1. **INTRODUCTION**

**OVERVIEW**

This report discusses the result of the work done in development of “Caption Generator” on Machine Learning Platform. The project aims at the development of an application that identifies the action portrayed in the given image.

**PURPOSE**

You saw an image and your brain can easily tell what the image is about, but can a computer tell what the image is representing?

With the advancement in Deep learning techniques and availability of huge datasets and computer power, it is possible to build models that can generate captions for an image.

This project involves **Computer Vision** and **Natural Language Processing** concepts to recognize the context of an image and describe them in a natural language like English.

The objective of the project is to build a working model of Caption Generator by implementing CNN with LSTM.

The basic working of the application is that the features are extracted from the images using pre-trained **VGG16 model** and then fed to the **LSTM model** along with the captions to train.

The trained model is then capable of generating captions for any images that are fed to it.

1. **LITERATURE SURVEY**

**2.1 BACKGROUND AND MOTIVATION**

**Existing Systems**

* **Captionbot.ai**
  1. It is a product of Microsoft.
  2. It is an ML application that can understand the content of any image.
  3. When a person upload a photo, it is sent to Microsoft for image analysis to return a caption
  4. The application will not store or publish the images anywhere.
  5. It uses Computer Vision API and Emotion API
* **How-Old.net**
  1. It is also a product of Microsoft.
  2. It estimates the age of the person in the given image.
  3. The age is generated as a caption
* **TwinsOrNot.net**
  1. It identifies whether the image has twins or not

**2.2 Research Papers**

* Title: Automatic Description Generation from Images: A Survey of Models, Datasets, and Evaluation Measures
* Authors: Raffaella Bernardi, Ruket Cakici, Desmond Elliott, Aykut Erdem, Erkut Erdem, Nazli Ikizler-Cinbis, Frank Keller, Adrian Muscat, Barbara Plank
* Publication: Cornell University
* Summary: Here in this paper, the authors have described their project as a challenging problem as the model was not working properly with natural images that have recently received a huge amount of attraction from the computer vision and natural language processing communities. Also, they have classified the existing approaches based on how they conceptualized this problem. They have helped in reviewing the detailed description of existing models along with their advantages and disadvantages.
* Title: An Empirical Study of Language CNN for Image Captioning
* Authors: Jiuxiang Gu, Gang Wang, Jianfei Cai, Tsuhan Chen
* Publication: 2017 IEEE International Conference on Computer Vision
* Summary: In this paper, the effectiveness of their approach is validated on two datasets: Flickr30K and MS COCO. The extensive experimental results show that their method outperforms the vanilla recurrent neural network-based language models and is competitive with state-of-the-art methods.
* Title: A Comprehensive survey of Deep Learning for Image Captioning
* Authors: MD. Zakir Hossain, Ferdous Sohel, Mohd Fairuz Shiratuddin, Hamid Laga
* Publication: Murdoch University, Australia
* Summary: In this paper, Although deep learning-based image captioning methods have achieved remarkable progress in recent years, a robust image captioning method that is able to generate high-quality captions for all images is yet to be achieved.
* Title: What is the Role of Recurrent Neural Networks (RNNs) in an Image Caption Generator?
* Authors: Marc Tanti, Albert Gatt, Kenneth P. Camilleri
* Publication: Cornell University
* Summary: In this paper, a recurrent neural network (RNN) is typically viewed as the primary ‘generation’ component. The authors suggest that the image features should be ‘injected’ into the RNN. They have viewed the RNN algorithm as only encoding the previously generated words. According to the authors, the RNN algorithm should only be used to encode linguistic features and that only the final representation should be ‘merged’ with the image features at a later stage. The paper compares these two architectures. As suggested RNNs are better viewed as encoders, rather than generators.

**2.3 OBJECTIVE**

The final goal of the project is:

A Machine Learning application that is used to portray the scenario depicted in the images uploaded by the user. This is possible as the application initially identifies the action portrayed in the given image.

The generated caption will describe about the image that will say what kind of actions is taking place in it.

**2.4 METHODOLOGY**

To implement the above goals, the following methodology needs to be followed:

1. Specifying the application and various components of the architecture.
2. Specifying the bindings between the various python packages and Machine Learning models.
3. Specifying the server ports between the modules of flask.
4. Analysis: Extracting the required data for analysis and then doing the analysis.

**2.5 ABOUT TOOLS AND TECHNOLOGIES**:

**APPLICATION DEVELOPMENT TECHNOLOGIES**:

This application is built using Python, Flask, HTML, CSS, and Bootstrap.

**Python** is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace.

**Flask** is a micro web framework written in Python. It is classified as a micro framework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions.

**HTML (Hypertext Markup Language)** is a micro web framework written in Python. It is classified as a micro framework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions.

**CSS (Cascading Style Sheets)** is a style sheet language used for describing the presentation of a document written in a markup language like HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.

**Bootstrap** is a free and open-source CSS framework directed at responsive, mobile-first front-end web development. It contains CSS- and JavaScript-based design templates for typography, forms, buttons, navigation, and other interface components.

**INTEGRATION TOOLS**:

**GIT AND GITHUB**:

Git is a distributed version-control system for tracking changes in source code during software development. It is designed for coordinating work among programmers, but it can be used to track changes in any set of files. Its goals include speed, data integrity and support for distributed, non-linear workflows.

GitHub is a web based hoisting service for version control using Git. It is mostly used for computer code. It offers all of the distributed version control and source code management functionality of Git as adding its own features.

**HEROKU**:

**Heroku** is a container-based cloud Platform as a Service (PaaS). Developers use **Heroku** to deploy, manage, and scale modern apps. Our platform is elegant, flexible, and easy to use, offering developers the simplest path to getting their apps to market.

1. **HARDWARE AND SOFTWARE REQUIREMENTS**

Based on analysis and literature survey regarding the present difficulties faced by the existing systems like Captionbot.ai, How-Old.net and TwinsOrNot.net as although Captionbot.ai does generate captions but it does not allow users to upload images as per their desires. How-Old.net does not give accurate predictions for all uploaded images and TwinsOrNot.net is a deprecated system.

Therefore, with the system of mine I am trying to generate captions for any images of any resolution once the users upload them.

**3.1 HARDWARE REQUIREMENTS:**

RAM: 8GB and above

Hard disk: 120GB and above

Processor: Intel i3 and above

**3.2 SOFTWARE REQUIREMENTS**:

Operating System: Windows 10 / Ubuntu

Front end: HTML5, CSS3, Bootstrap

Back end: Flask 1.x

Language: Python 3.7

Storage (Dataset): Google Drive

IDE: Jupyter Notebook

Cloud Deployment: Heroku Deployment

Other Technologies used: Git and GitHub

1. **SOFTWARE REQUIREMENTS SPECIFICATION**

**4.1 FUNCTIONALITIES**:

* **User Perspective**
  + User launches the application.
  + User uploads the images for which the captions are to be generated.
  + User clicks on the submit button.
  + Within a matter of seconds, the ML model recognizes and process the content of an image.
  + The result will be displayed on the page.

**Model Development and Deployment**

* + Data Gathering
    - Images for training the model
    - Corpus for Image captions
  + Pre-processing
    - Corpus – Removal of stop words, punctuation marks, digits
    - Generating the bag of words
    - Mapping of images with the Corpus
  + Feature extraction from images
  + Object identification
  + Model Generation using CNN
  + Building LSTM (Long Short Term Memory) model.
  + Validating the Model
  + Deploy the ML model in the web application

1. **SYSTEM DESIGN**

**5.1 DATA FLOW DIAGRAM**

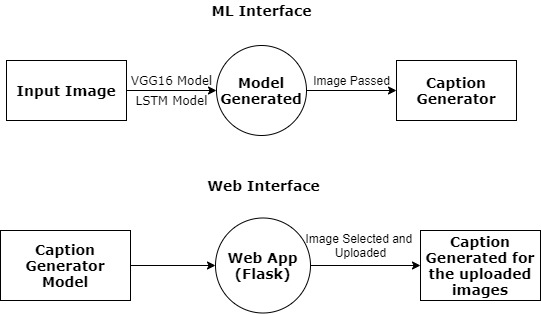


Figure 5.1.1 DFD Level 0

Our project consists of only one fold:

1. Visually impaired users: Users who would like to generate captions for the images that they had captured.

**5.2 PROCESS FLOW DIAGRAM**

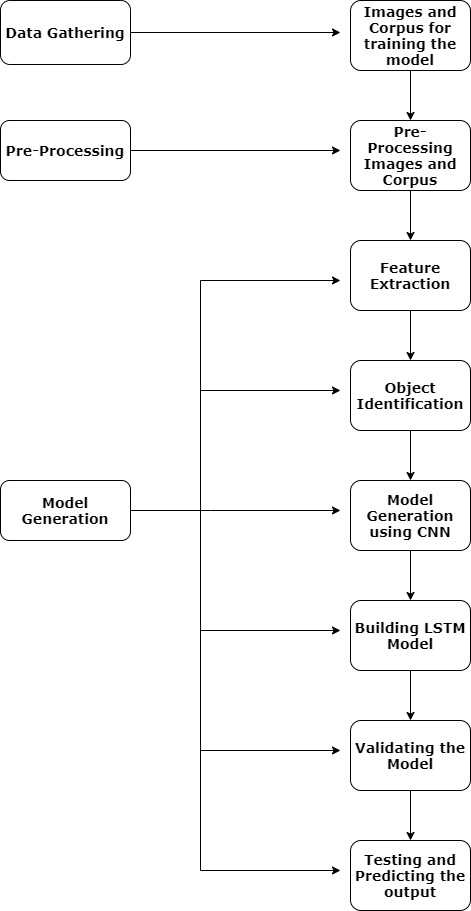


Figure 5.2.1 Process Flow Diagram

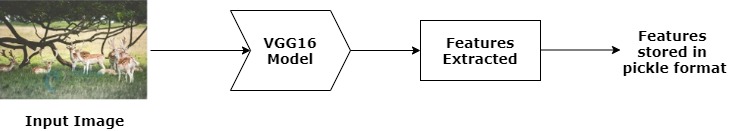
1. **DETAILED METHODOLOGY**

**6.1 How Images are Pre-Processed**

CNN model could be used directly as part of a broader image caption model. The problem is, it is a large model and running each photo through the network every time we want to test a new language model configuration is redundant. Instead, we can pre-compute the “photo features” using the pre-trained model and save them to file. We can load these features later and feed them into our model as the interpretation of a given photo in the dataset. It is no different to running the photo through the full VGG model.

This optimization will make training our models faster and consume less memory.

Once features are extracted, they are stored in pickle format as a file for future use.



**Figure 6.1.1 How Images are Pre-Processed**

**6.2 How Captions are Pre-Processed**

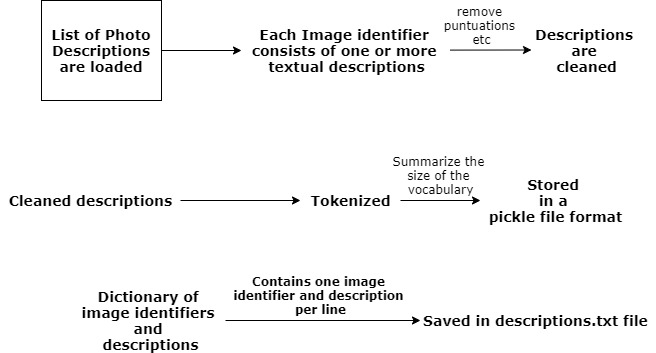
The dataset contains multiple descriptions for each photograph and the text of the descriptions requires some minimal cleaning. Once the file containing all of the descriptions of the images, the objective is to find the unique identifier of each image. This identifier is used on the photo filename and in the text file of descriptions.

Once the list of photo descriptions are loaded, it will return a dictionary of photo identifiers to descriptions. Each photo identifier maps to a list of one or more textual descriptions. Then the descriptions are needed to be cleaned. Then the cleaned descriptions are tokenized so that they are easier to work with.

The text is cleansed in order to reduce the size of the vocabulary of words. It is done to convert all words to lowercase, remove all punctuations, remove all words that are one character or less in length and remove all words with numbers in them. Once cleaned, we can summarize the size of the vocabulary.

Ideally, a small and expressive vocabulary is expected.

Finally, the dictionary of image identifiers and descriptions are saved to a new file called descriptions.txt with one image identifier and description per line.

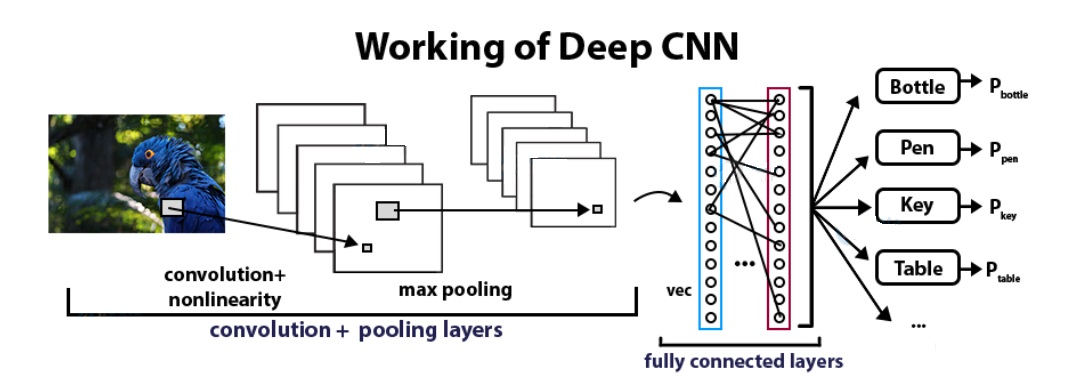


**Figure 6.2.1 How Captions are Pre-Processed**

**6.3 CNN (Convolutional Neural Networks)**

**CNN Algorithm** – With the help of CNN Algorithm, features are extracted from the images with the help of pre-trained VGG16 model.

CNN Algorithm is being used in this project as they are specialized deep neural networks which can process the data that has input shape like a 2D matrix. Images are easily represented as a 2D matrix and CNN is very useful in working with images.

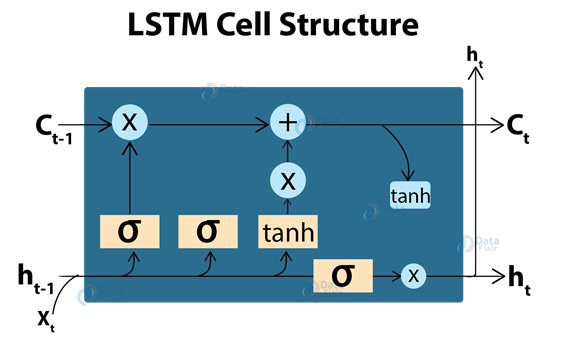


**Figure 6.3.1 Working of CNN Algorithm**

**6.4 LSTM (Long Short Term Memory)**

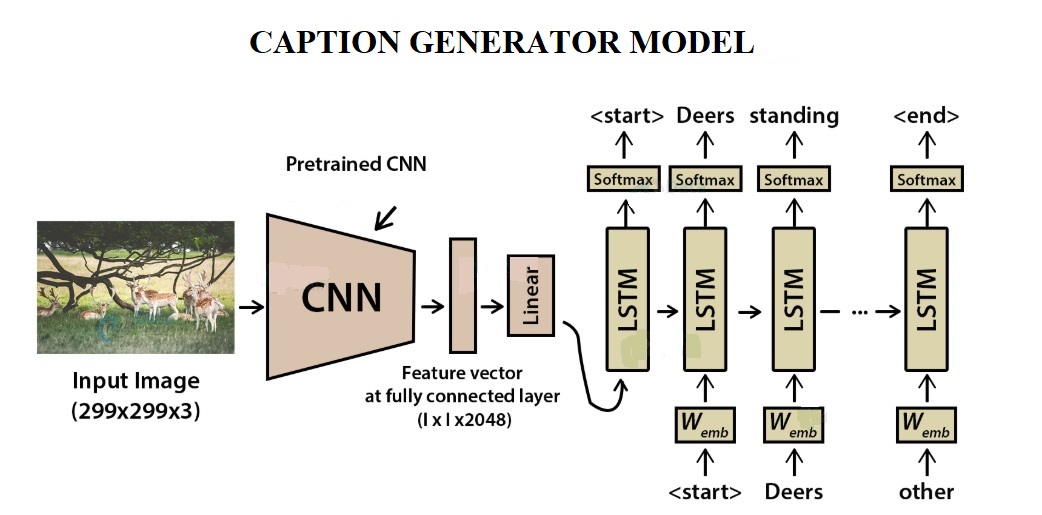
**LSTM Algorithm** – With the features being extracted from the images with the help of CNN Algorithm, they are now fed into the LSTM model that will be responsible for generating the image captions.

**LSTM (Long Short Term Memory)** is a type of RNN that is well suited for sequence prediction problems. With the help of LSTM, it can predict what the next word will be. LSTM will use the information from CNN to help generate a description of the image.



**Figure 6.4.1 LSTM Cell Structure**

**6.5 Model Generation**

****

**Figure 6.5.1 Working of Caption Generator**

Once the images and corpus are pre-processed, the images are passed through Pretrained CNN model (VGG16 Model) which has been used in this project. With the help of VGG16 Model, features are extracted from the images and stored in a pickle format file called ‘features.pkl’. Then with the help of LSTM Algorithm, ‘features.pkl’ and the pre-processed corpus of the images stored as ‘descriptions.txt’ along with the token of words stored in ‘tokenizer.pkl’ are passed through the LSTM Model which starts generating the final ML Model with the help of Epochs for generating captions for the images. Among the generated epochs of models, the best model is selected for development of application based on BLUE Score that helps in the evaluation of the translated text.

1. **IMPLEMENTATION**

**7.1 SAMPLE SOURCE CODE**

**Index.html**

<!DOCTYPE html>

<html>

<head>

<title>Caption Generator</title>

<link rel="stylesheet" href="https://unpkg.com/purecss@1.0.1/build/pure-min.css" integrity="sha384-oAOxQR6DkCoMliIh8yFnu25d7Eq/PHS21PClpwjOTeU2jRSq11vu66rf90/cZr47" crossorigin="anonymous">

<style>

.header {

padding: 60px;

text-align: center;

background: #ADD8E6;

color: white;

font-size: 30px;

}

.info {

font-style: italic;

font-family: Verdana;

}

.custom-file-input::-webkit-file-upload-button {

visibility: hidden;

}

.custom-file-input::before {

content: 'Select any image';

display: inline-block;

background-color: #ADD8E6;

border: 1px solid #999;

border-radius: 3px;

padding: 5px 8px;

outline: none;

white-space: nowrap;

-webkit-user-select: none;

cursor: pointer;

text-shadow: 1px 1px #fff;

font-weight: 700;

font-size: 10pt;

position: fixed;

left: 550px;

}

.custom-file-input:hover::before {

border-color: black;

}

.custom-file-input:active::before {

background: -webkit-linear-gradient(top, #e3e3e3, #f9f9f9);

}

input[type="submit"] {

border-radius: 3px;

background: #ADD8E6;

border: 1px solid #999;

cursor: default;

font-size: 10pt;

font-weight: 700;

padding: 5px 8px;

}

input[type="submit"]:hover {

box-shadow: 0 1px 1px rgba(0,0,0,0.1);

background: #f8f8f8;

border: 1px solid #c6c6c6;

box-shadow: 0 1px 1px rgba(0,0,0,0.1);

color: #222;

}

</style>

</head>

<div align="center">

<form method=post enctype=multipart/form-data>

<body bgcolor="LightGoldenRodYellow" id="body">

<div class="header">

<h1>CAPTION GENERATOR</h1>

</div>

<br>

<br>

<br>

<h3 class="info">Please upload the image for the analysis</h3>

<br>

<br>

<br>

<table>

<tr>

<td><input type="file" name="file" class="custom-file-input" required></td></div>

<td><input type="submit" value="Upload" class="custom-file-upload"></td>

</tr>

</table>

<br>

<br>

<p id="display"></p>

</body>

</form>

</div>

</html>

**Caption.html**

<html>

<head>

<title>Caption Generator</title>

<link rel="stylesheet" href="https://unpkg.com/purecss@1.0.1/build/pure-min.css" integrity="sha384-oAOxQR6DkCoMliIh8yFnu25d7Eq/PHS21PClpwjOTeU2jRSq11vu66rf90/cZr47" crossorigin="anonymous">

<style>

.caption-img {

display: block;

margin-left: auto;

margin-right: auto;

width: 50%;

}

.caption-txt {

font-style: italic;

font-family: Verdana;

color: red;

font-size: 30px;

}

</style>

</head>

<body>

<img src="{{ url\_for('static', filename= captioned\_image) }}" alt="Captioned Image" class="caption-img">

<p class="caption-txt" align="center">{{ caption }}</p>

</body>

</html>

**Image\_caption\_controller.py**

import os

from image\_caption\_generator import ImageCaptionGenerator

from flask import Flask, render\_template, request, redirect, url\_for, send\_from\_directory, flash

from werkzeug.utils import secure\_filename

UPLOAD\_FOLDER = os.path.join('D:', 'Study', 'Dataset', 'Flickr8k\_Dataset', 'Flicker8k\_Dataset')

ALLOWED\_EXTENSIONS = set(['png', 'jpg', 'jpeg'])

app = Flask(\_\_name\_\_)

app.config['UPLOAD\_FOLDER'] = UPLOAD\_FOLDER

def allowed\_file(filename):

return '.' in filename and \

filename.rsplit('.', 1)[1].lower() in ALLOWED\_EXTENSIONS

@app.route('/', methods=['GET', 'POST'])

def upload\_file():

if request.method == 'POST':

if 'file' not in request.files:

flash('No file part')

return redirect(request.url)

file = request.files['file']

if file.filename == '':

flash('No selected file')

return redirect(request.url)

if file and allowed\_file(file.filename):

filename = secure\_filename(file.filename)

file.save(os.path.join(app.config['UPLOAD\_FOLDER'], filename))

return redirect(url\_for('get\_caption',

filename=filename))

return render\_template("index.html")

@app.route('/caption/<path:filename>', methods=['GET'])

def get\_caption(filename):

imgcptgen = ImageCaptionGenerator()

full\_filename = os.path.join(app.config['UPLOAD\_FOLDER'], filename)

model, tokenizer, max\_length = imgcptgen.testing\_params()

caption = imgcptgen.test(model, tokenizer, max\_length, full\_filename)

return render\_template("caption.html", captioned\_image = 'Flicker8k\_Dataset/' + filename, caption = caption)

if \_\_name\_\_ == '\_\_main\_\_':

app.run()

**7.2 SCREEN SHOTS**

**HOME SCREEN**

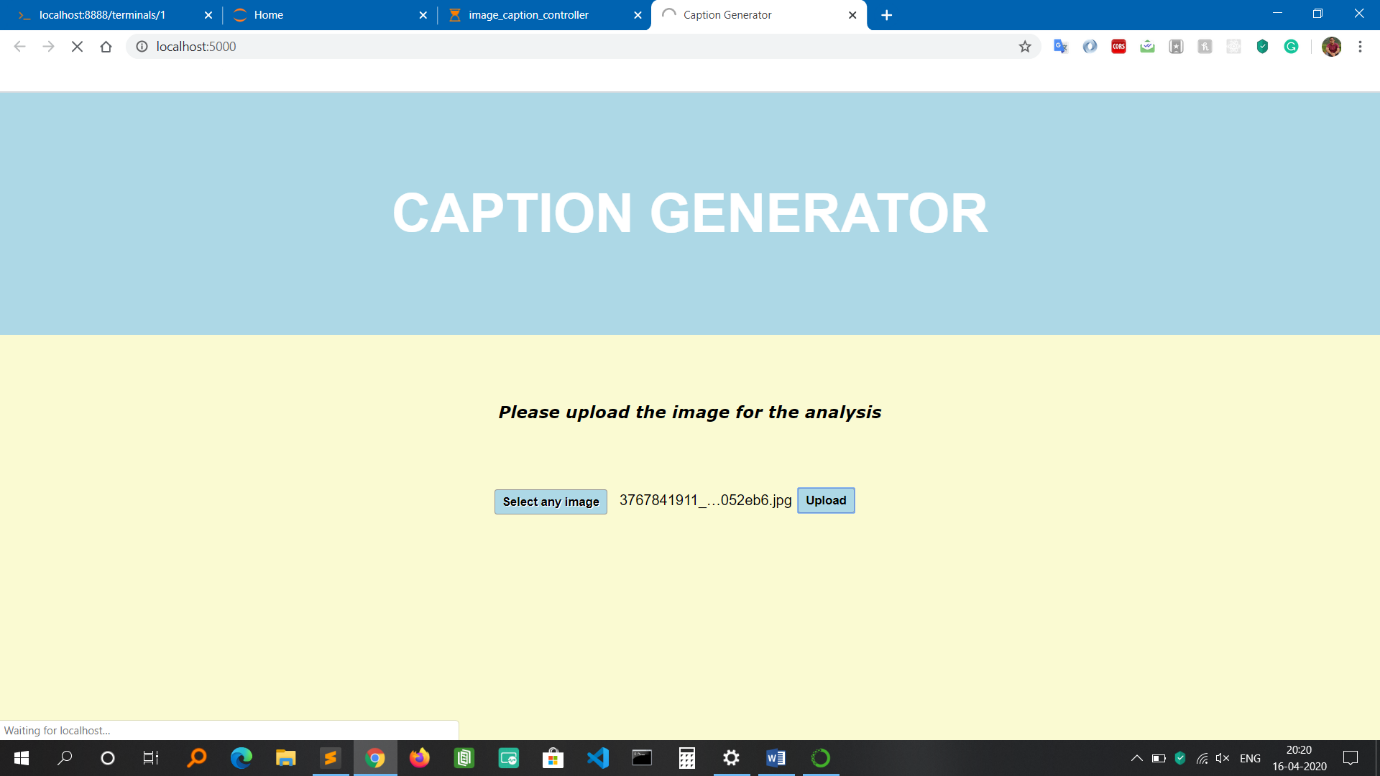


Figure 4.1.1 Home Screen

**RESULT PAGE**

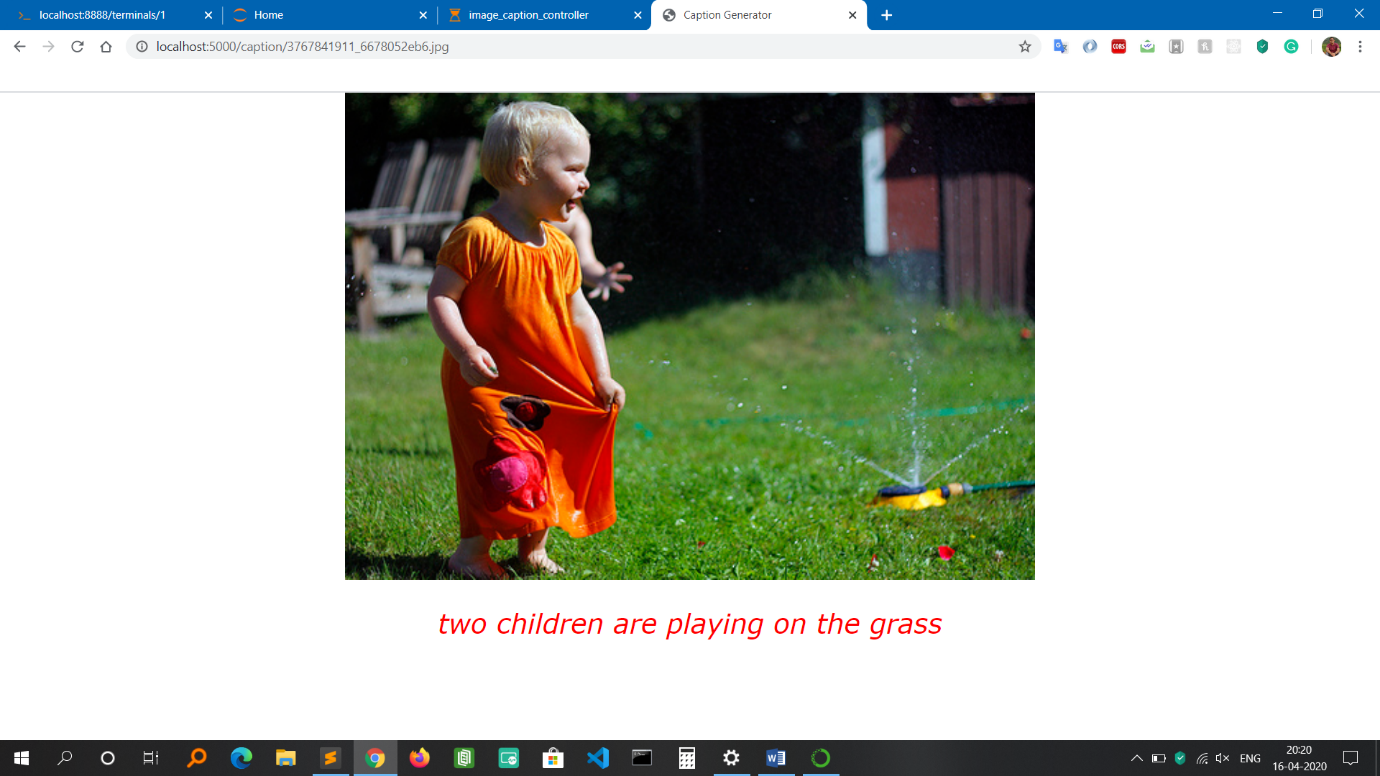


Figure 4.1.2 Result Page Screen

**HOME SCREEN 2**

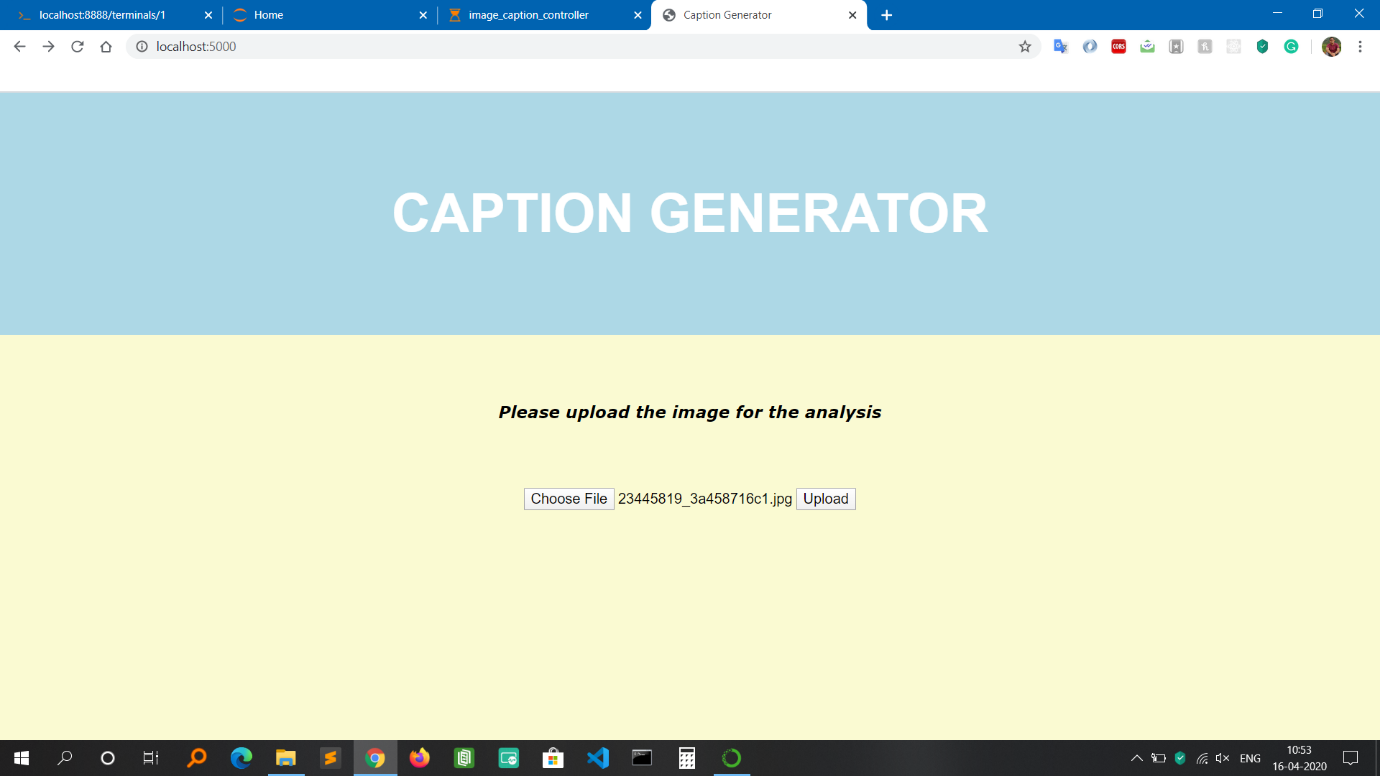
****

Figure 4.1.3 Home Screen

**RESULT PAGE 2**

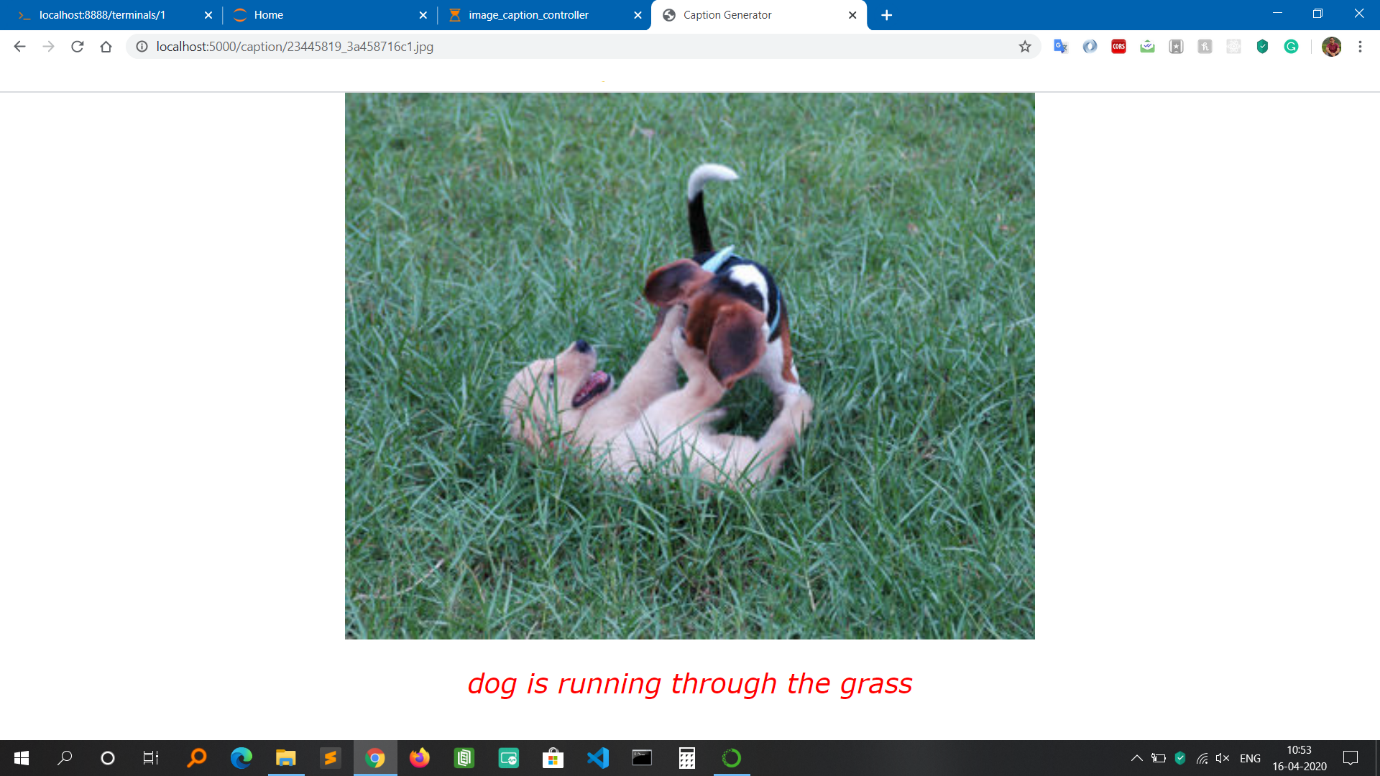
****

Figure 4.1.4 Result Page Screen

1. **SOFTWARE TESTING**

**8.1 Manual Test cases**

Table 8.1.1 Test Case

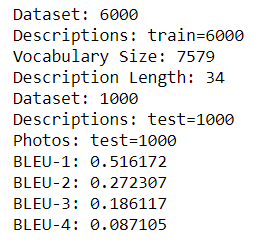
|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | | PES\_001 | **Test Case Description** | | Test the Functionality of uploading the image for analysis | | | | | | |
| **Created By** | |  | **Reviewed By** | |  | | **Version** | | 1 | | |
|  |  |  |  |  |  |  |  |  |  |  |
| **QA Tester’s Log** | |  | | | |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
| **Tester's Name** | |  | **Date Tested** | | 16-April-2020 | | **Test Case (Pass/Fail/Not Executed)** | | Pass | | |
|  |  |  |  |  |  |  |  |  |  |  |
| **S #** | **Prerequisites:** | | |  | **S #** | **Test Data** | | | | | |
| 1 | Access to Chrome Browser | | |  | 1 | Choose file = Upload images of any format like jpg, jpeg, png | | | | | |
| 2 |  | | |  | 2 |  | | | | | |
| 3 |  | | |  | 3 |  | | | | | |
| 4 |  | | |  | 4 |  | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |
| **Test Scenario** | Verify on ability to upload the images of any format for analysis after selecting the desired image | | | | | |  |  |  |  | |
|  |  |  |  |  |  |  |  |  |  |  |
| **Step #** | **Step Details** | | **Expected Results** | | **Actual Results** | | | **Pass / Fail / Not executed / Suspended** | | | |
|
| 1 | Navigate to <http://localhost:5000/> | | Caption generator homepage should open | | Successfully redirected | | | Pass | | | |
| 2 | Click on Choose File to select the image for uploading | | File Explorer should open for selecting the image | | File Explorer Window successfully opened | | | Pass | | | |
| 3 | Select the desired image | | Should display the name of the image selected in the home page of application | | Image name is being displayed on the home page of application | | | Pass | | | |
| 4 | Click on Upload button to upload the image for analysis | | ML processing should begin in the background and should redirect to the result page for displaying the caption with the respective image | | Page redirected and caption generated for the uploaded image. | | | Pass | | | |

Table 8.1.2 Test Cases

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | | PES\_002 | **Test Case Description** | | Test the Functionality of uploading the image for analysis without selecting any image | | | | | | |
| **Created By** | |  | **Reviewed By** | |  | | **Version** | | 1 | | |
|  |  |  |  |  |  |  |  |  |  |  |
| **QA Tester’s Log** | |  | | | |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
| **Tester's Name** | |  | **Date Tested** | | 16-April-2020 | | **Test Case (Pass/Fail/Not Executed)** | | Fail | | |
|  |  |  |  |  |  |  |  |  |  |  |
| **S #** | **Prerequisites:** | | |  | **S #** | **Test Data** | | | | | |
| 1 | Access to Chrome Browser | | |  | 1 | Click the upload button | | | | | |
| 2 |  | | |  | 2 |  | | | | | |
| 3 |  | | |  | 3 |  | | | | | |
| 4 |  | | |  | 4 |  | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |
| **Test Scenario** | Verify on whether the home page gets redirected to result page without selecting any image | | | | | |  |  |  |  | |
|  |  |  |  |  |  |  |  |  |  |  |
| **Step #** | **Step Details** | | **Expected Results** | | **Actual Results** | | | **Pass / Fail / Not executed / Suspended** | | | |
|
| 1 | Navigate to <http://localhost:5000/> | | Caption generator homepage should open | | Successfully redirected | | | Pass | | | |
| 2 | Click on Upload button to upload the image for analysis | | Validator should display message saying ‘Please select a file’ as no image is selected for uploading | | Page not redirected and Validator displays message saying ‘Please select a file’ | | | Fail | | | |

**8.2 BLUE Score**

**Figure 8.2.1 BLUE Score for Model Evaluation**

****

1. **CONCLUSION**
2. The objective of the project was to identify the action portrayed in the given image. The generated caption will describe the image that will say what kind of actions is taking place in it.
3. This has been solved with my application as it provides a service to users that helps in generating captions for any images of their choice once they upload it in my application.
4. Besides, the whole application has been deployed on Heroku platform. So in the future, if the user requests for any changes, it can be easily done through git version control.
5. **FUTURE ENHANCEMENT**
6. I can build a customized mobile app that will make the user more convenient to use.
7. In the future, with the advancement of algorithms, I should be able to achieve a higher accuracy that will help in better generation of captions for the images.

**APPENDIX A**

**BIBLIOGRAPHY**

1. Jason Brownlee (2007), Deep Learning for Natural Language, Edition v1.1
2. Raffaella Bernardi, Ruket Cakici, Desmond Elliott, Aykut Erdem, Erkut Erdem, Nazli Ikizler-Cinbis, Frank Keller, Adrian Muscat, Barbara Plank - Automatic Description Generation from Images: A Survey of Models, Datasets, and Evaluation Measures, Cornell University
3. Jiuxiang Gu, Gang Wang, Jianfei Cai, Tsuhan Chen - An Empirical Study of Language CNN for Image Captioning, 2017 IEEE International Conference on Computer Vision
4. MD. Zakir Hossain, Ferdous Sohel, Mohd Fairuz Shiratuddin, Hamid Laga - A Comprehensive survey of Deep Learning for Image Captioning - Murdoch University, Australia
5. Marc Tanti, Albert Gatt, Kenneth P. Camilleri - What is the Role of Recurrent Neural Networks (RNNs) in an Image Caption Generator? - Cornell University
6. <https://www.kaggle.com/>

1. <https://www.medium.com/>
2. <https://www.towardsdatascience.com/>
3. https://www.machinelearningmastery.com
4. https://www.sciencedirect.com
5. https://www.researchgate.net