МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования «Южно-Уральский государственный университет
(национальный исследовательский университет)»

Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой ЭВМ
_____Д. В. Топольский
«___»_____2022 г.

ГОЛОСОВОЕ УПРАВЛЕНИЕ ПК ДЛЯ ЛЮДЕЙ С ОГРАНИЧЕННЫМИ ВОЗМОЖНОСТЯМИ ПО ИСПОЛЬЗОВАНИЮ УСТРОЙСТВ РУЧНОГО ВВОДА

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУРГУ-09.03.01.2022.238 ПЗ ВКР

Руководитель работы,
к.п.н., доцент каф. ЭВМ
_____М. А. Алтухова
«___»_____2022 г.

Автор работы,
студент группы КЭ-406
_____А. А. Миронов
«___»_____2022 г.

Нормоконтролёр,
к.п.н., доцент каф. ЭВМ
_____М. А. Алтухова
«___»_____2022 г.

Челябинск-2022

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

УТВЕРЖДАЮ
Заведующий кафедрой ЭВМ
_____Д.В. Топольский
«___»_____2022 г.

ЗАДАНИЕ
на выпускную квалификационную работу бакалавра
студенту группы КЭ-406
Миронову Андрею Андреевичу
обучающемуся по направлению
09.03.01 «Информатика и вычислительная техника»

1. Тема работы: «Голосовое управление ПК для людей с ограниченными возможностями по использованию устройств ручного ввода»

2. Срок сдачи студентом законченной работы: 1 июня 2022 г.

3. Исходные данные к работе:

Голосовой интерфейс разрабатывается с целью упрощения взаимодействия с компьютером людей с ограниченными возможностями.

Приложение должно соответствовать следующим функциональным требованиям:

– распознавание голосовых команд пользователя на открытие файлов, запуск приложений, ввод поискового запроса в интернет-браузер;

– интерпретация голосовых команд пользователя в инструкции для компьютера;

– наличие функции добавления списка программ для быстрого открытия их голосовыми командами;

– наличие функции запрета для программ, чтобы голосовой интерфейс не реагировал на команды для запуска данного приложения;

– наличие возможности выбора микрофона, с которого будет прослушиваться речь;

– наличие возможности отключения или включения уведомлений;

– наличие возможности включения или отключения звука при появлении уведомлений;

– наличие возможности выполнения настроек как с помощью графического интерфейса, так и с помощью голосовых команд.

4. Перечень подлежащих разработке вопросов:

В процессе выполнения выпускной квалификационной работы требуется:

– провести анализ и сравнительный обзор аналогичных решений;

– на основе сравнительного анализа провести выбор средств реализации;

– уточнить требования к приложению;

– разработать приложение;

– протестировать работоспособность приложения на различных операционных системах.

Дата выдачи задания: 1 декабря 2021 г.

Руководитель работы _____ /М. А. Алтухова/

Студент _____ /А. А. Миронов/

КАЛЕНДАРНЫЙ ПЛАН

Этап	Срок сдачи	Подпись руководителя
Введение и обзор литературы	07.02.2022	
Разработка модели, проектирование (проведён анализ и сравнительный обзор аналогичных решений. На основе сравнительного анализа проведён выбор средств реализации. Уточнены требования к приложению)	07.03.2022	
Реализация системы (разработано приложение с пользовательским интерфейсом)	04.04.2022	
Тестирование, отладка (протестирована работоспособность приложения на различных операционных системах	10.05.2022	
Компоновка текста работы и сдача на нормоконтроль	16.05.2022	
Подготовка презентации и доклада	24.05.2022	

Руководитель работы _____ /М. А. Алтухова/

Студент _____ /А. А. Миронов/

АННОТАЦИЯ

А. А. Миронов Методические указания к выполнению выпускных квалификационных работ. – Челябинск: ФГАОУ ВО «ЮУрГУ (НИУ)», ВШ ЭКН; 2022, 65 с., 19 ил., библиогр. 12 наим.

В рамках выпускной квалификационной работы произведён детальный анализ актуальных на данный момент голосовых интерфейсов для ПК на ОС Windows. Рассмотрены их преимущества и недостатки. Определён необходимый функционал голосового интерфейса. Обоснован выбор языка программирования, средств для реализации и среды программирования. Разработана программа для распознавания и выполнения голосовых команд. Выполнен анализ работоспособности системы.

ОГЛАВЛЕНИЕ

ПЕРЕЧЕНЬ ОБОЗНАЧЕНИЙ И СОКРАЩЕНИЙ

ГБ – гигабайт

ГГц – гигагерц

Кбит/с – килобит в секунду

МБ – мегабайт

ОЗУ – оперативное запоминающее устройство

ОС – операционная система

ПК – персональный компьютер

ВВЕДЕНИЕ

Голосовое управление различными устройствами значительно упростило их использование. Голосовые помощники в основном распространены на мобильных устройствах и «умных» колонках, а на стационарных компьютерах имеют большую популярность голосовые интерфейсы. За последнее время, множество компаний стремятся к развитию голосового управления для различных технических устройств [1].

Со временем возможность использования голосового управления становится наиболее популярной функцей. Более 50% пользователей голосовых интерфейсов используют их дома для покупок в интернете и более 71% для поиска информации в интернете [2].

Голосовые интерфейсы наиболее актуальны для людей с ограниченными возможностями. Для них особенно важна именно реализация интерфейса программы. Данный вариант голосового помощника подходит для пользователей с нарушениями зрения, слуха, опорно-двигательного аппарата.

Голосовой интерфейс позволяет отказаться от ввода информации вручную, что значительно ускоряет и упрощает взаимодействие с устройствами. Использование голосового ввода также имеет свои минусы, самыми главными из них, является то, что на вводимую информацию могут повлиять фоновые шумы, что скажется на восприятии команды устройством.

Цель работы: создать программу, способную распознавать голосовые команды и интерпретировать их в инструкции для ПК.

Необходимые задачи для достижения цели ВКР:

– провести сравнительный обзор аналогичных решений;

– на основе сравнительного анализа провести выбор средств реализации;

– уточнить требования к приложению;

– разработать приложение;

– протестировать работоспособность приложения на различных операционных системах.

Работа состоит из пяти глав. В первой главе описаны существующие на данный момент аналоги. Во второй главе определены функциональные и нефункциональные требования разрабатываемого программного продукта. В третьей главе приведены алгоритмы работы программы. В четвертой главе представлена реализация голосового интерфейса. В пятой главе описано проведённое тестирование.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Обзор аналогов

Голосовое управление на персональном компьютере должно оцениваться по следующим критериям:

– количество функций;

– скорость работы;

– поддержка русского языка;

– стоимость;

– необходимость наличия интернет-соединения.

«Cortana» – голосовой интерфейс, интегрированный в операционную систему Windows 10 и созданный компанией Microsoft. Она помогает систематизировать и распланировать задачи на определенный период, напоминает о выполнении каких-либо действий, предоставляет по запросу пользователя информацию. Также имеет встроенный функционал для ответов на общие вопросы, используя поиск Bing. Вводить информацию можно с помощью голоса и клавиатуры в текстовой форме. Не обязательно наличие интернет-соединения.

Функции: поиск в интернете, поиск на компьютере, открытие файлов на компьютере, установка будильника, установка напоминаний.

Преимущества:

– возможность работы как с интернетом, так и без него;

– высокая скорость работы;

– имеет широкий набор функций.

Недостатки:

– отсутствие русского языка;

– нельзя использовать другие поисковые системы;

– не доступна в регионе РФ.

Typle – голосовой интерфейс, разработанный только под семейство ОС Windows (XP, Vista, Windows 7-10). Ввод информации осуществляется только голосом. Обязательно наличие интернет-соединения.

Функции: поиск на компьютере, открытие файлов на компьютере, предварительно внесённых в список.

Преимущества:

– возможность добавлять наиболее часто используемые сайты в список для быстрого открытия голосовой командой;

– полная поддержка русского языка.

Недостатки:

– воспринимает фоновый шум как команду;

– медленная скорость работы;

– не может открывать произвольные файлы на компьютере;

– может открывать только интернет-страницы и заранее занесённые в список файлы;

– ограниченное количество используемых голосовых команд – 15.

Горыныч – голосовой интерфейс, разработанный только под семейство ОС Windows (XP, Vista, Windows 7). Данный голосовой ассистент умеет не только осуществлять запрос пользователя по поиску различной информации, но и работать с программами и приложениями внутри системы. Вводить информацию можно с помощью голоса и клавиатуры в текстовой форме. Обязательно наличие интернет-соединения.

Функции: поиск в интернете, открытие файлов на компьютере.

Преимущества:

– полная поддержка русского языка;

– возможность работы с файлами и приложениями внутри системы.

Недостатки:

– для корректного восприятия команд, речь должна быть без дефектов;

– медленная скорость работы;

– в данный момент программа не разрабатывается;

– маленький запас кодовых слов;

– невозможность добавления собственных кодовых слов.

«Окей, блокнотик!» – голосовой интерфейс для создания заметок и напоминаний, работает лишь со звонками и сообщениями. Обязательно наличие интернет-соединения.

Функции: создание заметок, создание напоминаний.

Преимущества:

– полная поддержка русского языка;

– быстрая скорость работы.

Недостатки:

– ограниченная сфера применения.

Vani Dialer – голосовой интерфейс для работы со звонками и сообщениями. Обязательно наличие интернет-соединения.

Функции: создание напоминаний, возможность отправить текстовое сообщение голосом.

Преимущества:

– полная поддержка русского языка;

– быстрая скорость работы.

Недостатки:

– ограниченная сфера применения.

Результаты обзора аналогов обобщены в таблице 1.

Таблица 1 – Сводная таблица обзора аналогов

	Cortana	Typle	Горыныч	Окей, блокнотик!	Vani Dialer
Поиск в интернете	+	-	+	-	-
Поиск на компьютере	+	+	-	-	-
Создание заметок	+	-	-	+	+
Создание напоминаний	+	-	-	+	–
Установка будильника	+	-	-	-	-
Открытие файлов на компьютере	+	+	+	-	-
Время отклика программы	1 сек	3 сек	2.5 сек	0.5 сек	0.5 сек
Поддержка русского языка	-	+	+	+	+
Стоимость	360$	Бесплатно	Бесплатно	Бесплатно	Бесплатно
Интернет-соединение	Не обязательно	Обязательно	Обязательно	Обязательно	Обязательно

1.2 Выбор средств реализации

1.2.1 Выбор операционных систем

Для разработки голосового ассистента первым делом нужно определить, под какие версии Windows будет вестись разработка. Рассмотрим список наиболее популярных операционных систем (рисунок 1).

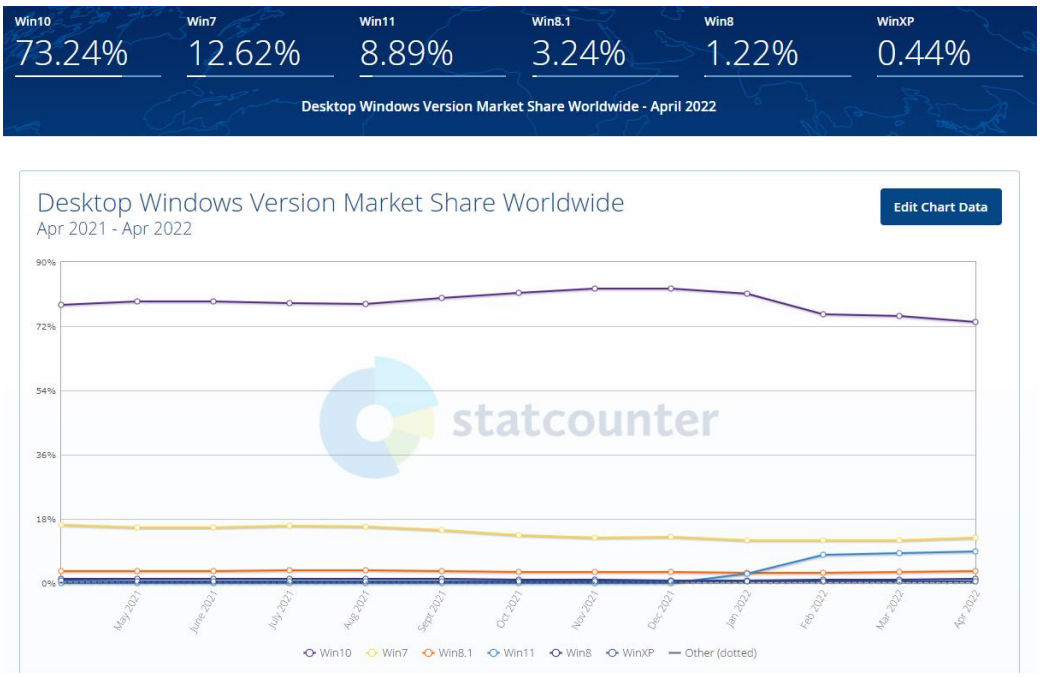

Рисунок 1 – Популярность операционных систем на апрель 2022 г.

Наиболее популярными являются Windows 10, Windows 7, Windows 11, Windows 8.1 [3]. Ввиду того, что Windows 11 вышла совсем недавно и на неё регулярно выходят обновления с исправлениями ошибок, её в качестве поддерживаемой операционной системы рассматривать не будем. Windows 8.1 является доработанной версией Windows 8 под персональные компьютеры, в следствии этого, будет рассматривать только Windows 8.1.

1.2.2 Выбор языка программирования и среды разработки

Разработка голосового интерфейса возможна на любом современном языке программирования. Рассмотрим наиболее популярные из них по версии сайта TIOBE [4] (рисунок 2).

May 2022	May 2021	Change	Programming Language	Ratings	Change
1	2	^	Python	12.74%	+0.86%
2	1	v	C	11.59%	-1.80%
3	3		C++	8.83%	+1.01%
4	4		C#	6.39%	+1.98%

Рисунок 2 – Рейтинг языков программирования по версии TIOBE

По итогу на май 2022 года самыми популярными языками программирования являются Python, C#, C/C++ [4].

Сравним данный рейтинг с наиболее популярными языками программирования по версии сайта PyPL [5] (рисунок 3).

Worldwide, May 2022 compared to a year ago:

Rank	Change	Language	Share	Trend
1		Python	27.85 %	-2.5 %
2		C#	7.62 %	+0.7 %
3		C/C++	7.0 %	+0.4 %

Рисунок 3 – Рейтинг языков программирования по версии PyPL

Можно заметить, что единственное различие состоит в месте языка C/C++.

Поэтому в качестве языков для разработки голосового ассистента будем рассматривать следующие языки: Python, C#, C++. Определим критерии выбора: скорость работы конечного продукта, количество библиотек, подходят ли библиотеки, созданные для этого языка, чтобы реализовать разрабатываемую программу, скорость разработки программы, способы типизации данных.

Типизация данных бывает двух типов: статическая и динамическая. При статической типизации данных, программист сам указывает, где и какой тип данных используется. При динамической типизации все типы данных определяются на этапе выполнения программы [6].

Рассмотрим Python: данный язык не отличается высокой скоростью работы из-за динамической типизации данных, но компенсирует это большим количеством библиотек, созданных для упрощения разработки программ на нём,

так и понятным синтаксисом, в следствии чего, программы на нём писать достаточно быстро и удобно. Также следует отметить, что библиотеки для языка Python хорошо подходят для реализации разрабатываемой программы.

Далее рассмотрим язык C#: данный язык имеет среднюю скорость работы, имеет статическую типизацию данных, данный язык имеет достаточно большое количество библиотек, которые ввиду используемой платформы для C# .NET плохо подходят для реализации разрабатываемой программы. Данный язык имеет достаточно простой и понятный синтаксис, в следствии чего разработка программ на нём занимает мало времени.

Последним языком для рассмотрения остался C++: данный язык имеет одну из самых высоких скоростей работы, что и определило его частое использование в вычислительной технике. C++ имеет строгую типизацию данных, из недостатков этого языка следует отметить его сложный синтаксис, в следствии чего, разработка программ на данном языке является не самой простой задачей. Также этот язык не имеет библиотек для реализации разрабатываемой программы.

Исходя из вышеперечисленного, для разработки программы выберем язык Python. На данный момент разработки программы этот язык имеет 4 актуальные версий: Python 3.7, Python 3.8, Python 3.9, Python 3.10 [7]. Для разработки выберем Python 3.7, который на данный момент имеет большее количество поддерживаемых библиотек.

Python имеет несколько сред разработки: Atom, SumlimeText, VisualStudio, Komondo IDE, PyCharm. Так как среда разработки отвечает только за графическую часть отображения кода, то её выбор основывается на личных предпочтениях, поэтому, выбираем PyCharm.

Для разработки пользовательского интерфейса используется библиотека PyQt5 [8] и сам Qt.

1.2.3 Выбор библиотеки для распознавания голосовых команд

Для языка Python есть множество библиотек для распознавания голоса, рассмотрим бесплатные: SpeechRecognition [9], PoketSphinx [10].

Библиотека PoketSphinx не может распознавать речь на русском языке, следовательно, выбираем SpeechRecognition.

Библиотека SpeechRecognition позволяет использовать следующие интерфейсы для распознавания голоса:

– CMU sphinx (работает без подключения к интернету);

– google speech recognition;

– google cloud speech API;

– wit.ai;

– microsoft azure speech;

– houndify API;

– IBM speech to text;

– snowboy hotword detection (работает без подключения к интернету);

Так как интерфейсы: Google Cloud Speech, Wit.ai, Microsoft Azure Speech, Houndify API, IBM Speech to Text, Snowboy Hotword Detection запрашивают регистрацию ключа, их можно отбросить сразу.

Недостатком CMU Sphinx, как ни странно, является его возможность работы без подключения к интернету. При первом запуске программы он создаёт внутренний словарь, по которому будет распознавать речи, а так как разрабатываемая программа должна запускаться сразу при включении компьютера, то данное ожидание, которое в среднем занимает 4 минуты, не позволительно.

Из вышеперечисленного остаётся только Google Speech Recognition, её и будем использовать в разработке.

1.2.4 Библиотеки, используемые в разработке

Так как пользователь может говорить команды и названия, отличающиеся от тех, которые прописаны в коде, нужно использовать библиотеку, для определения наибольшего соответствия. Для этого нам необходимо вычислять расстояние Левенштейна [11]. Расстояние Левенштейна – метрика, позволяющая определить «схожесть» двух строк – минимальное количество операций вставки одного символа, удаления одного символа и замены одного символа на другой, необходимых для превращения одной строки в другую. Данное математическое решение отлично реализует библиотека fuzzywuzzy.

Для работы с файлами на компьютере будет использоваться библиотека subprocess.

Для открытия поисковых запросов пользователя в интернете будет использоваться библиотека webbrowser.

Для записи данных в файл будет использоваться библиотека json.

1.3 Вывод

На основе анализа аналогов разрабатываемой программы, были определены их основные преимущества и недостатки. Главным недостатком был недостаточный функционал голосовых интерфейсов. Из преимуществ некоторых систем можно отметить возможность работы как с компьютером, так и с интернетом.

В качестве языка программирования был выбран Python, поскольку он имеет множество библиотек, подходящих для данной разработки.

Для выполнения выпускной квалификационной работы была выбрана интегрированная среда разработки – PyCharm.

2 ОПРЕДЕЛЕНИЕ ТРЕБОВАНИЙ

К общим требованиям можно отнести минимальные системные требования для персональных компьютеров:

– ОС: Windows 7 (64 бит) или выше;

– процессор: 2.1 ГГц Intel Core 2 Duo, AMD Athlon 64 Dual-Core 4000+ или аналогичный;

– оперативная память: 256 МБ ОЗУ;

– место на диске: 256 МБ;

– звуковая карта: звуковая карта, совместимая с DirectX;

– наличие микрофона;

– скорость интернет-соединения: 1024 Кбит/с.

Требования, перечисленные выше, были составлены на основе системных требований программ, а также на основе технических решений, используемых для разработки.

2.1 Функциональные требования

Система должна соответствовать следующим функциональным требованиям:

– распознавание голосовых команд пользователя на открытие файлов, запуск приложений, ввод поискового запроса в интернет-браузер;

– интерпретация голосовых команд пользователя в инструкции для компьютера;

– наличие функции добавления списка программ для быстрого открытия их голосовыми командами;

– наличие функции запрета для программ, чтобы голосовой интерфейс не реагировал на команды для запуска данного приложения;

– наличие возможности выбора микрофона, с которого будет прослушиваться речь;

– наличие возможности отключения или включения уведомлений;

– наличие возможности включения или отключения звука при появлении уведомлений;

– наличие возможности выполнения настроек как с помощью графического интерфейса, так и с помощью голосовых команд.

2.2 Нефункциональные требования

Система должна соответствовать следующим нефункциональным требованиям:

– приложение должно быть разработано на платформе, предоставляющей возможность бесплатного распространения готовой системы;

– приложение должно работать на компьютерах со слабыми техническими характеристиками;

– пользовательский интерфейс приложения должен иметь хорошо читаемый шрифт.

3 ПРОЕКТИРОВАНИЕ

3.1 Алгоритм работы программы

На рисунках 4–5 приведены алгоритмы работы программы.

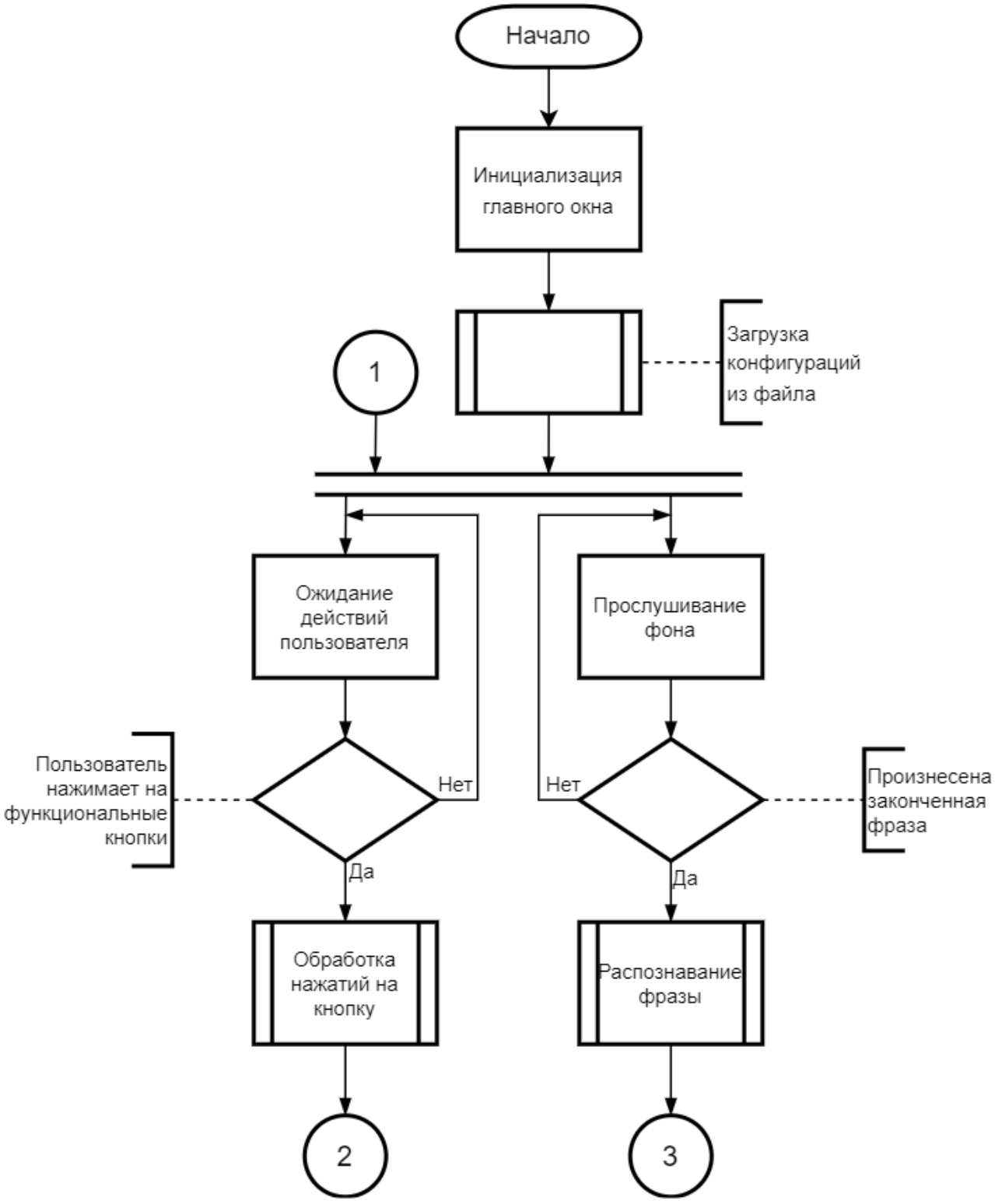

Рисунок 4 – Алгоритм работы программы

На рисунке 4 приведён алгоритм работы программы. Программа запускается при включении персонального компьютера. После этого программа будет постоянно слушать пользователя. Если пользователь начнёт говорить, то программа через 0,5 секунд после завершения фразы перейдёт на блок распознавания речи. Код, отвечающий за прослушивание голоса показан в приложении Д (листинги Д.3, Д.4). Одновременно с этим пользователь может работать с ассистентом вручную, посредством нажатий на кнопки. Код, отвечающий за работоспособность кнопок показан в приложении Г (листинг Г.1). Если голос пользователя распознался или пользователь нажал на одну из кнопок, то программа выполняет действия, описанные в алгоритме на рисунке 5.

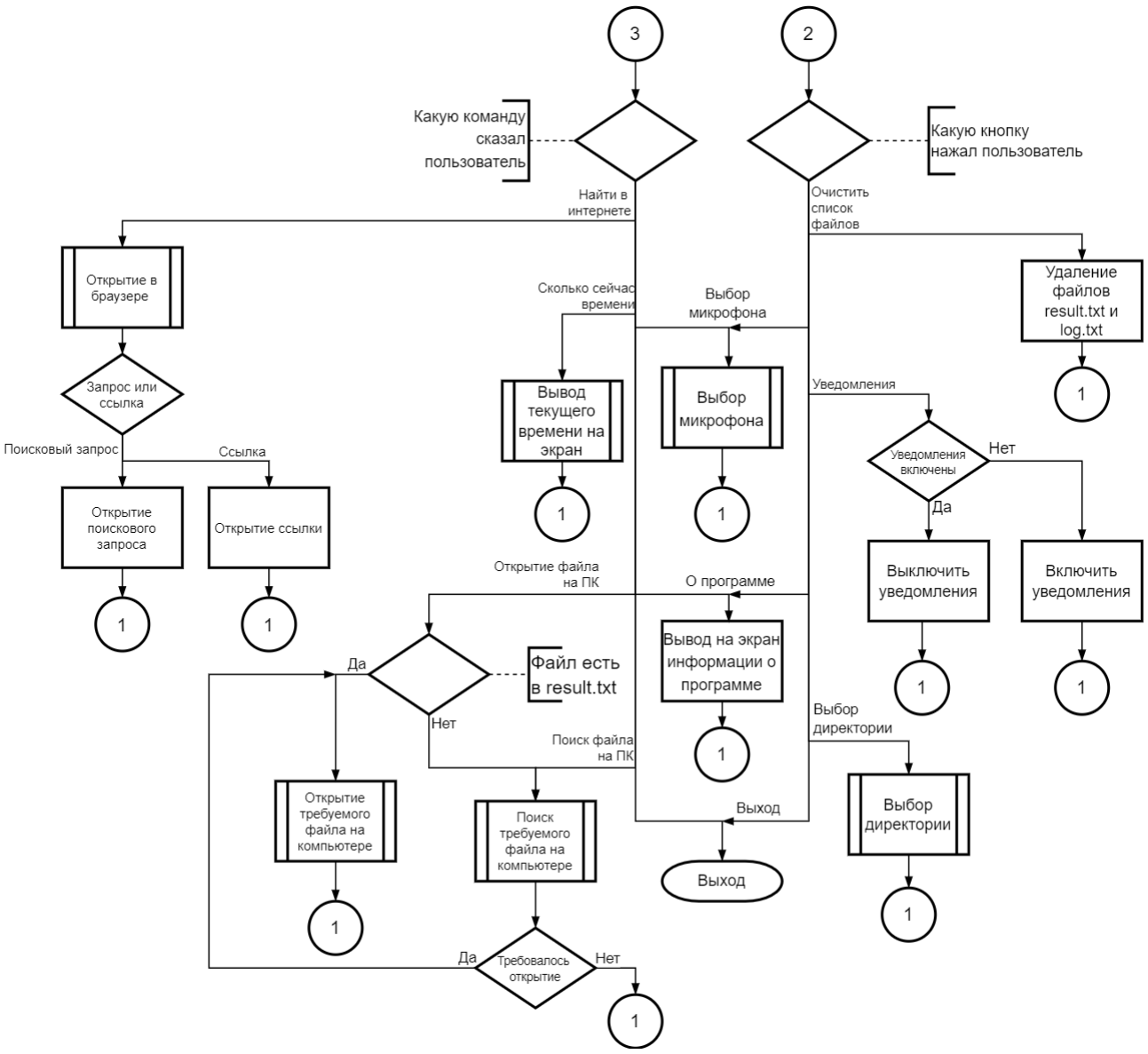

Рисунок 5 – Алгоритм распознавания команд и нажатий клавиш

Если переход на алгоритм, указанный на рисунке 5, произошёл после распознавания команды алгоритмом, приведённом на рисунке 4, то программа начинает определять, какую команду произнёс пользователь. Код, отвечающий за распознавание команды показан в приложении Д (листинги Д.1, Д.2) Программа распознаёт 7 видов голосовых команд:

– найти в интернете;

– сколько сейчас времени;

– выбор микрофона;

– о программе;

– открыть файл на ПК;

– поиск файлов на ПК;

– выход.

Команда «Найти в интернете» реализует возможность с помощью голоса ввести в поисковую строку браузера запрос или ссылку. Реализацию данной команды можно увидеть в приложении В (листинг В.7). Команда «Сколько сейчас времени» даёт пользователю возможность узнать текущее время. Код данной команды показан в приложении В (листинг В.6). Команда «Выбор микрофона» позволяет с помощью голоса поменять микрофон, с которого будет прослушиваться речь пользователя. Реализацию данной команды можно увидеть в приложении В (листинг В.9). Команда «О программе» вызывает всплывающее меню, в котором указана информация о программе. Код данной команды показан в приложении В (листинг В.4). Команда «Открыть файл на ПК» позволяет открыть любой файл с помощью голоса. Изначально проверяется текстовый документ с названием «result.txt», в котором содержатся все найденные файлы. Если в данном документе нет пути к требуемому файлу, то осуществляется поиск файла на ПК. После того, как поиск завершился, файл будет открыт сразу. Реализацию данной команды можно увидеть в приложении В (листинг В.2). Команда «Поиск файла на ПК» реализует поиск файла на компьютере, при нахождении файла его путь, его название и расширение записывается в «result.txt». Код данной команды показан в приложении В (листинг В.1). Команда «Выход» закрывает программу. Реализацию данной команды можно увидеть в приложении В (листинг В.5).

Если переход на алгоритм, указанный на рисунке 5, произошёл после нажатия на кнопку, то программа сразу попадает на функцию, соответствующую данной кнопке. Приложение содержит в себе 6 кнопок:

– очистить список файлов;

– уведомления;

– выбор микрофона;

– выбор директории;

– о программе;

– выход.

При нажатии на кнопку «Очистить список файлов» происходит удаление файлов «log.txt» и «result.txt». Реализацию данной кнопки можно увидеть в приложении Д (листинг Д.5). При нажатии на кнопку «уведомления» пользователь может отключить или включить уведомления. Код данной кнопки показан в приложении Д (листинг Д.6). При нажатии на кнопку «Выбор микрофона» происходит инициализация окна выбора микрофона. Код инициализации этого окна показан в приложении А (листинг А.2). Функционал данного окна описан в пункте 4.2. При нажатии на кнопку «Выбор директории» происходит инициализация окна выбора директории, инициализация данного окна продемонстрирована в приложении А (листинг А.3). Функционал данного окна описан в пункте 4.3. При нажатии на кнопку «О программе» вызывается всплывающее меню, в котором указана информация о программе. Код данной кнопки показан в приложении Д (листинг Д.7). Кнопка «Выход» закрывает программу. Реализацию данной кнопки можно увидеть в приложении Д (листинг Д.8).

3.2 Тенденции к дальнейшей разработке

В рамках данной ВКР было разработано приложение, отвечающее всем пунктам задания. Однако для комфортного использования программного средства желательно дальнейшая доработка с целью введения дополнительных возможностей:

– озвучивание голосом того, что делает интерфейс: для реализации этого необходимо использовать библиотеку pyttsx3 с дополнительными голосовыми модулями;

– открытие результатов поисковых запросов;

– работа с браузером посредством голосовых команд: так как каждый сайт является набором строк кода, структурированных определённым образом, возможно эту структуру просмотреть программно и выдать в виде, в котором

пользователь сможет воспринимать данную информацию. Для реализации этой части необходимо использовать парсинг. Парсинг – процесс сбора данных и их структурирования. Для этого необходимо «обойти» сайт или ссылку и найти данные, которые будут соответствовать запросу;

– предоставление пользователю возможности самостоятельной настройки фразы для активации голосового интерфейса.

4 РЕАЛИЗАЦИЯ

Для реализации программы использовалось:

– интегрированная среда разработки PyCharm;

– библиотеки для языка Python: PyQt5, SpeechRecognition, WebBrowser, subprocess, fuzzywuzzy;

– среда проектирования пользовательского интерфейса Qt.

Для сохранения файлов проекта использовался репозиторий на GitHub, взаимодействие с которым происходило с помощью программы для управления версий GitKraken.

При установке программы на ПК, она автоматически добавляется в автозапуск, поэтому дополнительных действий по включению программы после запуска компьютера не требуется.

4.1 Основное окно

При открытии программы вручную из области уведомлений, перед пользователем появляется главное окно программы, отображаемого на рисунке 6. На данном окне пользователь может перейти в разделы для дополнительных настроек, разделы дополнительных настроек обозначены как: «опции», а также ему показывается результат распознавания его команд. Инициализация основного окна показана в приложении А (листинг А.1).

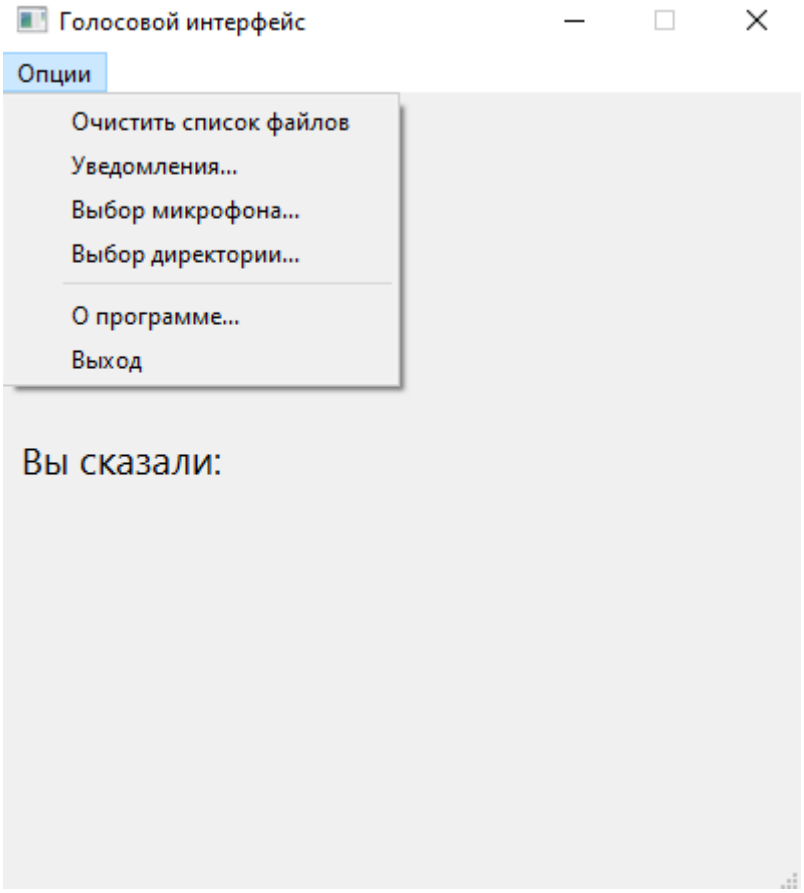

Рисунок 6 – Главное окно программы

Когда пользователь использует раздел «опции» появляется выпадающее меню, в котором содержатся пункты для отчистки файла с логами и результатами поиска, для отключения или включения уведомлений, для настройки микрофона, с которого будет прослушиваться речь пользователя, для выбора директории, куда будут сохраняться файлы, созданные программой, также пользователь сможет узнать о программе и закрыть её.

4.2 Окно выбора микрофона

При нажатии на строку «Выбор микрофона…» пользователь попадёт в окно, указанное на рисунке 7. Инициализация данного окна показана в приложении А (листинг А.2).

Рисунок 7 – Окно выбора микрофона

Данное окно содержит выпадающий список со всеми устройствами, которые могут идентифицироваться как входное звуковое устройство (рисунок 8).

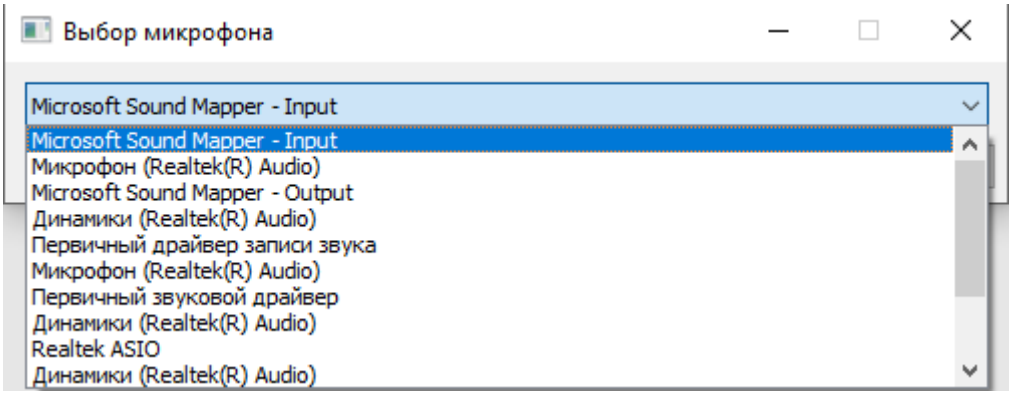

Рисунок 8 – Выпадающий список с звукозаписывающими устройствами

Также в данном окне присутствуют кнопки «Ок» и «Отмена». При нажатии на кнопку «Ок» пользователь подтверждает выбор микрофона, выбранный микрофон сохраняется в конфигурационный файл и в дальнейшем будет использоваться для восприятия речи пользователя программой. При нажатии на кнопку «Отмена» окно закрывается, выбранный микрофон не сохраняется.

Пользователь также может сказать голосовую команду для выбора микрофона в данном случае, микрофон сразу сохранится в конфигурационный файл, то есть подтверждения нажатием кнопки «Ок» не требуется.

Функционал данного окна прописан в приложении Б (листинг Б.1).

4.3 Окно выбора директории

При нажатии на строку «Выбор директории…» пользователь попадёт в окно, указанное на рисунке 9. Инициализация данного окна показана в приложении А (листинг А.3).

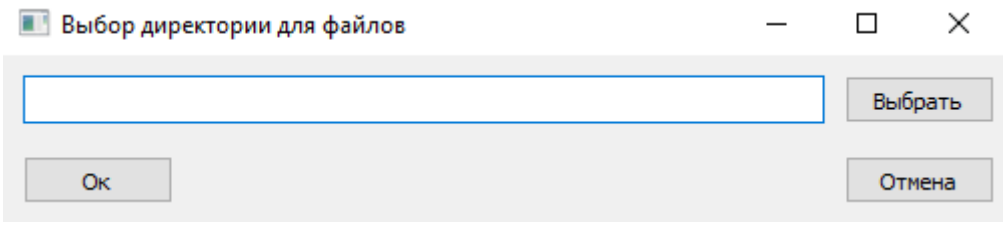

Рисунок 9 – Окно выбора директории

В данном окне пользователь может выбрать директорию, где будут созданы информационные файлы log.txt и result.txt. Файл log.txt будет хранить историю запросов пользователя, результат выполнения команд, а также информацию об непредвиденных ошибках.

Данное окно содержит в себе 3 кнопки: «Ок», «Отмена», «Выбрать», а также текстовое поле для голосового или письменного ввода директории.

При нажатии на кнопку выбрать, пользователю открывается всплывающее окно для ручного выбора директории (рисунок 10).

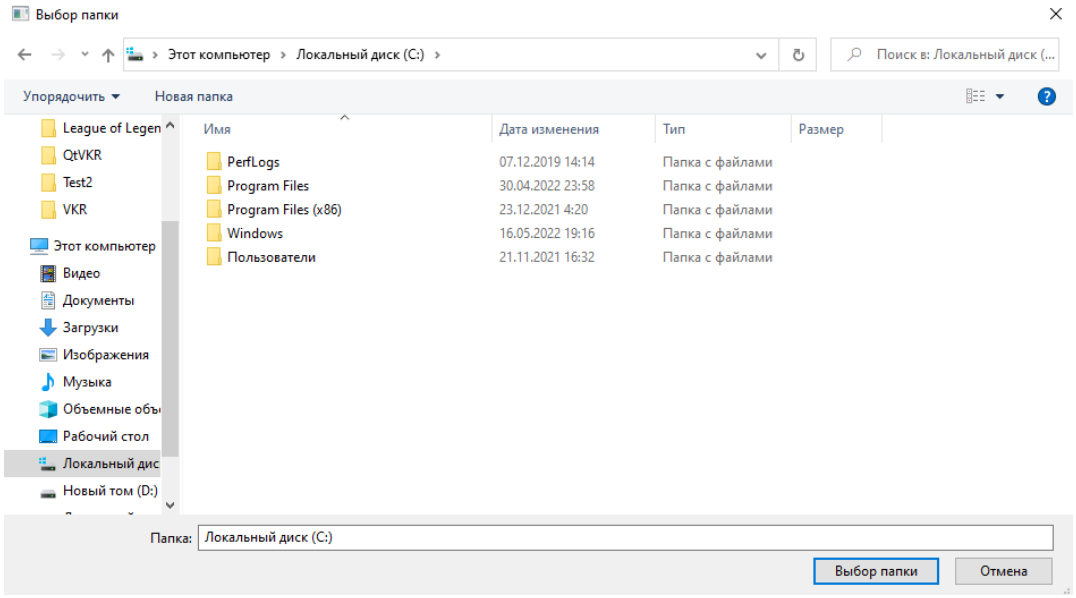

Рисунок 10 – Всплывающее окно ручного выбора директории

После выбора директории в ручном режиме, выбранная директория появляется в текстовом поле (рисунок 11).

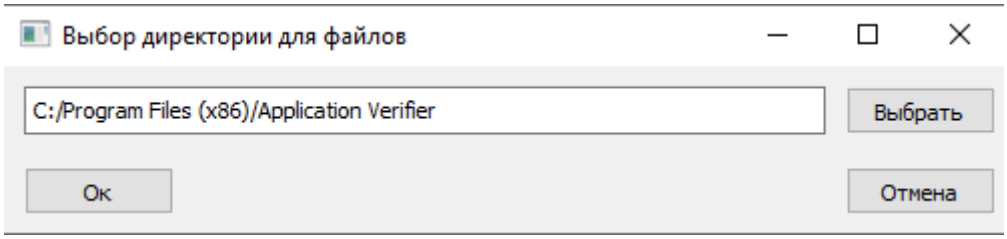

Рисунок 11 – Окно выбора с выбранной директорией

По аналогии с окном выбора микрофона: кнопка «Ок» – сохраняет директорию в конфигурационный файл, а кнопка «Отмена» – закрывает окно выбора без сохранения.

В следствии того, что данные файлы создаются и используются только программой и имеют закодированную структуру, то нет надобности добавлять голосовую команду для выбора данной папки.

Функционал данного окна прописан в приложении Б (листинг Б.2).

5 ТЕСТИРОВАНИЕ

Все системы имели 64 разрядную архитектуру. Для тестирования на ОС Windows 7 и Windows 8.1 была использована Oracle VM Virtual box [12], версия платформенного пакета: 6.1.34.

В качестве тестирования будут приведены данные из диспетчера задач, а также примеры распознавания команд пользователя. На экранных снимках диспетчера задач (рисунки 12-17) будут приведены по 2 рисунка, на которых будет показана фоновая работа программы и её работа в нагрузке.

5.1 Тестирование на Windows 10

Технические характеристики персонального компьютера №1, использованного в тестировании на Windows 10:

– процессор: AMD Ryzen 5 3550H (12 логических процессоров);

– оперативная память: 16384 МБ (виртуальная память 2432 МБ);

– диск: Твердотельный накопитель на 464 ГБ;

– операционная система: Windows 10 (версия 10.0.19044.1620);

– встроенный микрофон Realtek(R) Audio.

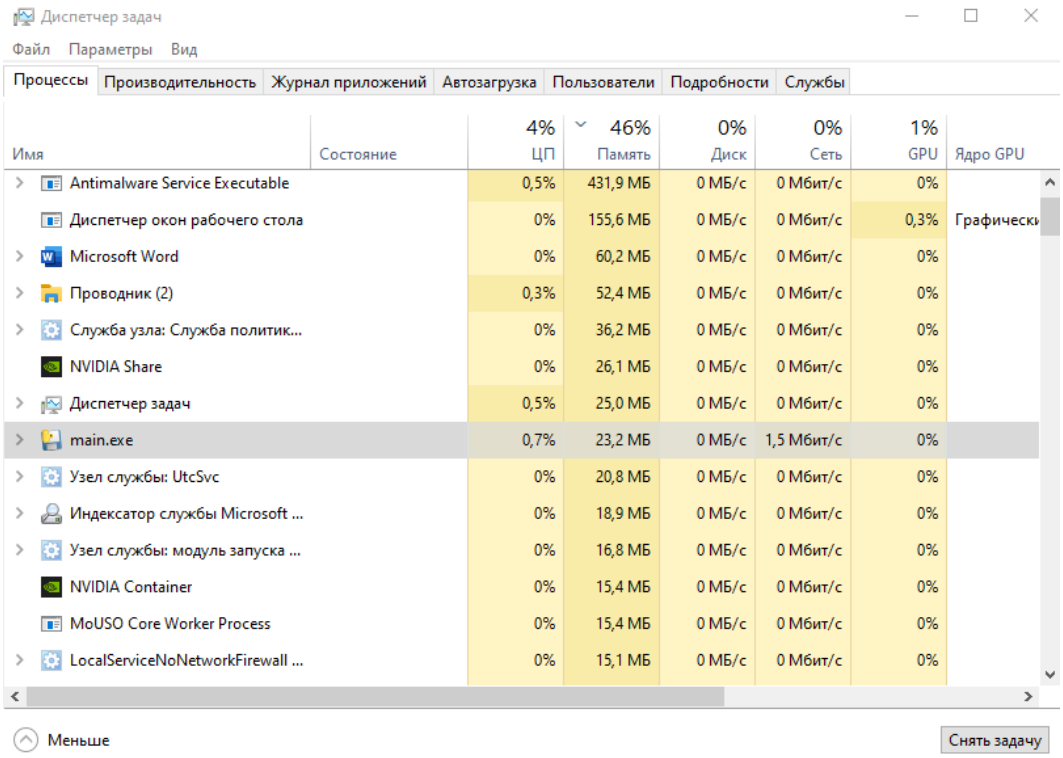

Рисунок 12 – Потребление ресурсов в фоновом режиме

Как видно на рисунке 12, потребление оперативной памяти в фоновом режиме находится около 23.2 МБ, что соответствует заявленным системным требованиям. А нагрузка на процессор составляет 0.7%, что так же соответствует заявленным системным требованиям.

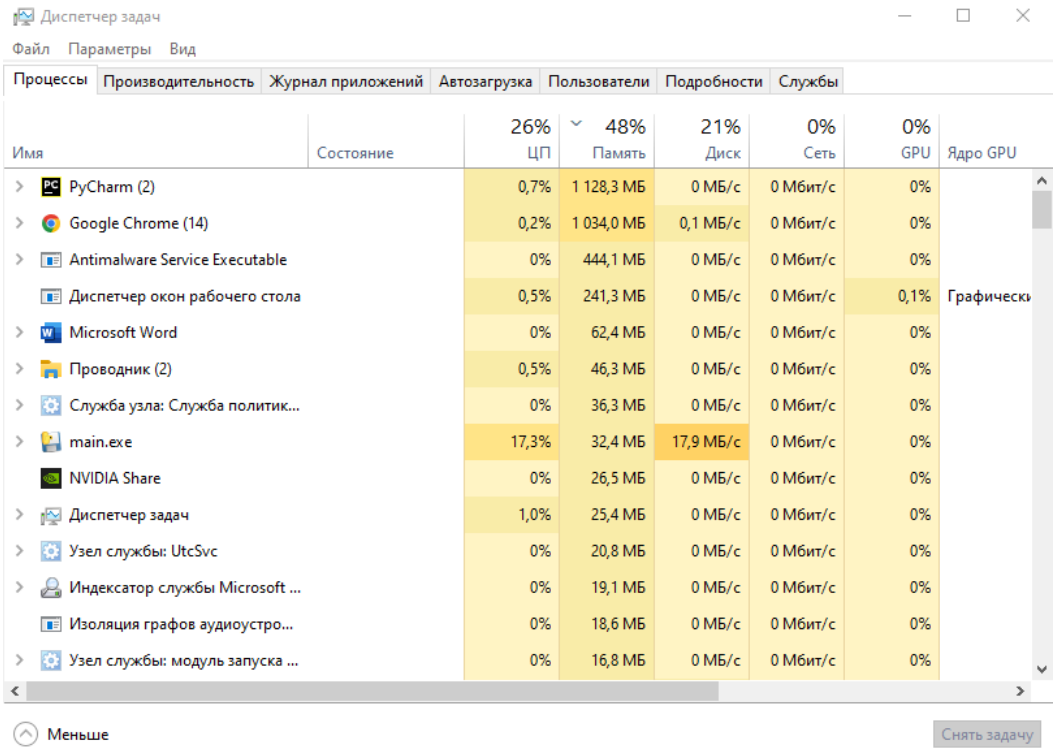

Рисунок 13 – Потребление ресурсов в режиме нагрузки

На рисунке 13 видно, что при выполнении команды пользователя оперативной памяти находится около значения 32.4 МБ, а вот нагрузка на процессор уже составляет около 17.3%, также возросла нагрузка на носитель информации.

Подводя итоги о нагрузке программы на компьютер можно сделать вывод, что потребление ресурсов находится в пределах заявленных системных требований.

5.2 Тестирование на Windows 8.1

Технические характеристики персонального компьютера №2, использованного в тестировании на Windows 8.1:

– процессор: AMD Ryzen 5 3550H (1 логический процессор);

– оперативная память: 2752 МБ (виртуальная память 2432 МБ);

– диск: Жесткие диск на 50 ГБ;

– операционная система: Windows 8.1 (версия 6.3.9600.17031);

– микрофон Realtek(R) Audio.

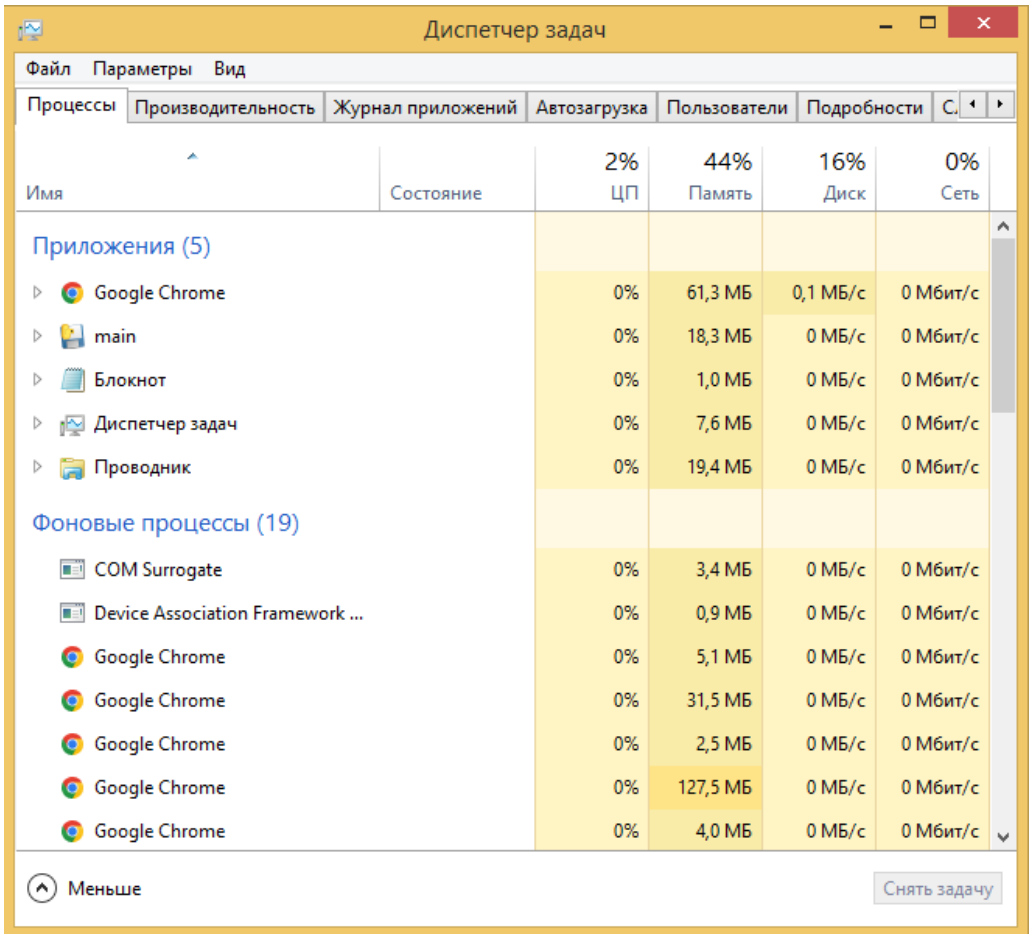

Рисунок 14 – Потребление ресурсов в фоновом режиме

Как видно на рисунке 14, потребление оперативной памяти в фоновом режиме находится около 18.3 МБ, что соответствует заявленным системным требованиям. А нагрузка на процессор составляет 0%, что так же соответствует заявленным системным требованиям.

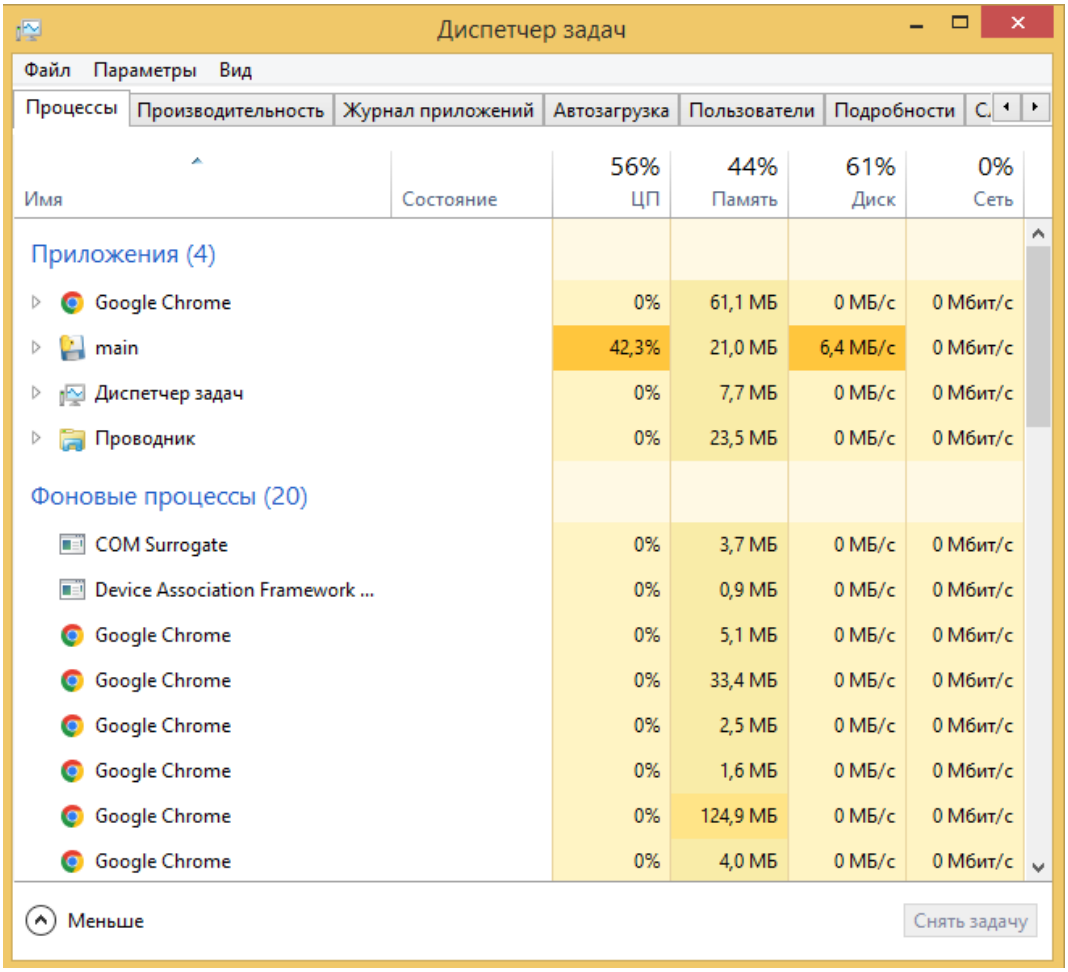

Рисунок 15 – Потребление ресурсов в режиме нагрузки

На рисунке 15 видно, что при выполнении команды пользователя оперативной памяти находится около значения 21 МБ, а вот нагрузка на процессор уже составляет около 42.3%, также возросла нагрузка на носитель информации.

Подводя итоги о нагрузке программы на компьютер можно сделать вывод, что потребление ресурсов находится в пределах заявленных системных требований.

5.3 Тестирование на Windows 7

Технические характеристики персонального компьютера №3, использованного в тестировании на Windows 7:

– процессор: AMD Ryzen 5 3550H (1 логический процессор);

– оперативная память: 2752 МБ (виртуальная память 2432 МБ);

– диск: Жесткие диск на 50 ГБ;

– операционная система: Windows 7 (версия 6.1.7601.25895);

– микрофон Realtek(R) Audio.

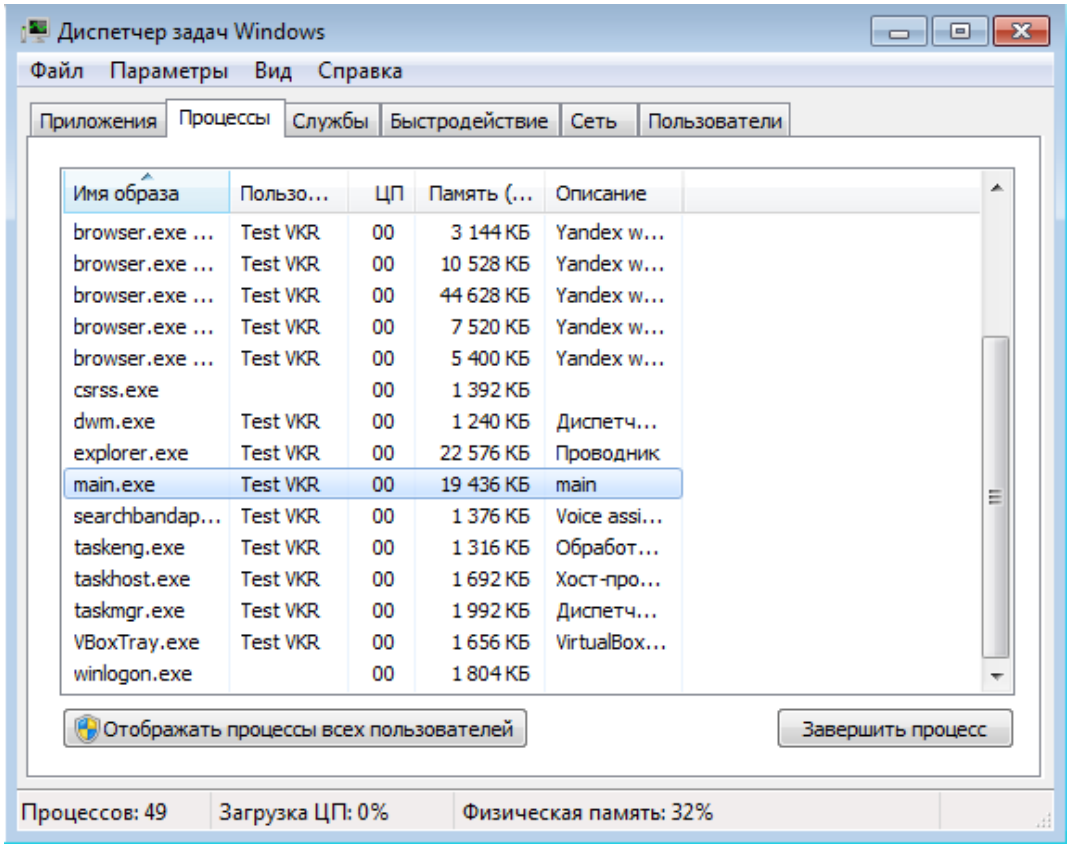

Рисунок 16 – Фоновая нагрузка на Windows 7

Как видно на рисунке 16, потребление оперативной памяти в фоновом режиме находится около 19.4 МБ, что соответствует заявленным системным требованиям. А нагрузка на процессор составляет 0%, что так же соответствует заявленным системным требованиям.

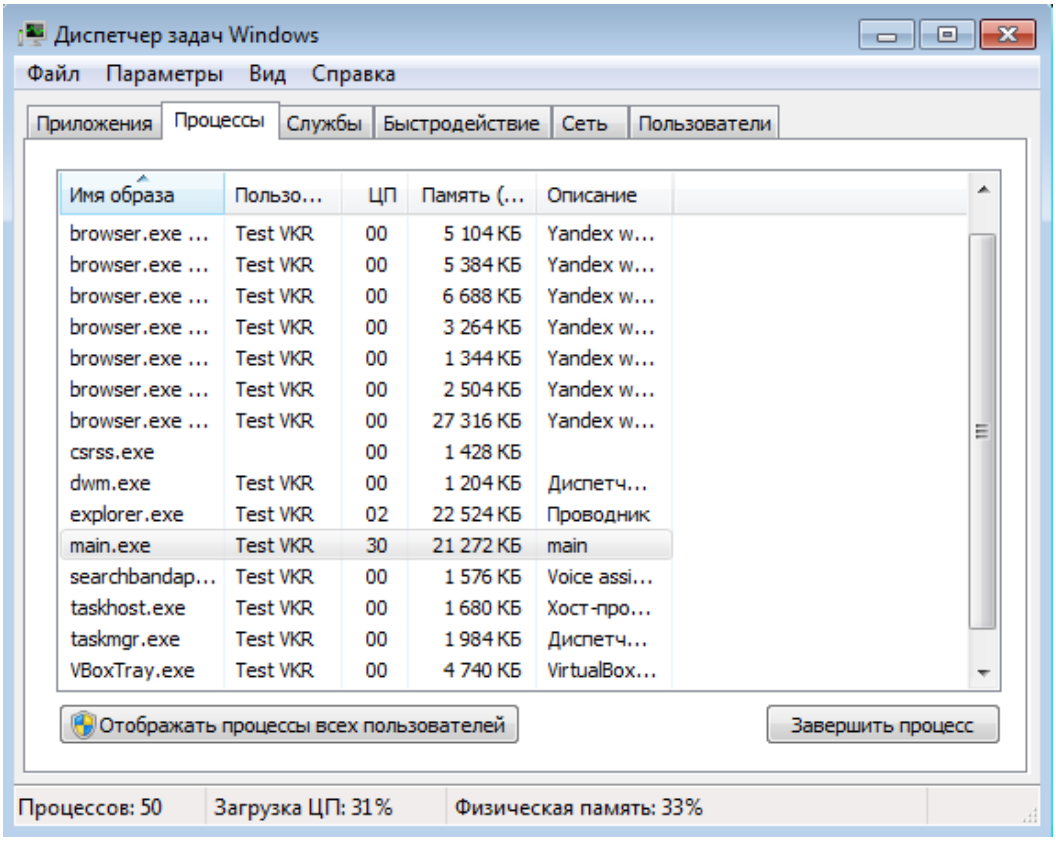

Рисунок 17 – Потребление ресурсов в режиме нагрузки

На рисунке 17 видно, что при выполнении команды пользователя оперативной памяти находится около значения 21.2 МБ, а вот нагрузка на процессор уже составляет около 30%.

Подводя итоги о нагрузке программы на компьютер можно сделать вывод, что потребление ресурсов находится в пределах заявленных системных требований.

5.4 Тестирование распознавания голоса

Для тестирования правильности распознавания речи выберем некоторые команды и произнесём их. Первой командой будет открыть сайт, второй командой будет запуск какого-нибудь приложения на компьютере, третьей командой будет поиск в интернете.

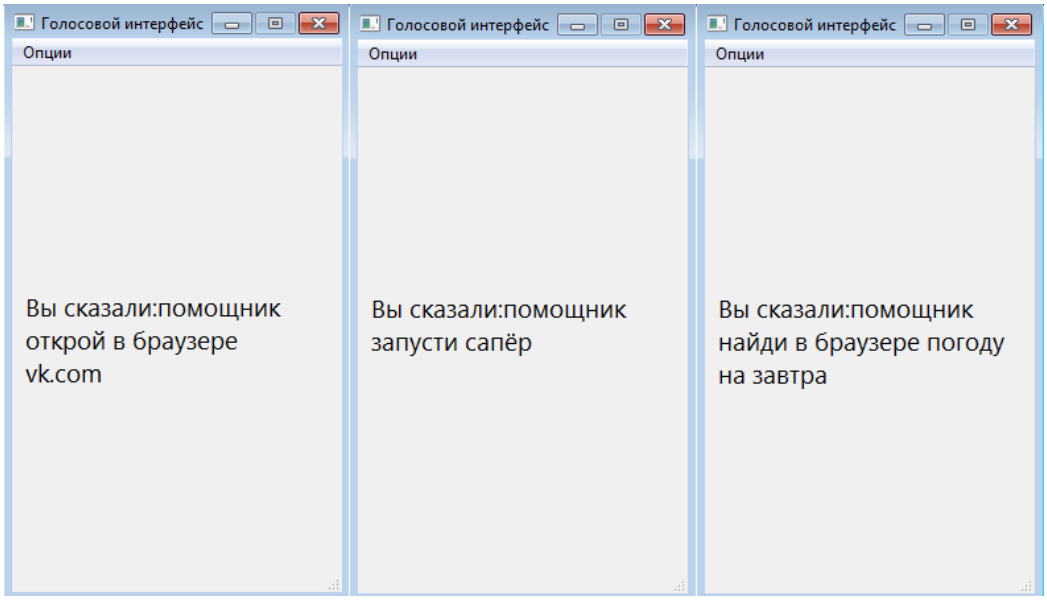

Рисунок 18 – Выполнение команд программой

Как видно из рисунка 18, программа правильно распознала все 3 фразы.

5.5 Тестирование голосового управления

В качестве тестирования голосового управления будет представлен скриншот файла «log.txt» (рисунок 19), который записывает в себя команду, произнесённую пользователем, а также результат её выполнения.

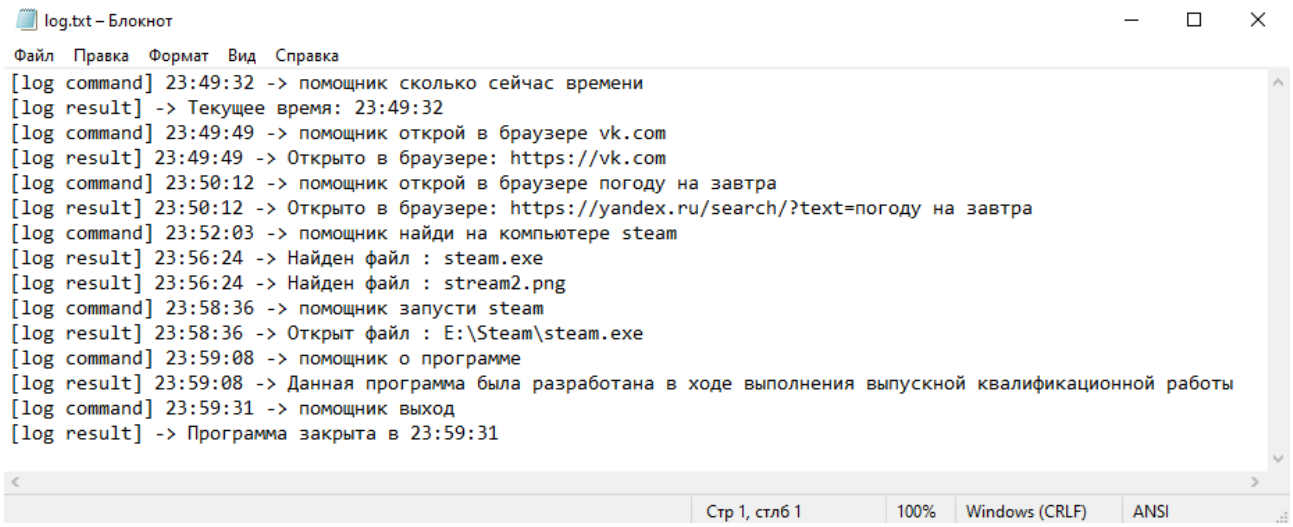

Рисунок 19 – Команда и результат её выполнения

Как видно из рисунка 19, программа откликается на каждую из произнесённых команд.

ЗАКЛЮЧЕНИЕ

В рамках выпускной квалификационной работы был спроектирован и реализован голосовой интерфейс для людей с ограниченными возможностями. Для достижения поставленной цели был проведен анализ существующих аналогов программных решений. Рассмотрены достоинства и недостатки средств и технологии, которые можно применять для разработки голосовых интерфейсов. Для реализации программного продукта был выбран язык программирования Python. В качестве среды разработке был выбран PyCharm. В качестве среды для разработки пользовательского интерфейса был выбран Qt Designer. Определен необходимый функционал голосового интерфейса. Приведены этапы проектирования, разработка и процедура тестирования функционала программы.

Разработанный голосовой интерфейс может быть полезным для людей с ограниченными возможностями, которые желают использовать персональные компьютеры.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Умные помощники: обзор рынка, тренды и перспективы. – Текст : электронный // Лента : [сайт]. – 2021. – URL: https://ict.moscow/news/voice-assistants-2021/ (дата обращения: 25.12.2021).

2. Куприна, Е. Тенденция роста использования голосовых интерфейсов в мире. – Текст : электронный / Е. Куприна // mildberry : [сайт]. – URL: https://www.mildberry.ru/cases/voice-assistants/ (дата обращения: 20.03.2022).

3. Desktop Browser Market Share Russian Federation. – Изображение (неподвижное ; двухмерное) : электронное // StatCounter: [сайт]. – 2022. – URL: https://gs.statcounter.com/browser-market-share/desktop/worldwide (дата обращения: 25.12.2021).

4. Рейтинг языков программирования TIOBE. – Текст : электронный // tiobe : [сайт]. – URL: https://www.tiobe.com/tiobe-index/ (дата обращения: 16.05.2022).

5. Рейтинг языков программирования PyPl. – Текст : электронный // pypl.github : [сайт]. – URL: https://pypl.github.io/PYPL.html (дата обращения: 16.05.2022).

6. Статическая и динамическая типизация данных. – Текст. Изображение (неподвижное ; двухмерное) : электронные // habr : [сайт]. – 3 декабря 2012. – URL: https://habr.com/ru/post/161205/ (дата обращения: 16.05.2022).

7. Версии Python / idlesign. – Текст : электронный // pythonz.net = versions : [сайт]. – URL: https://pythonz.net/versions/ (дата обращения: 16.05.2022).

8. Riverbank Computing : [официальный сайт] / Riverbank Computing Limited, The Qt Company. – США, 2021. – URL: https://www.riverbankcomputing.com/static/Docs/PyQt5/ (дата обращения: 25.12.2021). – Текст : электронный.

9. SpeechRecognition Documentation / Uberi. – Текст : электронный // github.com = speech_recognition : [сайт]. – URL: https://github.com/Uberi/speech_recognition#readme (дата обращения: 25.12.2021).

10. PocketSphinx Documentation / lenzo-duo. – Текст : электронный // github.com = pocketsphinx : [сайт]. – URL: https://github.com/cmusphinx/pocketsphinx (дата обращения: 16.05.2022).

11. Расстояние Левенштейна. – Текст. Изображение (неподвижное ; двухмерное) : электронные // habr : [сайт]. – 7 апреля 2011. – URL: https://habr.com/ru/post/117063/ (дата обращения: 16.05.2022).

12. VirtualBox : [официальный сайт] / Oracle. – США, 2022. – URL: https://www.virtualbox.org/ (дата обращения: 16.05.2022). – Текст : электронный.

ПРИЛОЖЕНИЕ А

Листинг А.1 – Инициализация главного окна программы

```
from PyQt5 import QtCore, QtGui, QtWidgets
class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(250, 420)
        MainWindow.setMinimumSize(QtCore.QSize(250, 420))
        MainWindow.setMaximumSize(QtCore.QSize(250, 420))
        MainWindow.setFocusPolicy(QtCore.Qt.NoFocus)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.message_from_user =
QtWidgets.QLabel(self.centralwidget)
        self.message_from_user.setGeometry(QtCore.QRect(10, 170,
221, 201))
        self.message_from_user.setAlignment(QtCore.Qt.AlignLeading
| QtCore.Qt.AlignLeft | QtCore.Qt.AlignTop)
        self.message_from_user.setObjectName("message_from_user")
        self.message_from_user.setWordWrap(True)
        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(MainWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 250, 22))
        self.menubar.setObjectName("menubar")
        self.options = QtWidgets.QMenu(self.menubar)
        self.options.setObjectName("options")
        #self.settings = QtWidgets.QMenu(self.menubar)
        #self.settings.setObjectName("settings")
        self.microphone_settings = QtWidgets.QMenu(self.options)

self.microphone_settings.setObjectName("microphone_settings")
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)
        self.notifications = QtWidgets.QAction(MainWindow)
```

```
        self.notifications.setObjectName("notifications")
        self.chose_microphone = QtWidgets.QAction(MainWindow)
        self.chose_microphone.setObjectName("chose_microphone")
        self.sens_microphone = QtWidgets.QAction(MainWindow)
        self.sens_microphone.setObjectName("sens_microphone")
        self.volume_microphone = QtWidgets.QAction(MainWindow)
        self.volume_microphone.setObjectName("volume_microphone")
        self.clear_files = QtWidgets.QAction(MainWindow)
        self.clear_files.setObjectName("clear_files")
        self.view_files = QtWidgets.QAction(MainWindow)
        self.view_files.setObjectName("view_files")
        self.directory = QtWidgets.QAction(MainWindow)
        self.directory.setObjectName("directory")
        self.quick_access = QtWidgets.QAction(MainWindow)
        self.quick_access.setObjectName("quick_access")
        #self.red_files = QtWidgets.QAction(MainWindow)
        #self.red_files.setObjectName("red_files")
        self.about_programm = QtWidgets.QAction(MainWindow)
        self.about_programm.setObjectName("about_programm")
        self.Exit = QtWidgets.QAction(MainWindow)
        self.Exit.setObjectName("Exit")
        self.options.addAction(self.clear_files)
        self.options.addAction(self.view_files)
        #self.options.addAction(self.red_files)
        self.options.addAction(self.notifications)

self.options.addAction(self.microphone_settings.menuAction())
        self.options.addAction(self.directory)
        self.options.addSeparator()
        self.options.addAction(self.about_programm)
        self.options.addAction(self.Exit)
        self.microphone_settings.addAction(self.chose_microphone)
        self.microphone_settings.addAction(self.sens_microphone)
        self.microphone_settings.addAction(self.volume_microphone)
        self.menubar.addAction(self.options.menuAction())
        #self.menubar.addAction(self.settings.menuAction()
```

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

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow",
"Голосовой интерфейс", "Voice assistant"))
        self.message_from_user.setText(_translate("MainWindow", "Вы
сказали: "))
        self.options.setTitle(_translate("MainWindow", "Опции"))
        #self.settings.setTitle(_translate("MainWindow",
"Настройки"))
        self.microphone_settings.setTitle(_translate("MainWindow",
"Микрофон"))
        self.notifications.setText(_translate("MainWindow",
"Уведомления..."))
        self.chose_microphone.setText(_translate("MainWindow",
"Выбор устройства..."))
        self.sens_microphone.setText(_translate("MainWindow",
"Чувствительность микрофона..."))
        self.volume_microphone.setText(_translate("MainWindow",
"Громкость микрофона..."))
        self.clear_files.setText(_translate("MainWindow",
"Отчистить список файлов"))
        self.view_files.setText(_translate("MainWindow",
"Посмотреть список файлов..."))
        self.directory.setText(_translate("MainWindow",
"Директория..."))
        self.quick_access.setText(_translate("MainWindow", "Быстрый
доступ"))
        #self.red_files.setText(_translate("MainWindow",
"Редактировать список файло..."))
        self.about_programm.setText(_translate("MainWindow", "О
программе..."))
        self.Exit.setText(_translate("MainWindow", "Выход"))
```

```
class MainWindow(QtWidgets.QMainWindow, Ui_MainWindow):
    def __init__(self):
        super().__init__()
        self.setupUi(self)
```

Листинг А.2 – Инициализация окна выбора микрофона

```
from PyQt5 import QtCore, QtGui, QtWidgets

class Ui_Choise_mic(object):
    def setupUi(self, Choise_mic):
        Choise_mic.setObjectName("Choise_mic")
        Choise_mic.resize(500, 70)
        sizePolicy =
QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Fixed,
QtWidgets.QSizePolicy.Fixed)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(Choise_mic.sizePolicy().hasHeightForWi
dth())
        Choise_mic.setSizePolicy(sizePolicy)
        Choise_mic.setMinimumSize(QtCore.QSize(500, 70))
        Choise_mic.setMaximumSize(QtCore.QSize(500, 70))
        self.comboBox = QtWidgets.QComboBox(Choise_mic)
        self.comboBox.setGeometry(QtCore.QRect(10, 10, 481, 22))
        self.comboBox.setObjectName("comboBox")
        self.ok = QtWidgets.QPushButton(Choise_mic)
        self.ok.setGeometry(QtCore.QRect(10, 40, 75, 24))
        self.ok.setObjectName("ok")
        self.cancel = QtWidgets.QPushButton(Choise_mic)
        self.cancel.setGeometry(QtCore.QRect(420, 40, 75, 24))
        self.cancel.setObjectName("cancel")

        self.retranslateUi(Choise_mic)
        QtCore.QMetaObject.connectSlotsByName(Choise_mic)
```

```
def retranslateUi(self, Choise_mic):
        _translate = QtCore.QCoreApplication.translate
        Choise_mic.setWindowTitle(_translate("Choise_mic", "Выбор
микрофона"))
        self.ok.setText(_translate("Choise_mic", "Ок"))
        self.cancel.setText(_translate("Choise_mic", "Отмена"))
```

Листинг А.3 – Инициализация окна выбора директории

```
from PyQt5 import QtCore, QtGui, QtWidgets

class Ui_Chose_dir(object):
    def setupUi(self, Chose_dir):
        Chose_dir.setObjectName("Chose_dir")
        Chose_dir.resize(500, 77)
        self.textEdit = QtWidgets.QTextEdit(Chose_dir)
        self.textEdit.setGeometry(QtCore.QRect(10, 10, 400, 24))
        self.textEdit.setObjectName("textEdit")
        self.Chose = QtWidgets.QPushButton(Chose_dir)
        self.Chose.setGeometry(QtCore.QRect(420, 10, 75, 24))
        self.Chose.setObjectName("chose")
        self.Ok = QtWidgets.QPushButton(Chose_dir)
        self.Ok.setGeometry(QtCore.QRect(10, 50, 75, 24))
        self.Ok.setObjectName("ok")
        self.Cancel = QtWidgets.QPushButton(Chose_dir)
        self.Cancel.setGeometry(QtCore.QRect(420, 50, 75, 24))
        self.Cancel.setObjectName("cancel")

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

    def retranslateUi(self, Chose_dir):
        _translate = QtCore.QCoreApplication.translate
        Chose_dir.setWindowTitle(_translate("Chose_dir", "Выбор
директории для файлов"))
        self.Chose.setText(_translate("Chose_dir", "Выбрать"))
        self.Ok.setText(_translate("Chose_dir", "Ок"))
        self.Cancel.setText(_translate("Chose_dir", "Отмена"))
```

ПРИЛОЖЕНИЕ Б

Листинг Б.1 – Выбор микрофона

```
from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.QtWidgets import QFileDialog, QMessageBox
from choice import Ui_Choise_mic

import speech_recognition as SR
import json

class NewWindowChoiceMic(QtWidgets.QWidget, Ui_Choise_mic):
    chosen_mic = ""

    def __init__(self, parent):
        super(NewWindowChoiceMic, self).__init__()
        self.setupUi(self)
        self.parent = parent
        self.add_funk()
        self.add_funk_btn()

    def choice_micro(self, signal):  # !!!
        if signal:
            self.show()
        else:
            self.hide()

    def add_funk(self):
        list_mic = SR.Microphone().list_microphone_names()
        self.comboBox.addItems(list_mic)
        self.comboBox.currentTextChanged.connect(self.text_changed)

    def text_changed(self, text):
        list_mic = SR.Microphone().list_microphone_names()
        self.chosen_mic =
list_mic.index(self.comboBox.currentText())

    def add_funk_btn(self):
```

```
        self.ok.clicked.connect(self.Ok)
        self.cancel.clicked.connect(self.Cansel)

    def Cansel(self):  # !!!
        self.choice_micro(False)

    def Ok(self):
        global config
        config.update(chosen_microphone=self.chosen_mic)
        self.ToJSON()
        self.choice_micro(False)

    def ToJSON(self):
        global config
        with open('config.txt', 'w') as outfile:
            json.dump(
                config,
                outfile,
                indent=4,
                ensure_ascii=False
            )
        outfile.close()
```

Листинг Б.2 – Выбор директории

```
from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.QtWidgets import QFileDialog, QMessageBox
from set_directory import Ui_Chose_dir

import speech_recognition as SR
import json

class NewWindowChoiceDir(QtWidgets.QWidget, Ui_Chose_dir):
    chosen_dir = ""

    def __init__(self, parent):
        super(NewWindowChoiceDir, self).__init__()
        self.setupUi(self)
```

```
    self.parent = parent
    self.add_funk_btn()

def choice_path(self, signal):
    if signal:
        self.show()
    else:
        self.hide()

def add_funk_btn(self):
    self.Ok.clicked.connect(self.ok)
    self.Cancel.clicked.connect(self.cancel)
    self.Chose.clicked.connect(self.chose)

def ok(self):
    global config
    config.update(chosen_directory=self.chosen_dir)
    self.ToJSON()
    self.choice_path(False)

     def cancel(self):
         self.choice_path(False)

     def chose(self):
         dirlist = QFileDialog.getExistingDirectory(self)
         self.chosen_dir = dirlist
         self.textEdit.setText(self.chosen_dir)

     def ToJSON(self):
         global config
         with open('config.txt', 'w') as outfile:
             json.dump(
                 config,
                 outfile,
                 indent=4,
                 ensure_ascii=False
```

```
        )
    outfile.close()
```

Листинг Б.3 – Обработка исключений при неправильно выбранных настройках

```
cfg = {
    "chosen_microphone": 1,
    "chosen_directory": 'D:\\'
}
try:
    with open('config.txt', 'r') as cfg:
        config = json.load(cfg)
        cfg.close()
except FileNotFoundError:
    with open('config.txt', 'w') as cfgFile:
        json.dump(
            cfg,
            cfgFile,
            indent=4,
            ensure_ascii=False
        )
        cfgFile.close()
except Exception:
    error = QMessageBox()
    error.setWindowTitle("Неизвестная ошибка")
    error.setText("Неизвестная ошибка, пожалуйста, переустановите
программу\n" + Exception)
```

ПРИЛОЖЕНИЕ В

Листинг В.1 – Поиск файлов в операционной системе

```
elif cmd == 'search_in_computer':
    for_open = self.command
    for x in for_open:
        if "." in x:
            for_open = for_open.split('.')
    for z in command_for_open_or_search_computer['tbr']:
        for_open = for_open.replace(z, "").strip()
    try:
        drives = win32api.GetLogicalDriveStrings()
        drives = drives.split('\000')[:-1]
        with open('result.txt', 'a') as result:
            for i in drives:
                for rootdir, dirs, files in os.walk(i):
                    for file in files:
                        print(rootdir, file)
                        if (file.split('.')[0]) == for_open:
                            print(os.path.join(rootdir, file))
                            result.write(os.path.join(rootdir,
file) + "\n")
            result.close()
    except Exception:
        now = datetime.datetime.now()
        with open('log.txt', 'a') as log:
            log.write("[log error] " + str(now.hour) + ":" +
str(now.minute) + ":" + str(now.second) +
                      "-> Файл имеет неизвестную кодировку" +
rootdir + file + "->" + Exception + "\n")
            log.close()
```

Листинг В.2 – Открытие файла в операционной системе

```
command_for_open_or_search_computer = {
    "tbr": ('на компьютере', 'файл', 'папку', 'документ', 'компе',
'найди', 'на', 'открой', 'запусти', 'запустить')
}
```

```
files_extension = {
    "type": ('exe', 'pdf', 'docx', 'doc' 'mp3', 'txt', 'mp4',
'avi', 'ppt', 'pptx', 'mpeg3', 'mpeg4',
             'png', 'gif', 'jpeg', 'zip', 'rar', '7zip')
}
elif cmd == 'open_in_computer':
    for_open = self.command
     #удаляются слова, мешающие поиску файла
    for z in command_for_open_or_search_computer['tbr']:
        for_open = for_open.replace(z, "").strip()
    try:
        with open('result.txt', 'r') as fast_open:
            result_file = fast_open.readlines()
            fast_open.close()
        ver_file = {
            'file': '',
            'percent': 0
        }
        # поиск файла, название которого наиболее соответствует
слову, которое сказал пользователь
        for p in result_file:
            for v in files_extension.items():
                for d in v:
                    for y in d:
                        vrt = fuzz.ratio(str(for_open) + str(y),
os.path.basename(p))
                        if vrt > ver_file['percent']:
                            ver_file['file'] = p
                            ver_file['percent'] = vrt
         # если процент совпадений меньше 30% осуществляется поиск
файлов на компьютере
        if ver_file['percent'] < 30:
            try:
                drives = win32api.GetLogicalDriveStrings()
                drives = drives.split('\000')[:-1]
```

```
                with open('result.txt', 'a') as result:
                    for i in drives:
                        for rootdir, dirs, files in os.walk(i):
                            for file in files:
                                print(rootdir, file)
                                if (file.split('.')[0]) ==
for_open:
                                    print(os.path.join(rootdir,
file))

result.write(os.path.join(rootdir, file) + "\n")
                    result.close()
            except Exception:
                now = datetime.datetime.now()
                with open('log.txt', 'a') as log:
                    log.write("[log error] " +
                              str(now.hour) + ":" + str(now.minute)
+ ":" + str(now.second) +
                              "-> Файл имеет неизвестную кодировку"
+ rootdir + file + "->" +
                              str(Exception) + "\n")
                    log.close()
            try:
                file_for_open = ver_file['file'].replace(r"\"",
r"\\"")
                print(file_for_open)
                subprocess.call(('start', "", ver_file['file']),
shell=True)
            except Exception:
                now = datetime.datetime.now()
                with open('log.txt', 'a') as log:
                    log.write("[log error] " + str(now.hour) + ":"
+ str(now.minute) + ":" + str(now.second) +
                              "  ->  Неизвестная ошибка: " +
str(Exception) + "\n")
                    log.close()
```

```
        else:
            file_for_open = ver_file['file'].replace(r"\"",
r"\\"")
            print(file_for_open)
            subprocess.call(('start', "", ver_file['file']),
shell=True)
    except Exception:

        now = datetime.datetime.now()
        with open('log.txt', 'a') as log:
            log.write("[log error] " + str(now.hour) + ":" +
str(now.minute) + ":" + str(now.second) +
                          "  ->  Неизвестная ошибка: " +
str(Exception) + "\n")
            log.close()
```

Листинг В.3 – Включение и выключение уведомлений

```
elif cmd == 'on_notification':
    self.notifications = True
elif cmd == 'off_notification':
    self.notifications = False
```

Листинг В.4 – Информация о программе

```
elif cmd == 'about program':
    try:
        about = QMessageBox()
        about.setWindowTitle("О программе")
        about.setText("Данная программа была разработана в ходе
выполнения выпускной квалификационной работы"
                      "студентом 4 курса южно-уральского
государственного университета"
                      "высшей школы электроники и компьютерных наук"
                      "кафедры Электронные вычислительные машины.\n"
                      "Версия программы: 0.1 alpha")
        about.exec_()
    except Exception:
```

```
        now = datetime.datetime.now()
        with open('log.txt', 'a') as log:
            log.write("[log error] " + str(now.hour) + ":" +
str(now.minute) + ":" + str(now.second) +

                      " ->  Неизвестная ошибка" + str(Exception) +
"\n")
            log.close()
```

Листинг В.5 – Выход из программы

```
elif cmd == 'exit':
    MainWindow.close()
```

Листинг В.6 – Текущее время

```
if cmd == 'time':
    now = datetime.datetime.now()
    self.message_from_user.setText(self.message_from_user.text() +
"\nОтвет: " + "Сейчас " + str(now.hour) + ":" + str(now.minute) +
":" + str(now.second))
    with open('log.txt', 'a') as log:
        log.write("[log result] -> Текущее время: " + str(now.hour)
                  + ":" + str(now.minute) + ":" + str(now.second) +
"\n")
        log.close()
```

Листинг В.7 – Поиск в интернете

```
command_for_open_internet = {
    "tbr": ('в браузере', 'в интернете', 'открой на компьютере',
            'открой', 'запусти', 'найди на компьютере', 'найди')
}
elif cmd == 'open_in_browser':
    for_open = self.command
    for z in command_for_open_internet['tbr']:
        for_open = for_open.replace(z, "").strip()
    if re.search(r'\.', for_open):
        webbrowser.open_new_tab('https://' + for_open)
        with open('log.txt', 'a') as log:
            log.write("[log result] -> Открыто в браузере: " +
'https://' + for_open + "\n")
```

```
            log.close()
    elif re.search(r'\ ', for_open):

webbrowser.open_new_tab('https://yandex.ru/search/?text='.format(fo
r_open))

        with open('log.txt', 'a') as log:
            log.write("[log result] -> Открыто в браузере: "
                      + 'https://yandex.ru/search/?text=' +
for_open + "\n")
            log.close()
    else:
webbrowser.open_new_tab('https://yandex.ru/search/?text='.format(fo
r_open))
        with open('log.txt', 'a') as log:
            log.write("[log result] -> Открыто в браузере: " +
                      'https://yandex.ru/search/?text=' + for_open
+ "\n")
            log.close()
```

Листинг В.8 – Изменение микрофона

```
elif cmd == "microphone settings":
    for_open = self.command

    with open('config.txt', 'r') as cfg:
        config = json.load(cfg)
        cfg.close()

    for z in settings_microphone['tbr']:
        for_open = for_open.replace(z, "").strip()

    config.update(chosen_microphone=for_open)
```

Листинг В.9 – Изменение микрофона голосом

```
for_open = self.command
with open('config.txt', 'r') as cfg:
```

```
    config = json.load(cfg)
    cfg.close()

for z in settings_microphone['tbr']:
    for_open = for_open.replace(z, "").strip()

config.update(chosen_microphone=int(for_open))
with open('config.txt', 'w') as outfile:
    json.dump(
        config,
        outfile,
        indent=4,
        ensure_ascii=False
    )
outfile.close()
```

ПРИЛОЖЕНИЕ Г

Листинг Г.1 – Подключение навигационных кнопок и глобальных переменных

```
from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.QtWidgets import QMessageBox
from PyQt5.QtCore import QTimer
from fuzzywuzzy import fuzz
from signals import NewWindowChoiceMic
from signals import NewWindowChoiceDir

import speech_recognition as SR
import webbrowser
import subprocess
import datetime
import win32api
import psutil
import json
import re
import os
import time

class MainWindow(QtWidgets.QMainWindow, Ui_MainWindow):
    def __init__(self):
        super().__init__()
        self.setupUi(self)
        self.newWindowChoiceMic = NewWindowChoiceMic(self)
        self.newWindowChoiceDir = NewWindowChoiceDir(self)

        # загрузка конфигурационного файла
        self.load_config()

        cho_micro = 0
        direct = "C:\\"
        command = ""
        notification = True

        isActive = True
```

```
        self.voiceRec(isActive)

        # функционал всех нажимных частей интерфейса
        self.notifications.triggered.connect(
            lambda: self.add_notifications
        )
        self.directory.triggered.connect(
            lambda: self.newWindowChoiceDir.choice_path(True)
        )
        self.chose_microphone.triggered.connect(
            lambda: self.newWindowChoiceMic.choice_micro(True)
        )
        self.Exit.triggered.connect(
            lambda: self.close_programm()
        )
        self.about_programm.triggered.connect(
            lambda: self.about_programm()
        )
def load_config(self):
    with open('config.txt', 'r') as cfgFile:
        temp = json.load(cfgFile)
        self.cho_micro = temp['chosen_microphone']
        self.direct = temp['chosen_directory']
```

ПРИЛОЖЕНИЕ Д

Листинг Д.1 – Список команд для управления интерфейсом

```
command_dict = {
    "commands": {
        "time": ('сейчас времени', 'текущее время', 'который час'),
        "open_in_browser": ('в браузере', 'в интернете'),
        "open_in_computer": ('открой', 'запусти', 'запустить'),
        "search_in_computer": ('найди', 'отыщи', 'поищи', 'найди'),
        "on_notification": ('выключи уведимления', 'отключи
уведомления'),
        "off_notification": 'включи уведомления',
        "settings": ('опции'),
        "about program": ('о программе', 'программа'),
        "exit": ('выключись', 'закройся', 'выход'),
        "hello": ('привет', 'здравствуй', 'добрый день', 'добрый
вечер', 'доброе утро', 'приветствую')
    },
    "tbr": ('сколько', 'узнай', 'узнать', 'компьютере'),  # tbr -
to be removed - должно быть удалено
    "uname": ('компьютер', 'помощник', 'устройство'),
    "where_need_search": ('гугл', 'яндекс')
}
```

Листинг Д.2 – Определение команды

```
def recognize_cmd(self, cmd):
    # примерный поиск
    RecCom = {
        'cmd': '',
        'percent': 0
    }
    for c, v in command_dict['commands'].items():
        for x in v:
            vrt = fuzz.ratio(cmd, x)
            if vrt > RecCom['percent']:
                RecCom['cmd'] = c
                RecCom['percent'] = vrt

    return RecCom
```

Листинг Д.3 – Прослушивание тишины в ожидании команды пользователя

```
def voiceRec(self, isActive):
    # проверка на активацию микрофона
    if isActive:
        rec = SR.Recognizer()
        mic = SR.Microphone(
            device_index=self.cho_micro
        )
        with mic as source:
            rec.adjust_for_ambient_noise(
                source=mic,
                duration=0.2
            )
        stop_listening = rec.listen_in_background(mic,
self.callback)
```

Листинг Д.4 – Удаление частей, помешающих распознаванию команды

```
def callback(self, recognizer, audio):
    try:
        voice = recognizer.recognize_google(
            audio,
            language="ru-RU"
        ).lower()
        self.message_from_user.setText('Вы сказали:' + voice)
        timenow = datetime.datetime.now()
        with open('log.txt', 'a') as log:
            log.write("[log command] " + str(timenow.hour) + ":" +
str(timenow.minute) + ":" + str(timenow.second) + " -> " + voice +
"\n")
            log.close()

        # проверка на обращение к ассистенту
        if voice.startswith(command_dict["uname"]):
            cmd = voice
            for y in command_dict['tbr']:
                cmd = cmd.replace(y, "").strip()
```

```
            for x in command_dict['uname']:
                cmd = cmd.replace(x, "").strip()

            # распознаём и выполняем команду
            self.command = cmd
            cmd = self.recognize_cmd(cmd)
            self.execute_cmd(cmd['cmd'])
    except SR.UnknownValueError:
        print("Голос не распознан!")
    except SR.RequestError as e:
        print("Could not request results from Google Speech
Recognition service; {0}".format(e))
```

Листинг Д.5 – Удаление файлов

```
def delete_files(self):

os.remove(os.path.join(os.path.abspath(os.path.dirname(__file__)),
'result.txt'))

os.remove(os.path.join(os.path.abspath(os.path.dirname(__file__)),
'log.txt'))
```

Листинг Д.6 – Включение или отключение уведомлений

```
def add_notifications(self):
    if self.notifications:
        self.notifications = False
    else:
        self.notifications = True
```

Листинг Д.7 – О программе

```
def about_programm(self):
    try:
        about = QMessageBox()
        about.setWindowTitle("О программе")
        about.setText("Данная программа была разработана в ходе
```

```
выполнения выпускной квалификационной работы"
                            "студентом 4 курса южно-уральского
государственного университета"
                            "высшей школы электроники и компьютерных
наук "
                            "кафедры Электронные вычислительные
машины.\n"
                            "Версия программы: 0.6 beta")
        about.setStandardButtons(QMessageBox.Ok)
        about.exec_()

    except Exception:
        now = datetime.datetime.now()
        with open('log.txt', 'a') as log:
            log.write("[log error] " + str(now.hour) + ":" +
str(now.minute) + ":" + str(now.second) +
                      " ->  Неизвестная ошибка" + str(Exception) +
"\n")
            log.close()
```

Листинг Д.8 – Выход их программы

```
def close_programm(self):
    app.quit()
```