



UNSW
A U S T R A L I A

COMP9900
Term 1, 2021
Information Technology Project

Project Report

Project Title: Chatbot
Group Name: 9900-W18A-OuterHeaven
Submission Date: 23-Apr-2021

	Name	Email	ZID	Role
1.	Pei Wang	z3290805@ad.unsw.edu.au	z3290805	Design & Developer
2.	Sriram Murali	z5232023@ad.unsw.edu.au	z5232023	Scrum Master, Tester
3.	Janvi Paresh Sheth	z5251994@ad.unsw.edu.au	z5251994	Design & Developer
4.	Ziyang Geng	z5192405@ad.unsw.edu.au	z5192405	Design & Developer
5.	Darshana Pritam Shah	z5298610@ad.unsw.edu.au	z5298610	Design & Developer

Tutor: Rachid Hamadi
Lecturer: Iwan Budiman

Table of Contents

1. Background for the Project	3
2. Aim / Project Objectives	4
3. System Architecture	6
Interface/Presentation Layer:	7
Fulfilment/Business Layer:	7
A Flask-based Server:	7
Database Layer:	8
Infrastructure Layer:	8
4. Key Features and Interface	8
4.1. Web User Interface for Chatbot Interaction	15
4.2. Recommendation Module	15
4.2.1 KNN based Movie Recommendation	15
4.2.2 Find New Friends	15
4.3. Auto-filling forms on behalf of the user	15
4.4. Virtual Collaboration	15
4.5. BI Module for Users and the Company	16
5. Third Party Tools and their Utility in the Project	16
5.1. Dialogflow	16
5.2. Datalab	19
5.3. APIs	19
5.4. WIX Website Builder	19
5.5. Ngrok	20
5.6. Dataflow	20
6. Implementation Challenges	24
6.1. Recommendation Module Algorithm	24
6.2. TABLEAU Integration	21
6.3. Movie Scheduler and Chat Feature	25
7. Installation Instruction	25
8. Project Management	36
8.1. Communication Plan	36
8.2. Sprint Schedule	36
8.3. Project Plan	37
9. References	37

1. Background for the Project

Recently, chatbots have started gaining momentum in helping businesses grow due to its various advantages. Chatbots have evolved from simple rule-based chatbots to advanced AI-based chatbots. Artificial intelligence and various machine learning algorithms are used to enhance the performance of chatbots. Chatbots are replacing jobs that require human interaction, for example, customer service agents.

Due to COVID-19, a lot of outlets have been closed which has impacted the communication between businesses and their customers. Chatbots would play an important role here in bridging this gap as the customers would be able to communicate with the businesses through the chatbots online. Movies are shifting from the traditional platforms (i.e., theatres) to online platforms which the viewers can access at any time from their laptops, TVs or even their smartphones. This shift has been further accelerated due to COVID-19.

Mainly, conversational chatbots are gaining popularity due to its user-friendly interface and personalised service to the users. An example of a conversational chatbot for the application of smart home assistant is Jamura which is built on Telegram and Google Dialogflow [1]. Its proposed framework includes RESTful APIs, webhooks and a UI interface. Jamura is designed mainly for Internet of Things (IoT) applications. Some of the drawbacks of Jamura are its limitation to work with restricted set of queries and responding to authorised users only due to hard-coded user IDs.

With the help of chatbots, the digital media platforms can build and target a variety of new audience. Digital streaming platforms like Netflix can also make use of chatbots to boost sales opportunities. With the help of artificial intelligence, the income of users can be predicted based on their location, viewing history, device type, etc. so that the appropriate packages/upgrades can be presented to the appropriate users to ensure a guaranteed sale thereby generating a guaranteed profit. Chatbots can be modelled to analyze a user's past behavior, learn about their preferences with the help of powerful machine learning algorithms and artificial intelligence. Through this, they can predict user behavior or make forecasts and thereby suggest movies and shows that the user is most likely to watch. They can also motivate users to continue watching content or redirect them to new content based on their preferences and behavior thereby promoting new content too.

Chatbots can also provide numerous services to viewers without any human intervention such as personalized content recommendations and quick suggestions which will save the time spent on navigating and browsing content. This will lead to an improved viewing experience for the users and increase engagement. There has been a substantial amount of work done related to providing recommendations to the users. An intelligent chatbot system has been implemented on a platform called Echo. This chatbot gathers user preferences and user knowledge base to provide user-centric recommendations in the travel domain. The chatbot uses Restricted Boltzmann Machine (RBM) along with Collaborative Filtering [2]. Another example uses user similarity to recommend movies to the user. User similarity is obtained by classifying similar user groups that have similar preferences through their ratings for each movie. Recurrent Neural Networks (RNN) learning methods is applied to learn about similar user groups. Following this, a model is

created to recommend movies to users [3]. Apart from recommendations, chatbots also provide an easy form of communication for the viewers. They can keep the users in loop by always being available and answering their queries irrespective of the time zones.

Furthermore, the digital media platforms in the entertainment industry can build their brand loyalty among their users by providing personalized treatments to them through chatbots. Usually there is limited customer support available online to assist the viewers and chatbots could be the solution. Offline 24/7 customer support is more expensive as the night shifts' payments are slightly on the higher side than day shifts' payments. Viewers tend to watch movies and shows in their free time which is generally after the working hours in the evening or at night. Due to higher expenses, companies generally do not invest in customer support that is available during these hours thereby causing communication issues. Chatbots help in solving these issues as they can work 24/7. They are also comparatively more efficient and less expensive.

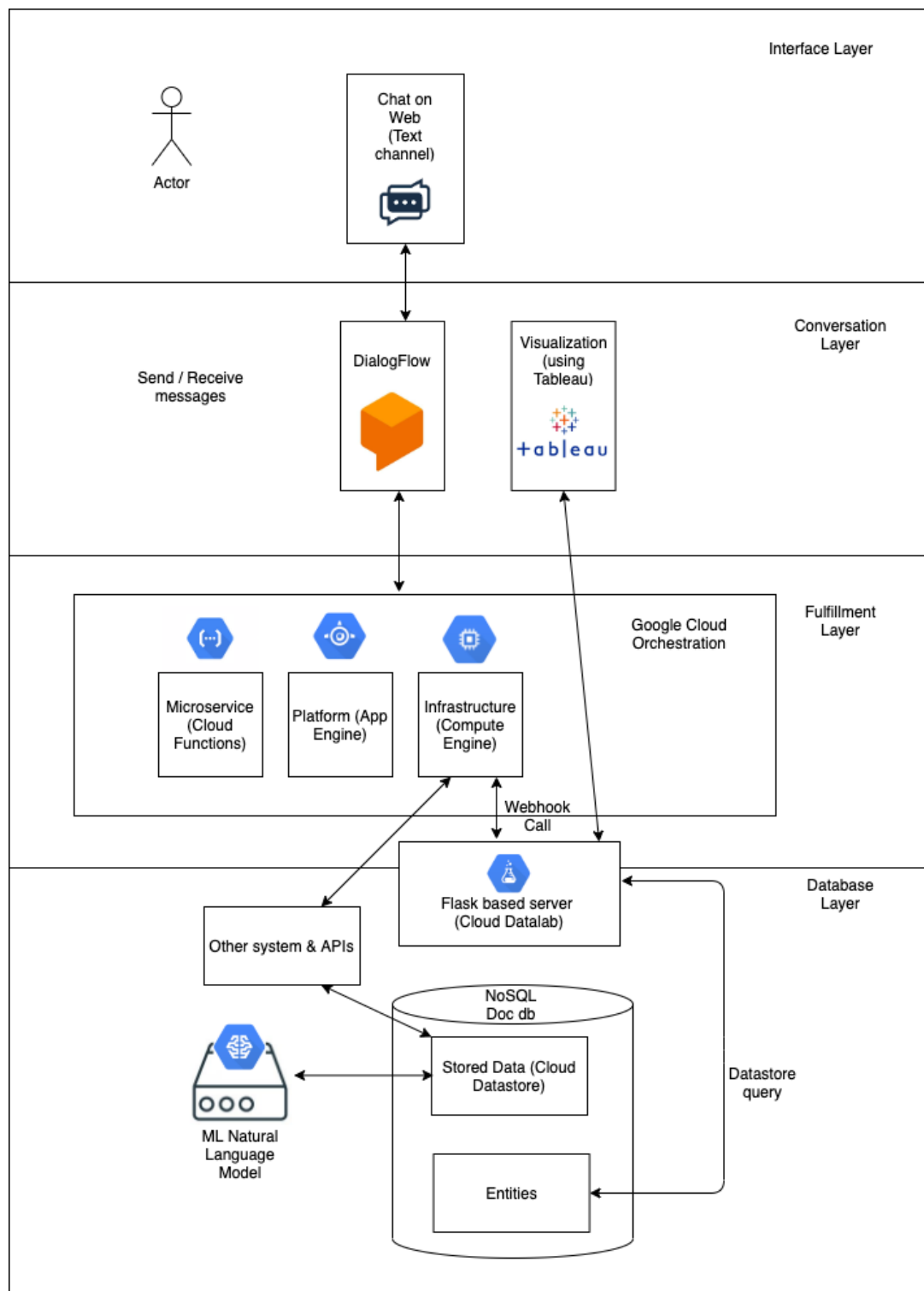
Through our project, we aim to build an interactive AI-based chatbot that could solve these issues and provide the benefits listed above.

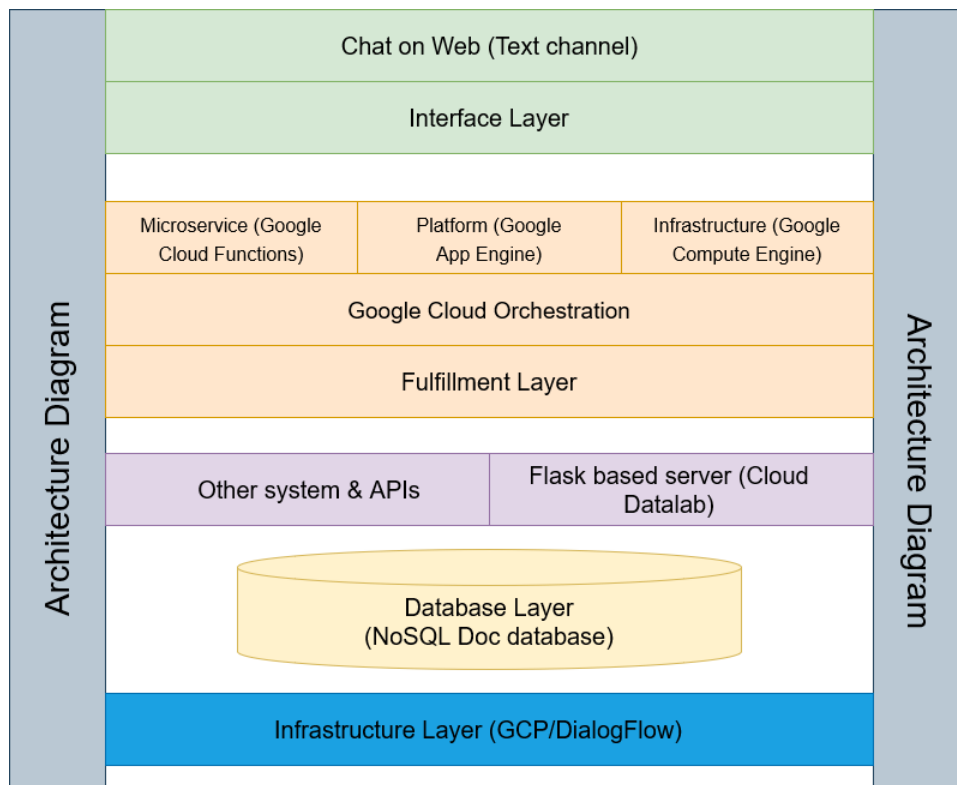
2. Aim / Project Objectives

1. To build a chatbot along with a knowledge extraction module based on the content and user behavior servicing an online content provider in the entertainment industry.
2. To build a dialog builder that leverages on existing APIs and keeps track of the contextual conversation between the user and the chatbot.
3. To build a web application to deploy the chatbot to a simple web chat.
4. To build a chat module with predefined set of questions based on the user FAQs in case the default module is unable to help the user with his/her objective.
5. To build a novel module that recommends content to the user based on their contacts' and groups' viewing history; this module will depend on the privacy settings of each user.
6. To build a module that will enable the chatbot to get instructions from the user and auto update various details such as liking and rating movies, shows, etc. in addition to auto populating webpages with user details and present it to the user for review and submission, such as payment processing for subscription updates thereby reducing the workload for users.
7. To build a novel module to enable the chatbot to organize a movie meeting with the user and the user's friends based on a simple 'Yes/No' response.

8. To build a module that will generate meaningful reports for the user and the company by analyzing user behavior patterns with the help of Business Intelligence (BI).

3. System Architecture





The system has the following components:

Interface/Presentation Layer:

The conversational interface of this chatbot primarily uses the text format, but it can also accept voice commands. The user can chat on a web interface using natural and rich interactions.

Fulfilment/Business Layer:

This layer is what connects the virtual agent to the external systems to access the backend such as the database, APIs, or other directories. It is used to fetch dynamic information to adhere to the user's request and keeps the conversation flowing. The fulfilment layer used Google Cloud Orchestration, which contains Google Cloud Functions, Google App Engine, and Google Compute Engine. The Compute Engine is what helps connect to the infrastructure such as the APIs and database. The visualisations can be performed by using Tableau.

A Flask-based Server:

For the compute engine to access the database a webhook call is sent to the Flask based server (that we build on Google Cloud Datalab) and that sends a datastore query to the database.

Database Layer:

The data from APIs is pre-processed in Google Cloud Datalab to create topics and is then extracted, and its associated synonym is saved for each topic. The resulting entities are stored in Cloud Datastore. Once, the data is available in the Datastore, the Flask-based server continuously accepts requests from the Dialogflow Fulfillment API and responds by using the data stored in the Datastore.

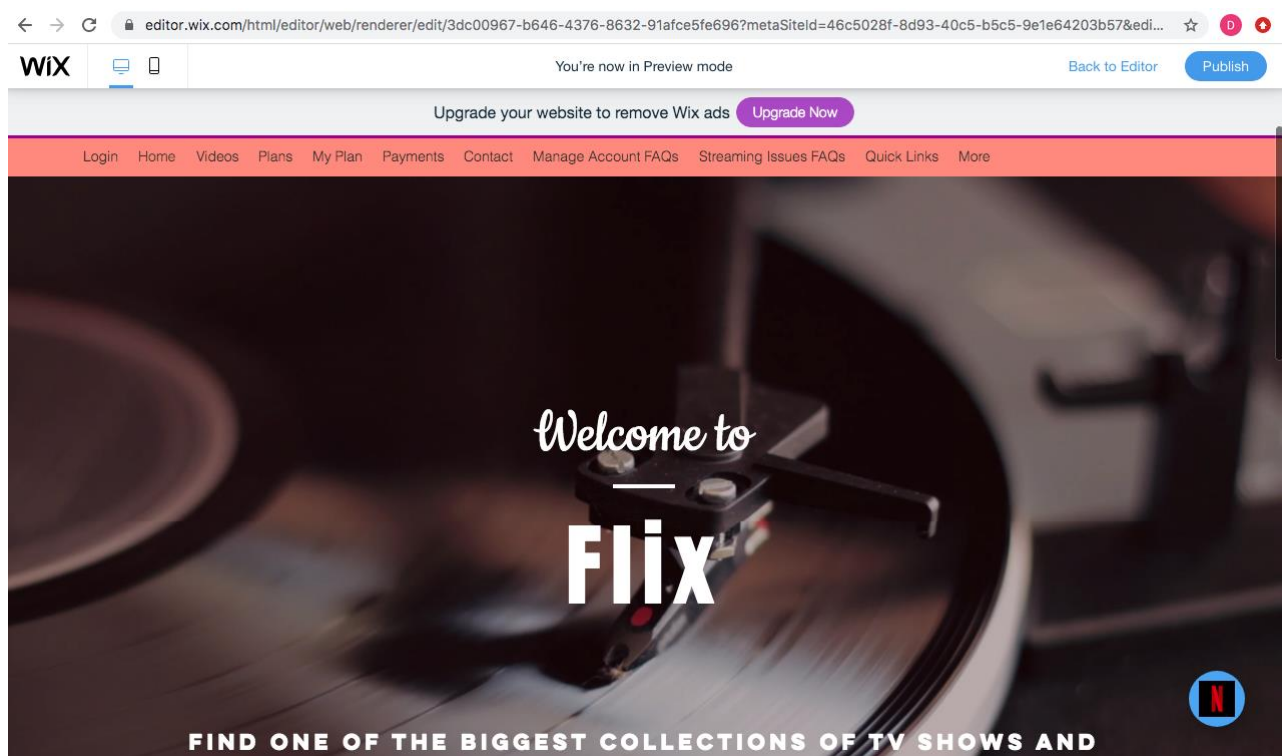
Infrastructure Layer:

The conversation is managed by Dialogflow, which is the main component of the interface as it enables the interaction using text and speech. The agent, which is created, using Dialogflow, manages the conversation flow based on the intent or intention extracted from the user conversation.

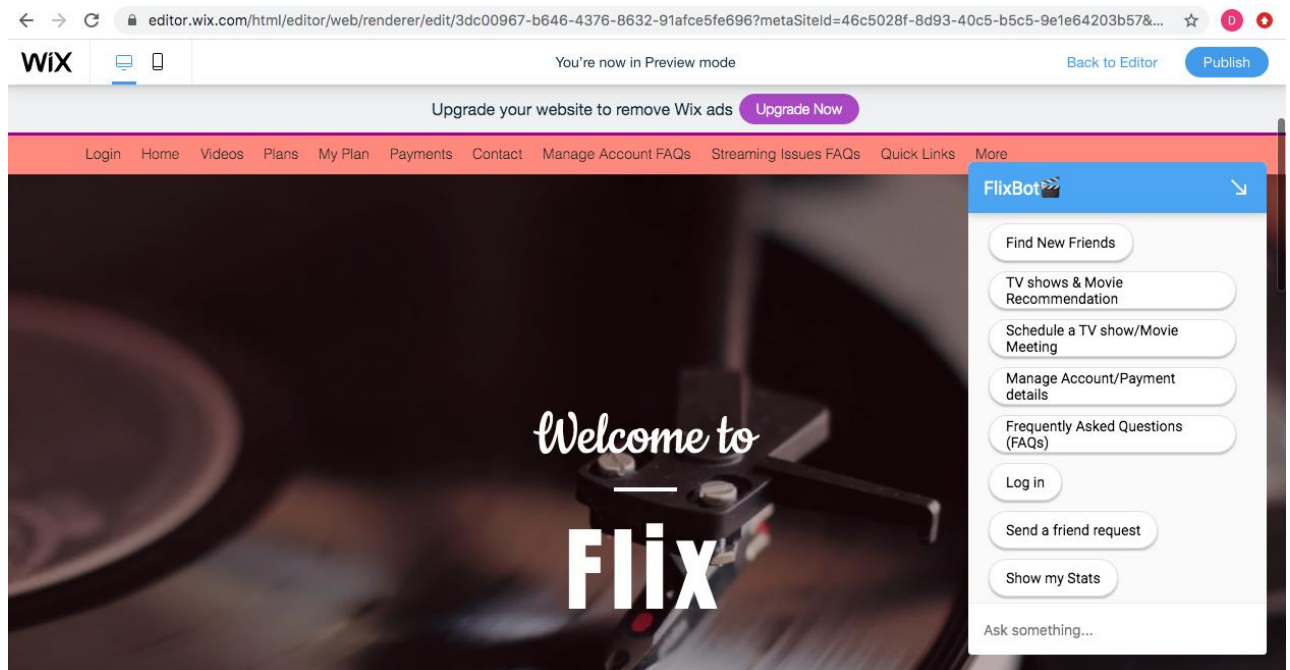
4. Key Features and Interface

The following images display what our chatbot would look like to a user once it's deployed on the web.

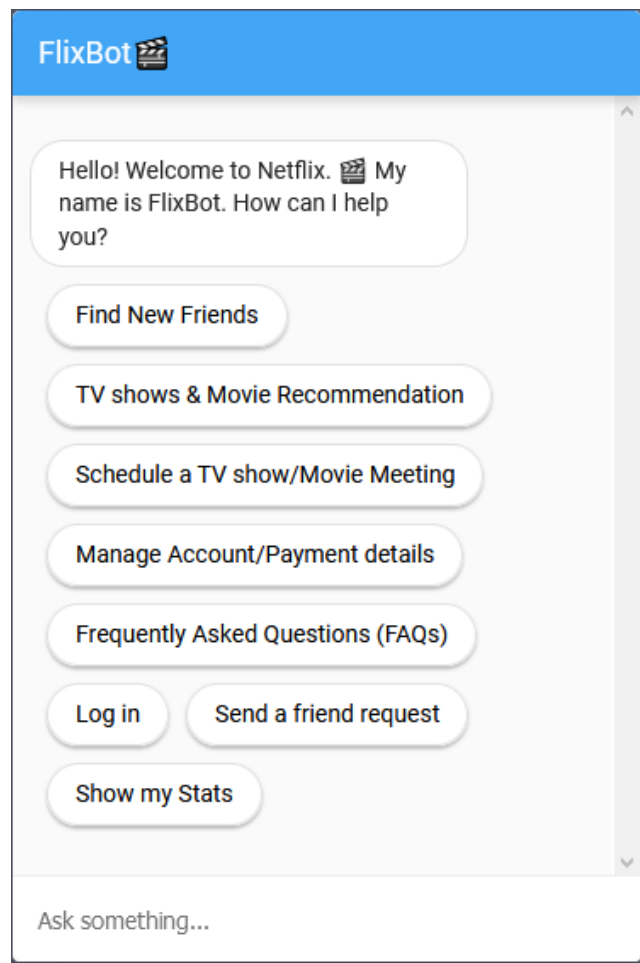
- The icon on the bottom right-hand corner of the website is our chatbot – Flixbot. The user can interact with the chatbot by clicking on that icon.



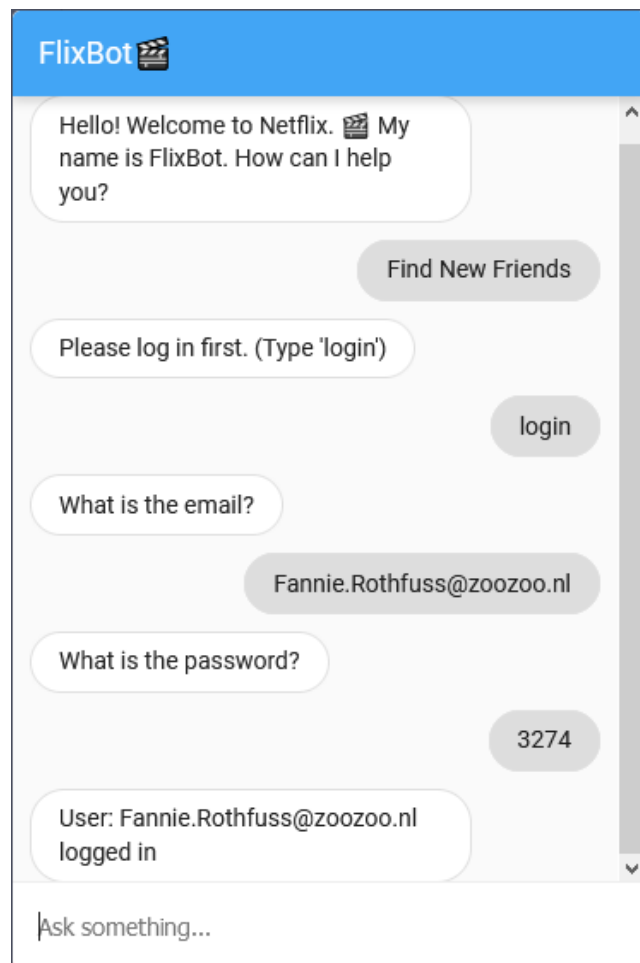
- When the user clicks on the icon, the chatbot starts a conversation with the user. The chatbot interface on the website is displayed in the image below.



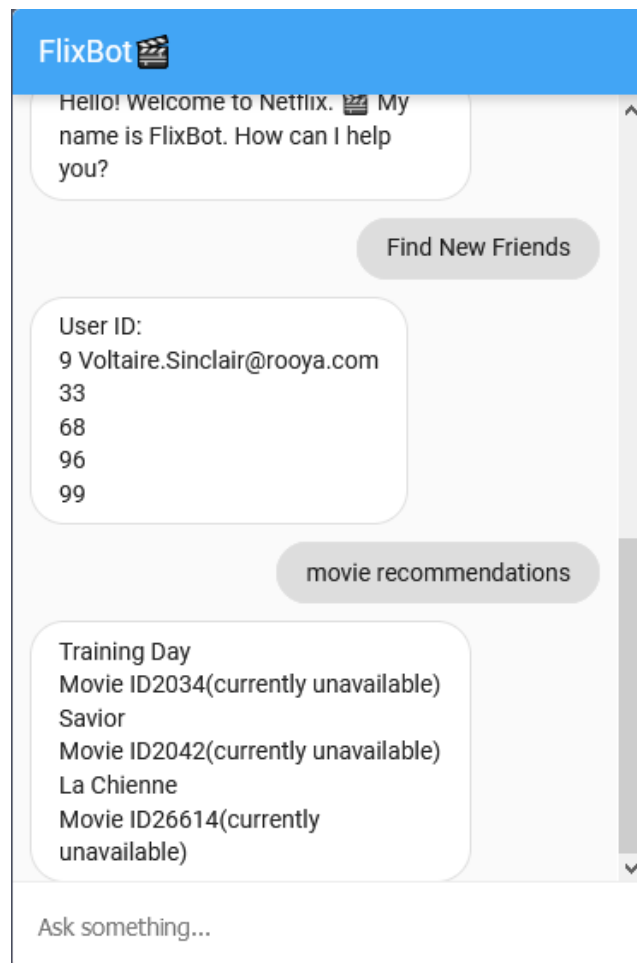
- The chatbot provides users with a list of clickable bubbles to access distinctive features as listed in the earlier sections with 8 goals. The clickable bubbles will send the corresponding text to the chatbot. In other words, chatbot also accepts text input.



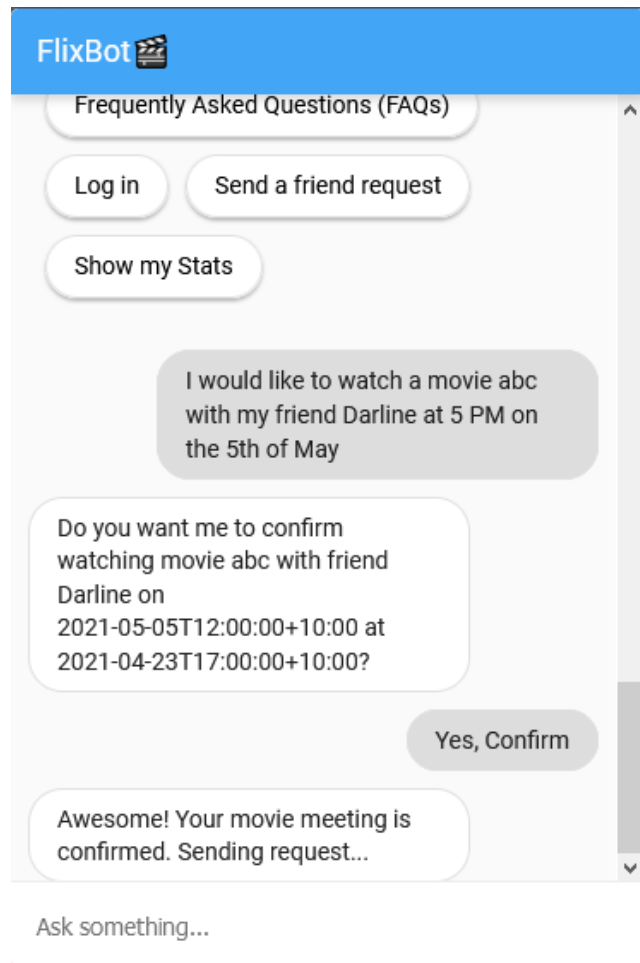
- It prompts user login when needed.



- It finds user with similar taste as the logged in user. It also recommends movies based on user history.



- It could send out meeting invites in emails to another user.



Fwd: pylCSParser invite20210505T170000Z ➤ [Inbox x](#)

chatbot google <chatbot9900@gmail.com> 12:20 PM (0 minutes ago) to me ▾

	<p>movie abc 20210505 @ 17:00</p> <p>When Thu May 6, 2021 3am – 5am (AEST)</p> <p>Who darline.amis@raale.nl, fannie.rothfuss@zoozoo.nl, organiser*</p> <p>Add to calendar »</p>	<p>Agenda</p> <p>Thu May 6, 2021</p> <p>No earlier events</p> <p>3am movie abc 20210505 @ 17:00</p> <p>No later events</p>
--	--	---

----- Forwarded message -----
 From: Chatbot9900 <chatbot9900@gmail.com>
 Date: Fri, 23 Apr 2021 at 12:16
 Subject: pylCSParser invite20210505T170000Z
 To: <chatbot9900@gmail.com>, <Fannie.Rothfuss@zoozoo.nl>, <Darline.Amis@raale.nl>

Chatroom url: <http://c801f39d04b7.ngrok.io/eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJyZXF1ZXN0ZXIiOiIyMDI1LTA0LTUzIDAyOjE2OjU0Lj0MDE0OCJ9.mW8ziFiT2QR3ASQwyCmYOvuhKFNFV>

If the meeting invite is not processed by the inbox of chatbot9900@gmail.com due to the none existing email created for the users, please try forwarding the email with the meeting invite to your Gmail account as shown above.

Flask_Chat_App

Harry Hello Stark

Stark Hello there!

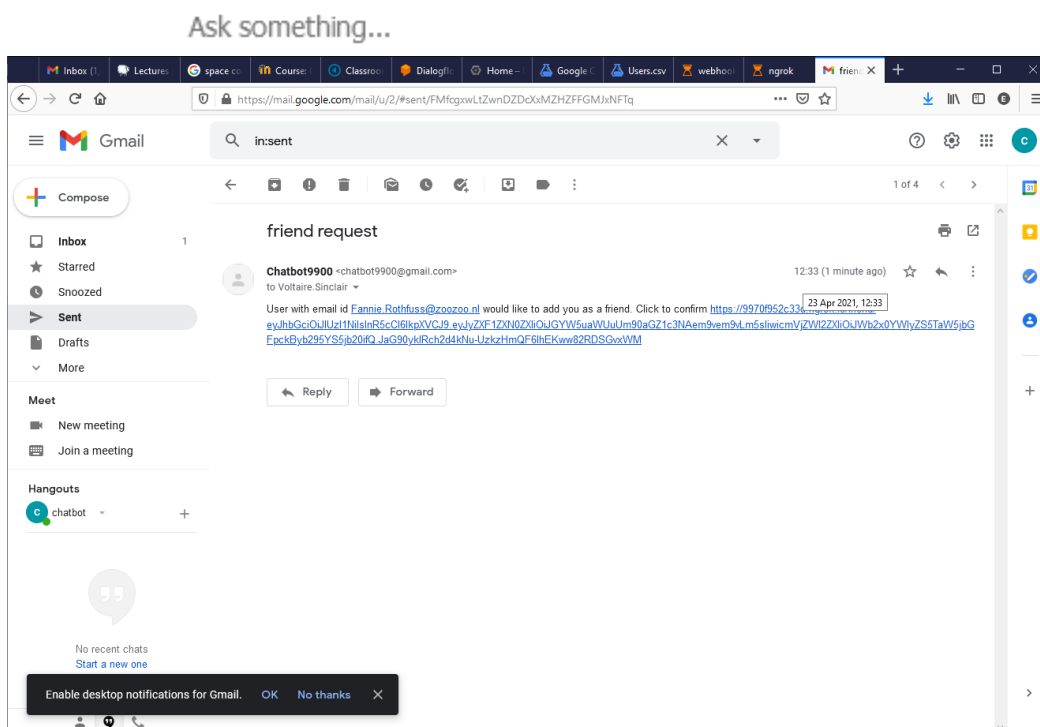
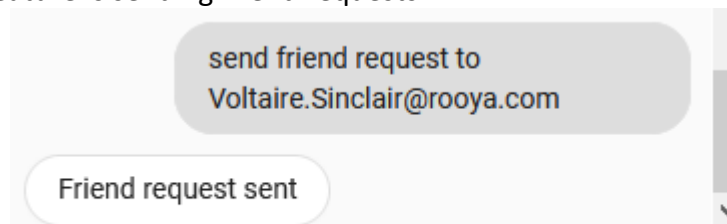
Harry

Messages

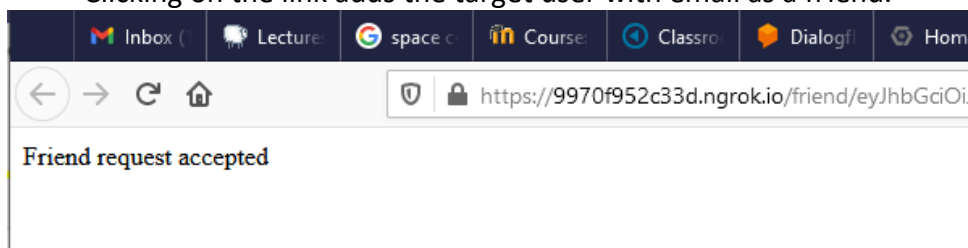
Submit

The URL within the email allows users to communicate in a simple chatroom.

- Other clickable bubbles supply links to a webpage to show potentials.
- A bonus feature is sending friend requests:



Clicking on the link adds the target user with email as a friend.



Some of our key features are explained below.

4.1. Web User Interface for Chatbot Interaction

This functionality will be the development of a UI or a web interface for a user to interact with the chatbot. User login will have an authentication and authorization mechanism to make sure “A user is valid – You are who you are, and you access what you are allowed to access”. Once authenticated, the chatbot will introduce itself to user and provide the user with some basic FAQs and present the user with relevant answers based on user choice. If the user is unable to get the necessary help with the FAQs, then the system will seamlessly transition to the smart chatbot.

4.2. Recommendation Module

4.2.1 KNN based Movie Recommendation

Assuming all the data are taken from the front end of our website, the tasks of the recommendation module should be providing recommendation results based on the user behavior stored in the user behavior dataset, also based on those users who are acting like the user.

Whenever a user logged in, recommendation module will take the user ID and calculate the personal results based on the user’s own interest. To meet the requirement of faster response in the backend of the project, we decide to use K nearest neighbor algorithm for the calculation.

4.2.2 Find New Friends

With the KNN approach mentioned above, we also developed a friend recommending function with the non-item-based version of the same algorithm. Like the movie recommending, it takes the user ID and gives back a few users that having similar taste to the user.

4.3. Auto-filling forms on behalf of the user

This functionality will focus on assisting a user with simple online updates. The chatbot provides links to services on a webpage which updates user profile and FAQs on the website.

4.4. Virtual Collaboration

This is a novel functionality that we want to develop, and we think will have a great potential in the future. The user can ask the chatbot to organize a virtual movie meeting with some of his/her friends. The chatbot will then send out meeting invites and based on a

simple Yes/No response from the user's friends will organize a virtual movie meeting collaboration with the group of users at a specified team. This will mean a group of friends can virtually watch the same movie at the same time and talk / comment about it from the comfort of their own couches at their home.

4.5. BI Module for Users and the Company

We intend to develop this module to data mine and run analytics on user activity and generate meaningful reports beneficial to the user and company. Reports can specify a user how much time he/she spends on the platform and on what type of content. This can help users take decision on the current plan – weather to upgrade or downgrade etc.

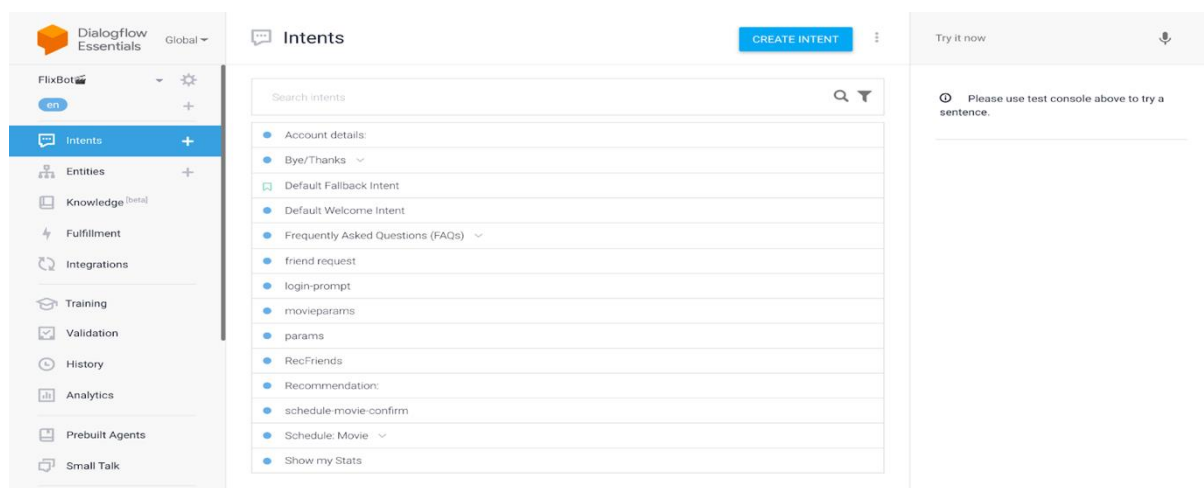
Reports can also benefit the company to understand what the popular contents are and what they need to focus on to improve their customer base etc.

5. Third Party Tools and their Utility in the Project

5.1. Dialogflow

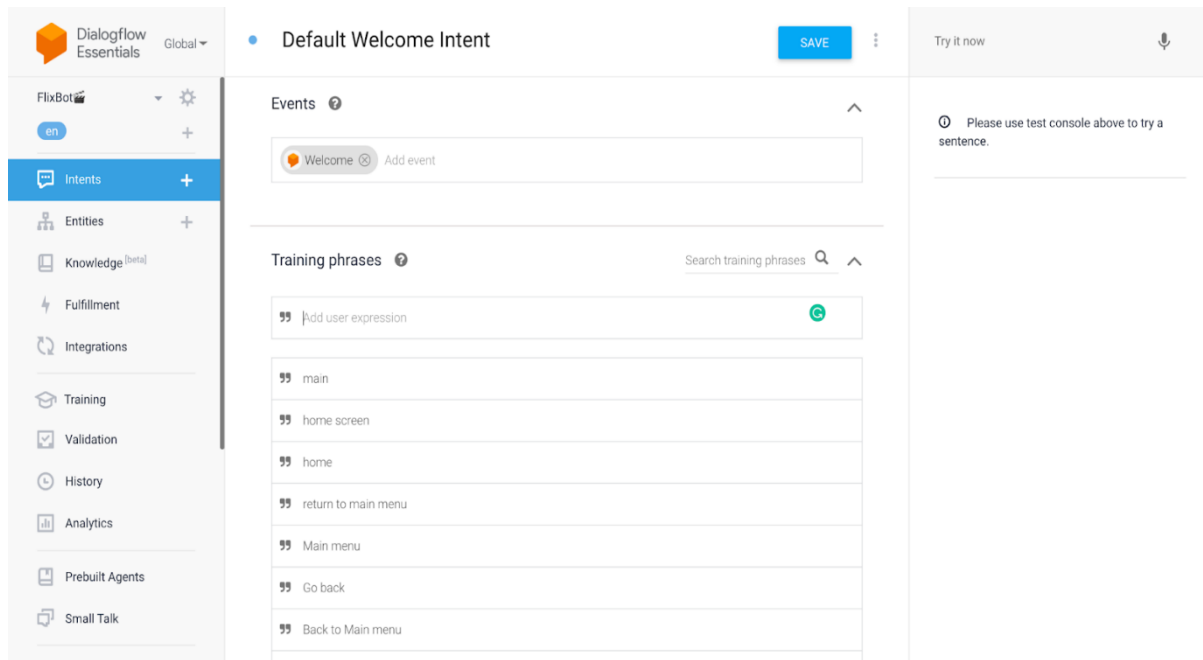
To create the chatbot we decided to use Google's Dialogflow. Its ease of integration with other Google Cloud Platform services such as Datalab, Datastore, compute engine along with its Natural Language Understanding characteristics, was the reason why we choose to use Dialogflow to design and create a conversational user interface to integrate it with our website.

Below you can see the chatbot's intent. An intent is what categorizes the user's intention or purpose.

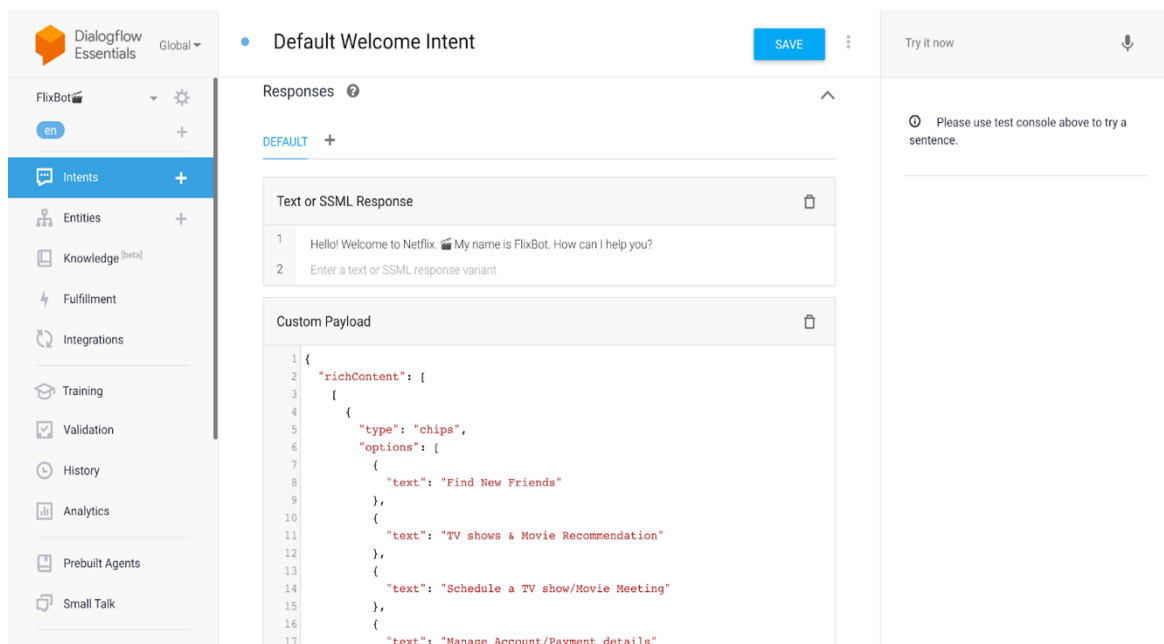


The intent contains:

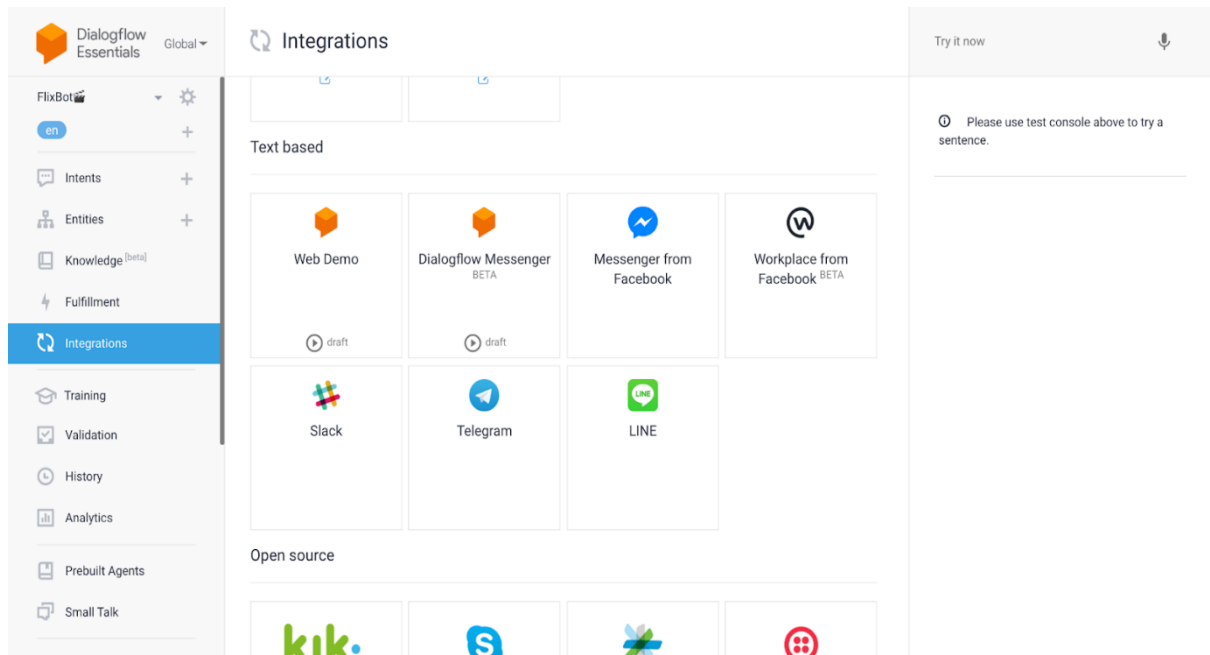
- Training phrases: Training phrases are sample phrases for what the users may type in the chatbot i.e. the input given by the chatbot. When the user's input matches one of these training phrases, Dialogflow recognises it as this intent.



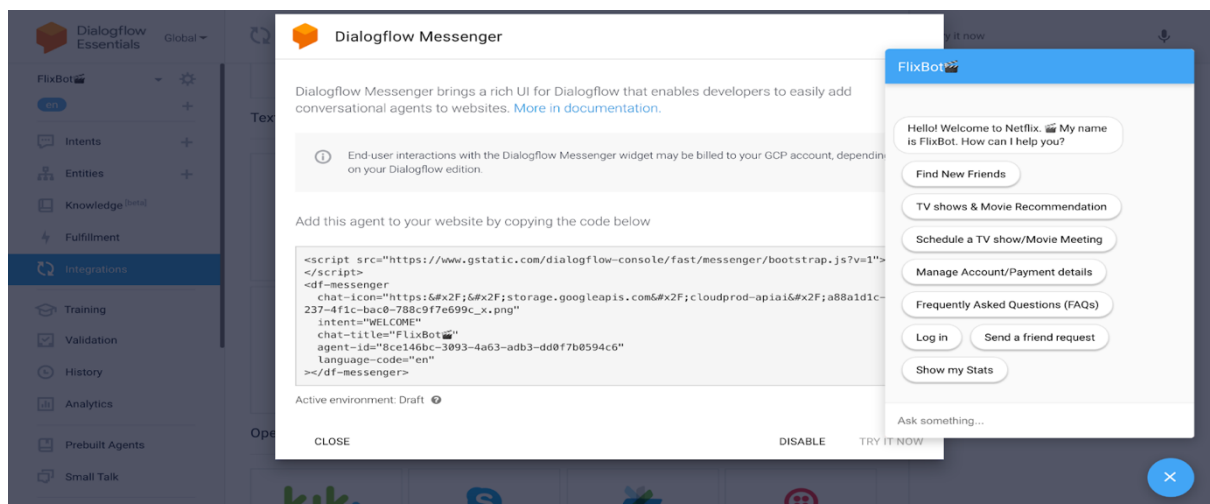
- Responses: For each intent we can decide the kind of response that we want to give to the user. These can either be in a text format or it can be a custom payload such as buttons, cards, images, etc.



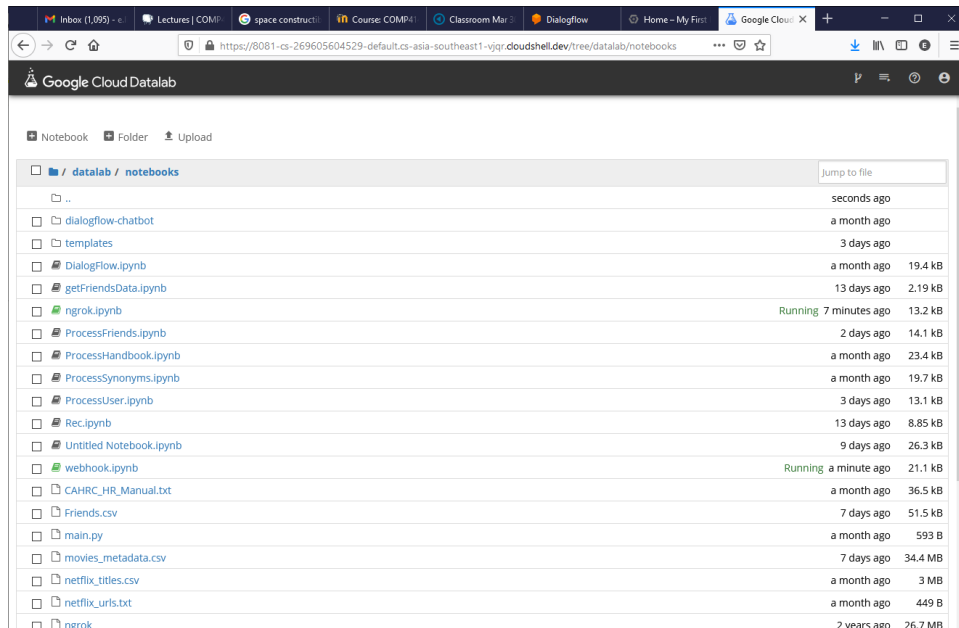
We then use integrations to test our chatbot. We have the options to connect our chatbot to Messenger, Slack, Telegram, etc., or view it as a website in Web Demo. But we have chosen Dialogflow Messenger as it allows us to use rich text such as buttons, cards, etc.



Here is a preview of how it looks while we are testing the chatbot.



5.2. Datalab

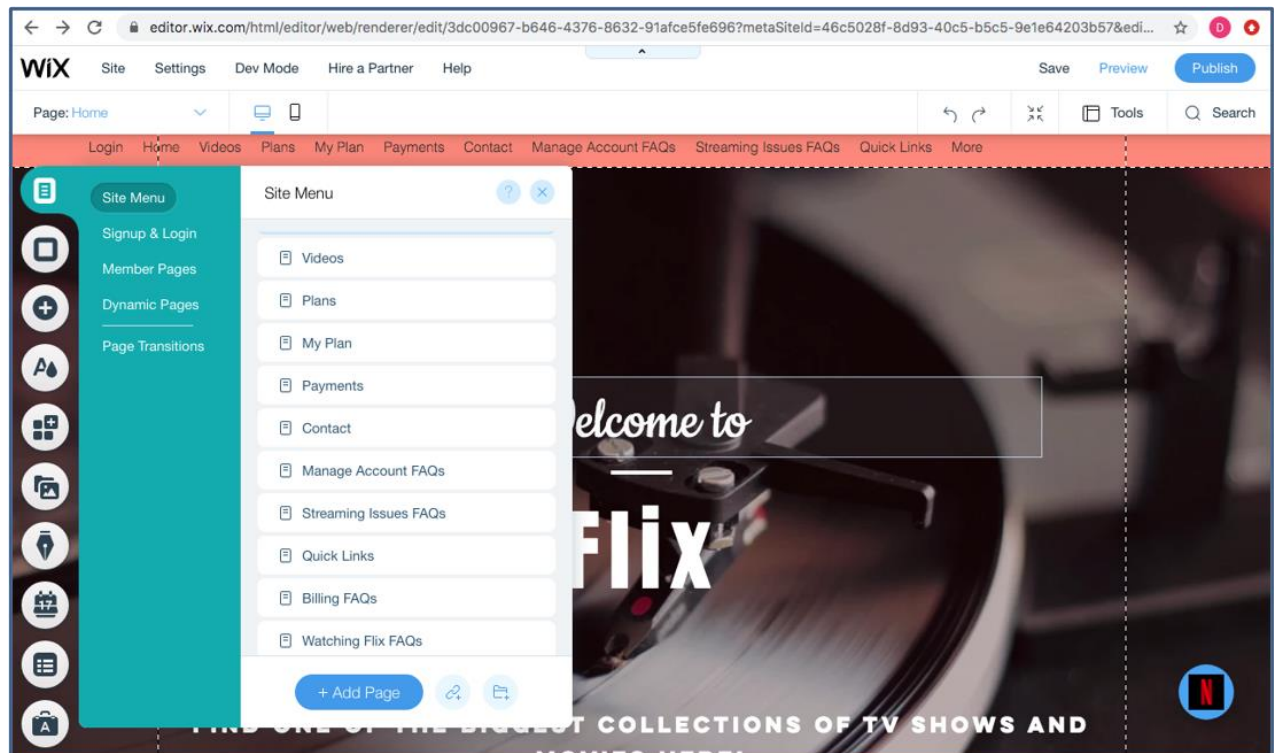


Datalab is a cloud backed notebook platform. It runs the webhook backend.

5.3. APIs

No API was used.

5.4. WIX Website Builder



WIX Website Editor was used to build a mockup website with the chatbot embedded. The website editor allowed us to:

- Create web pages
- Customise URLs of these web pages
- Customise the background of each webpage

- Add text and images on each webpage

5.5. Ngrok

```
Archive: ngrok-stable-linux-amd64.zip
replace ngrok? [y]es, [n]o, [A]ll, [N]one, [r]ename: ^C

1 ./ngrok http 5000

Running...

7- ngrok by @inconshreveable (Ctrl+C to quit)
Session Status      connecting          Version            2.3.35
Region              United States (us) Web Interface      http://127.0.0.1:4040
Connections
ttl    opn    rt1    rt5    p50    p90    0.00    0.00    0.00    0.00
Session Status      online            SsExpires1 hour, 59 minutesVrsion2.3.35      Rgio    United States (us) Web
Interface          http://127.0.0.1:4040
Connectionsttlopnr1 rt5 p5 p9
https://d22d90e90de2.ngrok.io -> http://localhost:
Connectionsttlopnr1 rt5 p5 p9
Update available (version 2.3.39, Ctrl-U to updateVrsion2.3.35      Rgio    United States (us) Web Interface:127.
0.1:4040      Forwarding      https://d22d90e90de2.ngrok.io -> http://localhost:
Connectionsttlopnr1 rt5 p5 p9
https://d22d90e90de2.ngrok.io -> http://localhost:5Forwarding      https://d22d90e90de2.ngrok.io -> http://localhost:
Connectionsttlopnr1 rt5 p5 p9
-----POST/webhook/200 OK10111      POST/webhook/200 OK200 OK203181      POST/webhook/200 OK200 OK301941      POST/webho
ok/200 OK500 INTERNAL SERVER ERROR401031543721      500 INTERNAL SERVER ERRORPOST/webhook/200 OK200 OK502432
6150
```

Ngrok was used for making the webhook reachable from the frontend. It was also used making the Flask chatroom service running on the local machine publicly available. Only free version is used. A paid version will generate perpetual public URL.

5.6. Dataflow

Datastore does not have a feature doing bulk removal of entities, so Dataflow must be used.

The screenshot displays the Google Cloud Platform Dataflow console. The main view shows the 'JOB GRAPH' for a job named 'delete kind 2'. The job is in a 'Succeeded' state, with a status of '2 of 2 stages succeeded'. The job graph consists of three stages:

- DatastoreCo...onEntities**: Succeeded, 7 sec, 2 of 2 stages succeeded.
- JavascriptT...Javascript**: Succeeded, 0 sec, 1 of 1 stage succeeded.
- DatastoreCo...EntityJson**: Succeeded, 19 sec, 1 of 1 stage succeeded.

The right-hand panel provides 'Job info' and 'Resource metrics'.

Job info	
Job name	delete kind 2
Job ID	2021-04-09_21_39_2 9-157218463515368 05261
Job type	Batch
Job status	Succeeded
SDK version	Apache Beam SDK for Java 2.27.0
Job region	australia-southeast1
Worker location	australia-southeast1-b
Current workers	0
Latest worker status	Worker pool stopped.
Start time	10 April 2021 at 14:39:30 GMT+10
Elapsed time	3 min 27 sec
Encryption type	Google-managed key

Resource metrics	
Current vCPUs	1
Total vCPU time	0.025 vCPU hr
Current memory	3.75 GB
Total memory time	0.096 GB hr

5.7. TABLEAU Integration

For creating visualizations, we have used Tableau, a data visualization software. We have created visualisations from the company's perspective (Business Intelligence Report) and the user's perspective (User Analytics). All these visualizations are presented on a webpage. The Business Intelligence Report cannot be accessed by the user and the chatbot. It is presented on a webpage that only the authorised staff of the company can access. As for the User Analytics, each user can see their own user analytics page from the link given by the chatbot after they have logged in.

For the Business Intelligence Report, we have used the 'movies_metadata.csv' file.

movies_metadata

Connection

Live

Extract

Filters

0

Add

movies_metadata.csv

Need more data?

Drag tables here to relate them. [Learn more](#)

Sort fields

Data source order

Show aliases

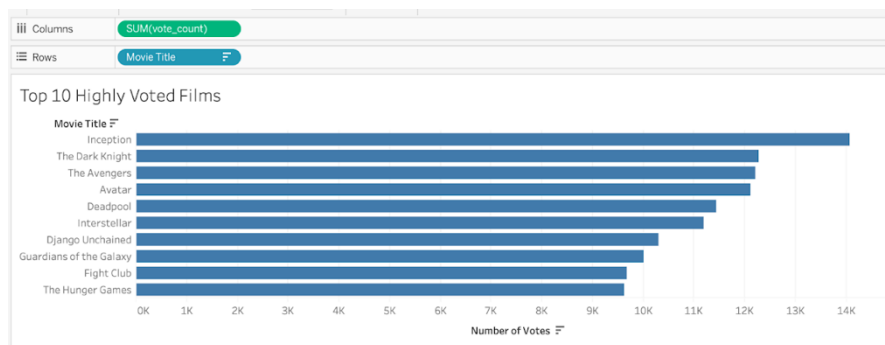
Show hidden fields

1,000

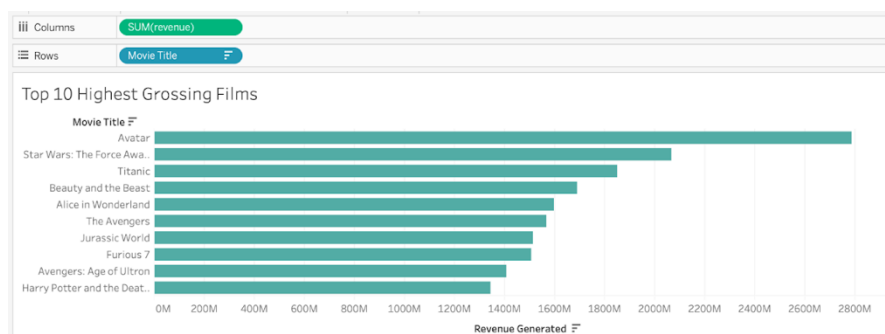
rows

release_date	revenue	runtime	spoken_languages	status	tagline	Movie Title	video	vote_average	vote_count
0/10/1995	373,554,033	81.000	[[{'iso_639_1': 'en', 'l...	Released	null	Toy Story	False	7.7000	5,415
5/12/1995	262,797,249	104.000	[[{'iso_639_1': 'en', 'l...	Released	Roll the dice and unle...	Jumanji	False	6.9000	2,413
2/12/1995	0	101.000	[[{'iso_639_1': 'en', 'l...	Released	Still Yelling. Still Figh...	Grumpier Old Men	False	6.5000	92
2/12/1995	81,452,156	127.000	[[{'iso_639_1': 'en', 'l...	Released	Friends are the peopl...	Waiting to Exhale	False	6.1000	34
0/2/1995	76,578,911	106.000	[[{'iso_639_1': 'en', 'l...	Released	Just When His World ...	Father of the Bride P...	False	5.7000	173
5/12/1995	187,436,818	170.000	[[{'iso_639_1': 'en', 'l...	Released	A Los Angeles Crime ...	Heat	False	7.7000	1,886
5/12/1995	0	127.000	[[{'iso_639_1': 'fr', 'n...	Released	You are cordially invi...	Sabrina	False	6.2000	141
2/12/1995	0	97.000	[[{'iso_639_1': 'en', 'l...	Released	The Original Bad Boys.	Tom and Huck	False	5.4000	45
2/12/1995	64,350,171	106.000	[[{'iso_639_1': 'en', 'l...	Released	Terror goes into over...	Sudden Death	False	5.5000	174
6/11/1995	352,194,034	130.000	[[{'iso_639_1': 'en', 'l...	Released	No limits. No fears. N...	GoldenEye	False	6.6000	1,194
7/11/1995	107,879,496	106.000	[[{'iso_639_1': 'en', 'l...	Released	Why can't the most p...	The American Presid...	False	6.5000	199
2/12/1995	0	88.000	[[{'iso_639_1': 'en', 'l...	Released	null	Dracula: Dead and Lo...	False	5.7000	210
2/12/1995	11,348,324	78.000	[[{'iso_639_1': 'en', 'l...	Released	Part Dog. Part Wolf. ...	Balto	False	7.1000	423

For the top 10 highly voted films we have arranged 10 movie titles having the highest number of total 'vote_count' in descending order.



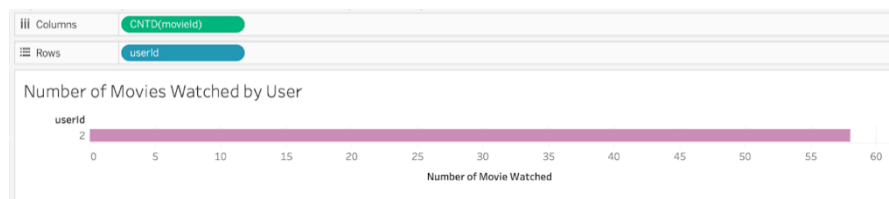
For the top 10 highest grossing films we have arranged 10 movie titles having the highest number of total 'revenue' in descending order.



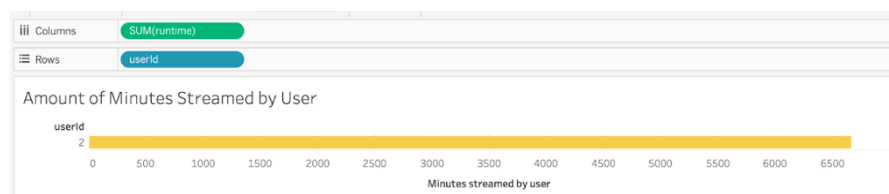
For the User Analytics Report, we have left joined 'ratings_small.csv' and 'movies_metadata.csv' based on the movie id, and we have hidden all the unnecessary columns.

#	#	#	Abc	#	#	#	#
movies_m...	movies_metadat...	movies_metadat...	movies_metadata.csv	user_2.csv	user_2.csv	user_2.csv	user_2.csv
id	revenue	runtime	Movie Title	userid	movieid	rating	timestamp
319	12,281,551	120.000	True Romance	2	319	1.00000	835,355,918
62	68,700,000	149.000	2001: A Space Odyssey	2	62	3.00000	835,355,749
223	6,000,000	130.000	Rebecca	2	223	1.00000	835,355,749
261	17,570,324	108.000	Cat on a Hot Tin Roof	2	261	4.00000	835,355,681
454	147,298,761	120.000	Romeo + Juliet	2	454	4.00000	835,355,604
500	14,661,007	99.000	Reservoir Dogs	2	500	4.00000	835,355,731
185	26,589,000	136.000	A Clockwork Orange	2	185	3.00000	835,355,511
144	3,200,000	128.000	Wings of Desire	2	144	3.00000	835,356,016
539	32,000,000	109.000	Psycho	2	539	3.00000	835,355,767
235	52,287,414	89.000	Stand by Me	2	235	3.00000	835,355,664
339	2,015,810	129.000	Night on Earth	2	339	3.00000	835,355,492
377	25,504,513	91.000	A Nightmare on Elm ...	2	377	3.00000	835,355,710
168	133,000,000	119.000	Star Trek IV: The Voy...	2	168	3.00000	835,355,710

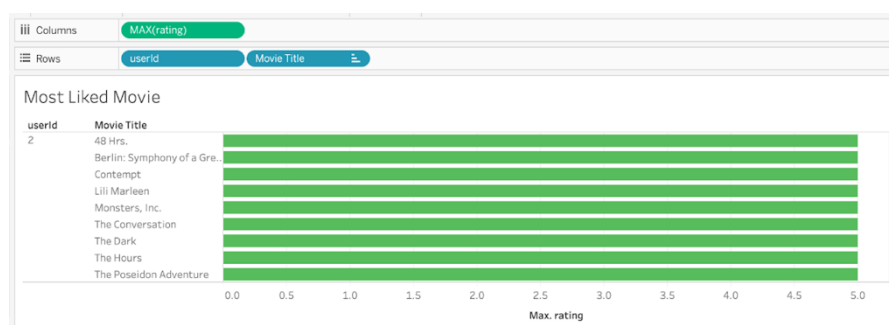
For the number of movies watched by the user, for each user we have counted the distinct number of movie ids of all the movies that the user has watched.



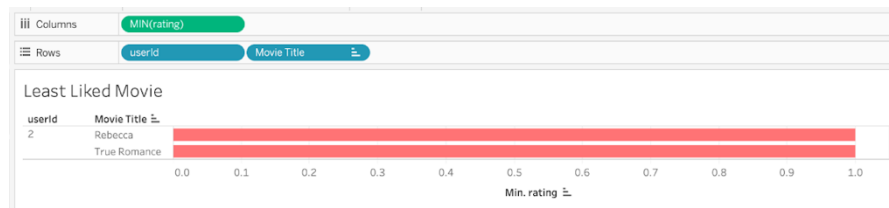
For the number of minutes streamed, for each user we have calculated the sum of the runtime of all the movies that the user has watched.



For the most liked movie, for each user we have found out all the movie titles for which the user has given the highest/maximum rating.



For the least liked movie, for each user we have found out all the movie titles for which the user has given the lowest/minimum rating.



6. Implementation Challenges

6.1. Recommendation Module Algorithm

In the recommendation part, we decided to implement it in the machine learning way. To do this, we found some useful datasets from Kaggle platform, and generated some mock-up data by Python.

Considering the data of the user behavior can be sparse, we tried to use the Matrix Factorization-based algorithms at beginning, like the famous SVD algorithm or the unbiased one, PMF (Probabilistic Matrix Factorization).

But, when the user experience is provided by the GCP, reaction speed can be critical. So, the way of recommendation was changed to a similarity approach, and it is also efficient, which is KNN approach. So, the basic theory is calculating the similarity (distance) between an item and other items, and rank them based on how similar they are, take first K items and adding up the weighted rating based on the similarities, and normalization on the result.

There is a function implemented to take a user's favorite movie that recently viewed and give back a list of five movies that matching the user's taste. It can be used to calculate the similar users as well.

During the integration, we found the chatbot has a time limit when passing data back to the front-end. Which means, the calculation in the recommendation module still took too much time, and it will return nothing when the user clicking on it.

To give back the recommendation results immediately, we came up with a solution that can perfectly solve this problem:

Whenever a user logging in, run the recommendation function in another thread right after we got the ID of the user, and store the results at the back end. In this way, the results can be accessed anytime, and not intervening the log in process.

6.2. Movie Scheduler and Chat Feature

Movie scheduler was implemented as a function in the webhook invoked by an intent in the frontend. If the target user is a friend of the logged in user, then a jwt string will be generated as the chatroom ID. This link along with an invitation will be sent by a Gmail account 'chatbot9900@gmail.com' to the logged in user and the target user. The link to the chatroom destinations to the local chatroom service made available by Ngrok. This is the reason that the public URL generated by the local Ngrok must be updated in the webhook. The chatroom service is a Flask app using Socketio. The form in the html page has a hidden rid which will be appended to the message emitted. This allows a chatroom only displaying messages sent within the room.

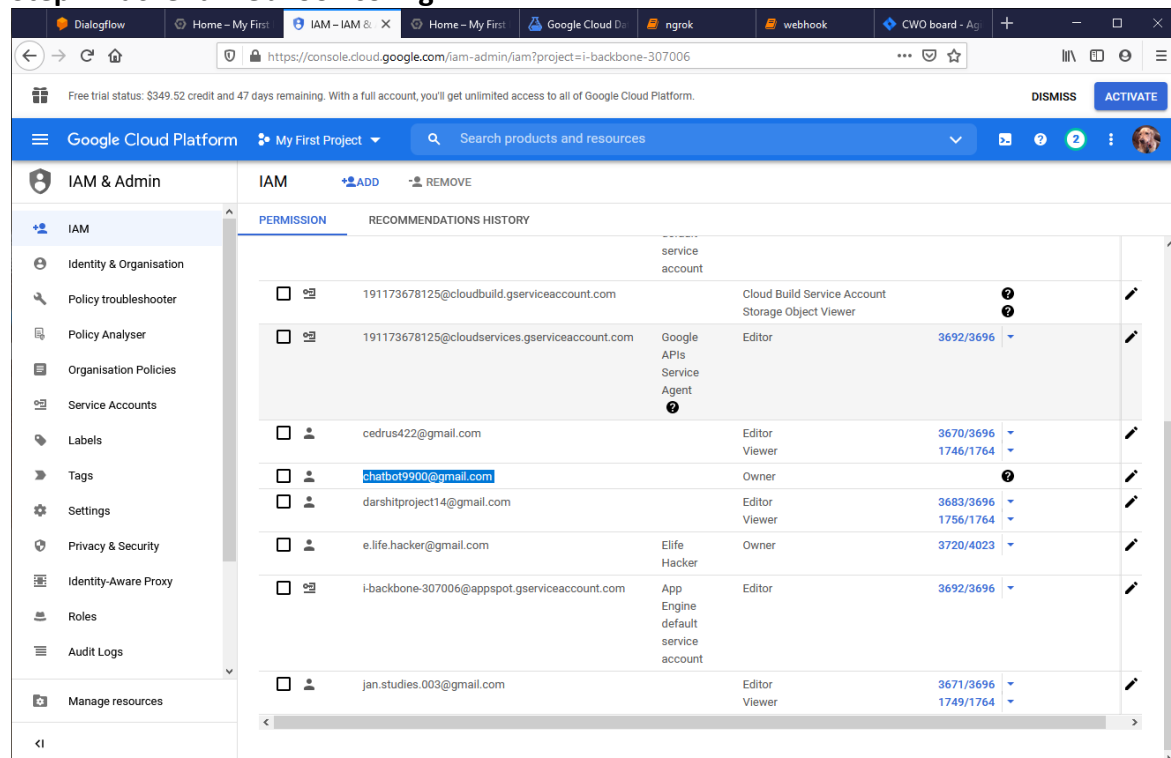
It was considered to run the chatroom service within the Datalab in the webhook, but there was compatibility issue that libraries to recognize TCP and UDP protocol could not be installed in Datalab. It was also tried to deploy the app to Google App Engine. The deployed app for some reasons does not display chat messages. Troubleshooting the app on the cloud requires more effort. It was settled to use Ngrok the locally deployed app which works fine.

7. Installation Instruction

The following steps were building on the documentation provided by Google. It is not necessary to follow it unless rebuilding the environment from scratch. Instead, an account has been created and granted permission to the cloud platform for the ease of accessing our work.

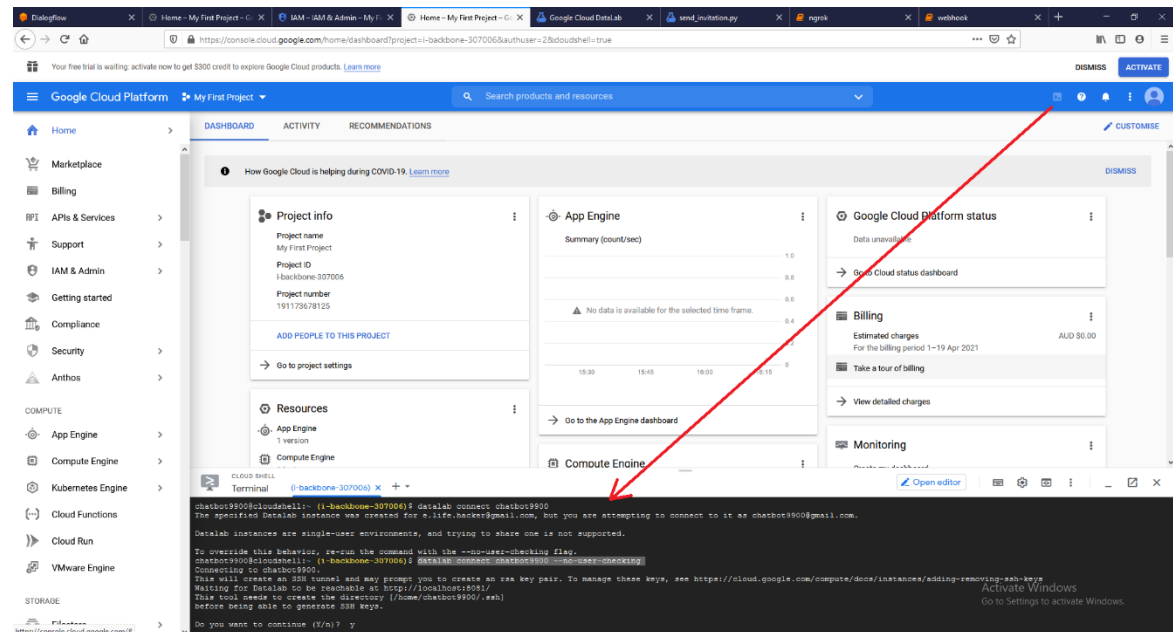
<https://cloud.google.com/solutions/building-chatbot-agent-dialogflow>

Step 1 Backend webhook config



The setup was done on the cloud. Please login to GCP as chatbot9900@gmail.com with password Chat#9900

This account has been granted ownership role to the dev environment of one of the team members as shown above.



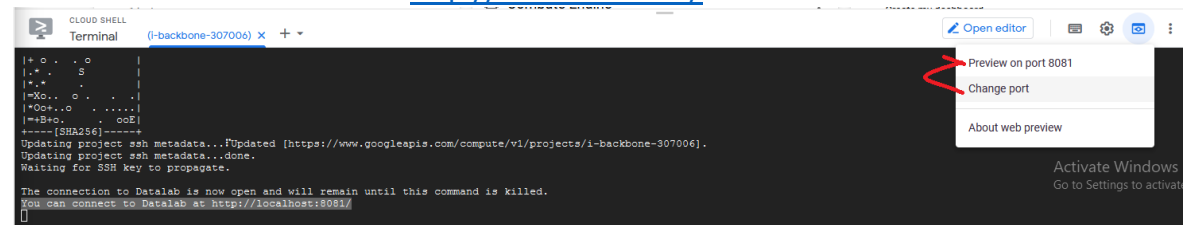
Then click on the button in the screenshot above. A terminal will be created. Run the command:

Datalab connect chatbot9900 --no-user-checking

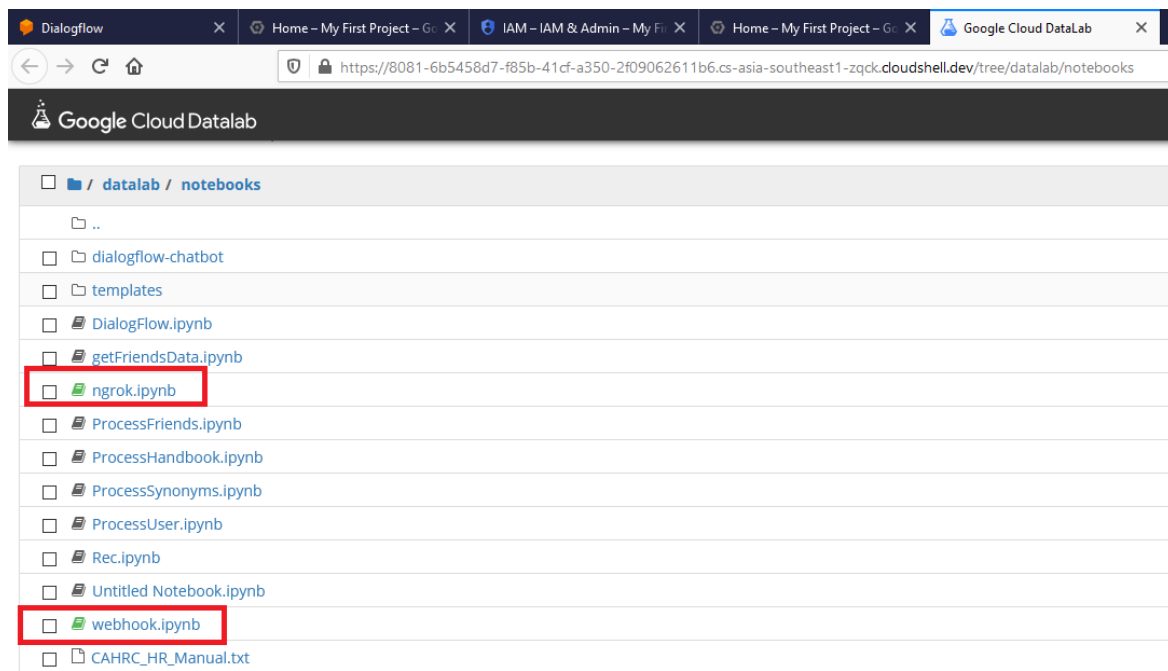
To start the Datalab.

Once below message is printed in the terminal, change the port to 8081 and Review.

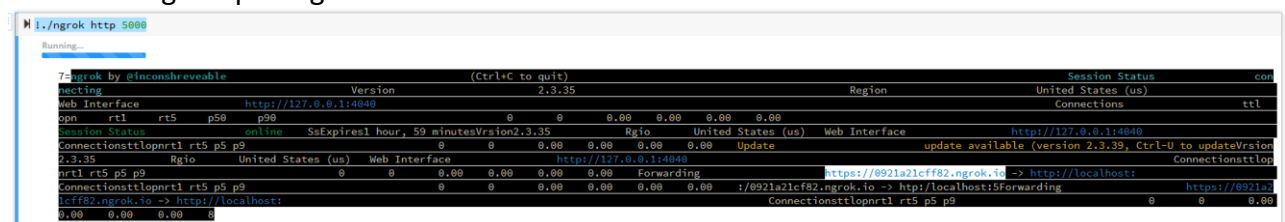
You can connect to Datalab at <http://localhost:8081/>



Two notebooks are required to be executed:

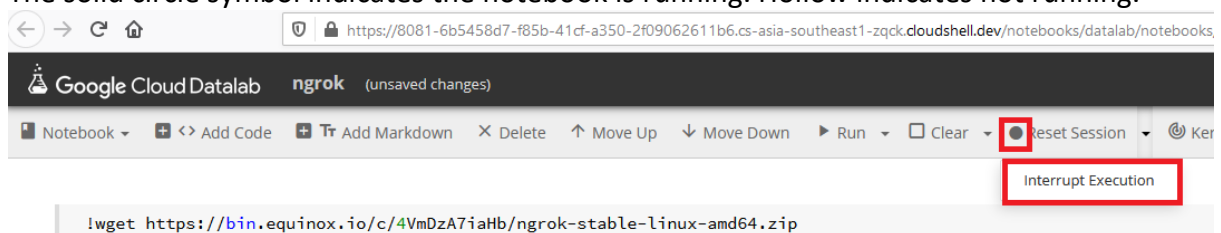


Ngrok is used to make the backend webhook reachable by the front end. Please run the third block with command '!./ngrok http 5000' only, because the first two are for downloading the package.

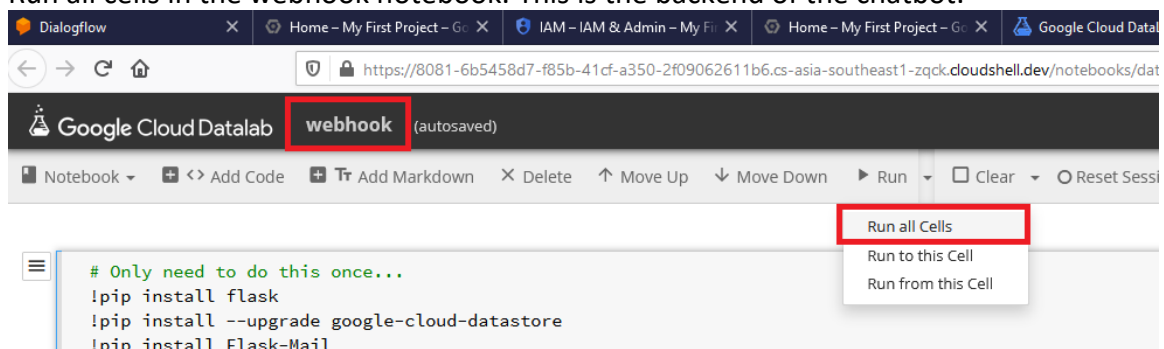


Noted down the URL created. In this case it would be <https://0921a21cff82.ngrok.io>. This will be used later.

If the first two blocks are executed by accident, you may need to interrupt the execution. The solid circle symbol indicates the notebook is running. Hollow indicates not running.



Run all cells in the webhook notebook. This is the backend of the chatbot.

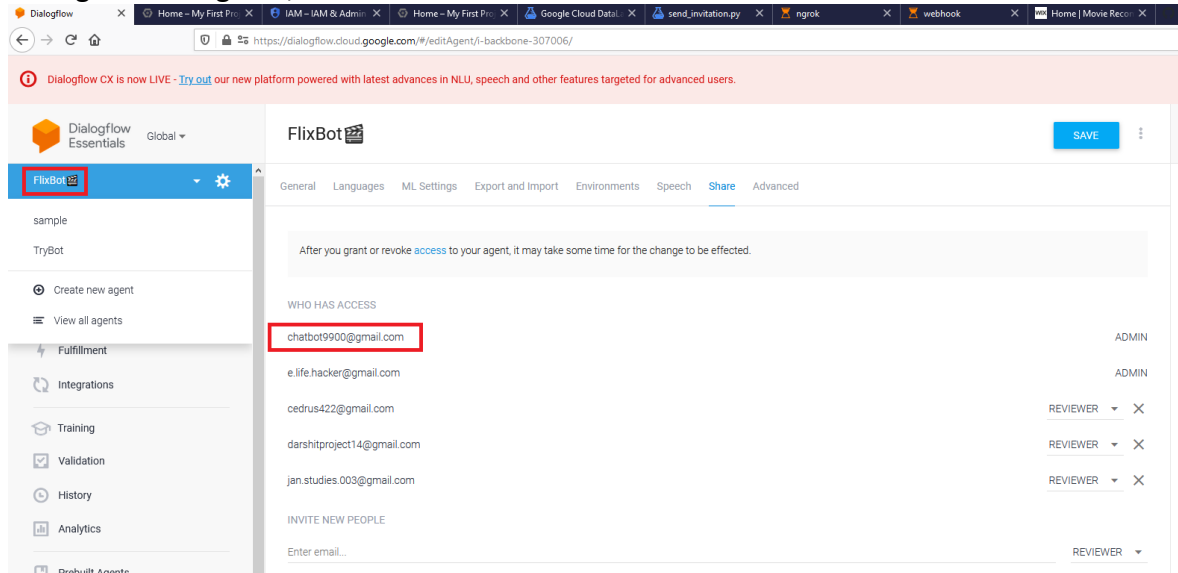


Once below message is shown on the bottom, it also indicates the service is running.

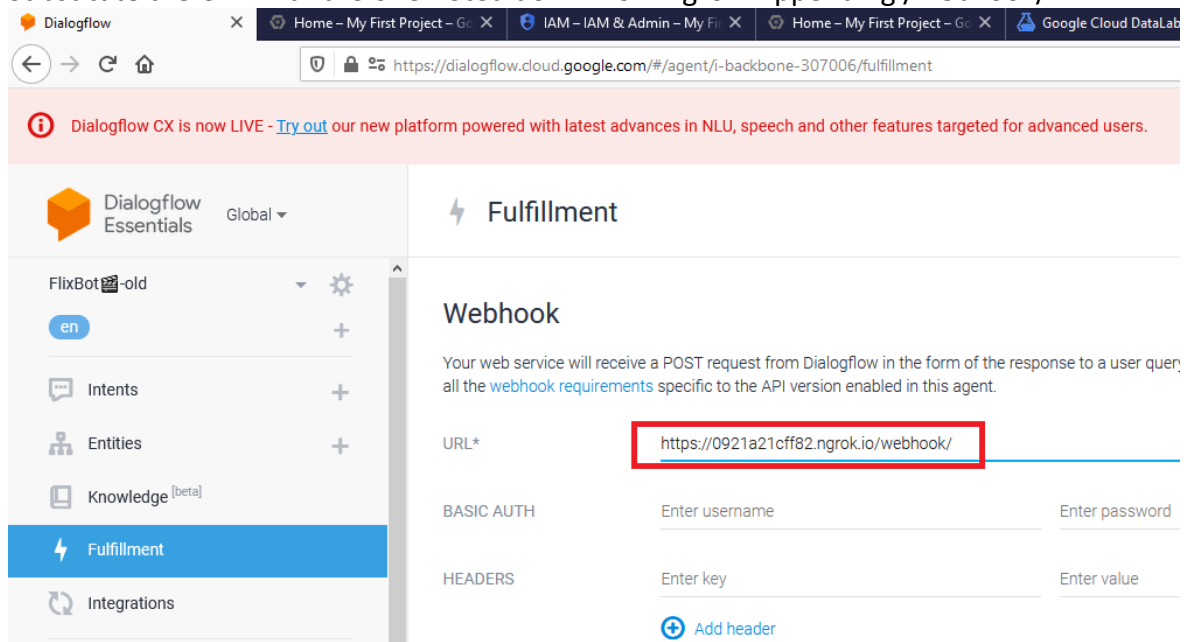
```
Running...
/usr/local/envs/py3env/lib/python3.5/site-packages/IPython/core/interactiveshell.py:2785: DtypeWarning: Columns (10) have mixed types. Specify dtype option on import or set low_memory=False.
interactivity=interactivity, compiler=compiler, result=result)
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
```

Step 2 Frontend Dialogflow Config

Now go to Dialogflow, find Flixbot which also has the account set as an admin.



Substitute the URL with the one noted down from ngrok. Appending /webhook/

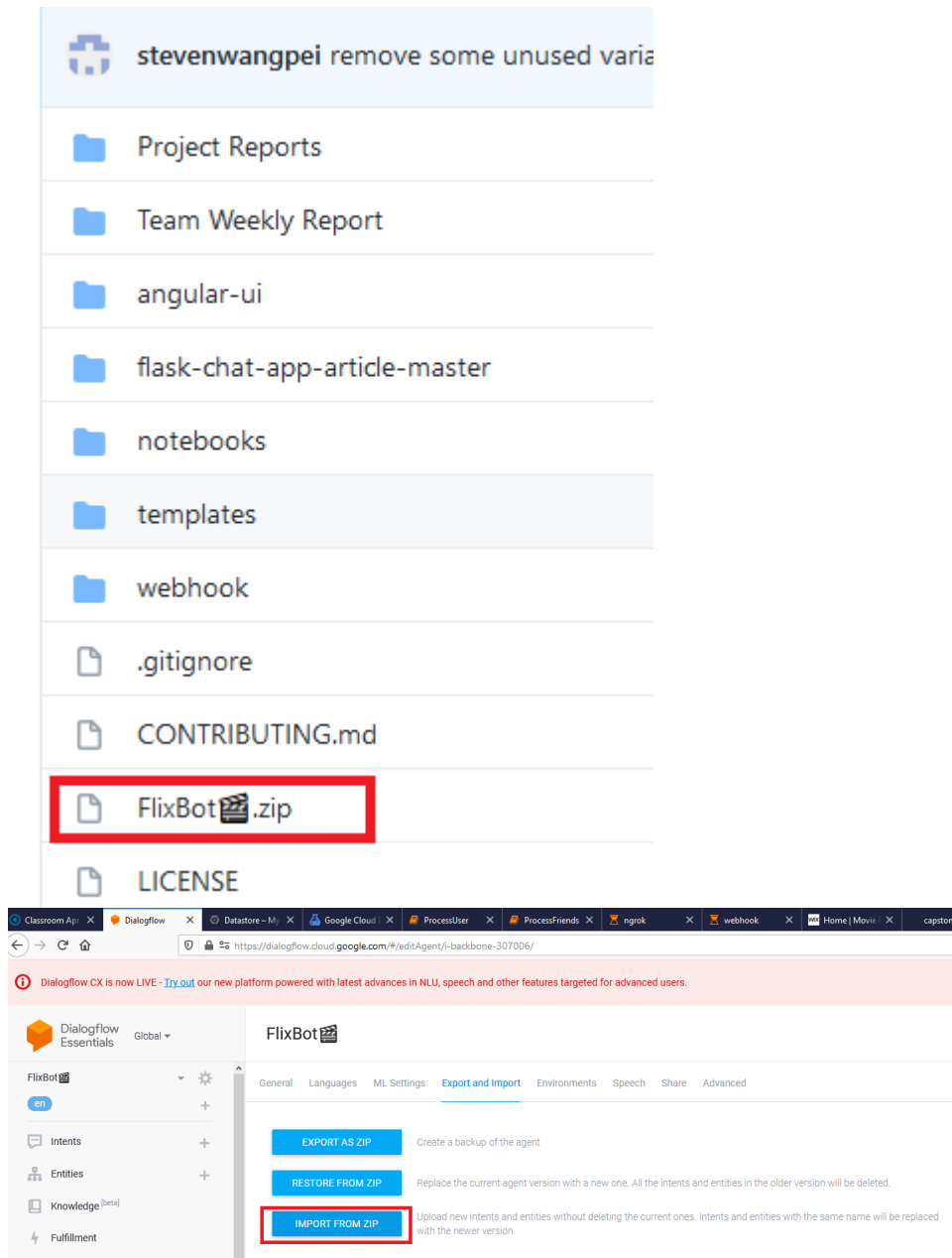


Click on Save on the bottom of the page.

Now the chatbot backend is online. You may use it in the link below.

<https://cedrus422.wixsite.com/website>

Should one want to setup this chatbot from scratch, please use the function below to import the zip.



Step 3 Chatroom service

This step is for the chatroom feature only. It is not required to run the chatbot. Basically, you need to run a flask service and make it public using ngrok.

https://github.com/uns-w-cse-comp3900-9900-21T1/capstone-project-9900-w18a-outerheaven		
master	5 branches	0 tags
Go to file	Add file	Code
JacobGeng2020 Merge branch 'master' of https://github.com/uns-w-cse-comp3900-9... f16ed46 20 hours ago 68 commits		
Project Reports	Project Docs	13 days ago
Team Weekly Report	Test update from prompt	20 hours ago
angular-ui	google sample code	last month
dialogflow	integrate with Janvi's interface	17 days ago
flask-chat-app-article-master	virtual collab	last month
notebooks	Add files via upload	23 hours ago
templates	allow sending friends request and update datastore	6 days ago
webhook	google sample code	last month
.gitignore	google sample code	last month
CONTRIBUTING.md	google sample code	last month
E_Bot.zip	allow sending friends request and update datastore	6 days ago
FlixBot.zip	Add files via upload	23 hours ago
LICENSE	google sample code	last month
README.md	update readme and weekly report	last month
Recipyb	Add files via upload	14 days ago
datastore1.png	integrate with Janvi's interface	17 days ago
datastore2.png	integrate with Janvi's interface	17 days ago
getFriendsData.ipynb	Add files via upload	14 days ago
ngrok-stable-windows-amd64.zip	upload window ngrok	17 days ago

Download these two files to your local windows machine. Now move to where flask-chat-app-article-master is located run the following command to install the necessary package.
(base) F:\unsw2021\9900\flask-chat-app-article-master>pip3 install -r requirements.txt

```

Anaconda Prompt (anaconda3)

(base) F:\unsw2021\9900\flask-chat-app-article-master>pip3 install -r requirements.txt
Requirement already satisfied: Flask==0.12.2 in f:\anaconda3\lib\site-packages (from -r requirements.txt (line 1)) (0.12.2)
Requirement already satisfied: flask-socketio==4.3.2 in f:\anaconda3\lib\site-packages (from -r requirements.txt (line 2)) (4.3.2)
Requirement already satisfied: eventlet==0.17.4 in f:\anaconda3\lib\site-packages (from -r requirements.txt (line 3)) (0.17.4)
Requirement already satisfied: gunicorn==18.0.0 in f:\anaconda3\lib\site-packages (from -r requirements.txt (line 4)) (18.0.0)
Requirement already satisfied: click>=2.0 in f:\anaconda3\lib\site-packages (from Flask==0.12.2->-r requirements.txt (line 1)) (7.1.2)
Requirement already satisfied: itsdangerous>=0.21 in f:\anaconda3\lib\site-packages (from Flask==0.12.2->-r requirements.txt (line 1)) (1.1.0)
Requirement already satisfied: Werkzeug>=0.7 in f:\anaconda3\lib\site-packages (from Flask==0.12.2->-r requirements.txt (line 1)) (1.0.1)
Requirement already satisfied: Jinja2>=2.4 in f:\anaconda3\lib\site-packages (from Flask==0.12.2->-r requirements.txt (line 1)) (2.11.2)
Requirement already satisfied: python-socketio<5,>=4.3.0 in f:\anaconda3\lib\site-packages (from flask-socketio==4.3.2->-r requirements.txt (line 2)) (4.6.1)
Requirement already satisfied: greenlet>=0.3 in f:\anaconda3\lib\site-packages (from eventlet==0.17.4->-r requirements.txt (line 3)) (0.4.17)
Requirement already satisfied: MarkupSafe>=0.23 in f:\anaconda3\lib\site-packages (from Jinja2>=2.4->Flask==0.12.2->-r requirements.txt (line 1)) (1.1.1)
Requirement already satisfied: python-engineio<4,>=3.13.0 in f:\anaconda3\lib\site-packages (from python-socketio<5,>=4.3.0->flask-socketio==4.3.2->-r requirements.txt (line 2)) (3.14.2)
Requirement already satisfied: six>=1.9.0 in f:\anaconda3\lib\site-packages (from python-socketio<5,>=4.3.0->flask-socketio==4.3.2->-r requirements.txt (line 2)) (1.15.0)
(base) F:\unsw2021\9900\flask-chat-app-article-master>pip3 install -r requirements.txt

```

Then run python main.py

```
Anaconda Prompt (anaconda3) - python main.py

(base) F:\unsw2021\9900\flask-chat-app-article-master>python main.py
* Restarting with windowsapi reloader
* Debugger is active!
* Debugger PIN: 272-833-906
* Detected change in 'F:\anaconda3\lib\site-packages\eventlet\hubs\_pycache\_queue.cpython-38.pyc', reloading
* Detected change in 'F:\anaconda3\lib\site-packages\eventlet\hubs\_pycache\_hub.cpython-38.pyc', reloading
* Detected change in 'F:\anaconda3\lib\site-packages\eventlet\hubs\_pycache\_selects.cpython-38.pyc', reloading
(4272) wsgi starting up on http://127.0.0.1:5000/
* Restarting with windowsapi reloader
* Debugger is active!
* Debugger PIN: 272-833-906
(5544) wsgi starting up on http://127.0.0.1:5000/
```

Notice the port being used is 5000.

Now run ngrok on port 5000 with
ngrok http 5000

```
F:\downloads\ngrok-stable-windows-amd64\ngrok.exe

EXAMPLES:
ngrok http 80 # secure public URL for port 80 web server
ngrok http -subdomain=baz 8080 # port 8080 available at baz.ngrok.io
ngrok http foo.dev:80 # tunnel to host:port instead of localhost
ngrok http https://localhost # expose a local https server
ngrok tcp 22 # tunnel arbitrary TCP traffic to port 22
ngrok tls -hostname=foo.com 443 # TLS traffic for foo.com to port 443
ngrok start foo bar baz # start tunnels from the configuration file

VERSION:
2.3.35

AUTHOR:
inconshreveable - <alan@ngrok.com>

COMMANDS:
authtoken save authtoken to configuration file
credits prints author and licensing information
http start an HTTP tunnel
start start tunnels by name from the configuration file
tcp start a TCP tunnel
tls start a TLS tunnel
update update ngrok to the latest version
version print the version string
help Shows a list of commands or help for one command

ngrok is a command line application, try typing 'ngrok.exe http 80'
at this terminal prompt to expose port 80
F:\downloads\ngrok-stable-windows-amd64>ngrok http 5000
```

Then copy the http address (not https) and replace the link in the get_chatroom_url() function in the webhook in Datalab.

The screenshot shows a Google Cloud Datalab notebook interface. At the top, a terminal window displays the ngrok service status, indicating it is online and forwarding traffic from a public URL to localhost:5000. Below the terminal, the notebook code editor contains a Python script. The script defines a function `get_chatroom_url` that returns a public URL from the ngrok service. The URL `"http://d5a7fa138cf2.ngrok.io"` is highlighted with a red box. An "Interrupt Execution" button is located in the top right corner of the code block.

```
from datetime import datetime

seed(datetime.now())

app.config['SECRET_KEY'] = 'SECRET_KEY_CHAT9900'

file_path = os.path.expanduser('ratings_small.csv')
movie_path = "movies_metadata.csv"
# friend_path = "friends.csv"

reader = Reader(line_format='user item rating timestamp', sep=',', skip_lines=1)
surprise_data = Dataset.load_from_file(file_path, reader=reader)
movies_df = pd.read_csv(movie_path)

# chatroom public url
def get_chatroom_url():
    # populate this from your local server
    return "http://d5a7fa138cf2.ngrok.io"

# url of the webhook.
# it is done by curl the local ngrok api
```

Now please interrupt the execution and run the last block again to allow this change to be updated in the webhook backend.

Step4 Datastore setups

The screenshot displays the Google Cloud Datastore console. The 'Entities' tab is active, showing a list of 9 users. The table below contains the following data:

Name/ID	email	id	name	randomnumber	surname
name=1	RmFubmLLJvd0hmdXNzQipb3pby5ubA==	MQ==	RmFubmLL	Mz3NA==	Um90a0Z1c3M=
name=2	RGfYbGlzSSBzWlZQJyYVWlM5s	Mg==	RGfYbGlzQ==	ODXNA==	QW1pcw==
name=3	UGF0d0kuQmFycm93BWFuQjhb0UyY2Bud...	Ma==	UGF0d0k=	NTM1Me==	QmFycm93BWFu
name=4	TWFyZWJ0GulJnVwW9uQ2dvb2JibC5ub...	NA==	TWFyZWJ0G=	MjUwHg==	UnVwW9u
name=5	Q3luZ0kuQmFycm93BWFuQ2dvb2JibC5ub...	NQ==	Q3luZ0k=	Nz3uNQ==	QmFycm93BWFu
name=6	TGV0aXRPYSSQYV50aWxvYU84ZWV4ZWVW...	Ng==	TGV0aXRPYQ==	NDQ5Ng==	UGFud0le0E=
name=7	VGF0d0kuQmFycm93BWFuQ2dvb2JibC5ub...	Nw==	VGF0d0k=	ODc2Ne==	V2lsZXM=
name=8	SmlVzc0yS0WkZWJyYV50QmFycm93BWFuQ2dvb2JibC5ub...	OA==	SmlVzcw=	NTA3Ne==	S0lsZ0V0cmFuZ0A=
name=9	Vm90d0F0cm93BWFuQ2dvb2JibC5ub...	OQ==	Vm90d0F0cm93	N254OA==	U2luY2h0XI=

Below the table is a terminal window showing connection logs for Datalab, including messages like 'Instance internal IP is 10.152.0.2', 'Connecting to chatbot=9900', and 'Waiting for Datalab to be reachable at http://localhost:8081/'.

This section is also option unless one would like to setup datastore from scratch. As shown in the screenshot above, only 9 users are loaded into the datastore. This was achieved by using these two notebooks, each read a csv file and populate the datastore with data about Users and Friends. The latter acts as a many to many relationships table keeping track of the friends of a certain user.

The screenshot shows the Google Cloud Datalab interface. The 'notebooks' folder is expanded, displaying a list of notebooks. The following notebooks are listed:

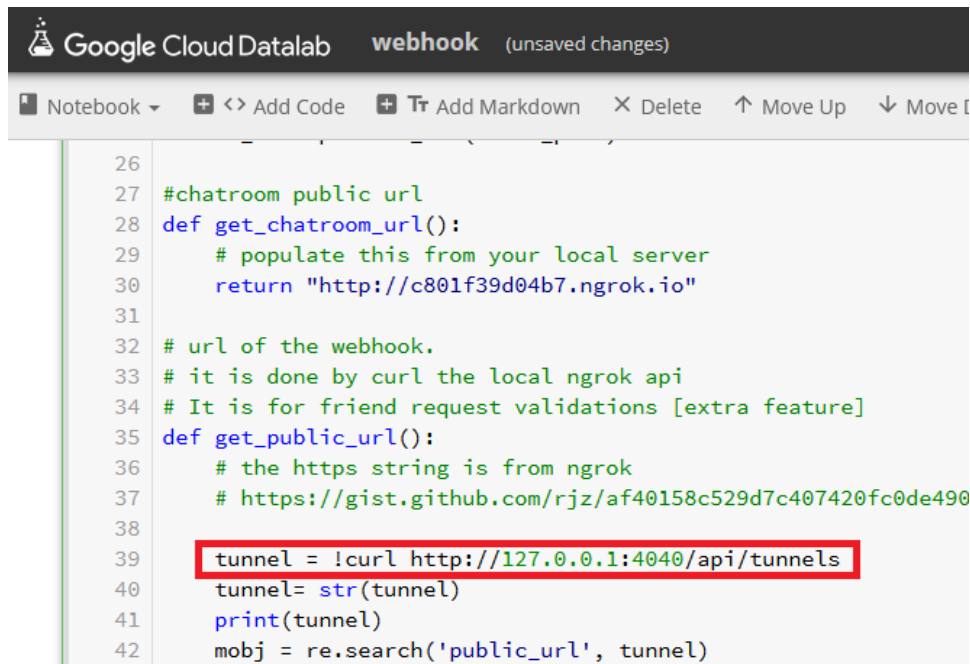
- dialogflow-chatbot
- templates
- DialogFlow.ipynb
- getFriendsData.ipynb
- ngrok.ipynb
- ProcessFriends.ipynb** (highlighted with a red box)
- ProcessHandbook.ipynb
- ProcessSynonyms.ipynb
- ProcessUser.ipynb** (highlighted with a red box)
- Rec.ipynb
- Untitled Notebook.ipynb

Please use the code below to control the number of lines loaded in to the datastore.

```
Google Cloud Datalab ProcessUser (autosaved)
Notebook < Add Code Add Markdown Delete Move Up Move Down
7 line_count = 0
8 header = ''
9
10 for row in csvreader:
11     if line_count == 0:
12         header = row
13         print header
14         line_count += 1
15         continue
16
17     kind = 'users'
18     topic_key = datastore_client.key(kind, row[0])
19     topic = datastore.Entity(key=topic_key)
20     for i in range(len(header)):
21         topic[header[i]] = row[i]
22
23     print topic
24     datastore_client.put(topic)
25     print('Saved {}: {}'.format(topic.key.name, topic))
26     line_count += 1
27
28 #for dev and testing
29 if line_count == 10:
30     break
31
```

```
Google Cloud Datalab ProcessFriends (autosaved)
Notebook < Add Code Add Markdown Delete Move Up Move Down
3
4 datastore_client = datastore.Client()
5 with open('Friends.csv') as csvfile:
6     csvreader = csv.reader(csvfile, delimiter=',', quotechar='"')
7     line_count = 0
8     header = ''
9     tempdict = dict()
10    for row in csvreader:
11        if line_count == 0:
12            header = row
13            print header
14            line_count += 1
15            continue
16
17        kind = 'friends'
18        print('row', row)
19        if row[0] not in tempdict:
20            tempdict[row[0]] = list()
21        tempdict[row[0]].append(int(row[1]))
22        line_count += 1
23        #for dev and testing
24        if line_count == 100:
25            break
26
27    print("tempdict", tempdict)
28
```

Step 5 Extra friend request feature

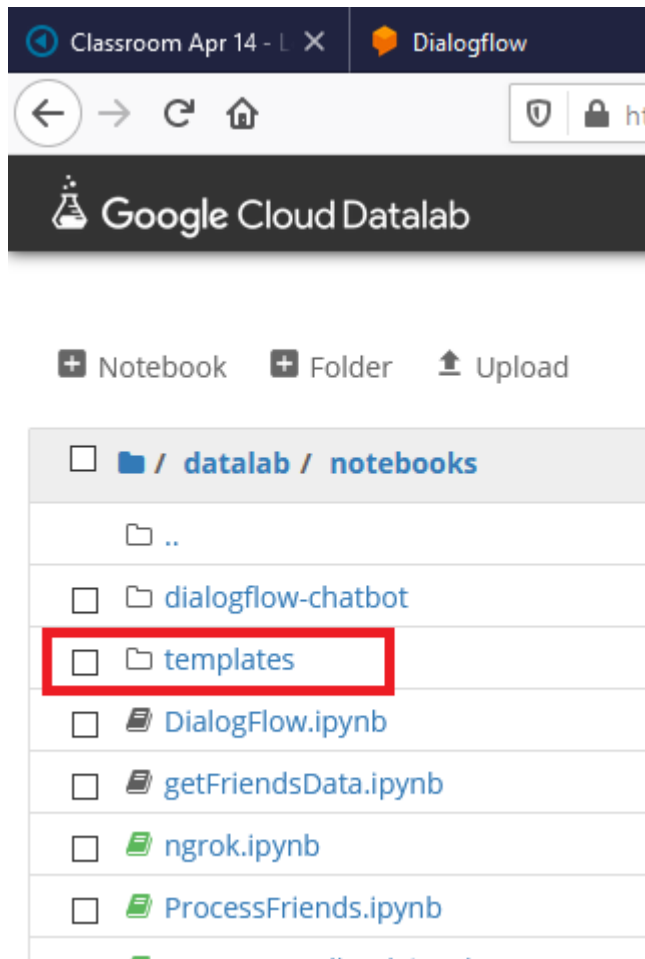


```
26
27 #chatroom public url
28 def get_chatroom_url():
29     # populate this from your local server
30     return "http://c801f39d04b7.ngrok.io"
31
32 # url of the webhook.
33 # it is done by curl the local ngrok api
34 # It is for friend request validations [extra feature]
35 def get_public_url():
36     # the https string is from ngrok
37     # https://gist.github.com/rjz/af40158c529d7c407420fc0de490
38
39     tunnel = !curl http://127.0.0.1:4040/api/tunnels
40     tunnel= str(tunnel)
41     print(tunnel)
42     mobj = re.search('public_url', tunnel)
```

Please manually update the URL above to the value below. The notebook always converted it to another URL automatically. This conversion may be a setup done by the GCP platform, which has become a bug in this project. You may confirm this by saving the notebook and opening it again to see the change in value.

```
tunnel = !curl http://127.0.0.1:4040/api/tunnels
```

Should one want to set up the friend request service on the cloud, the 'templates' should also be added to Datalab with its respective contents.



8. Project Management

8.1. Communication Plan

We ran a total of three Sprints and followed two weeks Sprint planning. Regular scrum meetings and catch up's were scheduled during the sprint to track project progress, remove roadblocks if any and keep pushing towards our objectives and final delivery. Meetings were held three times a week from 8:00 p.m. to 09:00 p.m. to showcase individual work, obtain feedback from others, and address blockers and plan for the next set of activities. We also conducted three sprint planning sessions and two retrospectives over the course of this project. We used WhatsApp and Microsoft Teams as our main communication platform. Blackboard collaborate sessions were used to communicate during the Lab weeks.

8.2. Sprint Schedule

We ran three Sprints over the course of this project.

Week 1: Introduction, Team Forming

Week 2: Selecting a topic for our project and research various design / architecture options

Week 3: Creation and submission of project proposal document, document requirements and create user stories

Sprint 1 (Week 4-5): Environment set up; establish connectivity, interfacing various layers and Demo 1

Week 6: Break week – so we don't plan to run a Sprint; continued developing some features

Sprint 2 (Week 7-8): Software development, testing, bug fixes and Demo 2

Sprint 3 (Week 9-10): Testing, Bug fixes, Performance improvement, project report, user manual, software installation instructions documentation and final Demo

8.3. Project Plan

	A	B	C	D	E	F	G	H	I	J	K
1	Project Milestone	Feb-21		Mar-21				Apr-21			
2		Week 3	Week 4	Week 1	Week 2	Week 3	Week 4	Week 1	Week 2	Week 3	Week 4
3	Introduction										
4	Team Formation										
5	Select Project										
6	Research Industry Trends										
7	Design and Architecture										
8	Requirements Gathering										
9	User Stories										
10	Project Proposal										
11	Dev Environment Set Up										
12	Establish Interfacing Layers Connectivity										
13	Software Development										
14	Unit Testing and Bug Fixes										
15	Integration Testing										
16	Performance Improvement										
17	Project Report and Documentation										
18	Project Demo										

9. References

1. S. Salvi, V. Geetha and S. Sowmya Kamath, "Jamura: A Conversational Smart Home Assistant Built on Telegram and Google Dialogflow," TENCON 2019 - 2019 IEEE Region 10 Conference (TENCON), 2019, pp. 1564-1571, doi: 10.1109/TENCON.2019.8929316.
2. A. Argal, S. Gupta, A. Modi, P. Pandey, S. Shim and C. Choo, "Intelligent travel chatbot for predictive recommendation in echo platform," 2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC), 2018, pp. 176-183, doi: 10.1109/CCWC.2018.8301732.
3. M. Kim, S. Jeon, H. Shin, W. Choi, H. Chung and Y. Nah, "Movie Recommendation based on User Similarity of Consumption Pattern Change," 2019 IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE), Sardinia, Italy, 2019, pp. 317-319, doi: 10.1109/AIKE.2019.00064.

4. A. F. Muhammad, D. Susanto, A. Alimudin, F. Adila, M. H. Assidiqi and S. Nabhan, "Developing English Conversation Chatbot Using Dialogflow," 2020 International Electronics Symposium (IES), Surabaya, Indonesia, 2020, pp. 468-475, doi: 10.1109/IES50839.2020.9231659.
5. What Netflix teaches us about using AI to create amazing customer experiences. (2019, February 04). Retrieved from <https://www.mycustomer.com/service/channels/what-netflix-teaches-us-about-using-ai-to-create-amazing-customer-experiences>
6. Amanda. (2019, October 04). AI bot online in the entertainment industry: Herobot platform. Retrieved from <https://herobot.app/messenger-chatbot/ai-bot-online-in-the-entertainment-industry/>
7. Ganguly, S. (2020, September 21). Examples of chatbots and AI in media and entertainment industry. Retrieved from <https://blog.kore.ai/examples-of-chatbots-and-ai-in-media-and-entertainment-industry>
8. Beyers, J. (2019, March 14). Advantages of Chatbot integration into entertainment industry. Retrieved from <https://chatbotslife.com/advantages-of-chatbot-integration-into-entertainment-industry-f62bfed3c003>
9. Chatbots for media and entertainment industry. (2020, December 22). Retrieved from <https://botcore.ai/media-and-entertainment-bots/>
10. Knight, C. (2020, January 08). The 2020 AI AND Chatbot landscape for sports, entertainment, and media. Retrieved from <https://thechatbot.net/2020-chatbot-sports/>
11. Conversational AI in media & ENTERTAINMENT: Artificial solutions. (2020, December 01). Retrieved from <https://www.artificial-solutions.com/conversational-ai-media-entertainment>
12. Samhita Alla (2018 May 26) Building your first Chat Application using Flask in 7 minutes. Retrieved from <https://codeburst.io/building-your-first-chat-application-using-flask-in-7-minutes-f98de4adfa5d>
13. Cloud Architecture Center (2021 April 04) Building a chatbot agent by using Dialogflow (part 1) Retrieved from <https://cloud.google.com/solutions/building-chatbot-agent-dialogflow>
14. Cloud Architecture Center (2021 April 04) Datastore Bulk Delete. Retrieved from <https://cloud.google.com/dataflow/docs/guides/templates/provided-utilities#datastore-bulk-delete>
15. The Movies Dataset Retrieved from <https://www.kaggle.com/rounakbanik/the-movies-dataset>