Team: Shroom ID

Shroomid.net

Software Design Document

Paul Smith
Haowen Yong
Ahmed Alweheiby
Joseph Santiago

Introduction **3**
Purpose 3
Scope 3
Overview 3

System Overview **3**

System Architecture **4**
Architectural Design 4
Design Rationale 4

Data Design **4**
Data Description 4

Human Interface Design **6**
Interface Overview 6
Screen Images 6
Default Home Page 6
Post Upload and Submit 7
Successful Results 7
Failure Results 7
Screen Object and Actions 8

1. Introduction

1.1. Purpose

This software design document is intended to describe shroom.net architecture.

1.2. Scope

Shroomid.net is designed to be a web app that can provide mushroom identification and upon identification tell the user if the mushroom is poisonous. The web app will implement a user interface that will allow users to upload mushroom photos for identification. Upon a successful upload, the system will run the image through a TensorFlow model and return to the results of the identification process.

1.3. Overview

In this document you will find the following sections:

- Introduction
- System Overview
- System Architecture
- Component Design
- Human Interface Design

2. System Overview

The idea of this project came to fruition from our team wanting to do a project that used machine learning and Professor Morrsion pointing our team in a direction where she thought there was currently a lack of products. Websites and phone applications already offer mushroom identification but most of the market uses a best guess approach where users pick similar looking mushrooms till a classification could be made. Our web app will retrain the Google Inception model to give a species identification so we could tell our users a particular mushroom is poisonous.

3. System Architecture

3.1. Architectural Design

The user begins using our web app when they go to the address “shroomid.net” in their browser. Our web app consists of a single page that updates with results. There are 2 buttons that the user can interact with. The first is the button that will load their devices file system so they can choose a mushroom image they want to upload. The second button sends a put command to an s3 bucket and, on confirmation of a successful upload, will send out a json with the jpg’s s3 file location through our API. Our system is serverless, so the processing of the image happens through a AWS Lambda function. When the Lambda function has finished processing the image, the results are sent back to the browser to be displayed.

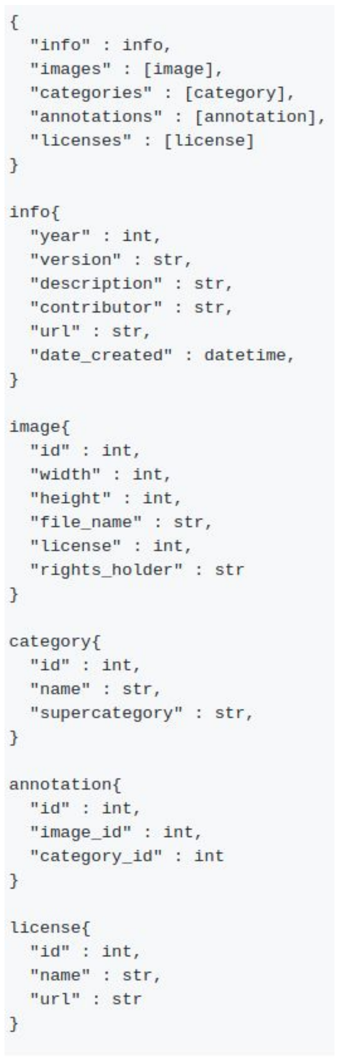
3.2. Design Rationale

Our web app design involved minimal user interactions with each transaction, but we wanted to be able to support a large number of concurrent users. By using a simple serverless web service architecture the logic component of our software would scale automatically. Each API call to our system will invoke its own Lambda function. This removes the need to manage multiple EC2 servers or worrying about a queuing service for our system. The main drawback is that Lambda functions only support CPU accelerations. As the size and complexity increase, GPU acceleration may be needed for our TensorFlow inference.

4. Data Design

4.1. Data Description

All communications between components of our system are done in JSON format. Information form our chosen data set (source:https://github.com/visipedia/fgvcx_fungi_comp#data) is saved in the following JSON format:



```
{
  "info" : info,
  "images" : [image],
  "categories" : [category],
  "annotations" : [annotation],
  "licenses" : [license]
}

info{
  "year" : int,
  "version" : str,
  "description" : str,
  "contributor" : str,
  "url" : str,
  "date_created" : datetime,
}

image{
  "id" : int,
  "width" : int,
  "height" : int,
  "file_name" : str,
  "license" : int,
  "rights_holder" : str
}

category{
  "id" : int,
  "name" : str,
  "supercategory" : str,
}

annotation{
  "id" : int,
  "image_id" : int,
  "category_id" : int
}

license{
  "id" : int,
  "name" : str,
  "url" : str
}
```

Trained TensorFlow model information is saved in the .pb format. With all of the graph definitions and weights of the model in the protobuf file it is the only piece of the trained model we need to reload it into our software. Pictures are temporarily stored in S3 buckets. Each file location is no longer accessible to the system after the results are given. No searching mechanism is implemented to retrieve previous files and the file name is not stored anywhere other than JSON communications which are also never saved by the system.

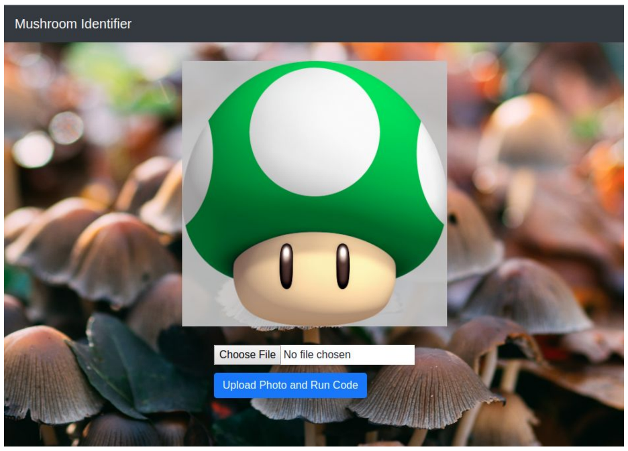
5. Human Interface Design

5.1. Interface Overview

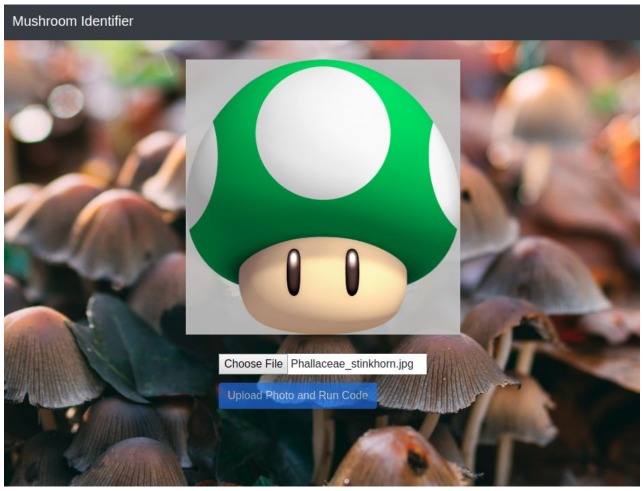
The user has the option of uploading a photo. The “Choose File” button will open up the file system for the device they are currently using. It is up to the user to navigate their own filesystem and find the photo they wish to upload. Once they have chosen the file, the name of the file will be displayed in the information box next to the button. The next step is to click on the “Upload Photo and Run Code” button. This will cause the transparency of the button to increase indicating that the user is no longer able to press it again. If the results are above the minimum threshold then the image of the photo will be shown along with the results of the identification. If the results were below the minimum threshold then an image will show the default failure img along with the phrase “Scores were too low” in the results box. From either of these results states the user can begin the process again by starting the steps over again. When the submit button is clicked this time it will function like it had before but will also reset the img and clear the previous results.

5.2. Screen Images

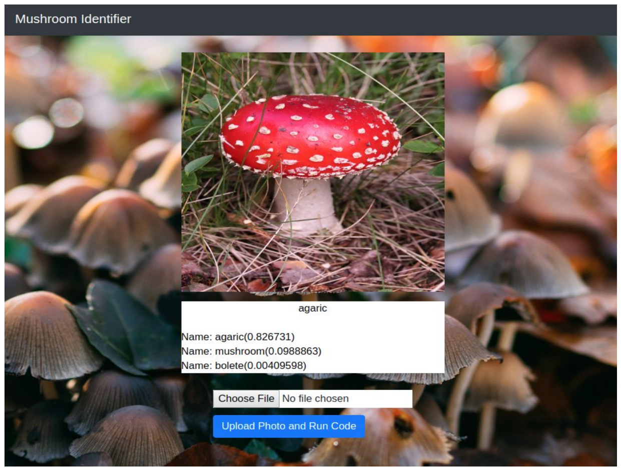
5.2.1. Default Home Page



5.2.2. Post Upload and Submit


Mushroom Identifier
Choose File Phallaceae_stinkhorn.jpg
Upload Photo and Run Code

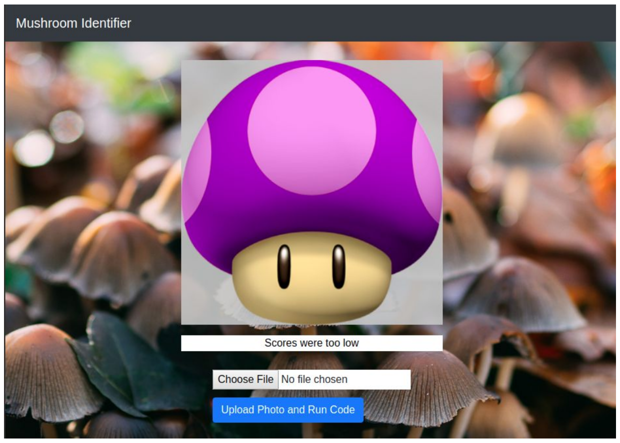
5.2.3. Successful Results



```
agaric

Name: agaric(0.826731)
Name: mushroom(0.0988863)
Name: bolete(0.00409598)
```

5.2.4. Failure Results



```
Scores were too low
```

5.3. Screen Object and Actions

The “Default Home Page” image shows the default state of the buttons, images and results box. In the “Post Upload and Submit " photo the user has selected an image for submission and pressed the “Upload Photo and Run Code” button. The transparency of the upload button can be seen indicating that it will no longer function. The “Successful Results” image shows the image that was submitted by the user and below the results of the top 3 selections are shown. This photo also shows that the upload button has regained its functionality. The final photo, “Failure Results”, shows the picture of the default failure image and the text in the result box telling the user that the “Scores were too low”. As with the previous results image the returned functionality of the submit button can be seen.