**Journeying through APIs**

Criosanna Allred, University of South Carolina Upstate

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Abstract**

This paper outlines the process needed to create a custom API to pull the number of COVID-19 hospitalizations and the number of Medicare Enrollments by state using three public APIs. The custom API has also been designed to accept user input to pull their specified state and zip code information to display the average number of COVID-19 hospitalizations and Medicare enrollments there are per zip code in their provided state.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. **Introduction**:

For those who are not familiar with APIs and RESTFul APIs I sought to find a source that would simply explain what they are for and how they function in language that anyone could understand. Unsurprisingly, Amazon has some documentation available for end-users who are interested in using their API services. Amazon describes an application programming interface (API) as a structure that defines the rules that must be followed to communicate with other software systems through the Internet. They go on to say that “developers expose or create APIs so that other applications can communicate with their applications programmatically.” 1 This project is designed to both access the exposed APIs created by other developers to make public data accessible to me, and in-turn I will be able to take that data and create a custom RESTful API which will serve as the interface between end-users and the data I will be providing them.

The goal of the project outlined in this paper is to design a custom API that links monthly COVID-19 hospitalizations from Centers for Medicare and Medicaid Services (CMS) and monthly Medicare enrollments from CMS to the number of postal codes in each state. Through this API the user can create a pin code to use to save their selected two-letter state abbreviation and zip code to pull their specific information related to COVID-19 Hospitalization and Medicare Enrollment data for their state. The output provided for the user will include the number of COVID-19 Hospitalizations, the number of Medicare Enrollments, the average number of COVID-19 Hospitalizations per zip code in the specified state, and the average number of Medicare Enrollments per zip code in the specified state.

Visual Studio Code was used to pull the data from the public APIs, parse the data, and create the custom API, while Postman along with my local command prompt was used for testing the API functionality. SQLite was used during the creation of the custom API to ensure that the data being parsed into the database was being committed to the database and was formatted correctly for the functionality of the custom API. The standard CRUD (create, read, update, and delete) structure was used for the creation of the custom API to create the database, get data for the user through an API call, update the data shown in the API call, and delete the user and associated state and zip code information should it be necessary.

1. **Related Work**:

When digging through API documentation for source data for a project there are times when you find exactly what you are looking for – the perfect data to put together a database with all the information you need and want to work with. However, there are times when you think you have done this, only to discover that the APIs have been retired and are no longer available. This was the beginning of the journey I didn’t realize I was going to take while going through this project.

The original APIs I had found were stored within what I didn’t realize was the [*old* website](https://data.cms.gov/provider-data) for the Centers for Medicare and Medicaid Services (CMS) data and public APIs. I had found a [Preventative Medicine Office Visit Costs](https://data.cms.gov/provider-data/dataset/0330-b6e0) dataset that peaked my interest that I had intended to combine with a [National Downloadable File](https://data.cms.gov/provider-data/dataset/mj5m-pzi6) regarding provider-level information. From here, I was going to use a postal code and state related API to provide as a link for the two CMS APIs so I would have a solid basis to dig into whether there could be a link between the number of providers available in any given area and the cost of preventative medicine for that area. However, as I worked through trying to set up the APIs to collect the data I was hoping to gather from these datasets, I noticed that I couldn’t seem to pull any data out of the CMS-related sources. I went digging for some documentation on why this might be and discovered CMS had hidden an important bit of information: *“The Public Dataset Endpoints on data.cms.gov have undergone a major overhaul and upgrade in the process of the recent migration of the platform on July 21, 2021 at 3 PM EST.”*2 The data I was looking to use was no longer available to use from the original CMS source and had yet to be re-published to the [new CMS data](https://data.cms.gov/search) storage website.

Since I needed to redesign the project based on public APIs I could access, I went to the new CMS data website and looked to see if there was any data I might be still interested in using from CMS. I discovered two new APIs I could still link together by state, which included a Medicare [COVID-19 Hospitalizations Trends](https://data.cms.gov/covid-19/medicare-covid-19-hospitalization-trends) data set as well as a [Medicare Monthly Enrollment](https://data.cms.gov/summary-statistics-on-beneficiary-enrollment/medicare-and-medicaid-reports/medicare-monthly-enrollment) data set. In addition to these two APIs, I decided to continue to use my original state and postal code mapping public API to bring them together and calculate the average number of Medicare Enrollments as well as the average number of COVID-19 hospitalizations for any given state.

Through this process I learned a valuable lesson regarding the use of publicly available APIs: sometimes you will hit dead ends and you must learn to adapt and pivot to find additional data sources to suit your needs, even if it means going almost all the way back to square one.

1. **Project Requirements and Tasks:**

For this project the main objective was to pull data from the three publicly available APIs, parse those data sets into something more meaningful and manageable to work with, create an externally facing custom API for users to call from, and successfully call on the custom API to ensure the resulting output was what I aimed to make available for users. I am going to walk through the process I used to achieve these goals.

**Call on Public APIs.** To achieve this goal, I first tested in Jupyter Notebook to ensure that the data would pull from each of the sources the way I wanted them to. After this, I created a module specifically to store the API URLs within my Visual Studio Code folder so that they would be contained and easy to find (Fig 3-1). Once this module was created, I created an additional module to store the function for calling on the APIs that would prefill my database (Fig 3-2).

A screen shot of a computer

Description automatically generatedFig 3-1: Storage of API URLs.

A screen shot of a computer program

Description automatically generatedFig 3-2: Function to call data from public APIs.

**Parse data for usage.** The next step that needed to be taken to make these data sets useable was to parse out the data and format it so that the state would be consistent from one data set to the next. While the zip code API and the Medicare Enrollments data sets both used a state abbreviation, the COVID-19 Hospitalizations used a full state name. I created a function to go through the states in the COVID-19 Hospitalizations data set and convert the full name states to abbreviations (Fig 3-3). Additionally, the zip codes file contained some zip codes that had leading zeros, which would have been truncated if not fixed. I handled this by evaluating each of the zip codes in the custom\_api.py file to see if the zip codes were less than values of 1,000 or 10,000, and if they were leading zeros were added to the zip code as they were converted to a string (Fig 3-4). Once all the data was parsed, it could be added to the database. For this, to prevent calling on the public APIs too many times, I have included a check to see whether the database has already been created. If the database has already been created, then a new one will not be created over top of it (Fig 3-5). Additional formatting was also added to this step to call out whether the database was already created or not using colored text when run in command prompt.

A screen shot of a computer

Description automatically generatedFig 3-3: Conversion of full state name to abbreviations (sample).

A computer code with many colorful text

Description automatically generated with medium confidenceFig 3-4: Adding leading zeros to truncated zip codes and conversion to strings.

A screen shot of a computer code

Description automatically generatedFig 3-5: Check to see if the database has already been created or not.

**Creation of the Custom API.** The next step included the creation of the CRUD (create/read/update/delete) structure for the custom API that users would be able to use to call their specified data from the database. Since I have set up this custom API to accept user-related information, it was necessary to include a Users Model within the database, as well (Fig 3-6). While the creation of a user is not necessary to pull the data from the database regarding the COVID-19 Hospitalizations and Medicare Enrollments by state, this functionality was built to provide some aggregated and more detailed data for the user that might be more pertinent to them, specifically. I created arguments for each type of request to be passed through to the custom API (Fig 3-7).

A screen shot of a computer code

Description automatically generatedFig 3-6: User Model with specified accepted inputs.

A screen shot of a computer program

Description automatically generatedFig 3-7: Arguments for processing user information

**Call the Custom API.** The arguments shown above are subsequently passed into the create, read, update, and delete requests for the custom API. The outputs for each vary but have been tailored to be applicable for each type of request depending on the expected inputs for each request, including prompts for whether there was no data found or if there is missing information in the request attempting to be passed to the custom API (Figs 3-8 through 3-14).

A computer screen shot of code

Description automatically generatedFig 3-8: Put (create) request for the custom API to create a new user.

A computer screen shot of a black screen

Description automatically generatedFig 3-9: Get (read) request to pull all COVID-19 hospitalization for each state.

A computer screen shot of a black background

Description automatically generatedFig 3-10: Get (read) request to pull all Medicare enrollment data for each state.

A computer screen shot of a black screen

Description automatically generatedFig 3-11: Get (read) request to view all user-level information.

A computer screen shot of a program code

Description automatically generatedFig 3-12: Get (read) request to pull user specified information for COVID-19 hospitalizations and Medicare enrollments by state, including the average number of each per zip code.

A computer screen shot of a program code

Description automatically generatedFig 3-13: Update request to update state and/or zip code information for a specified user.

A computer code on a black background

Description automatically generatedFig 3-14: Delete request to remove user data from the database.

**Testing the Custom API.** To save the length of this paper I will keep the testing output to a minimum and refer you to the video companion for this paper for more thorough testing results. However, I will mention that I did thoroughly test the API and have ensured that every piece functions as intended. The main goal of this project was to provide users with the number of COVID-19 hospitalizations for their specified state, the average number of COVID-19 hospitalizations per zip code in that state, the number of Medicare Enrollments for their specified state, and the average number of Medicare Enrollments per zip code in that state (Fig 3-15).

A screenshot of a computer

Description automatically generatedFig 3-15: Tested output for specific test user to show output.

1. **Reflections and Potential Improvements**

While working on this project there were certainly some challenges I encountered, as seen in the related work section of the paper. However, even as I completed the project and had successfully achieved the goal of creating a custom API for users to call on, I noticed some changes that I would have made if I had to go back through the process again.

The first of these changes would include a check to make sure that the provided zip code from the user was part of the state that they also specified for their get request. As of now, the user can put in *any* zip code, regardless of state, for their user create request and the output they receive from the get request is only applicable to the specified state. Ideally, I would like there to be a message that appears if the zip code provided by the user is not part of the state they have specified, which would read *“Zip code provided is not part of the specified state… Please update zip code to be part of your state…”*

Additionally, there were issues I had with pulling in all the Medicare Enrollment information from the public API provided by CMS. As of now the database only contains the Medicare enrollment counts alphabetically through the state of Kansas. While this was not the ideal output I wanted to achieve, it did allow me to test the functionality of the user-specific output since the *“No Medicare information”* output does appear.

While the results I was able to achieve were not necessarily 100% where I wanted them to be, I am very proud of the project I was able to complete. The creation of this project taught me a lot about how to create and use APIs efficiently and effectively, however, I know I could still stand to learn a lot. I also recognize that no API is ever going to be entirely what you want it to be. There can always be improvements and enhancements no matter how much work has been done, and there likely is no such thing as the “perfect” API despite what my perfectionist nature may tell me.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Citations:**

1 Amazon AWS. (n.d.). *What is restful API? - restful API explained - AWS*. What is a RESTFul API? <https://aws.amazon.com/what-is/restful-api/>

2 *Centers for Medicare & Medicaid Services Data API*. Centers for Medicare & Medicaid Services Data. (n.d.). <https://data.cms.gov/api-docs>