МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютеризованих систем управління

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри

Литвиненко О.Є.
"__"_____2022 р.

ДИПЛОМНИЙ ПРОЄКТ

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ЗДОБУВАЧА ОСВІТНЬОГО СТУПЕНЯ
"БАКАЛАВР"

Тема: "Програмний модуль голосового помічника для авіапасажира"_____

Виконавець: Сміян Богдан Миколайович_____

Керівник: д.т.н., доцент, Нечипорук Олена Петрівна_____

Консультанти з окремих розділів пояснювальної записки:

_____

_____

Нормоконтролер: Тупота Є. В._____

Київ 2022

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

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

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

ЗАТВЕРДЖУЮ
Завідувач кафедри
Литвиненко О. Є.
«__»_____2022 р.

ЗАВДАННЯ
на виконання дипломної роботи (проєкту)
Сміяна Богдана Миколайовича
(прізвище, ім'я, по батькові випускника в родовому відмінку)

1. Тема дипломної роботи (проєкту): «Програмний модуль голосового помічника для авіапасажира»

затверджена наказом ректора від «04» квітня 2022 р. № 340

2. Термін виконання роботи (проєкту): з 09.05.2022 р. по 06.06.2022 р.

3. Вихідні дані до роботи (проєкту): програмний модуль голосового помічника для авіапасажира

4. Зміст пояснювальної записки:

1. Аналіз існуючих аналогів та актуальності їх розробки.
2. Специфікація вимог до програмного модулю голосового помічника.
3. Створення прототипу програмного модулю.
4. Тестування прототипу програмного модулю.

5. Перелік обов'язкового графічного (ілюстративного) матеріалу:

1. Діаграма прецедентів функцій застосунку.
2. Блоки функцій модулю.
3. Діаграма послідовностей.
4. Інтерфейс застосунку.

Календарний план-графік

№ пор.	Завдання	Термін виконання	Відмітка про виконання
1.	Складання та затвердження графіку роботи дипломного проектування Написання 1 розділу, представлення керівнику	09.05.22 – 15.05.22	
2.	Попередній друк 1 розділу та допоміжних сторінок (чернетка) - титульної, завдання, графіка, реферат, список скорочень, зміст, вступ, список джерел, 1-й нормо-контроль.	16.05.22 – 17.05.22	
3.	Написання 2 розділу, представлення керівнику	18.05.22 – 22.05.22	
4.	Написання 3 розділу, представлення керівнику	23.05.22 – 28.05.22	
5.	Написання 4 розділу, представлення керівнику	29.05.22 – 01.06.22	
6.	Загальне редагування та друк пояснювальної записки, графічного матеріалу	02.06.22 – 03.06.22	
7.	Проходження нормо-контролю, перевірка на антиплагіат, перепліт пояснювальної записки.	03.06.22 – 04.06.22	
8.	Розробка тексту доповіді. Оформлення графічного матеріалу для презентації	04.06.22 – 07.06.22	
9.	Отримання відгуку керівника, рецензії.	07.06.22 – 12.06.22	
10.	Підготовка матеріалів для передачі секретарю ДЕК (ПЗ, CD-R з електронними копіями ПЗ, презентації, відгук керівника, рецензія) в папці	18.06.22 – 20.06.22	

7. Консультанти з окремих розділів

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

8. Дата видачі завдання: “16" квітня 2022 р.

Керівник дипломної роботи (проєкту) _____ Нечипорук Олена Петрівна
(підпис керівника) (П.І.Б.)

Завдання прийняв до виконання _____ Сміян Богдан Миколайович
(підпис випускника) (П.І.Б.)

РЕФЕРАТ

Пояснювальна записка до дипломної роботи «програмний модуль голосового помічника для авіапасажира»: 42 с., 24 рис., 8 табл., 8 інформаційних джерел.

ПРОГРАМНИЙ МОДУЛЬ, ГОЛОСОВИЙ ПОМІЧНИК, АВІАПАСАЖИР

Об’єкт розробки – програмний модуль голосового помічника для авіапасажира.

Мета роботи – створення програмного модулю голосового помічника для авіапасажира.

ABSTRACT

Explanatory note to the thesis "software module of voice assistant for air passengers": 42 pages, 24 figures, 8 tables, 8 information sources.

SOFTWARE MODULE, VOICE ASSISTANT, AIR PASSENGER

The object of development is the software module of the voice assistant for the air passenger.

The purpose of the work is to create a software module of a voice assistant for an air passenger.

ЗМІСТ

ПЕРЕЛІК ВИКОРИСТАНИХ СКОРОЧЕНЬ

ПЗ	Програмне забезпечення
IT	Information Technologies
UML	Unified Modeling Language
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
VA	Voice Assistant
AI	Artificial Intelligence

ВСТУП

Велика кількість користувачів не бажають марно витрачати час, а навпаки й з користю використовувати його, користуючись послугами авіакомпаній. У зв'язку з чим питання використання програмного модулю голосового помічника для авіапасажирів є актуальним на даний момент. Використовуючи голосового помічника можна лише за допомогою мови використовувати функції комп'ютера, ця проблема стає ще більш актуальною, якщо мова йде про людей у котрих наявні вади здоров'я й вони не можуть повноцінно використовувати комп'ютерні технології.

Використання мовного озвучення тексту чи елементів на екрані може допомогти таким людям почуватись комфортно та без проблем використовувати технологічні можливості. Користувачами можуть виступати люди різного віку. Це можуть бути й студенти, й батьки з дітьми, й пенсіонери, й люди середнього віку.

Кожен з перерахованих вище користувачів має спільну проблему – ефективне використання технологій, навіть маючи проблеми зі здоров'ям – зором, слухом чи інше. На це й направлений програмний модуль голосового помічника для авіапасажирів, що буде створюватись. Завдяки використанню передових технологій 21 століття, сучасних мов програмування, специфікації та проектуванню програмних засобів стає можливим здійснення процесу використання комп'ютерних засобів навіть маючи ряд певних проблем зі здоров'ям.

Тепер для задоволення потреби необхідно лише запустити програмний засіб, авторизуватися й почати діалог з помічником, все інше виконає він.

Мета дипломної роботи — створення програмного модулю голосового помічника для авіапасажирів.

Для досягнення мети визначено наступні задачі:

- аналіз та опис застосунку;
- визначення необхідних елементів і блоків у майбутньому модулі;
- вибір та опис використовуваних технологій;
- створення зручного і практичного інтерфейсу;
- розробка робочого прототипу.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ПРОГРАМНОГО МОДУЛЮ

1.1. Дослідження голосових помічників та їх порівняння

Зараз на ринку існує безліч голосових помічників, які здатні спростити життя людині. Тепер не треба вручну вбивати у пошуковик питання або шукати якусь пісню, достатньо лише сказати, що ти хочеш, а голосовий помічник знайде все сам. Але з кожним роком помічників стає все більше і звичайному користувачеві стає все важче вибирати, адже кожен асистент має свої особливості [1].

Найпопулярнішими голосовими помічниками у світі є Google Assistant, Apple Siri, Microsoft Cortana, а в Росії особливо популярний голосовий помічник від Яндекс Аліса [8]

Google assistant

Google assistant – є найпопулярнішим голосовим помічником у світі, що працює на пристроях Android, iOS та в браузері Chrome. З плюсів можна назвати швидкість, точність у побудові маршруту, відстеження користувача, взаємодію з управлінням нотаток, повідомленнями, відтворенням музики, великими даними.

Має також два недоліки у вигляді зайвої ініціативності (показує зайві пропозиції), і на відміну від Apple Siri та Аліси не здатний розмовляти на різні теми як людина.

Може взаємодіяти з більшістю програм та може встановлюватися практично на будь-який пристрій (див. рис. 1.1).

Рис. 1.1. Ok Google

Alexa

Amazon Alexa - програма вбудована в аудіопристрої Amazon (Echo, Echo Dot, Tap) та приставки Fire TV. У пристрій вбудовані функції на кшталт замовлень товарів та повідомлень про погоду або новини, включення музики, але він може працювати лише вдома, не може працювати з телефонами та не вміє підтримувати спілкування (див. рис. 1.2.).

Рис. 1.2. Alexa

Аліса

Аліса - голосовий помічник розроблений компанією Яндекс для російськомовної людини. За функціоналом практично не поступається Siri. Має непоганий пошук. Чудово розуміє навіть нечітку російську мову. З мінусів можна відзначити не сумісність із багатьма програмами.

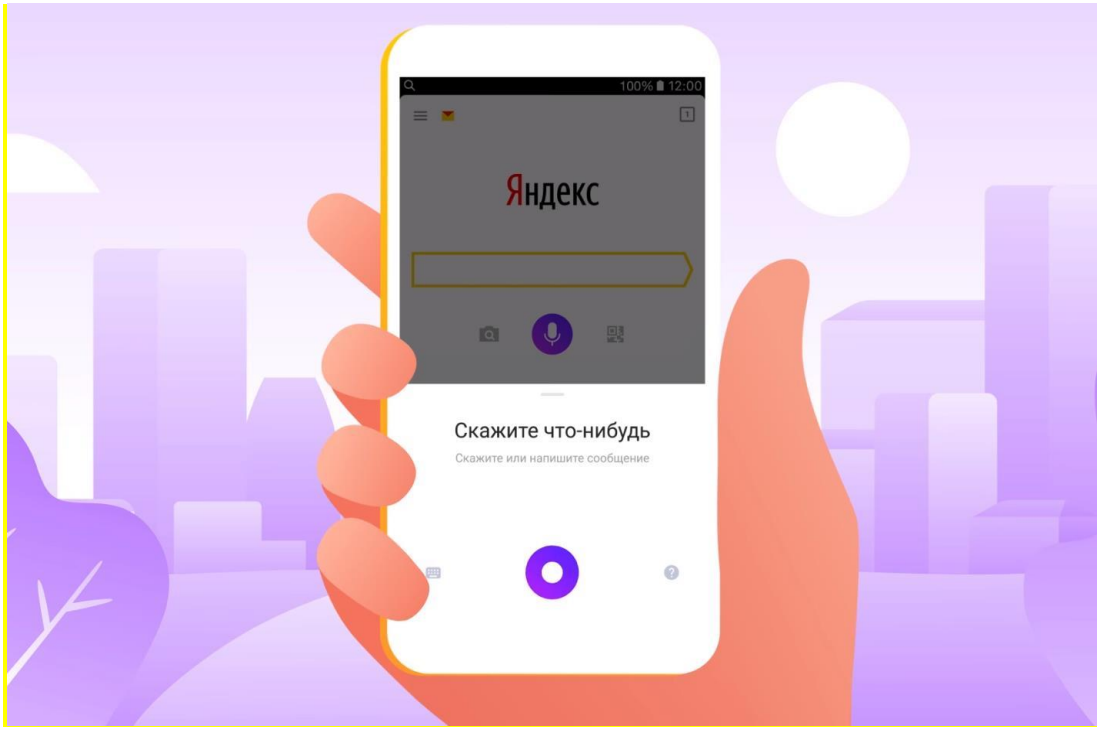

Рис. 1.3. Аліса

Apple Siri

Siri – це перший асистент з вмінням спілкуватись з користувачем та давати йому різні рекомендації (див. рис. 1.4.). Наприклад, відправляти користувачеві лише ті новини, що його цікавлять, а також надавати рекомендації виходячи з його вподобань – на новини, фільми, додатки чи інше.

У зв’язку з тим, що основною операційною системою для роботи помічника було обрано iOS, він є дуже добре оптимізованим. Розуміє й знаходить інформацію краще, ніж аналоги. Проте має декілька недоліків, серед яких: неповноцінність бесіди, невміння взаємодіяти з більшістю додатків, а також доступ лише для пристроїв з операційною системою iOS. Також, в порівнянні з іншими погане розуміння російської чи української мови.

Рис. 1.4. *Apple Siri*

Microsoft Cortana

Кортана – це віртуальний асистент, що доступний для Windows, iOS та Андроїд, може добре спілкуватись з людиною та навіть жартувати. Помічник створений для задоволення потреб користувача та для видачі тієї інформації що буде цікава йому на даний момент часу (див. рис. 1.5.). Інтерфейс помічника є доволі гнучким, що дозволяє користувачеві обрати налаштування та виділити необхідні йому функції чи можливості. Також наявна функція озвучення елементів, що знаходяться на екрані. Така функція дуже корисна для людей з проблемами зору, що виділяє цей помічник серед інших. Проте можуть виникати проблеми на основі поганої оптимізації на телефонах, користуватись на яких стає складніше, а функцій менше. Також помічник не має підтримки української мови, що робить його майже непотрібним, якщо не володіти англійською.

Рис. 1.5. *Cortana*

Питання який саме голосовий помічник ви використовуватимете – залежить виключно від самого користувача. Благо, на сучасному ринку цифрових гаджетів сьогодні можна підібрати помічника собі за смаком та фінансовими можливостями. Єдине обмеження – відсутність локалізації та локальна представленість деяких гаджетів.

Порівнюючи дані аналоги, Гугл, в порівнянні від Сірі та Алекси оптимізований на більшості приладів та ефективний у швидкодії, завдяки чому являється найбільш популярним асистентом у світі. Також в ньому багато реклами, яка в порівнянні від Сірі не підбирається індивідуально, в залежності від користувача.

Сірі працює тільки на iOS, не може використовуватись з іншими додатками, через що в порівнянні з Гугл, асистент не використовується доволі часто та не працює з системами, проте наявний потенціал розуміння людської мови.

Алекса може використовуватись тільки в будинку, так як взаємодія відбувається через прилади Амазону, проте, в порівнянні з аналогами, підтримує

роботу з більшістю додатків, тому буде корисною під час взаємодії у системи. Також не підтримує українську та не користується популярністю в СНГ.

Кортана має чудову систему повідомлень та оптимізацію, проте проблеми виникають під час використання на телефонах. Люди з вадами можуть використовувати цей асистент як слуховий помічник, котрий допоможе вирішити певні питання, проте в порівнянні з іншими не може взаємодіяти з приладами, що не дозволяє використовувати її в системах. Також не має підтримки української мови.

Аліса має підтримку російської мови, тому популярна в країнах СНГ, проте не має підтримки української. Крім цього має оптимізацію для пк та андроїд, але проблеми можуть виникати під час взаємодії з іншими додатками, проте розробники активно розвивають та активно видаляють помилки в роботі.

Після детального огляду аналогів, було складено таблицю (Таблиця 1.1. Оцінка існуючих аналогів по зазначеним характеристикам) з необхідними характеристиками рішення проблеми та проведено оцінку кожного застосунку по визначеним.

Виділено головні характеристики, які розглядались для аналізу – загальна оцінка аналогу на основі відгуків користувачів, особливість даного аналогу – що виділяє цей додаток від інших, наявність підтримки української мови, з якими проблемами можна зіштовхнутись під час використання та наявність реклами під час використання.

Таблиця 1.1.

Оцінка існуючих аналогів по зазначеним характеристикам

Помічник	Оцінка	Особливість	Підтримка української	Проблеми	Реклама
Гугл	4\5	Швидкість роботи	Наявне	Нав’язливість	Наявна
Сірі	4\5	Розуміння мови	Відсутнє	Інтеграція з іншими додатками	Відсутня
Алекса	4\5	Голосова озвучка	Відсутнє	Обмежене використання	Наявна
Кортана	4\5	Нагадування	Відсутнє	Погана оптимізація	Відсутня
Аліса	4\5	Розуміння мови	Відсутнє	Несумісність з додатками	Наявна

За проведеним дослідженням можна зробити висновки, що ні один продукт не задовольняє встановлених вимог і не є досконалим рішенням проблеми. Варто відмітити недоліки даних систем, а саме неправильне розпізнавання мови, не завжди коректний пошук команд та інші помилки різного роду. А отже питання створення голосового помічника є відкритим для вирішення.

1.2. Дослідження проблематики роботи

Об’єктом проблематики виступають персональні комп’ютери, що використовуються під час перельоту. Велика кількість користувачів не бажають марно витрачати час, а навпаки й з користю використовувати його, користуючись послугами авіакомпаній. У зв’язку з чим питання використання програмного модулю голосового помічника для авіапасажирів є актуальним на даний момент. Використовуючи голосового помічника можна лише за допомогою мови використовувати функції комп’ютера, ця проблема стає ще більш актуальною, якщо мова йде про людей у котрих наявні вади здоров’я й вони не можуть повноцінно використовувати комп’ютерні технології [2].

Використання мовного озвучення тексту чи елементів на екрані може допомогти таким людям почуватись комфортно та без проблем використовувати технологічні можливості [3].

Користувачами можуть виступати люди різного віку. Це можуть бути й студенти, й батьки з дітьми, й пенсіонери, й люди середнього віку.

Кожен з перерахованих вище користувачів має спільну проблему – ефективне використання технологій, навіть маючи проблеми зі здоров’ям – зором, слухом чи інше. На це й направлений програмний модуль голосового помічника для авіапасажирів, що буде створюватись. Завдяки використанню передових технологій 21 століття, сучасних мов програмування, специфікації та проектуванню програмних засобів стає можливим здійснення процесу використання комп’ютерних засобів навіть маючи ряд певних проблем зі здоров’ям.

Тепер для задоволення потреби необхідно лише запустити програмний засіб, авторизуватися й почати діалог з помічником, все інше виконає він.

1.3. Постановка задачі

Мета дипломної роботи — створення програмного модулю голосового помічника для авіапасажирів [4].

Для досягнення мети визначено наступні задачі:

- аналіз та опис програмного модулю;

- визначення необхідних елементів і блоків у майбутньому модулі;
- вибір та опис використовуваних технологій;
- створення зручного і практичного інтерфейсу;
- розробка робочого прототипу.

1.4. Висновки до розділу

В ході виконання першого розділу роботи було проведено аналіз предметної області програмного модулю голосового помічника для авіапасажиру.

В першому підпункті було досліджено історію створення голосового помічника – від початку зародження даної теми до нашого часу, проаналізовано основні моменти та досягнення в наш час.

В другому підпункті було проведене дослідження актуальних програмних аналогів голосового помічника, що використовуються у нас час. Було розглянуто такі помічники як Гугл, Сірі, Аліса та інші. Проаналізовано їх основні можливості, а також плюси й мінуси, складено таблицю аналізу даних помічників.

В третьому підпункті було проведено дослідження проблематики роботи – розглянуто актуальність та проблему, що вирішує програмний модуль голосового помічника.

РОЗДІЛ 2
СПЕЦИФІКАЦІЯ ВИМОГ ПРОГРАМНОГО МОДУЛЮ ГОЛОСОВОГО ПОМІЧНИКА

2.1. Загальний опис продукту

Завданням цього програмного модулю є полегшення використання комп'ютеру чи ноутбуку, що використовується під час здійснення авіа перельоту. Користувачами цього програмного модулю можуть бути люди різного віку, статі та знань у використанні комп'ютерних технологій. Завдяки використанню модулю голосового помічника користувач матиме можливість використовувати функції комп'ютеру за допомогою використання власного голосу, що дозволить упросити ти роботу з технікою та керувати нею навіть маючи вади зі здоров'ям [5]. За допомогою використанню голосового помічника можливі наступні дії:

- здійснювати пошук в мережі інтернет по запиту;
- відкривати зовнішні додатки;
- виводити список доступних команд помічника;
- переводити слова з однієї мови на іншу;
- конвертація тексту в мову та навпаки.

Іншими словами, будь-яка дія що доступна на комп'ютері може виконуватись за допомогою голосового помічника, якщо йому надати для цього доступ й можливості.

У першому розділі було проаналізовано аналоги, наразі доступна велика кількість голосових асистентів та помічників, частина з них є доволі сильними конкурентами на ринку, адже користувачі користуються ними вже довгий час. Проте при наявному плані та проекту підтримки та розвитку голосового модулю можливе створення конкурентного додатку на основі цього модулю.

Оскільки аудиторія користувачів є доволі різною, то необхідно забезпечити задоволення їх вимог до програмного забезпечення. Тому саме на їх потреби необхідно орієнтуватись під час створення програмного модулю та визначенню вимог.

Для оптимальної роботи програмного модулю необхідно буде забезпечити встановлення програмних модулів, що забезпечать роботу програмного модулю голосового помічника, а також доступ до мережі інтернет.

2.2. Формулювання вимог до програмного модулю голосового помічника

Формулювання вимог до програмного модулю голосового помічника буде залежати від користувачів, що будуть використовувати цей модуль. Так як основними користувачами є авіапасажири, то необхідно визначити їх потреби для використання програмного модулю голосового помічника. Під час здійснення перельоту авіапасажирі матимуть змогу використовувати персональні комп’ютери, саме для цієї категорії користувачів й будуть формуватись вимоги [6]. Перш за все розглянемо що саме можуть виконувати за комп’ютером користувачі. Варто поділити користувачів на декілька категорій, до першої категорії віднесемо людей, котрі виконують комп’ютери для роботи – задля забезпечення їх вимог до використання необхідно забезпечити використання голосового помічника для ведення роботи, сюди можна віднести планування дня, створення записок та розпорядку, відкриття програм для роботи, це можуть бути текстові редактори чи редактори коду, або інші програми, функції яких дозволяють виконувати певну роботу – створення дизайну чи відео, редагування фото чи медіа даних. Наступною категорію є користувачі, що будуть використовувати комп’ютери для персонального використання – тобто, частіше за все, даний тип користувачів захоче відпочити й

займатись в перельоті своїми справами, оскільки переліт може займати різний час, то для забезпечення виконання їх вимог необхідно надати можливості користувачеві обирати музику чи перегляд фільму, а також запуск ігор, що дозволить також зайняти себе під час перельоту. Також сюди можна віднести використання месенджерів чи соціальних мереж та браузеру, якщо на борту буде наявний доступ до інтернет мережі. Ще однією категорію користувачів є люди з обмеженими можливостями – наприклад, з вадами зору, у випадку вад слуху використання голосового модулю не принесе практичної користі, тому дану категорію користувачів не слід розглядати. Для забезпечення вимог цього типу користувачів необхідно забезпечити можливість озвучення кожної дії програмного модулю голосового помічника, що дозволить людині орієнтуватись під час використанню комп'ютеру та виконувати деякі доступні функції [7].

Отже, як можемо бачити, наявні різні категорії користувачів що можуть використовувати програмний модуль, проте вимоги у більшості з них є схожими та якщо їх підсумувати, то отримаємо наступні користувацькі вимоги до створення голосового помічника:

- дизайн голосового помічника має бути простим та мінімалістичним;
- інтерфейс має бути нескладним й простим у використанні;
- повинні бути відсутні складні елементи та форми;
- інтерфейс має бути зрозумілим;
- забезпечення конфіденційності;
- адаптивність дизайну;
- оптимальна швидкість відгуку на запит користувача;
- масштабованість – можливість розширювати функції програмного модулю.

Для забезпечення виконання функціональних вимог поділимо їх відповідно до категорій користувачів. Для категорії «Для роботи» необхідно забезпечити можливість використання голосового помічника для роботи, тому буде необхідно:

- виконувати запуск програм для роботи;
- виконувати запуск браузеру;

- здійснення пошуку в мережі інтернет;
- створення заміток;
- створення розкладу.

Для категорії «Для розваг» необхідно забезпечити можливість використання голосового помічника для розваг, тому буде необхідно:

- виконувати запуск плейлисту музики;
- виконувати запуск Youtube;
- виконувати запуск відео чи кінофільму;
- виконувати запуск ігор;
- виконувати запуск соцмереж чи месенджерів.

Для категорії «З вадами здоров’я» необхідно забезпечити можливість озвучення кожної дії та отриманих результатів виконання голосового помічника. Тому було прийнято рішення для кожної категорії створити окремий блок функцій, що зображено на рисунку 2.1.

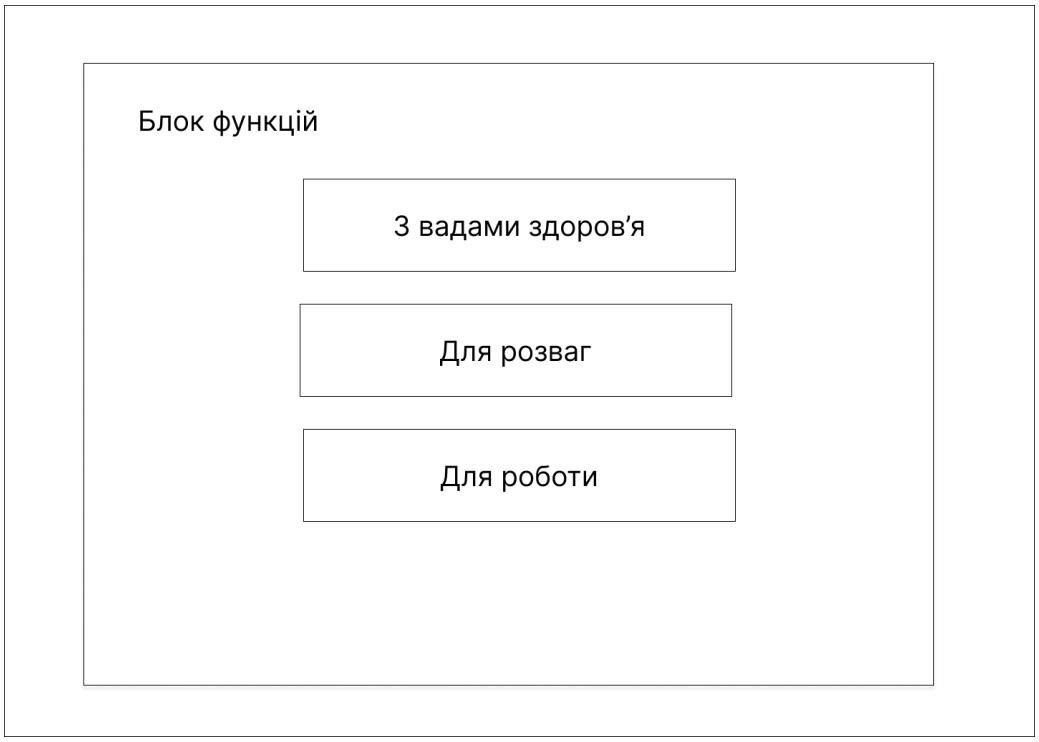

Рис. 2.1. Блоки функцій

Згідно до наведеної схеми, функції застосунку буде поділено на три частини – для користувачів, що використовують голосовий помічник для розваг, для роботи та окремий пункт для користувачів з вадами здоров’я.

Для відображення функцій програмного модулю голосового помічника використаємо діаграму прецедентів – що буде відображати відношення між акторами (в нашому випадку користувачами) та прецедентами – функціями. Діаграму прецедентів зображено на рисунку 2.2.

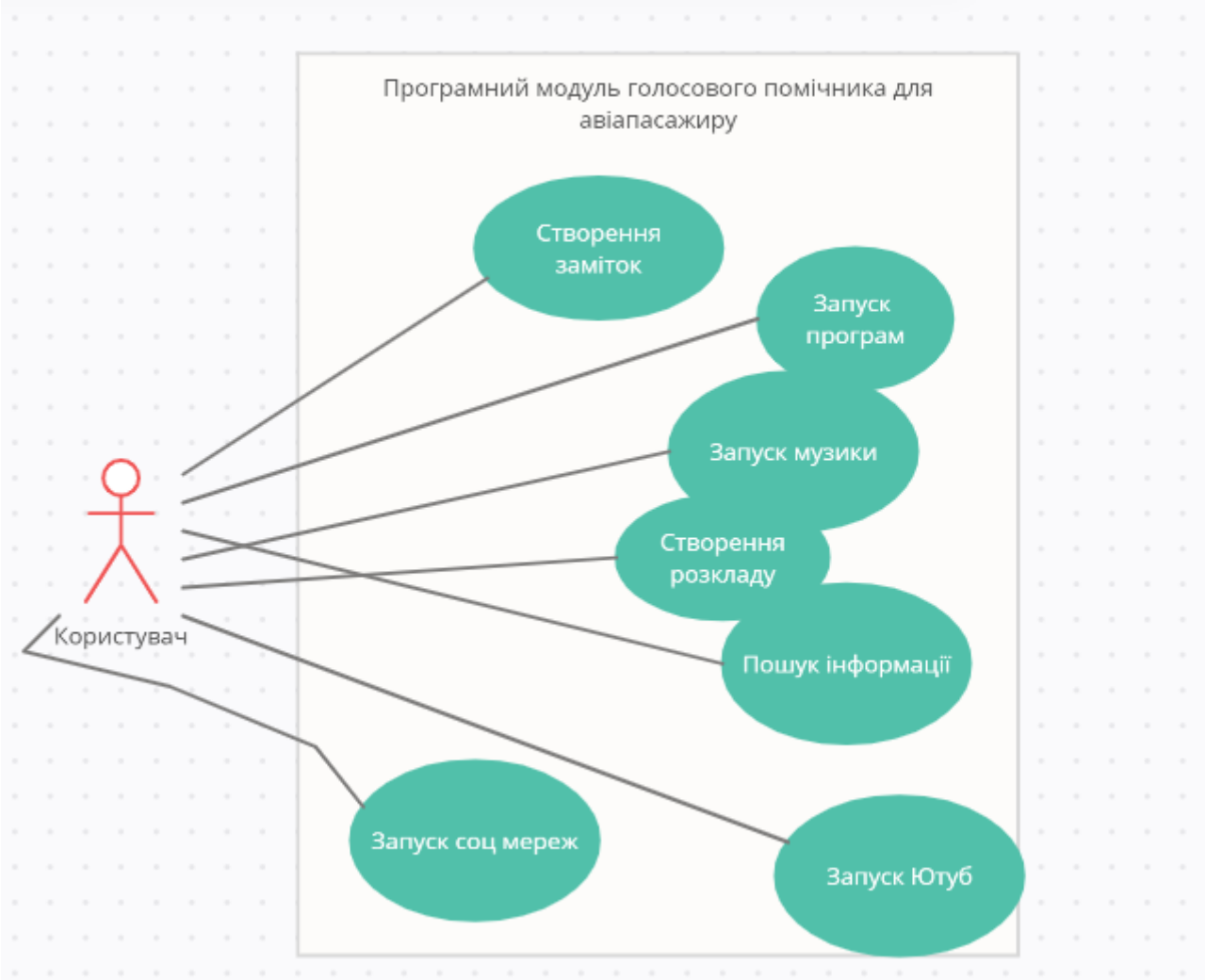

Рис. 2.2. Діаграма прецедентів

Для більш детального опису функцій програмного модулю голосового помічника виконаємо специфікацію прецедентів. В результаті чого будуть створені

сценарії взаємодії користувача та системи. Для цього оберемо кожну з функцій та створимо до неї таблицю, в котрій зазначимо сценарій використання функції.

Use Case прецеденту «Створення заміток» представлено в таблиці 2.1.

Таблиця 2.1.

Use Case створення заміток

Діючі лиця	Користувач, Система
Цілі	Створити замітки
Передумова	Немає
Успішний сценарій: 1. Система очікує на отримання команди від користувача. 2. Користувач проговорює ключову фразу. 3. Система отримує команду, ідентифікує її. 4. Користувач створює замітку.	
Результат	Користувач створює замітку

Use Case прецеденту «Запуск програм» представлено в таблиці 2.2.

Таблиця 2.2.

Use Case запуск програм

Діючі лиця	Користувач, Система
Цілі	Запустити програму
Передумова	Немає
Успішний сценарій: 1. Система очікує на отримання команди від користувача. 2. Користувач проговорює ключову фразу з назвою програми. 3. Система отримує команду, ідентифікує її. 4. Система виконує запуск програми.	
Результат	Система виконує запуск програми

Use Case прецеденту «Запуск музики» представлено в таблиці 2.3.

Таблиця 2.3.

Use Case запуск музики

Діючі лиця	Користувач, Система
Цілі	Запустити музику
Передумова	Немає
Успішний сценарій: 1. Система очікує на отримання команди від користувача. 2. Користувач проговорює ключову фразу з назвою музики. 3. Система отримує команду, ідентифікує її. 4. Система виконує запуск музики.	
Результат	Система виконує запуск музики

Use Case прецеденту «Створення розкладу» представлено в таблиці 2.4.

Таблиця 2.4.

Use Case створення розкладу

Діючі лиця	Користувач, Система
Цілі	Створити замітки
Передумова	Немає
Успішний сценарій: 1. Система очікує на отримання команди від користувача. 2. Користувач проговорює ключову фразу. 3. Система отримує команду, ідентифікує її. 4. Система створює розклад.	
Результат	Система створює розклад

Use Case прецеденту «Запуск Youtube» представлено в таблиці 2.5.

Таблиця 2.5.

Use Case запуск Youtube

Діючі лиця	Користувач, Система
Цілі	Запустити youtube
Передумова	Немає
Успішний сценарій: 1. Система очікує на отримання команди від користувача. 2. Користувач проговорює ключову фразу Youtube. 3. Система отримує команду, ідентифікує її. 4. Система виконує запуск Youtube.	
Результат	Система виконує запуск Youtube

Use Case прецеденту «Запуск соцмереж» представлено в таблиці 2.6.

Таблиця 2.6.

Use Case запуск соцмереж

Діючі лиця	Користувач, Система
Цілі	Запустити соціальну мережу
Передумова	Немає
Успішний сценарій: 1. Система очікує на отримання команди від користувача. 2. Користувач проговорює ключову фразу з назвою соціальної мережі. 3. Система отримує команду, ідентифікує її. 4. Система виконує запуск соціальної мережі.	
Результат	Система виконує запуск соціальної мережі

Use Case прецеденту «Пошук інформації» представлено в таблиці 2.7.

Таблиця 2.7.

Use Case пошук інформації

Діючі лиця	Користувач, Система
Цілі	Знайти інформацію
Передумова	Немає
Успішний сценарій: 1. Система очікує на отримання команди від користувача. 2. Користувач проговорює ключову фразу «Пошук». 3. Система отримує команду, ідентифікує її. 4. Система виконує запуск пошуку в інтернеті.	
Результат	Система виконує запуск пошуку в інтернеті

Представимо візуально роботу програмного модулю голосового помічника. Для цього створимо UML-діаграму, що буде відображати потоки діяльності програмного застосунку. Потоки діяльності програмного модулю представлені на рисунку 2.3.

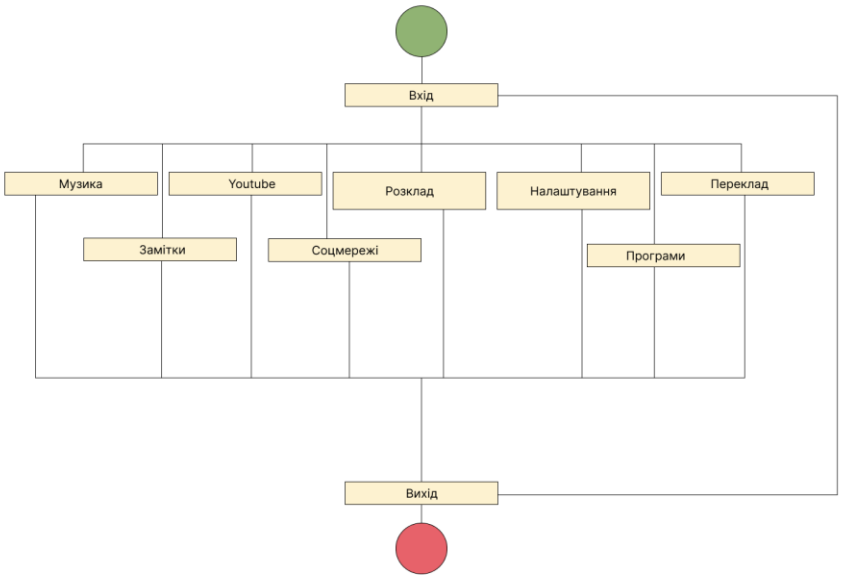

Рис. 2.3. Потоки діяльності програмного модулю

Для того щоб відобразити формування запитів в режимі реального часу буде створена діаграма послідовності. Діаграма послідовності представлена на рисунку 2.4.

Рис. 2.4. Діаграма послідовності

2.3. Висновки до розділу

У даному розділі було виконано загальний опис застосунку – було описано загальну інформацію програмного модулю голосового помічника, що буде створюватись, описано його функції та характеристики, а також основних користувачів.

Також було створено ряд вимог до застосунку, сюди можна віднести наступні вимоги: функціональні – що забезпечать повноцінну функціональність застосунку, відповідно до вимог користувача, не функціональні – що не залежать від функцій застосунку, проте відповідають вимогам користувачів, системні вимоги – вимоги до апаратної частини для запуску застосунку.

Також було створено логічне представлення блоків застосунку – у вигляді блоку функцій – сюди відносимо блок для роботи, для розваг та з вадами здоров’я.

Було визначено функції застосунку – сюди можна віднести здійснення пошуку в мережі інтернет, запуск різного роду застосунків, запуск музики та Youtube, створення заміток та розкладу.

До кожної функції було створено Use Case для більш детального опису та створення сценарію взаємодії користувача з програмним модулем голосового помічника. Крім цього були створені діаграми прецедентів та послідовності.

РОЗДІЛ 3.
СТВОРЕННЯ ПРОТОТИПУ ПРОГРАМНОГО МОДУЛЮ

3.1. Вибір середовища розробки

PyCharm – це середовище розробки програмного коду для мови програмування Пайтон. Доступна у вигляді двох варіантів є версія для користувачів, що є безкоштовною, та є професійна версія, яку потрібно купити. В даному середовищі наявна велика кількість функцій та можливостей.

До ключових можливостей даного середовища можна віднести наступне:

- потужний та функціональний редактор коду з підсвічуванням синтаксису, авто-форматуванням та авто-відступами для підтримуваних мов;
- проста та потужна навігація в коді;
- допомога в написанні коду, що включає авто доповнення та авто імпорт, шаблони коду, а також перевірку на сумісність версій інтерпретатору мови;
- швидкий доступ до документації для будь-якого елементу прямо в вікні редактора;
- велика кількість інспекцій коду;
- потужний рефакторинг коду, що представляє багато можливостей для виконання глобальних змін в проекті.

3.2. Вибір мови програмування

Python – це високорівнева об'єктно-орієнтована та структурна мова програмування загального призначення, яка відноситься до категорії інтерпретованих мов і не потребує компіляції. Він є скриптова мова і відрізняється високим ступенем універсальності. Завдяки цьому, оптимально підходить для безлічі платформ та завдань, від серверних ОС до мобільних додатків під iOS/Android.

Мова Python була розроблена голландським інженером Гвідо ван Россумом у 1991 році. На той час він працював у Національному дослідному інституті математики та інформатики, де займався створенням іншої мови програмування – ABC. Пітон був його аматорським проектом, який позиціонувався як зрозуміла і зручна мова, що легко вивчається з нуля. Россум розробив перший робочий прототип Python на своєму домашньому комп'ютері за кілька вихідних та назвав його на честь популярного тоді телешоу «Повітряний цирк Монті Пайтона».

До кінця 90-х років Пітон розвивався виключно як хобі його творця, а в 1999 Гвідо ванн Россум уклав контракт з компанією BeOpen, яка зайнялася просуванням технології та підтримкою її розробки. У роки співпраці з BeOpen було випущено масштабне оновлення мови – Python 2.0. У ньому всі дані проекту були перекладені на SourceForge – великий хостинг та спільноту для розробників відкритого ПЗ, що дозволило програмістам з усього світу підключитись до роботи над удосконаленням цієї мови.

Згодом Россум розірвав контракт із BeOpen, почавши співпрацювати з іншою компанією під назвою Digital Creations. При взаємодії з нею команда розробників продовжила розвивати Python, зокрема, з'явилася версія 2.1, куди було додано ієрархію функцій та нові об'єкти. У грудні 2008 року відбулося чергове масштабне оновлення мови – Python 3.0 (Python 3000, Py3k), яке відчутно вдосконалило технологію, позбавивши її від низки недоліків архітектури. Версія 3.0 є актуальною не сьогодні, а підтримка Python 2 була припинена у 2020 році.

До основних особливостей та можливостей використання мови програмування Пайтон можна віднести наступне:

- він підтримує безліч парадигм програмування, зокрема об'єктноорієнтоване, функціональне, імперативне, структурне, процедурне, логічне, контрактне, аспектно-орієнтоване, метапрограмування тощо;
- програмування на Python дозволяє розбивати програми на складові - модулі, які можна об'єднувати в пакети;
- підтримується повна інтроспекція, що дозволяє отримати інформацію про тип і внутрішню структуру будь-якого об'єкта в процесі виконання програми;
- велика стандартна бібліотека має набір модулів для роботи з ОС, різними мережевими протоколами, архівами, мультимедійними форматами, текстовими кодуваннями, регулярними виразами, криптографічними протоколами і т.д. Підтримує юніт-тестування;
- також можливості Пітона включають ітератори та генератори, обробку винятків, управління контекстом виконання, декоратори та багато іншого.

До переваг використання Пайтону можна віднести наступне:

- швидкість розробки – для створення програм необхідно в рази менше коду ніж за допомогою інших популярних мов програмування;
- логічний синтаксис – синтаксис цієї мови є доволі простим для читання та розуміння коду, тому й для створення програм;
- наявність широкого вибору бібліотек – крім стандартної бібліотеки існує велика кількість додаткових бібліотек, що обновлюються щодня й для різних цілей;
- масштабованість – створені на Пайтоні програми та додатки легко та швидко розширяються та масштабуються завдяки можливостям мови; • універсальність – Пайтон є мовою що інтерпретується, що використовується практично на всіх сучасних платформах, код не потребує

компіляції та його код можна створювати в звичайному текстовому документі. 31

- глобальна спільнота – одним з головних факторів такої популярності мови є спільнота користувачів цієї мови, що підтримує розробку мови.

Проте крім того Пайтон має також ряд й недоліків які варто відзначити:

- недостатня швидкодія – сам по собі пайтон не дуже підходить для високопродуктивних проектів, для цього його необхідно комбінувати з іншими мовами;
- динамічна типізація – через це Пайтон являється доволі ресурсномістким та потребує доволі багато пам’яті.

3.3. Створення прототипу

Після вибору середовища розробки та мови програмування наступним кроком є створення програмного модулю голосового помічника.

Для початку створюємо новий проєкт в середовищі розробки (див. рис. 3.1.)

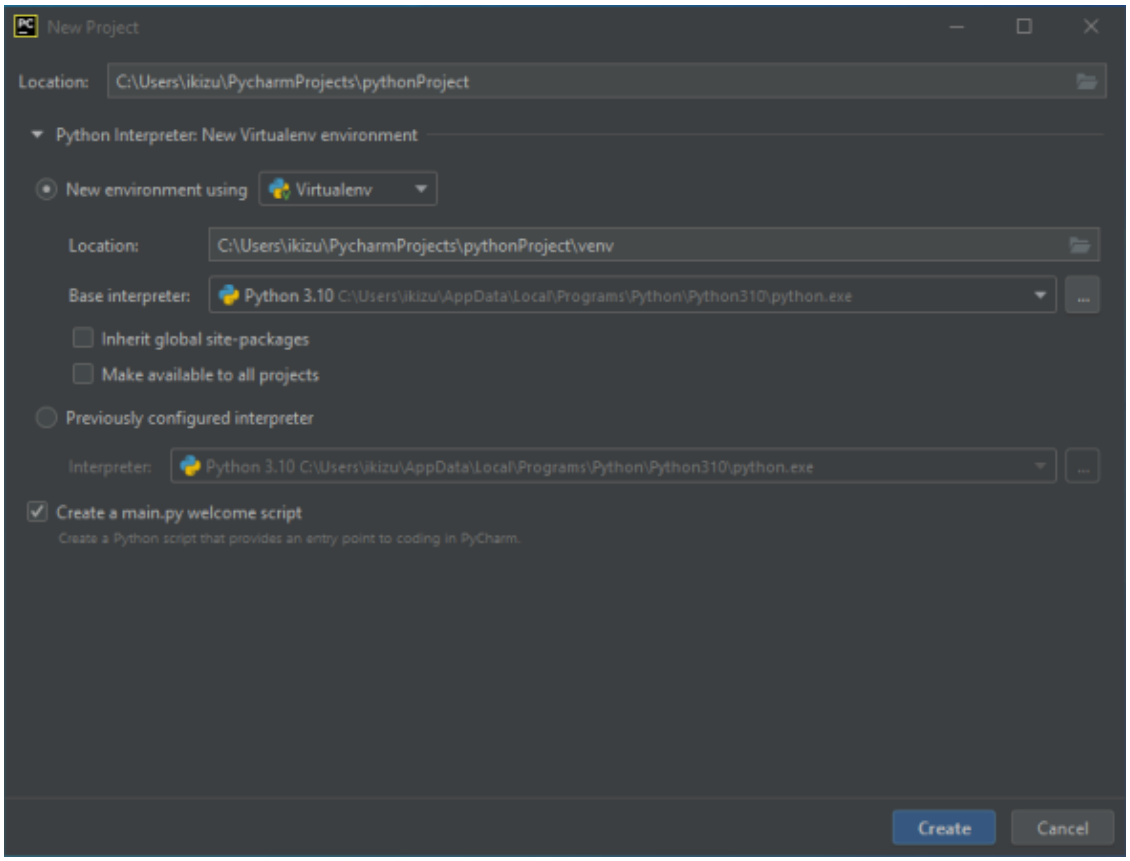

Рис. 3.1. Створення нового проєкту

Після створення нового проекту середовище розробки додає новий файл для запуску та створює відповідний каталог проєкту (див. рис. 3.2.).

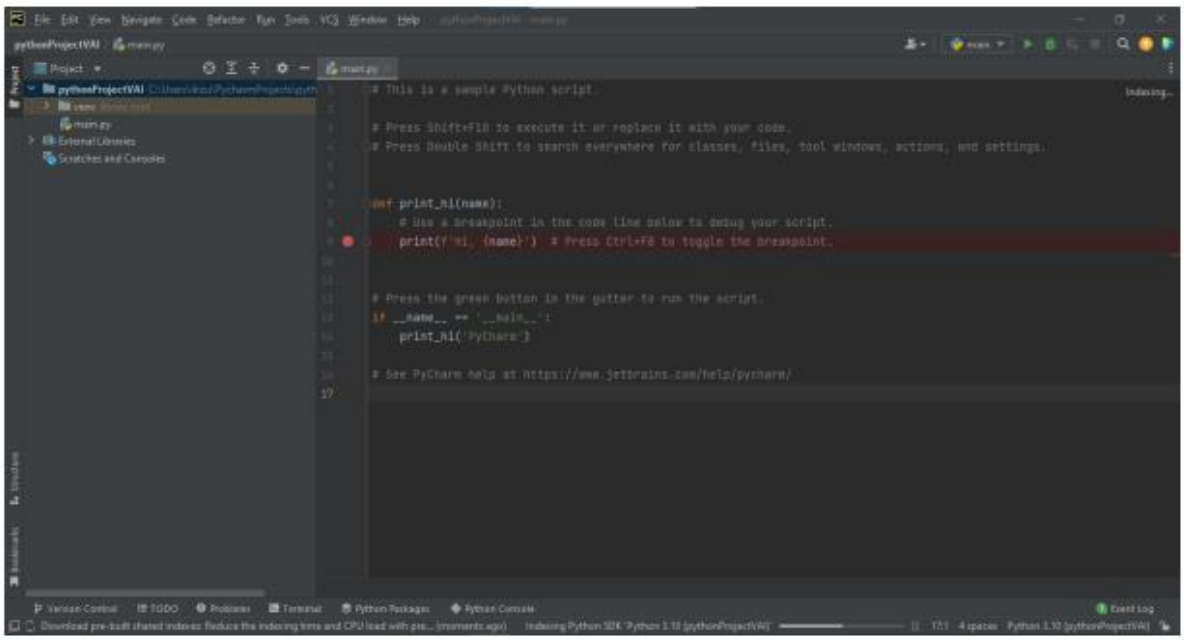

Рис. 3.2. Початок роботи

Далі необхідно створити файли, що будуть використовуватись під час розробки. В нашому випадку це наступна структура файлів (див. рис. 3.3.).

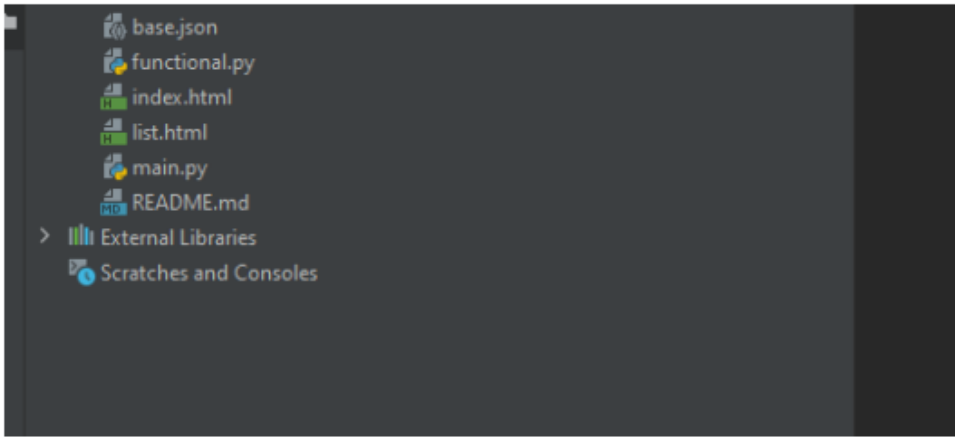

Рис. 3.3. Структура файлів

Для оптимальної роботи програмного модулю імпортує та встановимо необхідні модулі (див. рис. 3.4.).

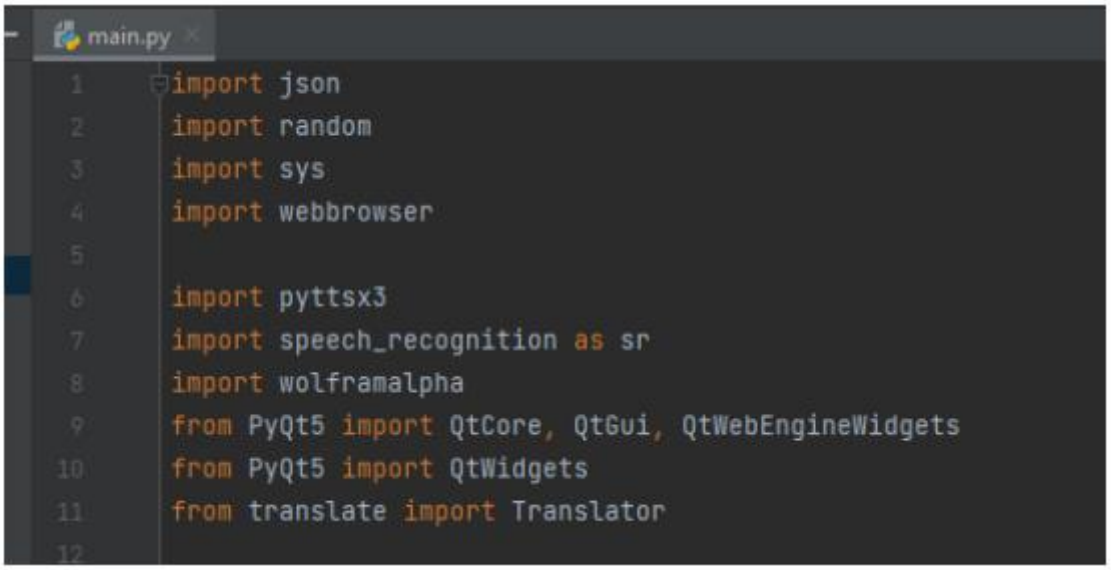

Рис. 3.4. Програмні модулі

Наступним кроком є створення головного інтерфейсу голосового помічника. Для цього використовуємо встановлений модуль та додаємо компоненти на головне вікно (див. рис. 3.5.)

Рис. 3.5. Головне вікно програмного модулю

Наступним кроком є додавання функцій до голосового помічника. Для цього використовуємо окремий файл, куди будемо додавати функції помічника відповідно до розглянутої раніше структури функцій. Наприклад, команда перегляду погоди виглядає наступним чином (див. рис.3.6.).

```
elif 'погода' in command:
    engine.say('Вот погода на ближайшие дни')
    engine.runAndWait()
    ui.textEdit.load(QtCore.QUrl(
        'https://www.google.com/search?q=%D0%BF%D0%BE%D0%B3%D0%BE%D0%B4%D0%B0&sxsrf=A
    addrobotphrasetohtml('Вот погода на ближайшие дни')
```

Рис. 3.6. Команда «Погода»

3.4. Висновки до розділу

У цьому розділі було розглянуто та обрано середовище для розробки – PyCharm, що підходить для програмування мовою Пайтон. Крім цього було розглянуто ключові особливості мови програмування Пайтон, проаналізовано переваги та недоліки використання цієї мови програмування. Наступним кроком було створення проєкту програмного модулю, створення відповідної файлової структури та головного вікна застосунку, що підтримує функції.

РОЗДІЛ 4.
ТЕСТУВАННЯ ПРОТОТИПУ

4.1. Тестування прототипу

Розглянемо створені інтерфейси прототипу програмного модулю голосового помічника. Першим кроком є перехід на головну сторінку програмного модулю. На даній сторінці користувач має можливість почати роботу, дізнатись інформацію про створений програмний модуль чи завершити роботу (див. рис. 4.1.).

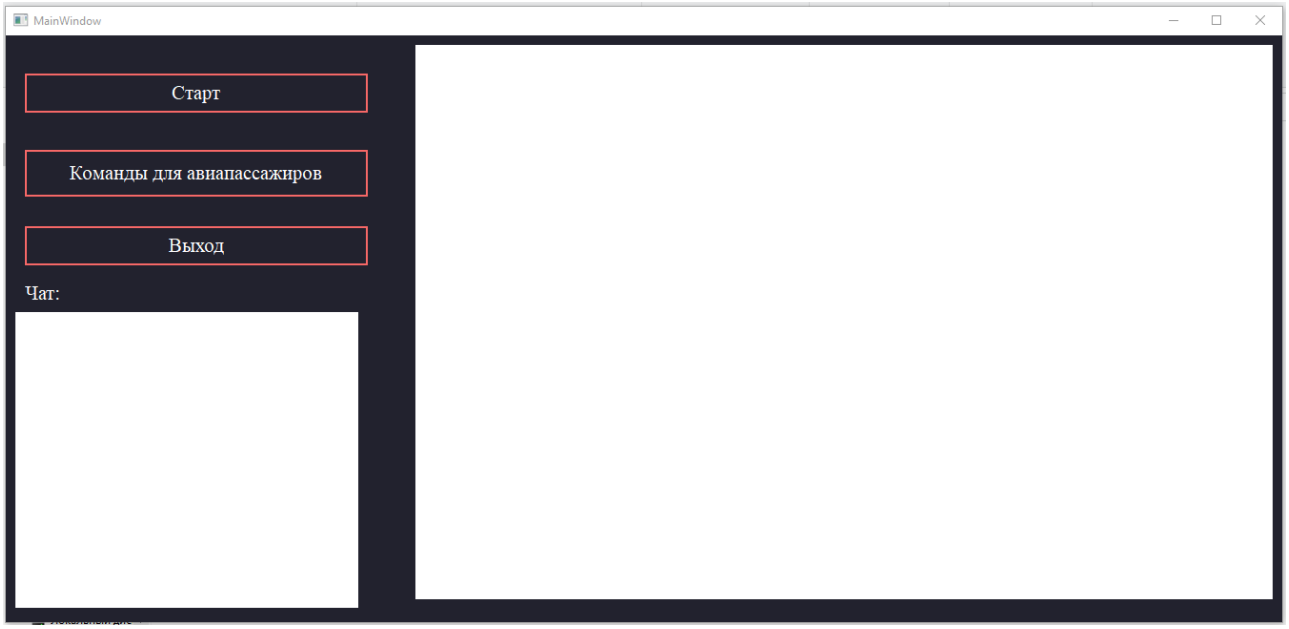

Рис. 4.1. Головна сторінка

На головній сторінці користувач може натиснути кнопку старт та почати працювати з помічником, натиснути команди для авіапасажирів – та отримати інформацію щодо доступних команд, кнопка вихід відповідає за вихід з програмного модулю. Екран поділено на декілька частин – це чат та область роботи.

Рис. 4.2. Кнопка «Команди для авіапасажирів»

Після натискання на дану кнопку голосовий помічник промовляє запрограмовану інформацію, яку можна змінити за бажанням. Перевіримо роботу голосового помічника за допомогою вводу команд та перевірки реакцій на них. Почнемо з перегляду погоди на даний момент (див. рис. 4.3.).

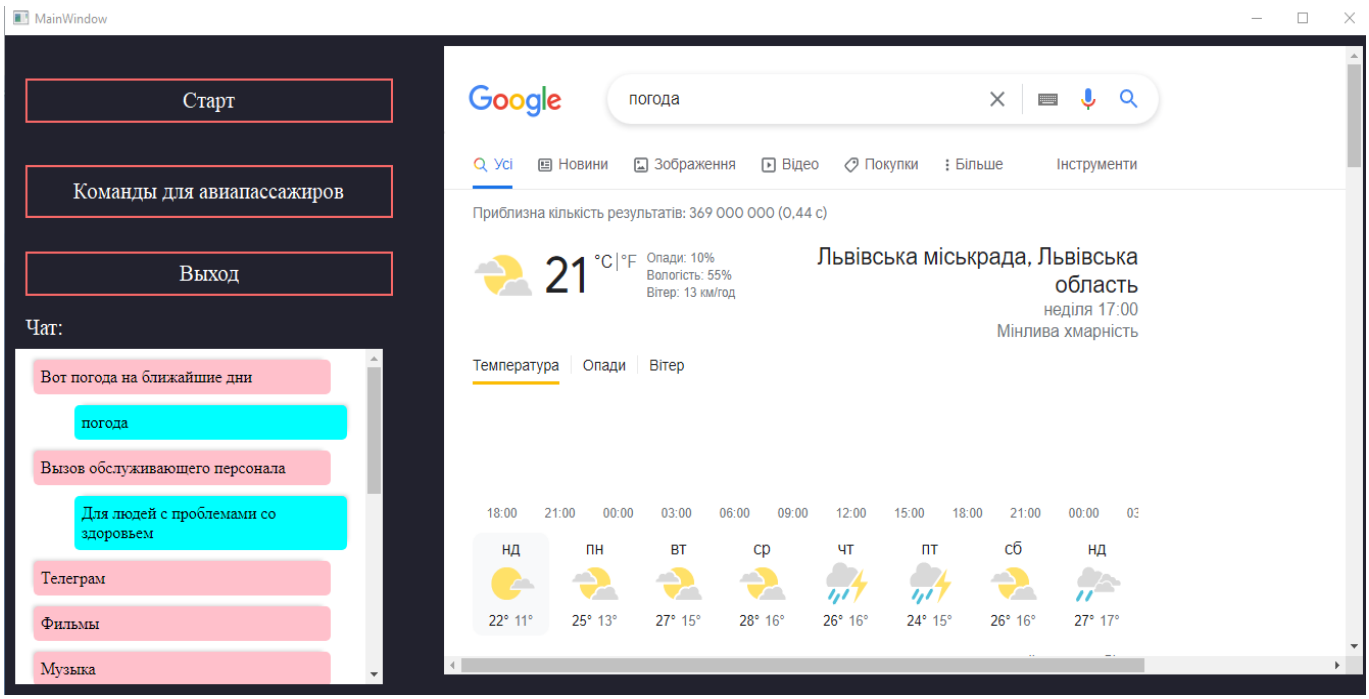

Рис. 4.3. Погода

Перевіримо чи відбувається робота функцій для розваг, для цього перевіримо відкриття Ютубу (див. рис. 4.4.).

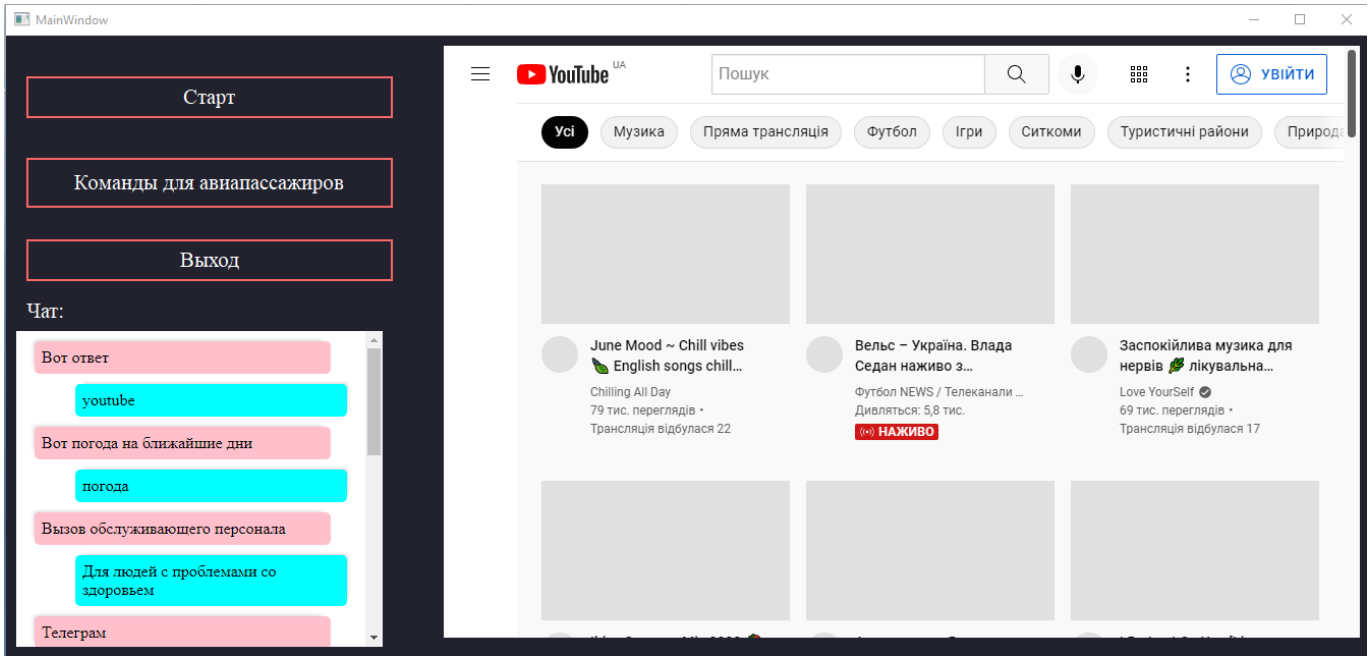

Рис. 4.4. Відкриття Ютубу

Також помічник може відкривати музику з Ютубу (див. рис. 4.5.).

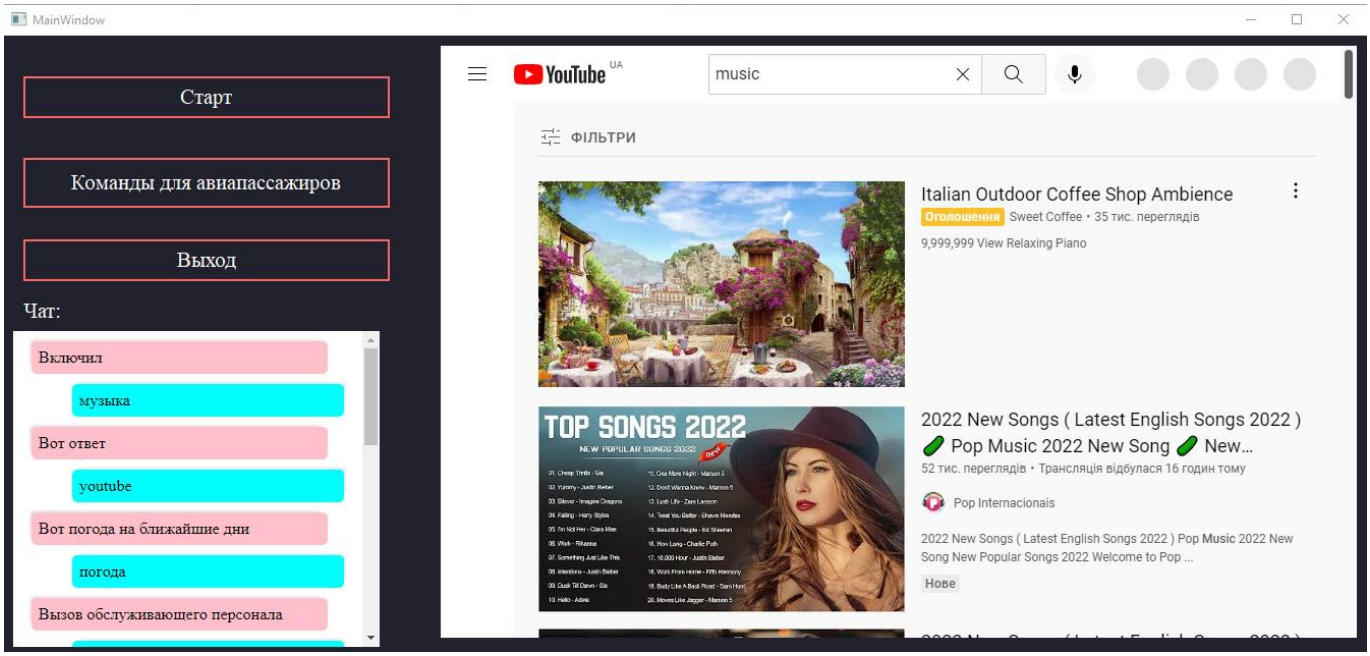

Рис. 4.5. Музика

Далі перевіримо можливість запуску пошти (див. рис. 4.6.).

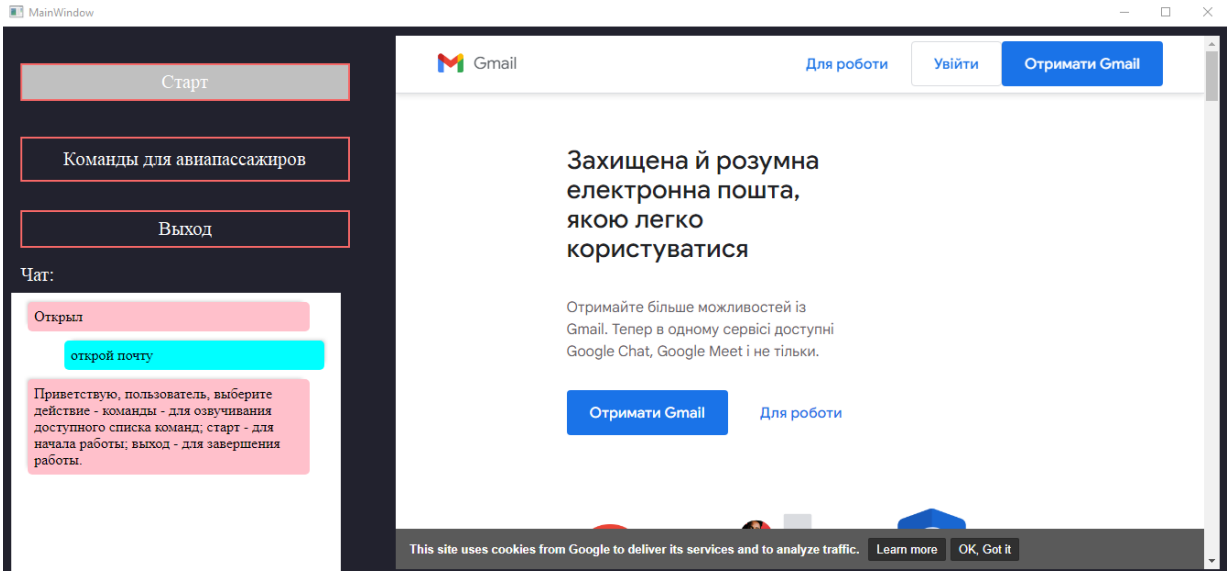

Рис. 4.6. Запуск пошти

Помічнику доступна функція перегляду новин на даний час (див. рис. 4.7.).

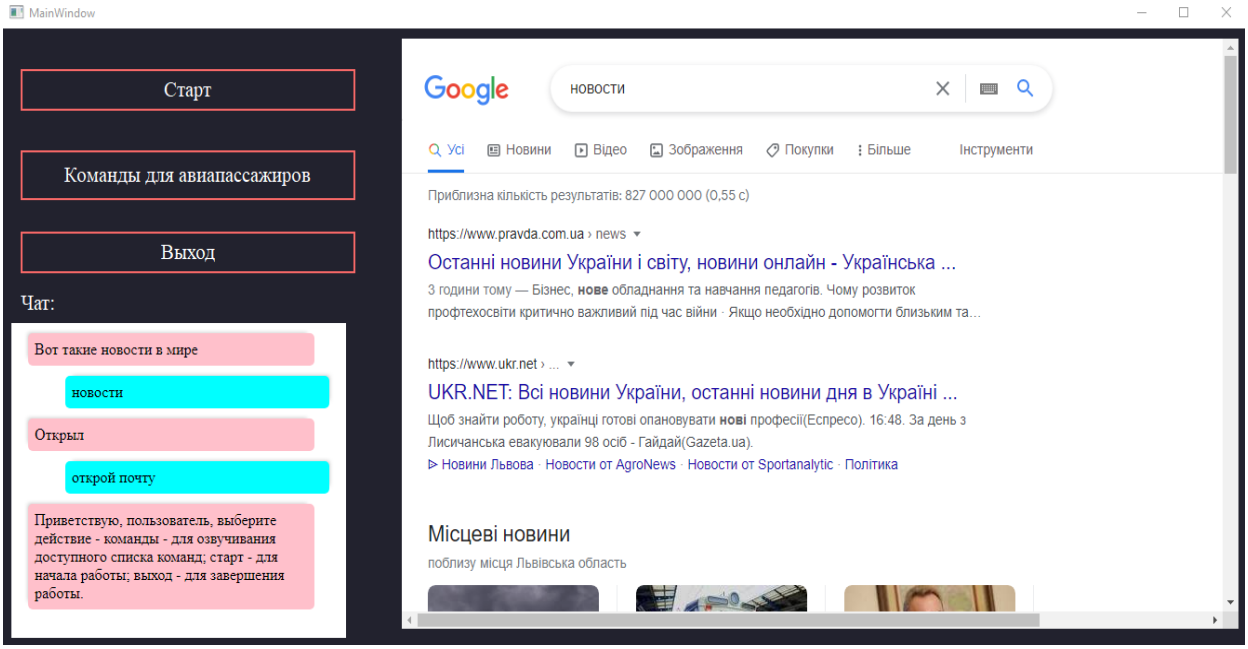

Рис. 4.7. Пошук новин

Перевіримо можливість виводу правил безпеки на борті літака, для цього скористаємось відповідною командою (див. рис. 4.8.).

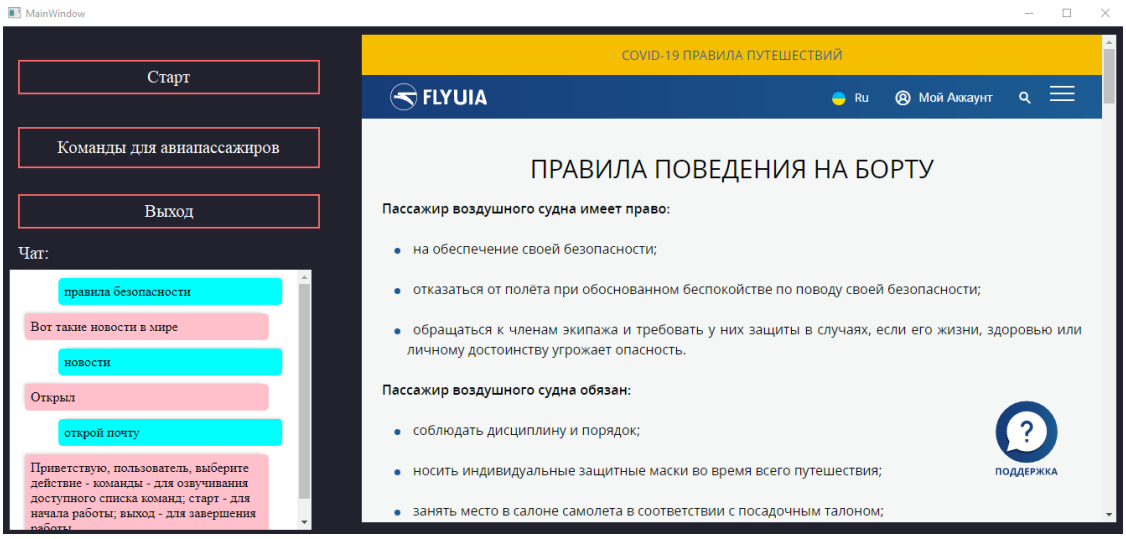

Рис. 4.8. Правила безпеки

Наступним кроком є перевірка отримання інформації про літаки компанії МАУ (див. рис. 4.9.).

4.2. Висновки до розділу

У ході виконання даного розділу було проведено тестування прототипу програмного модулю голосового помічника для авіапасажиру, у результаті чого було перевірено роботу та створені інтерфейси користувача. Було проведено тестування запуску помічника та перевірки працездатності функцій.

ВИСНОВКИ

В ході виконання першого розділу роботи було проведено аналіз предметної області програмного модулю голосового помічника для авіапасажиру.

В першому підпункті було досліджено історію створення голосового помічника – від початку зародження даної теми до нашого часу, проаналізовано основні моменти та досягнення в наш час.

В другому підпункті було проведене дослідження актуальних програмних аналогів голосового помічника, що використовуються у нас час. Було розглянуто такі помічники як Гугл, Сірі, Аліса та інші. Проаналізовано їх основні можливості, а також плюси й мінуси, складено таблицю аналізу даних помічників.

В третьому підпункті було проведено дослідження проблематики роботи – розглянуто актуальність та проблему, що вирішує програмний модуль голосового помічника.

У другому розділі було виконано загальний опис застосунку – було описано загальну інформацію програмного модулю голосового помічника, що буде створюватись, описано його функції та характеристики, а також основних користувачів.

Також було створено ряд вимог до застосунку, сюди можна віднести наступні вимоги: функціональні – що забезпечать повноцінну функціональність застосунку, відповідно до вимог користувача, не функціональні – що не залежать від функцій застосунку, проте відповідають вимогам користувачів, системні вимоги – вимоги до апаратної частини для запуску застосунку.

Також було створено логічне представлення блоків застосунку – у вигляді блоку функцій – сюди відносимо блок для роботи, для розваг та з вадами здоров’я.

Було визначено функції застосунку – сюди можна віднести здійснення пошуку в мережі інтернет, запуск різного роду застосунків, запуск музики та Youtube, створення заміток та розкладу.

До кожної функції було створено Use Case для більш детального опису та створення сценарію взаємодії користувача з програмним модулем голосового помічника. Крім цього були створені діаграми прецедентів та послідовності.

У третьому розділі було розглянуто та обрано середовище для розробки – PyCharm, що підходить для програмування мовою Пайтон. Крім цього було розглянуто ключові особливості мови програмування Пайтон, проаналізовано переваги та недоліки використання цієї мови програмування. Наступним кроком було створення проекту програмного модулю, створення відповідної файлової структури та головного вікна застосунку, що підтримує функції.

У ході виконання четвертого розділу було проведено тестування прототипу програмного модулю голосового помічника для авіапасажиру, у результаті чого було перевірено роботу та створені інтерфейси користувача. Було проведено тестування запуску помічника та перевірки працездатності функцій.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Mah Daniel C.H. Role of Satellite in 5G. SAS, Satellite Connectivity Workshop, Nadi, Fuji, 24, April, 2017.
2. Н.Б. Васильєва, Д.Я. Федорин. Проблеми створення систем розпізнавання мовлення для різних комп'ютерних платформ / «Штучний інтелект», 2013, № 4. – С. 158-167.
3. Шипота Н.А. Розвиток комунікаційної інфраструктури з використанням хмарних сервісів програмно-конфігурованих мереж // Актуальні проблеми гуманітарних та природничих наук: V Міжнародна науково-практична конференція. Харків, 2018. С. 165–166.
4. *Gorodetsky V. Internet of Agents: From Set of Autonomous Agents to Network Object* // *Second International Workshop on Internet of Agents*. 2017. pp. 1–17.
5. В.В. Робейко, М.М. Сажок. Розпізнавання спонтанного мовлення на основі акустичних композитних моделей слів у реальному часі. – "Штучний інтелект", № 4, 2012 (українською), УкрОбраз'2012 (in English).
6. Технологія Intel IoT Gateways. (2018). Офіційний сайт компанії Intel. https://software.intel.com/ru-ru/iot/hardware/gateways
7. Технології інтернету речей. Навчальний посібник [Електронний ресурс]: навч. посіб. Для студ. спеціальності 126 «Інформаційні системи та технології», спеціалізація «Інформаційне забезпечення робототехнічних систем» / Б. Ю. Жураковський, І.О. Зенів; КПІ ім. Ігоря Сікорського. – Електронні текстові дані (1 файл: 12,5 Мбайт). – Київ: КПІ ім. Ігоря Сікорського, 2021. – 271 с.
8. Голосовий асистент «Siri» [Електронний ресурс]. — Режим доступу: https://developer.apple.com/siri/

ДОДАТОК А

```
import json
import random
import sys
import webbrowser
import pyttsx3
import speech_recognition as sr
import wolframalpha
from PyQt5 import QtCore, QtGui, QtWebEngineWidgets
from PyQt5 import QtWidgets
from translate import Translator
import functional

translator = Translator(from_lang="ru", to_lang="eng")
translator2 = Translator(from_lang="eng", to_lang="ru")
# Инициализируем SAPI5
engine = pyttsx3.init()
# Получение списка голосов
voices = engine.getProperty('voices')
# Установка русского языка
engine.setProperty('voice', 'eng')
# Добавляем распознавание речи Google
r = sr.Recognizer()

# Получаем html шаблон для сообщений в окне чата
htmlcode = '<div class="robot">Что вас интересует? </div>';
engine.say(htmlcode)
engine.runAndWait()

f = open('index.html', 'r', encoding='UTF-8')
htmltemplate = f.read()
f.close()

# Работаем с Wolframaplha

client = wolframalpha.Client('52HHA3-8QTWVTA53Q')

file = open('base.json', 'r').read()
```

```
for voice in voices:
    if voice.name == 'Elena':
        engine.setProperty('voice', voice.id)
    else:
        pass
    # Получение доступа к микрофону

global text


def record_volume():
    with sr.Microphone(device_index=1) as source:
        r.adjust_for_ambient_noise(source, duration=0.5)  #
настройка посторонних шумов
        print('Слушаю...')
        engine.say('Слушаю')
        engine.runAndWait()
        audio = r.listen(source)
    print('Услышала.')
    engine.say('Услышала')
    engine.runAndWait()
    try:

        query = r.recognize_google(audio, language='ru-RU')

        text = query.lower()
        # query = text

        # engine.say('Вы сказали '+ query)
        # engine.runAndWait()
    except sr.UnknownValueError:
        return record_volume()
    except sr.RequestError:
        print('У меня нет доступа к серверам Google, для
распознования вашей команды')

    return text
```

```
def start():
    myquestion(record_volume())

def send():
    engine.say(
        'Привет пользователь это дема кнопка, а так меня
зовут Jarvis я мини искусственный интелект и голосовой
помощник, вскорем времени я надеюсь что мой разработчик меня
улучшит, вместо этой кнопки будет выступать сайт с
информацие обо мне')
    engine.runAndWait()
    addyouphrasetohtml('О нас')
    addrobotphrasetohtml(
        'Привет пользователь это дема кнопка, а так меня
зовут Jarvis я мини искусственный интелект и голосовой
помощник, вскорем времени я надеюсь что мой разработчик меня
улучшит, вместо этой кнопки будет выступать сайт с
информацие обо мне')


def quit():  # функция выхода из программы

    x = ['Пользователь, жаль что ты уходишь', 'рада была
помочь', 'всегда к вашим услугам']  # Несколько вариаций
    engine.say(random.choice(x))  # Проговорить переменную x
с помощью биб pyttsx3
    engine.runAndWait()
    engine.stop()
    exit(0)  # завершение


class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(937, 519)
        MainWindow.setStyleSheet("background-color:
#22222e")
        self.centralwidget = QtWidgets.QWidget(MainWindow)
```

```
        self.centralwidget.setObjectName("centralwidget")
        self.pushButton =
QtWidgets.QPushButton(self.centralwidget)
        self.pushButton.setGeometry(QtCore.QRect(20, 40,
360, 41))
        font = QtGui.QFont()
        font.setFamily("Times New Roman")
        font.setPointSize(16)
        self.pushButton.setFont(font)
        self.pushButton.setStyleSheet("QPushButton{\n"
                                      "background-color:
#22222e;\n"
                                      "border: 2px solid
#f66867;\n"
                                      "border-radius:
25%;\n"
                                      "color:white;\n"
                                      "}\n"
                                      "QPushButton::hover\n"
                                      "{\n"
                                      "    background-
color:silver;\n"
                                      "}\n"
                                      "\n"

"QPushButton::pressed\n"
                                      "{\n"
                                      "    background-
color:red;\n"
                                      "}")
        self.pushButton.setObjectName("pushButton")
        self.pushButton_2 =
QtWidgets.QPushButton(self.centralwidget)
        self.pushButton_2.setGeometry(QtCore.QRect(20, 120,
360, 49))
        font = QtGui.QFont()
        font.setFamily("Times New Roman")
        font.setPointSize(16)
        self.pushButton_2.setFont(font)
```

```
        self.pushButton_2.setStyleSheet("QPushButton{\n"
                                         "background-color:
#22222e;\n"
                                         "border: 2px solid
#f66867;\n"
                                         "border-radius:
25%;\n"
                                         "color:white;\n"
                                         "}\n"

"QPushButton::hover\n"
                                         "{\n"
                                         "    background-
color:silver;\n"
                                         "}\n"
                                         "\n"

"QPushButton::pressed\n"
                                         "{\n"
                                         "    background-
color:red;\n"
                                         "}")
        self.pushButton_2.setObjectName("pushButton_2")
        self.pushButton_3 =
QtWidgets.QPushButton(self.centralwidget)
        self.pushButton_3.setGeometry(QtCore.QRect(20, 200,
360, 41))
        font = QtGui.QFont()
        font.setFamily("Times New Roman")
        font.setPointSize(16)
        self.pushButton_3.setFont(font)
        self.pushButton_3.setStyleSheet("QPushButton{\n"
                                         "background-color:
#22222e;\n"
                                         "border: 2px solid
#f66867;\n"
                                         "border-radius:
25%;\n"
                                         "color:white;\n"
```

```
                                        "}\n"

"QPushButton::hover\n"
                                        "{\n"
                                        "    background-
color:silver;\n"
                                        "}\n"
                                        "\n"

"QPushButton::pressed\n"
                                        "{\n"
                                        "    background-
color:red;\n"
                                        "}")
        self.pushButton_3.setObjectName("pushButton_3")
        self.textEdit2 =
QtWebEngineWidgets.QWebEngineView(self.centralwidget)
        self.textEdit2.setGeometry(QtCore.QRect(10, 290,
360, 310))
        font = QtGui.QFont()
        font.setFamily("Times New Roman")
        font.setPointSize(16)
        self.textEdit2.setFont(font)
        self.textEdit2.setStyleSheet("background-color:
white;\n"
                                     "border: 2px solid
#f66867;\n"
                                     "border-radius: 25%;\n"
                                     "color:black;")
        self.textEdit2.setObjectName("plainTextEdit")
        self.label = QtWidgets.QLabel(self.centralwidget)
        self.label.setGeometry(QtCore.QRect(20, 260, 261,
21))
        font = QtGui.QFont()
        font.setFamily("Times New Roman")
        font.setPointSize(16)
        self.label.setFont(font)
        self.label.setStyleSheet("\n"
                                 "color:white;")
```

```
        self.label.setObjectName("label")
        self.label_2 = QtWidgets.QLabel(self.centralwidget)
        self.label_2.setGeometry(QtCore.QRect(500, 10, 191,
21))
        font = QtGui.QFont()
        font.setFamily("Times New Roman")
        font.setPointSize(14)
        self.label_2.setFont(font)
        self.label_2.setStyleSheet("color:white;")
        self.label_2.setObjectName("label_2")
        self.textEdit =
QtWebEngineWidgets.QWebEngineView(self.centralwidget)
        self.textEdit.setGeometry(QtCore.QRect(430, 10, 900,
581))
        self.textEdit.setStyleSheet("background-color:
white;\n"
                                    "border: 2px solid
#f66867;\n"
                                    "border-radius: 25%;\n"
                                    "color:black;font-
size:16px;")
        self.textEdit.setObjectName("textEdit")
        MainWindow.setCentralWidget(self.centralwidget)

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

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow",
"MainWindow"))
        self.pushButton.setText(_translate("MainWindow",
"Старт"))
        self.pushButton_2.setText(_translate("MainWindow",
"О нас"))
        self.pushButton_3.setText(_translate("MainWindow",
"Выход"))
        self.label.setText(_translate("MainWindow", "Чат:"))
```

```
        self.label_2.setText(_translate("MainWindow",
"Информация:"))


def addrobotphrasetohtml(phrase):
    global htmltemplate
    global htmlcode
    htmlcode = '<div class="robot">' + phrase + '</div>' +
htmlcode
    htmlresult = htmltemplate.replace('%code%', htmlcode)
    ui.textEdit2.setHtml(htmlresult,
QtCore.QUrl("file://"));
    ui.textEdit2.show()


# Добавление в html чат фразы пользователя
def addyouphrasetohtml(phrase):
    global htmltemplate
    global htmlcode
    htmlcode = '<div class="you">' + phrase + '</div>' +
htmlcode
    htmlresult = htmltemplate.replace('%code%', htmlcode)
    ui.textEdit2.setHtml(htmlresult,
QtCore.QUrl("file://"));
    ui.textEdit2.show()


def AI(command):
    pars = json.loads(file)
    if command in pars:
        z = random.choice(pars[command])
        dlina = len(z)
        if dlina > 1:
            engine.say(z)
            engine.runAndWait()
            addrobotphrasetohtml(z)
        else:
            z = pars[command]
            engine.say(z)
```

```
            engine.runAndWait()
            addrobotphrasetohtml(z)
    else:
        engine.say('я не знаю как на это отвечать, научите
меня')
        engine.runAndWait()
        with open('base.json', 'r') as json_file:
            pars = json.load(json_file)
        engine.say('Задайте вопрос')
        engine.runAndWait()
        a = record_volume()
        engine.say('скажите как я должен ответить на это')
        engine.runAndWait()
        b = record_volume()
        pars[a] = b
        with open('base.json', 'w') as json_file:
            json.dump(pars, json_file, indent=2,
sort_keys=True, ensure_ascii=False)


def myquestion(command):
    addyouphrasetohtml(command)
    if 'пока' in command:
        engine.say('До свидания  человек')
        engine.runAndWait()
        addrobotphrasetohtml('До свидания  человек')
        quit()
    elif 'погода' in command:
        engine.say('Вот погода на ближайшие дни')
        engine.runAndWait()
        ui.textEdit.load(QtCore.QUrl(

'https://www.google.com/search?q=%D0%BF%D0%BE%D0%B3%D0%BE%D0
%B4%D0%B0&sxsrf=ALeKk014aEh7iR8XI88ApKajO8NYopJYuw%3A1618161
291608&ei=iy5zYJXNJOLGrgTJw5uwDw&oq=%D0%BF%D0%BE%D0%B3%D0%BE
%D0%B4%D0%B0&gs_lcp=Cgdnd3Mtd2l6EAEYADIHCCMQsAMQJzIHCCMQsAMQ
JzIHCCMQsAMQJzIHCAAQRxCwAzIHCAAQRxCwAzIHCAAQRxCwAzIHCAAQRxCw
AzIHCAAQRxCwAzIHCAAQRxCwA1AAWABgkowCaABwAXgAgAFPiAGdAZIBATKYAQCqAQdnd3Mtd2l6yAEKwAEB&sclient=gws-wiz'))
```

```
        addrobotphrasetohtml('Вот погода на ближайшие дни')
    elif ('время' in command) or ('текущее время' in
command):
        functional.time()
        ui.textEdit.load(QtCore.QUrl(

'https://www.google.com/search?q=%D0%B2%D1%80%D0%B5%D0%BC%D1
%8F&oq=dhtv&aqs=chrome.1.69i57j0i1i10l5j46i1i10j0i1i10l3.144
9j0j7&sourceid=chrome&ie=UTF-8'))
        addrobotphrasetohtml('Вот ответ')
    elif ('запусти' in command):
        functional.zapusti(command)
        addrobotphrasetohtml('Запустил')
    elif ('ютуб' in command) or ('youtube' in command):
        ui.textEdit.load(QtCore.QUrl('https://youtube.com'))
        addrobotphrasetohtml('Вот ответ')

    elif ('открой почту' in command) or ('почта' in
command):
        engine.say('Сделано')
        engine.runAndWait()
        ui.textEdit.load(QtCore.QUrl('https://gmail.com'))
        addrobotphrasetohtml('Открыл')
    elif ('открой телеграм' in command) or ('открой tg' in
command):

ui.textEdit.load(QtCore.QUrl('https://web.telegram.org/z/'))
        addrobotphrasetohtml('Открыл')
    elif ('включи музыку' in command) or ('музыка' in
command):

ui.textEdit.load(QtCore.QUrl('https://www.youtube.com/result
s?search_query=music'))
        addrobotphrasetohtml('Включил')
    elif 'посмотреть фильм' in command:

webbrowser.open('https://www.youtube.com/results?search_quer
y=films')
        addrobotphrasetohtml('Вот ответ')
```

```
    elif ('новости' in command) or ('текущие новости' in
command):
        engine.say('Вот такие новости в мире')
        engine.runAndWait()

ui.textEdit.load(QtCore.QUrl('https://www.google.com/search?
q=%D0%BD%D0%BE%D0%B2%D0%BE%D1%81%D1%82%D0%B8&oq=%D0%BD%D0%BE
%D0%B2%D0%BE%D1%81%D1%82%D0%B8&aqs=chrome..69i57j35i39l2j0i6
7i457j0i67j69i61l3.706j0j7&sourceid=chrome&ie=UTF-8'))
        addrobotphrasetohtml('Вот такие новости в мире')

    elif ('перевести' in command) or ('переводчик' in
command):
        engine.say('перевести с английского или русского
языка')
        engine.runAndWait()
        z = record_volume()
        if 'с русского' in z:
            engine.say('Скажите что перевести')
            engine.runAndWait()
            word = record_volume()
            end_text = translator.translate(word)
            addrobotphrasetohtml(end_text)
            engine.say(end_text)
            engine.runAndWait()

        else:
            engine.say('Скажите что перевести')
            engine.runAndWait()
            word = record_volume()
            end_text = translator2.translate(word)

            addrobotphrasetohtml(end_text)
            engine.say(end_text)
            engine.runAndWait()
            # addrobotphrasetohtml(end_text)
    elif ('найти' in command) or ('найди' in command):
        words = (
            'найди', 'найти', 'ищи', 'кто такой',
```

```
            'что такое', 'о том')  # Создание словаря с
ключевыми словами для выполнения запроса
        remove = ["пожалуйста", "ладно", "давай", "сейчас"]
# Создание списка со слов которые будут удалены из запроса
        for i in words:  # Создание цикла для очистки слов
находящихся в словаре words в запросе
            command = command.replace(i, '')  # Очистка
ключевых слов, находящихся в словаре words с запроса
            for j in remove:  # Создание цикла для очистки
слов находящихся в списке remove в запросе
                command = command.replace(j, '')  # Очистки
слов находящихся в списке remove в запросе
                command = command.strip()  # Преобразование
переменной search в строку

        ui.textEdit.load(QtCore.QUrl(

f'https://www.google.com/search?q={command}&oq={command}'f'8
1&aqs=chrome..69i57j46i131i433j0l5.2567j0j7&sourceid=chrome&
ie=UTF-8'))  # выполнение запроса в браузере
        addrobotphrasetohtml('вот ответ')
    elif 'список команд' in command:
        webbrowser.open('list.html')
    else:
        AI(command)
```