Logo

Description automatically generated 

**User’s Guide**

University of Maryland Global Campus

SWEN 670 – Team B

Spring Semester

Version 1.0 

April 1, 2023

|  |  |  |
| --- | --- | --- |
| **Version** | **Issue Date** | **Changes** |
| 0.1 | 3/27/2023 | Initial Version |
| 1.0 | 4/01/2023 | Milestone 4 Submission |

[1. Introduction 4](#_Toc131365931)

[1.1 Overview 4](#_Toc131365932)

[1.2 Project Documents 4](#_Toc131365933)

[1.3 Definitions, Acronyms, and Abbreviations 4](#_Toc131365934)

[2. Hardware Requirements 4](#_Toc131365935)

[3. Software Requirements 4](#_Toc131365936)

[4. Launching the Application 4](#_Toc131365937)

[4.1 Local System using Visual Source Code 4](#_Toc131365938)

[4.2 From the Visual Source Code Terminal 4](#_Toc131365939)

[4.3 Cloud Deployment 4](#_Toc131365940)

[5. Step-by-Step Guide 5](#_Toc131365941)

[5.1 Create Tour 5](#_Toc131365942)

[5.2 Upload Images to Server 7](#_Toc131365943)

[5.3 Compute Tour 8](#_Toc131365944)

[5.4 Panoramic Image Creation 8](#_Toc131365945)

[5.5 Determining Transition Hotspots 10](#_Toc131365946)

[5.6 Extracting Text from Images 10](#_Toc131365947)

[5.7 Detecting and Blurring Human Faces 12](#_Toc131365948)

[5.8 Applying the Glow Filter 13](#_Toc131365949)

[5.9 Adding/Editing/Deleting Informational Hotspots 17](#_Toc131365950)

[5.10 Search Text 18](#_Toc131365951)

[6. Troubleshooting 19](#_Toc131365952)

[6.1 Problem: Computing the tour takes a long time 19](#_Toc131365953)

[6.2 Problem: Image not found 19](#_Toc131365954)

[6.3 Problem: Neighbors not found 20](#_Toc131365955)

# Introduction

## Overview

The purpose of the ViroTour backend application is to support the frontend application created by ViroTour Team A. All Application Programming Interface (API) calls will be detailed in this User Guide.

# Hardware Requirements

* 4vCPU: vCPU refers to virtual central processing unit. Each virtual machine (VM) is assigned one or more vCPUs within a cloud-based environment. The VM OS views every vCPU as a unique physical CPU kernel.
* RAM: 7 GB
* Disk space: 10 GB
* Operating systems: Windows® 10 or later, macOS\*, and Linux\*
* Environment platforms: Azure, AWS
* Cloud scalability: 3 instances (Cloud scalability is the increase or decrease in cloud resources to respond to load requests)

# Software Requirements

Postman is a software which is recommended for executing the requests. The link for installing the software is available here: <https://www.postman.com/downloads/>

# Launching the Application

## Local System using Visual Source Code

1. Click the “Run.py” file, and make sure it is selected as the entry point
2. From the menu, click the “Run” menu
3. Select either “Start Debugging”, or “Run without Debugging”
4. Open the browser, and go <http://127.0.0.1:5001> , or <http://localhost:5001>

## From the Visual Source Code Terminal

1. Activate the terminal/command line by clicking the “Terminal” menu
2. Run the “python run.py” command

## Cloud Deployment

1. Get the Web App service URL, such as <https://virotour2023-flask-server.azurewebsites.net/>
2. To Access the end points: {HostURL}/api/{end point}

# Step-by-Step Guide

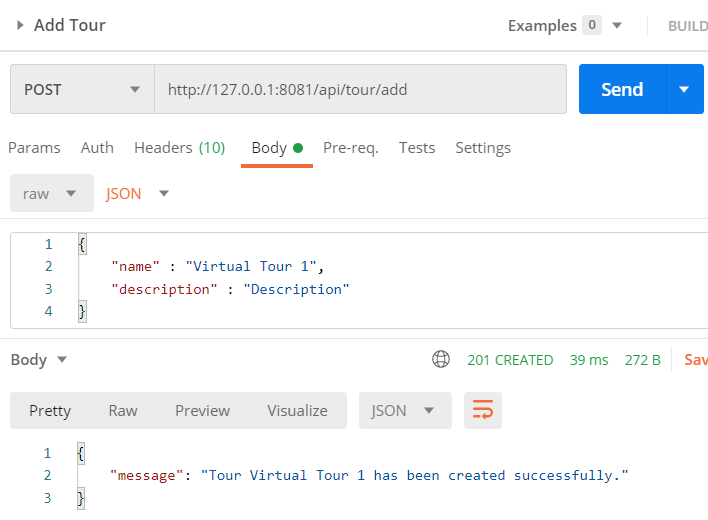
## Create Tour

To get started with the virtual tour creation software, you must first create a tour by giving it a name and a description.

**[POST] /api/tour/add**

Parameters:

* name: name of tour
* description: description of the tour

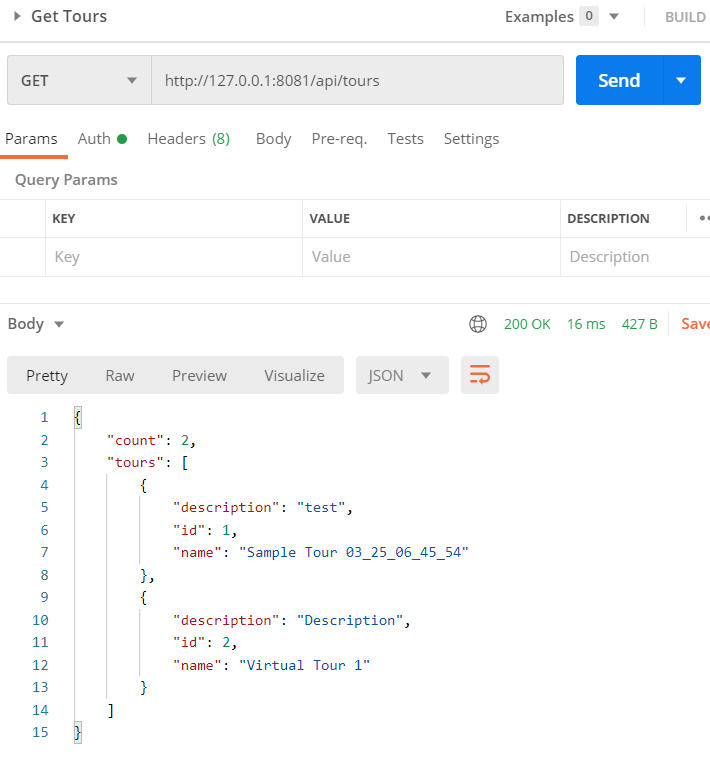


**Figure 5.1.1 - POST API Request and Response to Add a Tour.**

You may get a list of all of the available tours by sending a request as shown below:

**[GET] /api/tours**

Parameters: None



**Figure 5.1.2 - GET Request to Retrieve All Tours.**

## Upload Images to Server

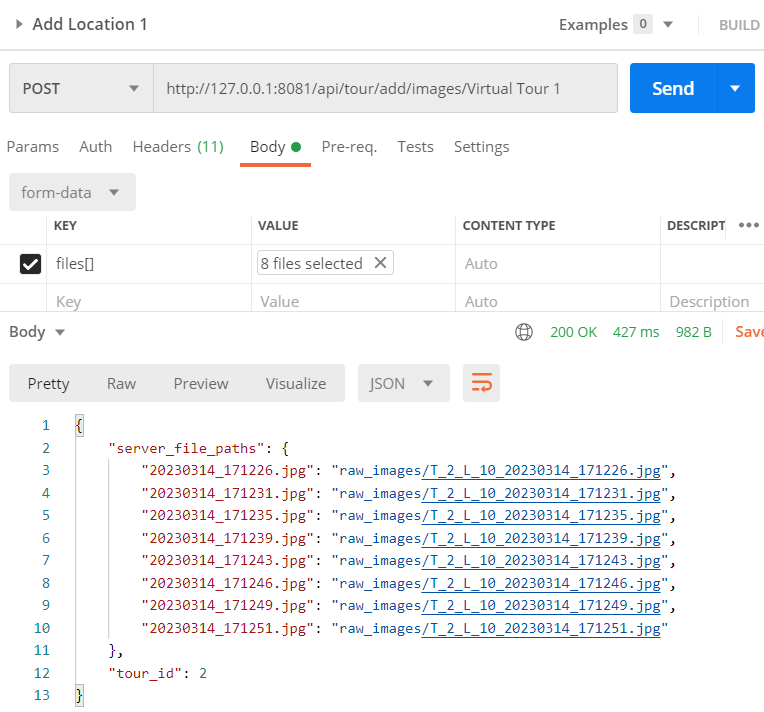
Once your tour has been created, you can start adding images to the server. You may upload a sequence of images, which represents a single location. These images will later be stitched together into panoramic images.

To add images, use the request below:

**[POST] /api/tour/add/images/<string:tour\_name>**

Parameters:

* body:
  + files[]: <file1.png>, <file\_2.png>, <file\_3.png>



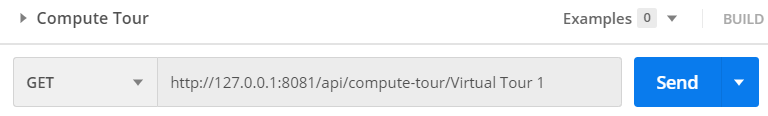
**Figure 5.2.1 - POST API Request and Response to Add One Location’s Pictures to the Server.**

## Compute Tour

Now that you have added all of the locations, it is time to let the server calculate the tour.

**[GET] /api/compute-tour/<string:tour\_name>**

Parameters: None



**Figure 5.3.1 – Compute Tour**

Please note, this may take a while! The expected runtime is roughly 10 minutes per 100 images. Behind the scenes, the server is executing a four-step algorithm to calculate the tour:

1. Compute panoramic for all locations.
2. Compute neighbors.
3. Blur faces.
4. Extract text.

The results are written to the database so that you can interact with it later.

## Panoramic Image Creation

When the tour is computed, you may access the panoramic images. Use the request below to list details of the tour.

**[GET] /api/get-tour/<string:tour\_name>**

Graphical user interface, text

Description automatically generated

**Figure 5.4.1 – Request to Retrieve Tour**

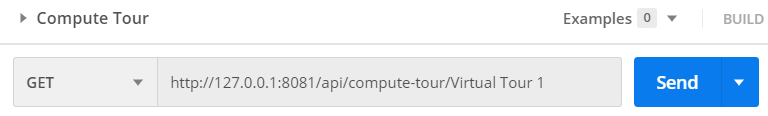
This shows the location ID’s within a tour. Use the request below to retrieve the panoramic image.

**[GET] /api/tour/images/panoramic-image/<string:tour\_name>/<int:location\_id>**Graphical user interface, text

Description automatically generated

**Figure 5.4.2 – Request to Retrieve Panoramic Images**

## Determining Transition Hotspots



**Figure 5.5.1 - GET API Request of a Tour Going Through Computations (Panoramic Image Stitching, Computing Hotspots, Blurring Faces, and Extracting Text).**

At the stage of computing transition hotspots, for each location, the API returns one neighbor. (Figure 5.5.2).

**[GET] /api/tour/get-tour/Sample%20Tour%2003\_25\_06\_45\_54**



**Figure 5.5.2 - GET API Request and Response for Neighbors.**

The x and y represent the position of the transition hotspot on the current image. location\_id represents the id of the following image indicated by this hotspot.

## Extracting Text from Images

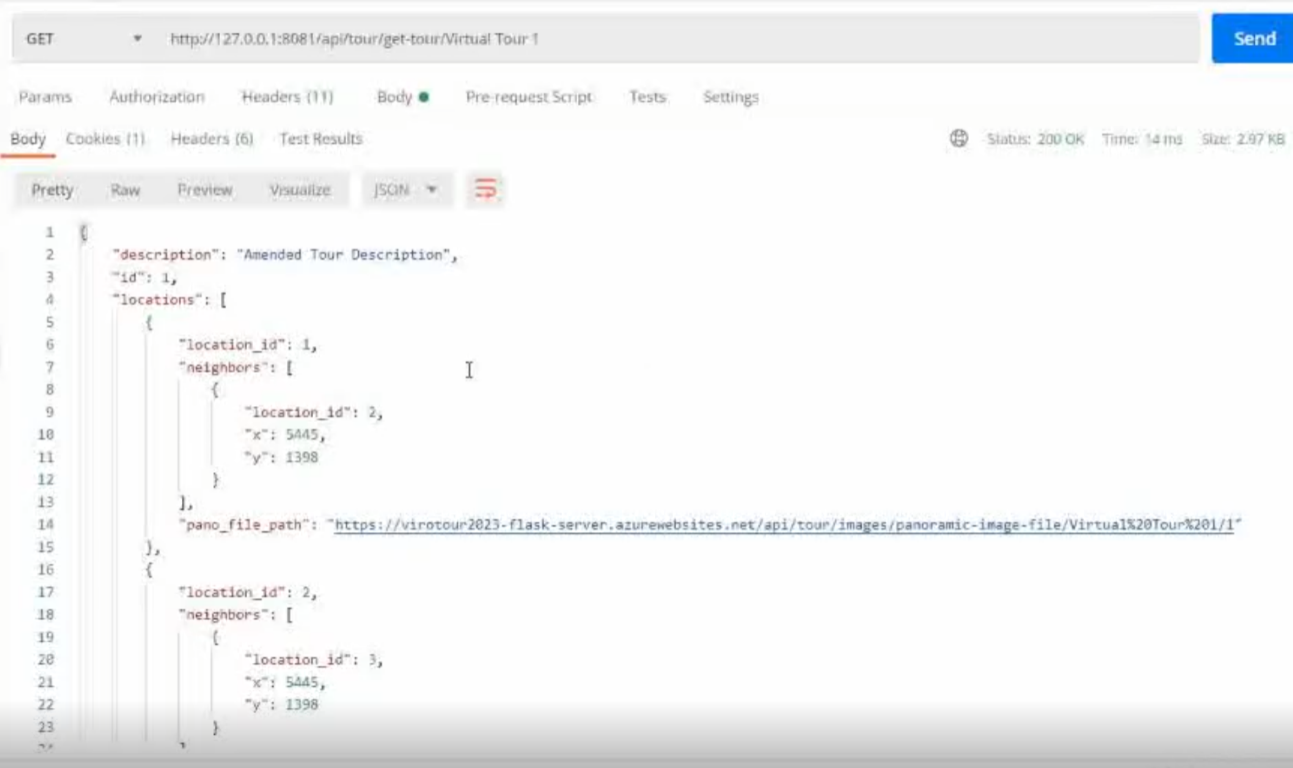
There is no API call to manually execute the text extract by itself. However, the text extraction functionality is computed when the panoramic image is processed (e.g., in section 6.2 above). At time of computation, the text is extracted from the image and stored into the location object. Assuming a tour has been created (e.g., with the name “Virtual Tour 1”), call the following API:

**[GET] /api/tour/get-tour/Virtual Tour 1**

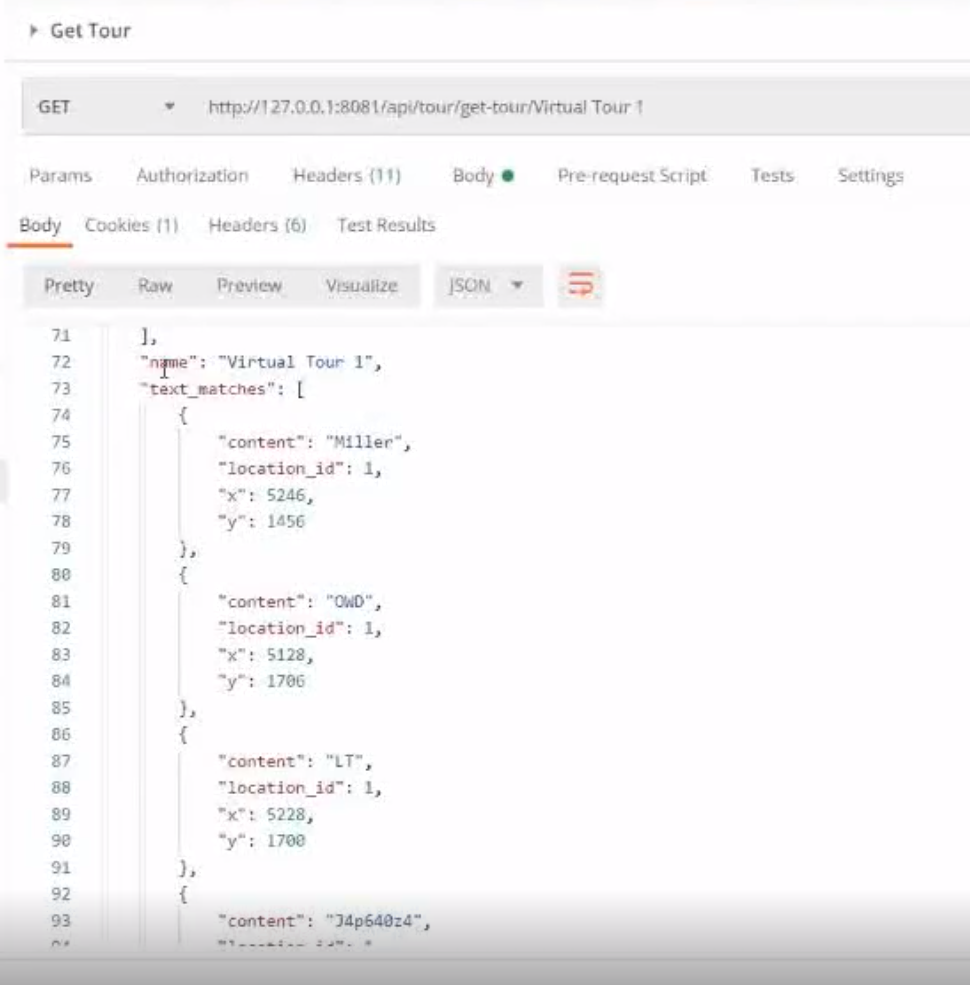
In doing so, the response will return an object that contains the following:

* Description (string): description of the tour
* Id (int): the number of the tour (unique identifier of all created tours)
* Locations (array of objects): all locations / transitional hotspots of a given tour
* Name (string): name of the tour
* Text\_matches (array of objects): all the extracted texts that were found at a given location

Below is an example of the response when the request tries to retrieve the tour named “Virtual Tour 1” that was previously created:



**Figure 5.6.1 - GET API Response of a Tour (Showcasing Description, Id, and Locations Properties).**



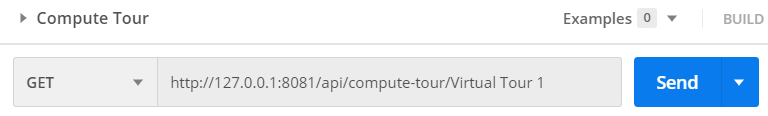
**Figure 5.6.2 - GET API Response of a Tour (Showcasing Name and Text Matches Properties).**

## Detecting and Blurring Human Faces

The detect and blur human faces feature finds human faces and applies a gaussian blur over each face. To apply a gaussian blur over each detected face in the panoramic image, submit a GET request to the REST API endpoint:

**[GET] /api/compute-tour/<string:tour\_name>**

This call will return the panoramic images along with their locations in json format. This is the call that is performed after all the initial images have been uploaded. The compute-tour/tour\_name call also handles computing the panoramic image (6.2), computing neighbors for transition hotspots (6.3), and computing the text from images (6.4). During this call all panoramic images have their detected human faces blurred and the new panoramic is saved.

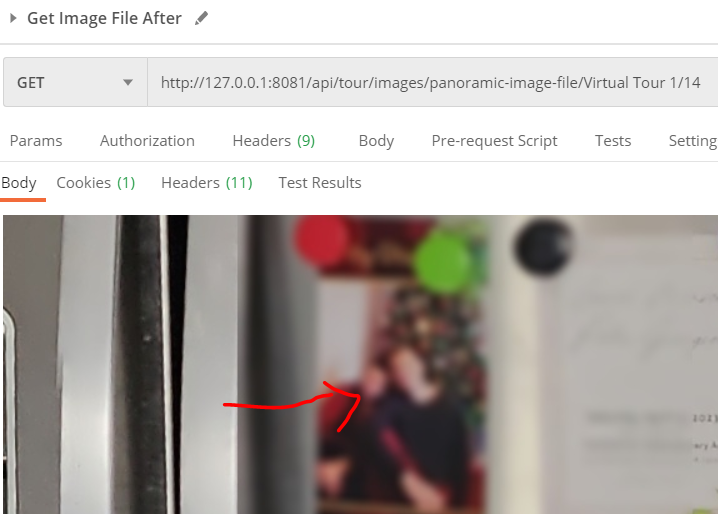


**Figure 5.7.1 - GET API Request of a Tour Going Through Computations (Panoramic Image Stitching, Computing Hotspots, Blurring Faces, and Extracting Text).**

Each tour’s panoramic image can be viewed by passing in the *tour\_name* and the *location\_id*.

**[GET] /api/tour/images/panoramic-image-file/<string:tour\_name>/<int:location\_id>**

Verify that the panoramic image has detected and blurred all human faces. Note that there could be false positives or human faces that are not blurred, depending on the tunned sensitivity of the algorithm.



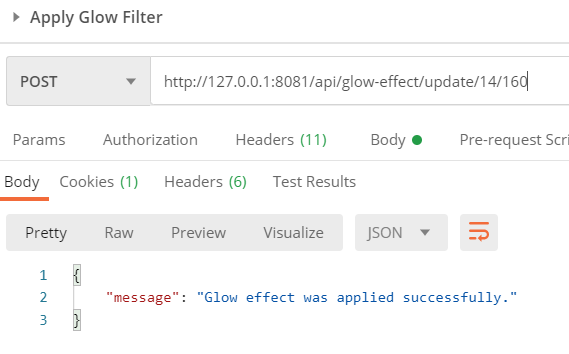
**Figure 5.7.2 - GET API Request and Response of Retrieving of a Computed Panoramic Image at a Specified Location (That Happens to Have Facial Blur).**

## Applying the Glow Filter

The glow effect adjusts the contrast and brightness of an image. To apply a glow filter to a panoramic image, submit a POST request to the REST API endpoint.

**[POST] /api/glow-effect/update/<int:location\_id>/<int:brightness\_value>**

The parameters that this endpoint expects are two integers; location\_id and brightness\_value. The brightness value is any integer ranging from 1 – 255. A high brightness\_value will render a brighter image. When this endpoint is processed, the brightness value is saved and associated with the location\_id in the database. The endpoint will send a success message if the brightness value was saved successfully.

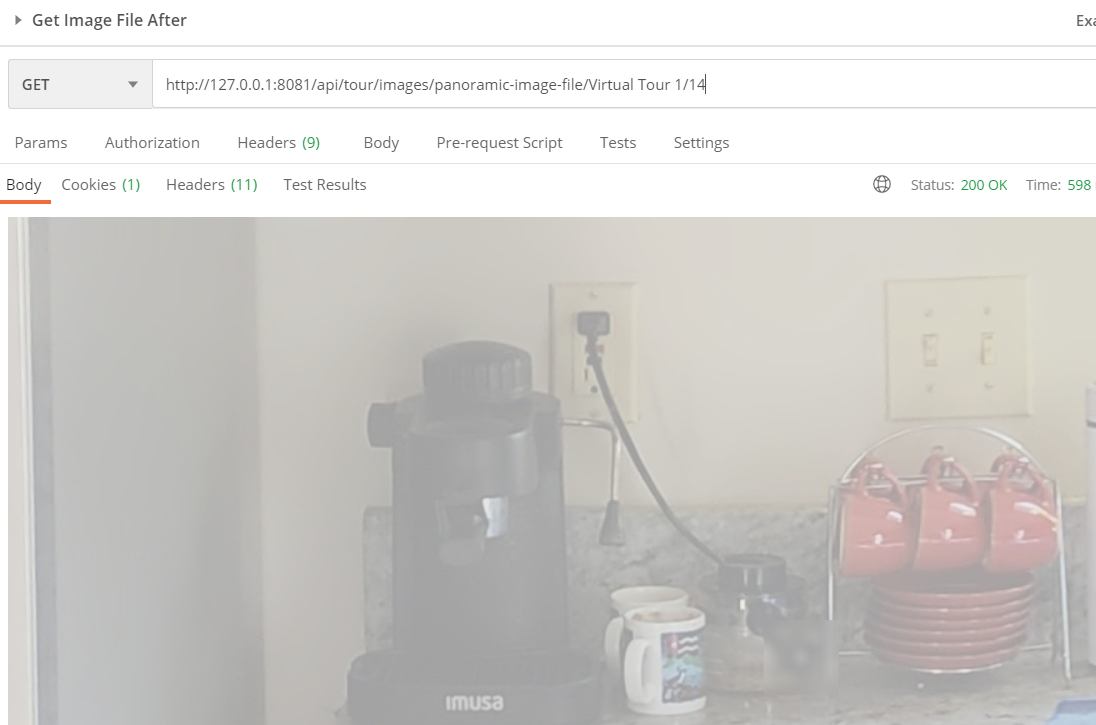


**Figure 5.8.1 - GET API Request and Response of a Applying a 160 Glow Filter Value to a Specific Location.**

To view the panoramic image with the glow filter applied, submit a GET request to the REST API endpoint:

**[GET] /api/tour/images/panoramic-image-file/<string:tour\_name>/<int:location\_id>**

To retrieve the panoramic image, pass the tour\_name and location\_id to the API endpoint.



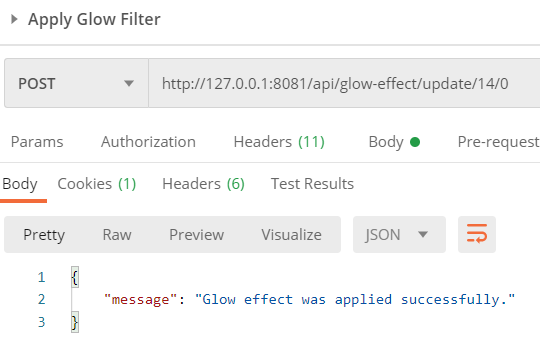
**Figure 5.8.2 - GET API Request and Response of a Panoramic Image at a Location within a Tour.**

If the brightness of the glow filter is unsatisfactory or the panoramic image needs to be restored to the original, submit a POST request to the REST API endpoint:

**[POST] /api/glow-effect/update/<int:location\_id>/**

To reset the panoramic image, pass the location\_id and a value of zero for brightness to the endpoint. The endpoint will send a success message if the brightness value was saved successfully.

Reset:

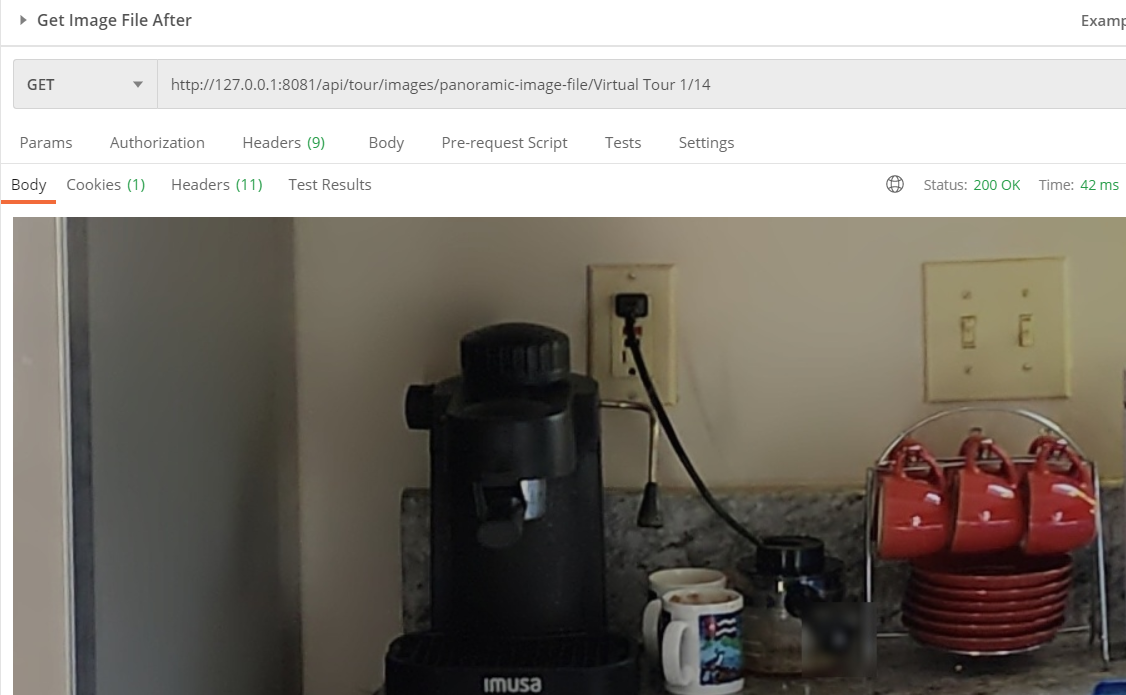


**Figure 5.8.3 - POST API Request and Response of Resetting a Glow Effect on a Panoramic Image at a Specified Location.**

To view the panoramic image, submit a GET request to the REST API endpoint:

**[GET] /api/tour/images/panoramic-image-file/<string:tour\_name>/<int:location\_id>**

To retrieve the panoramic image, pass the tour\_name and location\_id to the API endpoint.



**Figure 5.8.4 - GET API Request and Response of Retrieving a Panoramic Image in a Tour.**

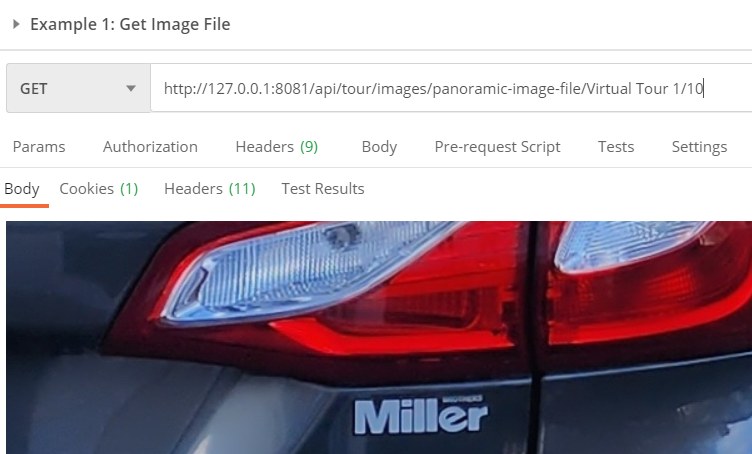
## Adding/Editing/Deleting Informational Hotspots

The adding, editing (e.g., moving or renaming info text), and deleting of informational hotspots have not been implemented in the current milestone of the project, and will later be developed in future releases. As a result, there are no instructions to implement these three routes and functionalities at this time in development, and would be future efforts.

## Search Text



**Figure 5.10.1 - GET API Request and Response of Searching for Text in a Tour.**



**Figure 5.10.2 - GET API Request and Response of Retrieving a Panoramic Image in a Specified Location within a Tour.**

# Troubleshooting

## Problem: Computing the tour takes a long time

|  |  |
| --- | --- |
| “Sending request..” stuck on screen. |  |
| Resolution: Check Server Logs, it may still be processing. |  |

## Problem: Image not found

|  |  |
| --- | --- |
| Image location. |  |
| “Image not Found” error. |  |
| Resolution: | The panoramic stitching probably failed. Try uploading images with better quality |

## Problem: Neighbors not found

|  |  |
| --- | --- |
| No neighbors for a given location. |  |
| Resolution: The hotspot detection failed to find a suitable neighbor. Please reach out to the development team for assistance. |  |