МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВІДОКРЕМЛЕНИЙ СТРУКТУРНИЙ ПІДРОЗДІЛ
«ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність:

123 – «Комп'ютерна інженерія»

Освітня програма:

«Обслуговування комп'ютерних систем і мереж»

Група: 4КС-56

Дипломний проект

студента денної форми навчання
КС 56.10.000.00 ДП ПЗ

КОДРЯНУ ПАВЛА МИХАЙЛОВИЧА

м. Одеса
2023 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «Одеський технічний фаховий коледж ОНТУ»

Спеціальність *123 – «Комп'ютерна інженерія»*

Освітня програма *«Обслуговування комп'ютерних систем і мереж»*

Група *4КС-56*

ПОЯСНЮВАЛЬНА ЗАПИСКА

До дипломного проекту (роботи) на тему: ***Проектування системи голосового управління "Розумним будинком"***

Проектний матеріал складається з пояснювальної записки на *81* сторінках та графічного матеріалу на *17* аркушах (слайдах).

Дипломник Код (*Кодряну П.М.*)

Керівник проекту (*Скорнякова О.В.*)

Консультанти:

з економічної частини (*Копайгородська Т.Г.*)

з охорони праці (*Чорновол Н.І.*)

за дотриманням вимог ЄСКД (*Петрашова В.І.*)

старший консультант (*Кривченко Ю.В.*)

До захисту допущений

Голова циклової комісії (*Кривченко Ю.В.*)

Завідувач відділенням (*Скорнякова О.В.*)

Захист «*19*» *червня* 202*3* р. Протокол ДКК № *1*

Оцінка ДКК *5 (відмінно)*

Секретар ДКК

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «Одеський технічний фаховий коледж ОНТУ»

Відділення *комп'ютерних систем* Комісія *КТ і ПІ*

Спеціальність *123 "Комп'ютерна інженерія"*

ЗАТВЕРДЖУЮ:

Заст. дир. з НВР _____

Беркань І.В.

"_____" _____ 20___ р.

ЗАВДАННЯ

на дипломний проект (роботу)

Здобувачу освіти **Кодряну Павлу Михайловичу**

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) **Проектування системи голосового управління "Розумним будинком"**

затверджена наказом по коледжу від "**17**" **10** 20**22** р. № **235-А2-ОД**

2. Термін здачі здобувачем освіти закінченого проекту (роботи) _____

3. Вихідні дані до проекту (роботи) **Операційна система Windows 10, мова програмування Python, редактор VS Code, бібліотеки для програми PyQt6, SpeechRecognition, Requests, gTTS, thefuzz, pydub, бібліотека для серверу Flask**

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) **1.Технологічний розділ. 2. Економічний розділ. 3. Охорона праці. Висновки. Список використаних джерел. Додатки.**

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) **Слайд 1 – Титульний (тема проекту, виконавець, керівник). Слайд 2 – Вступ. Слайд 3 – Огляд існуючих рішень. Слайд 4 – Формування кінцевого завдання на розробку. Слайд 5 – Підбір залежностей. Слайд 6 – Розробка програми. Слайд 7 – Клас MainWindow. Слайд 8 – Клас MainButtonThread. Слайд 9 – Функція language_func. Слайд 10 – Функція command_match. Слайд 11 – Функція send_command. Слайд 12 – Фукнція answer_to_user. Слайд 13 – Сервер на flask. Слайд 14 – Запуск розробки. Слайд 15 – Інтерфейс розробки. Слайд 16 – Демонстрація. Слайд 17 – Дякую за увагу.**

6. Консультанти по проекту (роботі), із зазначенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Розділ 1-3	*Скорнякова О.В.*		
Економ.частина	*Копайгородська Т.Г.*		
Охорона праці	*Чорновол Н.І.*		
Нормоконтроль	*Петрашова В.І.*		

7. Дата видачізавдання _____

Керівник *Скорнякова О.В.* _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

Пор. №	Назва етапів дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Примітка
1.	*Огляд літератури. Огляд існуючих рішень*	*20.02.2023*	
2.	*Формування кінцевого завдання на розробку. Вступ.*	*01.03.2023*	
3.	*Аналітичний огляд. Огляд існуючих рішень*	*20.03.2023*	
4.	*Конструкторський розділ. Вибір елементної бази*	*10.04.2023*	
5.	*Розробка алгоритму та управляючої програми*	*17.04.2023*	
6.	*Економічний розділ. Проведення розрахунків щодо*	*01.05.2023*	
	економічної доцільності розробки		
7.	*Виконання розділу «Охорона праці»*	*15.05.2023*	
8.	*Виконання графічної частини дипломного проекту*	*22.05.2023*	
9.	*Підготовка до попереднього захисту, підготовка до*	*01.06.2023*	
	захисту		
10.	*Підготовка доповіді та презентації для захисту*	*10.06.2023*	
11.	*Отримання рецензії, відповіді на зауваження*	*до 10.06.2023*	
	рецензента		
12.	*Захист роботи*	*до 30.06.2023*	

Дипломник _____
(підпис)

Керівник проекту _____
(підпис)

Формат	Зона	Поз.	Позначення	Назва	Кіл.	Примітка
				Документація		
			КС 56. 10 000. 00 ДП	Дипломний проект	1	
А4			КС 56. 10 000. 00 ДП ПЗ	Пояснювальна	1	
				записка		

Зм.	Арк.	№ докум.	Підпис	Дата	КС 56. 10 000. 00 ДП		
Розробив		Кодряну П.М.			Проектування системи голосового управління "Розумним будинком"	Літ.	Аркуш	Аркушів
Перевірив		Скорнякова О.В.				Н Д П		
Н. Контр.		Петрашова В.І.				ОТФК ОНТУ 4КС-56		
Затвердив		Кривченко Ю.В.						

ЗМІСТ

ВСТУП

Досить швидкі темпи розвитку сучасної цивілізації потребують забезпечення інноваційними розробками у всіх сферах життєдіяльності людини. Відколи в наших домівках появилася електроенергія, людство намагалося різними способами покращити своє житло та перебування в ньому, зробити свою звичну домашню роботу комфортнішою за допомогою різних електронних приладів. «Розумний будинок» – це нова концепція використання звичних нам пристроїв, які є в будинку.

У нашій країні це все ще нова тема для обговорення, де технічні новинки допомагають полегшити нам побут, зробити його безпечним і економічним. Прогрес у розвитку технології «розумний будинок» веде до того, що будинок стане єдиним механізмом контролю електронного мозку, де кожен елемент системи буде працювати на благо власника. Цю технологію взяли на розробку такі великі бренди, як: Samsung, Sony, Electrolux. На основі цього можемо зробити висновок, що перспективи на майбутнє дуже величезні.

На сьогоднішній день в світі розроблені і впроваджені сотні різних систем автоматизації. Але, по-перше, вони мало застосовуються в Україні без адаптації, так як розроблені з урахуванням європейських або американських стандартів управління. По-друге, практично не існує підтримки подібних рішень в Україні, автоматизація є тривалим і трудомістким процесом, що вимагає постійної участі предметного фахівця, розробника і постачальника устаткування. При цьому основна складність полягає в тому, що ці системи встановлюють комплектом, по заздалегідь підготовленій інженерно-архітектурної концепції, з якою не повинно бути невідповідностей. Всі ці фактори в сукупності роблять «розумний дім» недоступним для величезної частини споживачів вітчизняного ринку, так як у більшості немає коштів, для покупки цілого комплекту систем.

Система «розумний будинок» може бути спроектована та розроблена за допомоги одного контролера, який має можливість керувати та контролювати різні пов'язані між собою пристрої, такі як штепсельні вилки, світильники,

датчики температури та вологості, димові, газові та пожежні сповіщувачі, а також аварійні та охоронні системи. Одна з найбільших переваг системи полягає в тому, що можна легко керувати та управляти масивом пристроїв, таких як смартфон, планшет, настільний комп'ютер та ноутбук.

Голосовий інтерфейс - це програмний продукт, який за допомогою голосової або мовної платформи дозволяє взаємодіяти користувачеві з системою «Розумний будинок», віддаючи команди для запуску окремих сценаріїв і дій голосом і отримуючи інформацію в формі звукових повідомлень. Відповідно, основними специфічними функціями таких інтерфейсів є розпізнавання голосових команд і синтез звукових повідомлень.

Мета даного дипломного проекту - проектування системи голосового управління розумним будинком, яка має на меті полегшити управління системами розумного будинку. А саме проектування програмної частини системи, яка буде створюватись під вже існуючий розумний будинок.

В основному розділі проаналізовано існуючі аналоги, розглядаються питання реалізації автономних систем управління, їх типів та варіантів реалізації. Здійснено вибір елементної бази для розробки системи, що проектується у дипломному проекті, далі – питання функціонування та налаштування системи. Останні розділі присвячені економічному розрахунку вартості системи та розглядаються питання охорони праці.

1 ТЕХНОЛОГІЧНИЙ РОЗДІЛ

1.1 Огляд існуючих рішень та прототипів

Розумний будинок - це сучасна технологія, яка дозволяє забезпечити автоматизоване керування всіма системами будинку з використанням смартфону, планшету або голосового помічника. Такий будинок забезпечує зручність, комфорт і безпеку для мешканців.

Основною метою розумного будинку є покращення якості життя людей, що проживають в ньому. Це досягається завдяки автоматизованому контролю над освітленням, опаленням, вентиляцією, кондиціюванням повітря, системою безпеки і т.д.

Однією з головних переваг розумних будинків є зручне та просте управління системами будинку з використанням мобільного додатку або голосового помічника. Також, розумний будинок знижує витрати на електроенергію та інші ресурси завдяки автоматизованому контролю над системами будинку.

Технологія розумного будинку передбачає використання різноманітних систем, що дозволяють автоматизувати керування різними пристроями та електроприладами, які використовуються в будинку.

Пристрої розумного будинку взаємоз’єднуються в мережу та контролюються спеціалізованим програмним забезпеченням, яке підключається до пристроїв, контролює їх стан і реагує на події. Система виконує багато завдань, що дозволяє власниками будівлі витрачати більше часу на питання, які дійсно важливіші. Система управління не тільки забезпечує комфортність, зберігає повітря свіжим та температуру комфортною, контролює опаленням та вентиляцію, а також дозволяють здійснювати автоматизовані операції управління: освітленням; безпекою; побутовими приборами;в одопостачанням та газопостачанням [4].

Усі перелічені операції, можуть контролюватися, як у будинку або у віддаленому доступі (рис.1.1). Наприклад, існує можливість відкривати та

закривати двері та вікна будинка або вимикати всі вогні однією кнопкою у вашій спальні. Прокидатися під улюблені мелодії, та з кип'яченою водою у чайнику для готування кави, або чаю. Коли ви виїжджаєте з гаражу, брама відчиняється одною голосовою командою. Після виїзду, та промови голосової команди на закриття дверей, дверні замки автоматично зачиняються та система активується система безпеки. Розумний дім може запускати полив газону, відповідно до прогнозу погоди з Інтернету [5].

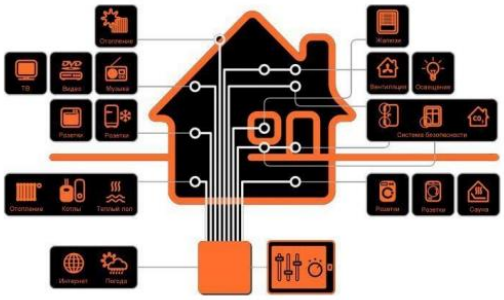


Рисунок 1.1 – Приклад системи «Розумний дім»

Розумні будинки можуть мати бездротові або дротові системи - або обидві. Бездротові системи простіше встановлювати. Встановлення бездротової системи домашньої автоматизації з такими функціями, як розумне освітлення, клімат-контроль і безпека, може коштувати кілька тисяч доларів, що робить її дуже вигідною.

Недоліком бездротових систем є те, що вам, швидше за все, знадобиться потужне покриття Wi-Fi і широкосмуговий зв'язок у всьому будинку. Це може вимагати від вас інвестицій у ретранслятори або дротові бездротові точки доступу. Бездротові системи розумного будинку, як правило, більше підходять для невеликих існуючих будинків або орендованої нерухомості через їх менші розміри.

Дротові системи, з іншого боку, вважаються більш надійними і їх, як правило, складніше зламати. Дротова система може збільшити вартість перепродажу будинку. Крім того, дротові системи розумного будинку можна легко масштабувати, тому вони часто є методом за замовчуванням при проектуванні нової будівлі або виконанні капітального ремонту.

Є один недолік - це досить дорого. Встановлення розкішної та дротової розумної системи може коштувати домовласникам десятки тисяч доларів. Крім того, необхідно мати місце для мережевого обладнання, включаючи кабелі Ethernet [1].

У бездротовій системі, пристрої в будинку можуть взаємодіяти між собою через бездротовий зв'язок, такий як Wi-Fi, Bluetooth або ZigBee, і керуватися зі смартфона, планшета або комп'ютера.

Ці види зв’язку мають свої плюси та мінуси:

Wi-Fi є популярною технологією для пристроїв розумного будинку, оскільки вона широко поширена та легко налаштовується. Вона також може підтримувати додатки з високою пропускною здатністю, такі як потокове відео. Однак пристрої Wi-Fi можуть бути дорогими та споживати багато енергії.

Bluetooth широко поширена та легко налаштовується. Вона також має низьке споживання енергії та може використовуватися для комунікації на короткій відстані між пристроями. Однак Bluetooth має обмежений радіус дії та може підтримувати лише невелику кількість пристроїв.

ZigBee призначена спеціально для застосувань з низьким споживанням енергії та пропускною здатністю, таких як автоматизація розумного будинку. Вона може підтримувати велику кількість пристроїв та має більший радіус дії, ніж Bluetooth. Однак пристрої ZigBee можуть бути дорогими та потребують хабу для підключення до Інтернету.

Загалом Wi-Fi найбільш підходить для додатків з високою пропускною здатністю, таких як потокове відео; Bluetooth найбільше підходить для комунікації на короткому радіусі дії між пристроями; а ZigBee найбільш підходить для застосувань з низькою потужністю та пропускною здатністю, таких як автоматизація розумного будинку [5,6].

Що стосується дротових систем, існують різні види, такі як RS-485, KNX, Modbus та BACnet.

Кожна з цих систем має свої переваги та недоліки. Наприклад, RS-485 є дешевшою та більш простою у встановленні системою, але вона не підтримує

багато функцій. KNX є більш гнучкою та масштабованою системою, але вона дорожча та складніша у встановленні.

Незважаючи на те, що ця ідея звучить дуже технічно та футуристично, справжній розумний дім інтуїтивно зрозумілий і простий у використанні, так що кожен, хто живе у родині, має змогу використовувати його можливості на всі сто відсотків. У майбутньому розумні будинки, швидше за все, стане такими ж поширеними, як пральні машини.

На сьогоднішній день кожен другий новий будинок в США має вбудовану систему [5]. Голосові помічники, такі як Amazon Alexa та Google Home [1], як і раніше, відчуваються нам дивними, але вони популярні в основному через те, наскільки зручні та прості у використанні. Саме з цієї причини вони є однією з перших розумних рішень, які споживачі додають до своїх будинків. Виробники сучасних пристроїв для домашньої автоматизації знають це і швидко беруть на борт, щоб їх електронні та освітлювальні прилади, могли контролюватися за допомоги голосових помічників.

Система голосового управління не залежить від необхідності встановлювати бездротову або дротову систему. Її можна інтегрувати в обидві системи. Але реалізовуватись система голосового управління розумним будинком буде через беспровідну систему на основі Wi-Fi технології, це оптимальне рішення. Реалізація через беспровідну систему на основі Wi-Fi вирішить багато проблем, таких як висока ціна провідних систем, та необхідність встановлювати додаткові пристрої.

Остаточне рішення буде реалізоване у вигляді програмної системи, котра буде виконувати роль голосового асистента, тобто приймати голосові команди користувача, та відповідати виконанням необхідної функції. Для початку користувачеві буде потрібно пробудити систему спеціальним словом чи фразою, наприклад – «Привіт, Будинок». Після цього система буде готова приймати основну команду від користувача, наприклад: «Увімкни кондиціонер», «Встанови 25 градусів у кондиціонера», «Переключи режим на обігріваючий», «Закрий вікна у залі», «Вимкни світло у спальні» і так далі.

На сучасному ринку, вибір систем голосового управління розумним будинком доволі великий, але лідерами залишаються такі варіанти як Amazon Alexa, Google Assistant та Apple Siri, розглянемо кожну з систем:

Amazon Alexa. Amazon Alexa, також відома як просто Alexa. Це технологія голосового асистента, яка базується на синтезаторі мовлення Ivona, придбаному Amazon у 2013 році. Для початку вона була використана у розумній колонці Amazon Echo (рис. 1.2) та інших колонках, створених Amazon. Вона може керувати кількома розумними пристроями, використовуючи себе для автоматизації систем будинку. Можливості Alexa можна розширити встановлюючи спеціальні «навички» (або додатки). Для виконання таких завдань Alexa використовує автоматичне розпізнавання мови, обробку природної мови та інші форми слабкого штучного інтелекту.

Рисунок 1.2 - Зображення Amazon Echo Show

Зазвичай пристрої з Alexa можуть бути активовані спеціальним wake-словом, наприклад: ‘Amazon’, ‘Alexa wake’ і так далі. Деякі пристрої активуються натисканням кнопки у спеціальному застосунку, наприклад у мобільному додатку на Android чи Ios.

У січні 2019 року команда розробників пристроїв Amazon оголосила, що вони продали понад 100 мільйонів пристроїв з підтримкою Alexa [2].

Функція для домашньої автоматизації була запущена 8 квітня 2015 року. Для створення власних навичок для керування розумним будинком використовується Alexa Skills Kit. Ці навички, розробляються третіми сторонами та після публікації стають доступні для всіх пристроїв підтримуючих Alexa.

Голос Alexa генерується штучною нейронною мережею з довгою короткочасною пам’яттю.

Переваги: має велику кількість підтримуваних пристроїв; забезпечує широкий спектр функцій для розумного будинку; має хорошу систему розпізнавання голосу.

Недоліки: інтерфейс може бути доволі важким для користувача що не звикли до Amazon Echo або подібних систем; amazon збирає деякі дані про користувача, тому приватність може бути проблемою.

Google Assitant. Google Assistant – це віртуальний асистент від компанії Google. В основному доступний на мобільних пристроях і пристроях домашньої автоматизації. Google Assitant вперше з'явився як частина додатку Allo, та колонки Google Home (рис.1.3). Після цього він був розгорнутий на інших пристроях Android починаючи з лютого 2017 року. В травні 2017 року він був випущений як окремий додаток для операційної системи IOS. Зараз Асистент підтримує багату кількість пристроїв, такі як автомобілі, розумні домашні прилади і так далі. Асистент може відповідати на запитання, планувати події, регулювати налаштування обладнання на пристрої, грати в ігри і таке інше.

У 2020 році Google Assistant вже був доступний на більш ніж 1 мільярді пристроїв, у більш ніж 90 країнах і більш ніж 30 мовами [3].

Рисунок 1.3 – Зображення Google Home

Переваги: інтегрован з більшістю пристроїв Google та здатен забезпечувати широкий спектр функцій для розумного будинку; перевірка приватності забезпечена на більш високому рівні ніж у Amazon Alexa.

Недоліки: має регіональні обмеження деяких пристроїв, популярних в різних регіонах; не має таку кількість функцій які забезпечує Amazon Alexa.

Apple Siri. Siri – це віртуальний помічник, який є частиною операційних систем IOS, iPadOS, watchOS, macOS і так далі, компанії Apple Inc. Він використовує голосові запити, управління жестами, відстеження фокусу та природно-мовний інтерфейс користувача, щоб відповідати на запитання. Також

по мірі використання вона адаптується до індивідуальних особливостей користувача враховуючи інтереси, мовні особливості, вподобання [6].

Siri була випущена як додаток для IOS у лютому 2010 року. Через два місяці Apple придбала її, та інтегрувала до Iphone 4S. З тих пір Siri – невід'ємна частина продуктів Apple таких як iPhone, iPad, iPod, Touch і подібних.

Для того щоб використовувати Siri для керування розумним будинком, необхідно підключити розумні пристрої до платформи HomeKit в Apple. HomeKit - це фреймворк для розумних будинків, створений компанією Apple, який дозволяє різним розумним пристроям співпрацювати між собою. Використовуючи систему HomePod можна дуже зручно керувати різними пристроями Apple через HomeKit.

Рисунок 1.4 – Зображення Apple HomePod

Переваги: має глибоку інтеграцію з пристроями Apple, та може бути доволі зручною для постійних користувачів продукції Apple; забезпечує стійкий захист персональних даних користувача.

Недоліки:

– має сильно обмежену підтримку пристроїв що не являються продуктами Apple, на відміну від Amazon Alexa та Google Assistant

– може бути менш функціональною порівняно з Alexa, та Google Assistant.

Для взаємодії Siri і «Розумного будинку» компанія Apple створила проект HomeKit. Дана технологія має такі можливості:

– Siri може регулювати роботу аксесуарів homekit, наприклад світлових приладів або термостата;

– Siri може регулювати роботу аксесуарів homekit, наприклад світлових приладів або термостата;

– Siri знає, які аксесуари homekit налаштовані в програмі «Дім», а також їх статус.

– Siri ідентифікує аксесуари за їхніми іменами, розташування і, за іншими відомостями, зазначеними для них в програмі «Дім». Якщо налаштований HomePod, Apple TV або iPad в якості домашнього центру, то можна використовувати Siri для управління будинком в будь-якому місці;

– якщо аксесуари організовані по кімнатах або зонам, можна управляти областями будинку однією командою;

– при використанні HomePod є можливість управляти всіма аксесуарами HomeKit в кімнаті, де знаходиться цей пристрій, однією командою сценарії дозволяють управляти декількома аксесуарами одночасно.

Недоліки:

– необхідно використовувати тільки пристрої від компанії Apple.

– для роботи можливо використовувати тільки сертифіковані компанією Apple компоненти для роботи «Розумного будинку».

Можливим рішенням реалізації голосового інтерфейсу всередині приміщення могло б бути розміщення мікрофонів і джерел звуку в різних точках приміщення, таким чином, щоб користувач міг у будь-який момент вимовити команду і почути відповідь. Однак спроби реалізувати такі системи зіткнулися з тим, що було потрібно велику кількість чутливих мікрофонів, тому система ставала технічно складної і дорогої.

Загалом, всі перелічені системи голосового управління розумним будинком, дуже якісні та технологічно просунуті. Вони виконують багатий перелік функій, та сильно облегшують керування розумним будинком. Але всі проблеми та недоліки витікають з проблем приватності та суміності. Компанії перелічених продуктів збирають інформацію про користувачів, для когось цей факт буде проблемою. Також ці продукти гарно і глибоко інтегровані з іншими продуктами, що випускаються родинними компаніями. Тобто, Apple сильно залежить від продуктів Apple, та погано інтегрується з іншими продуктами, від інших

компаній. Теж саме можна сказати про Amazon Alexa та Google Assistant, хоча в них ступінь інтеграції з іншими пристроями набагато вища.

Тому для забезпечення сильної сумісності з існуючою системою розумного будинку, та вже встановленими пристроями, найкращим варіантом буде виглядати створення власної системи голосового управління розумним будинком. Завдяки цьому можна уникнути проблем приватності та сумісності. Тому що система буде створюватись конкретно під існуючий розумний будинок, та буде повністю сумісна з усіма необхідними пристроями. Також така система буде доступнішою і дешевшою, через відсутність регіональних обмежень і подібних проблем.

Управління голосом системами розумного будинку, сьогодні вже не мода, а підхід до вирішення складних завдань на сучасному рівні. Ця функція була додана, щоб робота такого будинку була максимально продуктивною.

1.2 Формування кінцевого завдання на розробку

Враховуючи результати аналітичного дослідження, варто визначитися більш детально зі структурою системи голосового управління, яка має на меті полегшити управління системами розумного будинку. А саме визначимося з програмною частиною, яку маємо на меті реалізувати у вигляді додатку на операційній системі Windows 10.

Тепер можна визначитись з функціями додатку.

До основних функцій можна віднести:

– Активаційна фраза.

– Обробка інформації з мікрофону для визначення потрібної команди.

– Надсилання визначеної команди на хаб розумного будинку через мережу Wi-Fi.

– Відповідь до користувача з результатом роботи програми (у вигляді графічного повідомлення, або голосового повідомлення).

– Збереження даних на пристрої (необхідні команди для розумного будинку).

Оглянувши функції додатку можна створити архітектуру додатку:

– Інтерфейс користувача.

– Система обробки інформації для визначення необхідної команди.

– Система для відправлення визначеної команди до хабу розумного будинку, та отримання відповіді від розумного будинку.

– Система відповіді до користувача з результатом роботи додатку.

Оскільки додаток буде створено на операційній системі Windows 10. Для розробки була обрана мова програмування - Python.

Python – це високорівнева мова програмування загального призначення з динамічною суворою типізацією та автоматичним керуванням пам'яттю, орієнтована на підвищення продуктивності розробника, читабельності коду та його якості, а також на забезпечення переносимості написаних нею програм [15].

Користувальницький інтерфес. Це інтерфейс завдяки якому користувач буде взаємодіяти з додатком. Користувальницький інтерфейс повинен мати зрозумілий та лаконічний дизайн, щоб користувач не мав проблем працюючи з додатком використовуючи інтерфейс. На мові програмування Python існує багато бібліотек для реалізації графічного інтерфейсу. Найпопулярнішими вважаються Tkinter, PyQt, wxPython, Kivy. Розглянемо ці бібліотеки більш детально:

Tkinter - це де-факто стандартний пакет графічного інтерфейсу користувача (GUI) для Python. Це тонкий об'єктно-орієнтований шар поверх Tcl/Tk [10]. Це доволі проста у використанні бібліотека, що включає в себе необхідні стандартні графічні елементи для реалізації графічного інтерфейсу.

PyQt - це набір прив'язок Python для фреймворку додатків Qt від The Qt Company, який працює на всіх платформах, що підтримуються Qt, включаючи Windows, macOS, Linux, iOS та Android. PyQt6 підтримує Qt v6, PyQt5 підтримує Qt v5 та PyQt4 підтримує Qt v4. Прив'язки реалізовані у вигляді набору модулів Python і містять понад 1 000 класів. PyQt4 та Qt v4 більше не підтримуються і нових випусків не буде [11].

Це доволі потужна бібліотека що має велику кількість можливостей та застосувань, включаючи створенния графічних інтерфейсів. Також має власну середу для розробки графічних інтерфейсів Qt Designer. Qt Designer - це

інструмент Qt, який надає вам користувальницький інтерфейс "що бачиш, те й отримуєш" (WYSIWYG) для продуктивного та ефективного створення графічних інтерфейсів для ваших PyQt додатків [12]. Має велике ком'юніті, та безліч прикладів.

wxPython - це обгортка для крос-платформного API графічного інтерфейсу (часто званого "інструментарієм") wxWidgets (який написаний на C++) для мови програмування Python. Це одна з альтернатив Tkinter. Реалізовано як модуль розширення Python (нативний код) [13]. Доволі проста у використані бібліотека, для простих та стандартних інтерфейсів, мало розповсюджена.

Kivy - це бібліотека програмного забезпечення з відкритим вихідним кодом для швидкої розробки додатків з новими користувацькими інтерфейсами, наприклад, мультисенсорних додатків [14]. Доволі потужна бібліотека, дозволяє створювати кросплатформені додатки для таких платформ як Windows, IOS, Linux, Android. Має гарну документацію та непогане ком'юніті. Kivy гарно підходить для створення мультисенсорних додатків, і доволі легка у використанні та освоєнні.

Отже, з розглянутих бібліотек для створення графічного інтерфейсу можна виділити PyQt та Kivy. Тому що Tkinter та wxPython - це доволі примітивні бібліотеки, що не мають такого потенціалу та функціоналу як Kivy та PyQt, також для них немає такої великої кількості прикладів та підтримки як у Kivy та PyQt. Тому вибір біблотеки для створення користувальницького графічного інтерфейсу звужується до Kivy та PyQt. А саме була обрана бібліотека PyQt, тому що вона має дуже гарне ком'юніті, та безліч прикладів створення графічних інтерфейсів, саме ця бібліотека дозволить створити простий та лаконічний графічний інтерфейс для системи управління розумним будинком.

Система обробки інформації з мікрофона. Ця система має обробляти отриману інформацію, та перетворювати її на текст. Для цієї задачі зазвичай використовуються спеціально навчені нейромережі. Нейромережа це математична модель котра складається з простих обчислювальних блоків (нейронів). Для створення власної системи розпізнавання мови потрібно дуже

багато ресурсів та обчислювальних потужностей, також необхідна велика виборка даних для ефективного навчання нейромережі. Тому створення власної системи не має значного сенсу, великі компанії вже створили продукти з необхідним функціоналом, які вже працюють дуже точно та швидко, завдяки використанні багатої кількості обчислювальних потужностей, та можливості роботи з великою кількістю аудіо-даних. У python є модуль SpeechRecognition що дозволяє використовувати багато сервісів для розпізнавання мови, наприклад Google Speech Recognition, Google Cloud Speech Recognition, Sphinx/PocketSphinx, Vosk API, OpenAI whisper.

Розглянемо ці сервіси більш детально:

Google Speech Recognition. Використовує онлайн сервіс від google для розпізнавання мови. Підтримує більше 100 мов. Безкоштовний для користування. Має доволі високу швидкість розпізнавання мови, та гарну точність. За замовчуванням використовує стандартний API ключ і не потребує вводу нового API ключа.

Google Cloud Speech Recognition. Сучасна версія Google Speech Recognition. Заснована на останніх розробках в галузі розпізнавання мови. Використовує найсучасніші алгоритми нейронних мереж глибокого навчання Google для автоматичного розпізнання мови. Підтримує 125 різних мов. Має високу швидкість і точність розпізнавання мови. Має гарну систему адаптації мовлення, для налаштування розпізнавання мовлення длля транскрибування специфічних термінів і рідкісних слів або фраз. Також підтримує багатоканальне розпізнавання мови, наприклад під час відеоконференцій. Дозволяє незадовільний контент, ненормативну лексику, непрофесійний контент і так далі. Безкоштовно можна використовувати до 60 хвилин на місяць. Якщо використовується більше 60 хвилин на місяць, стягується плата у розмірі 0.016-0.078(залежить від моделі) за додаткову хвилину аудіо. Також ціна може змінюватися залежно від довжини та кількості надісланих аудіоданих, від кількості витрачених каналів і так далі.[16]

Sphinx/PocketSphinx. CMUSphinx об’єднує понад 20 років досліджень CMU. Сучасні алгоритми розпізнавання мови для ефективного розпізнавання мови.

Інструменти CMUSphinx розроблені спеціально для платформ з низькими ресурсами. Гнучкий дизайн. Зосередженість на розробці практичних додатків (не дослідження). Підтримує декілька мов: американська англійська, британьска англійська, французька, китайська, німецька, голандська. Російська та можливість створювати моделі для інших мов [17]. Працює в офлайні, для використання потребує встановлення спеціально навчених готових математичних моделей для розпізнавання мови. Має дуже велику гнучкість, відкритий код, та безкоштовне використання. Для роботи не потребує підключення до інтернету. Оптимізована для простих систем. Але не має таку високу швидкість і точність розпізнавання як сервіси від компанії Google.

Vosk API. Vosk - це офлайн-інструментарій для розпізнавання мови з відкритим вихідним кодом. Він дозволяє розпізнавати мову для 20+ мов і діалектів - англійської, індійської англійської, німецької, французької, іспанської, португальської, китайської, російської, турецької, в'єтнамської, італійської, голландської, каталонської, арабської, грецької, фарсі, філіппінської, української, казахської, шведської, японської, есперанто, хінді, чеської, польської. І так далі.

Моделі Vosk мають невеликий розмір (50 Мб), але забезпечують безперервну транскрипцію великого словника, реакцію з нульовою затримкою з потоковим API, словниковий запас, що реконфігурується, та ідентифікацію диктора.

Розпізнавання мови реалізовано для різних мов програмування, таких як Python, Java, Node.JS, C#, C++, Rust, Go та інших.

Vosk забезпечує розпізнавання мови для чат-ботів, розумних домашніх пристроїв, віртуальних асистентів. Він також може створювати субтитри для фільмів, транскрипцію для лекцій та інтерв'ю.

Vosk масштабується від невеликих пристроїв, таких як Raspberry Pi або Android-смартфон, до великих кластерів [18].

OpenAI Whisper. Whisper - це система автоматичного розпізнавання мови (ASR), навчена на 680 000 годин багатомовних і багатозадачних даних, зібраних з Інтернету. Використання такого великого та різноманітного набору даних

призводить до покращення стійкості до акцентів, фонового шуму та технічної мови. Крім того, це дозволяє транскрибувати дані кількома мовами, а також перекладати їх з цих мов на англійську. Моделі та код виведення у відкритому доступі. Архітектура Whisper - це простий наскрізний підхід, реалізований як кодер-декодер-трансформатор. Вхідний аудіосигнал розбивається на 30-секундні фрагменти, перетворюється на log-mel спектрограму, а потім передається на кодер. Декодер навчається передбачати відповідний текстовий підпис, змішаний зі спеціальними маркерами, які спрямовують єдину модель на виконання таких завдань, як ідентифікація мови, мітки часу на рівні фраз, багатомовна транскрипція мовлення та переклад мовлення на англійську мову [19]. Працює в офлайні, для роботи необхідно використовувати готову модель, котра буде встановлюватися на пристрій. Безкоштовне використання.

Підсумовуючи, всі описані системи вважаються ефективними, швидкими та точними. Вибір необхідної системи буде базуватися на ряді фундаментальних властивостей, а саме робота в онлайні/офлайні, платне/безоплатне використання, швидкість та точність роботи. З онлайн сервісів, можна виділити Google Speech Recognition, він працює з підключенням до інтернету, безкоштовний, працює швидко та точно. З офлайн сервісів можна виділити Vosk та Whisper, також дуже потужні та ефективні системи розпізнавання голосу, вони працюють в офлайні але мають суттєвий мінус, вони працюють не так швидко як онлайн системі, і потребують більше ресурсів комп'ютера, оскільки офлайн моделі повинні бути запущені та завантажені в операційну пам'ять, також обчислення будуть виконувася з використанням користувальницьких апаратних засобів. Тому для полегшення переносимості і реалізації було обрано онлайн сервіс розпізнавання голосу Google Speech Recognition він працює швидко, повністю безкоштовний, має підтримку багатьох мов, працює в онлайні та не потребує багато ресурсів комп'ютера.

Система для відправлення визначеної команди до хабу розумного будинку, та отримання відповіді від розумного будинку. Для реалізації системи відправлення визначеної команди до хабу розумного будинку потрібно

визначитись для якого протоколу буде створюватись система. Найпопулярнішими зараз можна вважати такі протоколи як HTTP, MQTT, AMQP. Коротко розглянемо ці протоколи:

MQTT. MQTT (Message Queuing Telemetry Transport) - це легкий протокол обміну повідомленнями, що використовується для зв'язку між пристроями з обмеженими ресурсами, такими як IoT-пристрої.

HTTP. Протокол HTTP (Hypertext Transfer Protocol) використовується для передачі даних через Інтернет. Він широко використовується для передачі даних з різних джерел, таких як вебсайти, мобільні додатки та інші.

AMQP. Протокол AMQP (Advanced Message Queuing Protocol) - це протокол, який використовується для передачі повідомлень між різними системами. Він широко використовується для передачі даних в хмарних сервісах та інших розподілених системах.

Отже, MQTT гарно підходе для передачі невеликої кількості інформації, для пристроїв з обмеженими ресурсами таких як датчики. HTTP гарно підходе для передачі даних між сайтами, мобільними додатками і так далі. AMQP більше підходе для роботи з хмарними сервісами.

Тому для реалізації системи відпарвлення визначеної команди на хаб розумного будинку, було обрано протокол HTTP. Так як реалізація системи буде у вигляді додатку на Windows 10, і в такому разі система не буде сильно обмежена по ресурсам, то було обрано саме цей протокол.

Для роботи з протоколом HTTP на мові Python існує три основні, та найуживаніші бібліотеки, а саме Requests, urllib, httplib. Розглянемо ці варіанти:

Requests. Requests дуже популярна та проста у вористанні бібліотека для роботи з HTTP протоколом на Python. Вона дозволяє легко виконувати HTTP запити, отримувати та обробляти вдповіді сервера, встановлювати заголовки запиту та багато іншого. Простий та зрозумілий інтерфейс, що дозволяє легко виконувати HTTP запити та обробляти відповіді сервера. Підтримує різні методи запитів, включаючи GET, POST, PUT, DELETE та інші. Не входить до стандартної

бібліотеки Python, тому потребує додаткового встановлення. Може бути повільнішим за інші бібліотеки, такі як httplib.

urllib. Стандартна бібліотека Python для роботи з HTTP протоколом. Вона містить модулі для виконання HTTP запитів, обробки відповідей сервера, роботи з URL-адресами та багато іншого. Входить до стандартної бібліотеки Python, тому не потребує додаткового встановлення. Містить різні модулі для роботи з HTTP протоколом, включаючи urllib.request, urllib.parse та інші. Може бути складним у використанні, особливо для початківців. Не має такого прого та зрозумілого інтерфейсу, як requests.

httplib. Ще одна стандартна бібліотека Python для роботи з HTTP протоколом. Вона дозволяє виконувати HTTP запити, обробляти відповіді сервера, встановлювати заголовки запиту та бато іншого. Входить до стандартної бібліотеки Python, тому не потребує додаткового встановлення. Містить різні класи та методи для роботи з HTTP протоколом, включаючи HTTPConnection, HTTPResponse та інші.-Може бути складним у використанні, особливо для початківців. Не має такого простого та зрозумілого інтерфейсу, як requests.

Основним мінусом Requests являється необхідність встановлення бібліотеки оскільки вона не входить до стандартного пакету Python. Але для реалізації системи вже було обрано чимало бібліотек що потребують встановлення. Тому цей мінус немає великого значення. Одже для реалізації системи відправлення визначеної команди до хабу розумного будинку, та отримання відповіді від розумного будинку була обрана бібліотека Requests, за її функціональність та значну простоту використання.

Система відповіді до користувача з результатом роботи додатку. Ця система може бути реалізована по різному, сповіщувати користувача щодо результатів роботи додатку можна виводом текстової або графічної інформації на екран пристрою, або синтезувати голос та відповісти користувачу саме голосом. Другий варіант виглядає набагато дружелюбніше для користувача, і при цьому набагато приємніше отримати відповідь у вигляді подібного до людського голосу. Для реалізації відповіді до користувача з результатом роботи додатку у вигляді

синтезу голосу існує кілька популярних бібліотек на мові Python, а саме pyttsx3, gTTS, Mozilla TTS. Розглянемо ці варіанти більш детально:

pyttsx3. pyttsx3 - це бібліотека перетворення тексту в мовлення мовою Python. На відміну від альтернативних бібліотек, вона працює у офлайн режимі і сумісна з Python 2 і 3 [20]. Офлайн робота, pyttsx3 працює без підключення до Інтернету, що забезпечує швидкість та конфіденційність. Незалежність від платформи, pyttsx3 підтримує Windows, macOS та Linux. Обмежений вибір голосів, pyttsx3 використовує системні голоси, тому вибір може бути обмеженим. Нижча якість голосу, порівняно з іншими бібліотеками, якість голосу може бути нижчою.

gTTS. gTTS (Google Text-to-Speech), бібліотека Python та інструмент CLI для взаємодії з API перетворення тексту в мовлення Google Translate. Записує вимовлені mp3-дані у файл, файловий об'єкт (байтовий рядок) для подальших маніпуляцій зі звуком або stdout [21]. Висока якість голосу, gTTS використовує передові технології Google для синтезу голосу, що забезпечує високу якість звучання. Підтримує багато мов, що робить його корисним для різних застосунків.

Вимагає підключення до Інтернету для роботи, що може бути обмеженням у деяких випадках. Обмеження API: оскільки gTTS використовує API Google, можуть виникнути обмеження щодо кількості запитів та швидкості.

TTS. TTS - це бібліотека для вдосконаленого перетворення тексту в мовлення. Вона заснована на останніх дослідженнях і була розроблена для досягнення найкращого компромісу між легкістю в навчанні, швидкістю та якістю. TTS постачається з попередньо навченими моделями, інструментами для вимірювання якості наборів даних і вже використовується на 20+ мовах для продуктів і дослідницьких проектів [22]. TTS є відкритим проектом, що дозволяє спільноті розробників вдосконалювати та розширювати його можливості. TTS дозволяє використовувати різні моделі та архітектури для синтезу голосу, що забезпечує гнучкість у виборі якості та швидкості. Складність налаштування порівняно з іншими бібліотеками, TTS може бути складнішим для налаштування та використання. Деякі моделі та архітектури, що використовуються в TTS,

можуть вимагати значних обчислювальних ресурсів, що може бути проблемою для слабких систем.

Нам потрібна невеличка бібліотека для простої задачі, з синтезу голосового повідомлення користувачу. Для цієї задачі найбільше підходе gTTS, вона дуже проста у використанні та має все необхідне для реалізації системи, також вона безкоштовна, працює доволі швидко. Тому для реалізації системи відповіді до користувача була обрана бібілотека синтезу голосу gTTS.

Отже, програмна система голосового управління розумним будинком буде створена у вигляді додатку на операційній системі Windows 10. Для розробки буде використана мова програмування Python.

Додаток буде складатися з таких основних систем:

– Інтерфейс користувача.

– Система обробки інформації для визначення необхідної команди.

– Система для відправлення визначеної команди до хабу розумного будинку, та отримання відповіді від розумного будинку.

– Система відповіді до користувача з результатом роботи додатку.

Для реалізації кожної з систем були обрані необхідні бібліотеки на мові програмування Python, а саме:

– Інтерфейс користувача. *Бібліотека для реалізації графічного інтерфейсу PyQt.*

– Система обробки інформації з мікрофону для визначення необхідної команди. *Бібілотека розпізнавання голосу Speech Recognition (сервіс Google Speech Recognition).*

– Система для відправлення визначеної команди до хабу розумного будинку, та отримання відповіді від розумного будинку. *Бібілотека для роботи з протоколом HTTP, Requests.*

– Система відповіді до користувача з результатом роботи додатку. *Бібліотека для синтезу голосу gTTS.*

1.3 Розробка програми

Після формування кінцевого завдання на розробку перейдемо до реалізації системи. Для розробки був обран текстовий редактор VS Code від Microsoft. Для початку встановимо мову програмування Python, перейдемо до офіційного сайту python.org/downloads, та завантажимо файл встановлення. Після цього згідно інструкції встановимо мову програмування Python. Створимо папку проекта VoiceAssistant. Через VS Code відкриваємо цю папку, та додаємо головний python файл проекту voice_assistant_main.py.

Тепер нам необхідно встановити всі необхідні залежності, бібліотеки які ми будемо використовувати. Переходимо до вбудованого терміналу у Vs Code, та через команду пакетного менеджера pip install встановлюємо всі бібліотеки.

pip install SpeechRecognition, requests, gtts, pydub, thefuzz, PyQt6, Flask

Для початку імпортуємо до нашої програми всі необхідні бібліотеки з якими ми будемо працювати:

```
1   import sys
2   import speech_recognition as sr
3   import requests
4
5   from time import sleep
6   from gtts import gTTS
7   from pydub import AudioSegment
8   from pydub.playback import play
9   from needs import uk_possible_commands, en_possible_commands, url
10  from thefuzz import fuzz
11  from PyQt6.QtCore import Qt, QSize, QThread, pyqtSignal, pyqtSlot
12  from PyQt6.QtWidgets import (
13          QApplication, QMainWindow, QPushButton, QGridLayout,
14          QSpacerItem, QSizePolicy, QWidget, QLabel, QMessageBox,
15          QDialog, QComboBox
16      )
17  from PyQt6.QtGui import QIcon
```

Рисунок 1.5 - Імпорти

Коротко про необхідні нам бібліотеки:

sys – дає інформацію щодо запущеного інтерпретатору Python.

speech_recognition – система для розпізнавання мови через ряд сервісів.

requests – бібліотека для роботи з HTTP протоколом

time – стандартна бібліотека для роботи з часом.

gtts – бібліотека для синтезу голосу через сервіс Google.

pydub – надає інструменти для роботи з аудіофайлами.

needs – створений модуль для зберігання інформації щодо даних користувача.

thefuzz – надає різні алгоритми для обчислення подібності між рядками.

Далі імпортуємо всі необхідні інструменти з бібліотеки PyQt6.

Клас MainWindow

Можна розпочати створення системи. Створимо клас головного вікна програми:

```
289  # Класс головного вікна програми, що наслідується від класу QMainWindow
290  # бібліотеки PyQt6
291  class MainWindow(QMainWindow):
292
293      # Ініціілізатор/конструктор, створює необхідні віджети та з'єднує їх між собою
294      def __init__(self):
295          super().__init__()
```

Рисунок 1.6 - Створення класу MainWindow

У python клас являє собою шаблон для створення об'єктів, що об'єднують дані (атрибути) і функціональність (методи) в одній сутності.

__init__ це спеціальний метод класу, який виконує роль конструктора, і викликається після створення екземпляру класу (конкретний об’єкт, створений на основі класа). Реалізуємо створення графічних елементів нашого додатку.

```
297          # Зберігаємо поточне налаштування мови у змінну
298          self.language = language_func(change=False)
299
300          # Встановлення назви головного вікна програми, розмірів
301          self.setWindowTitle("Home Voice Assistant")
302          self.setFixedSize(QSize(300, 180))
303
304          # Залежно від налаштованої мови створюємо основну кнопку та основний напис
305          if self.language == 'en':
306              self.mainbutton = QPushButton("Activate")
307              self.info_label = QLabel('Sleeping...')
308          else:
309              self.mainbutton = QPushButton("Активувати")
310              self.info_label = QLabel('Очікування...')
```

Рисунок 1.7 – Створення головної кнопки/надпису

Для початку отримуємо поточне налаштування мови, для того щоб визначити на якій мові створювати текст для наступних елементів, і встановлюємо деякі параметри класу вікна а саме - назву ‘Home Voice Assistant’ та фіксовані розміри 300 на 180 пікселів.

Поточне налаштування мови ми отримуємо з зовнішнього файлу user_language.txt за допомоги функції language_func() яку опишемо пізніше.

Коли ми знаємо яка мова налаштована, почнемо створювати графічні елементи. Додаток буде підтримувати дві мови, українську та англійську. Перед створенням перших графічних елементів перевіряємо чи встановлена англійська мова, якщо так то створюємо об'єкт головної кнопки та головного тексту, саме на англійській мові. Якщо встановлена не англійська, то створюємо ці об'єкти на українській мові (рис. 1.7).

```
312         # Створення параметрів основних параметрів для головної кнопки,
313         # властивість натискання, розміри
314         self.mainbutton.setCheckable(True)
315         self.mainbutton.setFixedSize(QSize(100, 80))
316
317         # З'єднання головної кнопки з функцією яку вона буде виконувати
318         # залежно від стану натиснуто/відпущено
319         self.mainbutton.toggled.connect(self.the_mainbutton_was_checked)
```

Рисунок 1.8 – Налаштування головної кнопки

Додаємо ряд параметрів до головної кнопки (рис. 1.8):

Можливість натискання встановлюємо на True використовуючи метод setCheckable(). Та розмір кнопки 100 на 80 пікселів. Також через метод .connect, з'єднуємо подію зміни стану кнопки (натиснуто/не натиснуто) з функцією яку ця кнопка буде виконувати self.the_mainbutton_was_checked (опишемо пізніше).

```
322         # Створення кнопки налаштувань, передача параметрів, розміри, стиль,
323         # замість тексту використовується картинка шестерні
324         self.settings_button = QPushButton()
325         self.settings_button.setFixedSize(QSize(17, 17))
326         icon = QIcon('settings.png')
327         self.settings_button.setStyleSheet('background-color: transparent; border: none;')
328         self.settings_button.setIcon(icon)
329         self.settings_button.setIconSize(self.settings_button.size())
330
331         # З'єдання кнопки налаштувань з функцією яку вона буде виконувати
332         # залежно від події кліку на кнопку
333         self.settings_button.clicked.connect(self.settings_button_click)
```

Рисунок 1.9 – Створення кнопки налаштувань

Далі створюємо кнопку налаштувань (рис.1.9), і також додаємо ряд параметрів. Встановлюємо розмір 17 на 17 пікселів. Замість тексту додаємо картинку шестерні, картинка була безкоштовна встановлена з інтернету (рис.1.10).

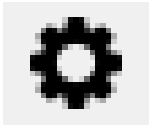


Рисунок 1.10 – Картинка для кнопки налаштувань

Також для нормального відображення картинки був встановлений стиль заднього фону кнопки на прозорий, через метод setStyleSheet(), та вдалені границі кнопки.

Далі з'єднуємо подію натиску на кнопку з фукнцією self.settings_button_click (опишемо пізніше).

Створюємо інформаційну кнопку (рис. 1.11):

```
336         # Створення інформаційної кнопки, передача параметрів, розміри, стиль,
337         # картинка літери 'i'
338         self.info_button = QPushButton()
339         self.info_button.setFixedSize(17, 17)
340         icon = QIcon('info_button_icon.png')
341         self.info_button.setStyleSheet('background-color: transparent; border: none;')
342         self.info_button.setIcon(icon)
343         self.info_button.setIconSize(self.info_button.size())
344
345         # З'єдання інформаційної кнопки з функцією яку вона буде виконувати
346         # залежно від події кліку на кнопку
347         self.info_button.clicked.connect(self.info_button_click)
```

Рисунок 1.11 – Створення інформаційної кнопки

Встановлюємо розміри 17 на 17 пікселів, замість тексту картинку (рис. 1.12), встановлюємо стиль прозорого заднього фону та вдаляємо границі кнопки, з'єднуємо з функцією self.info_button_click().

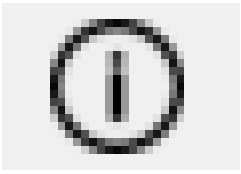


Рисунок 1.12 – Картинка для інформаційної кнопки

Створюємо інформаційний надпис що буде відповідати за відображення поточної встановленої мови на головному вікні додатку, та передаємо кілька параметрів головному надпису додатку, який відображає поточний етап роботи додатку для користувача, встановлюємо розмір 100 на 100, та додаємо властивість переносу тексту по границям встановленого розміру через метод setWordWrap() (рис. 1.13).

```
349         # Створення надпису що відображає поточну мову додатку, передача параметрів
350         self.lang_label = QLabel(self.language)
351
352         # Передача параметрів основному надпису який відображає поточний етап роботи
353         # додатку
354         self.info_label.setFixedSize(100, 100)
355         self.info_label.setWordWrap(True)
```

Рисунок 1.13 – Передача параметрів головному/інформаційному надпису

Створення об'єкту сітчатого макету (рис.1.14), сітчатий макет використовується для розміщення інших віджетів у програмі, віджети розміщуються по коміркам макету, у вигляді таблиці:

```
357         # Створення сітчатого макету, для розміщення усіх інших віджетів
358         self.gridLayout = QGridLayout()
359
360         # Створення віджету заповнювачу, для конфігурації росположення основних віджетів
361         # у сітчатому макеті
362         spacer = QSpacerItem(40, 20, QSizePolicy.Policy.Expanding, QSizePolicy.Policy.Expanding)
363
364         # Додавання всіх віджетів до макету, на необхідні місця
365         self.gridLayout.addWidget(self.mainbutton, 1, 1, 1, 2)
366         self.gridLayout.addWidget(self.settings_button, 0, 4)
367         self.gridLayout.addWidget(self.info_button, 0, 0)
368         self.gridLayout.setAlignment(self.settings_button, Qt.AlignmentFlag.AlignRight)
369         self.gridLayout.addWidget(self.info_label, 1, 4)
370         self.gridLayout.addWidget(self.lang_label, 2, 0)
371
372         # Додавання заовнювачів для конфігурації віджетів у макеті
373         self.gridLayout.addItem(spacer, 0, 0, 3, 1)
374         self.gridLayout.addItem(spacer, 0, 3, 3, 1)
```

Рисунок 1.14 – Створення сітчатого макету

Для більш точного розташування віджетів також використаємо об'єкт заповнювач, QSpacerItem, він використовується для створення об'єму між віджетами у макетах. Потім додаємо всі наші створені віджети до макету, на відповідні координати. Також додаємо заповнювачі, для створення об'єму.

Вставляємо отриманий сітчатий макет у віджет контейнер (рис. 1.15), для розташування його на головному вікні додатку:

```
376         # Створення віджету контейнеру, для сітчатого віджету
377         self.widget = QWidget()
378         self.widget.setLayout(self.gridLayout)
379
380         # Встановлення віджету контейнеру у якості центрального віджету вікна
381         self.setCentralWidget(self.widget)
```

Рисунок 1.15 – Створення віджет контейнеру, для розміщення сітчатого макету

Тепер перейдемо до створення функцій, які будуть виконувати наші кнопки, ці функції будуть створені у вигляді методів класу нашого головного вікна.

```
383     # Функція для основної кнопки, викликається при зміні стану основної кнопки
384     # натиснуто/відпущено
385     def the_mainbutton_was_checked(self, checked):
386
387         # Зміна тексту основної кнопки, та основного надпису залежно
388         # від налаштованої мови
389         if self.language == 'en':
390             if checked:
391                 self.mainbutton.setText('Activated')
392             else:
393                 self.mainbutton.setText('Activate')
394                 self.label_update('Sleeping...')
395         else:
396             if checked:
397                 self.mainbutton.setText('Активовано')
398             else:
399                 self.mainbutton.setText('Активувати')
400                 self.label_update('Очікування...')
401
402         # Передача стану основної кнопки натиснуто/відпущено,
403         # до потіку основної кнопки
404         main_button_thread.update_checked(checked)
```

Рисунок 1.16 – Функція основної кнопки

Створимо функцію для основної кнопки (рис. 1.16), ця функція з'єднана з подією кнопки (зміна стану натиснуто/не натиснуто). Ця функція буде викликатись коли змінюється стан кнопки. Далі йде перевірка на поточне налаштування мови, і на поточний стан кнопки, якщо кнопка натиснута то текст кнопки змінюється на Активовано/Activated, якщо кнопка не натиснута то текст змінюється на Активувати/Activate при цьому також зміниться головний текст щодо поточного етапу роботи програми на Очікування…/Sleeping… Після цієї перевірки стан кнопки передається до іншого потоку виконання програми, де буде виконуватися основна дія головної кнопки (реалізація основної дії була виконана у окремому потоці, щоб не заважати роботі графічного інтерфейсу, буде описано пізніше).

```
408     def info_button_click(self):
409
410         # Формується та виводиться інформаційне повідомлення
411         # у діалогове вікно MessageBox від поточної мови
412         if self.language == 'en':
413             info = 'Activation phrase: Hello home;\n\nAvailable commands:\n'
414             for el in en_possible_commands:
415                 info = info + '- ' + el + ';\n'
416
417             QMessageBox.information(None, 'Information', info)
418         else:
419             info = 'Фраза активації: Привіт будинок;\nДоступні команди:\n'
420             for el in uk_possible_commands:
421                 info = info + '- ' + el + ';\n'
422
423             QMessageBox.information(None, 'Інформація', info)
```

Рисунок 1.17 – Функція інформаційної кнопки

Створимо функцію для інформаційною кнопки (рис.1.17), ця функція з'єднана з подією натиску на кнопку, коли кнопка налаштувань буде натиснута, виконається ця функція. Тут також залежно від встановленої мови ми формуємо інформаційний текст для користувача, а саме інформацію щодо фрази активації, та щодо доступних команд. Після чого виводимо сформовану інформацію через діалогове вікно QMessageBox використовуючи метод information, що виводить саме інформаційне вікно.

```
426     def settings_button_click(self):
427
428         # Створення об'єкту діалогового вікна
429         self.dialog = QDialog()
430
431         # Створення об'єкту списку що випадає
432         dialog_combo_box = QComboBox()
433
434         # Створення надпису, кнопки, та назви діалогового вікна
435         # передача параметрів згідно налаштуванню поточної мови
436         if self.language == 'en':
437             dialog_label = QLabel('Select language:')
438             dialog_button = QPushButton('Ok')
439             self.dialog.setWindowTitle('Settings')
440         else:
441             dialog_label = QLabel('Оберіть мову:')
442             dialog_button = QPushButton('Ок')
443             self.dialog.setWindowTitle('Налаштування')
```

Рисунок 1.18 – Функція для кнопки налаштувань

Функція кнопки налаштувань (рис. 1.18) з'єднана з подією натиску на кнопку налаштувань, і буде викликатися кожен раз під час натиску на кнопку. Для

початку створюємо об’єкт діалогового вікна QDialog(), створюємо об’єкт випадаючого списку QComboBox(). Далі залежно від встановленої мови створюємо надпис на кнопку підтвердження, також встановлюємо назву діалогового вікна.

```
445         # Локальна функція для кнопки діалогового вікна
446         # що виклакається під час натискання на кнопку
447         def d_button_click():
448 
449             # Отримання поточної мови
450             c_lang = language_func(change=False)
451 
452             # Отримання встановленої користувачем мови у ComboBox
453             lan = dialog_combo_box.currentText()
454 
455             # Зміна/налаштування поточної мови, згідно вибору користувача
456             if c_lang == 'en' and lan == 'Українська':
457                 language_func(change=True, lang='uk')
458                 self.dialog.close()
459                 QMessageBox.warning(None, 'Warning', 'To apply the selected settings, restart the application')
460 
461                 self.lang_label.setText('uk')
462                 self.lang_label.setStyleSheet('QLabel { color: red; }' 'QToolTip { color: black; background-color: white; }')
463                 self.lang_label.setToolTip('To apply the language settings, restart the application')
464 
465             elif c_lang == 'uk' and lan == 'English':
466                 language_func(change=True, lang='en')
467                 self.dialog.close()
468                 QMessageBox.warning(None, 'Попередження', 'Для застосування вибраних налаштувать, перезапустіть додаток')
469 
470                 self.lang_label.setText('en')
471                 self.lang_label.setStyleSheet('QLabel { color: red; }' 'QToolTip { color: black; background-color: white; }')
472                 self.lang_label.setToolTip('Для застосування налаштувань мови, перезапустіть додаток')
473             else:
474                 self.dialog.close()
```

Рисунок 1.19 – Функція для кнопки підтвердження діалогового вікна

Створюємо функцію для кнопки підтвердження (рис.1.19) яка буде реагувати на натискання кнопки. У цій функції ми визначаємо поточну мову, та обрану користувачем мову через ComboBox у діалоговому вікні. Потім перевіряємо якщо обрана мова відрізняється від встановленої, то змінюємо встановлену мову на обрану. Закриваємо діалогове вікно, та відображаємо користувачу попередження, що для встановлення обраних налаштувань потрібно перезапустити додаток, після цього інформаційний надпис на головному вікні що відповідає за відображення поточної мови, змінюється на нову встановлену мову і змінює текст на червоний, при наведені користувача на цей надпис, з’явиться невеличка контекстна позначка, що буде також повідомляти за необхідність перезапуску додатку дл встановлення обраних налаштувань.

```
        # З'єдання кнопки діалогового вікна з функцією, передача параметрів
        dialog_button.clicked.connect(d_button_click)
        dialog_button.setFixedSize(60, 20)

        # Додавання об'єктів до списку що випадає в залежності від встановленої мови
        if language_func(change=False) == 'en':
            dialog_combo_box.addItems(['English', 'Українська'])
        else:
            dialog_combo_box.addItems(['Українська', 'English'])

        # Створення сітчатого макету для розміщення інших віджетів на діалоговому вікні
        gridLayout = QGridLayout()

        # Додавання віджетів до сітчатого макету на потрібні позиції
        gridLayout.addWidget(dialog_label, 0, 0)
        gridLayout.addWidget(dialog_combo_box, 0, 1)
        gridLayout.addWidget(dialog_button, 1, 1)

        # Встановлення сітчатого макету з віджетами у якості основного макету діалогового вікна
        self.dialog.setLayout(gridLayout)

        # Відображення діалогового вікна
        self.dialog.show()
```

Рисунок 1.20 – Передача параметрів діалогового вікна

Після цього з'єднуємо функцію натиску на кнопку підтвердження з самою кнопкою. Встановлюємо ряд параметрів, наповнюємо випадаючий список combobox варіантами мови. Створюємо локальний сітчатий макет, для встановлення всіх необхідних віджетів на діалогове вікно, встановлюємо отриманий сітчатий макет у якості макету для діалогового вікна, та відображаємо це вікно методом show() (рис. 1.20).

```
    # Модифікація основної функції наслідуючого класу, для закриття всіх далогових вікон, після
    # закриття програми
    def closeEvent(self, event):
        for widget in QApplication.topLevelWidgets():
            widget.close()

        event.accept()

    # Слот для передачі даних, з інших місць програми, до основного надпису на головному вікні
    @pyqtSlot(str)
    def label_update(self, label):

        # Оновлення головного надпису, згідно поточного етапу роботи потіка основної кнопки
        self.info_label.setText(label)
```

Рисунок 1.21 – Методи closeEvent, label_update

Останні методи які були створені для класу головного вікна, функція closeEvent відповідає за зачинення всіх діалогових вікон, після зачинення самого додатку.

Функція label_update створює слот, для спілкування між екземпляром головного вікна, та іншим поток для головної кнопки, у цій функції буде проходити зміна інформаційного надпису, щодо поточного етапу роботи програми, оскільки основна дія роботи додатку буде проходити у іншому потоці, тому для передачі інформації щодо поточного етапу з іншого потоку, був створений спеціальній слот з використанням pyqtSlot. Який дозволяє спілкуватися між різними потоками, шляхом викликання цього методу, коли був отриман спеціальний сигнал (рис.1.21).

Створення класу окремого потоку для головної кнопки.

```
117  # Клас потіку для основної кнопки, що наслідується від классу QThread бібліотеки pyqt6
118  class MainButtonThread(QThread):
119      finished = pyqtSignal()
120
121      # Ініціалізатор, створює необхідно змінні для екземпляру класа
122      def __init__(self):
123          super().__init__()
124          self.checked = False
125          self.is_running = True
126          self.language = language_func(change=False)
127
128      # Слот для передачі даних зміної checked потіку основної кнопку,
129      # з потіку основного вікна
130      @pyqtSlot(bool)
131      def update_checked(self, checked):
132          self.checked = checked
133
134      #  для передачі сигналу щодо зупинки роботи потіку основної кнопки
135      def stop(self):
136          self.is_running = False
137
```

Рисунок 1.22 – Створення класу потоку для основної кнопки

На початку під час ініціалізації створюємо кілька зміних, стан головної кнопки, стан виконання потоку, та поточна налаштована мова. Також створюємо слот для отримання від екземпляру головного вікна інформації щодо стану кнопки. Та метод для зупинки головного циклу після зупинки потоку (рис.1.22).

```
139      def run(self):
140          while self.is_running:
141              sleep(0.0001)
142
143              # Перевіряємо стан кнопки, та налаштовану мову додатку
144              # якщо встановлена англійська мова, починається виконання
145              # цієї частини коду
146              if self.checked and (self.language == 'en'):
147                  rec = sr.Recognizer()
```

Рисунок 1.23 – Функція з основною логікою роботи потоку

Потім йде функція з основною логікою роботи (рис.1.23). На початку перевіряється чи активний зараз потік, якщо активний то запускається головний цикл, де в нас знаходиться два основних блоки коду. Ці блоки однакові, але один блок працює для налаштування англійської мови, а інший блок для налаштування української мови. Розглянемо більш детально перший блок коду, який виконується для налаштування англійської мови. На початку ми перевіряємо поточну налаштову мову, і якщо вона англійська починаємо виконання цього блоку коду.

Створюємо об'єкт Recognizer бібліотеки SpeechRecognition, для наступного розпізнавання мови.

```
try:
    #створюємо контекстний менеджер для об'єкту мікрофона
    with sr.Microphone() as source:

        # Очікуємо фразу активації, шляхом зчитування аудіопотоку
        # шматками по 3 секунди, зчитування починається тільки при
        # фіксації мови
        window.label_update('Waiting for the activation phrase')
        audio = rec.listen(source, timeout=2, phrase_time_limit=3)
        if self.checked:
            audio = rec.recognize_google(audio, language='en-US')
        else:
            continue
```

Рисунок 1.24 – Прослуховування мікрофону/розпізнавання мови

Далі у обробнику помилок створюємо контекстний менеджер для роботи з мікрофоном системи (рис.1.24). Контекстний менеджер у python дозволяє керувати ресурсами, він автоматично виконує закриття файлу, мережевого з'єднання або у нашому випадку допомагає звільнити ресурси мікрофона, після завершення роботи контекстного менеджеру. Далі через слот label_update екземпляру нашого головного вікна, змінюємо інформаційний текст на головному вікні сигналізуючи про те що програма почала очікування фрази активації від користувача. Після цього викликається метод listen() у об'єкта розпізнавача, якому передається ряд параметрів, а саме джерело аудіо (наш об'єкт мікрофону), обмеження щодо часу очікування, та обмеження щодо довжини записуваного аудіо. Метод listen() починає слухати мікрофон, до тих пір поки не отримає якийсь

звук, або не пройде 2 секунди які ми вказали в аргументі timeout. Після отримання звуку метод буде записувати цей звук поки він не закінчиться, або поки не пройде 3 секунди через наше обмеження у аргументі phrase_time_limit.

Після цього отриманий звук зберігається за ім'ям audio. Потім ми ще раз додатково перевіряємо чи все ще натиснута кнопка, якщо так, то переходимо до розпізнавання мови. Метод recognize_google, об'єкту розпізнавача приймає наше записане аудіо, і аргумент щодо мови яку потрібно розпізнати, передаємо у аргумент 'en-US' (американську англійську). Після чого об'єкт розпізнавача відсилає це аудіо до сервісу розпізнавання мови від google, та повертає розпізнаний текст якщо не виникло помилок.

```
if 'hello home' in audio and self.checked:

    #програємо короткий звук повідомлення (щодо фіксації фрази активації)
    play(AudioSegment.from_mp3('listening_notification.mp3'))

    # Запускаємо цикл в якому будемо отримувати основні команди з очікуванням до 5
    # секунд (якщо після початку очікування пройде 5 секунд, асистент знову почне
    # очікувати фразу активації) у цей проміжок до 5 секунд, можна передавати почергово
    # основні команди, після передачі команди, таймер оновлюється, і у вас є ще 5 секунд
    # для передачі чергової команди
    activate = True
    while activate:
        activate = False
        window.label_update('Waiting for the main command')
        audio = rec.listen(source, timeout=5, phrase_time_limit=5)
        if self.checked:
            audio = rec.recognize_google(audio, language='en-US')
        else:
            continue
```

Рисунок 1.25 – Визначення фрази активаціїі/очікування основної команди

Після першого розпізнавання, ми перевіряємо отриманий текст на наявність фрази активації, якщо ця фраза була виявлена, програється короткий повідомляючий звук, щодо вдалого розпізнання фрази активації (рис. 1.25). Потім запускається цикл з очікуванням основної команди. Циклічна структура дозволяє організувати таку логіку роботи: після отримання фрази активації, програма 5 секунд буде очікувати основну команду, якщо команда не була отримана, програма знову почне вимагати фразу активації. Якщо була отримана команда, то цикл продовжується ще на 5 секунд, і так до тих пір поки команда отримана не буде. Тобто після вдалого розпізнання фрази активації, можна надсилати безліч основних команд почергово, поки не вийде час (5 секунд) після останньої

надісланої команди. Для реалізації такої логіки була обрана циклічна структура. Перед початком циклу ми створюємо змінну (перемикач) для перевірки чи вийшов час. Після цього починаємо цикл у якому перевіряється чи потрібно запитувати основну команду. Якщо визначено що потрібно, то перемикач одразу змінюємо на False (повідомляємо що час вийшов). Змінюємо інформаційний текст сигналізуючи що тепер додаток очікує основну команду. Після цього запускаємо такий самий блок з розпізнавання мови як при очікуванні фрази активації, тільки вказуємо інші аргументи для методу listen, а саме очікування аудіо до 5 секунд, та ліміт на довжину аудіо до 5 секунд.

```
# Визначаємо підходящу команду, з використанням функції command_match
matching_command = command_match(audio, en_possible_commands)

# Якщо команда визначена, відправляємо її до серверу
if matching_command:
    window.label_update(f'Sending <{matching_command}> to the servera')
    response = send_command(matching_command, url, source='Smart home voice control system', language='en')
```

Рисунок 1.26 – Визначення команди

Після розпізнавання основної команди порівнюємо розпізнану команду з можливими командами розумного будинку використовуючи функцію command_match() (буде описано пізніше), якщо визначено великий збіг, між розпізнаною командою та командою розумного будинку, то відправляємо команду до серверу розумного будинку за допомоги функції send_command() (рис. 1.27).

```
    if response == True:
        window.label_update(f'Command {matching_command} successfully sent')
        answer_to_user(f'Command {matching_command} successfully sent', 'en')
        activate = True
    else:

        # Якщо сервер відповів не позитивно, також сповіщуємо користувача
        # щодо помилки під час відправлення команди
        window.label_update(response)
        answer_to_user(f'Error while sending the command', 'uk')
        continue
else:

    # Якщо не було знайдено підходящу команду під час перевірки
    # сповіщуємо користувача синтезом голосу
    window.label_update('There is no such command')
    answer_to_user('There is no such command', 'en')
    continue
```

Рисунок 1.27 – Перевірка відповіді сервера/відповідь до користувача

Потім перевіряємо відповідь сервера (рис.1.27), якщо сервер відповів що команда успішно відправлена, то повідомляємо користувача що команда успішно

надіслана використовуючи функцію answer_to_user() (буде описано пізніше), та змінюємо інформаційний текст на головному вікні відповідно. Якщо сталася помилка, то повідомляємо користувача про помилку під час надсилання команди. Якщо під час перевірки команди, не було визначено підходящої команди, то повідомляємо користувача про відсутність запрошуваної команди.

```
except sr.UnknownValueError:

    # Обробляємо помилки, якщо під час спроби розпізнавання мови сервіс не зміг
    # визначити текст, відловлюємо цю помилки, та замість критичного вимикання програми
    # сповіщуємо користувача на секунду виводячі текст про помилку у вигляді об'єкту надпису
    # на головному окні програми
    window.label_update("Google Speech Recognition could not understand audio")
    sleep(1)
except sr.WaitTimeoutError:

    # Якщо отримана помилка щодо завершення слухання відрізку аудіо, то пропускаємо помилку, та
    # починаємо головний цикл спочатку
    pass
except sr.RequestError as e:

    # Якщо була отримана помилка отримання результатів від сервісу для розпізнавання мови
    # сповіщуємо користувача через об'єкт надпису на головному вікні
    window.label_update(f'Could not request results from Google Speech Recognition service; {e}')
    sleep(1)
```

Рисунок 1.28 – Обробка помилок

Весь попередній блок коду, було загорнуто у блок try: except: якщо під час виконання коду виникла одна з оброблюваних помилок виконується наступний код (рис.1.28). Помилка UnknowValueError, означає що сервіс, google speech recognition не зміг розпізнати мову, якщо виявлена така помилка, змінюємо інформаційний надпис повідомляючи користувача про цю помилку. Помилка WaitTimeoutError, означає що метод listen() нашого розпізнавача, не зафіксував аудіо за відведений час, це час котрий ми передавали через аргумент timeout, якщо зафіксована така помилка то нічого не відбудеться, програма просто повертається до очікування фрази активації. Помилка RequestError означає що виникла проблема з сервісом розпізнавання мови, змінюємо інформаційний текст на головному вікні повідомляючи користувача про це, і знову продовжуємо очікувати фразу активації.

Другий блок коду який відповідає за налаштування української мови, має точно таку логіку, тільки весь текст для користувача перекладений на українську, і сервіс розпізнавання мови очікує українську мову, так само функція відповіді до

користувача з використанням синтезу голосу, буде відповідати на українській мові. Тому більш детально цей блок розглядати не будемо.

Додаткові функції програми

Тепер розглянемо додаткові функції які не були описані до цього.

```
19  # Функція для отримання поточної встановленої мови, або її зміни
20  def language_func(change=bool, lang=str):
21
22      '''
23      change - флаг для встановлення режиму зміни поточної мови,
24      якщо встановлено True, то необхідно передати код встановлюваної мови,
25      у цьому режимі функція виконує роль процедури, змінюючі дані файли з мовою,
26      не повертаючи результату у програму. Якщо встановити флаг False, то виконується
27      у режимі функції, та повертає поточну встановлену мову до програми
28      '''
29
30      if change:
31          with open('user_language.txt', 'w') as file:
32              file.write(lang)
33      else:
34          with open('user_language.txt', 'r') as file:
35              return str(file.read())
```

Рисунок 1.29 - Функція language_func

Функція language_func (рис.1.29), відповідає за отримання поточної налаштованої мови, або її зміну, ця функція зчитує текстовий файл user_language.txt у якому зберігається тільки один рядок що містить код мови (en/uk). І залежно від переданих аргументів до функції, можна або повернути у програму поточну мову, або змінити мову у текстовому файлі, на передану.

Далі йде функція command_match (рис.1.30), ця функція приймає розпізнану команду, і команди розумного будинку. Потім використовуючі функцію fuzz.ratio, з бібліотеки thefuzz. Ми визначаємо відсоток схожості між розпізнаною командою, та почергово з кожною командою розумного будинку, використовуючи алгоритм Левенштейна. Після цього формується список з усіх команд що мають більше 50% схожості з розпізнаною командою, та обирається команда з найбільшим відсотком схожості серед можливих команд розумного будинку, і саме ця команда повертається у програму.

```python
# Функція для визначення підходящої команди
def command_match(user_command=str, possible_commands=list):
    '''
    Функція для встановлення найбільш підходящої команди серед можливих команд,
    використовує алгоритм левентшейна, для визначення підходящої команди
    user_command - приймає рядок з запитуваною командою
    possible_commands - приймає список, з можливими командами
    далі функція використовуючи алгоритм Левенштейна, визначає яка з можливих команд
    найбільше підходить до запитуваної команди, якщо не було досягнуто 50% схожості,
    то повертається False, якщо було досягнуто більше 50% схожості, повертається одна
    з можливих команд, з найбільшим відсотком схожості
    '''

    matches = []

    for command in possible_commands:
        match = fuzz.ratio(user_command, command)
        if match >= 50:
            matches.append(command)
        else:
            matches.append('None')

    best_match = max(matches, key=lambda match: fuzz.ratio(user_command, match))

    if best_match == 'None':
        return False

    return best_match
```

Рисунок 1.30 – Функція command_match

Функція send_command (рис. 1.31), відповідає за відправлення визначеної команди до серверу розумного будинку. Ця функція приймає саму команду, адресу сервера розумного будинку, ідентифікатор відправника (рядок вказуючий на відправника, у нашому випадку ‘Система голосового управління розумним будинком/Smart home voice control system’). І код мови на якій була відправлена команда. Після чого формується словник з цими даними, і у блоці try/except йде відправка POST запиту до серверу розумного будинка. Якщо була визначена помилка, то повертається ‘Помилка/Error’ у програму на місце виклику. Якщо помилки не було. Йде перевірка на отриманий код від серверу, якщо код дорівнює 200 (що означає OK), то повертаємо True у місце виклику, якщо код не дорівню 200, то повертаємо текст про помилку, на код яким відповів сервер на спробу запиту до нього.

```python
# Функція для відправлення команди на сервер розумного будинку
def send_command(command=str, url=str, source=str, language=str):

    '''

    '''

    command_data = {
    'command': command,
    'source': source,
    'language': language
    }

    try:
        response = requests.post(url, json=command_data)
    except Exception:
        if language_func(change=False) == 'en':
            return 'Error'
        else:
            return 'Помилка'

    if response.status_code == 200:
        return True
    else:
        return f'Error when sending a command - {response.status_code}'
```

Рисунок 1.31 – Функція send_command

Функція answer_to_user (рис.1.32) ця функція передає в аудіо потік користувача голосовое повідомлення щодо результату роботи програми. На вхід приймає рядок з текстом для якого потрібно синтезувати голос, та мову на якій потрібно це зробити. Потім викликається функція gTTS яка створює байт дані щодо аудіо файлу, після чого цей файл зберігається з розширенням mp3. Далі цей файл зчитується в змінну answer у вигляді байт даних, та програється користувачу з використанням функції play бібліотеки pydub.

```python
# Функція для відповіді до користувача з використанням синтезу голосу
def answer_to_user(command=str, language=str):

    tts = gTTS(command, lang=language)
    tts.save('answer_for_user.mp3')
    answer = AudioSegment.from_mp3('answer_for_user.mp3')
    play(answer)
```

Рисунок 1.32 – Функція answer_to_user

Сервер на Flask.

Також для тестування функції відправлення команди до серверу, був розгорнутий простий сервер на Flask (рис.1.33). Flask це веб-фреймворк для створення веб-додатків та сайтів, має вбудовані засоби для створення серверу для тестування веб-сайтів. Створимо файл example_of_server.py.

```python
from flask import Flask, request, jsonify

app = Flask(__name__)

# Роут для обробки запиту POST команди розумного будинку
@app.route('/api/smarthome', methods=['POST'])
def handle_command():
    data = request.get_json()
    command = data.get('command')
    source = data.get('source')
    language = data.get('language')

    # Формування відповіді для користувача
    response_data = {
        'command': command,
        'source': source,
        'language': language,
        'message': True
    }

    print(f'Received command:\n {command}\nSource:\n {source}\nLanguage:\n {language}')

    return jsonify(response_data), 200

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

Рисунок 1.33 – Простий сервер на Flask

Основна задача цього серверу це приймати POST запити, та виводити до вікна терміналу інформацію передану запитом. Для тестування програми, саме на цей локальний сервер відправлялись команди розумного будинку.

Запуск додатку

```python
# Створення екземпляру QApplication
app = QApplication(sys.argv)

# Створення екземпляру головного вікна
window = MainWindow()
window.show()

# Створення екземпляру потіку головної кнопки
main_button_thread = MainButtonThread()
main_button_thread.finished.connect(lambda x: print('Main thread stop working'))
main_button_thread.start()

# Запуск головного циклу подій QAplication
app.exec()
```

Рисунок 1.34 - Запуск

Для запуску додатку створимо об’єкт QApplication, це головний застосунок який ініціалізує основні компоненти PyQt6 та здійснює управління головним циклом подій програми.

Створимо екземпляр нашого вікна та відобразимо його методом show(), та потоку для головної кнопки.

Після цього запускаємо головний цикл методом exec() об’єкта QApplication. Та запускаємо сам файл voice_assistant_main.pyw (.pyw режим без вікна терміналу)

Отримаємо таке головне вікно програми (рис. 1.35):

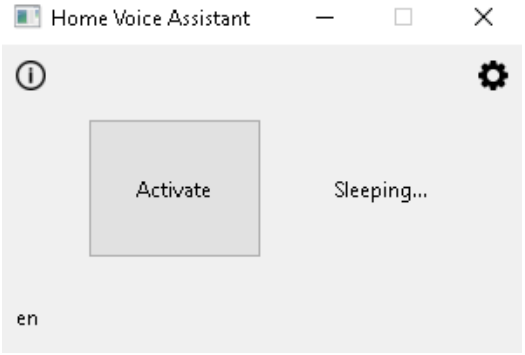


Рисунок 1.35 – Головне вікно програми

Перевіримо роботу програми, натиснемо на кнопку з інформацією, з’явилося таке вікно (рис. 1.36):

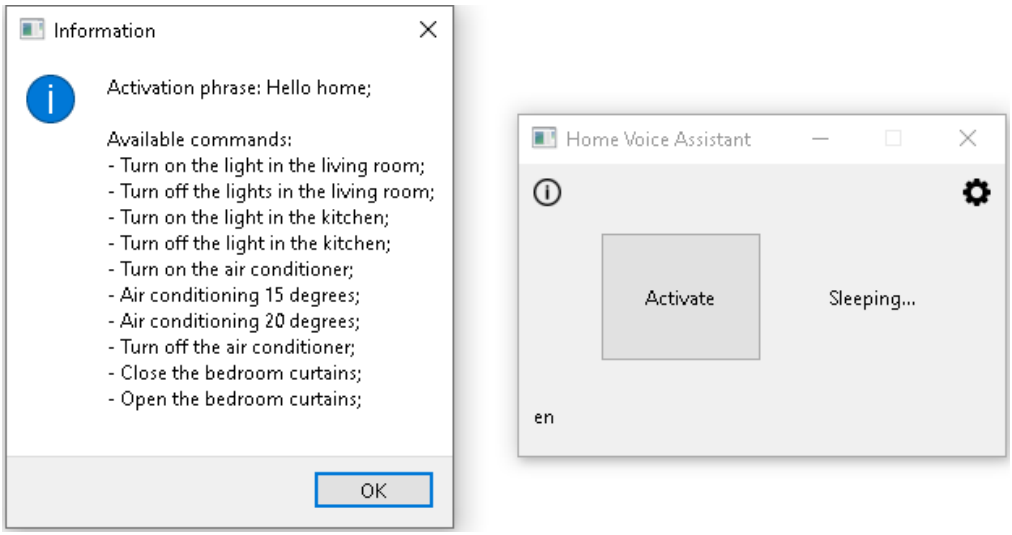


Рисунок 1.36 – Перевіра роботи програми

Зачинемо його, тепер спробуємо відкрити меню налаштувань та змінити мову (рис.1.37):

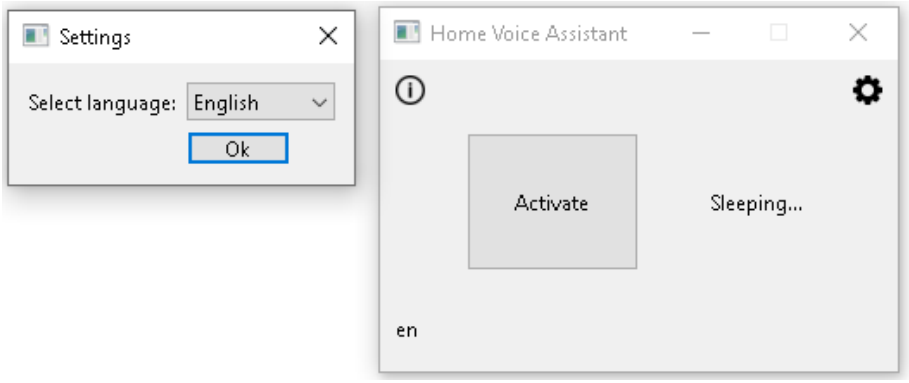


Рисунок 1.37 – Відкриття налаштувань

Відкриємо випадаючий список (рис. 1.38):

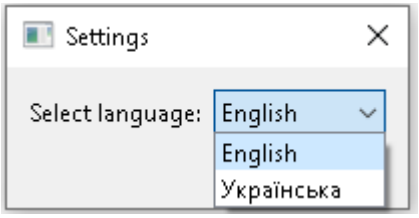


Рисунок 1.38 – Вибір мови

Встановимо українську мову та натиснемо Ok (рис. 1.39):

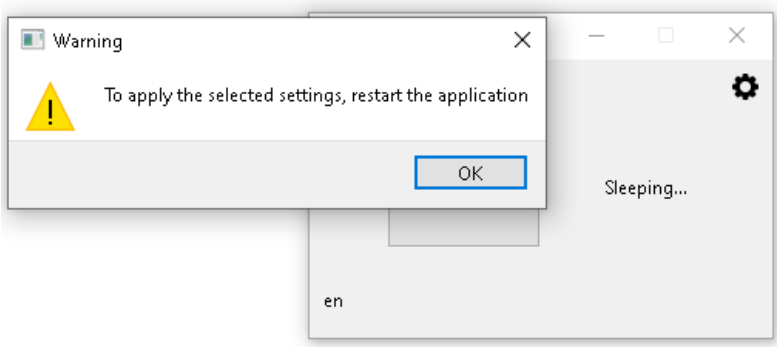


Рисунок 1.39 – Спроба змінити мову

З’явилося попередження, про необхідність перезапуску додатку для встановлення обраних налаштувать, також колір позначки щодо поточної мови змінився на обрану, та змінив колір на червоний (рис.1.40):

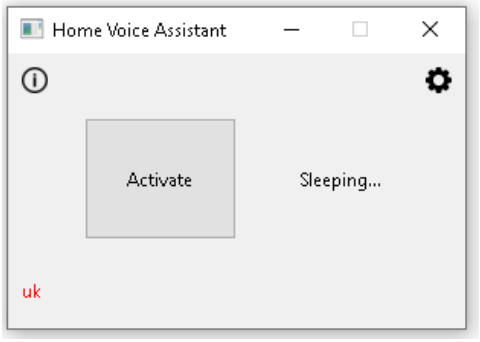


Рисунок 1.40 – Зміна позначки щодо встановленої мови

Тепер перезапустимо додаток і запустимо основний цикл програми натиском на основну кнопку (рис.1.41):

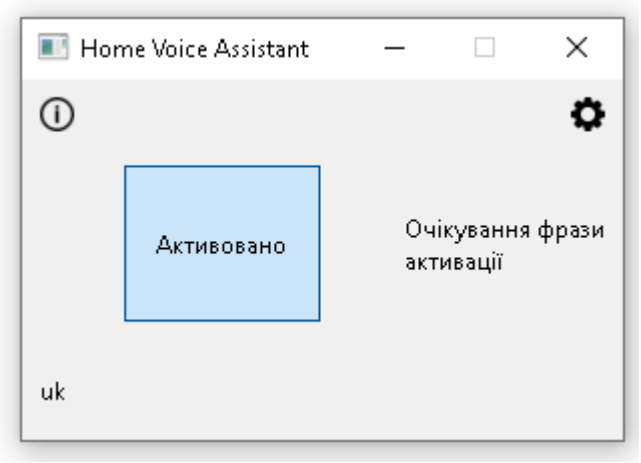


Рисунок 1.41 – Натиск на основну кнопку

Після перезапуску текст змінився на український, також після натиску на основну кнопку програма почала очікувати фразу активації, а основна кнопка змінила текст. Тепер промовимо фразу активації (рис.1.42).

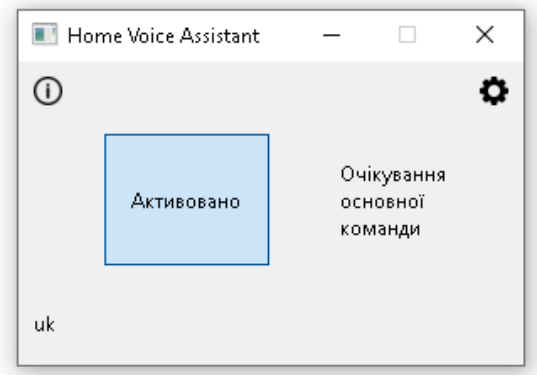


Рисунок 1.42 – Очікування основної команди

Коли програма приймає основну команду, промовимо команду щодо увімкнення світла у вітальні (рис.1.43).

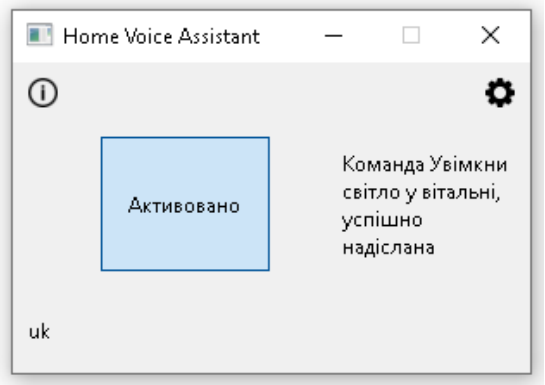


Рисунок 1.43 – Успішність надсилання команди

Команда була успішно розпізнана і надіслана до серверу, після чого я почув відповідь синтезованим голосом, щодо успішності роботи додатку, також якщо подивимось до терміналу нашого сервера, то побачимо успішне отримання команди (рис. 1.44).

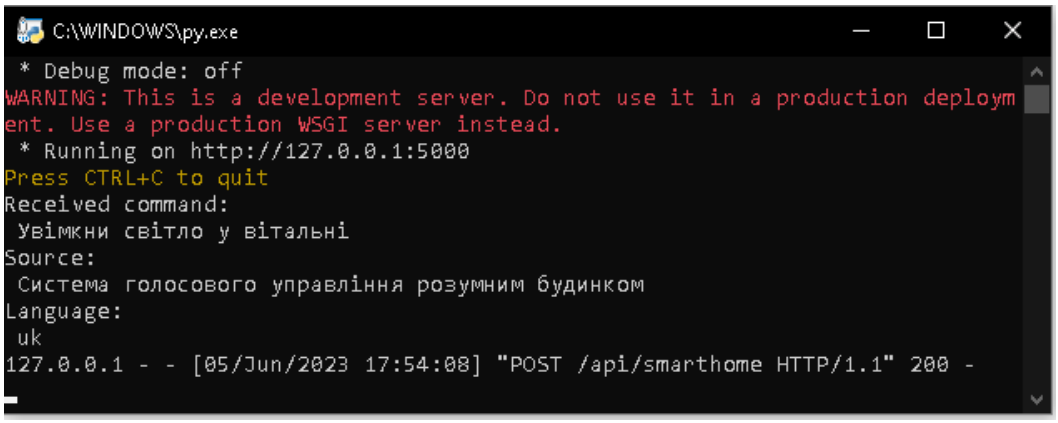


Рисунок 1.44 – Стан терміналу сервера, після отримання команди

Висновок. Отже була розроблена програмна система голосового управління розумним будинком, на Windows 10, з використанням мови програмування Python. Були реалізовані необхідні системи, а саме графічний інтерфейс для користувача, система розпізнавання мови, система відправлення команди до серверу розумного будинку та система відповіді до користувача через використання синтезу голосу.

2 ЕКОНОМІЧНИЙ РОЗДІЛ

2.1 Резюме

В даному дипломному проекті розроблена програмна система голосового управління розумним будинком. Голосовий інтерфейс - це програмний продукт, який за допомогою голосової або мовної платформи дозволяє взаємодіяти користувачеві з системою «Розумний будинок», віддаючи команди для запуску окремих сценаріїв і дій голосом і отримуючи інформацію в формі звукових повідомлень. Система була реалізована на операційній системі Windows 10 з використанням мови програмування Python. Були успішно реалізовані всі зумовлені завданням функції системи. Також була реалізована додаткова функція, підтримка двох мов українська/англійська, та можливість налаштування поточної мови системи. Ефективність кожного програмного продукту визначається його якістю таефективністю процесу розробки. Якість ПП визначається наступними складовими: з точки зору користувача; з позиції використання ресурсів; виконання вимог до програмного забезпечення.

Оцінка якості програмного продукту з точки зору користувача визначається необхідним на стадії функціонування розміром оперативної пам'яті ЕОТ, витратами машинного часу, пропускною спроможністю каналів передачі даних.

Оцінка якості програмного продукту включає визначення трудомісткості і вартості його створення.

2.2. Визначення трудомісткості розробки програмного забезпечення.

Тривалість розробки програмного продукту залежить від його обсягу, трудомісткості розробки, кваліфікації виконавців, а також планових термінів, визначених умовами ринку. Методом структурної аналогії по відповідних каталогах аналогів програмного забезпечення визначається обсяг програмних засобів, у тисячах умовних машинних команд програми аналога

Таблиця 2.1 - Каталог аналогів

Найменування ПП	Обсяг функції ПП – V_o, усл. машинних командах.
1. ПП автоматизованих розрахунків	1300 – 8600
3. ПП введення інформації	1060 – 5750
4. ПП оптимізації розрахунків	1300 – 4200

У таблиці 2.1 представлені аналоги програмного забезпечення, функції яких, у більшому або меншому ступені, виконує розроблений програмний продукт. Для нашого варіанта виділено сірим кольором.

Вибравши аналог ПП, що містить Vo в умовних машинних командах, трудомісткості визначати на основі табл.2.2.

Таблиця.2.2 – Визначення трудомісткості

Обсяг ПП, тис.умов.машинних команд	Норма часу, люд/год
1.00	229
2.00	244
3.00	262

На підставі отриманого значення, по довіднику, визначається укрупнена норма часу на розробку аналога програмного забезпечення (коректується поправочним коефіцієнтом враховуючої умови розробки ПП, тобто в умовах комп'ютера, Кк=0,7÷0,8): T^a = 229 х 0,8 = 183,2 (люд/годин).

Трудомісткість програмного продукту визначається по кожному етапу розробки окремо на підставі трудомісткості аналога з урахуванням складності розробки, ступеня новизни і ступеня використання в розробці стандартних модулів на підставі формул:

$$T_{ТЗ} = T^a p \times L_1 \times K_H \qquad (2.1)$$

$$T_{ТП} = T^a p \times L_2 \times K_H \qquad (2.2)$$

$$T_{РП} = T^a p \times L_3 \times K_H \times K_T \qquad (2.3)$$

Для розрахунку необхідні наступні коефіцієнти:

Li – питома вага і-го етапу розробки (див. табл. 2.3.);

Кн – поправочний коефіцієнт, що враховує ступінь новизни (див. табл. 2.4.);

Кт – поправочний коефіцієнт, що враховує ступінь використання в розробці типових програм (див. табл. 2.5.).

Таблиця 2.3 - Значення питомих коефіцієнтів трудомісткості стадії в загальній трудомісткості розробки ПП

Код стадії	Ступінь новизни		
	А	Б	В
ТЗ (L_1)	0,15	0,12	0,12
ТП (L_2)	0,16	0,15	0,11
РП (L_3)	0,55	0,58	0,61

Для нашого варіанта виділено сірим кольором.

Таблиця 2.4 - Значення поправочного коефіцієнта, що враховує ступінь новизни

Код ступеня новизни	Ступінь новизни	Значення $К_н$
А	Принципово нові ПП	1,75 – 1,2
Б	ПП – розвиток визначеного параметричного ряду	1,0 – 0,8
В	ПП маючий аналог	0,7

Для нашого варіанта виділено сірим кольором.

Таблиця 2.5 - Значення коефіцієнта ступеня використання в розробці типових програм

Ступінь охоплення реалізованих функцій розроблювального ПП типовими програмами, %	Значення $К_т$
60 і вище	0,6
40-60	0,7
20-40	0,8
До 20	0,9

Для нашого варіанта виділено сірим кольором.

Тепер розраховуємо трудомісткість по кожному етапу окремо:

Трудомісткість технічного завдання

$$Т_{тз}=Т^а * L_1 * К_н = 183,2*0,12*0,8 = 17,58 \quad \text{(люд/годин)} \qquad (2.1)$$

Трудомісткість розробки технічного проекту

$$Т_{тп}=Т^а * L_2 * К_н = 183,2*0,15*0,8 = 21,98 \quad \text{(люд/годин)} \qquad (2.2)$$

Трудомісткість розробки робочого проекту

$T_{рп}=T^{а}*L_{З}*К_{н}*К_{т}=183,2*0,58*0,8*0,7=59,5$ (люд/годин) (2.3)

Для подальших розрахунків визначили кількість папера, витраченого на кожен етап: технічне завдання $N_{тз}=2$ (стр), розробка ТП $N_{тп}=19$(стр), розробка робочого проекту $N_{рп}=22$(стр), пояснювальна записка відповідно $N_{пз}=22$ (стр) Розрахунок зведений у таблицю 2.6

Таблиця 2.6 - Розрахунок трудомісткості ПП

Найменування етапів	Розрахунок, годин		
1	2	3	4
1.Розробка ТЗ	$T_{Ртз}=17,58$	$T_{кк}=0,7*2=1,4$	$T_{нк}=0,15*2=0,3$
2.Розробка ТП	$T_{Ртп}=21,98$	$T_{кк}=0,7*19=13,3$	$T_{нк}=0,15*19=2,85$
3.Розробка РП	$T_{Ррп}=59,5$	$T_{кк}=0,7*22=15,4$	$T_{нк}=0,15*22=3,3$
4.Розробка ПЗ	$T_{пз}=1,5*$ $*22=33$	$T_{кк}=0,7*22=15,4$	$T_{нк}=0,15*22=3,3$
Усього, в т.ч.:	$\Sigma T=187,25$		
- на розробку	$\Sigma T_{р}=132$		
- контроль керівника		$\Sigma T_{кк}=45,5$	
- нормоконтроль			$\Sigma T_{нк}=9,75$

2.3 Розрахунок ціни програмного продукту

У цьому розділі для визначення ціни розраховуємо основну заробітну плату виконавців, матеріальні витрати, вартість машино – години і витрати на розробку ПО. Розрахунок основної заробітної плати виконавців приведений у таблиці 2.7. Відповідно до статті 8 «Закону про Державний бюджет України на 2023» встановлено мінімальну заробітну плату у місячному розмірі з 1 січня 2023 року - 6700 гривень; мінімальну погодинну тарифну ставку – 40.46 грн.

На підставі отриманих даних по окремих статтях витрат складена калькуляція планової собівартості в цілому ПП за формою, приведеною в таблиці 2.9.

Таблиця 2.7 - Розрахунок основної заробітної плати виконавців

Найменування робіт	Трудомісткість робіт, години	Погодинна тарифна ставка, грн.	Розрахунок, грн.
1.Розробка ПП	$\Sigma T_р$=132	40,46	5341
2.Контроль керівника	$\Sigma T_{кк}$=45,5	70.00	3185
3.Нормоконтроль	$\Sigma T_{нк}$=9,75	70.00	682
Усього	-	-	ΣЗо= 9208

Зробимо розрахунок матеріальних витрат на розробку ПП. Розрахунок зведемо в таблицю 2.8

Таблиця 2.8 - Розрахунок матеріальних витрат на розробку ПО

Найменування матеріальних витрат	Тип, модель	Кількість	Ціна одиниці, грн.	Вартість, грн.
Папір	Лист А4	81	1,5	$В_{мі}$=122,5
Транспортні витрати (10%)				$Втр\_з = 0{,}1 \times Вм1$= 0,1 * 122,5 = 12,25
Усього				$В_м=В_{мі}+В_{тр_з=}$122,5+12,25= 134,75

Таблиця 2.9 - Розрахунок статей витрат планової собівартості

Стаття витрат	Значення, грн.	Формула розрахунку
1. Матеріали	134,75	$В_м$ (див. табл. 2.8)
2. Основна заробітна плата	9208	$З_о$ (див. табл. 2.7)
3.Додаткова заробітна плата	1381	$Зд = 0{,}15 \times Зо$ =*0,15*9208*
4.Відрахування до єдиного фонду соціального внеску	2329,6	$Вє.с.в. = 0{,}22 \times (Зо + Зд)$ =0,22*(9208+ +1381)
5. Накладні витрати	2762	$Внак. = 0{,}3 \times Зо$ =0,3*9208
6. Повна собівартість	15815	$С_{пов}$= $В_м$ +$З_о$ + $З_д$ +Вє.с.в. + Внак.= =134,75+9208+1381+2329,6+ +2762

Розмір прибутку, що включається в ціну, визначаємо по наступній формулі:

$$П=(С_{пов}*Р)/100 = (15815*12)/100= 1898 \quad (2.4)$$

Де Р– плановий рівень рентабельності (10-15%).

Оптова ціна (кошторисна вартість) визначається по формулі:

$$Ц_о = С_{пов}+П = 15815+1898 = 17713 \quad (2.5)$$

Податок на додану вартість визначаємо по наступній формулі:

$$ПДВ = 0.2 * Ц_о = 0,2*17713=3542,6 \quad (2.6)$$

Виходячи з отриманих даних, ціна реалізації розробленого програмного продукту на основі наступної формули, становитиме:

$$Ц_р=Ц_о+ПДВ = 17713+3542,6=21255,6 \quad (2.7)$$

3 ОХОРОНА ПРАЦІ

Сучасний прогрес у технічному та технологічному секторах виробництва вимагає постійної автоматизації та оптимізації виробничих процесів. У зв'язку з широким застосуванням комп'ютерів працівниками для виконання своїх завдань, законодавство України чітко регулює вимоги та стандарти щодо використання комп'ютерної техніки на підприємстві, зокрема стосовно безпеки праці під час роботи з комп'ютером.

Питання охорони праці працівника необхідно вирішувати на всіх стадіях трудового процесу незалежно від виду професійної діяльності.

В дипломному проекті у розділі «охорона праці» розглянуті основні небезпечні та шкідливі виробничі фактори що можуть мати місце під час розробки системи голосового управління розумним будинком на Windows 10 з використанням мови програмування Python та розробка заходів забезпечення безпечної роботи програміста.

3.1 Аналіз небезпечних та шкідливих факторів, що впливають на програміста під час розробки системи

Забезпечення безпечних і здорових умов праці в значній мірі залежить від правильної оцінки небезпечних, шкідливих виробничих факторів. Однакові по складності зміни в організмі людини можуть бути викликані різними причинами.

Це можуть бути фактори виробничого середовища, надмірне фізичне і розумове навантаження, нервово-емоційна напруга, а також різне сполучення цих причин.

Визначення та вивчення факторів, що впливають на функціональний стан користувачів комп'ютерів дозволить виділити основні причини виникнення станів напруженості, стомлення, стресу і здійснити відповідні профілактичні заходи.

3.2 Виробниче приміщення

Площу приміщень, в яких розташовують персональні комп'ютери, визначають згідно з чинними нормативними документами з розрахунку на одне робоче місце, обладнане ПК:

– площа – не менше 6,0 кв.м,

– обсяг – не менше 20,0 куб.м, з урахуванням максимальної кількості осіб, які одночасно працюють у зміні.

Робочі місця з ПК заборонено облаштовувати у підвальних або цокольних приміщеннях будинків.

В обладнанні приміщень забороняється використання полімерних матеріалів (деревинно-стружкові плити, шпалери, що миються, рулонні синтетичні матеріали, шаруватий паперовий пластик тощо), що виділяють у повітря шкідливі хімічні речовини. Покриття підлоги повинно бути матовим, а поверхня – рівною, неслизькою, з антистатичними властивостями.

Заземлені конструкції, які знаходяться в приміщеннях (батареї опалення, водопровідні труби, кабелі із заземленим відкритим екраном тощо), повинні бути надійно захищені діелектричними щитками або сітками від випадкового дотику.

Найбільш сприятливі для нервової системи світлі, пастельні тони – зеленувато-блакитний, ясно-сірий, золотавий. Яскраві, контрастні поєднання (синій і жовтогарячий, червоний і фіолетовий) викликають втому, роздратування.

В приміщенні є вікна, які орієнтовані на південь . Тому основні кольори інтер'єру – стіни світло-блакитного кольору; підлога-зеленого.

У приміщеннях, де здійснюється робота з комп'ютерами, щодня вимагається проведення вологого прибирання з метою недопущення запиленості підлоги і меблів.

Також в цих приміщеннях повинні бути медичні аптечки першої допомоги та система автоматичної пожежної сигналізації з димовими пожежними сповіщувачами та переносними вуглекислотними вогнегасниками з розрахунку 2 шт. на кожні 20 кв.м площі приміщення. Підходи до засобів пожежогасіння повинні бути вільними.

Всі вказані вимоги до приміщень де виконується робота з ПК в дипломному проєкті виконані.

3.3 Вимоги до робочого середовища під час роботи з ПК

До причин, які зумовлюють виникнення професійних захворювань користувачів ВДТ, значне місце мають умови праці. Головними з них є ті, що створюються під впливом випромінювання з ВДТ, освітлювання, шуму, вмісту шкідливих речовин у повітрі робочої зони, іонного складу повітря, електростатичного поля.

3.4 Організація робочого місця с ПК

Організація робочого місця передбачає: правильне розташування робочого місця у виробничому приміщенні; вибір виробничих меблів; раціональне компонування комп'ютерного обладнання на робочому місці; урахування характеру та особливостей трудової діяльності.

Робочі місця повинні бути розташовані на відстані не менше ніж 1 м. від стіни з вікном, відстань між бічними поверхнями комп'ютерів має бути не меншою за 1,2м; відстань між тильною поверхнею одного комп'ютера та екраном іншого не повинна бути меншою 2,5 м ; прохід між рядами робочих місць має бути не менш ніж 1 м.

Конструкція робочого місця користувача забезпечує підтримку оптимальної робочої пози з такими ергономічними характеристиками: ступні ніг - на підлозі або на підставці для ніг; стегна – в горизонтальній площині; передпліччя - вертикально; лікті - під кутом 70º-90º, до вертикальної площини; зап'ястя зігнуті під кутом не більше 20º, відносно горизонтальної площини, нахил голови – 15º–20º, відносно вертикальної площини.

Обладнання розміщується на основному робочому столі з лівого боку. Висота робочої поверхні столу 680–800мм, а ширина – забезпечує можливість виконання операцій в зоні досяжності моторного поля. Розміри столу: висота – 725мм, ширина – 600–1400мм, глибина – 800мм–1000мм.

Робочий стіл для обладнаний підставкою для ніг шириною 400мм з можливістю регулювання по висоті. Підставка має рифлену поверхню та бортик на передньому краї заввишки 10 мм.

Робоче сидіння є підйомно-поворотним, регулюється за висотою, кутом нахилу сидіння та спинки. Поверхня сидіння є плоскою, передній край - заокруглений.

Рисунок 3.1 – Конструкція робочого місця, норми робочого простору

3.4.1 Освітлення робочого місця

Недостатність освітлення призводить до напруги зору, ослабляє увагу, приводить до настання передчасної стомленості. Надмірно яскраве освітлення викликає засліплення, роздратування і різь в очах. неправильний напрям світла на робочому місці може створювати різкі тіні, відблиски, дезорієнтувати працюючого. Всі ці причини можуть привести до нещасного випадку або профзахворювань, тому такий важливий правильний розрахунок освітленості.

Приміщення, в яких встановлені персональні комп'ютери, повинні мати природне та штучне освітлення відповідно доДБН В.2.5-28:2018 " Природне і штучне освітлення" Природне освітлення здійснюється через світлові прорізи, орієнтовані переважно на північ чи північний схід і забезпечує коефіцієнт природною освітленості (КПО) не нижче ніж 1,5%.

Штучне освітлення в приміщеннях з робочими місцями здійснюється системою загального рівномірного освітлення. У разі переважної роботи з документами, допускається застосування системи комбінованого освітлення (крім

системи загального освітлення додатково встановлюються світильники місцевого освітлення).

Рівень освітленості на поверхні робочого столу в зоні розміщення документів має становити 300-500 лк , а освітленість екрана має не перевищувати 300лк. Як джерела світла в разі штучного освітлення застосовуються переважно люмінесцентні лампи типу ЛБ. У разі влаштування відбитого освітлення у приміщеннях, де переважним чином працюють з документами, допускається застосування металогалогенних ламп потужністю 250Вт.

Допускається застосування ламп розжарювання у світильниках місцевого освітлення. Система загального освітлення має становити суцільні або переривчасті лінії світильників, розташовані збоку від робочих місць (переважно ліворуч), паралельно лінії зору працюючих.

Застосування світильників без розсіювачів та екрануючих грат заборонено. Яскравість світильників загального освітлення в зоні кутів випромінювання від 50 до 90 градусів з вертикаллю в повздовжній та поперечній площинах має становити не більше ніж 200 кд/м2 .

3.4.2 Мікроклімат

Приміщення для роботи з персональним комп'ютером мають бути обладнані системами опалення, кондиціонування повітря, або припливно-витяжною вентиляцією. У приміщеннях на робочих місцях мають забезпечуватися оптимальні значення параметрів мікроклімату: температури, відносної вологості й рухливості повітря у відповідності до ГОСТ 12.1.005-88, СН 4088-86. Згідно санітарних норм мікроклімату виробничих приміщень ДСН 3.3.6.042 99 нормування параметрів проводиться в залежності від періоду року та категорії важкості виконуваних робіт.

Робота оператора ПК за енерговитратами відноситься до категорії легких робіт Іа, Іб. У таблиці 3.1. наведені оптимальні параметри мікроклімату в приміщеннях, де виконуються роботи операторського типу.

Таблиця 3.1 – Параметри мікроклімату для приміщень з ПК

Період року	Параметр мікроклімату	Величина
Холодний	Температура повітря в приміщенні; відносна вологість; швидкість руху повітря	22...24°С; 40... 60%; до 0,1 м/с
Теплий	Температура повітря в приміщенні; відносна вологість; швидкість руху повітря	23...25 °С; 40...60%; 0,1...0,2 м/с

3.4.3 Шум

Джерелами шуму при роботі з комп'ютерною технікою є жорсткий диск, вентилятор блока живлення мережі, вентилятор розташований на процесорі, швидкісні CD-ROM, механічні сканери, пересувні механічні частини принтера.

При роботі матричних голкових принтерів шум виникає при переміщенні головки принтеру і в процесі удару голок головки по паперу. При роботі вентиляційної системи ПК, яка забезпечує оптимальний температурний режим електронних блоків ПК і вмонтована в задню панель, створюється аеродинамічний шум. Окрім того діють і інші зовнішні джерела шуму, не пов'язані з роботою ПК.

Для офісів та приміщень, обладнаних персональними комп'ютерами або технікою для бізнесу допустимий рівень шуму цілодобово - 50 дБА, а максимальний- 65 дБА

3.4.4 Електробезпека

Персональні комп'ютери, периферійні пристрої, інше устаткування (апарати управління, контрольно-вимірювальні прилади, світильники), електропроводи та кабелі за виконанням і ступенем захисту відповідають класу зони, мають апаратуру захисту від струму короткого замикання та інших аварійних режимів.

Лінія електромережі для живлення персональних комп'ютерів та периферійних пристроїв виконана як окрема групова трипровідна мережа шляхом прокладання фазового, нульового робочого та нульового захисного провідників. Нульовий захисний провідник використовується для заземлення (занулення)

електроприймачів. Персональні комп'ютери і периферійні пристрої повинні підключатися до електромережі тільки за допомогою справних штепсельних з'єднань і електророзеток заводського виготовлення.

При розміщенні в приміщенні до п'яти персональних комп'ютерів і периферійних пристроїв допускається прокладання трипровідникового захищеного проводу або кабелю в оболонці з негорючого чи важкогорючого матеріалу по периметру приміщення без металевих труб та гнучких металевих рукавів.

3.5 Пожежна безпека при роботі з ВДТ

Пожежна безпека забезпечується системою запобігання пожежі і системою пожежного захисту. У всіх службових приміщеннях обов'язково повинен бути "План евакуації людей при пожежі", що регламентує дії персоналу у разі виникнення вогнища загоряння і в якому зазначено місця розташування пожежної техніки. Пожежі становлять особливу небезпеку, тому що пов'язані з великими матеріальними втратами.

У сучасних ПК дуже висока щільність розміщення елементів електронних схем. У безпосередній близькості один від одного розташовуються сполучні дроти, кабелі. При протіканні по них електричного струму виділяється значна кількість теплоти. При цьому можливо оплавлення ізоляції. Для відведення надлишкової теплоти від ПК служать системи вентиляції та кондиціонування повітря. При постійному дії ці системи представляють собою додаткову пожежну небезпеку.

Джерелами запалювання уможуть бути електронні схеми від ПК, прилади, застосовувані для технічного обслуговування, пристрої електроживлення, кондиціонування повітря, де внаслідок різних порушень утворюються перегріті елементи, електричні іскри та дуги, здатні викликати загоряння горючих матеріалів.

До засобів гасіння пожежі відносяться внутрішні пожежні водопроводи (крани - ПК), вогнегасники (вуглекислотні та порошкові), сухий пісок тощо.

В будівлях пожежні крани встановлюють в коридорах, на майданчиках сходових кліток. Кожний пожежний кран укомплектований пожежним рукавом і розміщений у відповідних ящиках, які знаходяться на висоті 1,35 м від полу.

Для гасіння пожеж на початкових стадіях широко застосовуються вогнегасники. У виробничих приміщеннях це головним чином вуглекислотні вогнегасники, достоїнством яких є висока ефективність гасіння пожежі, збереження електричного устаткування. Розташовують вогнегасники на видних місцях, на висоті не більше як 1,5 м від полу.

Вогнегасники повинні бути розташовані на вертикальній перегородці або стіні, на спеціальних кронштейнах або в пожежних шафах. Інструкції та правила щодо їх використання, нанесені на корпус вогнегасника, повинні бути повернені назовні і добре видно в екстрених ситуаціях. Механізми запуску і двері пожежних шаф повинні мати захисні пломби.

Рисунок 3.2 – Засоби пожежогасіння

Виробничі приміщення мають запасні виходи. Двері повинні мати освітлений надпис «Запасний вихід». План евакуації вивішується на видному місці у основного виходу із приміщення.

ВИСНОВКИ

В дипломному проекті розроблялась програмна система голосового управління розумним будинком на Windows 10. Для розробки була обрана мова програмування Python. Об’єктом дослідження для розробки були технології розпізнавання мови STT (Speech to Text), та технології синтезу мови TTS(Text to Speech).

Загалом за весь період роботи були виконані наступні завдання:

– огляд існуючих систем голосового управління розумним будинком,

– формування структури системи (функцій, підсистем),

– огляд можливих технічних рішень та можливих існуючих реалізацій технологій STT/TTS,

– опис та огляд допоміжних бібліотек на мові Python для реалізації інших функцій (інтерфейс користувача, відправлення команд),

– створення програмної частини (написання коду та логіки).

Результатом стала програмна система голосового управління розумним будинком що включає інтерфейс користувача, систему розпізнавання мови, систему фрази активації, систему обробки команд, систему відправлення команд до серверу розумного будинку та систему відповіді до користувача з використанням технології TTS.

Були реалізовані такі функції: інтерфейс користувача, обробка голосових команд, фраза активації, відправлення команд до серверу розумного будинку, відповідь до користувача щодо результатів роботи додатку, збереження даних користувача (можливі команди). Також була додана підтримка двох мов англійська та українська.

Система вирішить ряд проблем приватності та сумісності, полегшить управління розумним будинком, простий та зрозумілий інтерфейс не вимагає додаткового навчання чи звикання. Розробка може бути використана для подальших модифікацій чи створення аналогів, і допоможе покращити розуміння технологій STT та TTS у звичайних користувачів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. AI-Heeti, Abrar (January 4, 2019) https://www.cnet.com/home/smart-home/amazon-has-sold-more-than-100-million-alexa-devices/ CNET. CBS Interactive.
2. Callaham, John (April 8, 2013). "Amazon Echo owners can now control WeMo and Philips Hue devices with their voice". Connectedly. Mobile Nations.
3. https://blog.google/products/assistant/ces-2020-google-assistant/ Google Blog.
4. Розумний дім: [Електронний ресурс]: // Sutem – Режим доступу до ресурсу: http://sutem.com.ua/.
5. Система розумний дім: [Електронний ресурс]: // Ecotown – Режим доступу до ресурсу: https://ecotown.com.ua/news/Systema-Rozumnyy-dim_zmenshuye-vytraty-na-komposluhy-do-30/
6. Функції та комфорт розумного дому: [Електронний ресурс]: // melask – Режим доступу до ресурсу: http://melask.com.ua/rozumniy-dim/komfort.html.
7. Розумний дім – технологія економії: [Електронний ресурс]: // right_choice – Режим доступу до ресурсу: http://right-choice.com.ua/rozumniy-dim_tehnologiya-ekonomiyi-zruchnosti-i-komfortu-visokogo-rivnya.
8. Розумний будинок: [Електронний ресурс]: // lady.tochka – Режим доступу до ресурсу: http://lady.tochka.net/ua/58959-chto-takoe-umnyy-dom/.
9. Керований пристрої «Розумного» будинку: [Електронний ресурс]: // sutem – Режим доступу до ресурсу: http://sutem.com.ua/7113smartbus.php.
10. https://wiki.python.org/moin/TkInter - python wiki
11. RiverbankComputing FAQ - https://riverbankcomputing.com/software/pyqt/
12. Qt Designer and Python: Build Your GUI Applications Faster - https://realpython.com/qt-designerpython/#:~:text=Qt%20Designer%20is%20a%20Qt,objects%20on%20an%20empty%20form.
13. wxPython wiki - https://wiki.wxpython.org/wxPython
14. kivy офіційний сайт - https://kivy.org/doc/stable/
15. Офіційний сайт python - https://www.python.org/

16. Cloud Speech-to-Text https://cloud.google.com/speech-to-text
17. Sphinx overview https://cmusphinx.github.io/wiki/tutorialoverview/
18. Vosk офіційний репозиторій на github - https://github.com/alphacep/vosk-api/
19. https://openai.com/research/whisper OpenAI whisper введення
20. Pyttsx3 – Pypi https://pypi.org/project/pyttsx3/
21. gTTS – Pypi https://pypi.org/project/gTTS/
22. TTS – Pypi https://pypi.org/project/TTS/
23. Основи охорони праці. навчально-методичний посібник для студентів вищих закладів педагогічного напрямку / [Укладачі: В.І. Кошель,Г.П. Сав'юк, Б.С. Дзундза] – Івано-Франківськ: НАІР, 2020. –182 с.
24. Козяр М.М., Щедрій Я.І., Станіславчук О.В. Основи охорони праці, безпеки життєдіяльності та цивільного захисту населення: Навч.посіб. -К.: Кондор, 2012.

```
import sys
import speech_recognition as sr
import requests

from time import sleep
from gtts import gTTS
from pydub import AudioSegment
from pydub.playback import play
from needs import uk_possible_commands, en_possible_commands, url
from thefuzz import fuzz
from PyQt6.QtCore import Qt, QSize, QThread, pyqtSignal, pyqtSlot
from PyQt6.QtWidgets import (
        QApplication, QMainWindow, QPushButton, QGridLayout,
        QSpacerItem, QSizePolicy, QWidget, QLabel, QMessageBox,
        QDialog, QComboBox
    )
from PyQt6.QtGui import QIcon

# Функція для отримання поточної встановленої мови, або її зміни
def language_func(change=bool, lang=str):

    '''
    change - флаг для встановлення режиму зміни поточної мови,
    якщо встановлено True, то необхідно передати код встановлюваної мови,
    у цьому режимі функція виконує роль процедури, змінюючі дані файли з
мовою,
    не повертаючи результату у програму. Якщо встановити флаг False, то
виконується
    у режимі функції, та повертає поточну встановлену мову до програми
    '''

    if change:
        with open('user_language.txt', 'w') as file:
            file.write(lang)
    else:
        with open('user_language.txt', 'r') as file:
            return str(file.read())

# Функція для визначення підходящої команди
def command_match(user_command=str, possible_commands=list):

    '''
    Функція для встановлення найбільш підходящої команди серед можливих
команд,
```

```
    використовує алгоритм левентшейна, для визначення підходящої команди
    user_command - приймає рядок з запитуваною командою
    possible_commands - приймає список, з можливими командами
    далі функція використовуючи алгоритм Левенштейна, визначае яка з можливих команд
    найбільше підходить до запитуваної команди, якщо не було досягнуто 50% схожості,
    то повертається False, якщо було досягнуто більше 50% схожості, повертається одна
    з можливих команд, з найбільшим відсотком схожості
    '''

    matches = []

    for command in possible_commands:
        match = fuzz.ratio(user_command, command)
        if match >= 50:
            matches.append(command)
        else:
            matches.append('None')

    best_match = max(matches, key=lambda match: fuzz.ratio(user_command, match))

    if best_match == 'None':
        return False

    return best_match

# Функція для відправлення команди на сервер розумного будинку
def send_command(command=str, url=str, source=str, language=str):

    '''
    Функція для відправлення команди, на сервер розумного будинку, на вхід приймає
    чотири аргумента, а повертає True якщо сервер відповів 200 (код успішної роботи).
    command - приймає рядок, з самою командою яку необхідно передати
    url - приймає рядок з адресою серверу, на який потрібно передати команду
    source - приймає рядок з джерелом (звідки була відправлена команда)
    language - приймає рядок з кодом мови, на якій була відправлена команда
    Повертає у основну програму True, якщо сервер відповів 200, або повідомлення про
    помилку, з кодом помилки
    '''
```

```
    command_data = {
    'command': command,
    'source': source,
    'language': language
    }

    try:
        response = requests.post(url, json=command_data)
    except Exception:
        if language_func(change=False) == 'en':
            return 'Error'
        else:
            return 'Помилка'

    if response.status_code == 200:
        return True
    else:
        return f'Error when sending a command - {response.status_code}'

# Функція для відповіді до користувача з використанням синтезу голосу
def answer_to_user(command=str, language=str):

    '''
    Функція синтезує mp3 файл, з озвученням переданого рядку тексту, і
    відтворює синтезований файл, в аудіо потік користувача.
    command - рядок з самою командою для якої потрібно синтезувати озвучення
    language - рядок з кодом мови, на якій потрібно синтезувати озвучення
    Функція працює у режимі процедури, не повертає нічого у основну програму,
    натомість запускає mp3 файл, з озвученням у аудіопотік користувача
    '''

    tts = gTTS(command, lang=language)
    tts.save('answer_for_user.mp3')
    answer = AudioSegment.from_mp3('answer_for_user.mp3')
    play(answer)

# Клас потіку для основної кнопки, що наслідується від классу QThread
бібліотеки pyqt6
class MainButtonThread(QThread):
    finished = pyqtSignal()

    # Ініціалізатор, створює необхідно змінні для екземпляру класа
    def __init__(self):
        super().__init__()
        self.checked = False
```

```
        self.is_running = True
        self.language = language_func(change=False)

    # Слот для передачі даних зміної checked потіку основної кнопку,
    # з потіку основного вікна
    @pyqtSlot(bool)
    def update_checked(self, checked):
        self.checked = checked

    #  для передачі сигналу щодо зупинки роботи потіку основної кнопки
    def stop(self):
        self.is_running = False

    # Основна функція потіку основної кнопки
    def run(self):
        while self.is_running:
            sleep(0.0001)

            # Перевіряємо стан кнопки, та налаштовану мову додатку
            # якщо встановлена англійська мова, починається виконання
            # цієї частини коду
            if self.checked and (self.language == 'en'):
                rec = sr.Recognizer()
                try:
                    #створюємо контекстний менеджер для об'єкту мікрофона
                    with sr.Microphone() as source:

                        # Очікуємо фразу активації, шляхом зчитування аудіопотоку
                        # шматками по 3 секунди, зчитування починається тільки при
                        # фіксації мови
                        window.label_update('Waiting for the activation phrase')
                        audio = rec.listen(source, timeout=2, phrase_time_limit=3)
                        if self.checked:
                            audio = rec.recognize_google(audio, language='en-US')
                        else:
                            continue

                        # Перевіряємо шматок тексту, на ідентичність до необхідної
                        # фрази активації, якщо була визначена фраза активації починаємо
                        # виконання коду для отримання основної команди
                        if 'hello home' in audio and self.checked:

                            #програємо короткий звук повідомлення (щодо фіксації фрази
активації)
                            play(AudioSegment.from_mp3('listening_notification.mp3'))
```

```
                    # Запускаємо цикл в якому будемо отримувати основні команди з очікуванням до 5
                     # секунд (якщо після початку очікування пройде 5 секунд, асистент знову почне
                    # очікувати фразу активації) у цей проміжок до 5 секунд, можна передавати почергово
                    # основні команди, після передачі команди, таймер оновлюється, і у вас є ще 5 секунд
                     # для передачі чергової команди
                     activate = True
                     while activate:
                        activate = False
                        window.label_update('Waiting for the main command')
                        audio = rec.listen(source, timeout=5, phrase_time_limit=5)
                        if self.checked:
                            audio = rec.recognize_google(audio, language='en-US')
                        else:
                            continue

                       # Визначаємо підходящу команду, з використанням функції command_match
                        matching_command = command_match(audio, en_possible_commands)

                        # Якщо команда визначена, відправляємо її до серверу
                        if matching_command and matching_command != 'Stop':
                         window.label_update(f'Sending <{matching_command}> to the servera')
                            response = send_command(matching_command, url, source='Smart home voice control system', language='en')

                            # Якщо сервер відповів позитивно, то сповіщуємо користувача, програючі
                            # синтезований голос, використовуючі функцію answer_to_user
                            if response == True:
                                window.label_update(f'Command {matching_command} successfully sent')
                                answer_to_user(f'Command {matching_command} successfully sent', 'en')
                                activate = True
                            else:
```

```
                        # Якщо сервер відповів не позитивно, також сповіщуємо користувача
                        # щодо помилки під час відправлення команди
                        window.label_update(response)
                        answer_to_user(f'Error while sending the command', 'uk')
                        continue
                    elif matching_command == 'Stop':

                        # Якщо не було знайдено підходящу команду під час перевірки
                        # сповіщуємо користувача синтезом голосу
                        window.label_update('Stoping')
                        answer_to_user('Stoping', 'en')
                        continue
                    else:
                        window.label_update('There is no such command')
                        answer_to_user('There is no such command', 'en')
                        continue

            except sr.UnknownValueError:

                # Обробляємо помилки, якщо під час спроби розпізнавання мови сервіс не зміг
                # визначити текст, відловлюємо цю помилки, та замість критичного вимикання програми
                # сповіщуємо користувача на секунду виводячі текст про помилку у вигляді об'єкту надпису
                # на головному окні програми
                window.label_update("Google Speech Recognition could not understand audio")
                sleep(1)
            except sr.WaitTimeoutError:

                # Якщо отримана помилка щодо завершення слухання відрізку аудіо, то пропускаємо помилку, та
                # починаємо головний цикл спочатку
                pass
            except sr.RequestError as e:

                # Якщо була отримана помилка отримання результатів від сервісу для розпізнавання мови
                # сповіщуємо користувача через об'єкт надпису на головному вікні
                window.label_update(f'Could not request results from Google Speech Recognition service; {e}')
                sleep(1)
```

```
        # Частина коду ідентична до попередньої, але реалізує всі дії на українській мові, на початку
        # перевіряємо налаштовану поточну мову, якщо вона українська, починаємо виконути точно ті ж самі дії
        # що у попередній частині коду, але на українській мові
        elif self.checked and (self.language == 'uk'):
            rec = sr.Recognizer()
            try:
                with sr.Microphone() as source:
                    window.label_update('Очікування фрази активації')
                    audio = rec.listen(source, timeout=2, phrase_time_limit=3)
                    if self.checked:
                        audio = rec.recognize_google(audio, language='uk-UA')
                    else:
                        continue

                    if 'Привіт будинок' in audio and self.checked:
                        play(AudioSegment.from_mp3('listening_notification.mp3'))
                        activate = True
                        while activate:
                            activate = False
                            window.label_update('Очікування основної команди')
                            audio = rec.listen(source, timeout=5, phrase_time_limit=5)
                            if self.checked:
                                audio = rec.recognize_google(audio, language='uk-UA')
                            else:
                                continue

                            matching_command = command_match(audio, uk_possible_commands)

                            if matching_command and matching_command != 'Зупинка':
                                window.label_update(f'Надсилання <{matching_command}> до серверу')
                                response = send_command(matching_command, url, source='Система голосового управління розумним будинком', language='uk')
                                if response == True:
                                    window.label_update(f'Команда {matching_command}, успішно надіслана')
                                    answer_to_user(f'Команда {matching_command}, успішно надіслана', 'uk')
                                    activate = True
                                else:
                                    window.label_update(response)
```

```
                        answer_to_user(f'Помилка відправлення команди', 'uk')
                        continue
                elif matching_command == 'Зупинка':
                    window.label_update('Зупинка...')
                    answer_to_user('Зупинка', 'uk')
                    continue
                else:
                    window.label_update('Такої команди нема')
                    answer_to_user('Такої команди нема', 'uk')
                    continue

            except sr.UnknownValueError:
                window.label_update("Помилка не вдалося розпізнати")
                sleep(1)
            except sr.WaitTimeoutError:
                pass
            except sr.RequestError as e:
                window.label_update(f'Помилка отримання відповіді від сервісу
розпізнавання мови {e}')
                sleep(1)

        self.finished.emit()

# Класс головного вікна програми, що наслідується від класу QMainWindow
# бібліотеки PyQt6
class MainWindow(QMainWindow):

    # Ініцілізатор/конструктор, створює необхідні віджети та з'єднує їх між собою
    def __init__(self):
        super().__init__()

        # Зберігаємо поточне налаштування мови у змінну
        self.language = language_func(change=False)

        # Встановлення назви головного вікна програми, розмірів
        self.setWindowTitle("Home Voice Assistant")
        self.setFixedSize(QSize(300, 180))

        # Залежно від налаштованої мови створюємо основну кнопку та основний
напис
        if self.language == 'en':
            self.mainbutton = QPushButton("Activate")
            self.info_label = QLabel('Sleeping...')
        else:
            self.mainbutton = QPushButton("Активувати")
```

```
            self.info_label = QLabel('Очікування...')

        # Створення параметрів основних параметрів для головної кнопки,
        # властивість натискання, розміри
        self.mainbutton.setCheckable(True)
        self.mainbutton.setFixedSize(QSize(100, 80))

        # З'єднання головної кнопки з функцією яку вона буде виконувати
        # залежно від стану натиснуто/відпущено
        self.mainbutton.toggled.connect(self.the_mainbutton_was_checked)


        # Створення кнопки налаштувань, передача параметрів, розміри, стиль,
        # замість тексту використовується картинка шестерні
        self.settings_button = QPushButton()
        self.settings_button.setFixedSize(QSize(17, 17))
        icon = QIcon('settings.png')
        self.settings_button.setStyleSheet('background-color: transparent; border: none;')
        self.settings_button.setIcon(icon)
        self.settings_button.setIconSize(self.settings_button.size())

        # З'єдання кнопки налаштувань з функцією яку вона буде виконувати
        # залежно від події кліку на кнопку
        self.settings_button.clicked.connect(self.settings_button_click)


        # Створення інформаційної кнопки, передача параметрів, розміри, стиль,
        # картинка літери 'i'
        self.info_button = QPushButton()
        self.info_button.setFixedSize(17, 17)
        icon = QIcon('info_button_icon.png')
        self.info_button.setStyleSheet('background-color: transparent; border: none;')
        self.info_button.setIcon(icon)
        self.info_button.setIconSize(self.info_button.size())

        # З'єдання інформаційної кнопки з функцією яку вона буде виконувати
        # залежно від події кліку на кнопку
        self.info_button.clicked.connect(self.info_button_click)

        # Створення надпису що відображає поточну мову додатку, передача
параметрів
        self.lang_label = QLabel(self.language)

       # Передача параметрів основному надпису який відображає поточний етап
роботи
        # додатку
```

```
        self.info_label.setFixedSize(100, 100)
        self.info_label.setWordWrap(True)

        # Створення сітчатого макету, для розміщення усіх інших віджетів
        self.gridLayout = QGridLayout()

        # Створення віджету заповнювачу, для конфігурації росположення
основних віджетів
        # у сітчатому макеті
        spacer = QSpacerItem(40, 20, QSizePolicy.Policy.Expanding,
QSizePolicy.Policy.Expanding)

        # Додавання всіх віджетів до макету, на необхідні місця
        self.gridLayout.addWidget(self.mainbutton, 1, 1, 1, 2)
        self.gridLayout.addWidget(self.settings_button, 0, 4)
        self.gridLayout.addWidget(self.info_button, 0, 0)
        self.gridLayout.setAlignment(self.settings_button, Qt.AlignmentFlag.AlignRight)
        self.gridLayout.addWidget(self.info_label, 1, 4)
        self.gridLayout.addWidget(self.lang_label, 2, 0)

        # Додавання заовнювачів для конфігурації віджетів у макеті
        self.gridLayout.addItem(spacer, 0, 0, 3, 1)
        self.gridLayout.addItem(spacer, 0, 3, 3, 1)

        # Створення віджету контейнеру, для сітчатого віджету
        self.widget = QWidget()
        self.widget.setLayout(self.gridLayout)

        # Встановлення віджету контейнеру у якості центрального віджету вікна
        self.setCentralWidget(self.widget)

    # Функція для основної кнопки, викликається при зміні стану основної кнопки
    # натиснуто/відпущено
    def the_mainbutton_was_checked(self, checked):

        # Зміна тексту основної кнопки, та основного надпису залежно
        # від налаштованої мови
        if self.language == 'en':
            if checked:
                self.mainbutton.setText('Activated')
            else:
                self.mainbutton.setText('Activate')
                self.label_update('Sleeping...')
        else:
            if checked:
```

```
            self.mainbutton.setText('Активовано')
        else:
            self.mainbutton.setText('Активувати')
            self.label_update('Очікування...')

    # Передача стану основної кнопки натиснуто/відпущено,
    # до потіку основної кнопки
    main_button_thread.update_checked(checked)

# Функція для інформаційної кнопки, викликається під час
# кліку на інформаційну кнопку
def info_button_click(self):

    # Формується та виводиться інформаційне повідомлення
    # у діалогове вікно MessageBox від поточної мови
    if self.language == 'en':
        info = 'Activation phrase: Hello home;\n\nAvailable commands:\n'
        for el in en_possible_commands:
            info = info + '- ' + el + ';\n'

        QMessageBox.information(None, 'Information', info)
    else:
        info = 'Фраза активації: Привіт будинок;\nДоступні команди:\n'
        for el in uk_possible_commands:
            info = info + '- ' + el + ';\n'

        QMessageBox.information(None, 'Інформація', info)

# Функція для кнопки налаштувань, викликається під час натискання на
кнопку налаштувань
def settings_button_click(self):

    # Створення об'єкту діалогового вікна
    self.dialog = QDialog()

    # Створення об'єкту списку що випадає
    dialog_combo_box = QComboBox()

    # Створення надпису, кнопки, та назви діалогового вікна
    # передача параметрів згідно налаштуванню поточної мови
    if self.language == 'en':
        dialog_label = QLabel('Select language:')
        dialog_button = QPushButton('Ok')
        self.dialog.setWindowTitle('Settings')
    else:
```

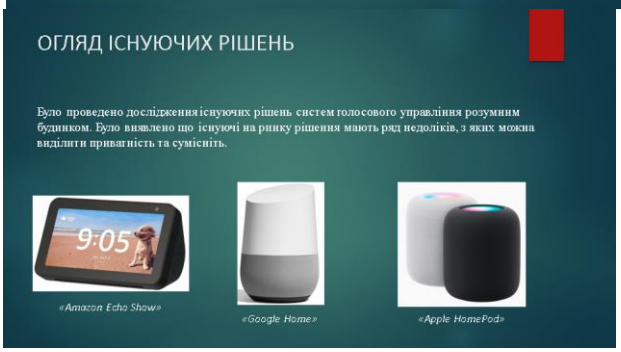
```
            dialog_label = QLabel('Оберіть мову:')
            dialog_button = QPushButton('Ок')
            self.dialog.setWindowTitle('Налаштування')

        # Локальна функція для кнопки діалогового вікна
        # що виклакається під час натискання на кнопку
        def d_button_click():

            # Отримання поточної мови
            c_lang = language_func(change=False)

            # Отримання встановленої користувачем мови у ComboBox
            lan = dialog_combo_box.currentText()

            # Зміна/налаштування поточної мови, згідно вибору користувача
            if c_lang == 'en' and lan == 'Українська':
                language_func(change=True, lang='uk')
                self.dialog.close()
                QMessageBox.warning(None, 'Warning', 'To apply the selected settings,
restart the application')

                self.lang_label.setText('uk')
                self.lang_label.setStyleSheet('QLabel { color: red; }' 'QToolTip { color:
black; background-color: white; }')
                self.lang_label.setToolTip('To apply the language settings, restart the
application')

            elif c_lang == 'uk' and lan == 'English':
                language_func(change=True, lang='en')
                self.dialog.close()
                QMessageBox.warning(None, 'Попередження', 'Для застосування
вибраних налаштувать, перезапустіть додаток')

                self.lang_label.setText('en')
                self.lang_label.setStyleSheet('QLabel { color: red; }' 'QToolTip { color:
black; background-color: white; }')
                self.lang_label.setToolTip('Для застосування налаштувань мови,
перезапустіть додаток')
            else:
                self.dialog.close()

        # З'єдання кнопки діалогового вікна з функцією, передача параметрів
        dialog_button.clicked.connect(d_button_click)
        dialog_button.setFixedSize(60, 20)
```

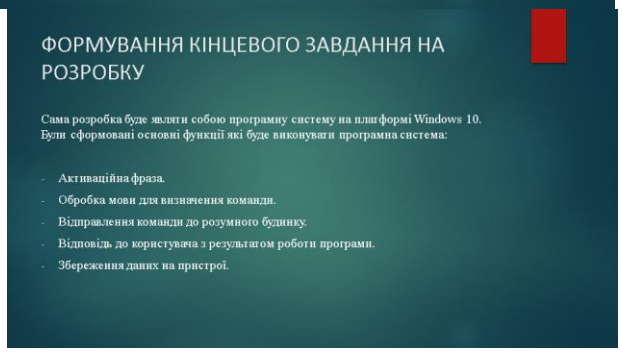
```
        # Додаваня об'єктів до списку що випадає в залежності від встановленої мови
        if language_func(change=False) == 'en':
            dialog_combo_box.addItems(['English', 'Українська'])
        else:
            dialog_combo_box.addItems(['Українська', 'English'])

        # Створення сітчатого макету для розміщення інших віджетів на діалоговому вікні
        gridLayout = QGridLayout()

        # Додавання віджетів до сітчатого макету на потрібні позиції
        gridLayout.addWidget(dialog_label, 0, 0)
        gridLayout.addWidget(dialog_combo_box, 0, 1)
        gridLayout.addWidget(dialog_button, 1, 1)

        # Встановлення сітчатого макету з віджетами у якості основного макету діалогового вікна
        self.dialog.setLayout(gridLayout)

        # Відображення діалогового вікна
        self.dialog.show()

    # Модифікація основної функції наслідуючого класу, для закриття всіх далогових вікон, після
    # закриття програми
    def closeEvent(self, event):
        for widget in QApplication.topLevelWidgets():
            widget.close()

        event.accept()

    # Слот для передачі даних, з інших місць програми, до основного надпису на головному вікні
    @pyqtSlot(str)
    def label_update(self, label):

        # Оновлення головного надпису, згідно поточного етапу роботи потіка основної кнопки
        self.info_label.setText(label)


# Створення екземпляру QApplication
app = QApplication(sys.argv)
```

```
# Створення екземпляру головного вікна
window = MainWindow()
window.show()

# Створення екземпляру потіку головної кнопки
main_button_thread = MainButtonThread()
main_button_thread.finished.connect(lambda x: print('Main thread stop working'))
main_button_thread.start()

# Запуск головного циклу подій QAplication
app.exec()
```

ДОДАТОК 2

ПРЕЗЕНТАЦІЙНІ МАТЕРІАЛИ ДО ДИПЛОМНОГО ПРОЕКТУ





Для створення зумовленої завданням системи була обрана мова програмування Python. Для реалізації зумовлених завданням функцій були обрані наступні бібліотеки на мові програмування Python.

- PyQt6 для інтерфейсу користувача.
- SpeechRecognition для обробки мови
- Requests для відправлення команд до розумного будинку
- gTTS для формування відповіді до користувача

Логотип Python

Логотип Requests

РОЗРОБКА ПРОГРАМИ

Програма складається з двох основних класів:

- MainWindow
- MainButtonThread

Та з ряду допоміжних функцій, таких як:

- language_func
- command_match
- send_command
- answer_to_user

КЛАС MainWindow

КЛАС MainButtonThread

ФУНКЦІЯ language_func

ФУНКЦІЯ command_match

ФУНКЦІЯ send_command

ФУНКЦІЯ answer_to_user

СЕРВЕР НА flask

ЗАПУСК РОЗРОБКИ

ІНТЕРФЕЙС РОЗРОБКИ

ДЕМОНСТРАЦІЯ

На цьому моменті перейдемо до демонстрації роботи розробленого застосунку.

ДЯКУЮ ЗА УВАГУ!

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

РЕЦЕНЗІЯ

на дипломний проект (роботу) здобувача (здобувачки) освіти

відділення комп'ютерних систем

Кодряну Павла Михайловича

(прізвище, ім`я та по батькові)

Спеціальність **123 "Комп'ютерна інженерія"**

Освітня програма **Обслуговування комп'ютерних систем та мереж**

Керівник дипломного проекту (роботи) **к.пед.н. Скорнякова Олена Володимирівна**

(прізвище, ім`я та по батькові)

Тема дипломного проекту (роботи):

Проектування системи голосового управління «розумним будинком»

Обсяг розрахунково-пояснювальної записки _____ сторінок

Обсяг графічної (презентаційної) частини _____ аркушів (слайдів)

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ (РОБОТИ)

а) заключення про ступінь відповідності виконаного дипломного проекту (роботи) завданню

Дипломний проект повністю відповідає завданню до дипломного проектування

б) характеристика виконання кожного розділу дипломного проекту (роботи)

Пояснювальна записка дипломного проекту виконана якісно, у повному обсязі. В дипломному проекті здобувачем проведено детальний аналіз існуючих рішень. Конкретизовано на основі проведеного аналізу вимоги до дипломного проекту, визначено завдання та визначено технічні рішення, що дозволяють реалізувати завдання дипломного проекту, здійснено вибір елементної бази та програмних інструментів для реалізації системи

в) оцінка якості виконання пояснювальної записки та графічної частини дипломного проекту (роботи)

Презентаційні матеріали виконані якісно, демонстративно та відповідають вмісту теоретичного матеріалу

г) перелік позитивних якостей дипломного проекту (роботи)

Тематика дипломного проекту є актуальною. Серед позитивних якостей – детальний аналітичний огляд існуючих рішень, виважений підхід до реалізації завдань до дипломного проекту та реалізації системи

д) основні недоліки дипломного проекту (роботи)

Варто було б в пояснювальній записці представити алгоритми функціонування системи голосового управління «розумним будинком». Не аргументовано вибір саме ОС Windows 10.

Оцінка розрахункової частини Відмінно

Оцінка графічної частини Відмінно

Загальна оцінка Відмінно

Прізвище, ім'я, по батькові рецензента **Василіу Євген Вікторович**

Місце роботи і посада рецензента **Державний університет інтелектуальних технологій і зв'язку, д.т.н., проф. кафедри КБ та ТЗІ, декан факультету інформаційних технологій та кібербезпеки**

Підпис:

« 16 » 06 2023 р.

ПІДПИС ПОСВІДЧУЮ
НАЧАЛЬНИК ВІДДІЛУ КАДРІВ ДУІТЗ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ * ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНТЕЛЕКТУАЛЬНИХ ТЕХНОЛОГІЙ І ЗВ'ЯЗКУ * 43997335 * ВІДДІЛ КАДРІВ

ВСП «Одеський технічний фаховий коледж ОНТУ»

ВІДГУК

Керівника на дипломний проект здобувача освіти

Кодряну Павла Михайловича

(прізвище, ім`я та по батькові)

Спеціальність: ***123 «Комп'ютерна інженерія»***

Тема дипломного проекту: ***Проектування системи голосового управління «розумним будинком***

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ

а) Обсяг і якість виконання проекту (графічного матеріалу і розрахунково-пояснювальної записки): *Пояснювальна записка дипломного проекту виконана якісно, у повному обсязі. В дипломному проекті здобувачем проведено огляд існуючих технологій та аналогів пристрою, що пропонується до створення, визначено структуру створюваної системи, розроблено програмну частини системи, описані принципи роботи системи та розроблено вдповідне програмне забезпечення. В дипломному проекті в останніх розділах проаналізовано питання економічної доцільності та охорони праці. Створено презентацію до захисту.*

б) Самостійність роботи над проектом: *Здобувач самостійно визначався з напрямом роботи, дослухався до рекомендацій керівника дипломного проекту, своєчасно надавав результати роботи, якісно виконував основні етапи роботи за вимогою керівника.*

в) Теоретична підготовка випускника: *Теоретична підготовка випускника в цілому відповідає державним вимогам до фахівців відповідного рівня кваліфікації*

г) Вміння розв`язувати виробничі і конструкторські питання на базі останніх досліджень науки і техніки, передових методів виробництва_____

В процесі роботи над дипломним проектом здобувач продемонстрував уміння використовувати останні досягнення науки та техніки в предметній галузі на підставі відповідної навчальної та науково-технічної літератури, достатньо впевнено користувався програмним забезпеченням при роботі над дипломним проектом та створенням презентації.

Оцінка розрахункової частини _____*відмінно*_____

Оцінка графічної частини_____*відмінно*_____

Загальна оцінка_____*відмінно*_____

Прізвище, ім`я, по батькові__*Скорнякова Олена Володимирівна*_____

_____

Місце роботи і посада керівника дипломного проекту: к.пед.н., викладач-методист комісії КТ та ПІ_ВСП «Одеський технічний фаховий коледж Одеського національного технологічного університету»_____

Підпис _____

«12» 06 2023 р.

UNICHECK

Ім'я користувача:
Наталія Вікторівна Копусь

ID перевірки:
1015543161

Дата перевірки:
10.06.2023 17:21:54 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
10.06.2023 17:38:19 EEST

ID користувача:
100011688

Назва документа: **4КС-56 Кодряну П.**

Кількість сторінок: **58** Кількість слів: **11521** Кількість символів: **84117** Розмір файлу: **2.38 MB** ID файлу: **1015195857**

19.4%
Схожість

Найбільша схожість: **3.32%** з Інтернет-джерелом (https://filesclub.net/5600605)

19.4% Джерела з Інтернету	912	..... Сторінка 60

Не знайдено джерел з Бібліотеки

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	18

ДОЗВІЛ
НА РОЗМІЩЕННЯ
ВИПУСКНОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
В ЕЛЕКТРОННОМУ РЕПОЗИТАРІЇ ВСП «ОТФК ОНТУ»

Ми, що нижче підписалися,

Кодряну Павло Михайлович,
здобувач освіти гр. 4КС-56, та

Скорнякова Олена Володимирівна,
керівник дипломного проекту,

не заперечуємо щодо розміщення електронного варіанту пояснювальної записки до випускної кваліфікаційної роботи молодшого спеціаліста на тему:

«Проектування системи голосового управління «розумним будинком» (автор роботи – Кодряну П.М., керівник роботи – Скорнякова О.В.)

виконаного у ВСП «Одеський технічний фаховий коледж Одеського національного технологічного університету» в 2023 році, у повному обсязі в електронному репозитарії ВСП «ОТФК ОНТУ» для вільного доступу через мережу Інтернет.

Несемо відповідальність за ідентичність електронного та друкованого варіантів випускної кваліфікаційної роботи, і даємо згоду на обробку персональних даних.

Виконавець _____ / Кодряну П.М./

Керівник _____ / Скорнякова О.В./

«12» 06 2023 р.