

# MASTER OF TECHNOLOGY PROJECT REPORT

---

Mini Museum Chatbot System

---



## TEAM MEMBERS

XU JIACHEN LI XINLIN

MASTER OF TECHNOLOGY

# Contents

<b>1.0 Executive Summary .....</b>	<b>1</b>
<b>2.0 Problem Description .....</b>	<b>2</b>
<b>2.1 Project Objective .....</b>	<b>2</b>
<b>2.2 Project Scope .....</b>	<b>3</b>
<b>2.3 Problem Assumptions .....</b>	<b>4</b>
<b>2.4 Project Team.....</b>	<b>5</b>
<b>3.0 Solution .....</b>	<b>6</b>
<b>3.1 System Architecture .....</b>	<b>6</b>
<b>3.2 Process Flow .....</b>	<b>9</b>
<b>3.3 System's Features .....</b>	<b>11</b>
<b>3.4 Limitations .....</b>	<b>12</b>
<b>4.0 Data Extraction and Improvement.....</b>	<b>12</b>
<b>4.1 Data Extraction .....</b>	<b>12</b>
<b>4.2 Data pre-labeling and pre-processing .....</b>	<b>14</b>
<b>5.0 Implementation .....</b>	<b>15</b>
<b>5.1 Set up the DialogFlow Agent .....</b>	<b>15</b>
5.1.1 Set up the Entities for tagging parameters .....	15
5.1.2 Set up the intents and the training phrases .....	15
<b>5.2 Implement the Flask Server as the Fulfillment of DialogFlow .....</b>	<b>17</b>
<b>5.3 Deploy the DialogFlow to Action on Google (Google Assistant).....</b>	<b>18</b>
<b>5.4 Challenges in the implementation process .....</b>	<b>20</b>
<b>6.0 Conclusion .....</b>	<b>22</b>
<b>7.0 Bibliography.....</b>	<b>23</b>
<b>8.0 APPENDICES .....</b>	<b>24</b>
<b>APPENDIX A.....</b>	<b>24</b>

## 1.0 Executive Summary

This report is to introduce a chatbot system for the National Museum of Singapore. National Museum of Singapore is the nation's oldest museum that seeks to inspire with stories of Singapore and the world. It offers a wide range of exhibitions and programmes for visitors of all ages and group size. Every year, many tourists go there to visit various excellent exhibitions and experience exciting programmes.

However, the official website of the National Museum of Singapore does not provide excellent guidance and introduction to various exhibitions and programmes. The reason is that the majority of the information on its website is relatively simple and not classified. Furthermore, major travel information providers, like Tripadvisor and DianPing.com, are more inclined to provide more comprehensive recommendation and introduction services for popular tourist attractions than the less attractive destination like museums.

Our team consists of two Chinese students, we hope that all tourists can have a fascinating and enjoyable journey in the National Museum of Singapore, so we decide to create a system to let tourists get accurate and customised information more conveniently, and tourists can communicate with it using natural language.

The system is the Mini Museum, an intelligent chatbot. Tourist can use Google assistant to talk with it for fundamental information, even getting the recommendation of exhibitions and programmes based on their travel plan in the National Museum of Singapore.

To complete this intelligent chatbot, our group first set out to investigate the National Museum of Singapore's Website to get an overall understanding of it. For data collection, we used Python Selenium to scrape data from the official website, and then we pre-labelled and pre-processed the data and stored it into an excel file. For chatbot agent implementation, we used the techniques imparted to us in lectures and did some extension. We used Google DialogFlow to identify the users' intent, Flask as the backend of the chatbot to provide the relevant response, the Google Assistant (Action on Google) as the frontend to interact with users. Tourists can communicate with the Mini Museum anywhere anytime if they can use Google Assistant.

Our team had an exciting time working on this project, hoping to attract more and more people for the Nation Museum of Singapore by getting them a better service and user experience. It is widely different from a complicated intelligent agent which can communicate with tourists and tourists can never tell any difference between it and a real human volunteer or museum staff. We are looking forward that we can have the opportunity and technique to build an actual intelligent agent in the future. The process of working on this project is exciting and helpful for us to learn more in the NLP field.

## 2.0 Problem Description

The National Museum is a popular tourist attraction. It offers a wealth of exciting activities and exhibitions for tourists every year, like a workshop, family fun programme. Not just tourists from abroad, many Singaporeans often go to museums to loosen up.

Nowadays, more and more people like to travel freely, and they would prefer defining their tour content and itinerary, so they collect various information before starting their travel. Although the National Museum does provide a large amount of information on its official website and its application and there are also some corresponding introductions on major travel websites and applications. There are still the following points that cannot meet the needs of the tourist community.

1. The information provided by the official website and its app is not clear enough, and many exhibitions and programmes information is not classified. The search provided is an only content-based result without classified by activity-type category or group sizes category.
2. Although tourists can get much information by browsing the official website and its app, the interaction is not very user-friendly. Users still need to search and choose by themselves, lacking the ability to interact based on natural language.
3. Although the National Museum of Singapore does provide volunteer inquiry services, in the face of tourists with different tourist destinations with their travel preferences in different countries, the related services still not enough.

Therefore, for improving the quality of service and increasing the tourist experience, the National Museum of Singapore needs a better way to provide fundamental information and customize recommendation efficiently. Of course, museums can meet this demand in the form of more volunteers, but this requires many manpower, and it still cannot work on providing customize recommendation services for each visitor. After all, everyone has different needs, and it is unrealistic to ask every volunteer to be sufficiently familiar with the tickets, content and exhibition location information of each exhibition in the museum.

For these questions, our team feels that having a chatbot system is a better solution for the National Museum of Singapore.

### 2.1 Project Objective

Our goal is to create an intelligent chatbot system National Museum of Singapore that will be better to serve the tourists. The chatbot can offer information and services for tourists who come to visit the National Museum of Singapore with natural language.

The solution integrates with Action on Google, and Google DialogFlow, which can identify the intent and relevant entities from users' questions and users do not require to install any another app if they have already installed the Google Assistant on their smartphone. They can use the Google Assistant to call Mini Museum.

Because this is an intelligent chatbot with the backend server to provide the answer and Google Assistant and DialogFlow to provide the ability of understanding users' utterance, it

can handle questions from multiple users at the same time, no matter where they are and when they ask as long as they have access to internet and Google Assistant.

By implementing the Mini Museum chatbot System, we firmly believe tourists' travel experience in National Museum of Singapore will be improved. They can design their travel plan without comparing every post or article on many travel website and app. National Museum of Singapore will also benefit from it as they can save time to hire lots of volunteers or even a full-time human assistant to help tourists plan their journey.

The objectives of the project:

- To build a chatbot system for the National Museum of Singapore based on the information of its website
- To answer questions related to the National Museum of Singapore fundamental information, activities information (both exhibitions and programmers), retail and dining information and support information
- To recommend the activities (both exhibitions and programmers) based on the users' questions with elements in them, including group sizes, activity type, price and date.

## 2.2 Project Scope

The Mini Museum chatbot system will be built on web-based technologies and includes the following:

1. By using Actions on Google and Google assistant as the frontend and UI framework, we can use Google Assistant Virtual Components and users who have Google Assistant do not need to install another application on their phones.
2. The solution will include integration with Google's Cloud DialogFlow, which to identify the intents and entities. An agent intelligent agent will be deployed on DialogFlow set up with a webhook server URL that gets triggered by DialogFlow for each intent which can be called by the backend API interface. The responses generated by the backend server are sent back to the DialogFlow, and then DialogFlow will send the response back to the users' Google assistant
3. The webhook server solution is a Flask Web server. This HTTP Server will use Flask-Assistant Library for processing the requests sent by the DialogFlow Agent. It can identify the corresponding intent and automatically parse the parameters in those requests.  
For generating responses, it also can use the Google Assistant's response rules to package original response no matter it is a simple response or a rich message response.  
For the searching answer part, we use the excel file as the dataset which is already pre-processed by our team, and backend logic can use the parameters got from the request to get the information and package it as an answer.
4. Seven Google's DialogFlow's intent object groups:
  - a. MuseumInfo Intents
  - b. ExhibitionInfo Intents
  - c. ProgrammeInfo Intents
  - d. RetailInfo Intents

- e. DiningInfo Intents
- f. SupportInfo Intents
- g. RecommendInfo Intents

### 2.3 Problem Assumptions

For Mini Museum to be deemed successful, we have defined the following scenarios and desired outcomes:

Scenario	Desired Outcome
For a tourist who is interested in the fundamental information of the National Museum of Singapore	Mini Museum can provide the information on introduction, open time, location, ticket price, guided tours, group visits, accessibility, photography and Filming rules and venue rental
For a tourist who is interested in the exhibition's information of the National Museum of Singapore	Mini Museum can provide the information on exhibitions which are available about the location, open date, price and brief introduce
For a tourist who is interested in the programmes information of the National Museum of Singapore	Mini Museum can provide the information on programmes which are available about the location, open date, price and brief introduce
For a tourist who is interested in the retail information of the National Museum of Singapore	Mini Museum can provide the information on retail about the location, available time and brief introduce
For a tourist who is interested in the dining information of the National Museum of Singapore	Mini Museum can provide the information on dining about the location, available time and brief introduce
For a tourist who is interested in the support information of the National Museum of Singapore	Mini Museum can provide the information of support about how to be a volunteer or a docent, how to do individual or corporate support, and the Fellowship of the Museum
For a tourist who wants to get some recommendation of his journey in the National Museum of Singapore	Mini Museum will ask several questions to confirm the demands of his journey and provide the recommendation of exhibitions and programmes by the information got from tourist, e.g. the group type (For Family, Students) , the activity type (Workshop, Lectures), price and availability date.

## 2.4 Project Team

The project members titles and roles are listed as followings:

Name and Title	Role
XU JIACHEN (Architect, Developer)	Solution Architect, Solution design, Software Development, App Testing Project Documentation
LI XINLIN (Analyst, Developer)	Data mining, Solution design, Software Development, App Testing Project Video

The team has met twice a week to discuss project status and updates. Tasks were assigned, and areas of responsibilities were clearly outlined for each member

Our team worked together on researching materials, case studies, doing the needed system analysis and brainstorming and making sure we cover all possible scenario and questions a user may want to ask. The project is successfully executed, and the proposed solution is implemented on time and target

### 3.0 Solution

#### 3.1 System Architecture

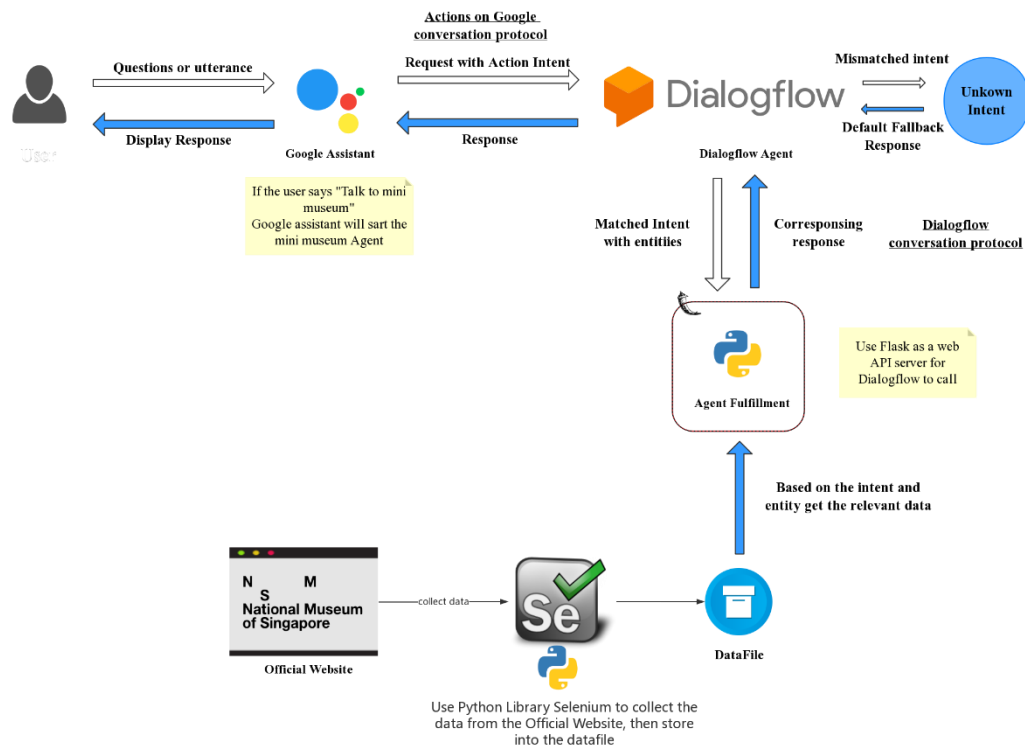


Figure 1 Mini Museum Chatbot (SystemArchitecture Design)

The Mini Museum chatbot system is a hybrid solution using the Google Assistant, Google's DialogFlow, Flask and Selenium.

The solution uses the Google Assistant as the frontend interface, which is provided by the DialogFlow's Actions on Google integration. With it, Google Assistant can easily invoke our agent, sends user utterances to it for matching the intent and gets responses back. When Dialogflow gets the request from the Google Assistant, it will call its fulfilment (a Flask Web Server) to get the response, translate the response with Actions on Google conversation protocol, and then sends back to Google Assistant.

DialogFlow is an excellent and easy-to-use NLU Engine provided by Google, which can identify the intent and entities through processing users' utterance based on what training phrases were set up and trained. After finishing the process, it will call the webhook backend fulfilment to generate the response if it can be called by the webhook based on the setting. If the webhook is not allowed to call the intent, Dialogflow will use the defined responses or generated by knowledge base set in the agent. In our case, we did not define any knowledge base in the Dialogflow.

For the intents and entities, we defined and trained eight groups, 50 intents and 21 entities except for the Default Welcome Intent and Default Fallback Intent.



Museum Intents	<ul style="list-style-type: none"> <li>• getMuseumDetail</li> <li>• getMuseumLocation</li> <li>• getMuseumLocationNC</li> <li>• getMuseumOpenTime</li> <li>• getMuseumOpenTimeNC</li> <li>• getMuseumGuidedTourInfo</li> <li>• getMuseumGroupVisits</li> <li>• getMuseumAmenitiesandEtiquetteInfo</li> <li>• getMuseumPhotographyAndFilmingInfo</li> <li>• getMuseumAccessibilityInfo</li> <li>• getMuseumTicketInfo</li> <li>• getMuseumTicketNC</li> <li>• getMuseumVenueRentalInfo</li> </ul>
Exhibition Intents	<ul style="list-style-type: none"> <li>• getExhibitionIntro</li> <li>• getExhibitionContent</li> <li>• getExhibitionLocation</li> <li>• getExhibitionLocationNC</li> <li>• getExhibitionTicket</li> <li>• getExhibitionTicketNC</li> <li>• getExhibitionTime</li> <li>• getExhibitionTimeNC</li> </ul>
Programme Intents	<ul style="list-style-type: none"> <li>• getProgrammeIntro</li> <li>• getProgrammeContent</li> <li>• getProgrammeLocation</li> <li>• getProgrammeLocationNC</li> <li>• getProgrammeTicket</li> <li>• getProgrammeTicketNC</li> <li>• getProgrammeTime</li> <li>• getProgrammeTimeNC</li> </ul>
Dining Intents	<ul style="list-style-type: none"> <li>• getDiningIntro</li> <li>• getDiningContent</li> <li>• getDiningTime</li> <li>• getDiningTimeNC</li> </ul>
Retail Intents	<ul style="list-style-type: none"> <li>• getRetailContent</li> <li>• getRetailIntro</li> <li>• getRetailTime</li> <li>• getRetailTimeNC</li> </ul>
Support Intent	<ul style="list-style-type: none"> <li>• getSupportDetail</li> <li>• getVolunteerInfo</li> <li>• getFellowshipInfo</li> </ul>
Recommend Intent	<ul style="list-style-type: none"> <li>• getRecommendInfo</li> <li>• getRecommendInfo_ActivityType</li> <li>• getRecommendInfo_GroupType</li> <li>• getRecommendInfo_Date</li> <li>• getRecommendInfo_DatePeriod</li> <li>• getRecommendInfo_Price_No</li> <li>• getRecommendInfo_Price_Yes</li> </ul>
Util	<ul style="list-style-type: none"> <li>• *getAction_intent_option_handler</li> <li>• Default Fallback Intent</li> <li>• Default Welcome Intent</li> </ul>

\*getAction\_intent\_option\_handler: this intent is to process tap event of List and Carousel option

There are two reasons why our team decide to define those intents.

The first reason is that we want to satisfy as many as possible scenarios, which may include the context demand. If a user wants to know the open time after he asked the question about the museum location, he does not need to say “museum open time” to get the open time information of the museum. Instead of it, he can just say “open time of it”, the chatbot will also provide the same with the open time information of the museum. We use the input and output context in the DialogFlow to remember the object or entity user mentioned before (based on the lifespan setting of each context)

The second reason is that we want to simplify the backend fulfilment by removing as many as possible “if-else” components.

For the backend server, we use one of the most popular python web framework Flask as the fulfilment of our Dialogflow agent, with a supporting library called Flask-Assistant. Using Flask-Assistant can simplify the router file in the chatbot system, Flask-assistant can pre-process the request from the DialogFlow for matching the corresponding function and get the parameters from the request (include the parameters in contexts). It also provides rich messages based on the Actions on Google Response Protocol, like Card, Suggest and List. However, it does not offer the Table Component of Actions on Google, so we define an additional table component based on the Actions on Google Response Protocol to improve that library.

For the data extraction part, we use Selenium. We also pre-label and pre-process those data collected from the official website of the National Museum of Singapore because the official website did not do the labelling and formatting, some of those catalogues or labels on the official website have never been used. Considering the size and the usage of the data, we did not use any database technology, just stored the data in the excel file.

### 3.2 Process Flow

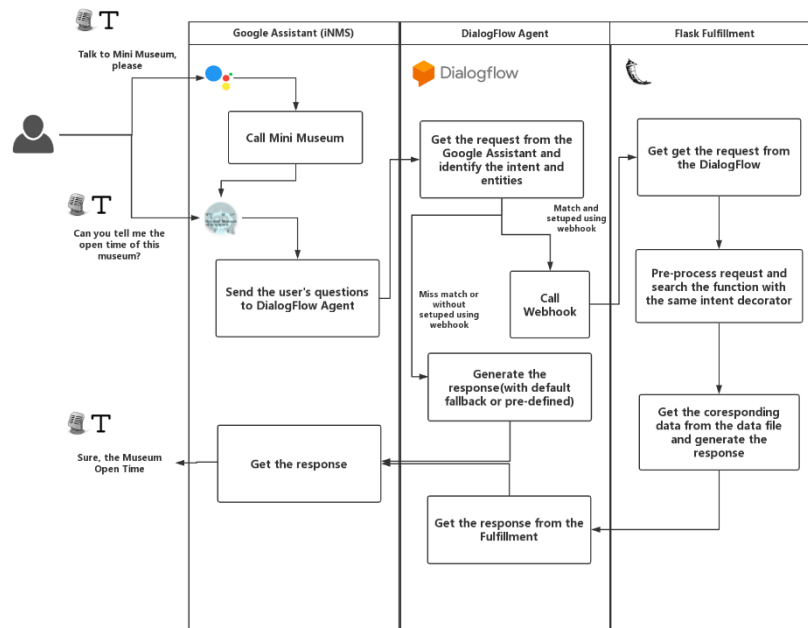


Figure 2 Process Flow

1. User Open the Google Assistant On the smartphone
2. Tap or say the "Talk to Mini Museum."
3. Google Assistant calls The Mini Museum and it starts the conversation with the user
4. User asks the question related to the National Museum of Singapore
5. Google Assistants uses the Actions on Google conversation protocol to package the user's query, and then sends to the DialogFlow Agent (Mini Museum)
6. DialogFlow Agent unpackage message and identify the intent and Entities from it. If the message miss-match all the intents. DialogFlow Agent will use the default fallback intent to generate the response. If the message is successfully matched to an intent. DialogFlow will call the webhook, then send the request packaged with DialogFlow conversation protocol to our Flask fulfillment.
7. Flask Fulfillment gets the request from the DialogFlow. With the support of the Flask-assistant, our web server will pre-process the request, which is getting the parameters and context and identifying the intent name from the request. Then match the intent name with the pre-defined function name in this decorator. After matching, the web server will use the corresponding function to generate the response with the parameters from the pre-process and sent the response back to the DialogFlow.
8. DialogFlow get the response from the Flask fulfillment, then use the Actions on Google conversation protocol to package the response and send back to the Google Assistant
9. The user gets the response displaying on the Google Assistant.

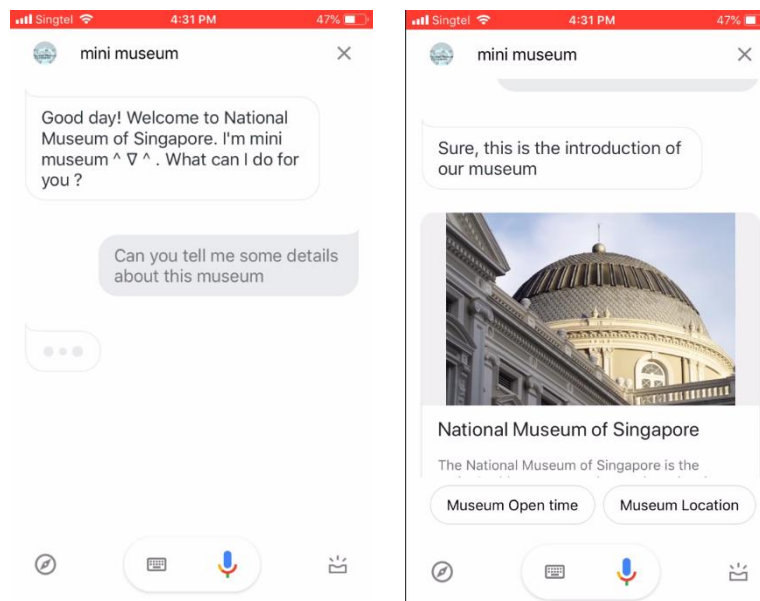


Figure 3 Museum Info

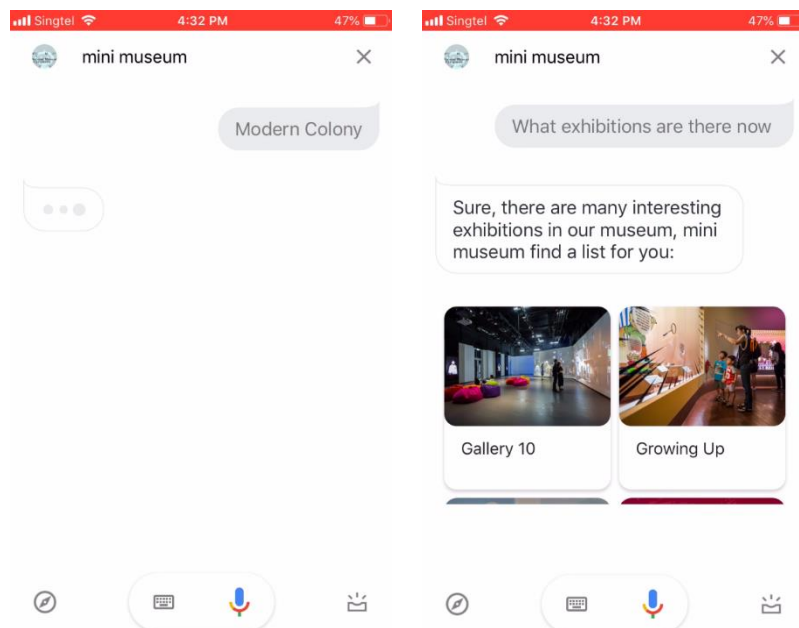


Figure 4 Exhibition Info

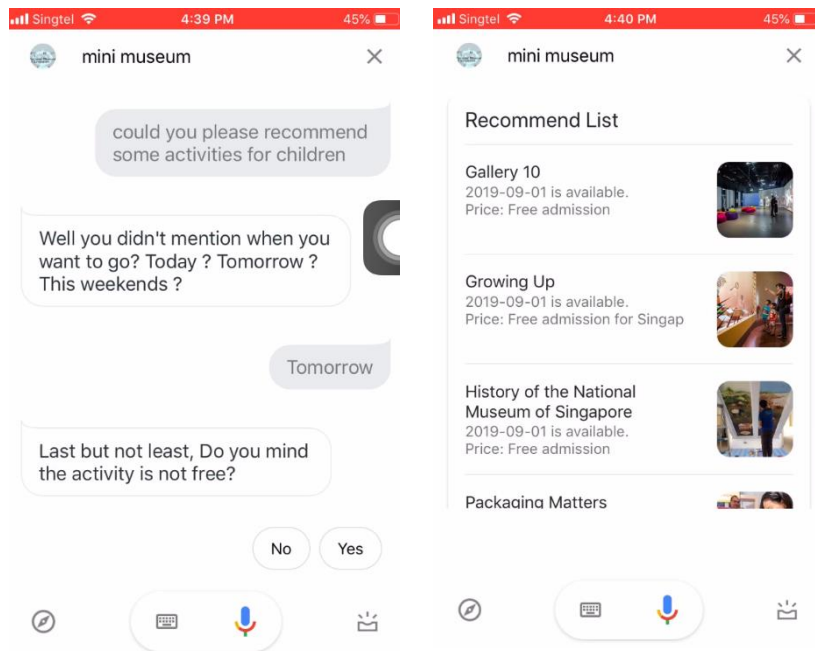


Figure 5 Recommend Info

### 3.3 System's Features

#### Ease of Access

The system is both easy to get and easy to use. No need to download any other application if the device has Google assistant. Any device with Google Assistant can communicate with the chatbot by typing or saying, "Talk to Mini Museum", Google Assistant will automatically call the Mini Museum and change the user interface to the chatbot. We choose to use the native Actions on Google components which already have been used in many other Actions on Google Application even in the Google Assistant itself, so people who are familiar with the Google assistant will not feel anything wrong when using our chatbot without any learning cost

#### Good Interactivity and Humanization Service

The system is a hybrid chatbot system which is consist of search and recommendation functions. It not only can understand users' questions and offer the related answer but also can recommend the activities in the National Museum of Singapore based on the users' demand or information they provide, which increase the tourists' experience before they start their tour of the museum.

#### More effective than use human resources in this field

The system can help the museum to reduce the number of those volunteers who mostly provide the introduce or answer the fundamental question for tourists. Then the museum can put more

workforce on the more critical field like guided tours and group visits. Separating volunteer service from simple inquiry service can not only improve the knowledge which students and social volunteers learn from but also provide tourists with better visiting services

### **3.4 Limitations**

The system can be further improved. However, it still has some limitations based on the current design and implementation

1. The chatbot can only get the official data from the official website of the National Museum of Singapore, most information is a brief introduction, especially for the exhibitions and programmes information. We cannot collect more data from it. Its knowledge is limited to what the team has gathered from the website
2. The chatbot can only respond in several models which are coded by our team rather than generate by itself because we did not use any machine learning or deep learning model with training data to train it

## **4.0 Data Extraction and Improvement**

### **4.1 Data Extraction**

In this project, our team focused on building a chatbot for the official website of the National Museum of Singapore. To achieve this objective, it is crucial to extract the data from the website effectively and decide how to pre-process and use the data.

Firstly, we found that the data on the website can be summarized into two parts. One part consists of museum relevant information include the admission fees, open time, rental information of this museum and so on. The other part consists of exhibition, programme, retail, dining and support relevant information. The reason why we put that information into one part is that those pages on the website have the same structure.

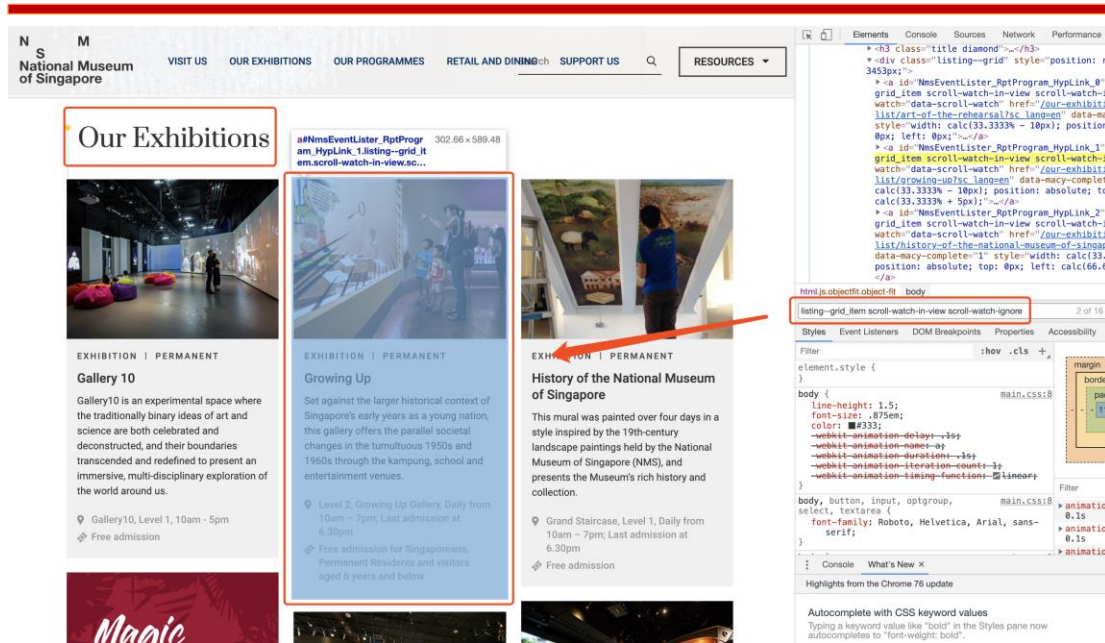


Figure 6 Website of exhibition

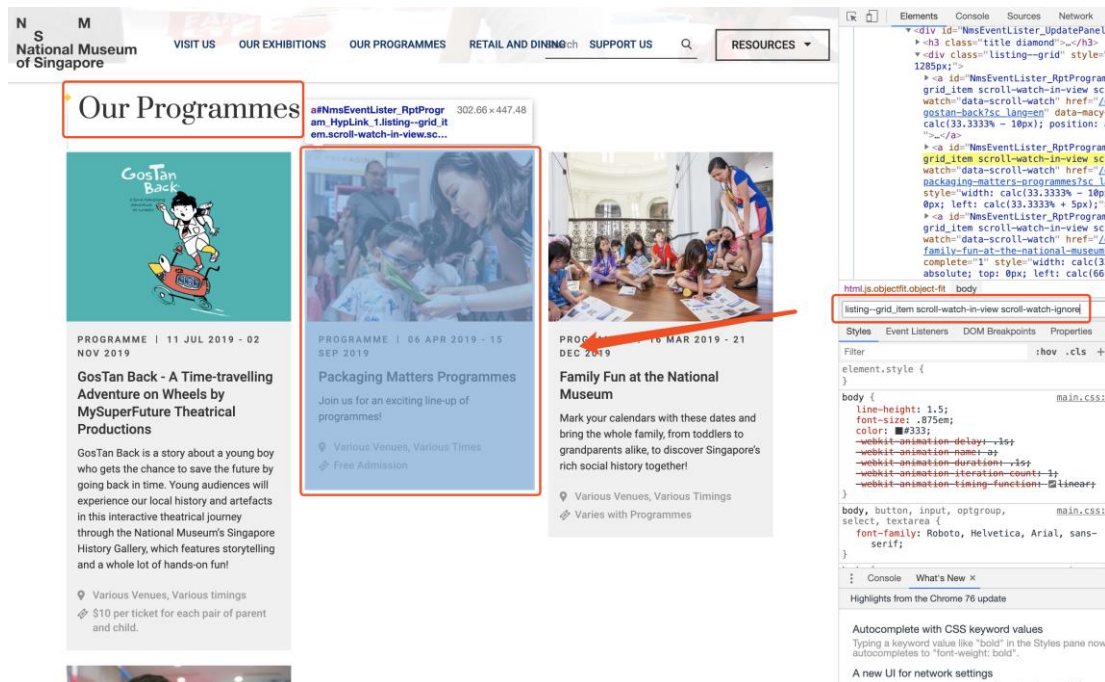


Figure 7 Website of programme

On the basis of this, we use python and selenium to extract the data from those pages. Firstly, we found out all the selectors to locate the structure components, include image, title, overview, location, open time and link. After that, we use selenium to drive the browser and find out the elements located by the selectors, then save the information we need to the lists automatically. Since those web pages can just load messages after scrolling down to its location, we wrote a script to scroll the pages automatically. After collecting all those data, we save it into the excel files.

The structure components and their selectors are listed below:



Structure	Selector
image	carousel--img
title	grid_item--title
overview	grid_item--content
location	grid_item--location
open time	grid_item--date
link	listing--grid_item scroll-watch-in-view scroll-watch-ignore

List of structure components and its selector

```

dataExtraction.py
7
8 def pagescroll():
9     jslist=['window.scrollTo(0,1200);','window.scrollTo(0,2400);','window.scrollTo(0,3600);','window.scrollTo(0,0);']
10    for js in jslist:
11        driver.execute_script(js)
12        time.sleep(0.5)
13
14 def get_data():
15     # driver = webdriver.Chrome()
16     # driver.get(url)
17     # pagescroll()
18     # exhlist = driver.find_elements_by_class_name(cn)
19     # for exh in exhlist:
20     #     print(exh.text)
21     # driver.quit()
22
23     exhurl='https://www.nationalmuseum.sg/our-exhibitions/exhibition-list'
24     prgurl='https://www.nationalmuseum.sg/our-programmes/programmes-list'
25     radurl='https://www.nationalmuseum.sg/retail-and-dinning/retail-and-dinning-list'
26     spturl='https://www.nationalmuseum.sg/support-us/support-us-list'
27     list_urls=[exhurl, prgurl, radurl]
28     list_buttons=['exhibitions', 'programmes', 'retail_and_dinning', 'support']
29     exhibitions=[]
30     programmes=[]
31     retail_and_dinning=[]
32     support=[]
33     dict={'exhibitions':exhibitions, "programmes":programmes, "retail_and_dinning":retail_and_dinning, "support":support}
34
35     for i in range(0,2):
36         driver.get(list_urls[i])
37         pagescroll()
38
39         element_name=driver.find_elements_by_class_name('grid_item--title')
40         element_mix=driver.find_elements_by_class_name('grid_item--location')
41         element_time=driver.find_elements_by_class_name('grid_item--date')
42         element_content=driver.find_elements_by_class_name('grid_item--content')
43         element_image=driver.find_elements_by_class_name('carousel--img')
44         element_url=driver.find_elements_by_css_selector("[class='listing--grid_item scroll-watch-in-view scroll-watch-ignore']")
45
46         mixlen = len(element_mix)
47         number = len(element_content)

```

Figure 8 Selector to locate the element

## 4.2 Data pre-labeling and pre-processing

Considering the missing categories of the programmes and exhibitions data, our team did pre-labeling and pre-processing step after data collections. Firstly, we used the one hot coding to label and classify those data, then we polished those data by adding some Markdown type tags following the Actions on Google Documentation to improve the answers' readability.

A	B	C	D	E	F	G	H	I	J	K
title	location	price	startdate	enddate	content	image	url	pagetext	Adults	Children
Gallery 10	Gallery10, Lev	Free admission	PERMANENT	PERMANENT	Gallery10 is an experimental space	https://www.na	https://www.na	Art of the Reh	1	1
Growing Up	Level 2, Growi	Free admission for Singaporean	PERMANENT	PERMANENT	Set against the larger historical con	https://www.na	https://www.na	The 1950s an	1	1
History of the	Grand Stairca	Free admission	PERMANENT	PERMANENT	This mural was painted over four da	https://www.na	https://www.na	This mural was	1	0
Magic and Mei	Level 2, Goh	Free admission for Singaporean	PERMANENT	PERMANENT	Step into the newly refreshed Goh	https://www.na	https://www.na	Step into the r	1	1
Modern Colon	Level 2, Mode	Free admission for Singaporean	PERMANENT	PERMANENT	By the end of the 19th century, Sin	https://www.na	https://www.na	With more edi	1	0
Singapore His	Level 1, Singa	Free admission for Singaporean	PERMANENT	PERMANENT	The Singapore History Gallery's upc	https://www.na	https://www.na	Singapura (12	1	0
Singapore, Ve	Level 2, Glass	Free admission for Singaporean	PERMANENT	PERMANENT	Created by renowned local photogr	https://www.na	https://www.na	At the bottom	1	0

Figure 9 Example Data after pre-labelling and pre-processing



## 5.0 Implementation

To implement this chatbot system requires three parts of setup and programming

1. Set up the DialogFlow Agent, including the entities and intents definition
2. Implement the Flask Server as the Fulfillment for DialogFlow
3. Add the Flask Server to the DialogFlow Fulfilment

### 5.1 Set up the DialogFlow Agent

Setting up the DialogFlow Agent, including two steps, to set up the entities for tagging parameters and to set up the intent and its training phrases for identifying the users' utterance.

#### 5.1.1 Set up the Entities for tagging parameters

There are 22 entities needed to set up in our system, including all the parameters needed to identify from the users' utterance or questions. For the setting part, we need to define several types of parameter's name(key), value and its synonyms, then in the next step for training the intent or in the real process flow, our DialogFlow agent will automatically identify and tag each related and similar entity and package them into the request. With that, the fulfilment can use them to search and generate the response for the specific intent

**InquiryLocation**

☒ Define synonyms ? ☐ Allow automated expansion

address	address
direction	direction
get to	get to
location	location
look for	look for
way to	way to
where	where

[Click here to edit entry](#)

**Entities**

Search entities

1 of 2

- @accessibility
- @amenities
- @BusinessHours
- @dining
- @exhibition
- @fellowship
- @groupvisits

Figure 10 Entities Setting

#### 5.1.2 Set up the intents and the training phrases

The setting intent the essential part of this system's implementation which define the number of question type or users' utterance the system should cover and set intent's name and parameters' name sent to the backend fulfilment, which is like the interface document to the backend.

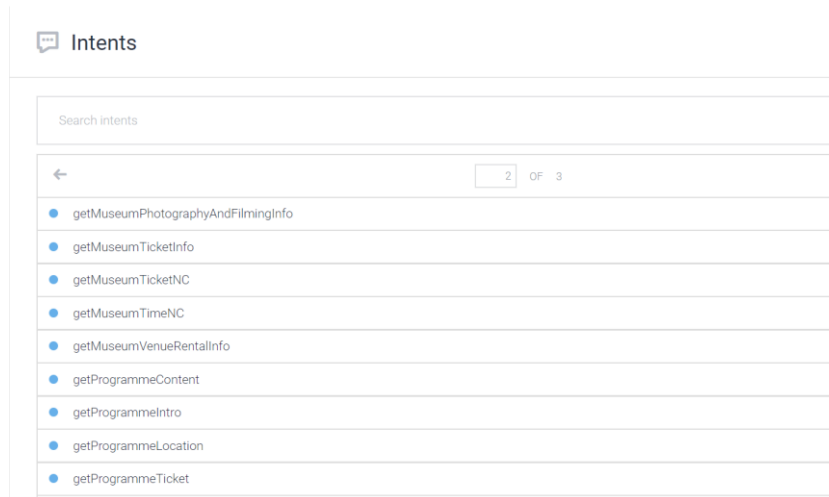


Figure 11 Intents

In our Mini Museum chatbot system, there are eight groups, 50 intents. (For the detail, See 3.1 System Architecture). They are responsible for different types of users' utterance and questions, respectively. We define different training phrases for each of intents to make them capable of identification and tagging entities

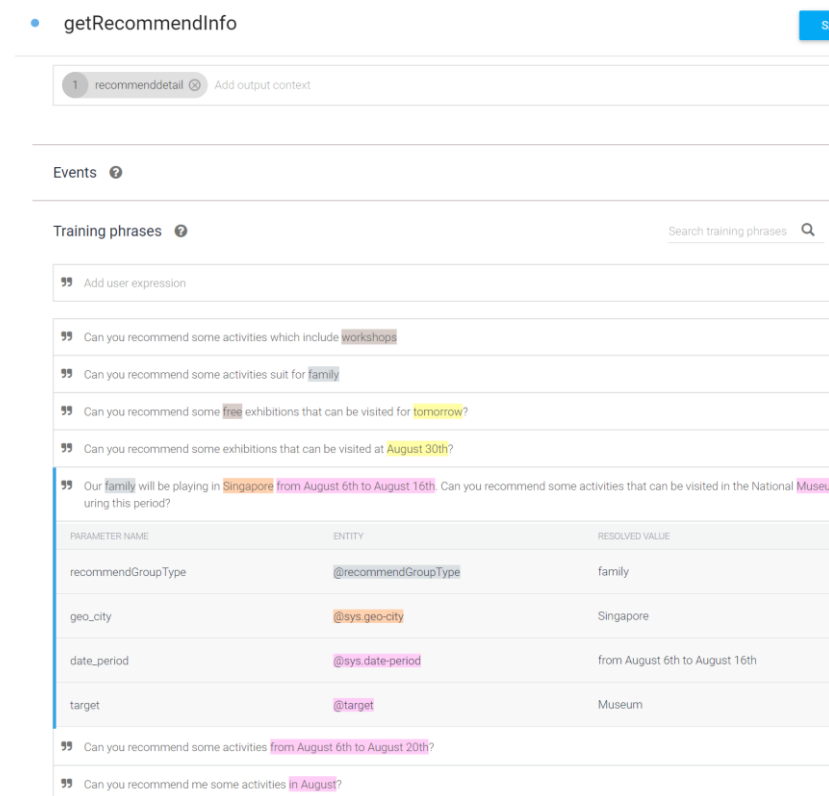


Figure 12 GetRecommendInfo Intent Detail

## 5.2 Implement the Flask Server as the Fulfillment of DialogFlow

For the fulfilment of DialogFlow agent, we decided to use the Flask as the Web framework of our system and the Flask-assistant to simplify and organize the intents processing function, which is a customized library for processing the request from the DialogFlow and generate the response following the Actions on Google documentation.

```
from flask import Flask
from flask_assistant import Assistant
from flask_assistant import ask, tell, event, build_item
from flask_assistant import context_manager

app = Flask(__name__)
app.config['INTEGRATIONS'] = ['ACTIONS_ON_GOOGLE']
assist = Assistant(app, route='/', project_id="minimuseum-snpwlr")
```

Figure 13 Backend app.py Flask-assistant setting

We use the Flask assistant to define the parameter of its decorator, and then it will pre-process the request from the DialogFlow and use the parameter of its decorator to match the target function then call that function to generate the response. This library makes the backend server more like a REST API Server without defining the flask's blueprint manually.

For the response generation part, we use the components provided by Flask-Assistant to assemble the answer searched from our data file, because our system's responses mostly use the rich message response. We should follow the following the Action on Google Documentation to generate and format our response. Then the rich message can be displayed on the Google Assistant. (Although this library is powerful and useful, there is still some inadequacy in it. Our team need to improve and modify it to make it capable for our system.)

```
# recommend
@assist.action('getRecommendInfo')
def getRecommendInfo(recommendGroupType, recommendActivityType, date_period, date, price):
    if recommendGroupType == "" and recommendActivityType == "":
        speech = "Of course, But What type of activity do you want me to recommend? For family? Is a Workshop ?"
        return ask(speech).suggest("Family Fun", "Workshop")
    if date_period == "" and date == "":
        speech = "Well you didn't mention when you want to go? Today ? Tomorrow ? This weekends ?"
        return ask(speech).suggest("Today", "Tomorrow", "This weekends")
    if price == "":
        speech = "Last but not least, Do you mind the activity is not free?"
        return ask(speech).suggest("No", "Yes")
    context_manager.clear_all()
    return getRecommendList(recommendGroupType=recommendGroupType,
                           recommendActivityType=recommendActivityType,
                           date=date,
                           date_period=date_period, price=price)
```

Figure 14 Backend app.py IntentHandler Example

Well, for searching and getting the relevant data from our data file, we use the pandas to read the data file and use the parameter from the Dialogflow Agent to locate the data. The process is like searching for a tree. For example, to get the data of Museum Open Time. The parameter is "Museum" and "Open Time". First, we use the "Museum" parameter to locate the sheet of data file we need to search, then use the "Open Time" parameter to find the target row which has the open time data of the museum. We also do the pre-processing on the data for making it more like a part of an answer to the users (4.0 Data Extraction and Improvement)

```
def isPgmExist(pgmname):
    _temp = PGM_DF[PGM_DF['title'] == pgmname]
    if _temp.empty:
        return False
    else:
        return True

def getPgmContent(pgmname):
    """
    desc :get the content of a specific programme
    param :the name of the specific programme
    return :the content of the specific programme
    """

    pgmcontent=PGM_DF[PGM_DF['title']==pgmname]['content'].values[0]
    pgmimage = PGM_DF[PGM_DF['title'] == pgmname]['image'].values[0]
    pgmlink=PGM_DF[PGM_DF['title'] == pgmname]['url'].values[0]
    return pgmcontent,pgmimage,pgmlink
```

Figure 15 Backend IntentHandler Searching Data Detail

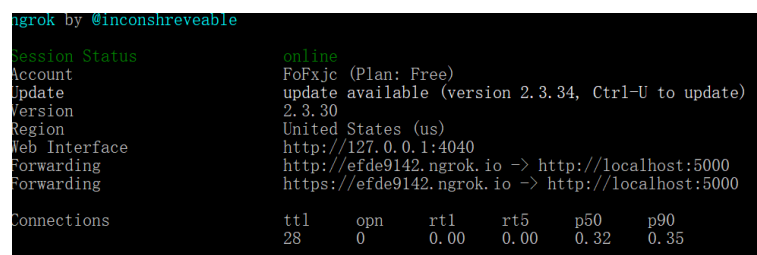
### 5.3 Deploy the DialogFlow to Action on Google (Google Assistant)

For Deployment, there are two steps needed to set up.

1. Setting the fulfilment of DialogFlow Agent to our Flask backend server

Because this system is still waiting for the customer or National Museum of Singapore to test, our team use the ngrok to expose flask backend running on the local machine to the internet

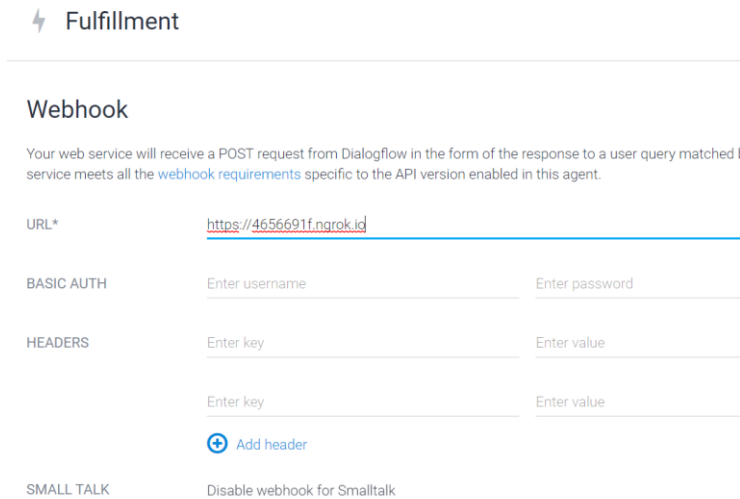
- 1) Use ngrok to get IP address used to expose the webserver



```
ngrok by @inconshreveable
Session Status      online
Account             FoFxcj (Plan: Free)
Update              update available (version 2.3.34, Ctrl-U to update)
Version              2.3.30
Region              United States (us)
Web Interface        http://127.0.0.1:4040
Forwarding           http://efde9142.ngrok.io -> http://localhost:5000
                    https://efde9142.ngrok.io -> http://localhost:5000
Connections
  ttl    opn    rt1    rt5    p50    p90
   28      0    0.00   0.00   0.32   0.35
```

Figure 16 start ngrok

- 2) Copy the IP address to the fulfilment setting page of the Dialogflow Agent



**Fulfillment**

### Webhook

Your web service will receive a POST request from Dialogflow in the form of the response to a user query matched to a service that meets all the [webhook requirements](#) specific to the API version enabled in this agent.

URL\*

BASIC AUTH

HEADERS

[+ Add header](#)

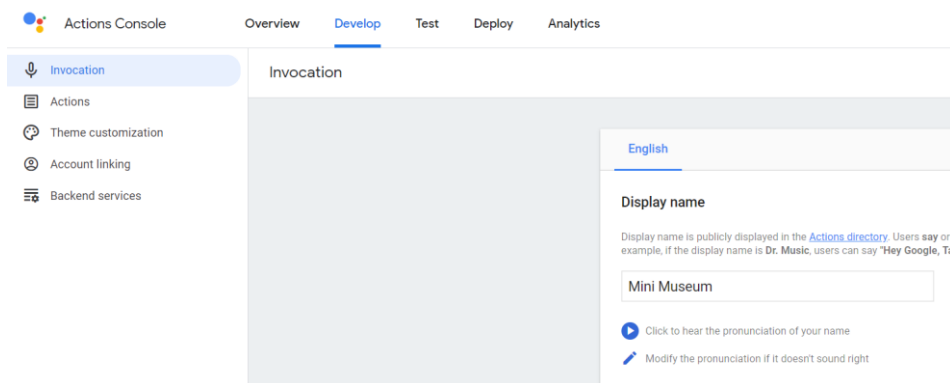
SMALL TALK ☐ Disable webhook for Smalltalk

Figure 17 Setting DialogFlow Agent's Fulfillment

- 3) Save the setting
- 4) Use the "python app.py" to launch the backend server

## 2. Deploy the DialogFlow agent to Google Assistant

Because this system is still waiting for the customer or the National Museum of Singapore to test, we cannot deploy the chatbot to the Google Assistant officially. We use our Google account to test and implement the system. The user experience will not be different from the real released system.



Actions Console Overview **Develop** Test Deploy Analytics

Invocation

English

**Display name**

Display name is publicly displayed in the [Actions directory](#). Users say or type the display name. For example, if the display name is Dr. Music, users can say "Hey Google, Tal Dr. Music".

[Click to hear the pronunciation of your name](#)

[Modify the pronunciation if it doesn't sound right](#)

Figure 18 Setting Display name for the invocation to Actions on Google

## 5.4 Challenges in the implementation process

### 1. Problem:

To use the Actions on Google or Google Assistant Components for the rich message response

#### **Solution:**

Firstly, we use the Flask-assistant library; it already has the Majority of Actions on Google components. We do not need to generate the JSON files following the Actions on Google Protocol by ourselves. But unfortunately, the library doesn't provide or create all the components, so our team decide to improve and modify it by adding some new function and class into it. We have read the source code and understand how this library generates and assembles the response and make them follow the Action on Google Protocol. After that, we added and modified the original code to fill in missing components. Though this library is not written by our team, we are very familiar with its logic and structure after the implementation

### 2. Problem:

How to use the context to improve the user experience, like remember key words users mentioned before and how to implement the recommendation function

#### **Solution:**

Our team have used context almost in all the intent to improve the users' experience.

We found that using context to remember those key words user mentioned before for a single field or single task is easy. Just set the fellow-up intent of the particular intent in the DialogFlow, then the primary intent will have an output context named by Dialogflow automatically, and the new-added fellow-up intent will have an input context and an output context with the same name. After setting, with the help of the Flask-assistant, we can easily get the parameter which stored in the context by the parent intent just like getting the parameters from the typical intent, but the only problem will be that you can't make the name of the parameter in the fellow-up intent same as the parameter stored in the context. Then the usage of context will be a success. For example, the user can type the "Open Time" after you have asked the "Museum Location", then you can get the accurate answer the Open Time of Museum.

However, In our case, we want to have multiple intents using context to remember multiple keywords, but according to the documentation of DialogFlow, if current conversation has one or more contexts, DialogFlow will only search or match the intents which have the same input context. For our system, multiple intents share the same entities. For example, both museum and exhibition have the @InqueryLocation entity. If the current conversation both have the museum and exhibition context, the DialogFlow may have the wrong matching.

To solve the problem above, our team decided to set every context having only one request lifespan and every potential follow-up intent having its own group's input and output context. After that, if a user keeps asking the same group question, the system will

always remember the entities the user mentioned before. If a user asks a different group question (like from museminfo to exhibitioninfo), the new context will be added into the conversation, and the last context will be expired immediately. Therefore, the old context will have no impact on the following request.

For implement the recommendation function, we not only use the context to remember the entities user has mentioned but also use the backend script to fill in the key information for recommendation by asking questions and do the one-hot code processing to the data file for getting the match with the parameter

### **3.Problem:**

How to solve the tap event in the Google assistant?

### **Solution:**

In the google assistant, there is a type of input called tapping the List or Carousel option. If a user taps one option or item of a list, the key or title of this option will be typed into the conversation just like the user type and send. According to the documentation, the tap will see an event named Action\_intent\_option and every intent with this event will be trigger without matching the training phrases. If the system has two intent which use tap event to get a different response, there will be a problem. So, after studying with the documentation of DialogFlow, we decided to define one particular intent to act as a listener to collect the Action\_intent\_option event then use the backend script to distribute the event

## 6.0 Conclusion

In this project, our team spent a lot of time researching and analysing the information obtained from the official website of the National Museum and experienced the work as a data analyst and data scientist, especially for data pre-processing and leaning. Ability to have the best presentation in Google Assistant for a great user experience

At the same time, we also spent much time researching DialogFlow and Action on Google documents. Although using Google Assistant as a front-end has many benefits, such as several predefined components, native integration support by DialogFlow support, many of them also bring us some trouble. Google Assistant does not support resizing the image of the Card or List. The tap events of it require additional intent to deal with, and only having better support for node.js, for python can only Write JSON or find a third-party library. Our team spent nearly two days researching Flask-assistant to analyse and understand how context and event can pass context parameters, and the display of the tickets has better user experience, and we have also optimized the Flask-assistant. Finally, a product that is more in line with our initial goals

In our implementation process, our group with two members had very active discussions and brainstorming about the architecture and the implementation. Because we are from different domains with different skill sets and mindset, it was tough and took much time to reach an agreement. However, though that process, we learnt a lot from others. It is an excellent opportunity for us not only to apply the cognitive system knowledge and solve real-life problems but also an excellent time for every group member to experience and learn new things.



## 7.0 Bibliography

Actions on Google integration <https://dialogflow.com/docs/integrations/actions/integration>

Flask-Assistant Document [https://flask-assistant.readthedocs.io/en/latest/quick\\_start.html](https://flask-assistant.readthedocs.io/en/latest/quick_start.html)

DialogFlow Document <https://dialogflow.com/docs>

Actions on Google Visual Components

<https://designguidelines.withgoogle.com/conversation/visual-components/>

## 8.0 APPENDICES

### APPENDIX A

#### Entities

Entity Name	Key	Value and synonyms
accessibility	• accessibility	• accessibility
amenities	• amenities	• amenities
	• etiquette	• etiquette
dining	• Flutes	• Flutes
	• Food for Thought	• FFT, Food for Thou..., Food for Thought
	• Janice Wong Singapore [CLOSED PERMANENTLY]	• Janice Wong S..., Janice Wong Singapore [CLOSED PERMANENTLY]
exhibition	• Galley 10	• Gallery 10, Gallery 10, gallery 10, gallery 10
	• Growing Up	• Growing Up
	• History of the National Museum of Singapore	• History of th..., History of the National Museum of Singapore
	• Magic and Menace	• Magic and Men..., Magic and Menace
	• Modern Colony	• Modern Colony
	• Moving Memories	• Moving Memories, Moving Memori...
	• Packaging Matters	• Packaging Mat..., Packaging Matters
	• REUNION	• REUNION
	• Singapore History Gallery	• Singapore His..., Singapore History Gallery
	• Singapore, Very Old Tree	• Singapore, Ve..., Singapore, Very Old Tree
	• Story of the Forest	• Story of th..., Story of the Forest
	• Surviving Syonan	• Surviving Syo..., Surviving Syonan
	• Voices of Singapore	• Voices of Sin..., Voices of Singapore
	• Wings of a Rich Manoeuvre	• Wings of a Ri..., Wings of a Rich Manoeuvre
fellowship	• NMSRF	• NMSRF, nmsrf
	• fellowship	• fellowship
groiupvisits	• groupvisits	• group visit, group visits, groupvisits
guidedtours	• guidedtours	• guided tour, guided tours, guidedtours
inquiryContent	• about	• about
	• content	• content
	• detail	• detail
	• details	• details
	• details about	• details about
	• information	• information
	• introduce	• introduce
	• talk about	• talk about
inquiryLocation	• address	• address
	• direction	• direction
	• get to	• get to
	• location	• location
	• look for	• look for
	• way to	• way to
	• where	• where
inquiryTicket	• book	• book
	• fee	• fee
	• price	• price
	• ticket	• ticket
	• tickets	• tickets
inquiryTime	• open time	• open time
	• time	• time
phtography	• filming	• filming, film, recording, vlog
	• photography	• camera, photo, photography, take a picture
programme	• Family Fun at the National Museum	• Family Fun at the National Museum, Family Fun at...
	• GosTan Back - A Time-travelling Adventure on Wheels by MySuperFuture Theatrical Productions	• GosTan Back - A Time-travelling Adventure on Wheels by MySuperFuture Theatrical Productions, GosTan Back - ...
	• HistoriaSG: Will Pragmatism Undermine Singapore's National Identity?	• HistoriaSG: Will Pragmatism Undermine Singapore's National Identity?, HistoriaSG:W...
	• National Day Open House	• National Day Open House, National Day ...
	• Packaging Matters Programmes	

	<ul style="list-style-type: none"> <li>School Programmes</li> </ul>	<ul style="list-style-type: none"> <li>Packaging Matters Programmes, Packaging Mat...</li> <li>School Programmes, School Progra...</li> </ul>
recommendActivityType	<ul style="list-style-type: none"> <li>Family Fun</li> <li>Film Screenings</li> <li>Lectures</li> <li>Seminars</li> <li>Storytelling</li> <li>Tours</li> <li>Workshops</li> </ul>	<ul style="list-style-type: none"> <li>Family Fun</li> <li>Film Screenings</li> <li>Lectures</li> <li>Seminars</li> <li>Storytelling</li> <li>Tours</li> <li>Workshops</li> </ul>
recommendGroupType	<ul style="list-style-type: none"> <li>Adults</li> <li>Children</li> <li>Families</li> <li>Seniors</li> <li>Special Needs</li> <li>Students</li> <li>Teachers</li> <li>Museum Label</li> </ul>	<ul style="list-style-type: none"> <li>Adults, adults</li> <li>Children</li> <li>Families</li> <li>Seniors</li> <li>Special Needs</li> <li>Students</li> <li>Teachers</li> <li>Museum Label</li> </ul>
retail	<ul style="list-style-type: none"> <li>Support</li> </ul>	<ul style="list-style-type: none"> <li>support</li> </ul>
support	<ul style="list-style-type: none"> <li>corporate</li> </ul>	<ul style="list-style-type: none"> <li>corporate</li> </ul>
supporttype	<ul style="list-style-type: none"> <li>donors</li> <li>individual</li> </ul>	<ul style="list-style-type: none"> <li>donor, donors</li> <li>individual</li> </ul>
target	<ul style="list-style-type: none"> <li>Dining</li> <li>exhibition</li> <li>museum</li> <li>programme</li> <li>retail</li> </ul>	<ul style="list-style-type: none"> <li>Dining, lunch, brunch, dining, dinner, restaurant</li> <li>exhibition exhibitions</li> <li>museum</li> <li>programme, programme, programmes</li> <li>retail store, retail, souvenir shop, souvenir shops, souvenirs</li> </ul>
venuerental	<ul style="list-style-type: none"> <li>rental</li> <li>venue rental</li> <li>rent</li> </ul>	<ul style="list-style-type: none"> <li>rental</li> <li>venue rental, VenueRental</li> <li>rent</li> </ul>
volunteer	<ul style="list-style-type: none"> <li>volunteer</li> <li>docent</li> </ul>	<ul style="list-style-type: none"> <li>be a service, volunteer</li> <li>docent</li> </ul>