Design and Development of

The Caption Creator System

A

MINOR PROJECT-II REPORT

Submitted in partial fulfillment of the requirements

for the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

By

GROUP NO.: 55

Aaditya Raj	**0187CS171001**
Aman Sahay	**0187CS171022**

Under the guidance of

Dr. Rajeev Kumar Gupta

Apr-2020

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
Sagar Institute of Science & Technology (SISTec)
Bhopal (M.P.)
An ISO 9001:2008 Certified Institution
Approved by AICTE, New Delhi & Govt. of M.P.
Affiliated to Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal (M.P.)

Sagar Institute of Science & Technology (SISTec), Bhopal
Department of COMPUTER SCIENCE & ENGINEERING
Bhopal (M.P.)

Apr-2020

CERTIFICATE

I hereby certify that the work which is being presented in the B.Tech. Minor Project Report entitled **The Caption Creator System,** in partial fulfillment of the requirements for the award of the degree of ***Bachelor of Technology*** in ***Computer Science & Engineering*** and submitted to the Department of Computer Science & Engineering, *Sagar Institute of Science & Technology (SISTec)***,** Bhopal (M.P.) is an authentic record of our own work carried out during the period from Jan-2020 to Apr-2020 under the supervision of **Dr. Rajeev Kumar Gupta.**
The content presented in this project has not been submitted by us for the award of any other degree elsewhere.

Name	***Enrollment No***	***Signature***
Aaditya Raj	***0187CS171001***	
Aman Sahay	***0187CS171022***	

This is to certify that the above statement made by the candidate is correct to the best of our knowledge.
Date:

Dr. Rajeev Kumar Gupta	**Prof. Ujjwal Nigam**	**Dr. Keshavendra Chaudhary**
Project Guide	***HOD***	***Principal***

ABSTRACT

Every day, we encounter a large number of images from various sources such as the internet news articles, document diagrams and advertisements. These sources contain images that viewers would have to interpret themselves. Most images do not have a description, but the human can largely understand them without their detailed captions. However, machine needs to interpret some form of image captions if humans need automatic image captions from it.

Image captioning is important for many reasons. For example, they can be used for automatic image indexing. Image indexing is important for Content-Based Image Retrieval (CBIR) and therefore, it can be applied to many areas, including biomedicine, commerce, the military, education, digital libraries, and web searching. Social media platforms such as Facebook and Twitter can directly generate descriptions from images. The descriptions can include where we are (e.g., beach, cafe), what we wear and importantly what we are doing there.

Image captioning is a popular research area of Artificial Intelligence (AI) that deals with image understanding and a language description for that image. Image understanding needs to detect and recognize objects. It also needs to understand scene type or location, object properties and their interactions. Generating well-formed sentences requires both syntactic and semantic understanding of the language

The aim of this project is to provide help in auto-captioning the images which would help in understand the image. It would tell us about the objects in the image or location and many more properties.

ACKNOWLEDGEMENT

It gives us immense pleasure to express our deepest sense of gratitude and sincere thanks to our highly respected and esteemed guide **Dr. Rajeev Kumar Gupta**, Department of Computer Science and Engineering, SISTec Gandhi Nagar Bhopal, for his valuable guidance, encouragement and help for completing this work. Their useful suggestions for this whole work and co-operative behavior are sincerely acknowledged.

We would like to express our sincere thanks to **Dr. Keshavendra Choudhary**, Principal, Vice-Principal **Dr. Swati Saxena** , SISTec, Gandhi Nagar Bhopal for giving us an opportunity to undertake this project.

We also wish to express our gratitude to **Prof. Ujjwal Nigam,** Head, Department of Computer Science and Engineering, for his kind hearted support.

At the end, we would like to express our sincere thanks to all our friends and others who helped us directly or indirectly during this project work.

Name	***Enrollment Number***	***Signature***
Aaditya Raj	***0187CS171001***	
Aman Sahay	***0187CS171022***	

Table Of Contents **PAGE NO.**

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ACRONYM	FULL FORM
SDLC	Software Development Life Cycle
CSS	Cascading Style Sheets
HTML	Hyper Text Markup Language
UML	Unified Modeling Language
js	JavaScript
BLEU	Bilingual Evaluation Understudy

Chapter 1
Introduction

CHAPTER-1
INTRODUCTION

1.1 ABOUT PROJECT

Image Captioning is the process of generating textual description of an image. It uses both Natural Language Processing and Computer Vision to generate the captions. Early image description generation methods aggregate image information using static object class libraries in the image and modeled using statistical language models. The image description is obtained by predicting the most likely nouns, verbs, scenes, and prepositions that make up the sentence. The Caption Creator System propose using a deep learning to detect objects in an image, classifying each image region and processing it by a prepositional relationship function and finally applying a conditional random field prediction image tag to generate a natural language description using LSTM.

This project is done using Python, Deep Learning, NLP and HTML, CSS, BootStrap as front-end and FLASK-Framework for the connectivity. The project Caption Creator System helps to analyses image and provide useful information or caption by analyzing image.

1.2 PROJECT OBJECTIVE

The proposed system would be overcoming the human intervention and increase automation of visual or computer vision helps automatic generation of caption ,what image says or it looks like ,it also provide useful information of an image like activity tracking which may be very useful in field of automation like auto-driving vehicles, automatic attendance, image tracking etc. The system would be extracting the useful keywords from the Image and combine them in an appropriate caption for that image.

The objective of our proposed system are:

1. Useful Information, activity and expression extraction from Image.
2. Need to upload an Image
3. Provide an appropriate caption for that image.

Chapter 2
Literature Survey

CHAPTER-2
LITERATURE SURVEY

2.1 EXISTING SYSTEM

Now a days there are various Image Detection, Image Tagging and Image Captioning software available , made by big corporates & MNCs like Microsoft(Captionbot.ai), IBM etc. but still they are not so much accurate and facing difficulties though they are trained on very high level GPUs.

2.1.1 Disadvantages of Current System

- Require much training for better results.
- Need high level Processors ,GPUs .
- Grammatical mistakes and lack of good Translation.
- Facing Problem with accuracy as it doesn't up to the mark or 100% accurate.

2.2 PROPOSED SYSTEM

"Design and Development of Caption Creator System", works on the problem of previous system by proving the facility to overcome the Grammatical mistakes & lack of good Translation happened with previous systems. Moreover, it trained on VGG16 model, so it takes much less time in compared with Google net or Inception network. Also, it is available on internet for all so, everyone can use it.

2.2.1 Characteristics of proposed system

- Easily Understandable and attractive interface .
- Needs less time to train.
- Result in very natural way or natural language .

Chapter 3
Problem Identification

CHAPTER-3
PROBLEM IDENTIFICATION

Problem Identification means the issues in working domain that are required to solve using automation.

3.1.1 Large Human Involvement or Manual Work.

Working with image manually, and maintaining its record were large challenge . Work like Image verification , classification or obtaining useful information from image was also a very big challenge which require more human involvement.

3.1.2 Extracting complicated information from input images.

It is very difficult to extract complicated information from image manually . Automation helps in finding pattern to analyse complex information . It is very time consuming to take record of all the image for operations.

3.1.3 Training & Testing

It was very tough to train our model & test has large values of data and extracting each data is very time consuming as data is image or (.jpg) file ,also we have bith jpg and text data and mapping is very difficult to make relation between them ,where text interprets as labels for given image.

3.1.4 Variation & Dissimilarities of Data

In dataset(i.e-Flicker8k) image file size is not same , thus first it is challenge to make all files in similar manner and then preprocess data. Similarly , during caption creation for each image it is very tough to combine appropriate and make sentence in very natural way.

Chapter 4
Proposed Methodology

CHAPTER-4
PROPOSED METHODOLOGY

4.1 Methods Used

4.1.1 Data collection

There are many open source datasets available for this problem, like Flickr 8k (containing8k images), Flickr 30k (containing 30k images), MS COCO (containing 180k images), etc.

But for the purpose of this case study, I have used the Flickr 8k dataset which you can download by filling form provided by the University of Illinois at Urbana-Champaign. Also training a model with large number of images may not be feasible on a system which is not a very high end PC/Laptop.

This dataset contains 8000 images each with 5 captions (as we have already seen in the Introduction section that an image can have multiple captions, all being relevant simultaneously).

These images are bifurcated as follows:

- Training Set — 6000 images
- Dev Set — 1000 images
- Test Set — 1000 images

4.1.2 Understanding the data

One of the files is “Flickr8k.token.txt” which contains the name of each image along with its 5 captions. We can read this file as follows:

```
filename = "/dataset/TextFiles/Flickr8k.token.txt"
file = open(filename, 'r')
doc = file.read()
```

The text file looks as follows:

```
101654506_8eb26cfb60.jpg#1   A dog is running in the snow
101654506_8eb26cfb60.jpg#2   A dog running through snow .
101654506_8eb26cfb60.jpg#3   a white and brown dog is running through a snow covered field .
101654506_8eb26cfb60.jpg#4   The white and brown dog is running over the surface of the snow
.

1000268201_693b08cb0e.jpg#0  A child in a pink dress is climbing up a set of stairs in an
entry way .
1000268201_693b08cb0e.jpg#1  A girl going into a wooden building .
1000268201_693b08cb0e.jpg#2  A little girl climbing into a wooden playhouse .
1000268201_693b08cb0e.jpg#3  A little girl climbing the stairs to her playhouse .
1000268201_693b08cb0e.jpg#4  A little girl in a pink dress going into a wooden cabin .
```

Figure 4.1 Text Dataset

4.1.3 Data Cleaning

When we deal with text, we generally perform some basic cleaning like lower-casing all the words (otherwise“hello” and “Hello” will be regarded as two separate words), removing special tokens (like ‘%’, ‘$’, ‘#’, etc.), eliminating words which contain numbers (like ‘hey199’, etc.).

```
def clean_descriptions(descriptions):
  # prepare translation table for removing punctuation
  table = str.maketrans('', '', string.punctuation)
  for key, desc_list in descriptions.items():
    for i in range(len(desc_list)):
      desc = desc_list[i]
      # tokenize
      desc = desc.split()
      # convert to lower case
      desc = [word.lower() for word in desc]
      # remove punctuation from each token
      desc = [w.translate(table) for w in desc]
      # remove hanging 's' and 'a'
      desc = [word for word in desc if len(word)>1]
      # remove tokens with numbers in them
      desc = [word for word in desc if word.isalpha()]
      # store as string
      desc_list[i] =  ' '.join(desc)
```

4.1.4 Loading the training set

The text file “Flickr_8k.trainImages.txt” contains the names of the images that belong to the training set. So we load these names into a list “train”.

```
filename = 'dataset/TextFiles/Flickr_8k.trainImages.txt'
doc = load_doc(filename)
train = list()
for line in doc.split('\n'):
    identifier = line.split('.')[0]
    train.append(identifier)
print('Dataset: %d' % len(train))
Dataset: 6000
```

```
from pickle import dump
# load training dataset (6K)
filename = 'Flickr8k_text/Flickr_8k.trainImages.txt'
train = load_set(filename)
print('Dataset: %d' % len(train))
# descriptions
train_descriptions = load_clean_descriptions('descriptions.txt', train)
print('Descriptions: train=%d' % len(train_descriptions))
# photo features
train_features = load_photo_features('features.pkl', train)
print('Photos: train=%d' % len(train_features))
# prepare tokenizer
tokenizer = create_tokenizer(train_descriptions)
dump(tokenizer, open('tokenizer.pkl', 'wb'))
vocab_size = len(tokenizer.word_index) + 1
print('Vocabulary Size: %d' % vocab_size)
# determine the maximum sequence length
max_length = max_length(train_descriptions)
print('Description Length: %d' % max_length)
# prepare sequences
X1train, X2train, ytrain = create_sequences(tokenizer, max_length, train_d
escriptions, train_features)

# dev dataset

# load test set
filename = 'Flickr8k_text/Flickr_8k.devImages.txt'
test = load_set(filename)
print('Dataset: %d' % len(test))
# descriptions
test_descriptions = load_clean_descriptions('descriptions.txt', test)
print('Descriptions: test=%d' % len(test_descriptions))
```

```
# photo features
test_features = load_photo_features('features.pkl', test)
print('Photos: test=%d' % len(test_features))
# prepare sequences
X1test, X2test, ytest = create_sequences(tokenizer, max_length, test_descr
iptions, test_features)
```

4.1.5 Data Preprocessing — Images

Images are nothing but input (X) to our model. As you may already know that any input to a model must be given in the form of a vector.

We need to convert every image into a fixed sized vector which can then be fed as input to the neural network. For this purpose, we opt for VGG16 .

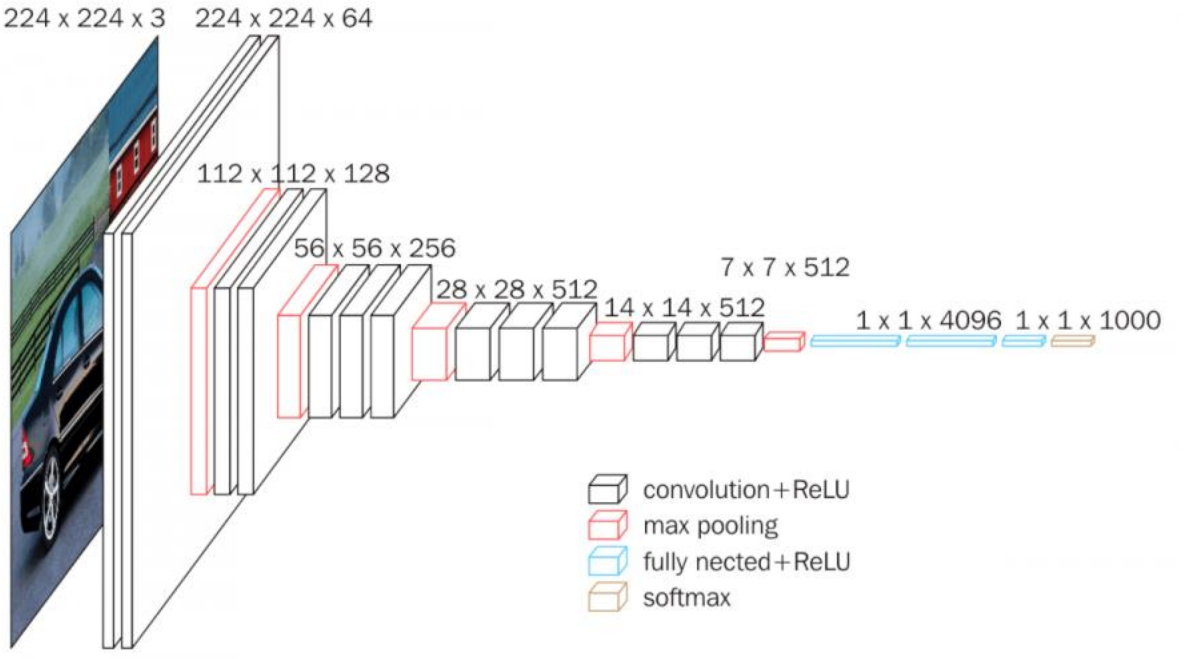

Figure 4.2 VGG16 Model

4.1.6 Data Preprocessing — Captions

We must note that captions are something that we want to predict. So during the training period, captions will be the target variables (Y) that the model is learning to predict.

```
# extract descriptions for images
def load_descriptions(doc):
  mapping = dict()
  # process lines
  for line in doc.split('\n'):
    # split line by white space
    tokens = line.split()
    if len(line) < 2:
```

```
      continue
    # take the first token as the image id, the rest as the description
    image_id, image_desc = tokens[0], tokens[1:]
    # remove filename from image id
    image_id = image_id.split('.')[0]
    # convert description tokens back to string
    image_desc = ' '.join(image_desc)
    # create the list if needed
    if image_id not in mapping:
      mapping[image_id] = list()
    # store description
    mapping[image_id].append(image_desc)
  return mapping

def clean_descriptions(descriptions):
  # prepare translation table for removing punctuation
  table = str.maketrans('', '', string.punctuation)
  for key, desc_list in descriptions.items():
    for i in range(len(desc_list)):
      desc = desc_list[i]
      # tokenize
      desc = desc.split()
      # convert to lower case
      desc = [word.lower() for word in desc]
      # remove punctuation from each token
      desc = [w.translate(table) for w in desc]
      # remove hanging 's' and 'a'
      desc = [word for word in desc if len(word)>1]
      # remove tokens with numbers in them
      desc = [word for word in desc if word.isalpha()]
      # store as string
      desc_list[i] =  ' '.join(desc)

# convert the loaded descriptions into a vocabulary of words
def to_vocabulary(descriptions):
  # build a list of all description strings
  all_desc = set()
  for key in descriptions.keys():
    [all_desc.update(d.split()) for d in descriptions[key]]
  return all_desc
# save descriptions to file, one per line
def save_descriptions(descriptions, filename):
  lines = list()
  for key, desc_list in descriptions.items():
    for desc in desc_list:
      lines.append(key + ' ' + desc)
  data = '\n'.join(lines)
  file = open(filename, 'w')
```

```
  file.write(data)
  file.close()
filename = 'Flickr8k_text/Flickr8k.token.txt'
# load descriptions
doc = load_doc(filename)
# parse descriptions
descriptions = load_descriptions(doc)
print('Loaded: %d ' % len(descriptions))
# clean descriptions
clean_descriptions(descriptions)
# summarize vocabulary
vocabulary = to_vocabulary(descriptions)
print('Vocabulary Size: %d' % len(vocabulary))
# save to file
save_descriptions(descriptions, 'descriptions.txt')
```

4.1.7 Data Preparation using Generator Function

This is one of the most important steps in this case study. Here we will understand how to prepare the data in a manner which will be convenient to be given as input to the deep learning model.

Hereafter, I will try to explain the remaining steps by taking a sample example as follows:

Consider we have 3 images and their 3 corresponding captions as follows:

- `group of people are standing in front of building`

Figure 4.3 Data Example 1

- `brown dog is running through the grass`

Figure 4.4 Data Example 2

- `boy in blue shirt is jumping on trampoline`

Figure 4.5 Data Example 3

4.1.8 Word Embeddings

As already stated above, we will map the every word (index) to a 200-long vector and for this purpose, we will use a pre-trained LSTM Model:

4.3ModelArchitecture

- VGG16 Model:

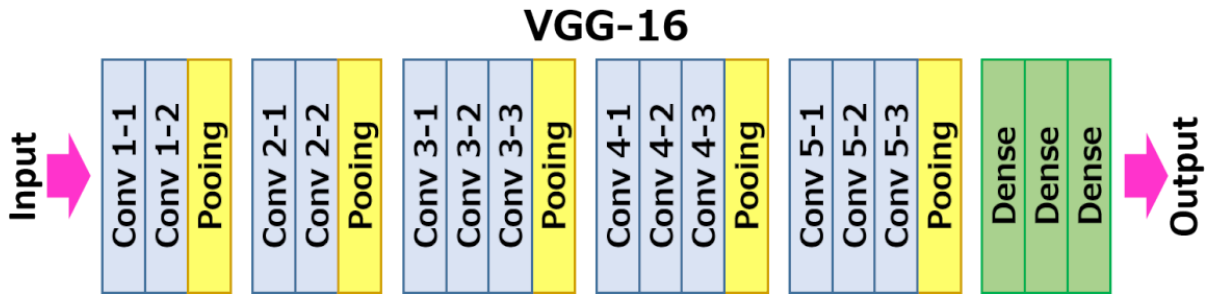

Figure 4.6 VGG-16

- LSTM Model

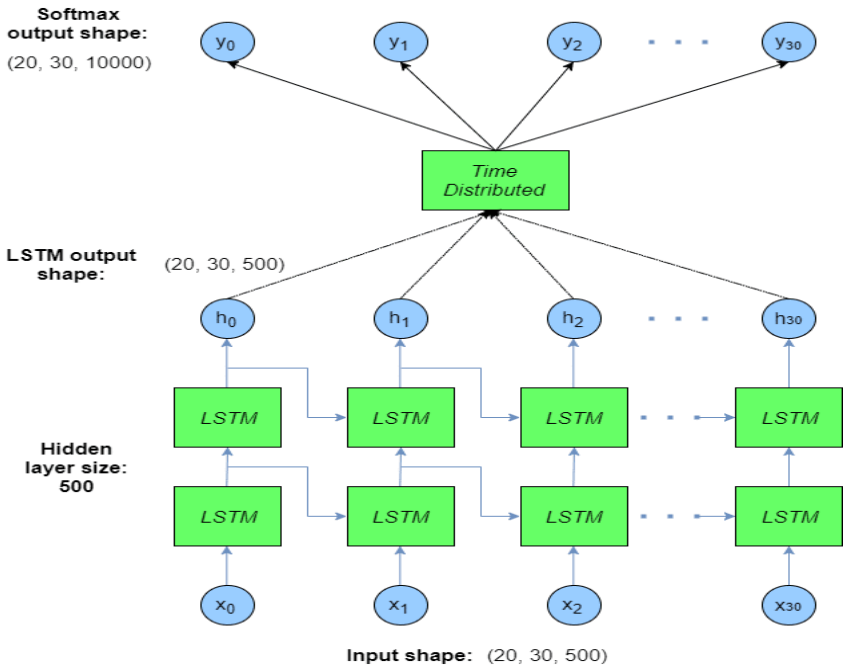

Figure 4.7 LSTM Model

Chapter 5
Software and Hardware Requirements

CHAPTER-5
SOFTWARE AND HARDWARE REQUIREMENTS

5.1 INTRODUCTION

To be used efficiently, all computer needs certain hardware component or other software resources to be present on a computer these perquisites are known as system requirements. Most software defines two sets of system requirements: minimum and recommended with increasing demand for higher processing power and resources in new version of software of system requirements to increase our time.

5.2 SOFTWARE REQUIREMENTS

Software is part of computer system that consist of data and computer instructions, in contrast to the physical hardware from which the system is built. In computer science and software engineering, computer software is all information processed by computer system, Program and data. Computer software include computer program library dependencies and related non-executable data such as online documentation or digital media in this project. We have used following software:

Table 5.1 Used Tools and Software

Technology	Machine Learning, Flask,Deep Learning
Tools	Jupyter Notebook, Visual Studio, | Sublime Text Editor, Google Colab
UI Designing	HTML, CSS, BootStrap
Programming Language	JavaScript, Python

5.3 HARDWARE REQUIREMENTS

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware or we can say hardware is the collection of all parts, which we can touch. The most common set of requirements defined by any operating system of software application is the physical computer resources also known as Hardware. Hardware requirements list often accompanied by a Hardware Compatibility List (HCL), especially case of operating systems. Hardware that is necessary for this project is:

Table 5.2: Hardware Requirements

Processor	Minimum Intel i5
Ram	25GB
Hard disk	4GB
GPU	Minimum 4GB

Chapter 6
Result Evaluation

CHAPTER-6
RESULT EVALUATION

6.1 Evaluation of Model

The model has been evaluated against a given dataset of photo descriptions and photo features. The actual and predicted descriptions are collected and evaluated collectively using the corpus BLEU score that summarizes how close the generated text is to the expected text.

6.1.1 BLEU

BLEU uses a modified form of precision to compare a candidate translation against multiple reference translations. The metric modifies simple precision since machine translation systems have been known to generate more words than are in a reference text.

The model has been trained for 20 epoches on 6000 training samples of Flickr8k Dataset. It acheives a BLEU-1 = ~0.69 with 1000 testing samples.

The NLTK Python library implements the BLEU score calculation in the **corpus_bleu()** function. A higher score close to 1.0 is better, a score closer to zero is worse.

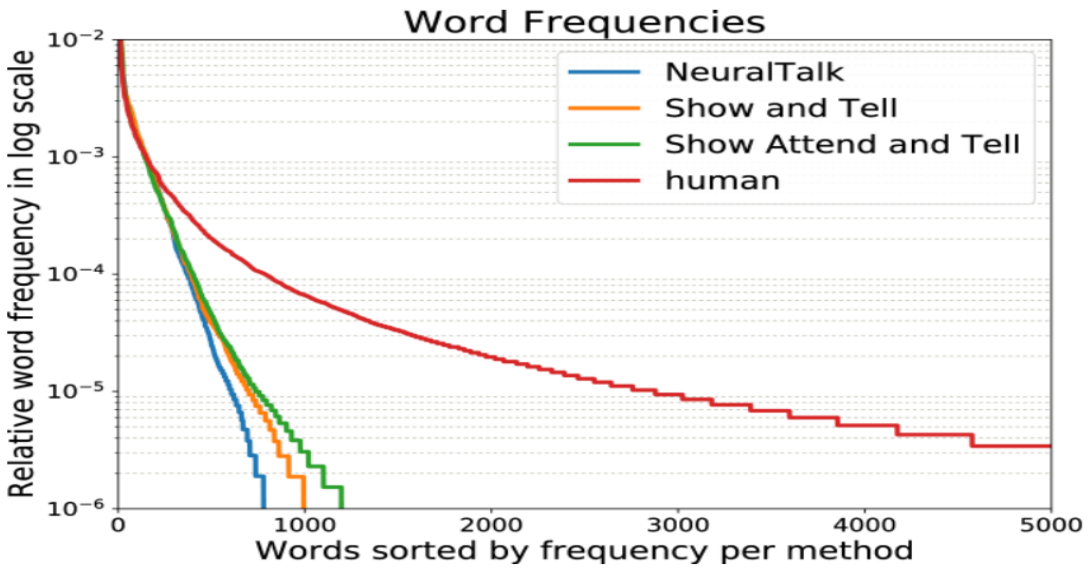

Figure 6.1 Word Frequencies

6.2 Output Screens

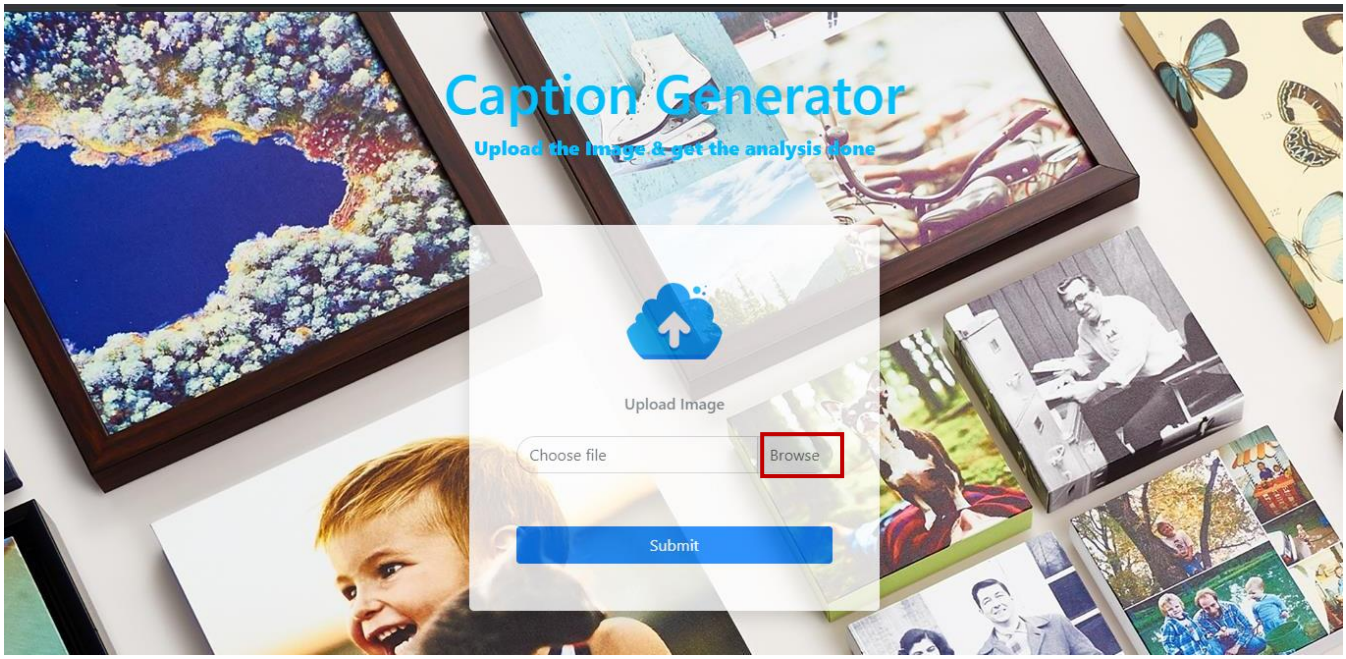

Figure 6.2 HOMEPAGE

- This is the homepage of the application.
- Now, Click on Browse for selecting the image file for which the caption has to be generated.

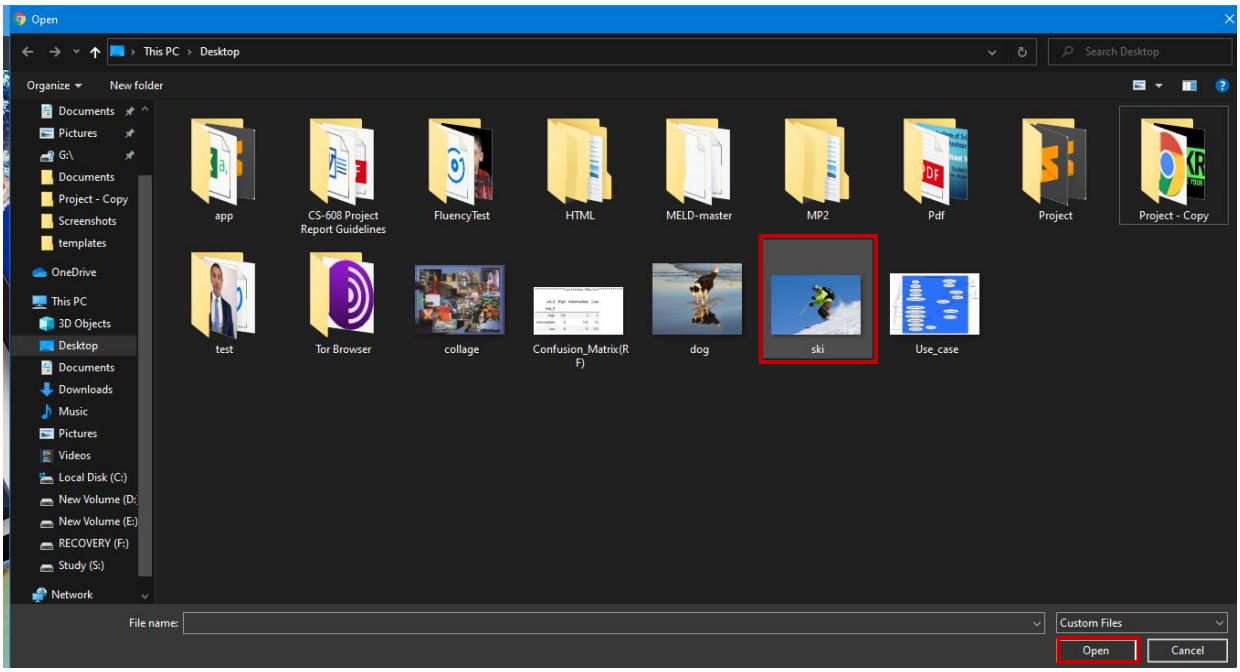

Figure 6.3 Uploading Image

- Select the image whose Caption you require.
- Click Open.

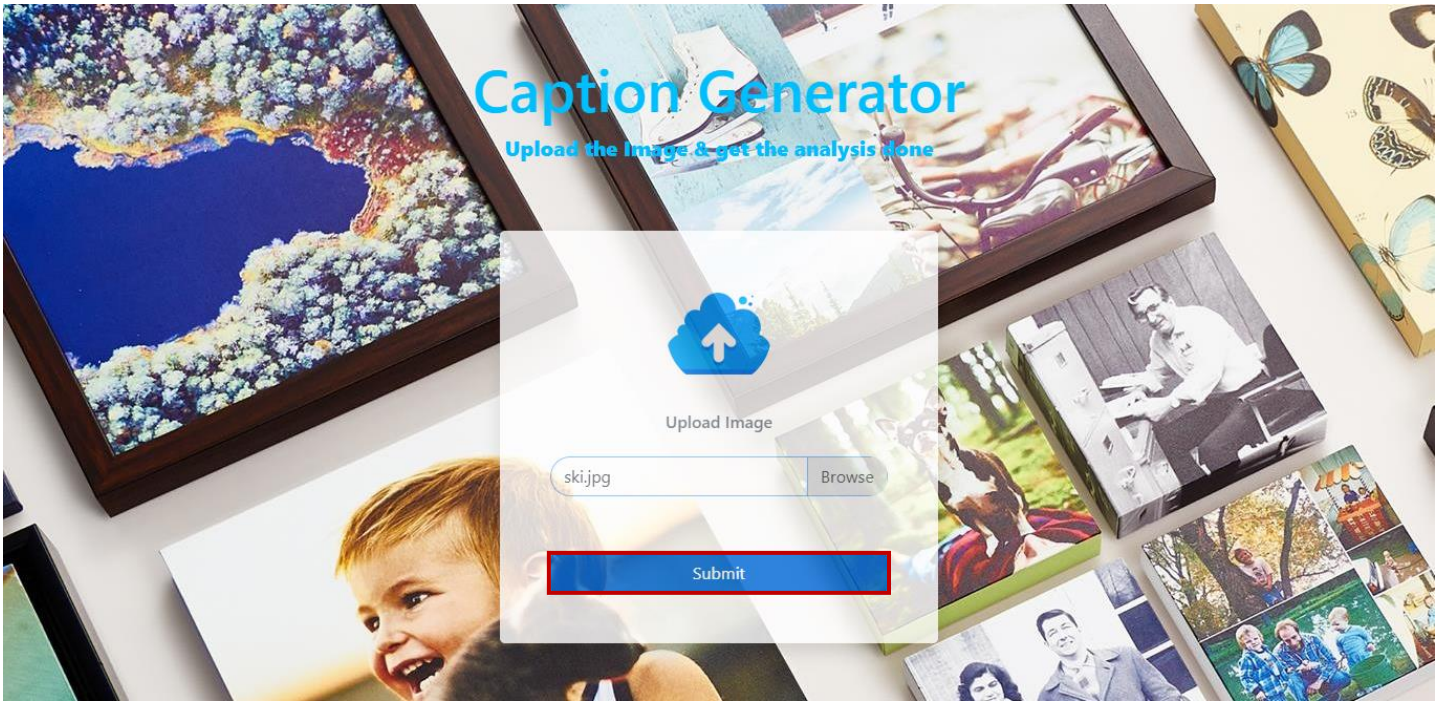

Figure 6.4 Submitting the Image

- After Selecting the image file, click on “Submit” Button.

Figure 6.5 The Result Page (Predicting the Caption for the Image)

- This the required result
- To check other images, click on “Check Other” Button

Chapter 7
Deployment

CHAPTER-7
DEPLOYMENT

7.1 INTRODUCTION

Software deployment brings many key advantages to enterprises. Tasks like installing, uninstalling and updating software applications on each computer are time consuming. Software deployment services reduce the time and make the process error free. The software can be easily controlled and managed through deployment. You can also monitor software information and the actions of users.

The web application has been deployed using Heroku cloud.

7.2 DEPLOYMENT OF WEBSITE

7.2.1 INTRODUCTION

In deployment we will cover installation and configuration of software (i.e., Heroku CLI) that we have used in this project. Here, we used Git for the deployment of Web-app on Heroku.

7.2.2 DOWNLODING GIT

Download git from https://git-scm.com/download/win

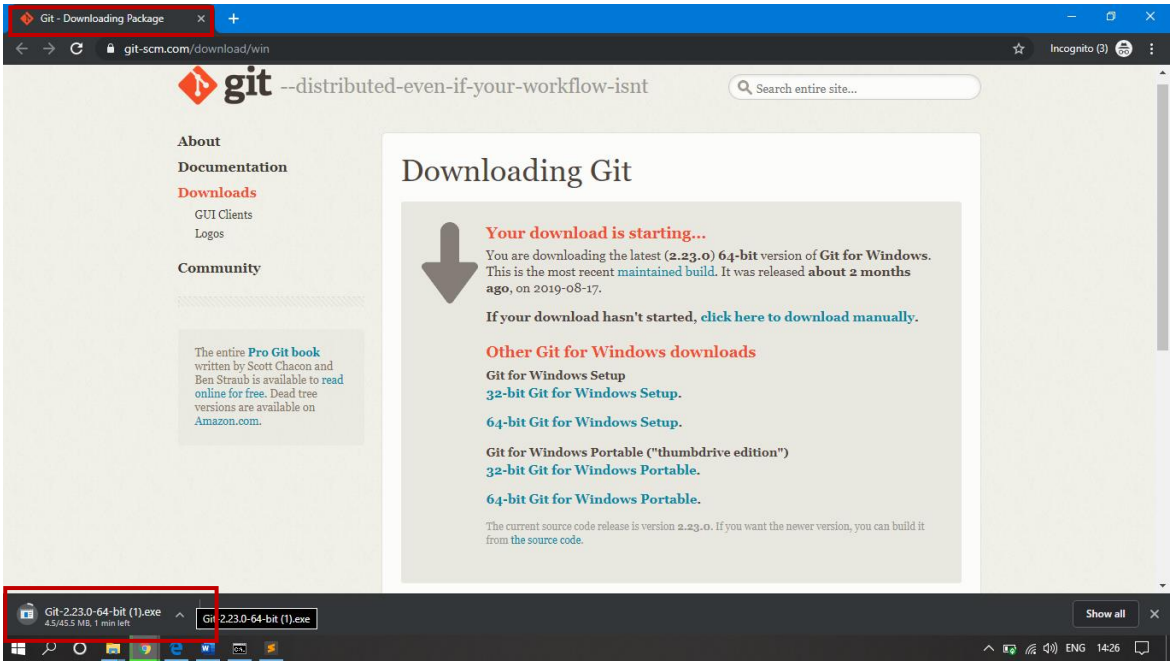

Figure 7.1 Downloading of Git

7.2.3 INSTALLING GIT

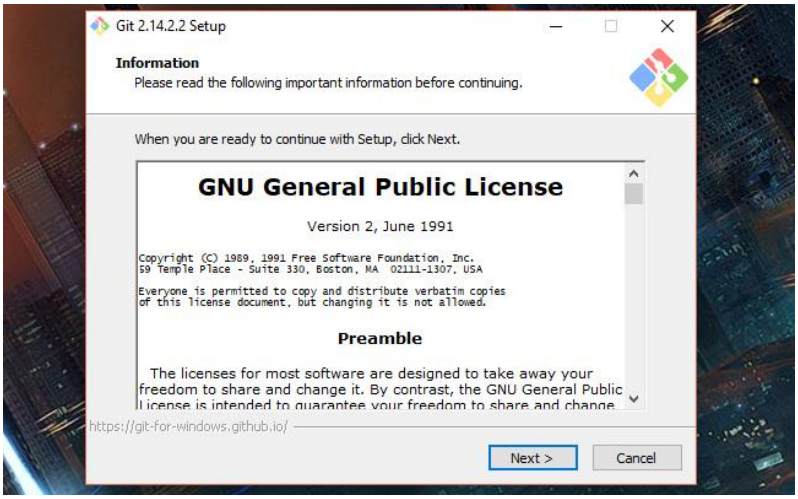

Figure 7.2 Installation of Git (Step 1)

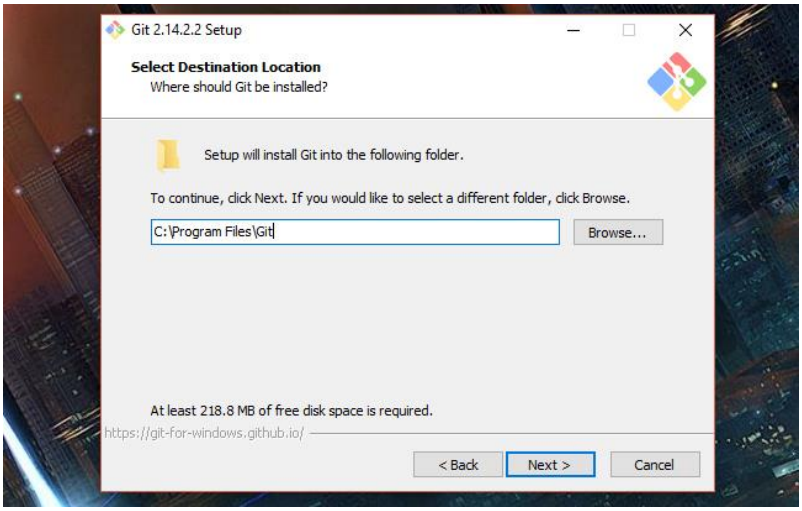

Figure 7.3 Installation of Git (Step 2)

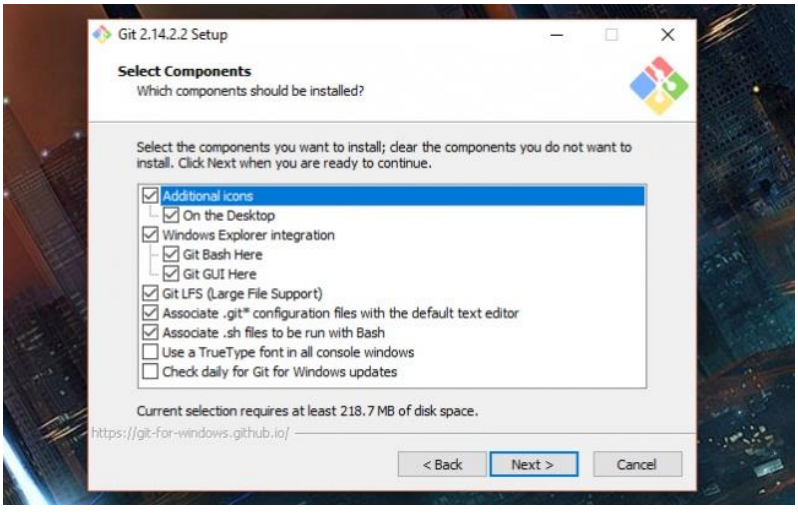

Figure 7.4 Installation of Git (Step 3)

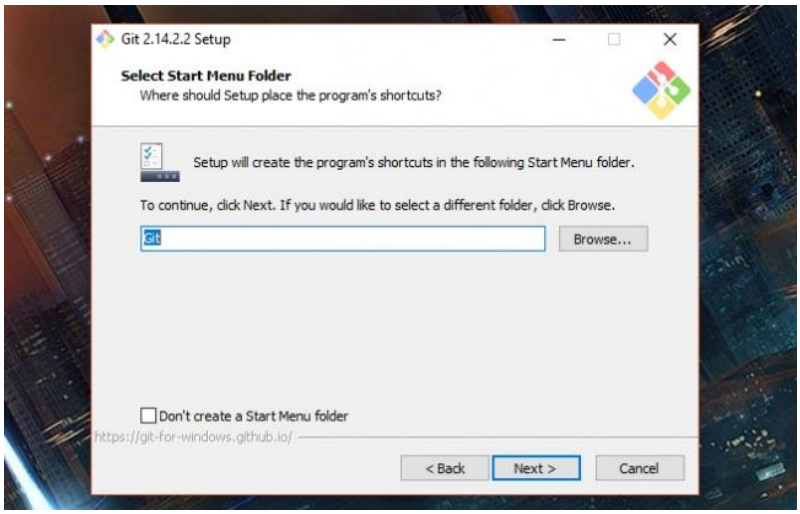

Figure 7.5 Installation of Git (Step 4)

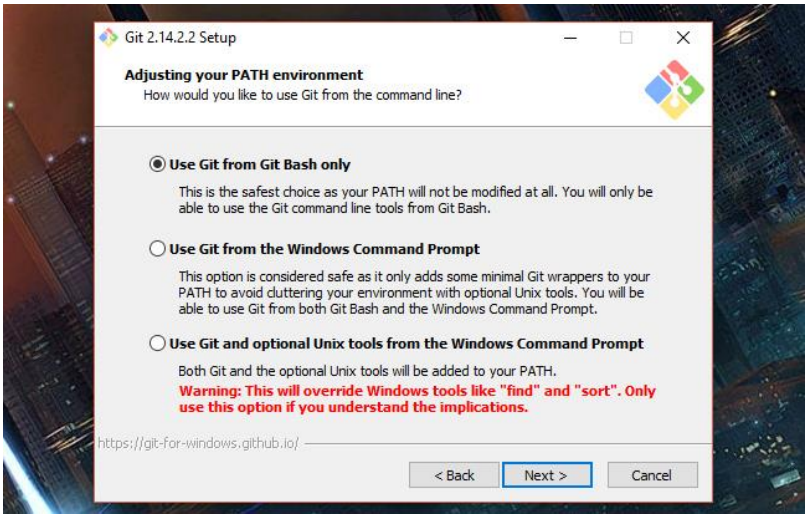

Figure 7.6 Installation of Git (Step 5)

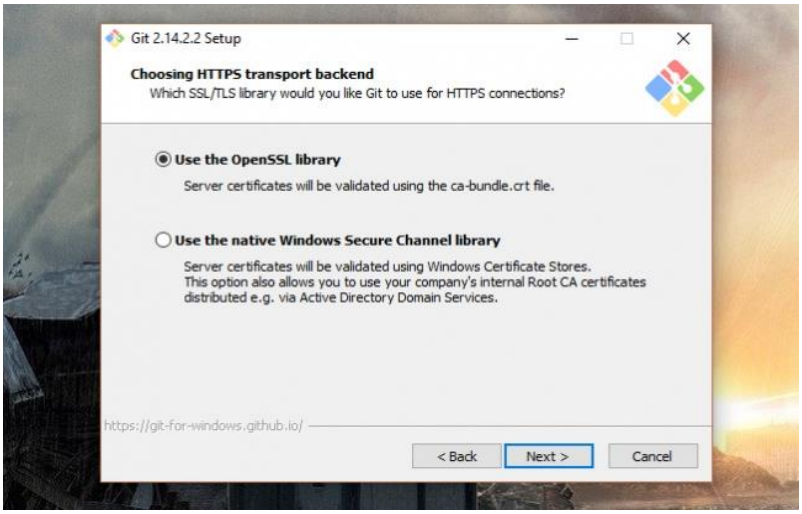

Figure 7.7 Installation of Git (Step 6)

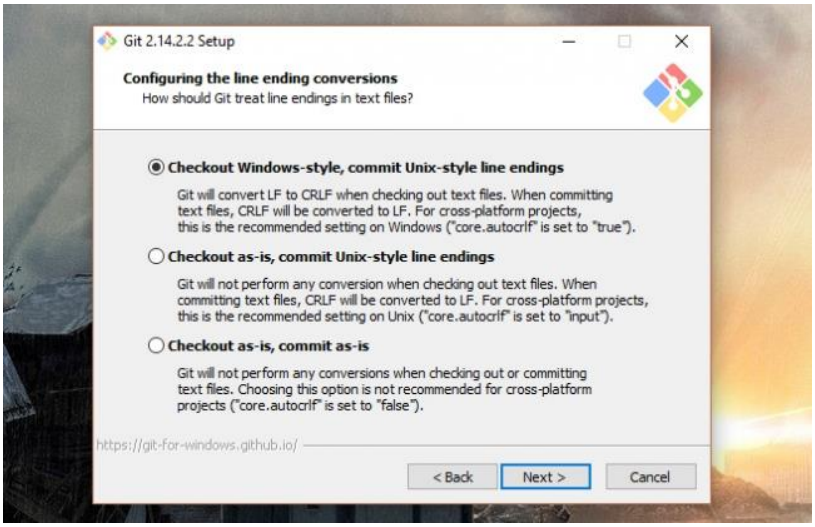

Figure 7.8 Installation of Git (Step 7)

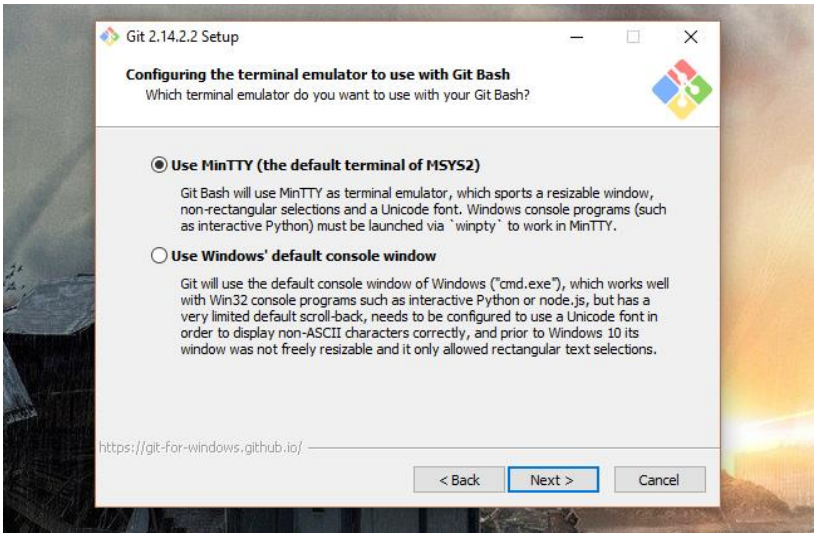

Figure 7.9 Installation of Git (Step 8)

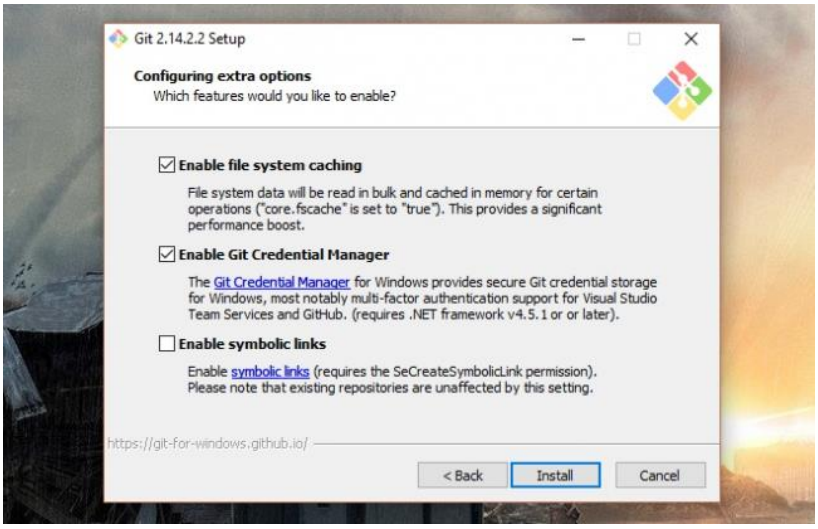

Figure 7.10 Installation of Git (Step 9)

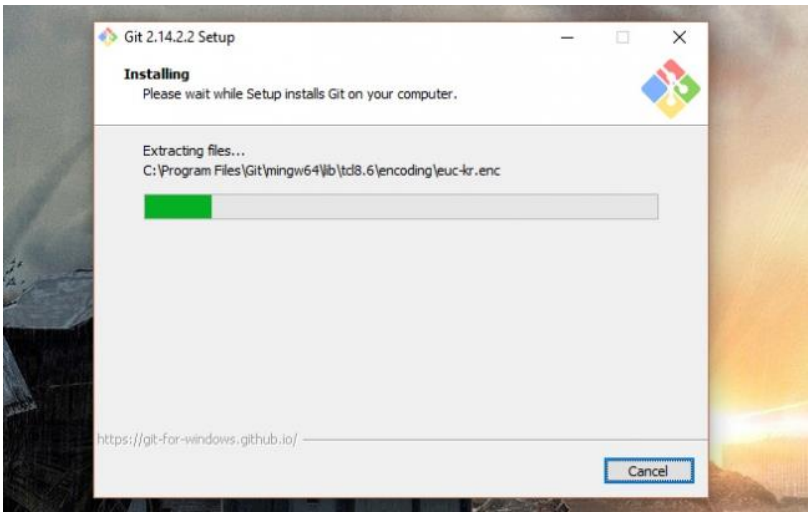

Figure 7.11 Installation of Git (Step 10)

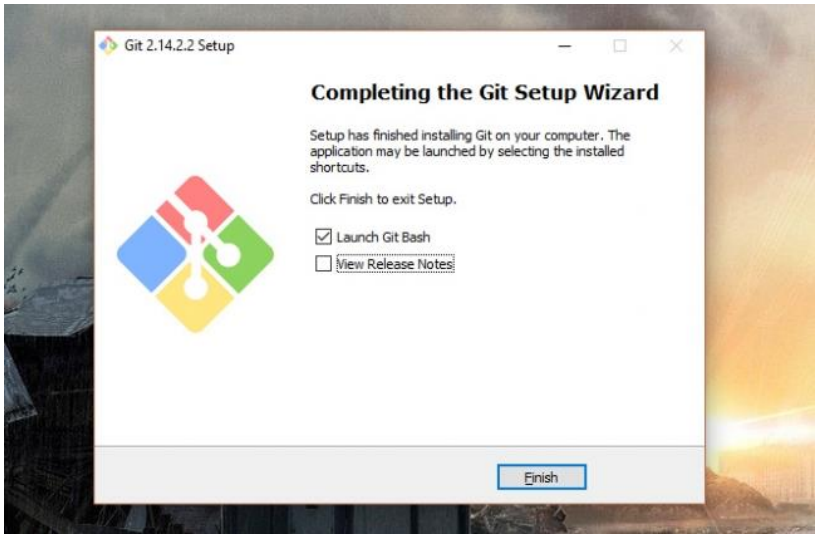

Figure 7.12 Installation of Git (Step 11)

7.2.4 DOWNLOADING HEROKU CLI

- Download Heroku CLI from https://devcenter.heroku.com/articles/heroku-cli

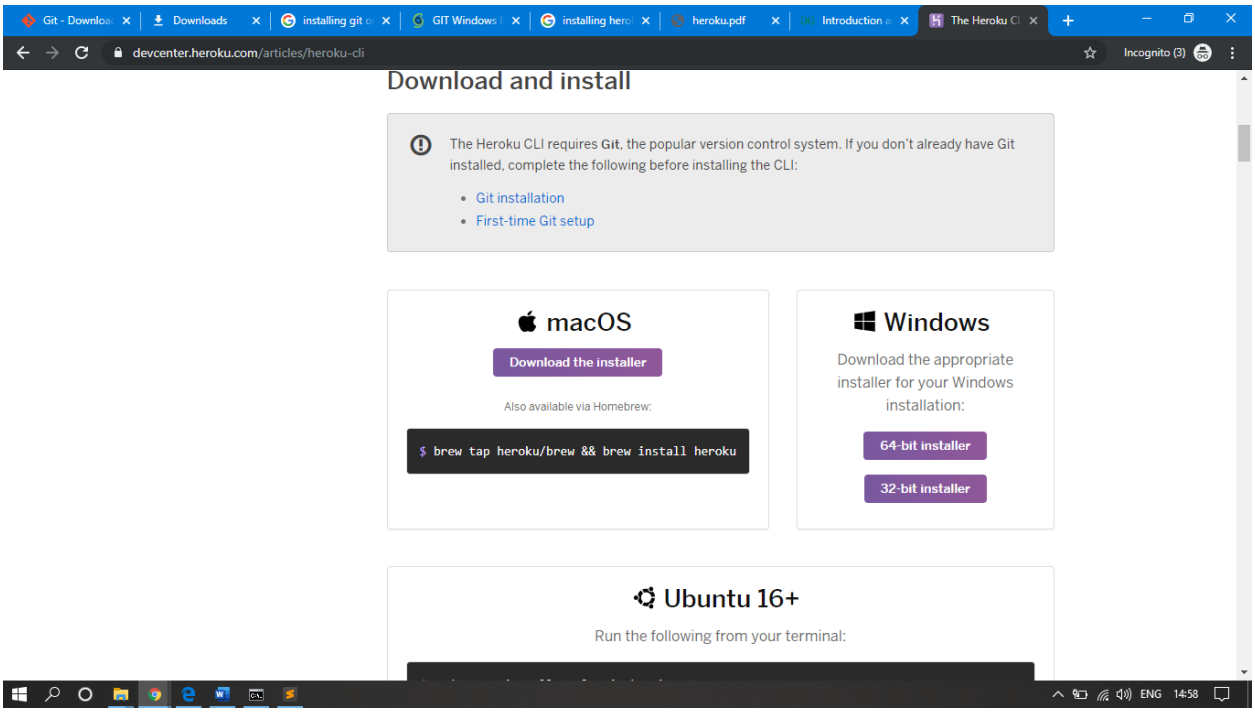

Figure 7.13 Downloading of Heroku CLI

7.2.5 INSTALLIING HEROKU CLI

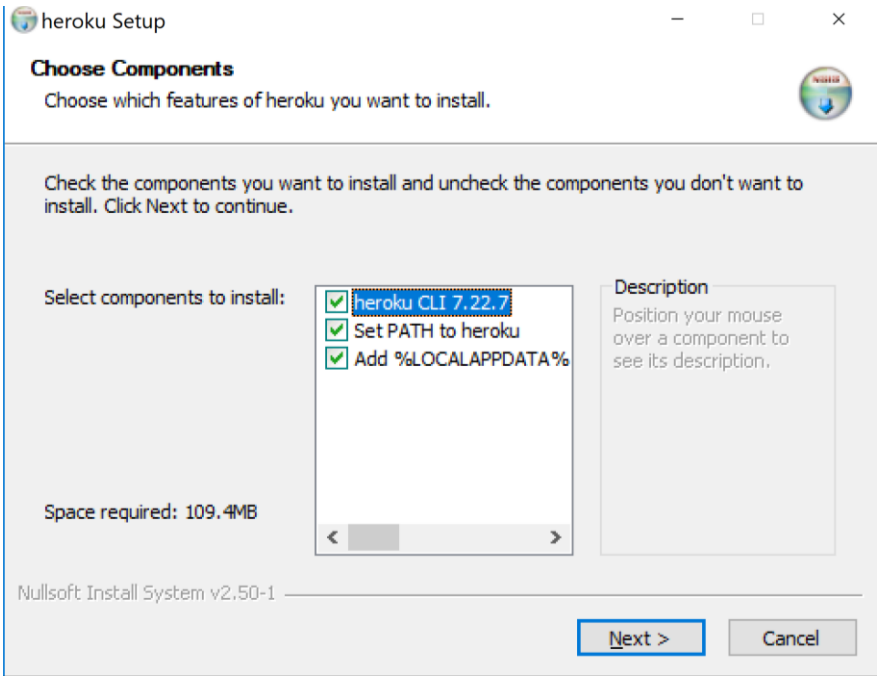

Figure 7.14 Installation of Heroku CLI (Step 1)

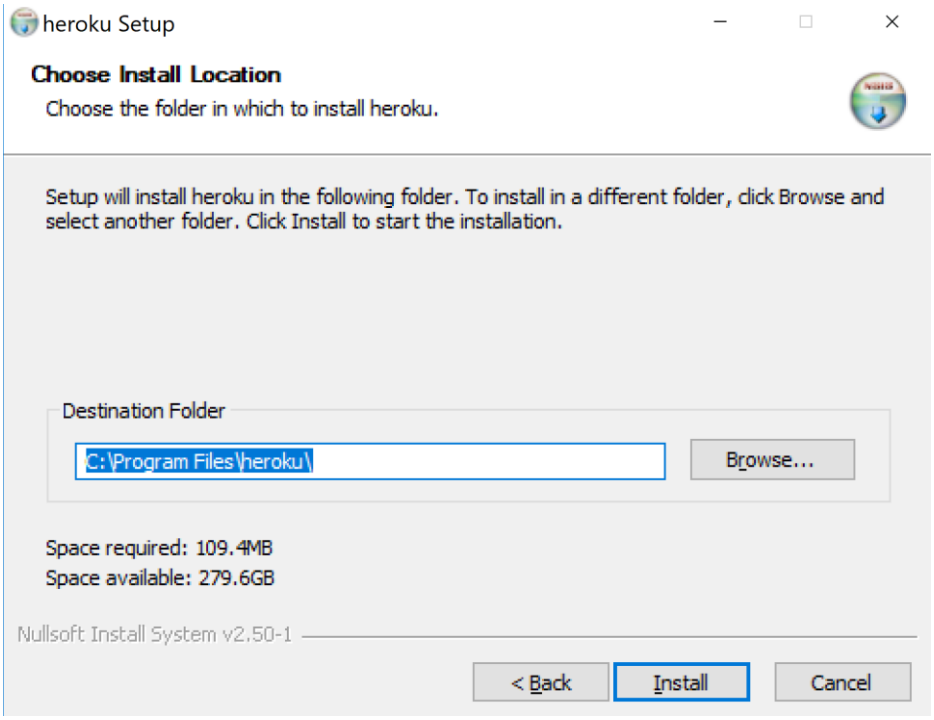

Figure 7.15 Installation of Heroku CLI (Step 2)

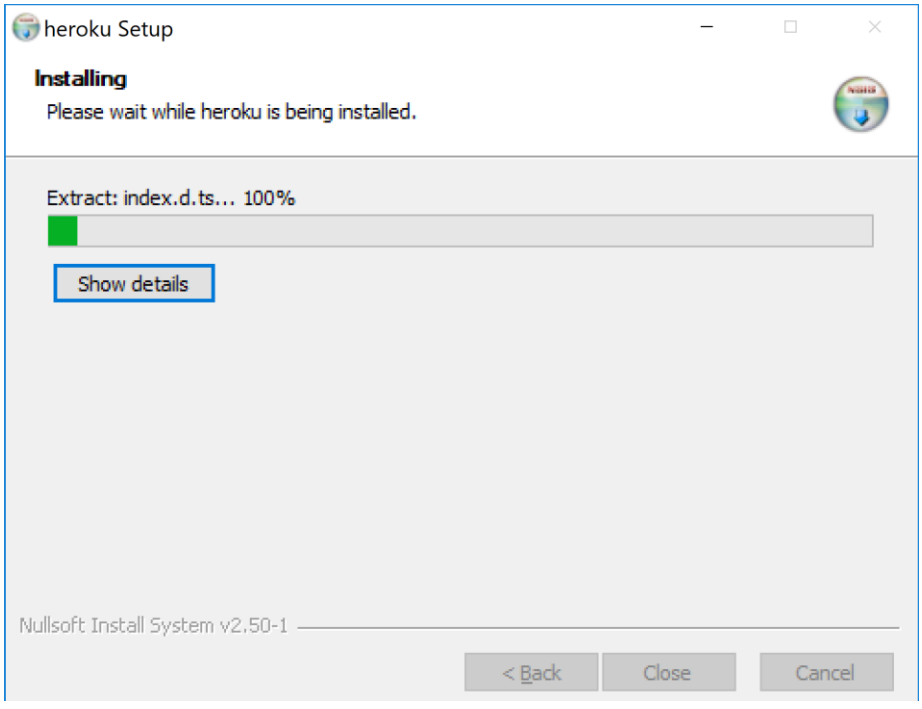

Figure 7.16 Installation of Heroku CLI (Step 3)

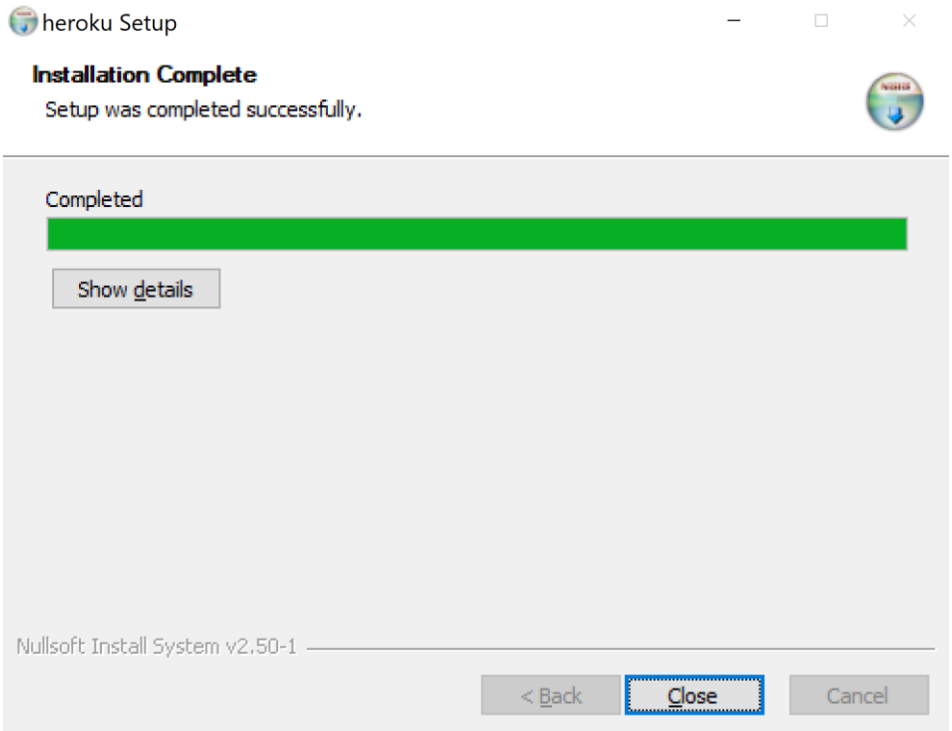

Figure 7.17 Installation of Heroku CLI (Step 4)

7.2.6 DEPLOYING WEBSITE

7.2.6.1 PREREQUISTIES

- Must have an account on Heroku.

7.2.6.2 DEPLOYING WEB

- Open the Project Directory with cmd

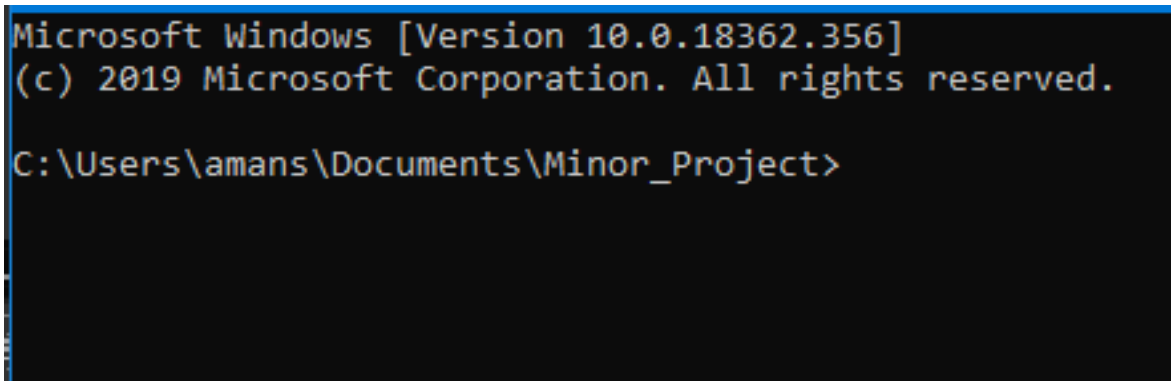

Figure 7.18 Project Directory

- Login with your Heroku account using command : **heroku login -i**

```
C:\Users\amans\Documents\Minor_Project>heroku login -i
heroku: Enter your login credentials
Email: shinu.sistec@gmail.com
Password: **********
Logged in as shinu.sistec@gmail.com

C:\Users\amans\Documents\Minor_Project>
```

Figure 7.19 Heroku Login

- Before using the git commands to send our files to Heroku, you need to tell git who you are. To do that, type this in the command line:

  git config --global user.email "you@example.com"

- Press enter and then type: **git config --global user.name "Your Name"**

```
C:\Users\amans\Documents\Minor_Project>git config --global user.email "aman.sahay09@gmail.com"

C:\Users\amans\Documents\Minor_Project>git config --global user.name "Aman Sahay"

C:\Users\amans\Documents\Minor_Project>
```

Figure 7.20 Git Configuration

- Create a local git repository by typing: **git init**

```
C:\Users\amans\Documents\Minor_Project>git init
Initialized empty Git repository in C:/Users/amans/Documents/Minor_Project/.git/

C:\Users\amans\Documents\Minor_Project>
```

Figure 7.21 Creating Local Git Repository

- Add all your local files to the online repository by typing: **git add .**

```
C:\Users\amans\Documents\Minor_Project>git add .
warning: LF will be replaced by CRLF in app/static/css/file2.css.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in app/static/css/style.css.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in app/templates/index.html.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in app/templates/res.html.
```

Figure 7.22 Adding Local File to Online Repository

- Commit your files with: **git commit –m "First commit"**

```
C:\Users\amans\Documents\Minor_Project>git commit -m "Firtst Commit"
Auto packing the repository in background for optimum performance.
See "git help gc" for manual housekeeping.
Enumerating objects: 9844, done.
Counting objects: 100% (9844/9844), done.
Delta compression using up to 4 threads
Compressing objects:  87% (8483/9750)
```

Figure 7.23 Commit Files

- Create an empty Heroku app: **heroku create appname**

  Here appname is the name that you want for your website's url. Once the command is executed successfully, you can see that your app has been created in your Heroku account under the Personal Apps menu.

- And the final step! Let's push our application to Heroku: **git push heroku master**

```
C:\Users\amans\Documents\Minor_Project>git push heroku master
Enumerating objects: 9844, done.
Counting objects: 100% (9844/9844), done.
Delta compression using up to 4 threads
Compressing objects: 100% (8805/8805), done.
Writing objects:   4% (431/9844), 10.44 MiB | 1.21 MiB/s
```

Figure 7.24 Push Application to Heroku

7.3 SOFTWARE DEVELOPMENT LIFE CYCLE MODEL

Software Development Life Cycle (SDLC) is a process used by the software industry to design, develop and test high quality software. The SDLC aims to produce a high-quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates.

- SDLC is the acronym of Software Development Life Cycle.
- It is also called as Software Development Process.
- SDLC is a framework defining tasks performed at each step in the software development process.
- ISO/IEC 12207 is an international standard for software life-cycle processes. It aims to be the standard that defines all the tasks required for developing and maintaining software.

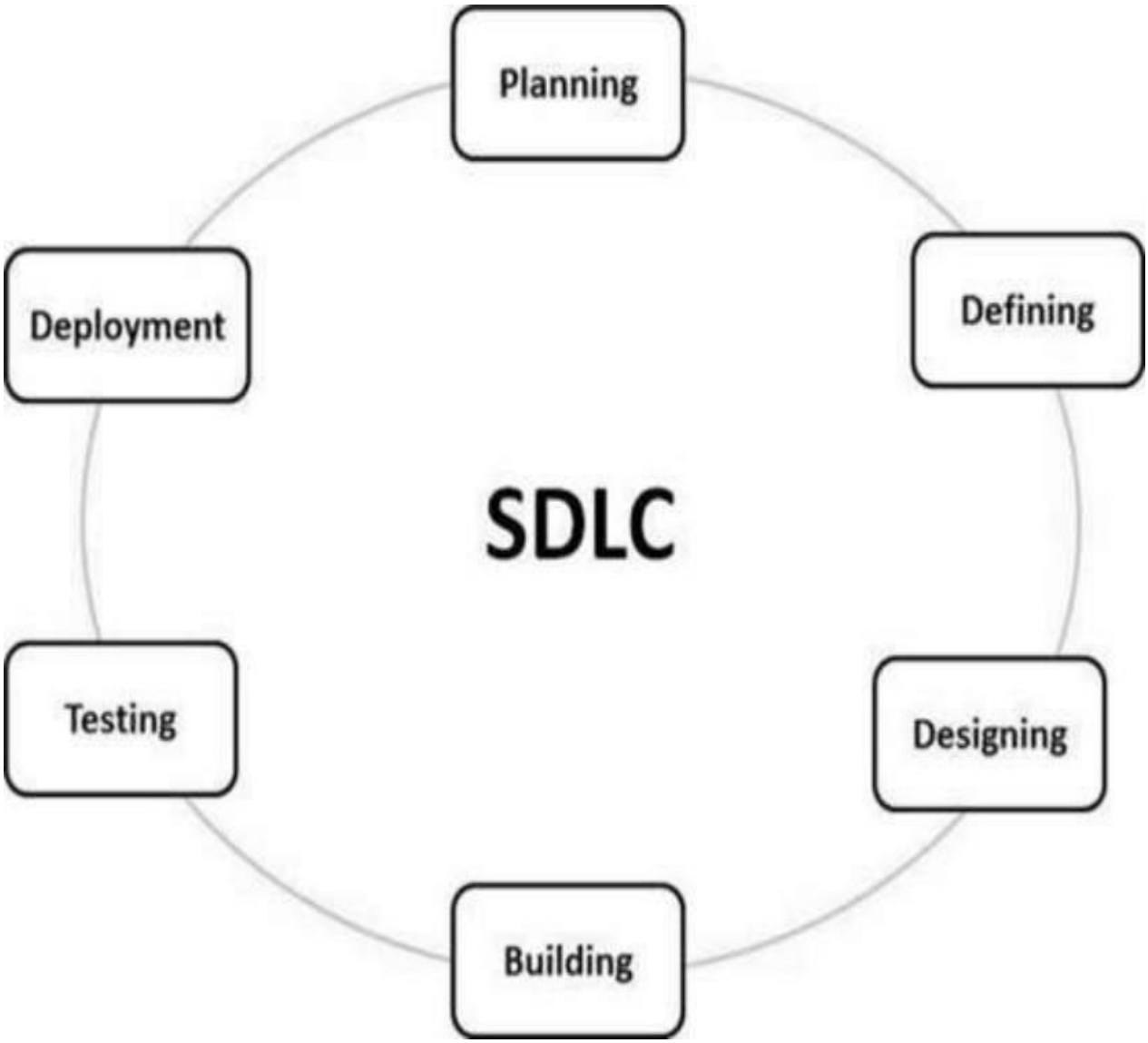

Figure 7.25 SDLC Models

7.3.1 Agile Model

Agile SDLC model is a combination of iterative and incremental process models with focus on process adaptability and customer satisfaction by rapid delivery of working software product. Agile Methods break the product into small incremental builds. These builds are provided in iterations. Each iteration typically lasts from about one to three weeks. Every iteration involves cross-functional teams working simultaneously on various areas like –

- Planning
- Requirements Analysis
- Design
- Coding
- Unit Testing and
- Acceptance Testing.

Agile model believes that every project needs to be handled differently and the existing methods need to be tailored to best suit the project requirements. In Agile, the tasks are divided to time boxes (small time frames) to deliver specific features for a release.

Iterative approach is taken and working software build is delivered after each iteration. Each build is incremental in terms of features; the final build holds all the features required by the customer.

Here is a graphical illustration of the Agile Model –

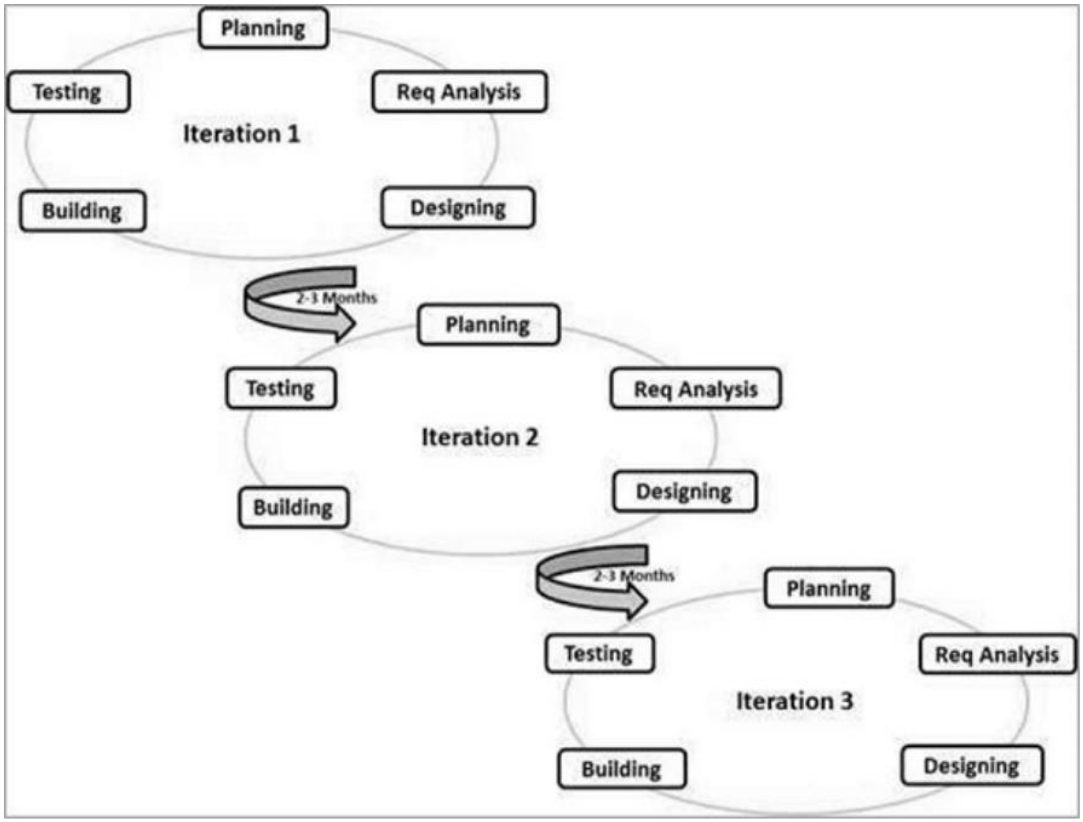

Figure 7.26 Agile Model

References

REFERENCES

WEBSITES (with exact URL up to page)

[1] https://arxiv.org/pdf/1411.4555.pdf

[2] https://www.analyticsvidhya.com/blog/2017/04/natural-language-processing-made-easy-using-spacy-%E2%80%8Bin-python/

[3] https://devcenter.heroku.com/articles/git

[4] https://www.tutorialspoint.com/flask/index.htm

[5] https://www.tutorialspoint.com/css/

APPENDIX-1 GLOSSARY OF TERMS

B

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

C

CSS CSS Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language. Although most often used to set the visual style of web pages and user interfaces written in HTML and XHTML, the language can be applied to any XML document, including plain XML, SVG and XUL, and is applicable to rendering in speech, or on other media. Along with HTML and JavaScript, CSS is a cornerstone technology used by most websites to create visually engaging web pages, user interfaces for web applications, and user interfaces for many mobile applications.

E

ER-Diagram An entity relationship diagram (ERD) shows the relationships of entity sets stored in a database. An entity in this context is a component of data. In other words, ER diagrams illustrate the logical structure of databases.

At first glance an entity relationship diagram looks very much like a flowchart. It is the specialized symbols, and the meanings of those symbols, that make it unique.

F

Flask Flask is a micro web framework written in Python. It is classified as a microframework 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. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools. Extensions are updated far more frequently than the core Flask program.

G

Google Colab

Google Colab is a free cloud service and now it supports free GPU! You can; improve your Python programming language coding skills. develop deep learning applications using popular libraries such as Keras, TensorFlow, PyTorch, and OpenCV.

H

HTML

Hypertext Markup Language (HTML) is the standard markup language for creating web pages and web applications. With Cascading Style Sheets (CSS) and JavaScript it forms a triad of cornerstone technologies for the World Wide Web. Web browsers receive HTML documents from a web server or from local storage and render them into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

Heroku

Heroku is a cloud platform as a service (PaaS) supporting several programming languages. One of the first cloud platforms, Heroku has been in development since June 2007, when it supported only the Ruby programming language, but now supports Java, Node.js, Scala, Clojure, Python, PHP, and Go. For this reason, Heroku is said to be a polyglot platform as it has features for a developer to build, run and scale applications in a similar manner across most languages. Heroku was acquired by Salesforce.com in 2010 for $212 million.

J

JavaScript

JavaScript, often abbreviated as JS, is a high-level, interpreted scripting language that conforms to the ECMAScript specification. JavaScript has curly-bracket syntax, dynamic typing, prototype-based object-orientation, and first-class functions.

Alongside HTML and CSS, JavaScript is one of the core technologies of the World Wide Web. JavaScript enables interactive web pages and is an essential part of web applications. The vast majority of websites use it, and major web browsers have a dedicated JavaScript engine to execute it.

As a multi-paradigm language, JavaScript supports event-driven, functional, and imperative (including object-oriented and prototype-based) programming styles. It has APIs for working with text, arrays, dates, regular expressions, and the DOM, but the language itself does not include any I/O, such as networking, storage, or graphics facilities. It relies upon the host environment in which it is embedded to provide these features.

P

Python

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. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.