Interoperability Final Project Narrative Document

Jared Keklak

USC-Upstate

762: Interoperability

Abstract

This paper is narrative document for a final programming project. The project includes the development of a custom RESTful API that uses FLASK framework. Additionally, 3 external public API’s were integrated into the custom API functionality. The theme of the API is cat classification. Everything is written in python language.

Keywords: API, Cat, Classification, Machine Learning,

Interoperability Final Project Narrative Document

This Document is meant to illustrate the process and planning used to programing the final project for INFO 761: Interoperability. More details of the specific programming used can be found in the Custom API Specification document. This document is a general narration of my experience when completing this project and the result. The project narrative has three main components: Developing the custom API, developing the integration for the 3 public APIs, and developing the driver script that operates all 4 API’s. All code referenced in this document can be explained in more detail in the Custom APS Specification document and can be explicitly viewed in the corresponding GitHub page.

**General Project Structure**:

The Python project is divided into 3 main components. The custom API: which includes the main script that generates the server which hosts the API, a put script for posting information to the API, a get script for getting information from the API, and an update script which is for updating entries. The public API’s: each public API is integrated into a function housed in its own script for calling. Lastly, the Driver file: which integrates the usage and ability of all the API’s.

# Custom API:

The custom API uses FLASK framework and hosts the local server that contains the SQL database that contains information about cats. The custom API has basic CRUD (Create, Read, Update, Delete) functionality. When the main.py file is running, the sever is actively being hosted and can have changes made through the other functions in the python project.

Aside from the main script that hosts the API, the other part of the API are the 3 functions that act as a liaison between the custom API and the rest of the project.

The put.py file contains the functionality to post information to the server through the API. It contains two methods, put() and put new(). Put is used for adding some default entries to the database. put\_new() is used for adding new entries.

The get.py file contains a simple method to return a Json object of information from the API. It queries the API based on primary key to return the text information in the corresponding database entry back in the form of a Json object.

The update.py file contains a simple method to update entries in the database. it accepts a primary key as parameters, a text field for identifying the attribute of the entry to be modified, and a text field for the new information that will be posted in the entry.

Each of the functions manipulates an SQL database hosted on the server. The SQL database used by the custom API only stores string information, this was decided to keep types between functions calls for each separate API consistent. All information returned from each API is converted to string format so it can easily read and posted by the custom API.

# Public API’s:

The 3 public API’s chosen were the Cat Breed Classifier API from Zyla Labs, then Cat Information API from Zyla Labs, and googles YouTube API.

# Cat Classifier API

The Cat Breed Classifier API from Zyla labs is based on a “well trained” machine learning algorithm that can identify cat breeds in an image. When an image URL is sent to the API, the API replies with a Json object which contain multiple scores and cat breeds. The score closest to 1 is the most accurate match. The file INFOproj\_CATCLASSFIER\_API.py houses the method which calls this API. The method classify\_cat() sends an image URL to the API, acquires the Json response, and then a simple for loop is used to iterate through the response, measure each accuracy score, and acquire the breed with the highest score. The name of the breed is returned.

# Cat Information API

The Cat Information API from Zyla labs is used to extract facts about a specific cat breed from an existing cat fact database. Some of the basic facts the API can provide based on cat breed include place of origin, average lifespan, a score of intelligence, and others. The file INFOproj\_CATFACTS\_API.py houses the method which calls tis API. The python file contains the method “get\_cat\_facts()” which accepts a string for breed name, and sends a request to the cat information API. The API returns a Json response of simple cat facts for the specified breed. The Json information returned is then filtered and formatted to a readable string that can be printed in the console.

# YouTube API

The YouTube API is a free API hosted on the Google developer platform. The YouTube API can be used to gather video analytic data, search results, and other information from the YouTube platform. The python file INFOproj\_YOUTUBE\_API.py houses the method which calls this API. It contains one method get\_youtube\_videos(). This method creates a request to the YouTube API, the YouTube API returns Json object representing a video based of the keyword sent to the method. One video result is returned. The video ID is extracted from the Json object and is added to the standard YouTube watch URL. The YouTube URL of the video is returned in string format. Below is snippet of the return statement for this method for clearer understanding.

A screen shot of a computer

Description automatically generated

1 getting the video ID from the response, appending it to the youtube.com/watch URL.

**Driver:**

The driver file (driver.py) is a simple and straightforward python script that showcases the functionality of the three public API’s and the custom API. It does not contain any classes or functions; it is purely a functional script that makes calls for information from the 3 public API’s and posts new entries and updates to the main server through the custom API. An example code snippet is below.

A computer screen shot of a computer program

Description automatically generated

2 Driver File code snippet

The output produced by the driver file looks like this:

A computer screen shot of a computer code

Description automatically generated

3 Driver File output

The Driver file showcases the functional uses of the API’s working together. The general process is adding a new entry to the database that has at least an image URL of a cat. Based off the image of the cat, and what breed it is classified as, the rest of the information in the database is filled out. When a new entry is created, an image URL is required. The image URL is sent to the cat breed classifier API using the INFOproj\_CATCLASSIFIER.py method. The breed’s name is returned. Then, the breed’s name is sent to the cat information API using the INFOproj\_CATCLASSIFIER.py method. The Json response is returned, filtered, and formatted into a string list and then posted to the DB. Lastly the YouTube API is used to search YouTube and return a video result about the cat breed.

**Conclusion:**

The end result of this programming project is a simple but function interoperable custom API that can manipulate information in a basic SQL database, and acquire data from 3 external public API’s.

Relevant Links:

Cat Information API: <https://zylalabs.com/api-marketplace/data/Cat+Information+API/3561>

Cat Breed Classifier: <https://zylalabs.com/api-marketplace/machine%20learning/Cat+Breed+Classification+API/499>

YouTube: <https://developers.google.com/youtube/v3>