

Министерство образования Республики Беларусь
Учреждение образования «Белорусский государственный университет
информатики и радиоэлектроники»

Факультет компьютерного проектирования
Кафедра проектирования информационно-компьютерных систем
Дисциплина «Объектно-ориентированное программирование»

«К ЗАЩИТЕ ДОПУСТИТЬ»
Руководитель курсового проекта
магистр техники и технологии,
ассистент А.С. Гриб
____.____.2022

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовому проекту
на тему:
ПРОГРАММНОЕ СРЕДСТВО «ГОЛОСОВОЙ АССИСТЕНТ»

БГУИР КП 1-40 05 01-10 018 ПЗ

Выполнил студент группы 014301
НЕВЕЙКОВ Андрей Сергеевич

(подпись студента)

Курсовой проект представлен на
проверку ____ . ____ . 2022

(подпись студента)

Минск 2022

РЕФЕРАТ

БГУИР КП 1-40 05 01-10 018 ПЗ

Невейков, А.С. Программное средство «Голосовой ассистент»/ А.С. Невейков. – Минск: БГУИР, 2022. – 52с.

Пояснительная записка 52 с., 16 рис., табл., источников, приложения
**ГОЛОСОВОЙ АССИСТЕНТ, ПРИЛОЖЕНИЕ, ГОЛОСОВАЯ
КОМАНДА, УПРАВЛЕНИЕ КОМПЬЮТЕРОМ, ПОЛЬЗОВАТЕЛЬСКИЙ
ИНТЕРФЕЙС, ХРАНЕНИЕ ПОЛЬЗОВАТЕЛЬСКИХ ДАННЫХ**

Предмет: голосовой ассистент.

Объект: процесс взаимодействия пользователя с программой.

Цель: разработать программное средство для выполнения голосовых команд пользователя, хранения пользовательских данных, вывода информации при помощи сгенерированного голоса и визуального интерфейса программы.

Методология проведения работы: в процессе разработки программного средства применялись методы преобразования человеческого голоса в текст, анализа преобразованных команд для выполнения соответствующих функций, работы с базой данных при помощи SQL-команд.

Результаты работы: на основе анализа литературных источников; разработано программное средство для взаимодействия с пользователем при помощи голосовых команд; спроектирован графический пользовательский интерфейс для удобства отображения результатов выполнения команд и отображения их результатов пользователю; создана база данных для хранения персональных данных пользователя.

Учреждение образования
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»
Факультет компьютерного проектирования
Кафедра проектирования информационно-компьютерных систем

УТВЕРЖДАЮ
Заведующий кафедрой ПИКС

В.В.Хорошко
« ____ » февраля 2022

ЗАДАНИЕ
к курсовому проекту по дисциплине
«Объектно-ориентированное программирование»

Группа 014301

Студенту Невейкову Андрею Сергеевичу

1.Тема проекта: Программное средство «Голосовой ассистент».

2.Сроки сдачи студентом законченного проекта: 14-20.05.2022 г.

3.Исходные данные к проекту:

3.1. Описание системы – клиентское программное средство с базой данных.

3.2. Назначение системы – упростить процесс взаимодействия с персональным компьютером при помощи выполнения голосовых команд

3.3.Язык и среда программирования – на выбор студента. Разработанное программное обеспечение должно быть реализовано на объектно-ориентированном языке.

3.4.Нормативные источники: 3.4.1.Положение о курсовом проектировании БГУИР.
3.4.2. СТП 01-2017. Стандарт предприятия. Дипломные проекты (работы). Общие требования. 3.4.3. ГОСТ 19.701-90 ЕСПД. Схемы алгоритмов, программ, данных и систем. Обозначения условные и правила выполнения.

4.Содержание расчетно-пояснительной записки (перечень подлежащих разработке вопросов):

Титульный лист. Реферат. Задание. Содержание. Перечень условных обозначений, символов и терминов.

Введение (1 – 2 стр. Актуальность темы курсового проекта; цель и перечень задач, которые планируется решить; детальная постановка задачи).

4.1. Анализ исходных данных на курсовое проектирование. 4.1.1. Анализ исходных данных к курсовому проекту. 4.1.2. Обоснование и описание выбора языка программирования, средств разработки, используемых технологий и сторонних библиотек.

4.2. Проектирование и разработка программного средства. 4.2.1. Проектирование архитектуры и описание состояний программного средства. 4.2.2. Объектная модель программного средства. 4.2.3. Проектирование и разработка графического интерфейса. 4.2.4. Описание и реализация используемых в программном средстве алгоритмов.

4.3. Эксплуатация программного средства. 4.3.1. Ввод в эксплуатацию и обоснование минимальных технических требований к оборудованию. 4.3.2. Руководство по эксплуатации программного средства.

Заключение (1 стр. Выводы по курсовому проекту).

Список литературных источников.

Приложения (листинг программного кода; справка о проверке курсового проекта на плагиат; ведомость курсового проекта).

5.Перечень графического материала (с указанием обязательных чертежей и графиков):

5.1.Схема алгоритма (формат А2/А3).

5.2.UML диаграмма классов (формат А2/А3).

5.3.Диаграмма последовательности (формат А2/А3).

5.4.Диаграмма состояний (формат А2/А3).

5.5.Структура графического пользовательского интерфейса (формат А2/А3).

6.Консультанты по проекту: старший преподаватель ГОРБАЧ Антон Петрович (ауд. 415а-1 корп.), ассистент ГРИБ Александр Сергеевич (ауд. 415-1 корп.).

7.Дата выдачи задания: 11.02.2022 г.

8.Календарный график работы над проектом на весь период проектирования (с указанием сроков выполнения и трудоемкости отдельных этапов):

№ п/п	Наименование этапов курсового проекта	Срок выполнения этапов проекта	Примечание
1.	1-я опрoцентoвка (4.1, 5.1)	09-11.03.2022	30%
2.	2-я опрoцентoвка (4.2, 5.2-5.3)	06-08.04.2022	60%
3.	3-я опрoцентoвка (введение, 4.3, 5.4-5.5, заключение)	04-07.05.2022	80%
4.	Сдача курсового проекта на проверку	14-20.05.2022	100%
5.	Защита курсового проекта	25.05-03.06.2022	Согласно графику

Руководитель

_____ А.С. Гриб

Задание принял к исполнению 11.02.2022

(_____)

(подпись студента)

(расшифровка подписи)

СОДЕРЖАНИЕ

Реферат	2
перечень условных обозначений и сокращений	6
Введение.....	7
1 АНАЛИЗ ИСХОДНЫХ ДАННЫХ НА КУРСОВОЕ ПРОЕКТИРОВАНИЕ	8
1.1 Анализ исходных данных к курсовому проекту	8
1.2 Обоснование и описание выбора языка программирования, средств разработки, используемых технологий и сторонних библиотек.	9
2 ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА	11
2.1 Проектирование архитектуры и описание состояний программного средства	11
2.2 Объектная модель программного средства	13
2.3 Проектирование и разработка графического интерфейса	16
2.4 Описание и реализация используемых в программном средстве алгоритмов	19
3 ЭКСПЛУАТАЦИЯ ПРОГРАММНОГО СРЕДСТВА	22
3.1 Ввод в эксплуатацию и обоснование минимальных технических требований	22
3.2 Руководство по эксплуатации программного средства	23
Заключение	28
Список использованных источников	29

ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧЕНИЙ И СОКРАЩЕНИЙ

В настоящей пояснительной записке применяются следующие термины, обозначения и сокращения.

Аутентификация – процедура проверки подлинности.

Аутентификационные данные – это данные, по которым сервис идентифицирует вас как пользователя.

База данных – упорядоченный набор структурированной информации или данных, которые обычно хранятся в электронном виде в компьютерной системе.

Интерфейс – метод организации взаимодействия между отдельными системами.

Голосовой ассистент – это сервис, основанный на искусственном интеллекте, распознающий человеческую речь и выполняющий различные действия, в ответ на голосовые команды.

Библиотека – сборник подпрограмм или объектов, используемых для разработки программного обеспечения.

Модуль – это отдельная функционально осмысленная и законченная программная единица (подпрограмма), которая обеспечивает решение некоторой задачи и в идеале может работать самостоятельно или в другом окружении и быть переиспользуемой.

БД – база данных.

ЭВМ – электронно-вычислительная машина.

ПО – программное обеспечение.

ОС – операционная система.

ЛКМ – левая кнопка мыши.

ПКМ – правая кнопка мыши.

ВВЕДЕНИЕ

На выполнение промежуточных действий при работе с ЭВМ: ввод поисковых запросов, поиск музыки или видео в интернете может уходить значительное количество времени, а запомнить различные пароли от множества интернет-ресурсов для человека сложно, из-за чего люди используют один пароль для всех сайтов, что небезопасно. Разработанное программное средство решает эти проблемы при помощи обработки и выполнения голосовых команд. Пользователю не нужно вручную набирать запросы в поисковой строке, вместо этого нужно сказать голосовому ассистенту выполнить определенное действие, что гораздо быстрее. Для хранения пользовательских данных реализована база данных с возможностью вывода данных по названию ресурса.

В курсовой работе поставлена цель: проектирование программного средства для упрощения взаимодействия пользователя с ЭВМ при помощи выполнения голосовых команд, оповещения пользователя об успешности их выполнения, возможности использования спроектированного интерфейса для работы в интернете и встроенного хранилища для аутентификационных данных.

Особенно актуально применение данного программного средства для людей с нарушением мелкой моторики рук (которым сложно набирать текст на клавиатуре) и для представителей старшего поколения, которым тяжело запомнить расположение программ и данные для аутентификации.

Для достижения поставленной цели необходимо решить следующие задачи:

- 1 Спроектировать графический пользовательский интерфейс.
- 2 Создать функции приема и обработки голосовых команд.
- 3 Реализовать функции, которые чаще всего использует пользователь.
- 4 Добавить механизм выбора нужной функции.
- 5 Спроектировать механизм ответного взаимодействия программы.
- 6 Организовать возможность отменить команду.
- 7 Подключить базу данных.
- 8 Реализовать пользовательский интерфейс для взаимодействия с базой данных.

1 АНАЛИЗ ИСХОДНЫХ ДАННЫХ НА КУРСОВОЕ ПРОЕКТИРОВАНИЕ

1.1 Анализ исходных данных к курсовому проекту

Ритм и динамика жизни людей с каждым днем прогрессирует. У людей нет времени читать длинные посты, смотреть длинные видеоролики, у них даже нет времени писать текст. Поэтому люди все чаще пользуются голосовыми помощниками. По прогнозам Gartner [1] в скором времени 30% сеансов просмотра веб-страниц будут происходить без использования экранов, а 50% из всех запросов будут голосовыми.

Голосовые ассистенты дают возможность минимизировать, а иногда и вовсе устранить необходимость использовать руки и глаза для просмотра контента в интернете.

Голосовой помощник – это сервис, основанный на искусственном интеллекте и распознающий человеческую речь.

Голосовые помощники были созданы для того, чтобы люди не тратили лишнее время на простые ежедневные задачи. Функционал голосовых помощников достаточно обширен. Они могут:

1. Общаться с пользователем.
2. Искать информацию в интернете и коротко отвечать на запросы, поступающие от пользователя.
3. Вызывать такси.
4. Звонить, писать сообщения.
5. Включать музыку.
6. Составлять маршрут.
7. Заводить будильник.
8. Искать автозаправки поблизости.

Крупными компаниями уже представлены реализации голосовых ассистентов, таких как Siri, Яндекс Алиса, Google Assistant, Microsoft Cortana, Amazon Alexa, которые могут работать как на компьютерах и мобильных устройствах, так и являться частью умного дома, контролируя открывание жалюзи, дверей гаража, включение света и телевизора, настройку термостата и системы кондиционирования [2].

Основой для голосовых ассистентов является искусственный интеллект, который отвечает за распознавание речи и генерацию голоса ассистента. Когда ассистент распознает команду, он, по ключевым словам, определяет, какую функцию необходимо выполнить. Для обратной связи с пользователем могут использоваться голосовые ответы, чат или их комбинация. В данной работе реализован чат, вынесенный в левый нижний угол, но основу взаимодействия составляют голосовые ответы, что делает его похожим на Яндекс Алису.

Разработанное программное средство «Голосовой ассистент» предназначенное для управления компьютером при помощи голосовых команд способно запускать установленные приложения, вводить запросы в

поисковике, искать музыку и видео по названию, хранить, добавлять, удалять, выводить на экран данные для аутентификации, отвечать на вопросы пользователя, рассказывать случайные анекдоты.

1.2 Обоснование и описание выбора языка программирования, средств разработки, используемых технологий и сторонних библиотек.

Python проектировался как объектно-ориентированный язык программирования. Это означает, что он построен с учетом следующих принципов:

1. Все данные в нем представляются объектами.
2. Программу можно составить как набор взаимодействующих объектов, посылающих друг другу сообщения.
3. Каждый объект имеет собственную часть памяти и может состоять из других объектов.
4. Каждый объект имеет тип.
5. Все объекты одного типа могут принимать одни и те же сообщения (и выполнять одни и те же действия).

Выбор использования данного языка основан высокой скорости выполнения программ, написанных на нем. Это связано с тем, что основные библиотеки Python написаны на C++ и выполнение задач занимает меньше времени, чем на других языках высокого уровня. Так же упрощает разработку наличие большого количества подключаемых библиотек и модулей, предназначенных для обработки и генерации голоса. В программе использованы следующие библиотеки и модули:

1. Speech recognition — библиотека для передачи речевых API от компаний (google, microsoft, sound hound, ibm, а также pocketsphinx).
2. Threading — стандартная библиотека Python, которая содержит необходимые классы для работы с потоками.
3. Signal — Модуль, который предоставляет механизмы для использования обработчиков сигналов в Python (Сигналы - это функция операционной системы, которая позволяет уведомлять программу о событии и обрабатывать его асинхронно).
4. PyQt — набор расширений графического фреймворка Qt для языка программирования Python, выполненный в виде расширения Python.
5. OS — это библиотека функций для работы с операционной системой. Включенные методы позволяют определять тип операционной системы, получать доступ к переменным окружения, управлять директориями и файлами.
6. Sys — Модуль, который предоставляет программисту набор функций, которые дают информацию о том, как интерпретатор Python взаимодействует с операционной системой.
7. Time — библиотека для работы со временем.

8. Torch — библиотека с открытым исходным кодом, предоставляет большое количество алгоритмов для глубинного обучения и научных расчётов, написанная на языке C.

9. Sounddevice — библиотека, предоставляющая возможность записи аудио с помощью Python.

10. Urllib — это модуль Python, который можно использовать для открытия URL-адресов.

11. Subprocess — модуль, который дает разработчику возможность запускать процессы программ из Python.

12. Webbrowser — модуль, который предоставляет собой интерфейс (высокоуровневый), позволяющий просматривать веб-документы.

13. Requests — это HTTP-библиотека для языка программирования Python, используемая, чтобы сделать HTTP-запросы более простыми и удобными для программиста.

14. bs4 — это пакет Python для анализа документов HTML и XML. Он создает дерево синтаксического анализа для проанализированных страниц, которое можно использовать для извлечения данных из HTML, что полезно для парсинга веб-страниц.

15. re — модуль для работы с регулярными выражениями. Регулярные выражения — это язык для поиска, извлечения и работы с определенными текстовыми шаблонами большего текста. Он широко используется в проектах, которые включают проверку текста, NLP (Обработка естественного языка) и интеллектуальную обработку текста.

16. tkinter — это пакет для Python, предназначенный для работы с библиотекой Tk. Библиотека Tk содержит компоненты графического интерфейса пользователя.

17. Sqlite3 — это C библиотека, реализующая легковесную дисковую базу данных (БД), не требующую отдельного серверного процесса и позволяющую получить доступ к БД с использованием языка запросов SQL.

Так же преимуществом языка является его кроссплатформенность, т.е. скрипты, написанные при помощи Python выполняются на большинстве современных ОС. Такая переносимость обеспечивает Python применение в самых различных областях. Python подходит для любых решений в области программирования, будь то офисные программы, веб-приложения, GUI-приложения и т.д.

2 ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА

2.1 Проектирование архитектуры и описание состояний программного средства

Проектирование программного обеспечения – это процесс концептуализации требований к программному обеспечению в реализацию программного обеспечения.

Архитектура программного обеспечения — совокупность важнейших решений об организации программной системы. Архитектура включает: выбор структурных элементов и их интерфейсов, с помощью которых составлена система, а также их поведения в рамках сотрудничества структурных элементов [3].

Хорошая архитектура это прежде всего выгодная архитектура, делающая процесс разработки и сопровождения программы более простым и эффективным. Программу с хорошей архитектурой легче расширять и изменять, а также тестировать, отлаживать и понимать. То есть, на самом деле можно сформулировать список разумных и универсальных критериев [4]:

1. Эффективность системы: программа должна решать поставленные задачи и выполнять свои функции, причем в различных условиях. Сюда можно отнести такие характеристики, как способность справляться с увеличением нагрузки, надежность, безопасность, производительность.

2. Гибкость системы: чем быстрее и удобнее можно внести изменения в существующий функционал, чем меньше проблем и ошибок это вызовет — тем гибче и конкурентоспособнее система. Изменение одного фрагмента системы не должно влиять на ее другие фрагменты. Хорошая архитектура позволяет откладывать принятие ключевых решений и минимизирует «цену» ошибок.

3. Расширяемость системы: возможность добавлять в систему новые сущности и функции, не нарушая ее основной структуры. На начальном этапе в систему имеет смысл закладывать лишь основной и самый необходимый функционал, но при этом архитектура должна позволять легко наращивать дополнительный функционал по мере необходимости. Причем так, чтобы внесение наиболее вероятных изменений требовало наименьших усилий.

4. Масштабируемость процесса разработки: возможность сократить срок разработки за счёт добавления к проекту новых людей. Архитектура должна позволять распараллелить процесс разработки, так чтобы множество людей могли работать над программой одновременно.

5. Тестируемость: код, который легче тестировать, будет содержать меньше ошибок и надежнее работать.

6. Возможность повторного использования: систему желательно проектировать так, чтобы ее фрагменты можно было повторно использовать в других системах.

7. Сопровождаемость: хорошо структурированный, читаемый и понятный код. Хорошая архитектура должна давать возможность относительно легко и быстро разобраться в системе новым людям. Проект должен быть хорошо структурирован, не содержать дублирования, иметь хорошо оформленный код и желательно документацию. И по возможности в системе лучше применять стандартные, общепринятые решения привычные для программистов.

Не смотря на разнообразие критериев, все же главной при разработке больших систем считается задача снижения сложности по сути речь идет об декомпозиции. Сложная система должна строится из небольшого количества более простых подсистем, каждая из которых, в свою очередь, строится из частей меньшего размера, и так далее до тех пор, пока самые небольшие части не будут достаточно просты для непосредственного понимания и создания.

Для снижения сложности проектируемого программного средства была применена функциональная декомпозиция.

Деление на модули/подсистемы были произведены исходя из тех задач, которые решает система. Основная задача разбивалась на составляющие ее подзадачи, которые могут решаться независимо друг от друга. Каждый модуль отвечает за решение какой-то подзадачи и выполняет соответствующую ей функцию.

Модули спроектированы так, чтобы большую часть своих функций модуль мог выполнить самостоятельно, без помощи остальных модулей, лишь на основе своих входящих данных.

Таким образом, декомпозиция была основана, прежде всего, на анализе функций системы и необходимых для выполнения этих функций данных.

Связь между модулями осуществляется при помощи паттерна проектирования «Посредник (Mediator)».

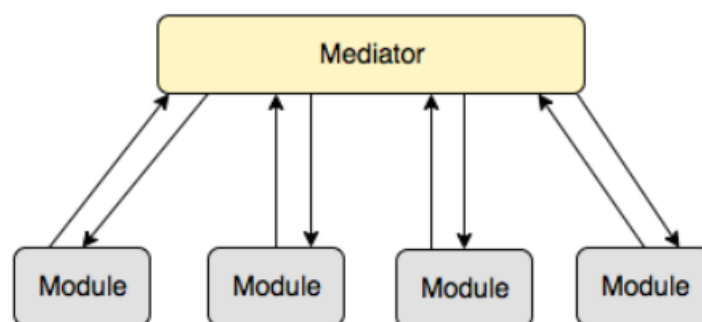


Рис. 2.1 – Визуализация паттерна проектирования «Посредник» [4]

Медиатор выступает в качестве посредника в общении между модулями, действуя как центр связи и избавляет модули от необходимости явно ссылаться друг на друга. В результате взаимодействие модулей происходит лишь с посредником («один со всеми»). Говорят, что посредник инкапсулирует взаимодействие между множеством модулей.

Таким образом была спроектирована эффективная, расширяемая программа, с возможностью масштабирования, удобная для тестирования, написанная хорошо структурированным, понятным кодом.

2.2 Объектная модель программного средства

Выполнения приложения начинается в модуле Assistant_main_file.py. В нем, реализован основной пользовательский интерфейс, описанный в классе ProgramWindow. После создания объекта данного класса, используется метод show библиотеки PyQt5, после чего появляется окно интерфейса.

```
window = ProgramWindow()

window.resize(1340, 615)    # Window size
window.show()
```

В классе ProgramWindow переопределен метод eventFilter, который обрабатывает момент начала распознавания голосовой команды и возвращает стартовую страницу, если предыдущая команда использовала браузер.

```
# Handling a click on the image
def eventFilter(self, obj, event):
    """
    Function that handles clicks on an image

    :param obj: the object on which the action is performed
    :param event: type of event
    :return: changed object
    """
    if event.type() == 2:
        mouse_button = event.button()
        if mouse_button == 1:
            listen_command()

        elif mouse_button == 2:
            self.label.setText("<center><img
src='file:///"+os.getcwd()+"/img/img_greetings.jpg'></center>")
            return_menu_html = open('feature_list.html', 'r',
encoding='UTF-8')

            returned_feature_list_html = return_menu_html.read()
            self.browser2.setHtml(returned_feature_list_html,
QtCore.QUrl("feature_list"))
            return_menu_html.close()

    return super(QMainWindow, self).eventFilter(obj, event)
```

После приема голосовой команды, фраза попадает в функцию-селектор, которая, по ключевым словам, определяет, какая функция, в каком модуле будет выполняться.

Если функция не связана с базой данных, модуль Assistant_functions.py выполнит нужную функцию и вернет результат ее выполнения в модуль Assistant_main_file.py.

Если функция связана с базой данных, запрос пользователя перенаправится в функцию выбора операции selecting_database_function класса DatabaseFunctionSelector в модуле Assistant_database.py.

```
class DatabaseFunctionSelector:
    def __init__(self):
        self.error_answer = 'Ошибка работы с базой данных'

    def selecting_database_function(self, phrase):
        """
        Selecting the function of interaction with the database

        :param phrase: User command
        :return: Function Success Phrase
        """
        answer = self.error_answer
        phrase = Assistant_functions.clean_phrase(phrase,
                                                    ['база', 'баз',
                                                     'данных'])
        if not os.path.exists(f'{DATABASE_NAME}'):
            WorkingWithDatabaseUsingSQL().create_database()

            if (phrase.find("базу") != -1) and ((phrase.find("очисти") != -
1) or (phrase.find("очистить") != -1) or (phrase.find("удали") != -1)
or (phrase.find("удалить") != -1)):
                path =
os.path.join(os.path.abspath(os.path.dirname(__file__)),
'login_details.db')
                os.remove(path)

                WorkingWithDatabaseUsingSQL().create_database()
                answer = 'База данных очищена'

            elif ((phrase.find("добавь") != -1) or
(phrase.find("добавить") != -1) or (phrase.find("записать") != -1) or
(phrase.find("записать") != -1)) and ((phrase.find("логин") != -1) or
(phrase.find("пароль") != -1) or (phrase.find("сайт") != -1) or
(phrase.find("данные") != -1)):
                DatabaseUserInteraction().input_authentication_data()
                answer = ""Данные успешно добавлены в базу,\nвы можете
просмотреть их с помощью голосовой команды.""

            elif ((phrase.find("удали") != -1) or (phrase.find("удалить")
!= -1)) and ((phrase.find("данные") != -1) or (phrase.find("запись")
!= -1) or (phrase.find("сайт") != -1)):
                DatabaseUserInteraction().delete_authentication_data()
                answer = 'Готово! Помните о безопасности ваших
персональных данных!!!'

            elif ((phrase.find("напомни") != -1) or (phrase.find("какой")
!= -1)) and ((phrase.find("пароль") != -1) or (phrase.find("логин") !=
-1)):
                DatabaseUserInteraction().output_authentication_data()
                answer = 'Готово! Помните о безопасности ваших
```

персональных данных!!!'

return answer

Далее срабатывает одна из высокоуровневых функций взаимодействия с базой данных из класса DatabaseUserInteraction. Возможные функции: вывод, добавление или удаления информации. Эти функции работают при помощи низкоуровневых методов класса WorkingWithDatabaseUsingSQL, которые передают SQL-запросы в базу данных.

```
class WorkingWithDatabaseUsingSQL:
    def __init__(self):
        self.database_name = DATABASE_NAME
        self.db_table_name = TABLE_NAME

    def create_database(self):
        """
        Creating a database to store user authentication data

        :return: Nothing
        """
        conn = sqlite3.connect(f'{self.database_name}')
        curs = conn.cursor()

        curs.execute('''CREATE TABLE login_details
        (website VARCHAR(30) PRIMARY KEY,
        login VARCHAR(30),
        password VARCHAR(30))''')

        curs.close()
        conn.close()

    def insert_in_database(self, inserted_data):
        """
        Adds user data to the database

        :param inserted_data: User data
        :return: nothing
        """
        conn = sqlite3.connect(f'{self.database_name}')
        curs = conn.cursor()
        insert_template = f'''INSERT INTO {self.db_table_name}
        (website, login, password) VALUES(?, ?, ?)'''
        curs.execute(insert_template,
                      (f'{inserted_data[0]}', f'{inserted_data[1]}',
                      f'{inserted_data[2]}'))
        conn.commit()
        curs.close()
        conn.close()

    def delete_from_database(self, deleted_data):
        """
        Delete user data to the database

        :param deleted_data: User data
```

```

        :return: nothing
        """
        print(deleted_data)
        conn = sqlite3.connect(f'{self.database_name}')
        curs = conn.cursor()
        curs.execute(f'''DELETE FROM {self.db_table_name}
                        WHERE website = "{deleted_data}"''')
        conn.commit()
        curs.close()
        conn.close()

    def get_from_database(self, requested_website):
        """
        Searches the required user data in the database

        :param requested_website: The site for which the data is
needed
        :return: User authentication data
        """
        conn = sqlite3.connect(f'{self.database_name}')
        curs = conn.cursor()
        curs.execute(f'''SELECT * FROM {self.db_table_name}
                        WHERE website = "{requested_website}"''')
        authentication_data = curs.fetchall()
        curs.close()
        conn.close()
        if not authentication_data:
            return 0
        return authentication_data[0]

```

После отработки функций, модуль Assistant_database.py возвращает статус выполнения пользовательской команды в главный модуль и далее выводится в интерфейс.

Таким образом управление программным средством осуществляется через главный модуль, а вспомогательные модули выполняют функции, соответствующие запросу пользователя.

2.3 Проектирование и разработка графического интерфейса

Графический интерфейс пользователя является частью пользовательского интерфейса и определяет взаимодействие с пользователем на уровне визуализированной информации.

Для проектирования графического интерфейса использовались стандартный графический пакета Python – Tkinter и его расширение – PyQt, версия 5. В PyQt5 визуальные контроллы называются виджетами – стандартизированные компоненты графического интерфейса, с которыми взаимодействует пользователь (окна веб-интерфейса (QWebEngineView), вывод графической информации пользователю (QLabel) и т.д.).

PyQt5 поддерживает язык CSS и более 600 виджетов (включая пользовательские), что позволяет создавать различные вариации пользовательских интерфейсов. Такое разнообразие позволило создать главное окно интерфейса, разделенное на логические элементы: окно

выполнения команд, окно отображающее статус работы голосового ассистента и чат с пользователем.

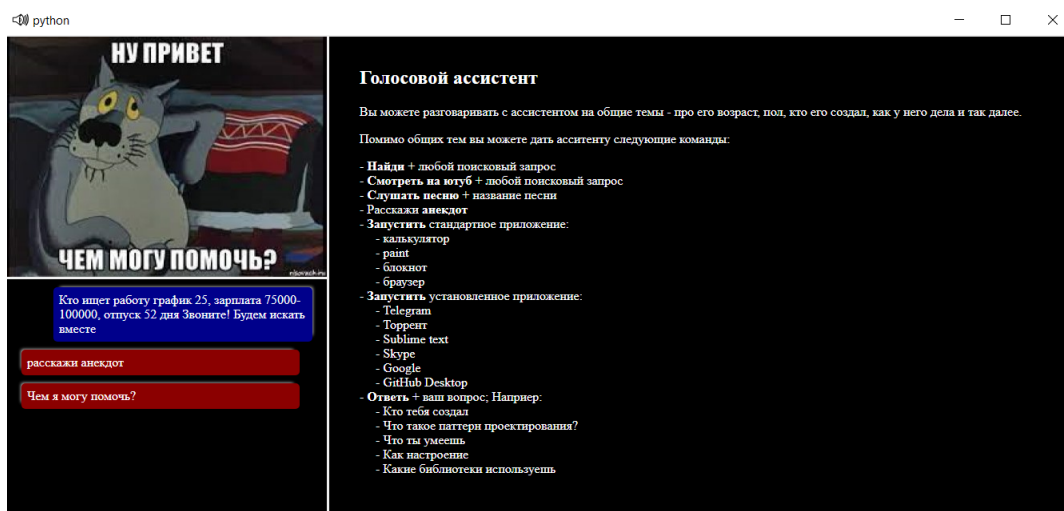


Рис. 2.2 – Главное окно пользовательского интерфейса

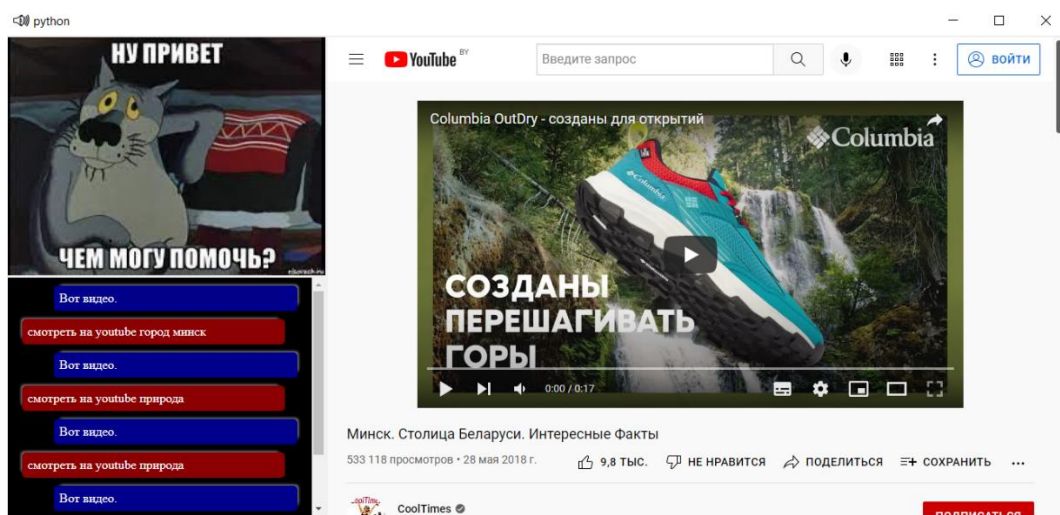


Рис. 2.2 – Главное окно пользовательского интерфейса, где правая часть используется в качестве браузера

TKinter использовался для создания диалоговых окон при работе с базой данных, где интерфейс должен быть простым и удобным. Для создания интерфейса в TKinter использовались следующие виджеты: Label – область где размещаются остальные виджеты, Entry – поле для ввода данных, PhotoImage – для вывода на экран картинки, Button – кнопки для взаимодействия с интерфейсом.

Будьте осторожны со своей персонально...

Введите данные


Название ресурса:

Логин:

Пароль:

Сохранить

Будьте осторожны с персональными данными



Готово


Рис. 2.3 – Окно добавления аутентификационных данных

Будьте осторожны со своей персонально...

Введите нужный ресурс

Удалить

Будьте осторожны с персональными данными



Готово

Рис. 2.4 – Окно удаления аутентификационных данных

Интерфейс функций работы с базой данных выполнен в едином стиле, чтобы пользователю было легче ориентироваться в программе.



Рис. 2.5 – Окно поиска аутентификационных данных

В интерфейсе предусмотрено предупреждение для пользователей о безопасности, так как в общественных местах посторонние люди могут, посмотрев в экран ноутбука увидеть логин и пароль.

2.4 Описание и реализация используемых в программном средстве алгоритмов

В разработанном программном средстве реализованы алгоритмы проверки вводимых данных. Основой для алгоритмов является метод `get_from_database`, которая проводит поиск данных в БД, согласно запросу пользователя.

```
def get_from_database(self, requested_website):
    """
    Searches the required user data in the database

    :param requested_website: The site for which the data is needed
    :return: User authentication data
    """
    conn = sqlite3.connect(f'{self.database_name}')
    curs = conn.cursor()
    curs.execute(f'''SELECT * FROM {self.db_table_name}
    WHERE website = "{requested_website}"''')
    authentication_data = curs.fetchall()
    curs.close()
    conn.close()
    if not authentication_data:
        return 0
    return authentication_data[0]
```

После получения ответа от базы данных при помощи функции fetchall, библиотеки sqlite3 и записи его в переменную authentication_data, существует два варианта:

1. В базе данных были искомые данные и переменная authentication_data содержит список со списком полученных переменных (название сайта, логин и пароль).

2. В базе данных не было искомым данных и переменная authentication_data содержит пустой список.

Если список пуст, возвращает значение 0, если нет, возвращает данные для аутентификации.

На основе этого происходит проверка на уникальность данных: если пользователь вводит название ресурса, который есть в БД, ему выводится оповещение об ошибке.

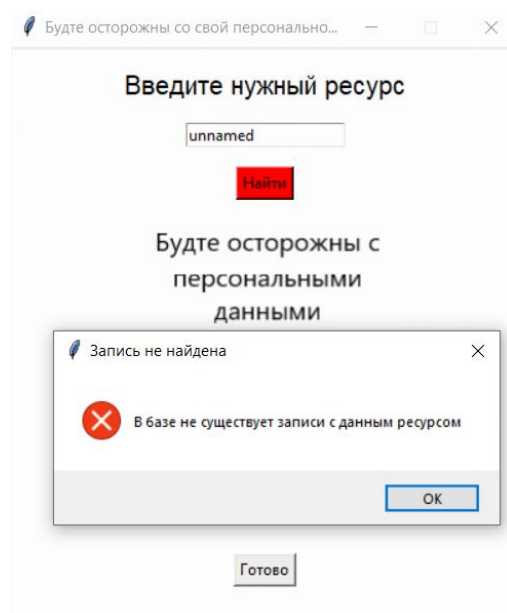


Рис. 2.6 – Результат не пройденной проверки на существование данных

Код функции проверяющей уникальность записи:

```
authentication_data = WorkingWithDatabaseUsingSQL().  
.get_from_database(site)  
  
if authentication_data == 0:  
    messagebox.showerror(title='Запись не найдена',  
                          message=f'''В базе не существует записи с  
данным ресурсом''')  
  
else:  
    login = str(authentication_data[1])  
    password = str(authentication_data[2])  
    messagebox.showwarning(title='Данные',  
                           message=f'''Логин: {login}\nПароль: {password}''')
```

Аналогично происходит проверка на существование данных: если пользователь пробует добавить ресурс, который уже есть в базе, ему

выводится оповещение о невозможности данного действия и его данные к этому ресурсу.

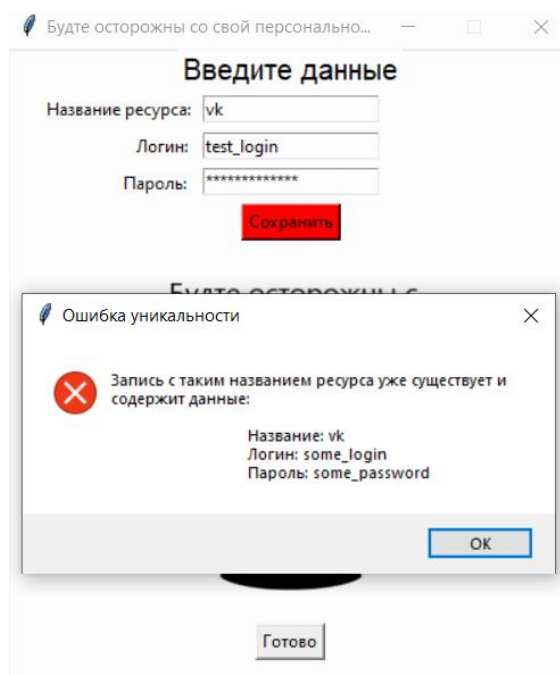


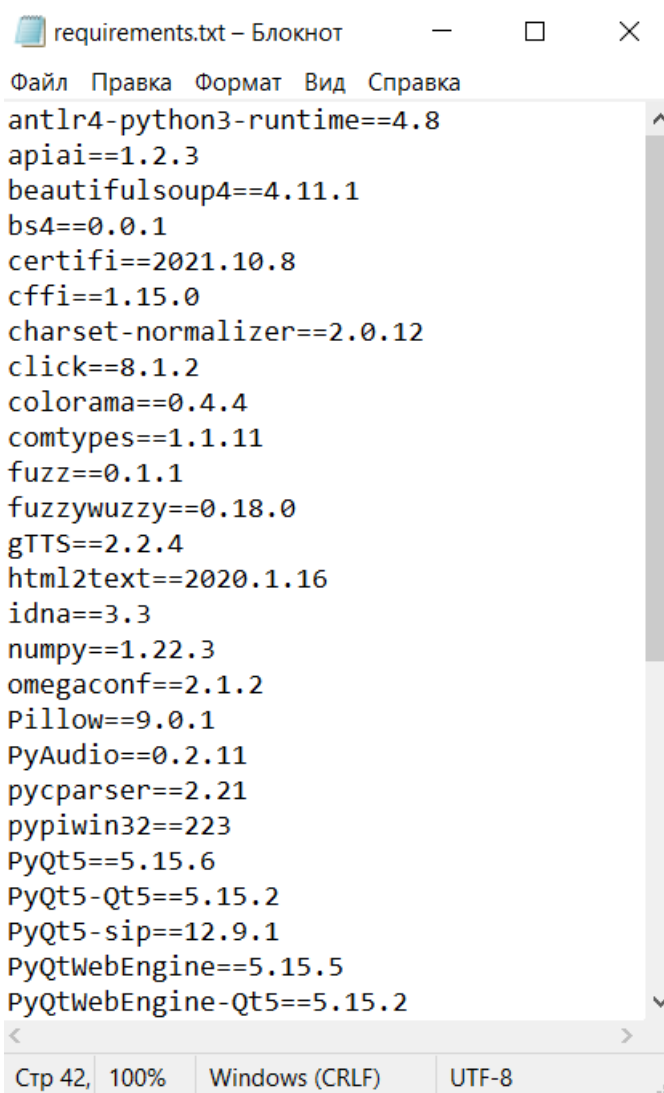
Рис. 2.7 – Результат не пройденной проверки на уникальность данных

Также предусмотрена ошибка удаления ресурса, записи о котором нет в БД.

3 ЭКСПЛУАТАЦИЯ ПРОГРАММНОГО СРЕДСТВА

3.1 Ввод в эксплуатацию и обоснование минимальных технических требований

Для ввода в эксплуатацию программного средства и его корректной работы на компьютере пользователя должны быть только: ОС Windows 10, Интернет-соединение и Python версии 3.8.13 и выше. Интернет подключение нужно для работы библиотеки распознавания голоса, так как она отправляет полученный результат на внешний сервер для обработки. Пользователю необходимо, только использовать в консоли команду `python -m pip install -r requirements.txt` [5] для установки необходимых библиотек. В файл `requirements.txt` записаны названия этих библиотек.



```
requirements.txt - Блокнот
Файл  Правка  Формат  Вид  Справка
antlr4-python3-runtime==4.8
apiai==1.2.3
beautifulsoup4==4.11.1
bs4==0.0.1
certifi==2021.10.8
cffi==1.15.0
charset-normalizer==2.0.12
click==8.1.2
colorama==0.4.4
comtypes==1.1.11
fuzz==0.1.1
fuzzywuzzy==0.18.0
gTTS==2.2.4
html2text==2020.1.16
idna==3.3
numpy==1.22.3
omegaconf==2.1.2
Pillow==9.0.1
PyAudio==0.2.11
pysparser==2.2.1
pywin32==223
PyQt5==5.15.6
PyQt5-Qt5==5.15.2
PyQt5-sip==12.9.1
PyQtWebEngine==5.15.5
PyQtWebEngine-Qt5==5.15.2
Стр 42, 100% Windows (CRLF) UTF-8
```

Рис. 3.1 – Содержание файла `requirements.txt` к программному средству

Данный способ установки используется при распространении граграмм через сервис `github.com`. Чтобы загрузить свою программу на GitHub необходимо [6]:

1. Открыть консоль.
2. Перейти в директорию проекта.
3. Определить текущую директорию, как директорию репозитория.

4. Добавить файлы в новый локальный репозиторий, чтобы добавить файлы в первый коммит.
5. Сделать коммит файлов, который вы добавили в коммит в ваш локальный репозиторий.
6. Скопировать ссылку на репозиторий.
7. В командной строке добавить ссылку на удаленный репозиторий.
8. Отправить изменения локального репозитория в Git.

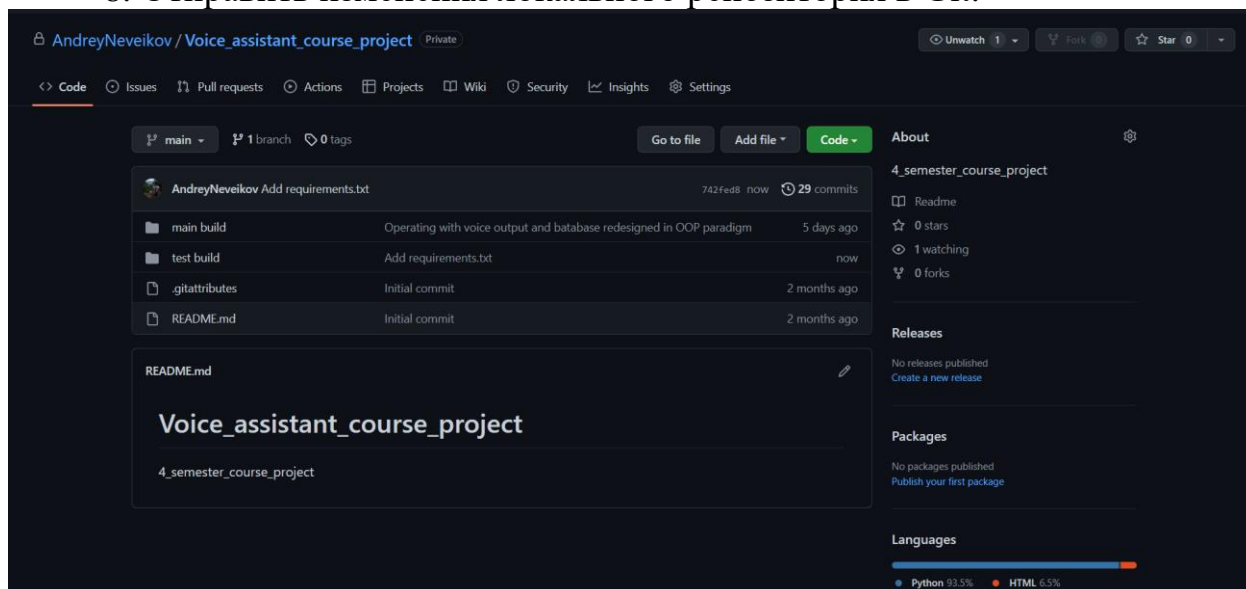


Рис. 3.2 – Программное средство размещенное на github.com

Также распространять программу можно с помощью Docker [7]. Docker создаёт образ виртуальной машины с установленными в ней приложениями. Дальше этот образ разворачивается как абсолютно автономная виртуальная машина. Запущенная копия образа называется «контейнер». Пользователь может запустить на сервере любое количество образов, и каждый из них будет отдельной виртуальной машиной со своим окружением.

Виртуальная машина – это инкапсулированное место на сервере с ОС, в которой установлены приложения. В любой ОС обычно установлено большое количество приложений, в данной же находится одно.

Особенности использования Docker: единственная программа, необходимая для деплоя копии приложения на любом устройстве — это Docker. Если разработчик запустил своё приложение в докере у себя на ЭВМ, оно гарантированно с тем же успехом запустится в любом другом докере. И ничего, кроме докера, устанавливать не нужно.

Таким образом, при использовании Docker, минимальным системным требование является наличие Docker.

3.2 Руководство по эксплуатации программного средства

При запуске программного средства на экране у пользователя появится начальная страница, разделенная на три зоны. В левом верхнем углу находится кнопка приема команды (активируется одиночным нажатием ЛКМ). В левом нижнем углу расположен чат, в него записывается текст

голосовой команды, а затем ответ ассистента, слева расположена рабочая область, изначально в ней выводится список функций. И руководство по их вызову.

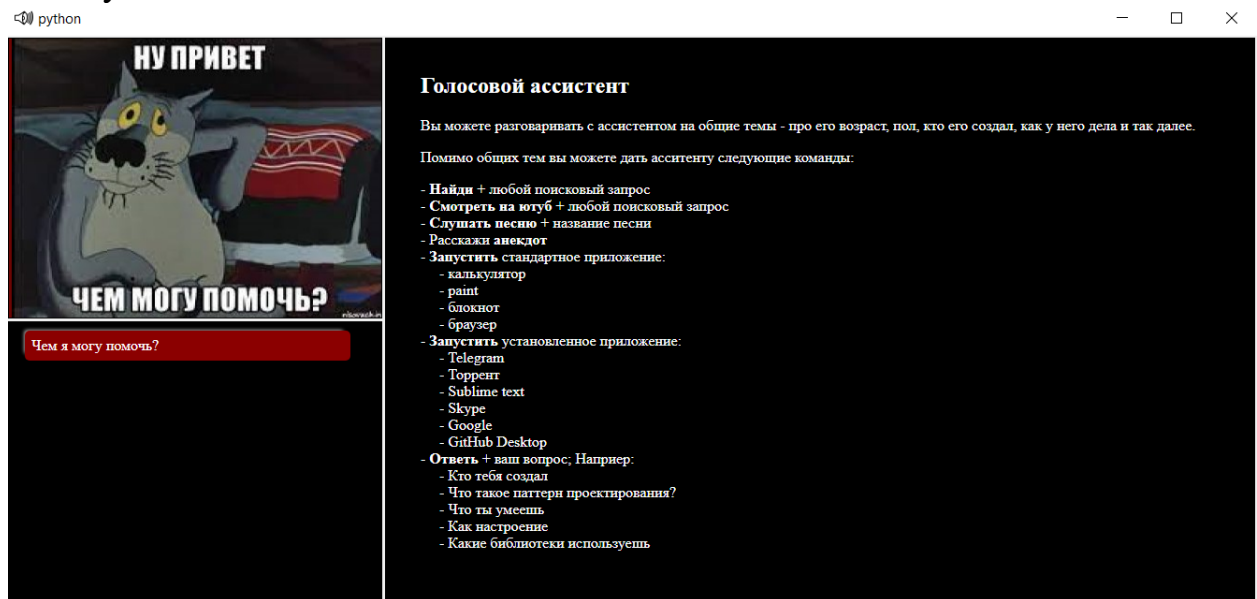


Рис. 3.3 – Начальная страница

При нажатии на левую верхнюю область, картинка меняется, а ассистент начинает приём голосовой команды.

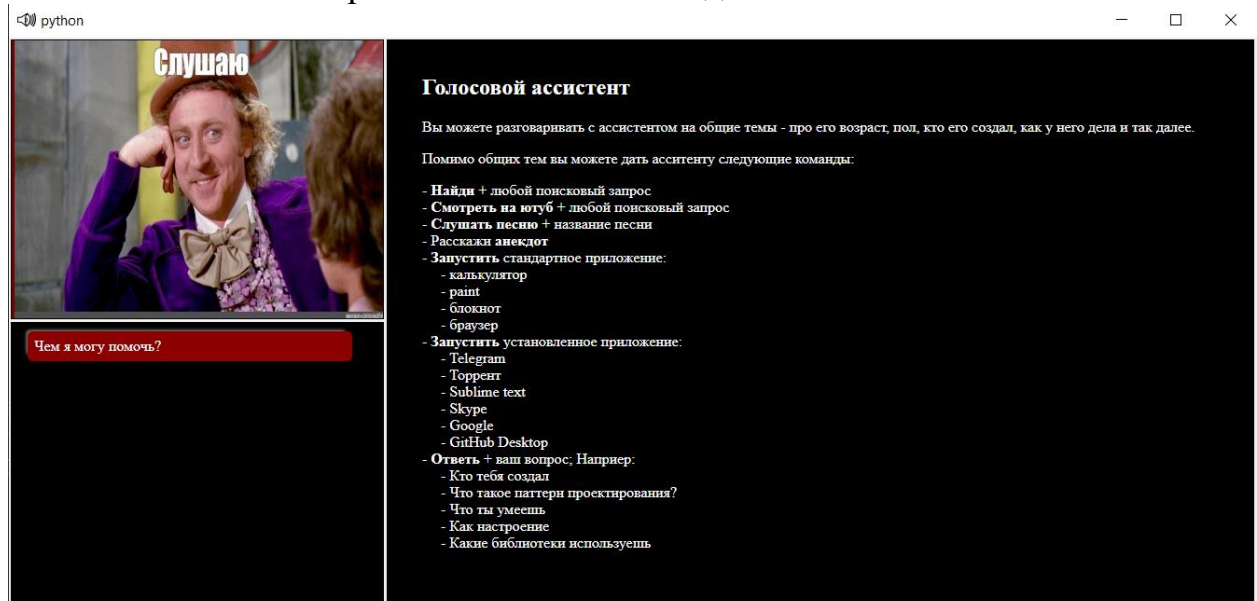


Рис. 3.4 – Голосовой ассистент в режиме приема команды

Если ассистент не смог распознать слов во фразе, он сообщит об этом изменяя картинку в левом верхнем углу. Данная ситуация происходит крайне редко, так как в разработанном программном средстве используются алгоритмы распознавания речи от Google, способные распознавать русскую и английскую речь. Если ассистент обнаружил слова на записи голоса, но эти слова не являются запрограммированной командой, ассистент ответит: «Я не смогу выполнить запрос».

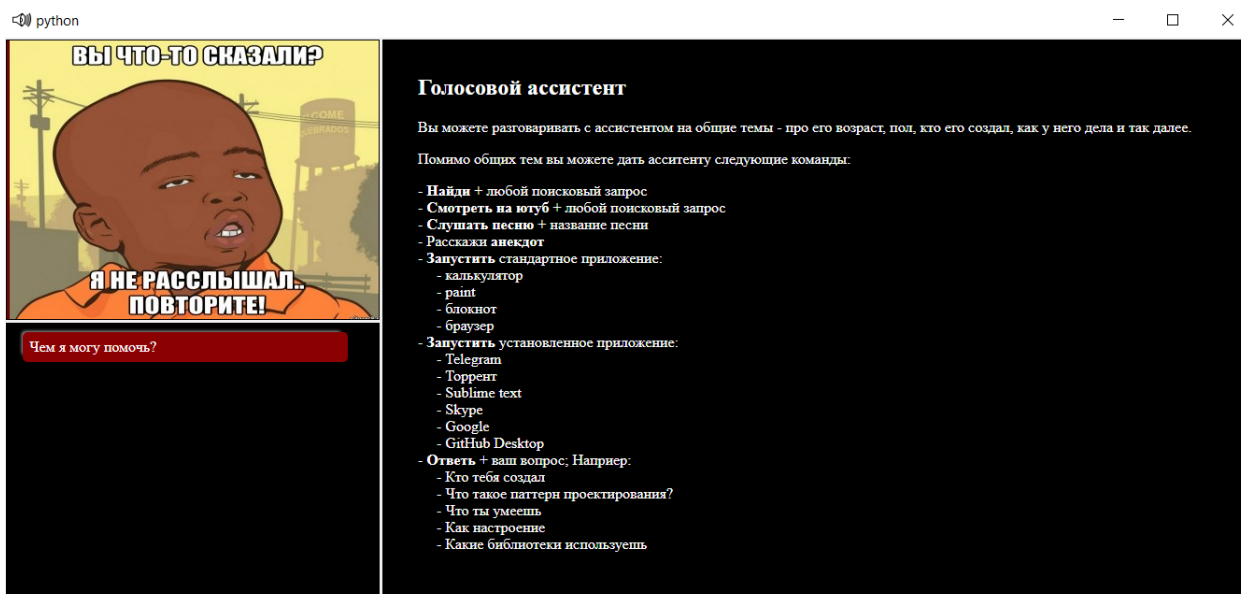


Рис. 3.5 – Голосовой ассистент, когда не обнаружил слов на аудиозаписи

Если программная реализация не предусматривает распознанную команду, ассистент кроме голосового ответа, продублирует «Я не смогу выполнить запрос» в чат. При использовании программы полезно смотреть в чат, чтобы видеть как вас услышал ассистент и понимать, какие слова стоит произносить внятнее.

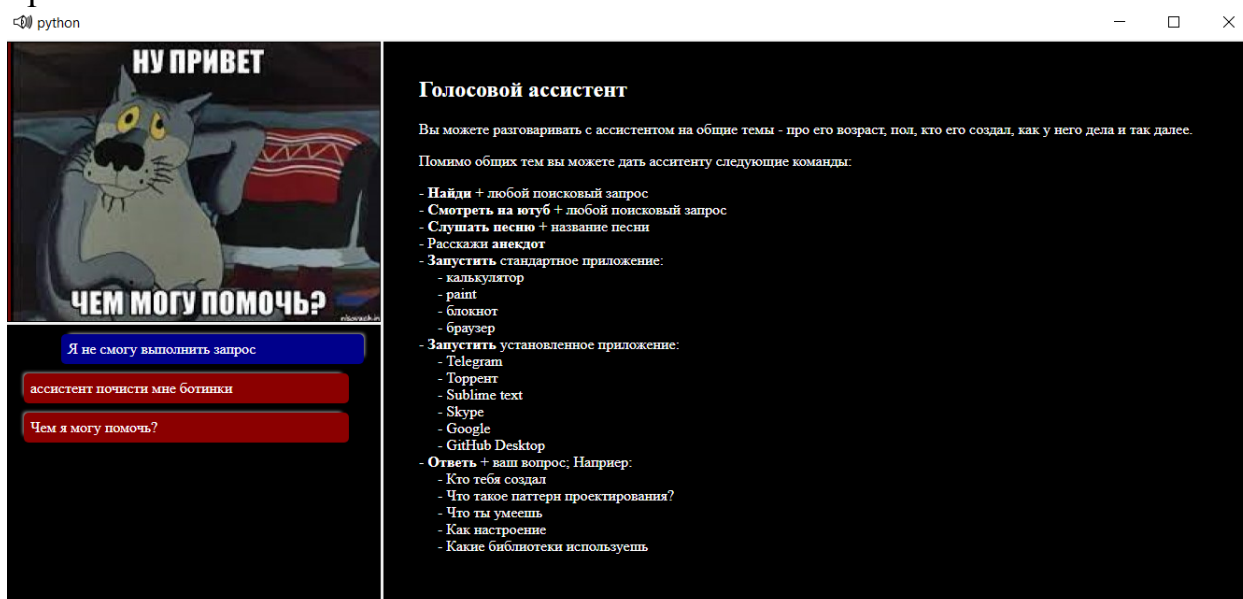


Рис. 3.6 – Голосовой ассистент, когда не может выполнить команду

Если голосовая команда задействует браузер, то список команд справа, заменяется на интерактивную веб-страницу. На данной странице доступен весь функционал, кроме полноэкранного режима и использования плагинов, таких как AdBlock, Volume Master, Grammarly и других.

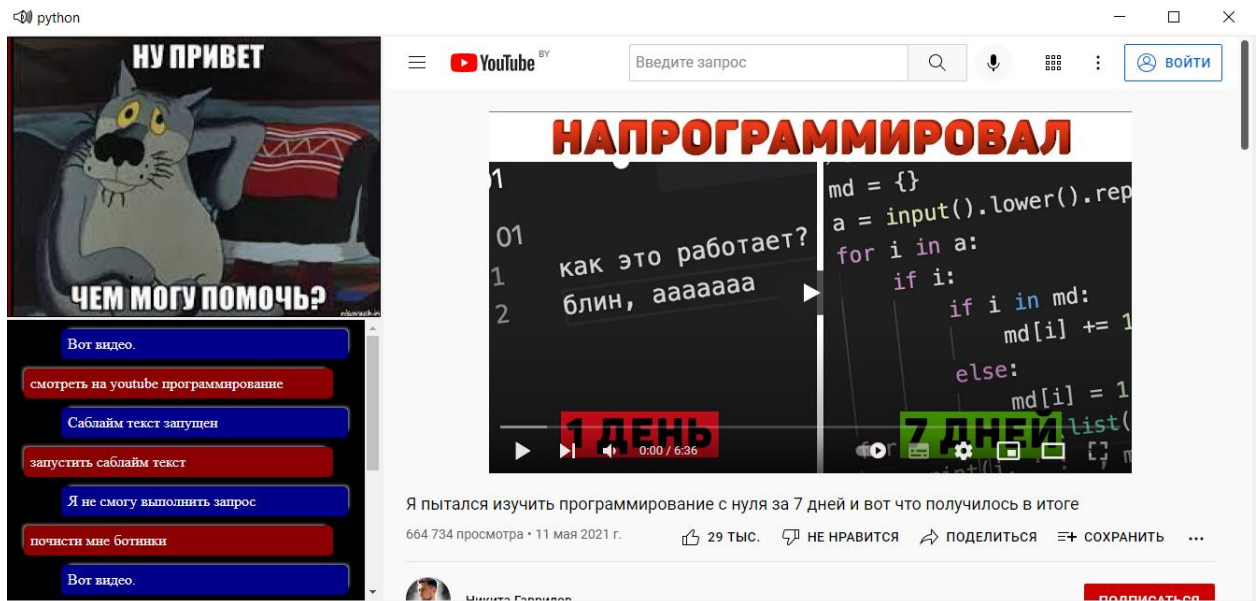


Рис. 3.7 – Выполнение команды команду связанной с браузером

Если голосовая команда задействует установленное приложение, то оно запускается в отдельном окне.

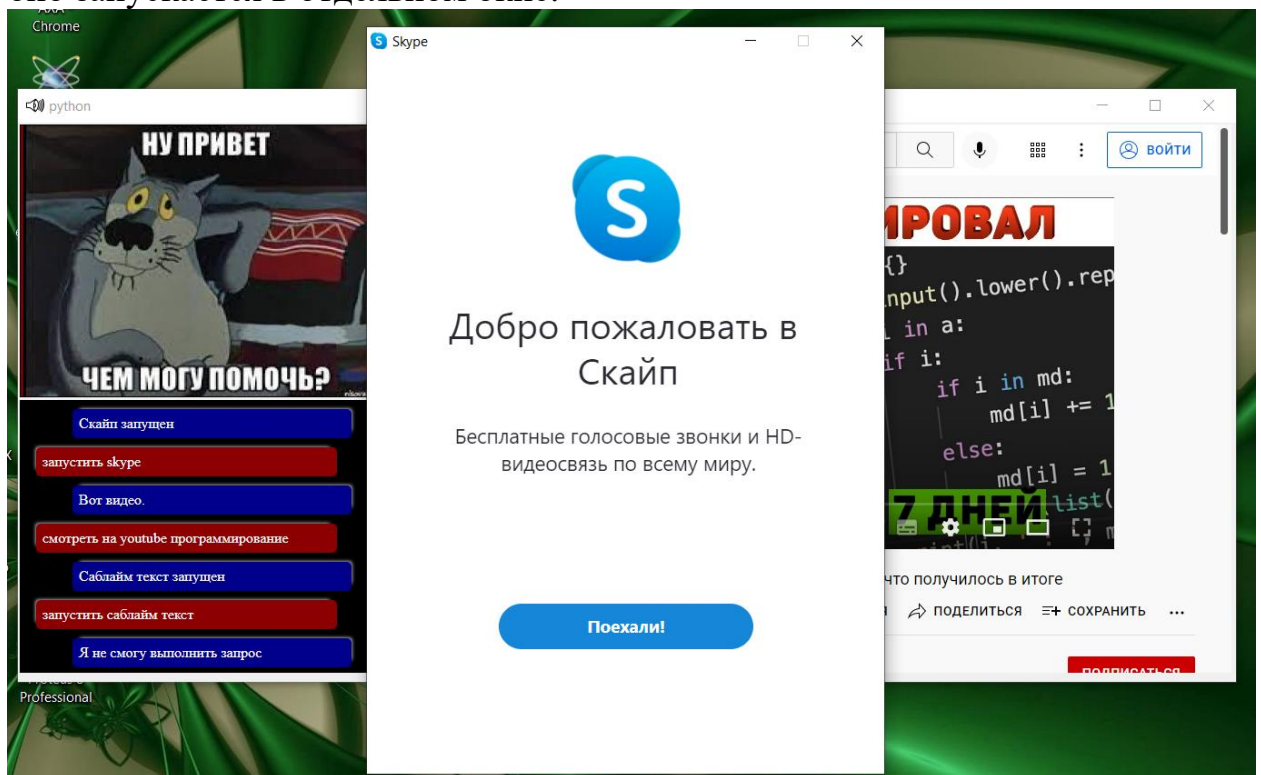


Рис. 3.8 – Выполнение команды запуска Skype.

Если пользователю нужно посмотреть на список доступных команд, а правая часть интерфейса уже используется, пользователь может нажать на левый верхний блок интерфейса при помощи ПКМ. Справа появится список команд, история чата сохранится.

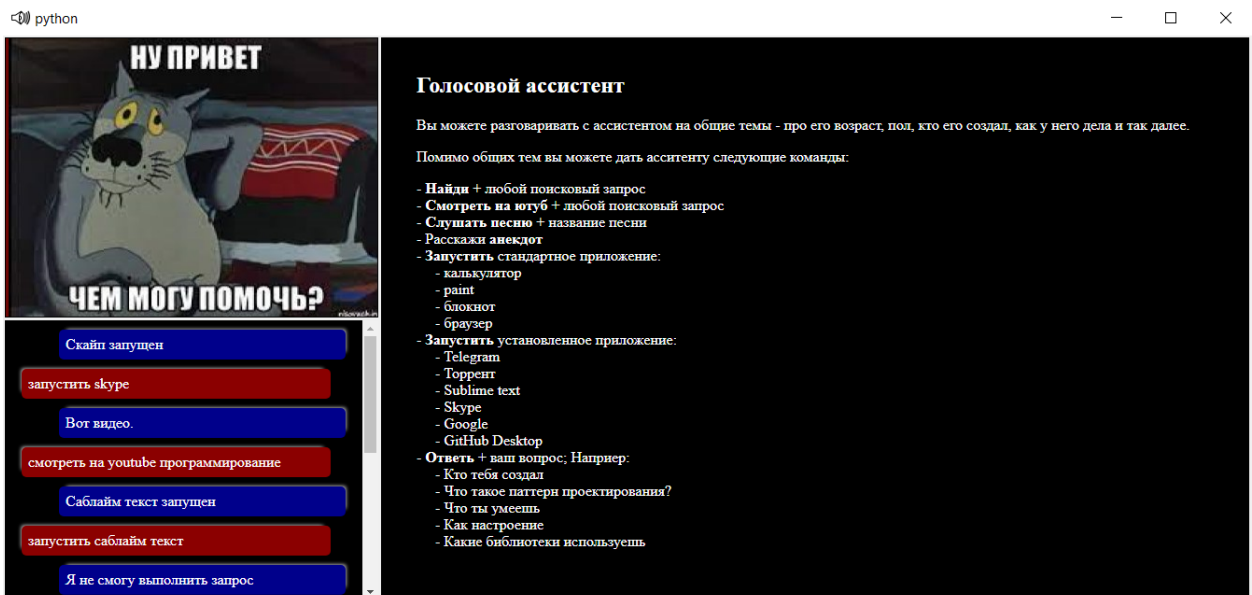


Рис. 3.9 – Заккрытие браузера

При работе с базой данных предусмотрены отдельные окна. Чтобы обратиться к функциям, использующих базу данных, необходимо в начале запроса сказать «База данных».

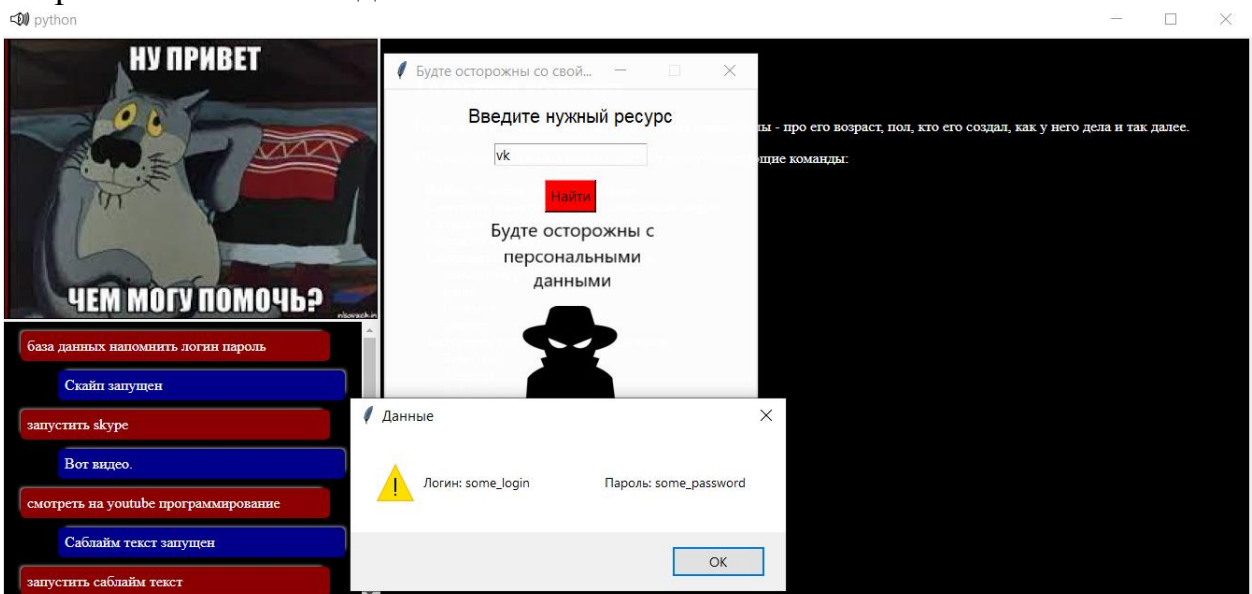


Рис. 3.9 – Вывод логина и пароля из базы данных

Таким образом эксплуатация программного средства простая и понятная: пользователю надо просто произносить команды. Практически всё остальное программа выполнит сама.

ЗАКЛЮЧЕНИЕ

В процессе работы над курсовым проектом получены знания об способах обработки голосовых команд. Изучены и применены на практике возможности библиотек: `speech_recognition`, `threading`, `sounddevice`, `torch`, `urllib`, `PyQt5`, `sqlite3`, `sys`, `os`, `re`, `bs4`. В процессе создания программы удалось выполнить основные принципы хорошей архитектуры: эффективность, расширяемость, масштабируемость, удобство тестирования, понятность кода[8].

В данной работе использовался язык Python, как один из лучших многофункциональных, скриптовых языков, поддерживающих реализацию ООП.

Было найдено несколько областей для практического применения созданного программного средства, на основе анализа его работы.

Одну из главных ролей в программе играет пользовательский интерфейс, реализованный не только при помощи визуальной составляющей, но и аудио взаимодействия. При помощи искусственного интеллекта был воссоздан голос, похожий на человеческий для комфорта пользователя. Интерфейс реализован в темных цветах, чтобы можно было использовать ночью.


Программа содержит полный функционал для выполнения поставленных задач, а также имеет интерфейс для удобного управления программой и отображения результатов выполнения функций.

Таким образом поставленные в курсовом проекте задачи были выполнены.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- [1] Gartner [Электронный ресурс]. – Режим доступа <https://www.gartner.com/smarterwithgartner/gartner-predicts-a-virtual-world-of-exponential-change>
- [2] Лайкни [Электронный ресурс]. – Режим доступа: <https://www.likeni.ru/glossary/golosovoy-pomoshchnik/>
- [3] Data Science [Электронный ресурс]. – Режим доступа: <https://data.science.eu/ru/компьютерное-зрение/архитектура-программного-обеспечения/>
- [4] Habr.com [Электронный ресурс]. – Режим доступа: <https://habr.com/ru/post/276593/>
- [5] devman [Электронный ресурс]. – 2020. – Режим доступа: https://dvmn.org/encyclopedia/pip/pip_requirements_txt/
- [6] *OiPlug* [Электронный ресурс]. – Режим доступа: <https://oiplug.com/blog/git/5049/>
- [7] Habr [Электронный ресурс]. – Режим доступа: <https://habr.com/ru/post/448094/>
- [8] PEP 8 - руководство по написанию кода на Python [Электронный ресурс]. – Режим доступа: <https://pythonworld.ru/osnovy/pep-8-rukovodstvo-po-napisaniyu-koda-na-python.html>

ПРИЛОЖЕНИЕ А
(обязательное)
Результат проверки на «Антиплагиат»

<input type="checkbox"/> Название ↕	Дата загрузки ↕	Оригинальность	
<input type="checkbox"/> PDF Пояснительная записка	 30 Мая 2022 18:57	92,9%	ПОСМОТРЕТЬ РЕЗУЛЬТАТЫ

ПРИЛОЖЕНИЕ Б

(обязательное)

Листинг программного кода

Assistant_main_file.py

```
# Interface
from PyQt5.QtWidgets import QLabel, QApplication, QMainWindow
from PyQt5.QtWebEngineWidgets import *
from PyQt5 import QtCore, QtGui
from PyQt5.QtGui import QIcon

# Functional
import speech_recognition as sr
import threading
import signal
import sys
import os

# Files
import Assistant_voice_output_settings
import Assistant_functions
import Assistant_database

# Get the html page for messages in the chat window
html_code = '<div class="robot">Чем я могу помочь?</div>'
file = open('chat.html', 'r', encoding='UTF-8')
html_chat = file.read()
file.close()

# Get the html page feature list
file = open('feature_list.html', 'r', encoding='UTF-8')
feature_list_html = file.read()
file.close()

r = sr.Recognizer()      # Variable for speech recognition from Google

def thread(my_func):
    """
    Function that creates a separate thread
    (used as decorator)

    :param my_func: a function to run on a new thread
    :return: wrapper
    """
    def wrapper(*args, **kwargs):
        my_thread = threading.Thread(target=my_func, args=args,
        kwargs=kwargs)
        my_thread.start()
        return wrapper

global interrupted_thread
```

```

def signal_handler(thread_signal, frame):
    """
    Function for signals between threads

    :param thread_signal: thread signal
    :param frame: signal handler
    :return: nothing, just changes the state of the thread
    """
    global interrupted_thread
    interrupted_thread = True

def interrupt_callback():
    """
    A function that accesses an interrupted thread

    :return: a thread interrupted by another thread
    """
    global interrupted_thread
    return interrupted_thread

@thread
def listen_command():
    """
    Activates Speech Recognition to recognize commands

    :return: recognized phrase or handled error
    """
    global listen
    global request
    global not_listen

    listen.emit([1])    # Monitoring the state of the assistant
    (listens or speaks)
    with sr.Microphone() as source:    # Listen to the microphone
        audio = r.listen(source)
    try:
        # Send the record to Google, get the recognized phrase
        voice_record = r.recognize_google(audio, language="ru-
RU").lower()
        listen.emit([2])    # Change the assistant's state from
        listening to answering
        # Send the recognized phrase for processing to the
        response_to_user_request function
        request.emit([voice_record])
        # In case of an error, change the state of the assistant to
        "didn't hear"
    except sr.UnknownValueError:
        print("Ассистент не расслышал фразу")
        not_listen.emit(['00'])
    except sr.RequestError as error:
        print("Ошибка сервиса; {0}".format(error))

```



```

signal.signal(signal.SIGINT, signal_handler)    # Thread signal
processing

global p_urls
global p_cmd

# Blanks for the state of threads
answer = ''
listen = ''
request = ''
not_listen = ''
speaking = ''

class ProgramWindow(QMainWindow):
    """
    Create a PyQt interface
    """
    # Declare signals that come from asynchronous functions
    thread_signal = QtCore.pyqtSignal(list, name='thread_signal')
    assistant_listen = QtCore.pyqtSignal(list,
name='assistant_listen')
    user_request = QtCore.pyqtSignal(list, name='user_request')
    unrecognized_speech = QtCore.pyqtSignal(list,
name='unrecognized_speech')

    def __init__(self, *args):
        super().__init__()
        self.setWindowIcon(QIcon("img\\app_icon.png"))
        self.setAnimated(False)
        self.flag = True
        self.centralwidget = QMainWindow()
        self.centralwidget.setObjectName("centralwidget")
        self.setCentralWidget(self.centralwidget)
        # Label in which we will load pictures
        self.label = QLabel(self.centralwidget)
        # Attach a click handling function to the Label
        self.label.installEventFilter(self)
        # Customize the appearance of the cursor on the picture

self.label.setCursor(QtGui.QCursor(QtCore.Qt.PointingHandCursor))
        # Position the Label inside the window
        self.label.setGeometry(QtCore.QRect(2, 2, 400, 300))
        self.label.setStyleSheet("QLabel { \n"
                                "color: white;\n"
                                "background-color: #6c0503;\n"
                                "border: 1px solid #000000;\n"
                                "border-radius: 0;\n"
                                "}\n"
                                "\n")

        # Declare the QWebEngineView element to display the html page
with the chat
        self.browser = QWebEngineView(self.centralwidget)
        # Declaring the QWebEngineView element to display YouTube
videos, texts and web pages
        self.browser2 = QWebEngineView(self.centralwidget)

```

```

# Position the QWebEngineView inside the window
self.browser.setGeometry(QtCore.QRect(2, 305, 400, 300))
self.browser2.setGeometry(QtCore.QRect(405, 2, 930, 603))

# Load the html page with the chat into QWebEngineView
global html_chat
global html_code
global feature_list_html

html_result = html_chat.replace('%code%', html_code)
self.browser.setHtml(html_result, QtCore.QUrl("file:///"))
self.browser.show()
self.browser2.setHtml(feature_list_html,
QtCore.QUrl("file:///"))
self.browser2.show()
self.label.setText("<center><img src='file:///"+os.getcwd() +
"/img/img_greetings.jpg"></center>")

# Connect signals and class functions
global answer
answer = self.thread_signal
global listen
listen = self.assistant_listen
global not_listen
not_listen = self.unrecognized_speech
global request
request = self.user_request
self.assistant_listen.connect(self.picture_change,
QtCore.Qt.QueuedConnection)
self.user_request.connect(self.response_to_user_request,
QtCore.Qt.QueuedConnection)

self.unrecognized_speech.connect(self.response_to_unrecognized_speech,
QtCore.Qt.QueuedConnection)

# Handling a click on the image
def eventFilter(self, obj, event):
    """
    Function that handles clicks on an image

    :param obj: the object on which the action is performed
    :param event: type of event
    :return: changed object
    """
    if event.type() == 2:
        mouse_button = event.button()
        if mouse_button == 1:
            listen_command()

        elif mouse_button == 2:
            self.label.setText("<center><img
src='file:///"+os.getcwd()+"/img/img_greetings.jpg"></center>")
            return_menu_html = open('feature_list.html', 'r',
encoding='UTF-8')
            returned_feature_list_html = return_menu_html.read()
            self.browser2.setHtml(returned_feature_list_html,
QtCore.QUrl("feature_list"))

```

```

        return_menu_html.close()

    return super(QMainWindow, self).eventFilter(obj, event)

def picture_change(self, data):
    """
    Function of changing the picture depending on
    whether the assistant is listening or talking

    :param data: assistant mode
    :return: nothing, just change an image of the assistant's mode
    """
    if data[0] == 1:
        # Assistant listens
        self.label.setText("<center><img src='file:///\" +
os.getcwd() +
                                \"/img/img_listen.jpg'></center>")
    if data[0] == 2:
        # Assistant speaks
        self.label.setText("<center><img src='file:///\" +
os.getcwd() +
                                \"/img/img_greetings.jpg'></center>")

def adding_response_to_chat_by_assistant(self, phrase):
    """
    Adding an assistant's phrase to the html chat

    :param phrase: assistant answer
    :return: nothing, writes assistant answer in the html chat
    """
    global html_chat
    global html_code
    html_code = '<div class="robot">' + phrase + '</div>' +
html_code
    html_result = html_chat.replace('%code%', html_code)
    self.browser.setHtml(html_result, QtCore.QUrl("file:///"))
    self.browser.show()

def adding_query_to_chat_by_user(self, phrase):
    """
    Adding a user phrase to the html chat

    :param phrase: assistant request
    :return: nothing, writes user request in the html chat
    """
    global html_chat
    global html_code
    html_code = '<div class="you">' + phrase + '</div>' +
html_code
    html_result = html_chat.replace('%code%', html_code)
    self.browser.setHtml(html_result, QtCore.QUrl("file:///"))
    self.browser.show()

    @staticmethod
    def pronounce_assistant_answer(phrase):
        """
        Redirects a phrase to the voiceover function

```

```

:param phrase: written phrase
:return: nothing
"""
speaker_list = ['aidar', 'baya', 'kseniya', 'xenia', 'random']

Assistant_voice_output_settings.Speaker(speaker_list[3]).pronounce_assistant_answer(phrase)

def response_to_user_request(self, data):
    """
    Answer by selection function

    :param data: list of keywords
    :return: assistant answer
    """
    global p_urls
    global p_cmd

    phrase = data[0].lower() # Get phrase from user
    self.adding_response_to_chat_by_assistant(phrase) # Display
the user's phrases in the chat
    assistant_answer = 'Я не смогу выполнить запрос' # Default
response

    try:
        # Perform an action depending on the presence of keywords
in the phrase
        if 'ответь' in phrase:
            assistant_answer =
Assistant_functions.assistant_answering_dialogue_phrase(phrase)

            elif ((phrase.find("база") != -1) and
(phrase.find("данных") != -1)) \
                or ((phrase.find("пароль") != -1) or
(phrase.find("логин") != -1))
                    or ((phrase.find("добавить") != -1) and
(phrase.find("данные") != -1))
                        or ((phrase.find("записать") != -1) and
(phrase.find("данные") != -1)
                            and (phrase.find("сайта") != -1))):
                assistant_answer =
Assistant_database.DatabaseFunctionSelector().selecting_database_function(phrase)

            elif (phrase.find("запустить") != -1) or
(phrase.find("запусти") != -1):
                assistant_answer =
Assistant_functions.start_application(phrase)

            elif ((phrase.find("youtube") != -1) or
(phrase.find("ютюб") != -1) or
                    (phrase.find("ютуб") != -1) or (phrase.find("you
tube") != -1))\
                and (phrase.find("смотреть") != -1):

```

```

self.browser2.load(QtCore.QUrl(Assistant_functions.find_on_you_tube(ph
rase)))

        assistant_answer = 'Вот видео.'

        elif ((phrase.find("анекдот") != -1) or
(phrase.find("шутка") != -1) or
        (phrase.find("анек") != -1) or
(phrase.find("прикол") != -1))\
        or (phrase.find("смешной") != -1):
            assistant_answer = Assistant_functions.tell_joke()

        elif (phrase.find("слушать") != -1) and
((phrase.find("песн") != -1) or (phrase.find("песню") != -1)):

self.browser2.load(QtCore.QUrl(Assistant_functions.find_on_you_tube(ph
rase)))

        assistant_answer = 'Вот песня.'

        elif ((phrase.find("найти") != -1) or
(phrase.find("найди") != -1)) \
        and not (phrase.find("статью") != -1):
            user_request =
Assistant_functions.clean_phrase(phrase,

['найти', 'найди', 'про', 'про то', 'о том'])
            question =
Assistant_functions.browser_search(user_request)
            self.browser2.load(QtCore.QUrl(question[0]))
            assistant_answer = 'Ответ найден'

        elif phrase == 'пока' or phrase == 'выход' or phrase ==
'выйти' or phrase == 'до свидания':
            assistant_answer = 'Ещё увидимся!'
            self.adding_query_to_chat_by_user(assistant_answer)
            self.pronounce_assistant_answer(assistant_answer)
            sys.exit(app.exec_())

    except():
        assistant_answer = 'Я не поняла запрос'      # Default
response

        self.adding_query_to_chat_by_user(assistant_answer)      # Add
response to the chat
        self.pronounce_assistant_answer(assistant_answer)      # Speak
out the answer

    def response_to_unrecognized_speech(self, *args):
        """
        Function that changes the picture if the assistant did not
hear you

        :param args: QtCore.Qt.QueuedConnection
        :return: nothing, just change the picture
        """
        self.label.setText("<center><img src='file:///"+os.getcwd() +
"/img/img_response_to_unrecognized_speech.jpg"></center>")

```

```
# Run the program
app = QApplication([])
window = ProgramWindow()

window.resize(1340, 615)    # Window size
window.show()
app.exec_()
```

Assistant_voice_output_settings.py

```
# Functional
import sounddevice as sd
import torch
import time

class Speaker:
    """
    Contain the voice acting settings and voice acting method
    """

    def __init__(self, speaker_voice):
        """
        Initializing Voice Assistant Attributes

        :param speaker_voice: selected voice
        """
        self.speaker = speaker_voice
        self.language = 'ru'
        self.model_id = 'ru_v3'
        self.sample_rate = 48000
        self.put_accent = True
        self.put_yo = True
        self.device = torch.device('cpu')

        self.model, _ = torch.hub.load(repo_or_dir='snakers4/silero-
models',
                                     model='silero_tts',
                                     language=self.language,
                                     speaker=self.model_id)

        self.model.to(self.device)

    def pronounce_assistant_answer(self, what: str):
        """
        Say the phrase aloud with speech synthesis

        :param what: written phrase
        :return: nothing
        """
        audio = self.model.apply_tts(text=what+"..",
                                     speaker=self.speaker,
                                     sample_rate=self.sample_rate,
                                     put_accent=self.put_accent,
                                     put_yo=self.put_yo)
```

```

sd.play(audio, self.sample_rate * 1.05)
time.sleep((len(audio) / self.sample_rate) + 0.5)
sd.stop()

```

Assistant_functions.py

```

# Functional
from urllib.parse import quote
from urllib import request
import urllib.request
import subprocess
import webbrowser
import requests
import bs4
import re
import os

def clean_phrase(statement, words_list):
    """
    Cleans keywords in a phrase

    :param statement: phrase
    :param words_list: keywords
    :return: clean phrase
    """
    for word in words_list:
        statement = statement.replace(word, '')
    statement = statement.strip()
    return statement

def tell_joke():
    """
    A function that gives a random joke (anecdote)

    :return: joke from site
    """
    joke = requests.get('http://anekdotme.ru/random')
    joke_parser = bs4.BeautifulSoup(joke.text, "html.parser")
    parsed_joke = joke_parser.select('.anekdot_text')
    joke = (parsed_joke[0].getText().strip())
    reg_ex = re.compile('[^0-9a-zA-Za-яA-я .,!?-]')
    joke = reg_ex.sub('', joke)
    return joke

def assistant_answering_dialogue_phrase(phrase):
    """
    Answers user questions

    :param phrase: user question
    :return: prepared answer
    """
    answer = 'Пожалуйста, повторите фразу!'
    phrase = clean_phrase(phrase,

```

```

        ['ответь', 'скажи'])

    if (phrase.find("кто") != -1) and (phrase.find("тебя") != -1) and
(phrase.find("создал") != -1):
        answer = 'Меня создал Невейков Андрей'

    elif (phrase.find("как") != -1) and (phrase.find("тебя") != -1)\
        and (phrase.find("зовут") != -1):
        answer = 'Можете обращаться просто ассистент.'

    elif ((phrase.find("сколько") != -1) and (phrase.find("тебе") != -
1)
        and (phrase.find("лет") != -1)) or ((phrase.find("твой") !=
-1)
        and
(phrase.find("возраст") != -1)):
        answer = 'Можете обращаться просто ассистент.'

    elif (phrase.find("какие") != -1) and (phrase.find("библиотеки")
!= -1)\
        and (phrase.find("ты") != -1) and
(phrase.find("используешь") != -1):
        answer = '''Извините за акцент: urllib, subprocess,
webbrowser,
requests, bs4, re, os, sqlite3, PyQt5, speech recognition,
threading, pyttsx3, signal, sys'''

    elif (phrase.find("такое") != -1) and ((phrase.find("ооп") != -1)
        or (phrase.find("офп") != -
1)
        or (phrase.find("о о п") !=
-1)
        or (phrase.find("о п") != -
1)):
        answer = '''
        Объектно-ориентированное программирование – методология
программирования,
        основанная на представлении программы в виде совокупности
объектов,
        каждый из которых является экземпляром определённого класса,
а классы образуют иерархию наследования.
        '''

    elif (phrase.find("такое") != -1) and (phrase.find("паттерн") != -
1) \
        and (phrase.find("проектирования") != -1):
        answer = '''
        Паттерн проектирования – это повторяемая архитектурная
конструкция,
        представляющая собой решение проблемы проектирования в рамках
некоторого часто возникающего контекста.
        '''

    return answer

def open_url(url):

```



```

"""
Function that opens the site in a browser

:param url: link to site
:return: nothing, just open site
"""
webbrowser.open(url)

def os_run(cmd):
    """
    Runs an external OS command to
    run a standard application as a subprocess

    :param cmd: Abbreviated notation for command line
    :return: nothing, just create a subprocess and runs an application
    """
    pipe = subprocess.PIPE
    p = subprocess.Popen(cmd, shell=True, stdin=pipe, stdout=pipe,
stderr=subprocess.STDOUT)

def start_application(statement):
    """
    The function responsible for selecting the desired application to
    run

    :param statement: phrase
    :return: reply to a chat with an assistant
    """
    answer = 'Это команде меня не научили'
    statement = clean_phrase(statement, ['запусти', 'запустить'])

    if (statement.find("торент") != -1) or (statement.find("торрент")
!= -1) \
        or (statement.find("медиает") != -1) or
(statement.find("mediaget") != -1):
        os.startfile('C:\\Users\\User\\MediaGet2\\mediaget.exe')
        answer = 'Торент запущен'

    elif ((statement.find("visual") != -1) or
(statement.find("визуал") != -1)
        or (statement.find("вижуал") != -1)) \
        and ((statement.find("studio") != -1) or
(statement.find("студио") != -1)):
        os.startfile("E:\\VS\\Common7\\IDE\\devenv.exe")
        answer = 'Визуал студио запущен'

    elif ((statement.find("sublime") != -1) or
(statement.find("саблайм") != -1)) \
        and ((statement.find("text") != -1) or
(statement.find("текст") != -1)):
        os.startfile("E:\\Sublime Text 3\\sublime_text.exe")
        answer = 'Саблайм текст запущен'

    elif (statement.find("скайп") != -1) or (statement.find("skype")
!= -1):

```

```

        os.startfile("C:\\Program Files (x86)\\Microsoft\\Skype for
Desktop\\Skype.exe")
        answer = 'Скайп запущен'

        elif (statement.find("телеграм") != -1) or
(statement.find("telegram") != -1):
            os.startfile("D:\\Telegram\\Telegram Desktop\\Telegram.exe")
            answer = 'Телеграмм запущен'

        elif (statement.find("гугл") != -1) or (statement.find("гугол") !=
-1) \
            or (statement.find("google") != -1):
            os.startfile("C:\\Program Files
(x86)\\Google\\Chrome\\Application\\chrome.exe")
            answer = 'Гугл запущен'

        elif (statement.find("калькулятор") != -1) or
(statement.find("calculator") != -1):
            os_run('calc')
            answer = 'Калькулятор запущен'

        elif (statement.find("блокнот") != -1) or
(statement.find("notepad") != -1):
            os_run('notepad')
            answer = 'Блокнот запущен'

        elif (statement.find("paint") != -1) or (statement.find("паинт")
!= -1):
            os_run('mspaint')
            answer = 'Пэинт запущен'

        elif (statement.find("browser") != -1) or
(statement.find("браузер") != -1):
            open_url('http://google.ru')
            answer = 'Запускаю браузер'

        elif (statement.find("проводник") != -1) or
(statement.find("файловый менеджер") != -1):
            os_run('explorer')
            answer = 'Проводник запущен'

        elif (statement.find("гитхаб") != -1) or (statement.find("github")
!= -1):
            subprocess.run(['C:\\Users\\User\\AppData\\Local\\GitHubDesktop\\GitHu
bDesktop.exe'])
            answer = 'Гитхаб запущен'

    return answer

def find_on_you_tube(phrase):
    """
    Gives link on YouTube video code for any search query

    :param phrase: youtube video request
    :return: link to the first video in the issue

```

```

"""
phrase = clean_phrase(phrase,
                        ['хочу', 'на ютубе', 'на ютюбе', 'на ютуб',
 'ютуб', 'на youtube',
                        'на you tube', 'на youtub', 'youtube', 'ю
туб', 'ютубе',
                        'посмотреть', 'смотреть'])
tmp_list_for_ends_of_links = []
compound_query =
'http://www.youtube.com/results?search_query='+quote(phrase)
doc =
urllib.request.urlopen(compound_query).read().decode('cp1251',
errors='ignore')
match = re.findall(r"\?v=(.+?)\"", doc)

if not(match is None):
    for link_collector in match:
        if len(link_collector) < 25:
            tmp_list_for_ends_of_links.append(link_collector)

tmp_dict_for_ends_of_links = dict(
    zip(tmp_list_for_ends_of_links,
tmp_list_for_ends_of_links)).values()
tmp_list_for_link = []
for ends_of_links in tmp_dict_for_ends_of_links:
    tmp_list_for_link.append(ends_of_links)
compound_youtube_link = tmp_list_for_link[0]
compound_youtube_link = 'https://www.youtube.com/watch?v=' + \
    compound_youtube_link+'?autoplay=1'
return compound_youtube_link

def browser_search(user_request):
    """
    A function that finds links to sites that match the query

    :param user_request: browser search request
    :return: list of several links to suitable sites
    """
    doc = urllib.request.urlopen(
        'http://go.mail.ru/search?fm=1&q=' +
quote(user_request)).read().decode(
    'unicode-escape', errors='ignore')
    parsed_page = re.compile('title:"(.*?)orig').findall(doc)
    tmp_search_result = []
    search_result = []

    for elements in parsed_page:
        if (elements.rfind('wikihow') == -1) and
(elements.rfind('an.yandex') == -1)\
            and (elements.rfind('wikipedia') == -1) and
(elements.rfind('otvet.mail.ru') == -1)\
            and (elements.rfind('youtube') == -1) and
(elements.rfind('.jpg') == -1)\
            and (elements.rfind('.png') == -1) and
(elements.rfind('.gif') == -1):
            answer = elements.replace(',', ' ')

```

```

        answer = answer.replace('"', '')
        answer = answer.replace('<b>', '')
        answer = answer.replace('</b>', '')
        answer = answer.split('url:')

        if len(answer) > 1:
            user_request = answer[0].split('}')
            tmp_search_result.append(user_request[0])
            user_request = answer[1].split('}')
            user_request = user_request[0].split('title')
            search_result.append(user_request[0])

    return search_result

```

Assistant_database.py

```

# Interface
from tkinter import messagebox
from tkinter import *

# Functional
import sqlite3
import os

# Files
import Assistant_functions

DATABASE_NAME = 'login_details.db'
TABLE_NAME = 'login_details'

class DatabaseFunctionSelector:
    def __init__(self):
        self.error_answer = 'Ошибка работы с базой данных'

    def selecting_database_function(self, phrase):
        """
        Selecting the function of interaction with the database

        :param phrase: User command
        :return: Function Success Phrase
        """
        answer = self.error_answer
        phrase = Assistant_functions.clean_phrase(phrase,
                                                    ['база', 'баз',
                                                     'данных'])
        if not os.path.exists(f'{DATABASE_NAME}'):
            WorkingWithDatabaseUsingSQL().create_database()

        if (phrase.find("базу") != -1) and \
            ((phrase.find("очисти") != -1) or
             (phrase.find("очистить") != -1)
              or (phrase.find("удали") != -1) or
              (phrase.find("удалить") != -1)):
            path =
os.path.join(os.path.abspath(os.path.dirname(__file__)),

```

```

'login_details.db')
    os.remove(path)

    WorkingWithDatabaseUsingSQL().create_database()
    answer = 'База данных очищена'

    elif ((phrase.find("добавь") != -1) or
(phrase.find("добавить") != -1)
        or (phrase.find("записать") != -1) or
(phrase.find("записать") != -1))\
        and ((phrase.find("логин") != -1) or
(phrase.find("пароль") != -1)
        or (phrase.find("сайт") != -1) or
(phrase.find("данные") != -1)):
        DatabaseUserInteraction().input_authentication_data()
        answer = ""Данные успешно добавлены в базу,\nвы можете
просмотреть их с помощью голосовой команды.""

    elif ((phrase.find("удали") != -1) or (phrase.find("удалить")
!= -1)) \
        and ((phrase.find("данные") != -1) or
(phrase.find("запись") != -1) or (phrase.find("сайт") != -1)):
        DatabaseUserInteraction().delete_authentication_data()
        answer = 'Готово! Помните о безопасности ваших
персональных данных!!!'

    elif ((phrase.find("напомни") != -1) or (phrase.find("какой")
!= -1)) \
        and ((phrase.find("пароль") != -1) or
(phrase.find("логин") != -1)):
        DatabaseUserInteraction().output_authentication_data()
        answer = 'Готово! Помните о безопасности ваших
персональных данных!!!'

    return answer

class WorkingWithDatabaseUsingSQL:
    def __init__(self):
        self.database_name = DATABASE_NAME
        self.db_table_name = TABLE_NAME

    def create_database(self):
        """
        Creating a database to store user authentication data

        :return: Nothing
        """
        conn = sqlite3.connect(f'{self.database_name}')
        curs = conn.cursor()

        curs.execute('''CREATE TABLE login_details
(website VARCHAR(30) PRIMARY KEY,
login VARCHAR(30),
password VARCHAR(30))''')

        curs.close()

```

```

        conn.close()

    def insert_in_database(self, inserted_data):
        """
        Adds user data to the database

        :param inserted_data: User data
        :return: nothing
        """
        conn = sqlite3.connect(f'{self.database_name}')
        curs = conn.cursor()
        insert_template = f'''INSERT INTO {self.db_table_name}
        (website, login, password) VALUES(?, ?, ?)'''
        curs.execute(insert_template,
                      (f'{inserted_data[0]}', f'{inserted_data[1]}',
f'{inserted_data[2]}'))
        conn.commit()
        curs.close()
        conn.close()

    def delete_from_database(self, deleted_data):
        """
        Delete user data to the database

        :param deleted_data: User data
        :return: nothing
        """
        print(deleted_data)
        conn = sqlite3.connect(f'{self.database_name}')
        curs = conn.cursor()
        curs.execute(f'''DELETE FROM {self.db_table_name}
        WHERE website = "{deleted_data}"''')
        conn.commit()
        curs.close()
        conn.close()

    def get_from_database(self, requested_website):
        """
        Searches the required user data in the database

        :param requested_website: The site for which the data is
needed
        :return: User authentication data
        """
        conn = sqlite3.connect(f'{self.database_name}')
        curs = conn.cursor()
        curs.execute(f'''SELECT * FROM {self.db_table_name}
        WHERE website = "{requested_website}"''')
        authentication_data = curs.fetchall()
        curs.close()
        conn.close()
        if not authentication_data:
            return 0
        return authentication_data[0]

```

```

class DatabaseUserInteraction:

```

```

def __init__(self):
    self.root = Tk()

def output_authentication_data(self):
    """
    Displaying user data in the interface

    :return: Nothing
    """

    def btn_click():
        site = str(site_input.get())
        if len(site) > 0:
            authentication_data =
WorkingWithDatabaseUsingSQL().get_from_database(site)
            if authentication_data == 0:
                messagebox.showerror(title='Запись не найдена',
                                     message=f'''В базе не
существует записи с данным ресурсом''')
            else:
                login = str(authentication_data[1])
                password = str(authentication_data[2])
                messagebox.showwarning(title='Данные',
                                     message=f'''Логин: {login}\
Пароль: {password}''')
        else:
            messagebox.showerror(title='Поле не заолнено',
                                 message=f'''Необходимо заполнить
поле: "Название ресурса"''')

    self.root['bg'] = '#ffffff'
    self.root.title('Будте осторожны со своей персональной
информацией')
    self.root.wm attributes('-alpha', 0.99)
    self.root.geometry('400x450')

    self.root.resizable(width=False, height=False)

    frame = Canvas(self.root, bg='white')
    frame.place(relx=0.0, rely=0.0, relwidth=1.0, relheight=1.0)

    img_png = PhotoImage(file='img/data_security.png')
    frame.create_image(200, 250, image=img_png)

    title = Label(frame, text='Введите нужный ресурс', bg='white',
font=40)
    title.pack(pady=15)

    site_input = Entry(frame, bg='white')
    btn_find = Button(frame, text='Найти', bg='red',
command=btn_click)
    btn_done = Button(frame, text='Готово',
command=self.root.destroy)

    site_input.pack(side=TOP)
    btn_find.pack(side=TOP, pady=15)
    btn_done.pack(side=BOTTOM, pady=30)

```

```

        self.root.mainloop()

    def delete_authentication_data(self):
        """
        Deleting user data by the interface

        :return: Nothing
        """

    def btn_click():
        site = str(site_input.get())
        if len(site) > 0:
            authentication_data =
WorkingWithDatabaseUsingSQL().get_from_database(site)
            if authentication_data == 0:
                messagebox.showerror(title='Запись не найдена',
                                     message=f'''В базе не
существует записи с данным ресурсом''')
            else:

WorkingWithDatabaseUsingSQL().delete_from_database(site)
                messagebox.showwarning(title='Успех!',
                                     message=f'''Данные успешно удалены''')
            else:
                messagebox.showerror(title='Поле не заолнено',
                                     message=f'''Необходимо заполнить
поле: "Название ресурса"''')

        self.root['bg'] = '#ffffff'
        self.root.title('Будте осторожны со своей персональной
информацией')
        self.root.wm_attributes('-alpha', 0.99)
        self.root.geometry('400x450')

        self.root.resizable(width=False, height=False)

        frame = Canvas(self.root, bg='white')
        frame.place(relx=0.0, rely=0.0, relwidth=1.0, relheight=1.0)

        img_png = PhotoImage(file='img/data_security.png')
        frame.create_image(200, 250, image=img_png)

        title = Label(frame, text='Введите нужный ресурс', bg='white',
font=40)
        title.pack(pady=15)

        site_input = Entry(frame, bg='white')
        btn_find = Button(frame, text='Удалить', bg='red',
command=btn_click)
        btn_done = Button(frame, text='Готово',
command=self.root.destroy)

        site_input.pack(side=TOP)
        btn_find.pack(side=TOP, pady=15)
        btn_done.pack(side=BOTTOM, pady=30)

```



```

self.root.mainloop()

def input_authentication_data(self):
    """
    Displays a window for writing user data

    :return: Nothing
    """

def btn_click():
    """
    Saves data when button is pressed

    :return: nothing
    """
    site = site_input.get()
    login = login_input.get()
    password = password_input.get()

    if len(site) > 0 and len(login) > 3 and len(password) > 3:
        unique_site_check =
WorkingWithDatabaseUsingSQL().get_from_database(site)
        if unique_site_check == 0:

WorkingWithDatabaseUsingSQL().insert_in_database([site, login,
password])
        else:
            messagebox.showerror(title='Ошибка уникальности',
                                message=f'''Запись с таким
названием ресурса уже существует и содержит данные:

                                Название: {unique_site_check[0]}
                                Логин: {unique_site_check[1]}
                                Пароль: {unique_site_check[2]}''')

    else:
        fields = [site, login, password]
        fields_names = ['Название ресурса', 'Логин', 'Пароль']

        fields_that_must_be_filled = [fields_names[x] for x in
range(len(fields)) if len(fields[x]) < 4]
        messagebox.showerror(title='Поля не заолнены',
                                message=f'''Необходимо заполнить
поле: {fields_that_must_be_filled}''')

    self.root['bg'] = '#ffffff'
    self.root.title('Будте осторожны со своей персональной
информацией')
    self.root.wm_attributes('-alpha', 0.99)
    self.root.geometry('400x450')

    self.root.resizable(width=False, height=False)

    frame = Canvas(self.root, bg='white')
    frame.place(relx=0.0, rely=0.0, relwidth=1.0, relheight=1.0)

    img_png = PhotoImage(file='img/data_security.png')

```

```

        frame.create_image(200, 270, image=img_png)

        title = Label(frame, text='Введите данные', bg='white',
font=40)
        title.pack()

        site_input = Entry(frame, bg='white')
        site_input.pack(pady=4)

        user_label_site = Label(bg='white')
        user_label_site.pack(anchor=NW, padx=25, pady=5)

        user_label_site = Label(text="Название ресурса:", bg="white")
        user_label_site.pack(anchor=NW, padx=25, pady=1)

        login_input = Entry(frame, bg='white')
        login_input.pack(pady=3)

        user_label_login = Label(text="Логин:", bg="white")
        user_label_login.pack(anchor=NW, padx=88, pady=4)

        password_input = Entry(frame, bg='white', show='*')
        password_input.pack(pady=3)

        user_label_password = Label(text="Пароль:", bg="white")
        user_label_password.pack(anchor=NW, padx=79, pady=1)

        btn = Button(frame, text='Сохранить', bg='red',
command=btn_click)
        btn.pack(pady=3)

        btn_done = Button(frame, text='Готово',
command=self.root.destroy)
        btn_done.pack(side=BOTTOM, pady=20)

        self.root.mainloop()

```

ПРИЛОЖЕНИЕ В
(обязательное)
Ведомость документов

Зона	Обозначения	Наименование	Дополнительные сведения
		<u>Текстовые документы</u>	
	ГУИР КП 1-40 05 01-10 018	Пояснительная записка	52 с.
		<u>Графические документы</u>	
	ГУИР.501693.02 18 Д1	Схема алгоритма	Формат А3
	ГУИР.501693.018 Д2	UML диаграмма классов	Формат А3
	ГУИР.501693.018 Д3	Диаграмма последовательности	Формат А3
	ГУИР.501693.018 Д3	Диаграмма состояний	Формат А3
	ГУИР.501693.005 Д3	Структура графического	Формат А3
		Пользовательского интерфейса	
		БГУИР КП 1-40 05 01-10 018 ПЗ	
Изм.	Лист	№ документа	Подпись
Разраб.		Невейков А.С.	
Провер.		Горбач	
Т. Контр.		Гриб	
Утв.		Хорошко В.В.	
		Программное средство «Голосовой ассистент»	<div> <div>Литера</div> <div>Масса</div> <div>Масштаб</div> </div> <div> <div>Лист</div> <div>52</div> <div>Листов</div> <div>52</div> </div>
			Кафедра ПИКС, группа 014301