# 

УТВЕРЖДЕН

RU.17701729.07.04-01 51 01-1-ЛУ

|  |  |
| --- | --- |
| ***Подп. и дата*** |  |
|  |
| ***Инв. № дубл.*** |  |
|  |
| ***Взам. инв. №*** |  |
|  |
| ***Подп. и дата*** |  |
|  |
| ***Инв. №***    ***подл*** |  |
|  |

**РАЗРАБОТКА MVP СЕРВИСА СИНТЕЗА РЕЧИ НА АНГЛИЙСКОМ ЯЗЫКЕ**

**Программа и методика испытаний**

**RU.17701729.** **07.04-01 51 01-1**

**Листов 15**

**Москва 2022**

ОГЛАВЛЕНИЕ

[АННОТАЦИЯ 3](#_Toc103864152)

[1. ОБЪЕКТ ИСПЫТАНИЙ 4](#_Toc103864153)

[1.1 Наименование программы 4](#_Toc103864154)

[1.2 Область применения 4](#_Toc103864155)

[2. ЦЕЛЬ ИСПЫТАНИЙ 5](#_Toc103864156)

[3. ТРЕБОВАНИЯ К ПРОГРАММЕ 6](#_Toc103864157)

[3.1 Требования к составу выполняемых функций 6](#_Toc103864158)

[3.1.1 Требования к Acoustic Service 6](#_Toc103864159)

[3.1.2 Требования к Telegram bot service 6](#_Toc103864160)

[3.1.3 Требование к организации взаимодействия сервисов 7](#_Toc103864161)

[3.1.4 Требование к взаимодействию клиентской и серверной частей 7](#_Toc103864162)

[3.1.5 Требования к клиентской части 7](#_Toc103864163)

[3.2 Требования к надежности 8](#_Toc103864164)

[3.2.1 Требования к обеспечению надежного (устойчивого) функционирования программы 8](#_Toc103864165)

[3.2.2 Время восстановления после отказа 8](#_Toc103864166)

[3.2.3 Отказы из-за некорректных действий оператора 8](#_Toc103864167)

[4. ТРЕБОВАНИЯ К ПРОГРАММНОЙ ДОКУМЕНТАЦИИ 9](#_Toc103864168)

[4.1 Состав программной документации 9](#_Toc103864169)

[4.2 Специальные требования к программной документации 9](#_Toc103864170)

[5. СРЕДСТВА И ПОРЯДОК ИСПЫТАНИЙ 10](#_Toc103864171)

[5.1 Технические средства, используемые во время испытаний 10](#_Toc103864172)

[5.2 Программные средства, используемые во время испытаний 10](#_Toc103864173)

[6. МЕТОДЫ ИСПЫТАНИЙ 11](#_Toc103864174)

[6.1 Испытание выполнения требований к программной документации 11](#_Toc103864175)

[6.2 Проверка требований к функциональным характеристикам 11](#_Toc103864176)

[6.3 Проверка требований к надежности 14](#_Toc103864177)

[ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ 15](#_Toc103864178)

АННОТАЦИЯ

Программа и методика испытаний – это документ, в котором содержится информация о программном продукте, а также полное описание приемочных испытаний для данного программного продукта.

Настоящая Программа и методика испытаний для «Web-приложение для запоминания стихов» содержит следующие разделы: «Объект испытаний», «Цель испытаний», «Требования к программе», «Требования к программной документации», «Средства и порядок испытаний».

В разделе «Объект испытаний» указано наименование и область применения программы.

В разделе «Цель испытаний» указана цель проведения испытаний. Раздел «Требования к программе» содержит основные требования к программе, которые подлежат проверке во время испытаний.

Раздел «Требования к программным документам» содержит состав программной документации, которая представляется на испытания.

Раздел «Средства и порядок испытаний» содержит информацию о технических и программных средствах, которые следует использовать во время испытаний, а также порядок этих испытаний.

Раздел «Методы испытаний» содержит информацию об используемых методах испытаний.

1. ОБЪЕКТ ИСПЫТАНИЙ
   1. Наименование программы

Наименование программы на русском языке: «Разработка MVP сервиса синтеза речи на английском языке».

Наименование программы на английском языке: «Development of an MVP service for speech synthesis in English».

* 1. Область применения

Синтезом речи называется процесс, в ходе которого из текста на некотором языке создается аудио, содержащего речь, информационное сообщение в которой носитель данного языка воспринимает как эквивалентное, передаваемому текстом. Синтез речи часто используется в системах с голосовым управлением, например, в голосовых помощниках.

В некоторых ситуациях возможно предварительно записать куски речи диктора и “склеивать” их при необходимости вывода информации в виде аудио. Такой подход является жизнеспособным в системах с ограниченном числом реплик, например, в навигаторах. Однако, в более сложных программах требуются синтезирование произвольных фраз, что делает описанный ранее подход невозможным. Данная программа использует NLP-подходы, что помогает решать описанные ограничения и позволяет превращать произвольные фразы на английском языке в речь.

Данная программа является MVP проектом, представляющим Telegram бота, позволяющего синтезировать аудио файл с речью на английском языке по введенному пользователем тексту.

Данная программа и методика испытаний раскрывает детали тестирования Acoustic Service и Telegram bot service, а также всей программы, если это необходимо для тестирования указанных сервисов.

1. ЦЕЛЬ ИСПЫТАНИЙ

Испытания проводились с целью проверки корректности выполнения функций программы, перечисленных в разделе «Требования к программе».

1. ТРЕБОВАНИЯ К ПРОГРАММЕ
   1. Требования к составу выполняемых функций

Программа должна соответствовать следующим требованиям к надежности и составу выполняемых функций, указанным в Техническом задании к данному проекту.

* + 1. Требования к Acoustic Service

1) Отвечает на GET запрос с параметром текста.

(вид запроса:

http://{адрес сервиса в локальной сети}:5000/?{текст+сообщения+пользователя} )

2) Полученный текст отправляется на обработку акустической модели. В сервисе используется Open-source реализации модели Fastpitch [15] (в основе архитектура Transformer [16]), заранее обученная на датасете LJSpeech-1.1 [20].

3) Ответ на запрос содержит сериализованную в формат json матрицу мел-спектрограммы.

* + 1. Требования к Telegram bot service

1. После начала диалога пользователем (команда пользователя ‘/start’) бот отправляет приветственное сообщение с предложением ввести текст на английском языке.
2. После ввода пользователем текстового сообщения бот отправляет http запрос к Dispatcher service с текстом, введенным пользователем, в качестве параметра.

(вид запроса:

http://{адрес Dispatcher service в локальной сети}:7000/?{текст+сообщения+пользователя})

1. Передаваемый в ответ аудиофайл в формате wav отправляется пользователю как аудиосообщение.
2. При возникновении ошибки синтеза речи на любом этапе или некорректном тексте исходного сообщения, пользователь информируется ответным сообщением об возникновении ошибки.
3. При вводе команды ‘/analytics’ или ‘/latency\_test {Количество параллельных пользователей} {время теста в секундах} ’ пользователем, чей id не включен в список id разработчиков, пользователю отправляется сообщение об ошибке.
4. При вводе команды ‘/analytics’ пользователем, чей id включен в список разработчиков, отправляется http запрос к Dispatcher service вида:

http://{адрес Dispatcher service в локальной сети}:7000/analytics.

1. Передаваемый в качестве ответа словарь в формате json со статистикой синтеза речи, представляется в виде текста и изображений и отправляется пользователю в виде текстовых сообщений и картинок соответственно.
2. После ввода команды ‘/latency\_test {Количество параллельных пользователей} {время теста в секундах} ’ Бот выполняет в течение заданного времени нагрузочное тестирование с заданным количеством пользователей.
3. Нагрузочное тестирование выполняется параллельными ‘пользователями’ (параллельно работающими функциями), каждый из которых делает один запрос в секунду. Каждый запрос имеет небольшую стандартную длину и некоторый шум для избежания получения результата из кэшированных данных.
4. Измеренные во время нагрузочного тестирования характеристики отправляются пользователю, который ввел соответствующую команду, в сообщения с текстом и изображения с графиком.
5. Отчет о нагрузочном тестировании включает в себя характеристики длительности выполнения запроса.
   * 1. Требование к организации взаимодействия сервисов

Каждый микросервис запускается в отдельном Docker контейнере Инструкции для программы docker [17] представлены в отдельных для каждого микросервиса файлах Dockerfile.

Инструкции для отдельного запуска и взаимодействия с микросервисами описаны в файлах ReadMe.md.

Также для запуска всех микросервисов при помощи команды ‘docker-compose up –build’ из корня MVP прописаны инструкции программе docker-compose [19] в файле docker-compose.yml. При запуске данной команды каждый микросервис запускается в отдельном контейнере, при этом контейнеры объединяются в локальную сеть, что обеспечивает взаимодействие сервисов.

* + 1. Требование к взаимодействию клиентской и серверной частей

Взаимодействие между клиентской и серверной частями должно осуществляться с помощью Telegram API.

* + 1. Требования к клиентской части

Клиентская часть приложения представлена чат ботом на в мессенджере «Telegram».

1. Начать диалог с ботом можно при помощи команды ‘/start’
2. Обычному пользователю после начала диалога должно быть выведено приветственное сообщение с предложением ввести текст на английском языке.
3. На каждое введенное сообщение на английском языке бот присылает аудиосообщение с синтезированной по введённому тексту речью или сообщение об ошибке в случае, если процесс завершился некорректно.
4. Для разработчиков (пользователей, чей id входит в ограниченный список) доступна команда ‘/analytics’. После получения данной команды пользователь получает сообщения с аналитическими данными о работе сервисов синтеза речи, собираемыми dispatcher service.
5. Для разработчиков (пользователей, чей id входит в ограниченный список) доступна команда ‘/latency\_test {Количество параллельных пользователей} {время теста в секундах} ’. После получения данной команды пользователь получает сообщения с результатами проведенного нагрузочного тестирования.
6. Бот отправляет сообщения пользователю на английском языке.
   1. Требования к надежности
      1. Требования к обеспечению надежного (устойчивого) функционирования программы

Программа не должна аварийно завершаться при любом наборе входных. Программа должна обеспечивать проверку корректности входных данных.

* + 1. Время восстановления после отказа

В случае отказа работы серверной части и последующей недоступности Telegram бота, время восстановления не должно превышать одни рабочие сутки.

* + 1. Отказы из-за некорректных действий оператора

После запуска программы на сервере отказ программы вследствие некорректных действий оператора должен быть исключён. В том числе должна быть исключена возможность непреднамеренного выключения программы, не связанного с техническими неполадками сервера.

1. ТРЕБОВАНИЯ К ПРОГРАММНОЙ ДОКУМЕНТАЦИИ
   1. Состав программной документации

Техническое задание (ГОСТ 19.201-78)

Программа и методика испытаний (ГОСТ 19.301-78)

Пояснительная записка (ГОСТ 19.404-79)

Текст программы (ГОСТ 19.401-78)

Руководство оператора (ГОСТ 19.505-79)

* 1. Специальные требования к программной документации

Все документы к программе должны быть выполнены в соответствии с ГОСТ 19.106-78 [6] и ГОСТ к этому виду документа.

Вся документация сдается в печатном виде, при этом она должна быть обязательно подписана руководителем организации, утвердившей документ на разработку, руководителем разработки и исполнителем перед сдачей курсовой работы в информационно-образовательную среду НИУ ВШЭ Smart LMS (Learning management system) и ее защитой.

1. СРЕДСТВА И ПОРЯДОК ИСПЫТАНИЙ
   1. Технические средства, используемые во время испытаний

Программа запускается на операционной системе Ubuntu 20.04.4.

У программы должна быть возможность использовать NVIDIA® Tesla® V100 в Yandex Cloud для синтеза аудио.

Необходимо иметь зарегистрированного в телеграмме бота.

Для испытания бота необходимо наличие устройства с доступом к платформе Telegram.

* 1. Программные средства, используемые во время испытаний

Дополнительных средств кроме Docker и Docker Compose необходимых для развертывания программы не нужно.

* 1. **Порядок проведения испытаний**

Испытания проводятся после запуска программы в соответствии с Руководством оператора.

Испытания должны проводиться в следующем порядке:

1. Проверка требований к программной документации
2. Проверка требований к функциональным характеристикам
3. Проверка требований к надежности
4. МЕТОДЫ ИСПЫТАНИЙ
   1. Испытание выполнения требований к программной документации

Состав программной документации проверяется визуально, проверяется наличие всех подписей и наличие программной документации в системе Smart LMS. Также визуально проверяется соответствие документации требованиям ГОСТ. Все документы удовлетворяют представленным требованиям.

* 1. Проверка требований к функциональным характеристикам

Проверка функциональных характеристик микросервисов проводится только в составе всего MVP, так как интерпретировать результаты работы Acoustic service без преобразования мел-спектрограмм в звук затруднительно, а цель работы Telegram Bot service - связь пользователя и Dispatcher service.

1. Для проверки работоспособности MVP и связи микросервисов необходимо запустить программу на сервере командой docker-compose up –build и убедиться, что все микросервисы запустились без ошибок.
2. Для начала диалога необходимо ввести в диалог с ботом команду ‘/start’ и убедиться, что бот выводит приветственное сообщение.

Изображение выглядит как текст

Автоматически созданное описание

1. Для проверки синтеза речи необходимо ввести текст на английском языке. Корректно работающая программа пришлет аудио сообщение. Убедиться, что аудио сообщение содержит речь на английском языке (озвученный текст, который был прислан ранее).

Изображение выглядит как текст

Автоматически созданное описание

1. Для проверки обработки некорректных сообщений необходимо ввести текст, содержащий нелатинские буквы (например, буквы русского алфовита), сообщение – смайлик и сообщение содержащее более 250 символов. Корректно работающая программа не реагирует на нетекстовые сообщения, а на некорректные текстовые выдает следующий результат.

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

1. Для проверки дополнительных функций программиста необходимо отправить сообщения с командами ‘/latency\_test {Количество параллельных пользователей} {время теста в секундах} ’ и ‘/analytics’ от лица пользователей чей id не входит в список программистов. Программа должна вывести сообщение об ошибке.

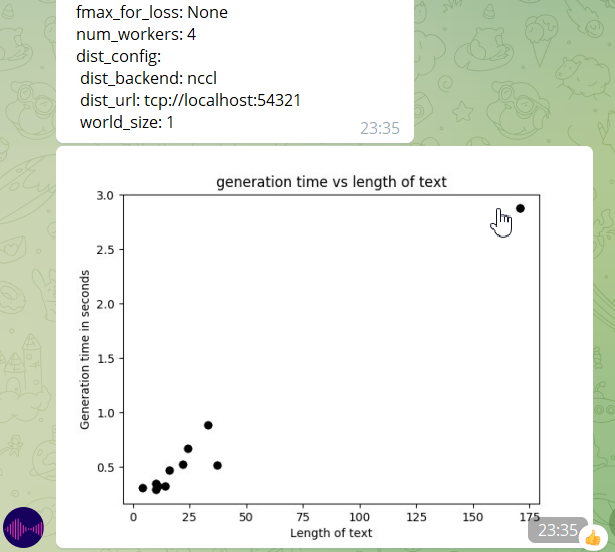
Изображение выглядит как текст

Автоматически созданное описаниеИзображение выглядит как текст

Автоматически созданное описание

1. Для проверки вывода аналитики работы синтеза речи необходимо ввести команду ‘/analytics’ от лица пользователя, чей id входит в список разработчиков. Корректно работающая программа выдает следующий результат.

Изображение выглядит как текст

Автоматически созданное описание

1. Для проверки проведения нагрузочного тестирования необходимо ввести команду

‘/latency\_test {Количество параллельных пользователей} {время теста в секундах} ’. Корректно работающая программа выдаст следующий результат

Изображение выглядит как текст

Автоматически созданное описание

В ходе выполнения предыдущих пунктов также автоматически тестируются требования к взаимодействию сервисов, требования к взаимодействию клиентской и серверной частей программы, а также все остальные явно непрописанные в предыдущих пунктах требования к функциональной части программы.

* 1. Проверка требований к надежности

Проверка надежности сервиса осуществляется путем наблюдения за работой программы. Корректно работающая программа не должна выключаться без явной команды программиста docker-compose down.

При перезапуске программы пользователю должно быть предложено перезапустить диалог с помощью команды ‘/start’. При этом команды программиста перезапуска диалога не требуют.

Изображение выглядит как текст

Автоматически созданное описание

# ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Лист регистрации изменений | | | | | | | | | |
| Номера листов (страниц) | | | | | Всего листов (страниц в докум.) | № документа | Входящий № сопроводительного докум. и дата | Подп. | Дата |
| Изм. | Измененных | Замененных | Новых | Аннулированных |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |