_all.bind(on_release=partial(self.replace, 0))
        blank_widget = BoxLayout()
        button_repeat = Button(text='[b]ПОВТОР[/b]', markup=True, color=(0.15, 0.15, 0.15, 1), font_size='20sp',
                               size_hint=(1, 1), background_color=(0.85, 0.85, 0.85, 1),
                               background_normal='', background_down='Colors/Red.jpg')
        button_repeat.bind(on_release=partial(self.play_sound, screen))
        box_layout.add_widget(button_all)
        box_layout.add_widget(blank_widget)
        box_layout.add_widget(button_repeat)
        self.root_widget.add_widget(box_layout)
    elif num == 1:
        box_layout = BoxLayout(orientation='horizontal', padding=dp(10), spacing=dp(10),
                               size_hint_y=None, height=Window.height * 0.1)
        button = Button(text='[b]НАЗАД[/b]', markup=True, color=(0.15, 0.15, 0.15, 1), font_size='20sp',
                        size_hint=(1, 1), background_color=(0.85, 0.85, 0.85, 1),
                        background_normal='', background_down='Colors/Red.jpg')
        button.bind(on_release=partial(self.replace, 1))
        blank_widget1 = BoxLayout()
        blank_widget2 = BoxLayout()
        box_layout.add_widget(button)
        box_layout.add_widget(blank_widget1)
        box_layout.add_widget(blank_widget2)
        self.root_widget.add_widget(box_layout)
```

Рисунок 3.8 – Метод create_top

Метод створення шапки кожного екрану. Шапка складається з трьох елементів, розташованих горизонтально один за одним. На усіх екранах де програється аудіо цими елементами є: кнопка переходу до списку інструкцій, пустий блок, який слугує розмежувачем та кнопка повтору аудіо. А оскільки на власне екрані списку аудіо не програється, замість кнопки повтору створюється ще один пустий блок.

За вимогами, програма передбачає інтерактивність та можливість перемикатися між існуючими екранами. Для цього було вирішено створити на

кожен існуючий екран по власному методу та викликати його, коли він потрібен. Ці методи фактично збирають цей екран з самого верху, додаючи потрібні для кожного екрану елементи.

Так, першим екраном є перевірка на притомність потерпілого.

```
def assemble_consciousness(self):
    self.root_widget.clear_widgets()
    self.create_top(0, -1)
    self.grid_layout = GridLayout(cols=1, padding=dp(10), spacing=dp(10), size_hint=(1, 1))
    self.grid_layout.bind(minimum_height=self.grid_layout.setter('height'))
    self.root_widget.add_widget(self.grid_layout)
    self.create_image('ПЕРЕВІРТЕ ЧИ ПРИТОМНА ЛЮДИНА', 'Images/consciousness.jpg')
    self.create_button('ПРИТОМНА', 7)
    self.create_button('НЕПРИТОМНА', 0)
```

Рисунок 3.9 – Метод assemble_consciousness

Перед доданням віджетів, у цих методах збірки потрібно додавати команду clear_widgets() для видалення усіх поточних елементів. Далі створюється GridLayout, у який додаються усі основні елементи: зображення з підписом та дві кнопки варіантів. На прикладі можна побачити, що при створенні кнопки, їй надається номер, такий як 7 або 0. За ним і буде викликатися відповідний екран при натисканні.

За таким самим прикладом створюються відповідно такі методи:

1. assemble_breath, для екрану перевірки дихання;
2. assemble_heartbeat, для екрану перевірки серцебиття;
3. assemble_other, для екрану перевірки на різноманітні тілесні ушкодження;
4. assemble_check, для екрану перевірки притомної людини;
5. assemble_cpr, для екрану інструкції серцево-легеневої реанімації;
6. assemble_bleed, для екрану інструкції допомоги при кровотечах;
7. assemble_burn, для екрану інструкції допомоги при опіках;
8. assemble_injury, для екрану інструкції допомоги при травмах;
9. assemble_heart, для екрану інструкції допомоги при серцевому нападі;
10. assemble_choking, для екрану інструкції допомоги при удушенні;

11. assemble_anaphylaxis, для екрану інструкції допомоги при анафілактичному шоці;

```
def assemble_cpr(self):
    self.root_widget.clear_widgets()
    self.create_top(0, 2)
    self.grid_layout = GridLayout(cols=1, padding=dp(10), spacing=dp(10), size_hint=(1, 1))
    self.grid_layout.bind(minimum_height=self.grid_layout.setter('height'))
    self.root_widget.add_widget(self.grid_layout)
    self.create_image('30 КОМПРЕСІЙ', 'Images/cpr1.jpg')
    self.create_image('2 ВДИХИ', 'Images/cpr2.jpg')
```

Рисунок 3.10 – Метод assemble_cpr

Для коректної реалізації функції перемикання на список інструкцій і назад, було створено метод replace, що зупиняє усе поточне аудіо та запускає екран з усіма варіантами. Якщо метод викликаний вже на екрані зі списком, то відкривається екран, що був відкритий попереднім.

```
def replace(self, *args):
    self.sound.stop()
    self.beat.stop()
    num = args[0]
    if num == 0:
        self.assemble_all()
    elif num == 1:
        self.play_sound(self.screen)
```

Рисунок 3.11 – Метод replace

Таким чином, залишилося лише додати команду виклику методу assemble_consciousness у головний метод build. Тим самим при запуску програми, користувачу одразу буде виведений екран з першим питанням алгоритму. Після чого він вже має можливість іти далі по алгоритму, або перейти до екрану з усіма інструкціями та обрати потрібну опцію.

Так, розробивши програму у єдиному класі додатку ми маємо зрештою таку діаграму класів:

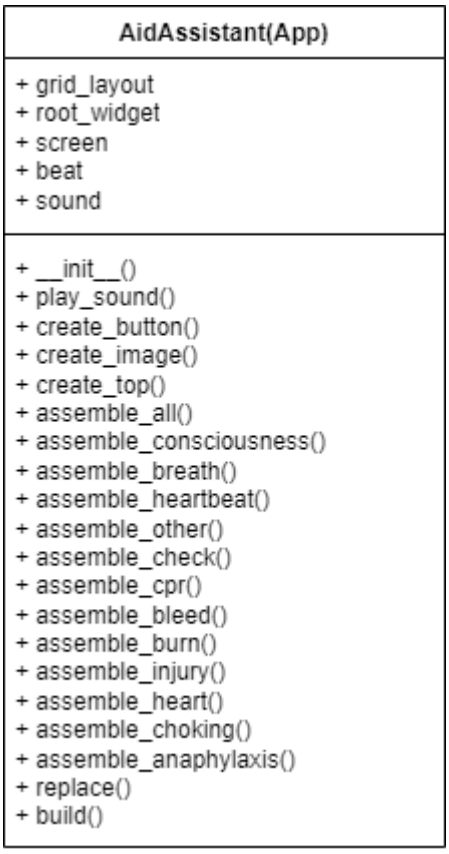


Рисунок 3.12 – Діаграма класів

Завдяки сервісу Google також було знайдено необхідні для розуміння інструкцій схематичні зображення.

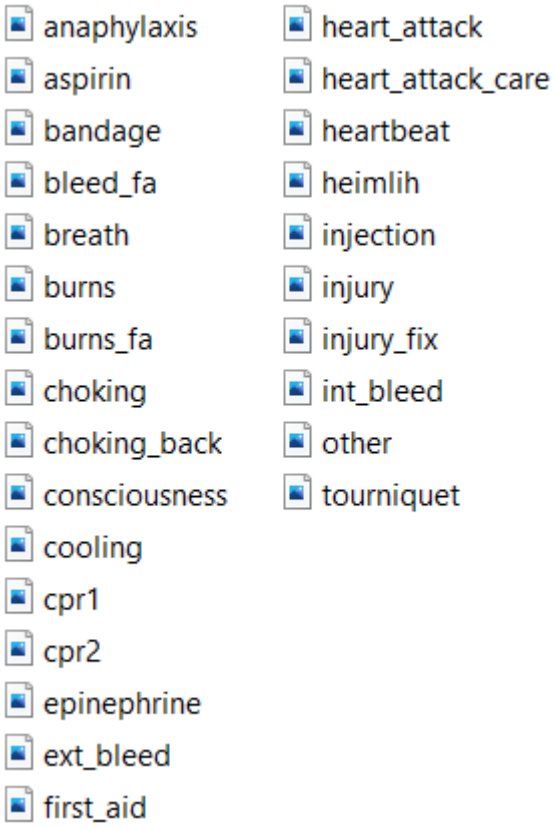


Рисунок 3.13 – Зображення у папці

Після додання усіх необхідних зображень та аудіо файлів, програма має такий вигляд:

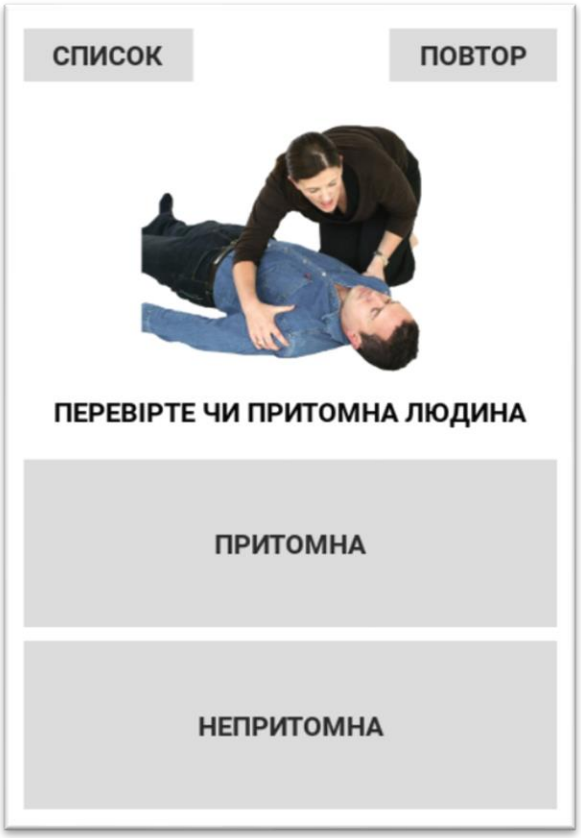


Рисунок 3.14 – Готовий інтерфейс вибору варіанту

Рисунок 3.15 – Готовий інтерфейс списку усіх інструкцій

Рисунок 3.16 – Готовий інтерфейс інструкції

3.3 Валідація програмного забезпечення та проведення тестування

В якості методології тестування було обрано Тестування білої скриньки, оскільки відома уся внутрішня структура програми.

Функціональне тестування:

1. Інтерактивність

   Користувач має можливість вільно пересуватись додатком та відкривати усі необхідні екрани;

2. Прослуховування аудіо

   При відкритті кожного екрану, програється відповідний до нього аудіо файл;

3. Повтор аудіо

Була додана кнопка повтору аудіо, яка працює із будь-яким поточним файлом;

4. Вибір зі списку інструкцій

   Була додана кнопка списку інструкцій, яка відкриває екран з усіма варіантами. Натискання на будь-який варіант запускає відповідну інструкцію;

5. Перезапуск алгоритму

   До екрану з усіма інструкціями була додана кнопка для запуску алгоритму із самого початку, яка працює на будь-якому етапі алгоритму;

Тестування інтерфейсу:

Усі екрани додатку відкриваються належним чином, згідно розробленого інтерфейсу. Усі інструкції додатку та його інтерфейс реалізовані українською мовою.

3.4 Підсумки розробки та тестування

Було розроблено програмне забезпечення для підтримки процесу надання першої допомоги мовою Python. Проведено тестування білої скриньки по усьому функціоналу програми.

Після проведеного тестування можна зробити висновок, що програма відповідає усім поставленим вимогам, та усі виконані тести не виявили жодного багу.

ВИСНОВКИ

У першому розділі цієї дипломної роботи був проведений аналіз предметної області. На основі научної літератури, пов’язаної з наданням першої допомоги було написано набір інструкцій для майбутнього алгоритму. Загалом список містить 7 головних інструкцій надання першої допомоги, а також 5 інструкцій для огляду потерпілого.

Виконаний аналіз додатків аналогів. Виявлені позитивні та негативні сторони кожного з додатків. Сформована таблиця-підсумок проведеного аналізу. Виявлено, що головним недоліком програмного забезпечення, що надає інструкції першої допомоги є недостатня оперативність додатку, тобто незрозумілий інтерфейс, або суцільний, неформатований текст без схематичних зображень.

Спираючись на проведені аналізи була поставлена задача – зменшити час надання першої допомоги за рахунок розробки мобільного додатку.

У другому розділі дипломної роботи, орієнтуючись на поставлену задачу, було здійснено вибір технологій та платформи розробки. Було прийнято рішення розроблювати мобільний додаток для платформи Android, з використанням мови програмування Python та його фреймворку Kivy. Для створення аудіозаписів інструкцій, було вирішено застосувати технології штучного інтелекту та конвертувати текстові інструкції у голос на спеціалізованому онлайн сервісі Voicer.

Далі було сформовано необхідні функціональні та нефункціональні вимоги за якими буде проводитись розробка програмного продукту та його тестування.

Завдяки онлайн сервісу Figma було розроблено увесь потрібний інтерфейс мобільного додатку з мінімалістичним дизайном для спрощення розуміння додатку користувачем.

Користуючись вже написаними інструкціями був сформований єдиний алгоритм першої допомоги. Після чого було спроектовано та описано архітектуру розроблюваного програмного забезпечення, спираючись на сформовані вимоги та розроблений інтерфейс.

У третьому розділі, користуючись готовою архітектурою, було розроблено програмне забезпечення. А саме:

1. Методи, що відповідають за створення елементів інтерфейсу;
2. Методи, що збирають відповідні екрани програми, користуючись віджетами Kivy та іншими, попередньо розробленими методами;
3. Було зібрано, налаштовано та організовано їх роботу.

Проведено тестування розробленого програмного забезпечення, яке не виявило багів та підтвердило виконання усіх поставлених вимог та якість програмного продукту.

ПЕРЕЛІК ПОСИЛАНЬ

1. Comprehensive Guide for First Aid & CPR [Електронний ресурс] // Canadian Red Cross – Режим доступу до ресурсу: https://www.redcross.ca/crc/documents/comprehensive_guide_for_firstaidcpr_en.pdf.

2. Furst J. The Complete First Aid Pocket Guide / John Furst., 2018. – 284 с. – (Adams Media). – (1st edition).

3. Top 5 First Aid Apps for Your Smart Phone [Електронний ресурс] – Режим доступу: https://www.mountainmanmedical.com/top-5-first-aid-apps-for-your-smart-phone/.

4. Voicer.net [Електронний ресурс] – Режим доступу: https://voiser.net/account/text-to-speech/.

5. Mobile Operating System Market Share Ukraine [Електронний ресурс] – Режим доступу: https://gs.statcounter.com/os-market-share/mobile/ukraine.

6. Figma [Електронний ресурс] – Режим доступу: www.figma.com.

7. draw.io [Електронний ресурс] – Режим доступу: https://www.drawio.com.

8. Many UK adults not confident they can administer first aid in an emergency [Електронний ресурс] // ZME Science. – 2022. – Режим доступу до ресурсу: https://www.zmescience.com/science/many-uk-adults-not-confident-they-can-administer-first-aid-in-an-emergency/.

9. Knowledge of first aid in school students and teachers [Електронний ресурс] // ResearchGate. – 2022. – Режим доступу до ресурсу: https://www.researchgate.net/publication/362631298_Knowledge_of_first_aid_in_school_students_and_teachers.
https://cs.emis.de/LNI/Proceedings/Proceedings259/1745.pdf.

10. Перша допомога — МФЧХ і ЧП [Електронний ресурс] // IFRC. – Режим доступу до ресурсу: https://play.google.com/store/apps/details?id=com.cube.gdpc.fa.

11. First Aid Fast [Електронний ресурс] // Rapid Response Revival PTY. – Режим доступу до ресурсу: https://play.google.com/store/apps/details?id=com.app.firstaidfastrrr.

12. First Aids and Emergency techniques [Електронний ресурс] // Fatbelly. – Режим доступу до ресурсу: https://play.google.com/store/apps/details?id=com.fatbelly.firstaidsandemergencytechniques.

13. METHOD OF CHOOSING THE PROGRAMMING ENVIRONMENT FOR SOFTWARE [Електронний ресурс] // ResearchGate. – 2021. – Режим доступу до ресурсу: https://www.researchgate.net/publication/360083873_METHOD_OF_CHOOSING_THE_PROGRAMMING_ENVIRONMENT_FOR_SOFTWARE.

14. Preparing Kivy for Android Application Development [Електронний ресурс] // ResearchGate. – 2019. – Режим доступу до ресурсу: https://www.researchgate.net/publication/336529001_Preparing_Kivy_for_Android_Application_Development.

ДОДАТОК А

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка програмного забезпечення для підтримки процесу надання першої допомоги мовою Python

Виконав студент 4 курсу
групи ПД-41
Спіцин Андрій Ярославович
Керівник роботи
К.т.н, доц, доцент кафедри ІПЗ Трінтіна Наталя Альбертівна

Київ – 2023

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- Мета дослідження – зменшити час надання першої допомоги за рахунок застосування мобільного додатку, створеного мовою Python.
- Об'єкт дослідження – процес надання першої допомоги.
- Предмет дослідження – програмне забезпечення для підтримки процесу надання першої допомоги.

2

ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

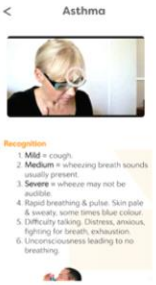
- Проаналізувати літературні джерела та визначити особливості процесу надання першої допомоги.
- Проаналізувати сучасні додатки-аналоги та визначити їх переваги та недоліки.
- Сформулювати вимоги до програмного забезпечення, враховуючи недоліки додатків-аналогів.
- Спроектувати та розробити додаток для підтримки процесу надання першої допомоги.
- Провести тестування розробленого додатку.

3

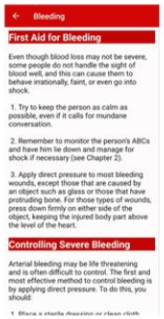
АНАЛІЗ АНАЛОГІВ

Перша допомога — МФЧХ і ЧП

First Aid Fast



First Aids and Emergency techniques



4

АНАЛІЗ АНАЛОГІВ

	Перша допомога — МФЧХ і ЧП	First Aid Fast	First Aids and Emergency techniques	Розроблений додаток
Інструкції першої допомоги	+	+	+	+
Наявність картинок	+	+	–	+
Тести	+	–	–	–
Виклик швидкої	+	–	–	–
Доступність українською	+	–	–	+
Знайти лікарню	–	+	–	–
Повне озвучення	+/–	+/–	–	+

5

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Функціональні вимоги:

- Інтерактивність

Програма має забезпечити інтерактивність та можливість вільно пересуватись додатком

- Прослуховування аудіо

Програма має надавати аудіо інструкції щодо кожного етапу огляду потерпілого, а також його стабілізування

- Повтор аудіо

Користувач повинен мати можливість повторювати усі інструкції натисканням відповідної кнопки

- Вибір зі списку інструкцій

Програма має давати можливість обрати потрібний розділ зі списку на будь-якій стадії роботи програми

- Перезапуск алгоритму

Програма має давати можливість починати алгоритм спочатку, натисканням відповідної кнопки

6

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Нефункціональні вимоги:

- Інтерфейс

Інтерфейс у програмі має бути вільний від зайвих деталей, а також мати просту для розуміння навігацію між екранами. Простота інтерфейсу є важливою вимогою через терміновіть у якій це програмне забезпечення буде використовуватись. Бо коли на екрані відсутня значна кількість елементів, це виділяє ті що лишилися. І погляд користувача інстинктивно буде спрямований на виділені елементи.

- Мова

Програма має бути повністю українською мовою. Враховуючи текст та аудіо інструкції.

7

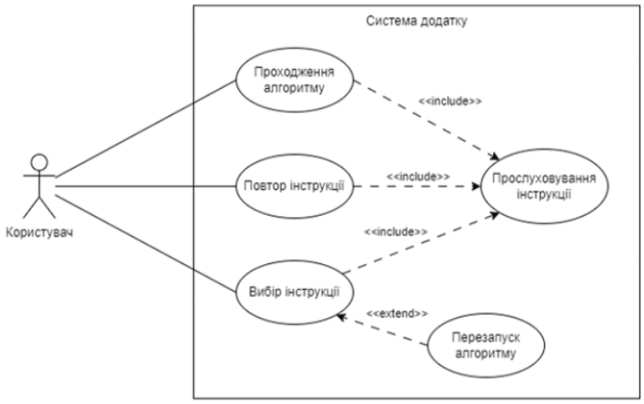
ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

Мова програмування

Фреймворк Kivy

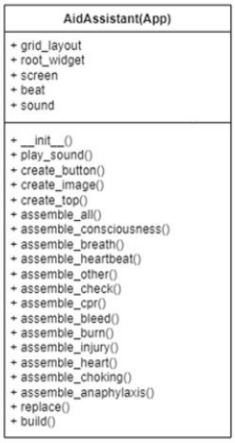
8

ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ



9

ДІАГРАМА КЛАСІВ

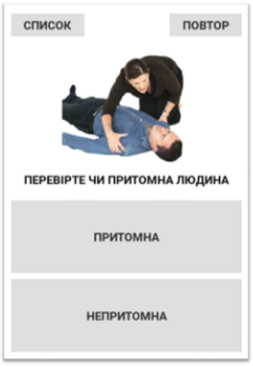


10

СХЕМА РОБОТИ ПРОГРАМИ

11

ЕКРАННІ ФОРМИ



Головний екран додатку

Екран списку інструкцій

Екран інструкції

12

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Спіцин А.Я. Розробка програмного забезпечення для підтримки процесу надання першої допомоги / Спіцин А.Я., Трінтіна Н.А. // Застосування програмного забезпечення в інфокомунікаціних технологіях: Матеріали міжнародної науково-технічної конференції. Збірник тез 20.04.2023, ДУТ, м. Київ – К.: ДУТ, 2023. – С. 28.

2. Спіцин А.Я. Актуальність програмного забезпечення для підтримки процесу надання першої допомоги / Спіцин А.Я., Трінтіна Н.А. // Застосування програмного забезпечення в інфокомунікаціних технологіях: Матеріали міжнародної науково-технічної конференції. Збірник тез 20.04.2023, ДУТ, м. Київ – К.: ДУТ, 2023. – С. 73.

13

ВИСНОВКИ

- Проаналізовано наукові джерела, пов'язані з наданням першої допомоги. Визначено особливості процесу, основні етапи та проблеми що виникають при наданні першої допомоги.
- Проведено аналіз сучасних додатків-аналогів, що надають інструкції першої допомоги. У систем було виявлено проблеми, що впливають на оперативність доступу користувачів до інформації.
- Спроектовано та розроблено додаток, що надає оперативний доступ до інструкцій першої допомоги. Головною перевагою розробленої системи є повне озвучення усіх інструкцій.
- Проведено функціональне тестування, яке не виявило жодних помилок у роботі програми.

14

ДЯКУЮ ЗА УВАГУ!

ДОДАТОК Б

```
from kivy.app import App
from kivy.metrics import dp
from kivy.uix.button import Button
from kivy.core.audio import SoundLoader
from kivy.uix.gridlayout import GridLayout
from kivy.uix.boxlayout import BoxLayout
from kivy.uix.image import Image
from functools import partial
from kivy.core.window import Window
from kivy.uix.label import Label


class AidAssistant(App):

    def __init__(self, **kwargs):
        super(AidAssistant, self).__init__(**kwargs)
        self.grid_layout = None
        self.root_widget = None
        self.screen = None
        self.beat = None
        self.sound = None

    def play_sound(self, *args):
        self.sound.stop()
        self.beat.stop()
        num = args[0]
        self.screen = num
        if num == -1:
            self.sound = SoundLoader.load("Audio/intro.mp3")
            if self.sound:
                self.sound.play()
            self.assemble_consciousness()
        if num == 0:
            self.sound = SoundLoader.load("Audio/breath.mp3")
            if self.sound:
                self.sound.play()
            self.assemble_breath()
        if num == 1:
            self.sound = SoundLoader.load("Audio/heartbeat.mp3")
            if self.sound:
                self.sound.play()
            self.assemble_heartbeat()

        if num == 2:
            self.sound = SoundLoader.load("Audio/cpr.mp3")
            if self.sound:
                self.sound.play()
            if self.beat:
```

```
            self.beat.play()
        self.assemble_cpr()

    if num == 3:
        self.sound = SoundLoader.load("Audio/other.mp3")
        if self.sound:
            self.sound.play()
        self.assemble_other()
    if num == 4:
        self.sound = SoundLoader.load("Audio/bleeding.mp3")
        if self.sound:
            self.sound.play()
        self.assemble_bleed()
    if num == 5:
        self.sound = SoundLoader.load("Audio/burn.mp3")
        if self.sound:
            self.sound.play()
        self.assemble_burn()
    if num == 6:
        self.sound = SoundLoader.load("Audio/injury.mp3")
        if self.sound:
            self.sound.play()
        self.assemble_injury()
    if num == 7:
        self.sound = SoundLoader.load("Audio/consious.mp3")
        if self.sound:
            self.sound.play()
        self.assemble_check()
    if num == 8:
        self.sound = SoundLoader.load("Audio/heart_attack.mp3")
        if self.sound:
            self.sound.play()
        self.assemble_heart()
    if num == 9:
        self.sound = SoundLoader.load("Audio/choking.mp3")
        if self.sound:
            self.sound.play()
        self.assemble_choking()
    if num == 10:
        self.sound = SoundLoader.load("Audio/anaphylaxis.mp3")
        if self.sound:
            self.sound.play()
        self.assemble_anaphylaxis()

def create_button(self, text, num):
    button = Button(text=f'[b]{text}[/b]', markup=True, color=(0.15, 0.15, 0.15, 1), font_size='20sp',
                    size_hint=(1, 0.5), background_color=(0.85, 0.85, 0.85, 1),
                    background_normal='', background_down='Colors/Red.jpg')
    button.bind(on_release=partial(self.play_sound, num))
    self.grid_layout.add_widget(button)
```

```
    def create_top(self, num, screen):
        if num == 0:
            box_layout = BoxLayout(orientation='horizontal', padding=dp(10), spacing=dp(10), size_hint=(1, 0.11))
            button_all = Button(text='[b]СПИСОК[/b]', markup=True, color=(0.15, 0.15, 0.15, 1), font_size='20sp',
                                size_hint=(1, 1), background_color=(0.85, 0.85, 0.85, 1),
                                background_normal='', background_down='Colors/Red.jpg')
            button_all.bind(on_release=partial(self.replace, 0))
            blank_widget = BoxLayout()
            button_repeat = Button(text='[b]ПОВТОР[/b]', markup=True, color=(0.15, 0.15, 0.15, 1),
font_size='20sp',
                                   size_hint=(1, 1), background_color=(0.85, 0.85, 0.85, 1),
                                   background_normal='', background_down='Colors/Red.jpg')
            button_repeat.bind(on_release=partial(self.play_sound, screen))
            box_layout.add_widget(button_all)
            box_layout.add_widget(blank_widget)
            box_layout.add_widget(button_repeat)
            self.root_widget.add_widget(box_layout)
        elif num == 1:
            box_layout = BoxLayout(orientation='horizontal', padding=dp(10), spacing=dp(10),
                                   size_hint_y=None, height=Window.height * 0.1)
            button = Button(text='[b]НАЗАД[/b]', markup=True, color=(0.15, 0.15, 0.15, 1), font_size='20sp',
                            size_hint=(1, 1), background_color=(0.85, 0.85, 0.85, 1),
                            background_normal='', background_down='Colors/Red.jpg')
            button.bind(on_release=partial(self.replace, 1))
            blank_widget1 = BoxLayout()
            blank_widget2 = BoxLayout()
            box_layout.add_widget(button)
            box_layout.add_widget(blank_widget1)
            box_layout.add_widget(blank_widget2)
            self.root_widget.add_widget(box_layout)

    def create_image(self, text, file):
        box_layout = BoxLayout(orientation='vertical', size_hint=(1, 1))
        image = Image(source=file, size_hint=(1, 0.8))
        label = Label(text=f'[b]{text}[/b]', markup=True, font_size='20sp',
                      size_hint=(1, 0.2), color=(0, 0, 0, 1))
        box_layout.add_widget(image)
        box_layout.add_widget(label)
        self.grid_layout.add_widget(box_layout)

    def assemble_all(self):
        self.root_widget.clear_widgets()
        self.create_top(1, 0)
        self.grid_layout = GridLayout(cols=2, padding=dp(10), spacing=dp(10), size_hint=(1, 1))
        self.grid_layout.bind(minimum_height=self.grid_layout.setter('height'))
        self.root_widget.add_widget(self.grid_layout)
        self.create_button('СЛР', 2)
        self.create_button('КРОВОТЕЧА', 4)
        self.create_button('ОПІК', 5)
```

```
        self.create_button('ТРАВМА', 6)
        self.create_button('СЕРЦЕВИЙ НАПАД', 8)
        self.create_button('УДУШЕННЯ', 9)
        self.create_button('АНАФІЛАКСІЯ', 10)
        self.create_button('ПОЧАТИ СПОЧАТКУ', -1)

    def assemble_consciousness(self):
        self.root_widget.clear_widgets()
        self.create_top(0, -1)
        self.grid_layout = GridLayout(cols=1, padding=dp(10), spacing=dp(10), size_hint=(1, 1))
        self.grid_layout.bind(minimum_height=self.grid_layout.setter('height'))
        self.root_widget.add_widget(self.grid_layout)
        self.create_image('ПЕРЕВІРТЕ ЧИ ПРИТОМНА ЛЮДИНА', 'Images/consciousness.jpg')
        self.create_button('ПРИТОМНА', 7)
        self.create_button('НЕПРИТОМНА', 0)

    def assemble_breath(self):
        self.root_widget.clear_widgets()
        self.create_top(0, 0)
        self.grid_layout = GridLayout(cols=1, padding=dp(10), spacing=dp(10), size_hint=(1, 1))
        self.grid_layout.bind(minimum_height=self.grid_layout.setter('height'))
        self.root_widget.add_widget(self.grid_layout)
        self.create_image('ПЕРЕВІРТЕ ДИХАННЯ ЛЮДИНИ', 'Images/breath.jpg')
        self.create_button('ДИХАЄ', 1)
        self.create_button('НЕ ДИХАЄ', 2)

    def assemble_heartbeat(self):
        self.root_widget.clear_widgets()
        self.create_top(0, 1)
        self.grid_layout = GridLayout(cols=1, padding=dp(10), spacing=dp(10), size_hint=(1, 1))
        self.grid_layout.bind(minimum_height=self.grid_layout.setter('height'))
        self.root_widget.add_widget(self.grid_layout)
        self.create_image('ПЕРЕВІРТЕ ПУЛЬС', 'Images/heartbeat.jpg')
        self.create_button('Є', 3)
        self.create_button('НЕМА', 2)

    def assemble_other(self):
        self.root_widget.clear_widgets()
        self.create_top(0, 3)
        self.grid_layout = GridLayout(cols=2, padding=dp(10), spacing=dp(10), size_hint=(1, 1))
        self.grid_layout.bind(minimum_height=self.grid_layout.setter('height'))
        self.root_widget.add_widget(self.grid_layout)
        self.create_image('ЗОВН. КРОВОТЕЧІ', 'Images/ext_bleed.jpg')
        self.create_image('ВНУТР. КРОВОТЕЧІ', 'Images/int_bleed.jpg')
        self.create_image('ОПІКИ', 'Images/burns.jpg')
        self.create_image('ТРАВМИ', 'Images/injury.jpg')
        self.grid_layout = GridLayout(cols=1, padding=dp(10), spacing=dp(10), size_hint=(1, 1))
        self.grid_layout.bind(minimum_height=self.grid_layout.setter('height'))
        self.root_widget.add_widget(self.grid_layout)
        self.create_button('КРОВОТЕЧА', 4)
```

```
        self.create_button('ОПІК', 5)
        self.create_button('ТРАВМА', 6)

    def assemble_cpr(self):
        self.root_widget.clear_widgets()
        self.create_top(0, 2)
        self.grid_layout = GridLayout(cols=1, padding=dp(10), spacing=dp(10), size_hint=(1, 1))
        self.grid_layout.bind(minimum_height=self.grid_layout.setter('height'))
        self.root_widget.add_widget(self.grid_layout)
        self.create_image('30 КОМПРЕСІЙ', 'Images/cpr1.jpg')
        self.create_image('2 ВДИХИ', 'Images/cpr2.jpg')

    def assemble_bleed(self):
        self.root_widget.clear_widgets()
        self.create_top(0, 4)
        self.grid_layout = GridLayout(cols=1, padding=dp(10), spacing=dp(10), size_hint=(1, 1))
        self.grid_layout.bind(minimum_height=self.grid_layout.setter('height'))
        self.root_widget.add_widget(self.grid_layout)
        self.create_image('ЗАТИСНІТЬ РАНУ', 'Images/bleed_fa.jpg')
        self.create_image('НАКЛАДІТЬ ТУРНІКЕТ', 'Images/tourniquet.jpg')

    def assemble_burn(self):
        self.root_widget.clear_widgets()
        self.create_top(0, 5)
        self.grid_layout = GridLayout(cols=1, padding=dp(10), spacing=dp(10), size_hint=(1, 1))
        self.grid_layout.bind(minimum_height=self.grid_layout.setter('height'))
        self.root_widget.add_widget(self.grid_layout)
        self.create_image('ОХОЛОДІТЬ ВОДОЮ', 'Images/burns_fa.jpg')
        self.create_image('НАКЛАДІТЬ ПОВ\'ЯЗКУ', 'Images/bandage.jpg')

    def assemble_injury(self):
        self.root_widget.clear_widgets()
        self.create_top(0, 6)
        self.grid_layout = GridLayout(cols=1, padding=dp(10), spacing=dp(10), size_hint=(1, 1))
        self.grid_layout.bind(minimum_height=self.grid_layout.setter('height'))
        self.root_widget.add_widget(self.grid_layout)
        self.create_image('ЗАФІКСУЙТЕ ДІЛЯНКУ', 'Images/injury_fix.jpg')
        self.create_image('ОХОЛОДЖУЙТЕ', 'Images/cooling.jpg')

    def assemble_check(self):
        self.root_widget.clear_widgets()
        self.create_top(0, 7)
        self.grid_layout = GridLayout(cols=2, padding=dp(10), spacing=dp(10), size_hint=(1, 1))
        self.grid_layout.bind(minimum_height=self.grid_layout.setter('height'))
        self.root_widget.add_widget(self.grid_layout)
        self.create_image('СЕРЦЕВИЙ НАПАД', 'Images/heart_attack.jpg')
        self.create_image('УДУШЕННЯ', 'Images/choking.jpg')
        self.create_image('АНАФІЛАКСІЯ', 'Images/anaphylaxis.jpg')
        self.create_image('ІНШЕ', 'Images/other.jpg')
        self.create_button('СЕРЦЕВИЙ НАПАД', 8)
```

```
        self.create_button('УДУШЕННЯ', 9)
        self.create_button('АНАФІЛАКСІЯ', 10)
        self.create_button('ІНШЕ', 3)

    def assemble_heart(self):
        self.root_widget.clear_widgets()
        self.create_top(0, 8)
        self.grid_layout = GridLayout(cols=1, padding=dp(10), spacing=dp(10), size_hint=(1, 1))
        self.grid_layout.bind(minimum_height=self.grid_layout.setter('height'))
        self.root_widget.add_widget(self.grid_layout)
        self.create_image('ЗРУЧНА ПОЗИЦІЯ', 'Images/heart_attack_care.jpg')
        self.create_image('АСПІРИН', 'Images/aspirin.jpg')
        self.create_button('ЗУПИНКА СЕРЦЯ', 2)

    def assemble_choking(self):
        self.root_widget.clear_widgets()
        self.create_top(0, 9)
        self.grid_layout = GridLayout(cols=1, padding=dp(10), spacing=dp(10), size_hint=(1, 1))
        self.grid_layout.bind(minimum_height=self.grid_layout.setter('height'))
        self.root_widget.add_widget(self.grid_layout)
        self.create_image('5 УДАРІВ ПО СПИНІ', 'Images/choking_back.jpg')
        self.create_image('5 СТИСКАНЬ ЖИВОТА', 'Images/heimlih.jpg')

    def assemble_anaphylaxis(self):
        self.root_widget.clear_widgets()
        self.create_top(0, 10)
        self.grid_layout = GridLayout(cols=1, padding=dp(10), spacing=dp(10), size_hint=(1, 1))
        self.grid_layout.bind(minimum_height=self.grid_layout.setter('height'))
        self.root_widget.add_widget(self.grid_layout)
        self.create_image('ЕПІНЕФРИН', 'Images/epinephrine.jpg')
        self.create_image('ІН\'ЄКЦІЯ', 'Images/injection.jpg')
        self.create_button('ЗУПИНКА ДИХАННЯ', 2)

    def build(self):

        self.sound = SoundLoader.load("Audio/intro.mp3")
        if self.sound:
            self.sound.play()

        self.beat = SoundLoader.load('Audio/cpr_beat.mp3')
        self.screen = -1

        Window.clearcolor = (1, 1, 1, 1)

        self.root_widget = BoxLayout(orientation='vertical')

        self.assemble_consciousness()

        return self.root_widget
```

```
    def replace(self, *args):
        self.sound.stop()
        self.beat.stop()
        num = args[0]
        if num == 0:
            self.assemble_all()
        elif num == 1:
            self.play_sound(self.screen)


if __name__ == '__main__':
    AidAssistant().run()
```