**Visual Workbench Readme**

Visual Workbench (VW) is a platform to create mock-up dashboards rapidly based on the report requirements definition, the input to can be text or voice​.

**NOTE**: Kindly have a look at the “Runbook” file to run or start the instance of the application.

**sMongoDB:**

* ORM Used is here is PyMongo
* Import the required packages such as pymongo and etc
* Set the pymongo client with local MongoDB instance IP – “mongodb://localhost:27017/”
* Create a Database and Collections wherever necessary.

**Application Design/Features:**

* Import the necessary libraries and packages.
* Error logs are created using logging packages.
* Set a custom path and for logs.
* CORS package must be installed and imported in the main application python file.
* Each line in the User story is called as widget.
* Number of lines is equal to number of widgets.
* The dashboard is created using Synthetic Data which is automatically created using python.
* The keywords here are extracted using standard python libraries which includes NLP, text extract, POS tagging, tokenization etc.
* The keywords here are of 4 types – Categorical Dimension, Measures, Time Line Dimension and Goal Measure.
* Categorical Dimension – A categorical variable is a variable type with two or more categories. Sometimes called a discrete variable, it is mainly classified into two (nominal and ordinal). Geography is also included.
* Time Line Dimension – keywords which represent a patterns accordance with the time and date.
* Measures – The keywords which represents a measurable quantity.
* Goal Measure – Represents a word which is part or mean of measures.
* Based on the Dimensions and Measures different kind of charts will be suggested using the mapping logic(Mapping Logic will be explained in the below slides.)
* The charts are arranged in a priority , so the first suggested chart will be followed by different kind of charts.
* Different kind of Templates are available to visualize in a better way.
* Filters are provided to filter the time period and as well as the dimension.
* app.py – The main file through which python interpreter is executed (Initiates the backend). Contains all the consolidated payload data.
* chart\_generator.py – File which generates the charts.
* constants.py – File used for speech to text. Contains the Subscription key, Location, Headers and the audio file name etc.
* data\_generator.py – The data for the charts are generated here.
* drill\_down.py - The data for the drill down are generated.
* entity\_extractor.py – The consolidated data are generated here( parser data are getting generated, text are converted into keywords, dataset)
* keywords\_ner.py – Used for the generating the version 3 keywords(NLP model).
* newChartTables.py – Used to insert the chart details in the DB.
* suggestions.py – Used to generate the suggestions for different kind of charts and data.
* Port Hosted – 7000

**API (Application Programming Interface)**

**USER:**

* Route - api.add\_resource(Users, '/user’)
* User API - Calls are split into 4 types.
* Major types are POST, GET, PUT DELETE
* Creation of New User, Delete Existing User, Modification of User Profile and View User.
* User Password is Encrypted.

**LOGIN:**

* Route - api.add\_resource(Login, '/login’)
* Major call – POST
* Used to make Successful / Failure login.

**WORKSPACES:**

* Route - api.add\_resource(Login, '/login’)
* Major call – POST
* Used to make Successful / Failure login.

**WIDGET:**

* Routue - api.add\_resource(Widget, '/widget’)
* Major call – POST, GET, PUT, DELETE
* Each sentence in the User Story is called as Widget.
* Number of Sentence in a User Story is equal to Number of Widget.
* Used to create a widget, modify / update a widget, view the existing widget and delete the existing widget.

**Data API:**

* Route - api.add\_resource(DataTable, '/data’)
* Major call – GET, PUT, DELETE
* The Dataset generation taken place here. On successful creation of workspace Dataset is generated.
* Modification / Updation, View the Data and Delete the Data taken place using the route.

**AllData:**

* Route - api.add\_resource(AllData, '/allData’)
* Major call – GET
* The Main API Call. Used to get the consolidated data from overall application.

**Chart Settings:**

* Route - api.add\_resource(ChartSetting, '/chart\_settings’)
* Major call – POST, PUT.
* Used to store the user chart setting.

**New Chart:**

* Route - api.add\_resource(NewChart, '/newchart’)
* Major call – POST, GET, PUT
* Used to add a New Chart in existing workspace, Modify the newly created Chart and to view the data.

**Chart Mapping:**

* Route - api.add\_resource(ChartMapping, '/chartmapping’)
* Major call – POST, GET, PUT, DELETE
* Chart Mapping Logic(Numeric) is created in order to generate the charts. Whereas each and every logic is associated with particular Chart.
* Used to create, update/modify, view and delete the mapping logic.

**Chart Filter:**

* Route - api.add\_resource(ChartFilter, '/chartfilter’)
* Major call – POST, GET, PUT
* Filters are applied to the existing workspaces to filter out specific data.
* Period filter and Category filter are the major type of filter.
* On creation of workspace the chart filter are automatically created.

**Uploader:**

* Route - api.add\_resource(Uploader, '/uploader’)
* Used to train the model.
* This is called during the creation of workspace.
* Major call – POST, PUT
* The keywords are trained here, the pattern should be followed to make a successful model.

**Layout :**

* Route - api.add\_resource(Layout, '/layout’)
* Major call – PUT
* Each and every chart as a layout to fit into the Dashboard.
* The layout for different kind of chart are created automatically during workspace creation.
* The modification of layout are carried over here.

**Training Keywords:**

* Route - api.add\_resource(Training\_Keywords, '/training\_keywords’)
* Major call – POST, GET, PUT, DELETE
* Keywords which are not recognised using NLP are trained manually in Temporary / Permanent Manner.
* Temporary Training – Keywords trained here are valid only in the current workspace.
* Permanent Training – Keywords trained here are valid across all the workspace.

**Second Instance:**

* Route - api.add\_resource(SecondInstance, '/second\_instance’)
* A new model only for generating the NLP (Keywords and it’s classification)

**Select All:**

* Route - api.add\_resource(SellectAll, '/deleteSelected’)
* Major call - DELETE
* Used to select all the required workspace and delete the selected at one go.

**Copy Paste:**

* Route - api.add\_resource(CopyPaste, '/copypaste’)
* Major call – POST
* Used to Duplicate a chart in the dashboard.

**Client Logo Uploader:**

* Route - api.add\_resource(ClientLogoUploader, '/clientLogoUploader’)
* Major call – POST, PUT, GET
* Used to upload a Logo Image from the User.
* Updation, Modification are made using API call.

**New Keywords:**

* Route - api.add\_resource(NewKeywords, '/newKeywords’)
* Major call – POST
* New-keywords are added over here.

**Drill Down:**

* Route - api.add\_resource(DrillDown, '/drillDown’)
* Major call – POST
* Automatically generated while creation of charts.
* Used to drill down the data in the chart to view the specific data in a much-advanced manner.

**New Chart Mapping:**

* Route - api.add\_resource(NewChartMapping, '/newChartMapping’)
* Major call – POST
* New Mapping Logic contains Mapping Logic Fields and it’s sum and total.
* Insertion of new mapping logic can be created using “\*” which implies all the numbers. Chart can be generated for different kind of pattern (Numeric Pattern). For example, Input to the sum will be given as 8 and all other fields are filled with zero then the pattern will be generated automatically from 000 to 888 automatically.

**New Drill Down:**

* New Drill Down Route - api.add\_resource(NewDrillDown, '/newDrillDown’)
* Major call – POST
* Doesn’t require any input, the drill down is made automatically using categorical dimensions.

**Design Document:**

* Route - api.add\_resource(DesignDocument, '/designDocument’)
* Major call – POST
* Data required for generating the Design Document are passed over here.

**Speech To Text:**

* Route - api.add\_resource(SpeechToText, '/speechToText’)
* Major call – POST
* Used to the speech converted into text to create a workspace.