Carnegie Mellon University Qatar

Team Argo

(Qur'anic Botanic Garden)

Web presence and news aggregation system

System Design Document

Version 4.0

Prepared for: 67-373 (Software Development Project), Summer 2017

Prepared by: Abbas Ali, Abdul Wasay, Amna Al-Ansari, Umair Qazi

Professor: Selma Limam Mansar

May 29th, 2017

TABLE OF CONTENTS – Needs to be fixed

REVISION HISTORY

Date	Description	Author	Comments
May 24th 2017	Version 1.0 Initial Document setup	Umair	Setup of design and structure of the document with headings
May 28th 2017	Version 2.0 Completed Sections	Team Argo	Filled in content for different sections
May 29th 2017	Version 3.0 Compilation and proofread	Umair	Compiled different sections in one file plus proofreading
May 29th 2017	Version 4.0	Team Argo	Corrected different sections based on feedback

1. Executive Summary

This system design document explains different components of ARGO (web presence and news aggregation system for Quranic Botanic Garden). We follow a design philosophy that is rooted in three principles: functional aesthetics, sustainable design and simplicity over sophistication. ARGO's design is intended to be minimalist and user friendly. The document explains the Architectural Design of ARGO and describes the function of the three major components of the system: Front End, Back End and the Database. The Front End provides the user with three interfaces: Scrape, Query and View. These components will help the system admin to retrieve relevant information. This document explains how ARGO is based on a three-tier architecture design with the aim of automating the news aggregative for Quranic Botanic Garden. The document explains the design of the database which revolves around devising a system based on the system components explain above while ensuring third normal form. The document also explores a user interface that focuses on providing a sustainable solution to the client in ARGO. Finally, the documents elaborate of different technologies used in ARGO such as PDFKit, different Official APIs, a SQLite 3 database and more.

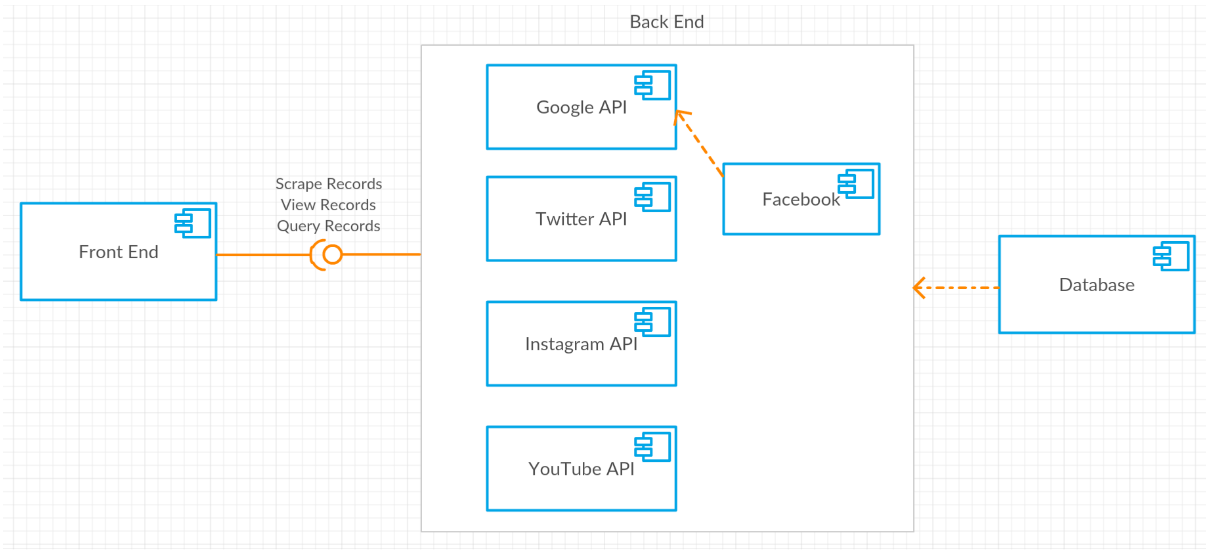
2. Introduction

Qur’anic Botanic Garden (QBG) is currently a Qatar Foundation project tasked at discovering and cataloguing all plants and herbs mentioned in the Holy Qur’an and hadith and raising awareness of their benefits to the public, they have plans to expand as an organization and maintain conservatories that will be part of the Oxygen Park. QBG has tasked our team with the task to produce a system that will get a handle on their web presence to analyze their influence and impact on the community at large. The solution that we have come up with is ARGO, a web presence and news aggregation system that will enable QBG to download all news posts, tweets, Instagram posts, YouTube posts and articles on the web which are related to Qur’anic Botanic Garden. ARGO will help Qur’anic Botanic Garden to measure their influence and impact on the web.

ARGO aims to greatly simplify the current problem facing QBG of manually having to look through the internet and gathering the data and compiling it themselves. The system will scrape many different sources on the internet: Google, YouTube, Twitter and Instagram and gather the data for them. We believe this solution is feasible in many aspects, for example, technically we will only need a computer (that we will configure and setup) connected to the internet to run the program, nothing else will be required in terms of software and hardware, economically feasible because it requires no purchases to develop or maintain. Lastly, the design philosophy that we follow focuses on providing a sustainable solution. This document lays out that design of our proposed system: ARGO

3. Architectural Design

a. Component Diagram



b. Component and Internal Interfaces

<u>**Back-End**</u>

The Back-End is the brain of the system. This Back-End is responsible for fetching results from the web and incorporating results from Google API, Twitter API, Instagram API and YouTube API. Results are also obtained from Facebook but that is done through the Google API by using the keyword 'inurl:facebook.com' as one of the keywords in the search. The Back-End is responsible for combining all records from different sources. This component is also responsible for automatically saving all the scraped results in a local database so that it can queried and accessed at a later point in time. This component also converts all the records to PDG or PNG format for easy access if the url becomes inactive later after the results were scraped using ARGO.

<u>**Front End**</u>

The Front End is responsible for providing the user to scrap, query and view records related to the organization. Front End has three components: the scrape component, the query component and the view component. It allows admin to scrape the full web for posts/news related to QBG. It also allows the admin to scrape the web based on time criteria. It also allows automatic saving of new posts.

<u>Database</u>

The Database allows querying and retrieval of records based on different criteria. It stores the data elements like PDF or PNG files in the it. It also admin easily query and retrieve different records. The rotational schema and the Entity relationship diagram will be detailed our later in the document.

c. Interfaces

The front end provides three different interface which give different functionality to the user:

<u>Scrape Component</u>

Allows admin to scrape the full web for posts/news related to QBG

Allows admin to scrape the web based on time criteria.

Allows automatic saving of new posts (old posts stay in DB)

<u>Query Component</u>

Allows admins to query the local database for records based on the following criteria:

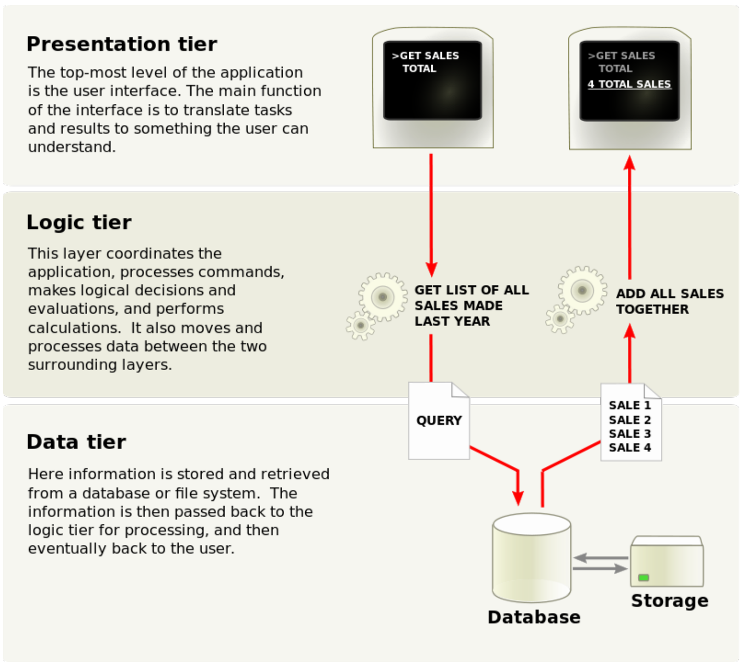
Time, Tag, Event, Source

<u>View Component</u>

Allows the admin to view different records which are filtered by the Query component

Allows admin to view posts/news found on the web in PDF format

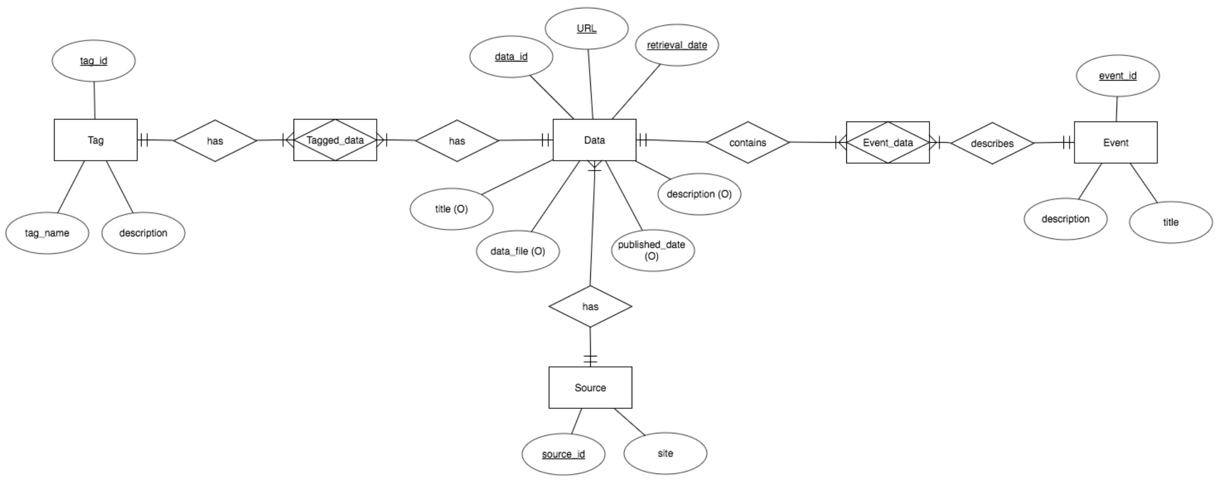
4. Design Decisions


Presentation tier
The top-most level of the application is the user interface. The main function of the interface is to translate tasks and results to something the user can understand.
>GET SALES TOTAL
>GET SALES TOTAL
4 TOTAL SALES
Logic tier
This layer coordinates the application, processes commands, makes logical decisions and evaluations, and performs calculations. It also moves and processes data between the two surrounding layers.
GET LIST OF ALL SALES MADE LAST YEAR
ADD ALL SALES TOGETHER
QUERY
SALE 1
SALE 2
SALE 3
SALE 4
Data tier
Here information is stored and retrieved from a database or file system. The information is then passed back to the logic tier for processing, and then eventually back to the user.
Database
Storage

For the design, we will follow the Three tier architectural style. It connects the user-interface (front-end) where users can access the system easily and directly to the system (back-end) and then to the database where the data is stored. In terms of the overall design and the user interface design, we focused on developing a user-friendly searchable database that will make it easy for the users to scrape the internet for new and updated results and to also allow the user to query the local database that includes all the posts, PDFs and Snapshots related to QBG that are found on the web based on time, source, events, and tags.

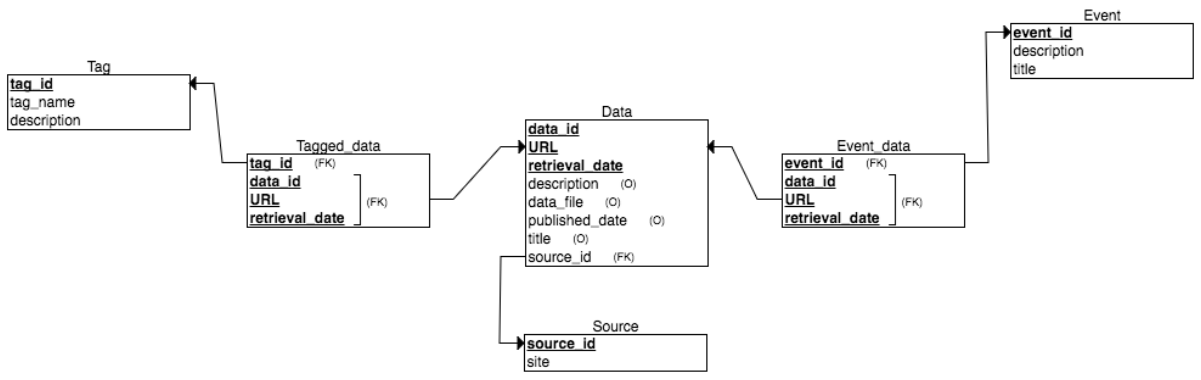
5. Database Design

a. Conceptual Data Model



The above entity relationship diagram demonstrated that we have three main entities in our database, the Tag, Event and Data entities, the Source entity is just a supporting table to the Data entity to avoid any normalization issues. The Source entity only includes the unique attribute source_id and site, for example 1 and Facebook. The Event entity contains the unique attribute event_id, description of the event and title of the event. The Tag entity contains the unique attribute tag_id, tag_name and description. The data entity is the main table for our cataloguing use, it is the master table, including the unique attribute as a composite key with data_id, URL and retrieval_date, and has other optional attributes such as, title, description, data_file and published_date. A number of data records can be searched by a certain tag, and a number of tags can describe a single data record. Similarly, several records can document a certain event and a number of events can be mentioned in a single record.

b. Logical Data Model in 3NF



The above logical data model or relational schema is a derivation of the aforementioned entity relationship diagram. This model shows a clearer understanding of the database as it shows how the source_id is referenced in the Data table as a foreign key, so there wouldn't be any data redundancies from repeated values in the source column. The associative entities between the Event and Data entity and the Tag and Data entity have also been made into separate tables as it is a many-to-many relationship between them, combining their respective primary keys into one table each. These two associative entity tables will be very useful in our database for grouping data. The Tagged_data table can be used to narrow search criterias for records and the Event_data table can be used for categorizing saved data according to each event that they describe or mention.

c. Database Choice

The came up with three different alternatives for the selection of our database system, MySQL, SQLite3 and File I/O.

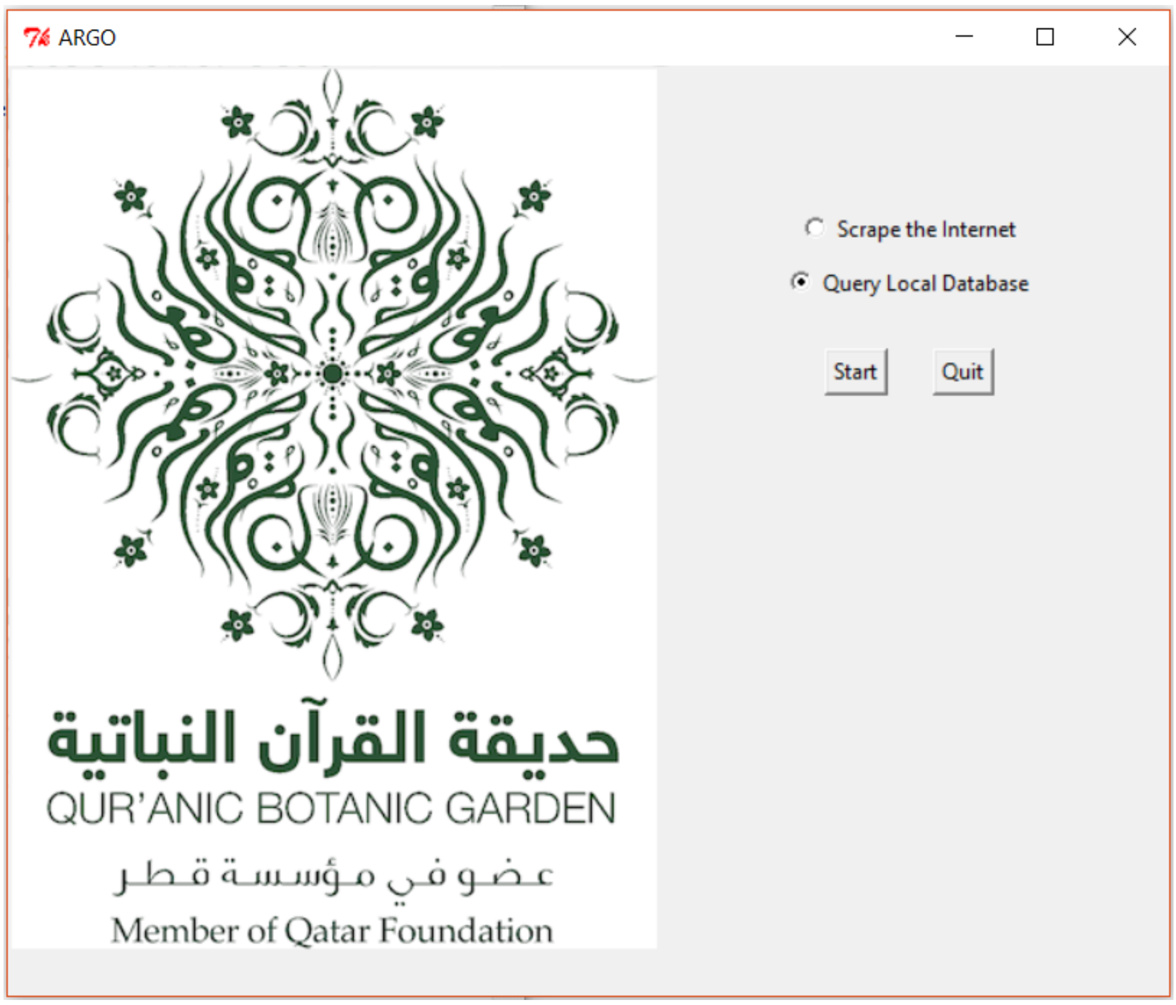
We wanted to use a **File I/O system** to implement our database because it would give us more freedom and customization in displaying data on our interface by saving and querying data that we would have saved locally on a file through Python. We decided not to use a File I/O system because for us to implement a database using this concept we would have to implement full searching and querying functionality from scratch, and the functionality that we create may not be as robust within the time frame that we have either. So, creating a whole new database system from scratch we decided would be a waste of our time and effort especially if well tested and widely used and supported database systems already exist.

Two of the python supported database systems we considered were **MySQL** and **SQLite3**, the problem with MySQL was that it doesn't work that well with python and most suitable to use with PHP and it had a lot more functionality outside the scope of our requirements.

In the end, we have decided to use **SQLite3** since it works best with python, is 'lightweight' so allows us to be able to do exactly what we need for the scope of our project and making use of saving in UNICODE and 'blob' attributes allowing us to save all characters and all forms of data, such as PDF, JPEG and text files that we will be saving in our database for our project.

6. User Interface Design

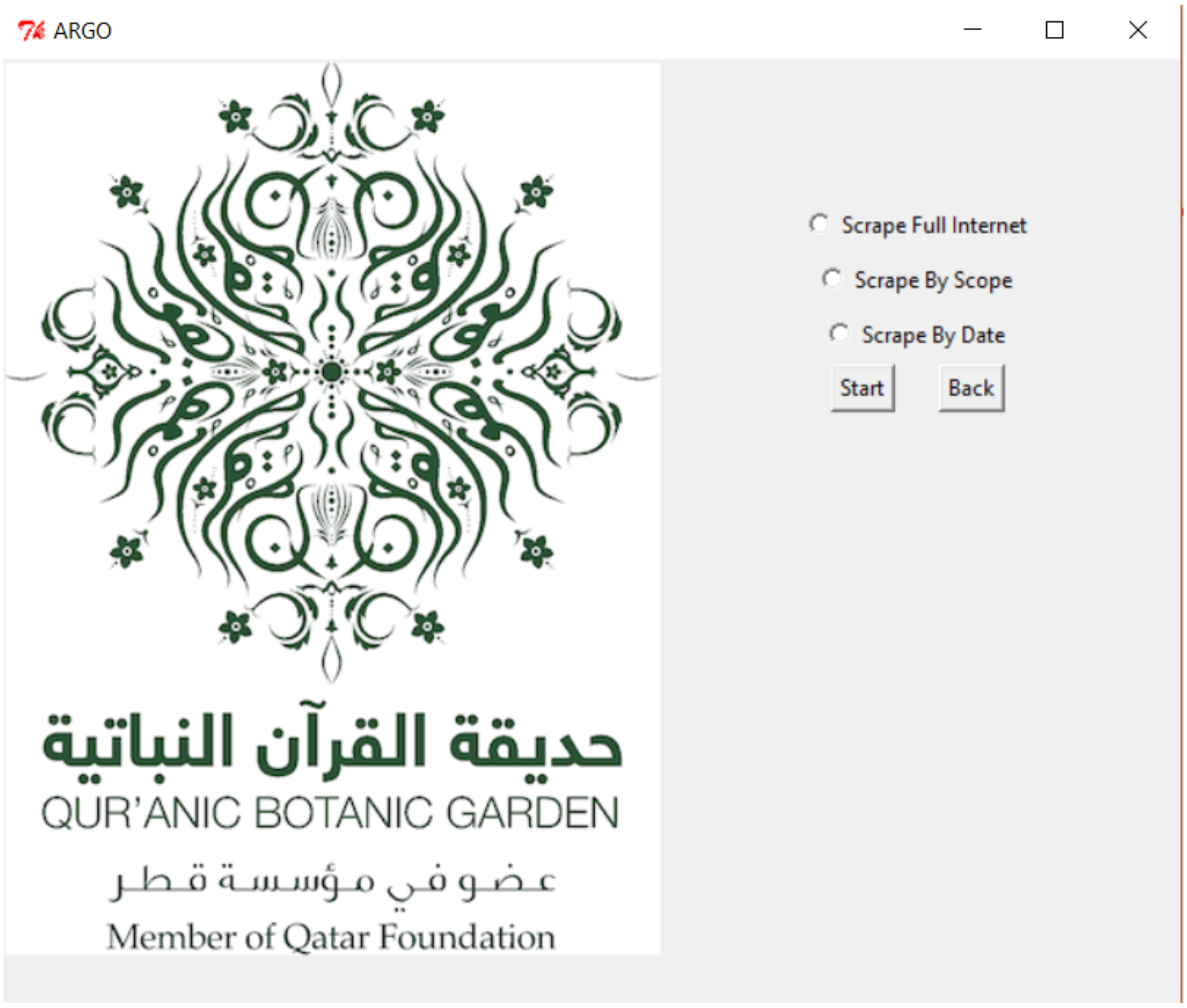
Main Screen



Scrape the Internet [✓ Query Local Database]

The Users can choose between two options when first opening up the program, either scrape the internet for news about QBG or search through the records in the local database about QBG. Users can confirm selected choice by pressing start button. Users can quit the program by pressing the quit button.

Scrape Screen



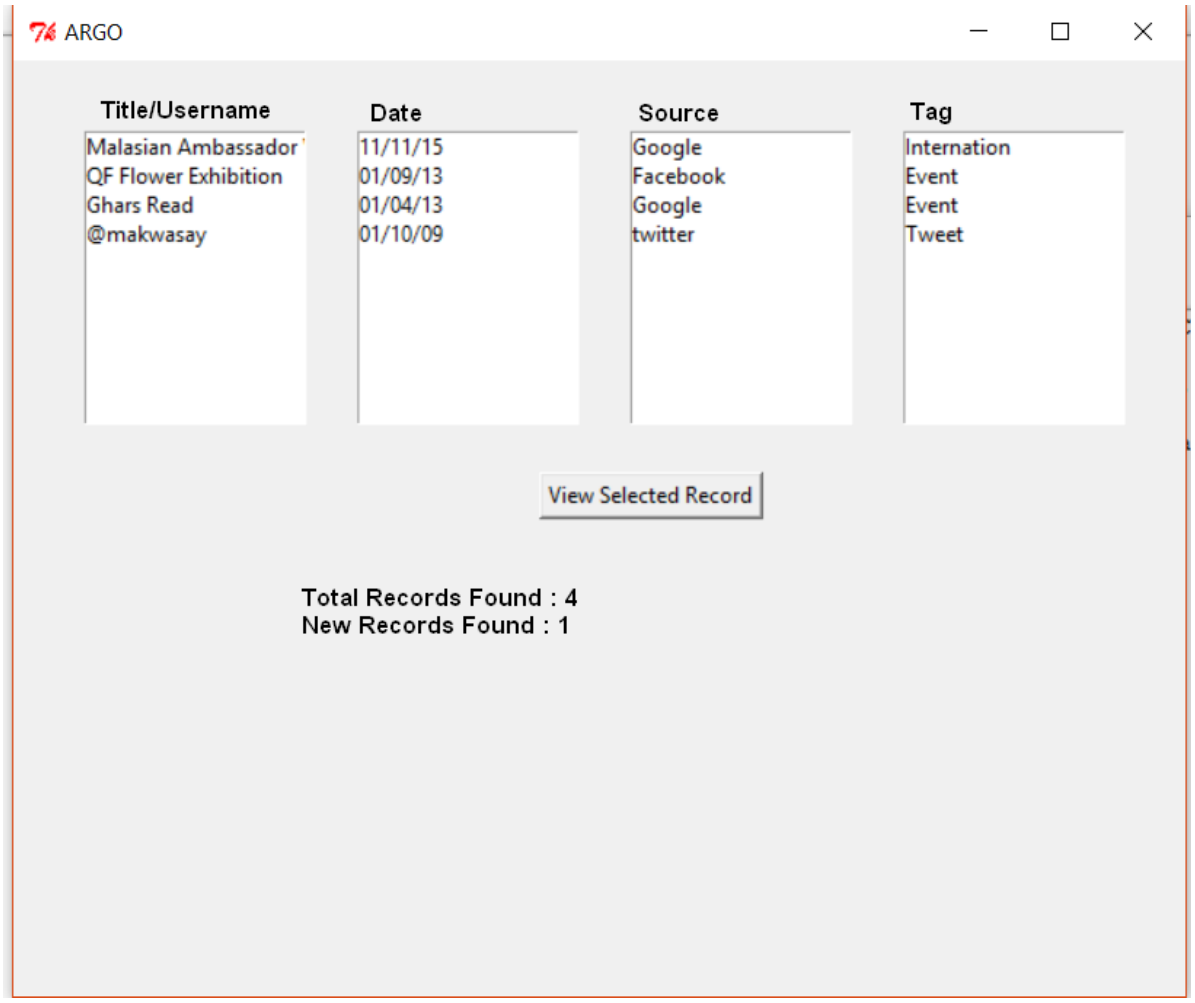
Users choosing to scrape the internet, can press start and then choose multiple options in which to scrape.

Users can scrape the whole internet from all sources.

Users can scrape internet by source.

Users can scrape internet by last publication date.

Results Screen



Title/Username	Date	Source	Tag
Malasian Ambassador	11/11/15	Google	Internation
QF Flower Exhibition	01/09/13	Facebook	Event
Ghars Read	01/04/13	Google	Event
@makwasay	01/10/09	twitter	Tweet

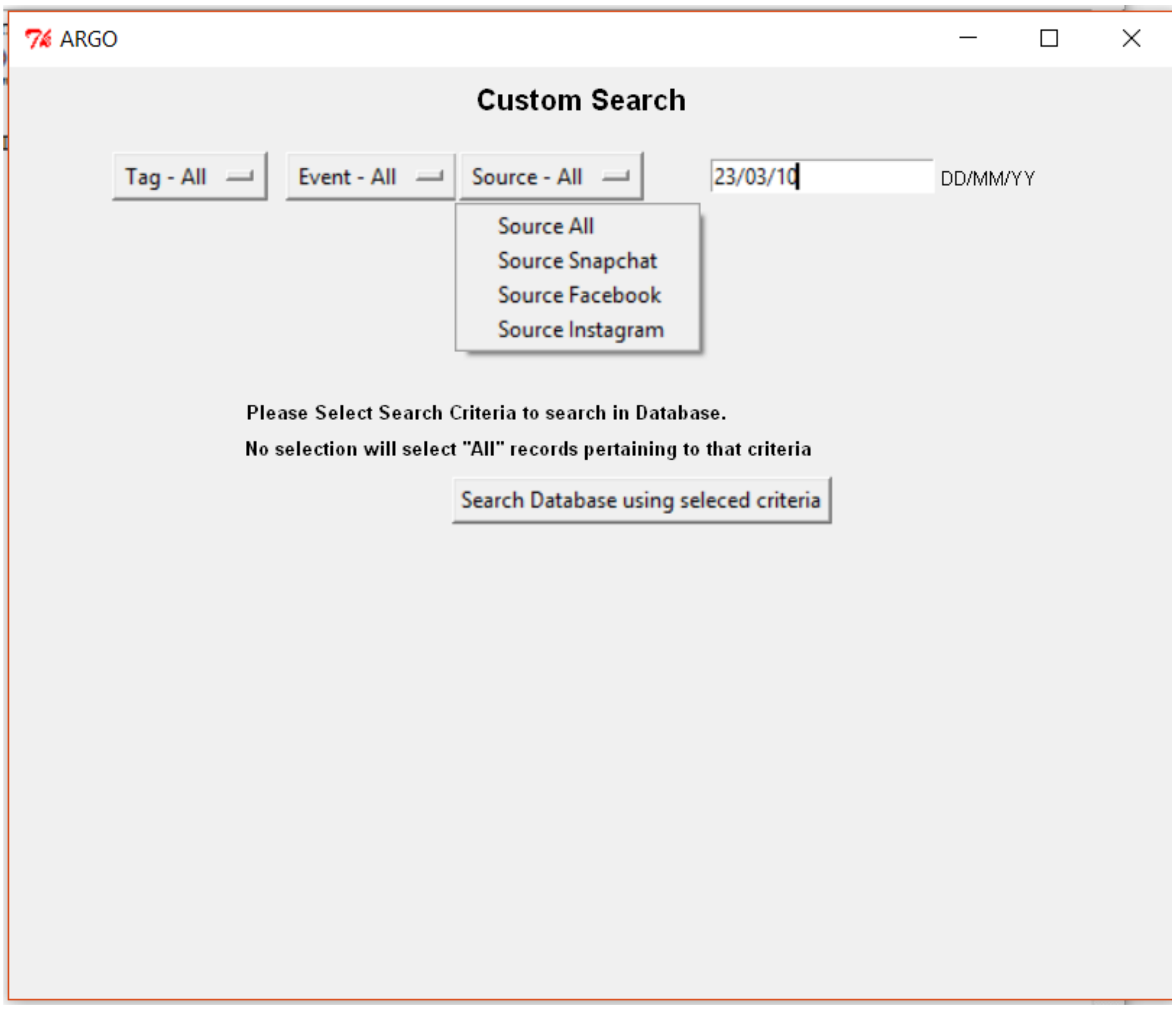
Total Records Found : 4
New Records Found : 1

Users can view scrape results in a table.

Users can see how many results the scrape yielded.

Users can see how many new results the scrape yielded when compared to the local database. It allows to view pdfs.

Search Screen



Please Select Search Criteria to search in Database.
No selection will select "All" records pertaining to that criteria

Users can choose to view records in the database categorized with a particular event.

Users can specify which event to search for using a drop-down menu.

Users can choose to view records in the database categorized with a particular time, source event or tag.

7. Technologies and software development tools

Technology	Implementations
Python	A manageable and usable programming language, with the widest support for APIs (YouTube, google, twitter, PDF conversion) required for the project.
SQLite 3	A database that includes all the posts, PDFs and Snapshots related to QBG that are found on the web.
ERD plus	Is a database modeling tool used to create Entity Relationship diagrams required for the project.
TkInter	Is a Python standard GUI programming language that will be used to display information required for the project.
PDFkit	Convert documents downloaded to PDF version
Visio	For the UML case diagram required for the project
YouTube API	API used to get the Youtube information related to QBG that are found on the web.
Instagram API	API used to get the Instagram information related to QBG that are found on the web.
Google API	API used to get the Google information related to QBG that are found on the web.
Twitter API	API used to get the Twitter information related to QBG that are found on the web.
PNGCrush	It is a command-line utility used to optimize PNG image files (posts) found.
teamgantt	Used to create the Gantt chart required for the project
Google Drive	A shared folder that includes all presentations and documents shared between the team members.

8. Conclusion

The problems faced by the client is that of not being able to efficiently and effectively find news about the organization on the internet. This occurs because news about QBG (client) is sent out by a third party that does not communicate back where the news has been shared. Hence news and articles must be searched for manually. This takes time and effort and runs the risk of links expiring and the news disappearing forever. The proposed system that is solving this problem uses multiple APIs to search the web for QBG news/articles and catalogs them in a local database for the organization's personal use as well as their annual reports. Therefore, ARGO's design with its focus on minimalism and user-friendliness with help QBG deal with this issue head-on.

References

Bartledan. (2009). Overview of a three-tier application. Retrieved May 29, 2017, from https://commons.wikimedia.org/wiki/File:Overview_of_a_three-tier_application_vectorVersion.svg

Google API Library. (2017). Retrieved May 18, 2017, from https://console.developers.google.com/apis/library

Qur'anic Botanic Garden. (2015). Retrieved May 18, 2017, from http://qbg.org.qa/

REST APIs — Twitter Developers. (2017). Retrieved May 18, 2017, from https://dev.twitter.com/rest/public

Appendices

Agenda	Client Meeting	Team Meeting	Date/Time	Done
Meeting to discuss possible questions for client.		x	May 9th, 10:30 to 11:30	✓
Introduction and discuss basic requirements.	x		May 10th, 8:30 to 10:30	✓
Discussion of first client meeting.		x	May 11th, 10:30 to 11:00	✓
Discussion of solutions.		x	May 13th, 12:00 to 2:00	✓
Suggestion and discussion of possible solutions.	x		May 14th, 1:25 to 2:10	✓
Division of the work amongst the team members for proposal.		x	May 15th, 9:00 to 9:40	✓
Discuss DB+GUI+APIs		x	May 21st, 1:00 to 2:00	✓
Meeting with clients and Dean Selma	x		May 24th, 10:30 to 11:20	✓
Team meeting after the Client & Dean meeting		x	May 24th, 11:20 to 12:00	✓
Team meeting (Wasay + Abbas + Umair)		x	May 25th, 9:00 to 10:00	✓
Meeting to discuss progress on the design doc + application		x	May 27th, 1:00 to 2:00	✓