GovTech VICA Mini Project

# Technical Design Document

Version 1.0

Document Version Control

|  |  |  |  |
| --- | --- | --- | --- |
| Date Issued | Version | Description | Author |
| 11th Oct 2021 | 1.0 | Initial Draft | Mrinal |

Contributors

The content of this document has been authored with the combined input of the following group of key individuals.

|  |  |
| --- | --- |
| Name | Section Worked Upon |
| Mrinal | Initial Draft |

Table of Contents

[1 Technical Design Document 2](#_Toc84857459)

[1. Introduction 4](#_Toc84857460)

[High level objectives 4](#_Toc84857461)

[2 Workflow Overall 5](#_Toc84857462)

[Application Flow Pipeline 5](#_Toc84857463)

[3 Workflow Data Ingestion 6](#_Toc84857464)

[3.1 Data Sources: 6](#_Toc84857465)

[3.2 Technical solution design 7](#_Toc84857466)

[3.2.1 Data Insertion 7](#_Toc84857467)

[3.3 Method Definition 7](#_Toc84857468)

[3.2.2 Data Processing 7](#_Toc84857469)

[4 Deployment Steps 9](#_Toc84857470)

[5 Limitationand Future Scope 10](#_Toc84857471)

[6 Logging 11](#_Toc84857472)

[6.1 Technical solution design 11](#_Toc84857473)

[6.2 Common Logging Framework Code 11](#_Toc84857474)

# Introduction

In this project, you are provided with sample chatbot data extracted from our staging MongoDB (see Section 3 Data Schema for more information):

1. Sessions – Includes all the chat sessions of individual chatbots on our platform

2. Transcripts – Includes the corresponding transcripts of the sessions from (1). Note that each session consists of at least 1 transcript, and it can be generated by a different actor.

3. Apps – Includes the names of all chatbots.

**Objective: -**

You are required to analyse the data and build a scalable data ingestion pipeline, which can support the analytics APIs consumed by our VICA platform.

Here are some examples of the APIs:

1. Number of sessions of a chatbot application (appId)

2. Average session duration of a chatbot application

3. Average number of user queries per session of a chatbot application given the start date, end date and the chatbot’s appId

**Approach: -**

**Note: All the code will be written in python version 3.8**

## High level objectives

The high-level objectives are:

1. Reading/loading of data from the \*.json sources and convert them into pandas dataframe(details mentioned in the Data Ingestion Section).
2. Create a Pipeline for data ingestion , Transformation and Extract

# Workflow Overall

## Application Flow Pipeline

# Workflow Data Ingestion

## Data Sources:

3.1.1 Sessions

On a typical month, we record about a million sessions (50,000 are sesisons with actual user queries; the remaining are system-initiated sessions when people visit the websites which embeds the chat applications.

Schema:

• \_id : The session ID

• channel : The channel in which the chat session occured over (Message Bird, Web,

whatapps)

• app : The chatbot app ID in which this session occured

• createdAt : The session entry’s creation time

3.1.2 Transcripts Transcript can be generated by 4 different actors: user, agent, bot or system.

1. User transcripts – Questions/ utterances asked by the user when they interaction with the

chat application

2. Agent transcripts – Messages by the live support agent (e.g. a real person responding to

the chat session)

3. System transcripts – System-triggered messages, mainly for transactional/ functional purposes

(e.g. trigger welcome message)

4. Bot transcripts – Chatbot responses to the user questions/ utterances.

Each session has about 3 to 10 transcripts, depending on the popularity of the chat application, as well as how the chatbot responses are configured.

Schema:

• \_id : The transcript ID

• data : The data exchanged. For user / agent (live agent responding to the user), this is the

query they asked/ answered the chatbot. For bot / system, this is the json data exchanged

while communicating between the front and backend

• from : The actor of the exchange of data (user, agent, bot, system) • session : The session

ID in which this transcript occured

• createdAt : The transcript entry’s creation time 3.3 Apps Each chat application has a name.

Schema:

• \_id : The chatbot app ID

• appId : The alternative (readable) name of the chatbot application

## Technical solution design

## 3.2.1 Data Insertion

1) Load the Json file into data frame df\_app, df\_session,df\_transcripts using json\_normalize method of dataframe

2) Upon successful load it will return dataframe df\_app,df\_session,df\_transcripts

## Method Definition

|  |  |  |
| --- | --- | --- |
| **Class Name** | **Data\_Getter** |  |
| Method Name | def \_\_init\_\_ |  |
|  | Method Description | Initialize input datafile and logger class |
|  | Input parameter names | self, file\_object, logger\_object |
|  | ouptput | None |
|  | On Exception | Write the exception in the log file.  Raise an exception with the appropriate error message |
| Method Name | get\_data |  |
|  | Method Description | This method reads the data from source. |
|  | Input parameter names | self |
|  | ouptput | df\_app,df\_session,df\_transcripts |
|  | On Exception | Write the exception in the log file.  Raise an exception with the appropriate error message |

## 3.2.2 Data Processing

1) Change the column namd to valid column name

2) Merge all the data frame dataframe df\_app,df\_session,df\_transcripts using key columns

3) Transform the data to generate desired output using group by and filter Data

|  |  |  |
| --- | --- | --- |
| **Class Name** | dataTransform |  |
| Method Name | def \_\_init\_\_ |  |
|  | Method Description | Initialize input datafile and logger class |
|  | Input parameter names | self, file\_object, logger\_object, df\_app,df\_session,df\_transcripts |
|  | ouptput | None |
|  | On Exception | Write the exception in the log file.  Raise an exception with the appropriate error message |
| Method Name | changecolumnheader |  |
|  | Method Description | Change the column name |
|  | Input parameter names | self |
|  | ouptput | None |
|  | On Exception | Write the exception in the log file.  Raise an exception with the appropriate error message |
| Method Name | dataprocessing |  |
|  | Method Description | Transform the data to generate desired output using group by and filter Data |
|  | Input parameter names | self |
|  | ouptput | df\_app\_id\_cnt.csv  df\_app\_session\_cnt.csv  df\_app\_user\_cnt.csv |
|  | On Exception | Write the exception in the log file.  Raise an exception with the appropriate error message |

# Deployment Steps

Normal Setup

1. Conda create -n vica python=3.8
2. Conda activate vica
3. pip install -r requirements.txt
4. python main.py

Docker instance

1. Download the file.
2. Docker fle details

FROM python:3.8-slim-buster

WORKDIR /app

COPY . /app

RUN pip install -r requirements.txt

CMD [ "python", "./main.py" ]

1. docker build -t vica .
2. docker run --name test2 -v D:\Mrinal\DataPipeline\VICA\_pipeline:/app vica

Environment . Windows Server

Text

Description automatically generated

Text

Description automatically generated

Output

Graphical user interface, text, application

Description automatically generated

# Limitationand Future Scope

1. For local server pipeline we can try Airflow and compare which one the better option
2. Still need to work on to pull data from shared location as I have assumed that file will be present in root directory to ingest that data
3. Connectivity with rest api is not done
4. We can use NLP to check the sentiments on actor user
5. We can try for more analytical but retricted to the one which is mentioned in the problem statement.

# Logging

## Technical solution design



## Common Logging Framework Code

from datetime import datetime

class App\_Logger:

def \_\_init\_\_(self):

pass

def log(self, file\_object, log\_message):“””This method will be used for logging all the information to the file.”””

self.now = datetime.now()

self.date = self.now.date()

self.current\_time = self.now.strftime("%H:%M:%S")

file\_object.write(

|  |  |
| --- | --- |
| Class Name | App Logger |
| Method Name | log |
| Method Description | This method will be used for logging all the information to the file. |
| Input parameter names | self,file\_object, log\_message |
| Input Parameter Description | file\_object: the file where the logs will be written  log\_message: the message to be logged |
| ouptput | A log file with messages |

str(self.date) + "/" + str(self.current\_time) + "\t\t" + log\_message +"\n")