# **Programming Challenge: Secret Santa**

I thoroughly enjoyed working on this interesting task. The given problem is a very common one so I thought rather than reinventing the wheel I could collect references online and make use of it in my base program.

I selected Python as my base language as I have been recently working a lot with Python and I visualised it as a simple python script to get it working

The input to the application can be done via 2 methods

1. Load a Csv file which stores value as

{name} |{email\_id} | {family\_members\_comma\_seperated}

1. Direct input to the program

Part 1 of the challenge gathers the names and assigns random secret santa to each individual such that the santa and recipient is not same.

For Part 2 of the challenge I went with a simple approach of storing the history in a Dictionary per user using a simple file created through Python. Named it SantaHistory.txt. If the users change, the history has to be cleared before having a re-run.

The output is stored in an output file named as SantaAllocations.txt for easy viewing and even mail can be sent to the respective users.

On repetitive runs the dictionary is appended with names. On the fourth run the first name corresponding to the user is cleared and a new entry is added to the end thus maintaining the history for 3 yrs.

Part 3 of the challenge has been tackled by getting the family information directly at the start or via txt file.

And similar logic like the history check was applied to avoid family names too.

Deployed the code on Heroku.

Scope of Improvement :

1. More edge cases could be found and covered
2. Adding more users to the existing user list with history preserved
3. Backend integration with a DB
4. Unit Tests
5. Cleaner and leaner code

Scaling :

My thought process of scaling the application is to provide a frontend UI page to capture the user session so that multiple users can use the application at the same time and having the history and family info per logged in user. connecting the script to a relational database which would store the user session info and the corresponding history and family information.

Production Ready :

The application can be deployed to any cloud server like AWS, GCP, Azure, Heroku.

Having good hands-on with AWS I would suggest to deploy the application on a AWS Lambda or AWS Glue where the api to get/fetch/store information can be deployed on AWS Gateway through which the UI can be connected.

We could also containerize it on Docker for sharing it with others for local use or deploying it.

Output :

Input :

gautam | gaut@abc.com | apurva  
apurva | apu@abc.com | gautam  
prashant | prash@abc.com  
paritosh | pari@abc.com | aditya, karnika  
vishvesh | vishu@abc.com | rutuja  
ash | ash@abc.com  
rutuja | rutu@abc.com | vishvesh  
aditya | adit@abc.com | paritosh, karnika  
karnika | karnika@abc.com | aditya, paritosh

History :

{

'gautam': ['paritosh', 'rutuja', 'karnika'],

'apurva': ['aditya', 'karnika', 'prashant'], 'prashant': ['vishvesh', 'ash', 'paritosh'],

'paritosh': ['ash', 'vishvesh', 'rutuja'],

'vishvesh': ['prashant', 'aditya', 'ash'],

'ash': ['karnika', 'paritosh', 'gautam'],

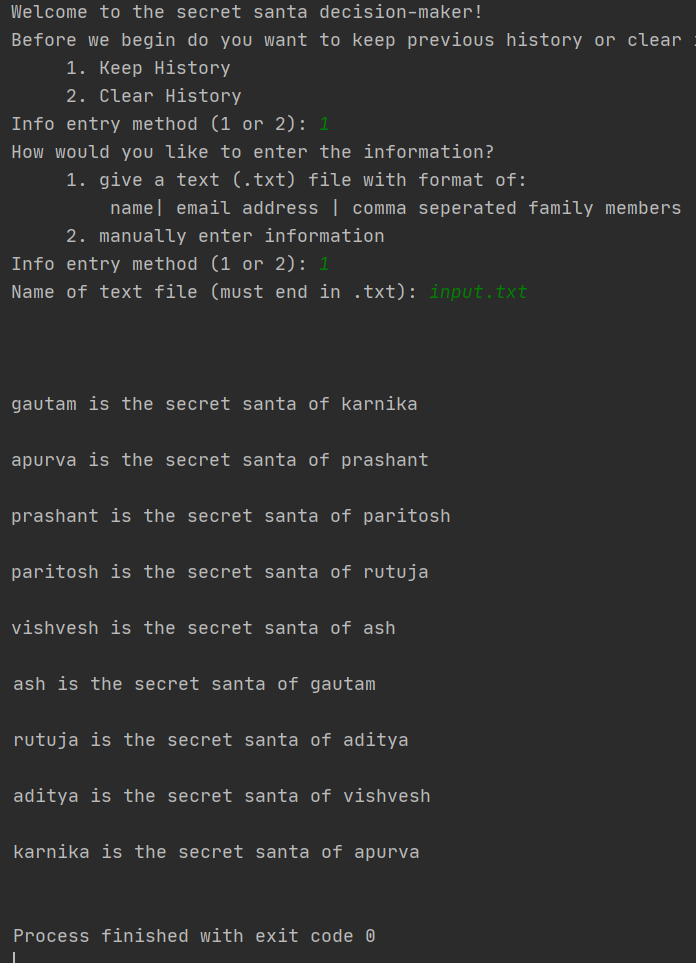
'rutuja': ['apurva', 'prashant', 'aditya'],

'aditya': ['gautam', 'apurva', 'vishvesh'],

'karnika': ['rutuja', 'gautam', 'apurva']

}

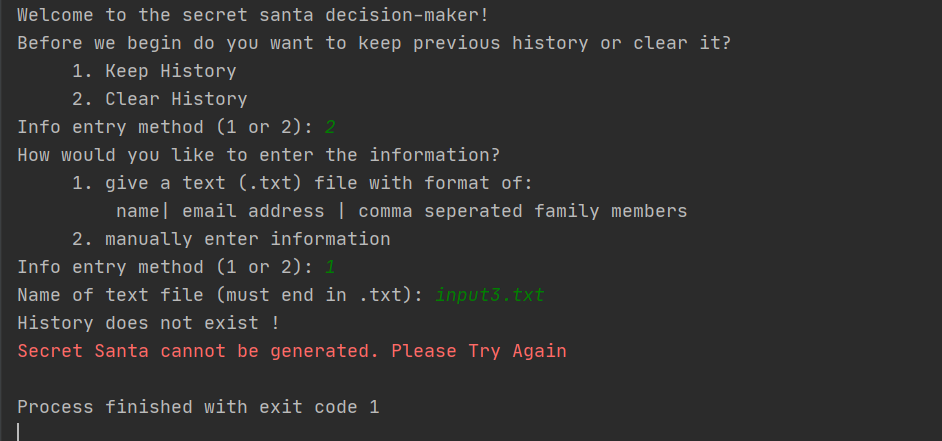
Successful Run :



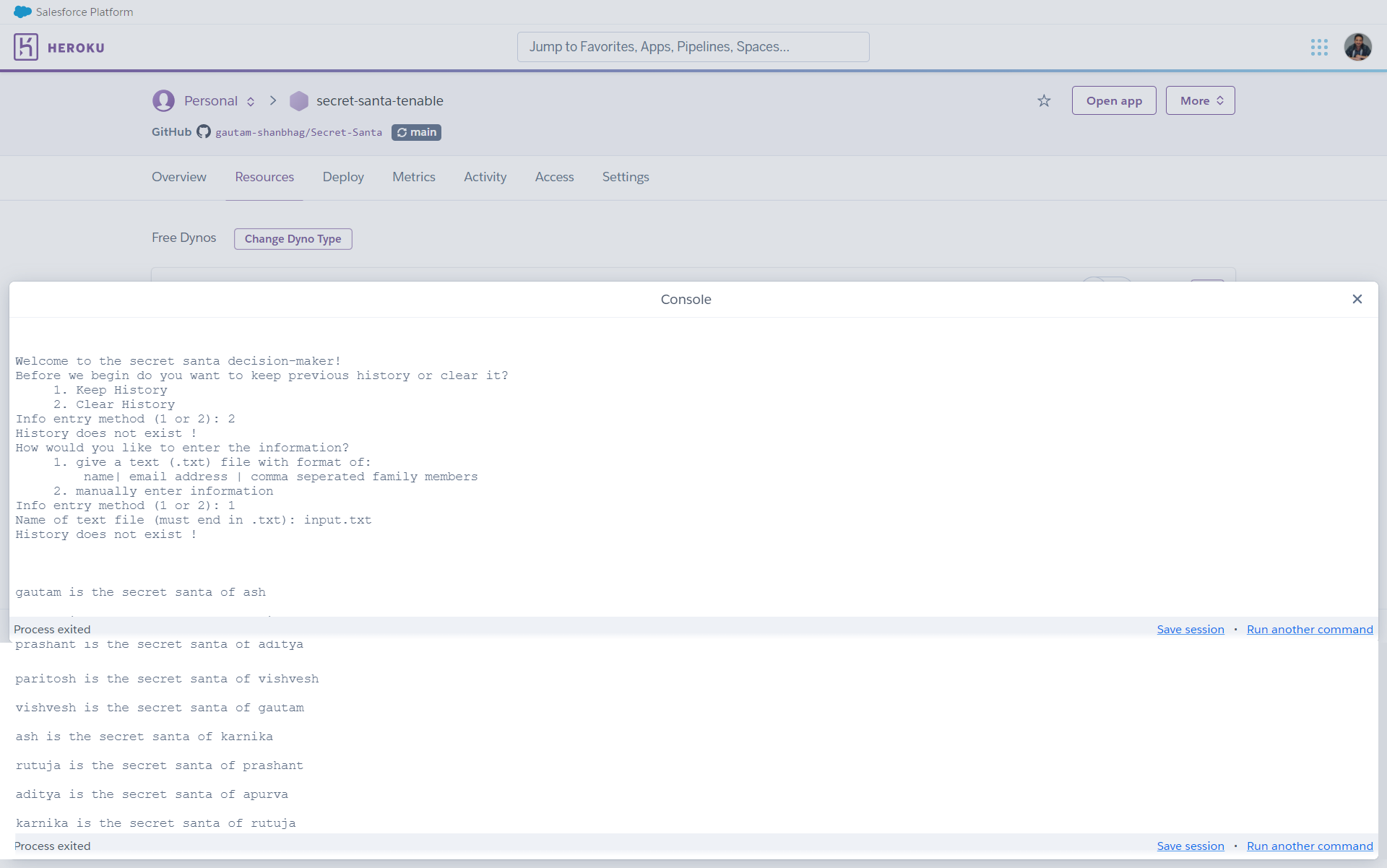
Input :

gautam | gaut@abc.com | apurva, prashant, ash  
apurva | apu@abc.com | gautam, prashant, ash  
prashant | prash@abc.com | apurva, ash, gautam  
ash | ash@abc.com | gautam, prashant, apurva  
paritosh | pari@abc.com |

Deadlock condition :



Heroku:



Reference :

* <https://medium.com/analytics-vidhya/i-made-a-contactless-secret-santa-algorithm-with-python-7374d4a79c56> by Monique Cheng