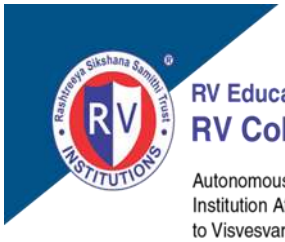


Go, change the world

A Project Report on

DEVELOPMENT OF A WEB SEARCH ENGINE USING WEB SCRAPING

Submitted in Partial Fulfillment of the Requirement

for the IV Semester MCA Academic Minor Project – I
18MCA46

MASTER OF COMPUTER APPLICATIONS

By

USN	STUDENT NAME
1RD19MCA31	VEENA REDDY
1RD19MCA32	VIJAYLAXMI PATIL

Under the Guidance of

Dr. S. Anupama Kumar

Associate Professor

Department of Master of Computer Applications
RV College of Engineering®, Mysuru Road
RV Vidyanikethan Post, Bengaluru – 560059

June -2021

RV Educational Institutions ®
RV College of Engineering ®
Autonomous Institution Affiliated to Visvesvaraya Technological University, Belagavi
Approved by AICTE, New Delhi
Accredited by NBA

Go, change the world

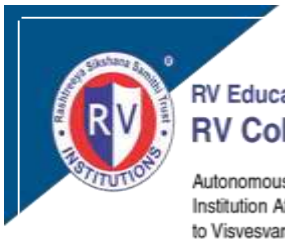
DEPARTMENT OF MASTER OF COMPUTER APPLICATIONS

CERTIFICATE

This is to certify that the project entitled **"Development of a Web Search Engine using Web Scraping"** submitted in partial fulfillment of Minor Project-I (18MCA46) of IV Semester MCA is a result of the bonafide work carried out by Veena Reddy (1RD19MCA31) and Vijaylaxmi Patil (1RD19MCA32), duringthe Academic year 2020-21.

Dr. S. Anupama Kumar
Associate Professor
Department of MCA,
RV College of Engineering®

Dr. Andhe Dharani
Professor and Director
Department of MCA,
RV College of Engineering®



RV Educational Institutions ®
RV College of Engineering ®

Autonomous Institution Affiliated to Visvesvaraya Technological University, Belagavi

Approved by AICTE, New Delhi

Accredited by NBA

Go, change the world

UNDERTAKING BY THE STUDENT

We, Veena Reddy (1RD19MCA31) and Vijaylaxmi Patil (1RD19MCA32) hereby declare that the Minor project-I "Development of Web Search Engine using Web Scraping" is carried out and completed successfully by us and is our original work.

SIGNATURE

Veena Reddy

Vijaylaxmi Patil

Acknowledgement

The satisfaction and euphoria that accompany the success of any work would be incomplete unless we mention the name of the people, who made it possible, whose constant guidance and encouragement served a beacon light and served our effort with success.

We express our whole-hearted gratitude to Dr. K.N. Subramanya, Principal, R V College of Engineering for providing me an opportunity and support to complete the project.

We express our special thanks to Dr. Andhe Dharni., Professor and Director, Department of MCA, R V College of Engineering for her constant support and guidance.

We express our special thanks to Dr. S. Anupama Kumar, Associate Professor, Department of MCA, R V College of Engineering for his constant support and guidance.

On a moral personal note, our deepest appreciation and gratitude to my beloved family and friends, who have been a fountain of inspiration and have provided unrelenting encouragement and support.

Date-29-05-2021

Veena Reddy

(1RD19MCA31)

Vijaylaxmi Patil

(1RD19MCA32)

Abstract

The World Wide Web (WWW) allows people to share information or data from the large database repositories globally. We need to search the information with specialized tools known generically as search engines. There are many search engines available today, where retrieving meaningful information is difficult. However to overcome this problem of retrieving meaningful information intelligently in common search engines, semantic web technologies are playing a major role. Web search engines are built to serve all users, for retrieving relevant information from web, independent of the special needs of any individual user. Personalization of Web search is to carry out information retrieval for each user assimilating his/her interests.

In this Project, a web search engine is developed for searching web pages and images using python programming language. It includes three modules namely, Registration, Search and Result module.

CONTENTS

LIST OF TABLES

Chapter 1: Introduction

This chapter gives the brief introduction about the project.

1.1 Project Description

Search Engine provides the gateway for most of the users trying to explore the huge information base of web pages. Search engines are a program that search documents for specified keywords on search for information on the World Wide Web and returns a list of the documents where the keywords were found. the term is often used to specifically describe systems like Google, Bing and Yahoo! search, that enable users to search for documents on the World Wide Web. In this Project, the Development of Web Search Engine is build through Web Scraping.

Web scraping is a technique to fetch data from websites. While surfing on the web, many websites don't allow the user to save data for personal use. One way is to manually copy-paste the data, which both tedious and time-consuming. Web Scraping is the automation of the data extraction process from websites. This event is done with the help of web scraping software known as web scrapers. They automatically load and extract data from the websites based on user requirements. These can be custom built to work for one site or can be configured to work with any website.

The information extracted using web scraping can be used to replicate in some other website or can be used to perform data analysis. The uses and reasons for using web scraping are as endless as the uses of the World Wide Web. Web Scraping tools are specifically developed for extracting data from the internet. Also, known as web harvesting tools or data extraction tools, they are useful for anyone trying to collect specific data from websites as they provide the user with structured data extracting data from a number of websites.

Since the internet was created, people have stored and published vast quantities of easily accessible online data. The internet now holds an unprecedented amount of valuable information.
Search engines are important for finding, sorting, storing and classifying the importance of online information.

Chapter 2: Literature Review

This chapter gives insight of the literature survey.

2.1 Literature Survey

Sl.no	Author and Paper title	Details of Publication	Summary of the Paper
1	Martin Breuss "Beautiful Soup : Build a Web Scraper With Python"	This page was published on 15 March 2021, at 11:42,	• Details about web scraping • Use of beautiful soup in implementation
2	"A Novel Web Scraping using the additional information obtained from web pages"	IEEE ACCESS,Date of publication March 31, 2020, Department of Computer Engineering, Çorlu Faculty of Engineering, Tekirdag Namik Kemal University, 59860 Tekirdag, Turkey	• This paper gives the detail about design and the algorithms to be used developing the search engine
3	Bo Zhao, "Web Scraping", University of Washington Seattle	May 2017 Publisher: Springer International Publishing AG (outside the USA) 2017 L.A. Schintler, C.L. McNeely (eds.), Encyclopedia of Big Data, DOI 10.1007/978-3-319-32001-4_483-1	• The details about web extraction , acquiring web resources • Extracting desired information from the acquired data.
4	Geeks for Geeks "Scraping the web data from google using python"	27 May 2020, 5th Floor, A-118, Sector-136, Noida, Uttar Pradesh - 201305	• Fetching the data from the websites, modules required and the implementation part.

5	Alberto Martin Martin "How I scraped a data from the Google scholar"	How I scraped a data from the Google scholar, Alberto Martin Martin	• It has a lot of data about free-to- read publications, and we know that Google Scholar links to sources for articles that are not covered by other databases.
6	Amint Phaujdar "8 Best Web ScrapingTools, on Web Scraping"	This page was published on 6 February 2021.	• The contents like methods of scraping, uses of web scrapping, scrapping tools and scripts which help in the development of search engine.
7	Sneha Nain, Bhumika Lall , "Web Data ScraperTools: Survey"	Published: 31/05/2014 Faculty of Computer Science Department, MDU University, India	• Survey on Web Data Scraper tools in order to extract the data from the web. • How tools help the user in extracting the data without using much effort.
8	"Working of web scraping"	2 May, 2020	• Details about the web scraping,working, theoretical concepts of scraping.
9	DI Dr. Gerd Holweg "Web scraping data extraction from websites"	Published on 04.02.2018 Bachelor of Sciencein Engineering at the University of Applied Sciences Technikum Wien–Degree Program	• Results in purpose, market analysis and research of scraping, software tools required.

		Business Informatics.	
10	O' Reilly Ryan Mitchell, " Web scraping with Python"	Published by O'Reilly Media, Inc., 1005 Gravenstein Highway, North, Sebastopol, CA 95472.	• explores a variety of more specific tools and applications to fit any web scraping

2.1 Existing and Proposed System

Exisiting System

In Exisiting system, the World Wide Web has contributed a lot in searching information. But still there is room for improvement because current search engines do not consider the specific user's interest and serves each user equally. For the generic search engine, it become difficult to identify what the user actually want. When different users give same query, same result will be returned by a typical search engine, no matter which user submitted the query. This might not be appropriate for users which require different information. While searching for the information from the web, users need information based on their interest. For the same keyword two users might require different piece of information. There is no central server. It works in distributed environment. Since it has a distributed environment, so it may not be aware about others collection. Hence problem of duplicity occurs and so for this more network bandwidth is consumed.

Proposed System

Usually, with the help of present web search engines users often miss the goal of their searching or receive the ambiguous results. We have proposed a simple and efficient model which ensures good suggestions as well as promises for effective and relevant information retrieval. It exploits user information and search context to learning in which sense a query refer. Proposed approach reduces duplicity problem. So it also reduces network bandwidth. It has a central server.

Gaps identified

1. The developed search engine doesn't include the ADs as like other search engines.
2. The developed search engine doesn't have a search history.

2.2 Tools and Technologies used

Tool : PyCharm

Web frame: Django

Frontend : HTML5 and CSS

Backend : MySQL

Packages Used : Beautiful Soup, request.

2.3 Hardware and Software Requirements

To develop the project, the programmer needs certain tools. These tools are easily available to any of the developer or programmer. The tools are:

1. Desktop/ Laptop: To install and use the tools the programmer needs a windows or Unix platform laptop or desktop.

2. Processor core i3 or more: Since the tool needs to fetch the data from multiple sites, so time processing speed must be faster.

3. RAM min 4GB or higher: The RAM should be at least 4GB otherwise the compilations and reporting will be slower.

4. Hard disk: 5GB or more.

5. Ethernet Connection(LAN) or Wireless adapter (Wi-Fi): Minimum of 2Mbps or more.

6. I/O Device: Monitor, Mouse, Keyboard.

2.4 Software Requirements

The software requirements are description of features and functionalities of the target system. Requirements convey the expectations of users from the software product.

Operating System : Linux, Windows

Programming Language : Python, HTML5, CSS

IDE : PyCharm

Web frame : Django

Browser : Chrome, Firefox

Chapter 3: Software Requirement Specifications

3.1 Introduction

System Requirements Specifications, often recognized by SRS, has been the expression used to designate the comprehensive overview of the technology platform to be created. This is known to be among the early phases of development. Consider it as a diagram that leads you to your final result.
Search Reporter System (SRS) along with a search engine offers that when we search for a keyword, then it provides the search result exactly related to the keywords. The SRS along with a search engine is very user friendly and simple also. A user can easily search and get the informative result from this system. The SRS is designed in such a way that a user will never feel boring with the system. If a user searches for a keyword, then the search reporter loads the search results from any search engine such as Ask.com, Bing.com Etc..,
By the passing of time the use of search engine is increasing. As increased use of search engine for searching information, a system has been developed that helps users to search information. When a person wants to search anything he simply places his words in search engine. Then search engine returns him relevant information according to his/her words based on many more criteria and user can view necessary information.
On the stage of developing the SRS at first, admin places a keyword in the field that is defined for him. Then the system which is connected to any search engine such as Google will get all the titles, URLs and descriptions and then check and count the keyword in the web pages of the URLs. The the titles, URLs, descriptions, number of matches of the keyword and an associated id against the keyword are presented to the user.

3.2 General Description

Product Perspective: Most people who are using a search engine are doing it for research purposes. They are generally looking for answers or at least to data with which to make a decision. They're looking to find a site to fulfill a specific purpose.
Search engines aid in organizing the vast amount of information that can sometimes be scattered in various places on the same web page into an organized list that can be used more easily.

Product Functions: Application will take the user query and provide the relevant information.

User Characteristics: User should be able to see the related links, search engine relevant pages and images.

General Constraints: The project is build through web scraping and scraping usually extracts the data from the websites and presents the related searches, images and links.

Assumptions and Dependencies: The search engine is dependent on the user query and also the website through which web scraping is carried out.

3.3 Functional Requirement

• When user enter a query fetch all the web links, related search links and present it to the user.

• User clicks on images button it should fetch all the images related to that query and present it to the user.

• Spell checking after user enter a query to search.

Registration Page :

Input – Username, Password and Email ID.

Process – Authentication of the user using the front and back end.

Output – Redirection to the Home Page.

Search Page :

Input – User Query

Process – Searches for the relevant pages.

Output – Redirects to the result module.

Result Page :

Input - User Query

Process - Extracting relevant images and web page links.

Output - Image web pages and related search result.

3.4 External Interfaces Requirements

User Interface: The user shall enter a query of his/her own. This is done by registering or logging in directly. Once after clicking the search button, the search engine displays the relevant information required by the user.

3.5 Non Functional Requirements

- **Availability** – The website will be running continuously for 24/7 which will make it accessible to customers all the time.
- **Usability -** The website will have a user friendly interface which allow all users to seamlessly interact with it.
- **Extensibility** - The website will be extensible in case there will be need to add new features and requirements.
- **Performance** - The website will be able to handle any number of simultaneous users at a given time.

Chapter 4: System Design

4.1 System Perspective /Architectural Design

4.1.1 Problem Specification - Search engines provide users with the information they are looking for, and not necessarily the information that marketers would like them to see. Search engines essentially acts as filters for the wealth of information available on the internet.

On the basis of recent studies made on the structure and dynamics of the web itself, it has been analyzed that the web is still growing at a high pace.

4.1.2 Block Diagram: A block diagram is a diagram of a system in which the principal parts or functions are represented by blocks connected by lines that show the relationships of the blocks.

User enter the query using keywords fetch related SERP, images and related search links and present it to the user

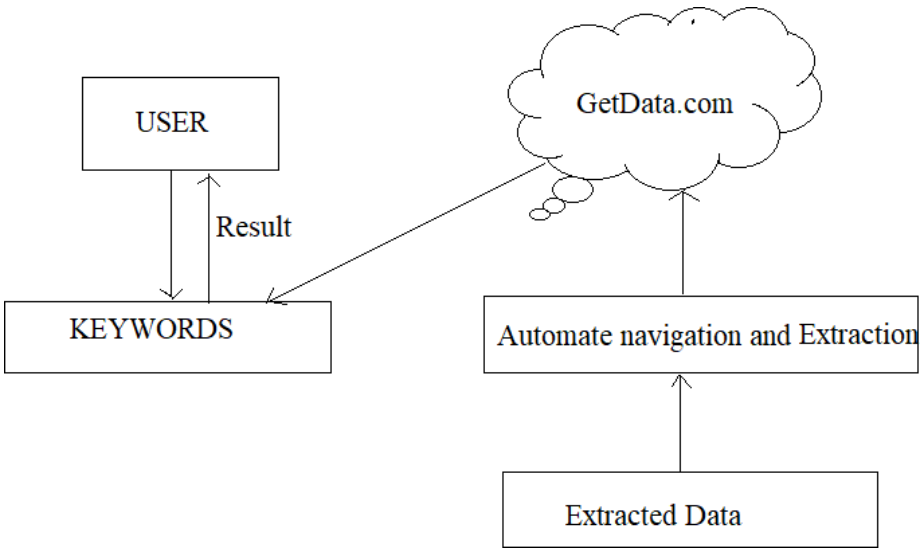


Fig 4.1.2 Block Diagram

Data definition/Dictionary: SERP – Search Engine Result Pages.

Module specification: Registration module, Search module, Result module.

4.2 Context Diagram

The context diagram is used to establish the context and boundaries of the system to be modeled: which things are inside and outside of the system being modeled, and what is the relationship of the system with these external entities.

User can register into the website and login to the system and admin can login and view and manage all user account

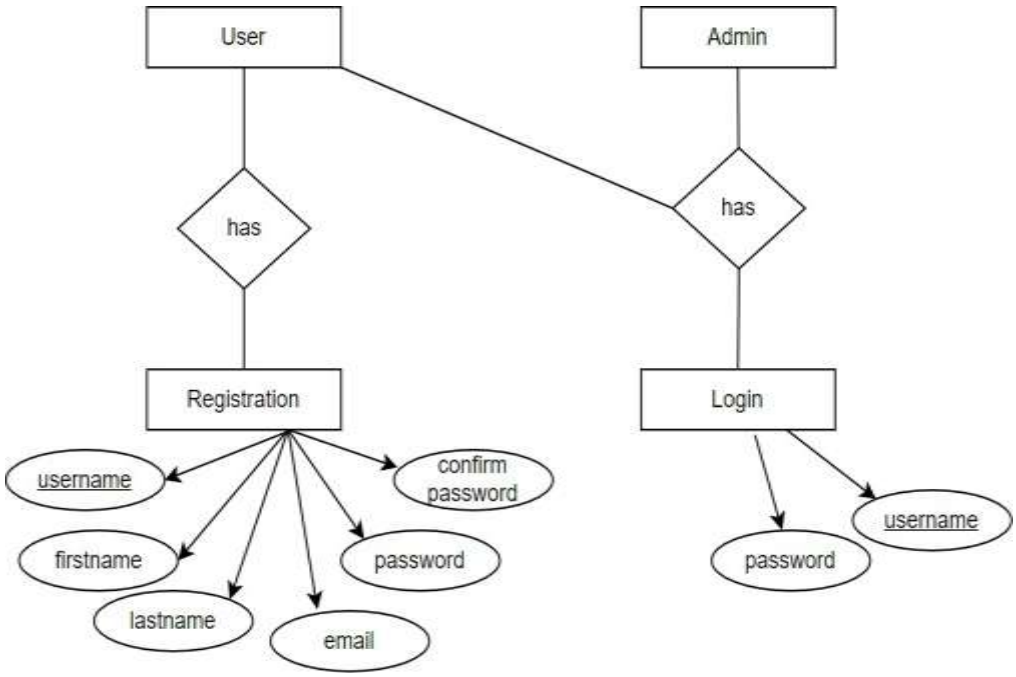


Fig 4.1.3 Context Diagram

Chapter 5: Detailed Design

5.1 System Design

5.1.1 Dynamic Model: Dynamic Model involves states, events and state diagram (transition diagram) on the model. Main concepts related with Dynamic Model are states, transition between states and events to trigger the transitions. Predefined relationships in object model are aggregation (concurrency) and generalization.

5.1.1.1 Class Diagram:

User can register, login, logout and search for the result, using those keywords system will display SERP, images and related search links

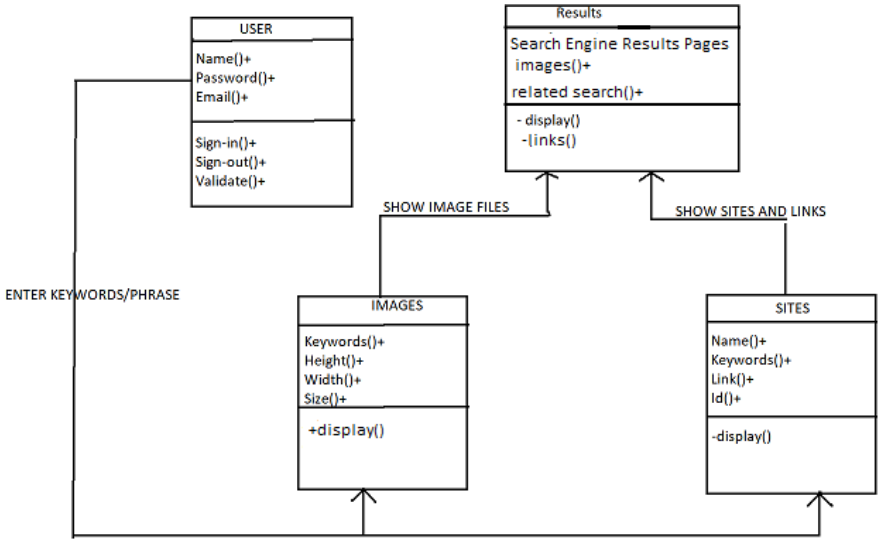


Figure 5.1.1.1 Class Diagram

5.1.1.2 UseCase Diagram

User can register, login , search for websites, search image and login

Admin can login, view all user accounts and manage user account and logout

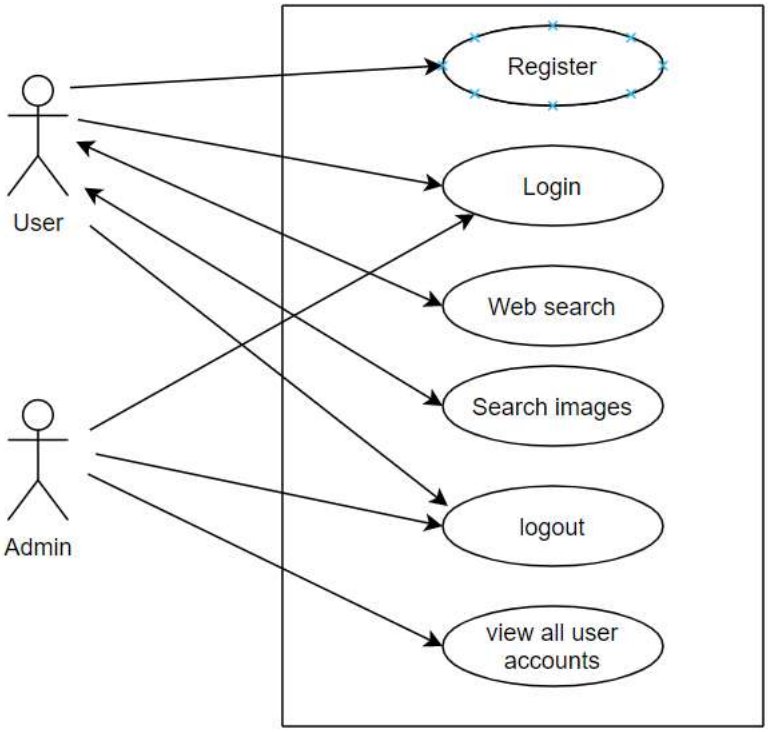


Figure 5.1.1.2 Usecase Diagram

5.1.1.3 Activity Diagram

User has to register in order to create an account then login into the siteand search websites and images

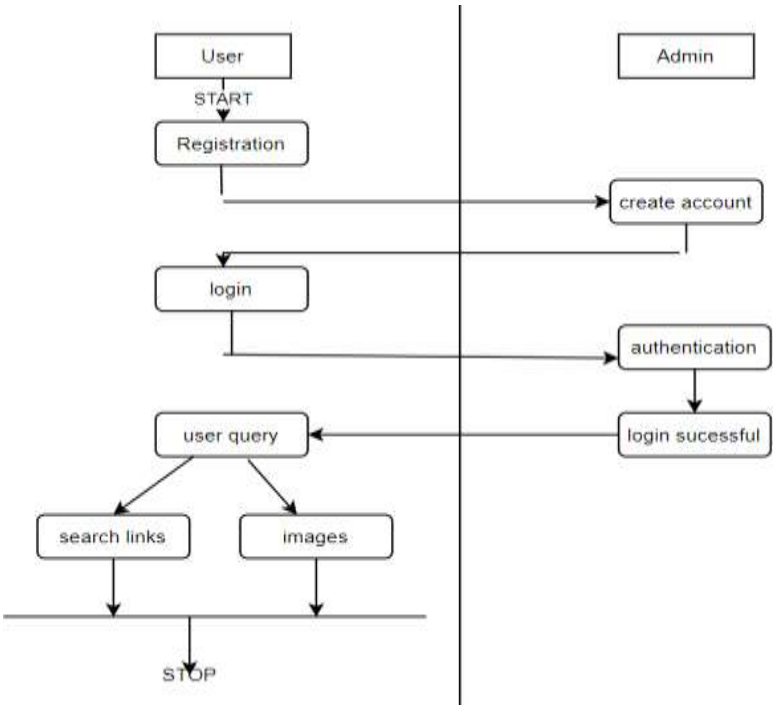


Fig 5.1.1.3 Activity Diagram

5.1.1.4 Sequence Diagram

User has to register himself if registration is successful the login if login is successful then enter the query if misspelled, correct the query the present SERP, image and related search links

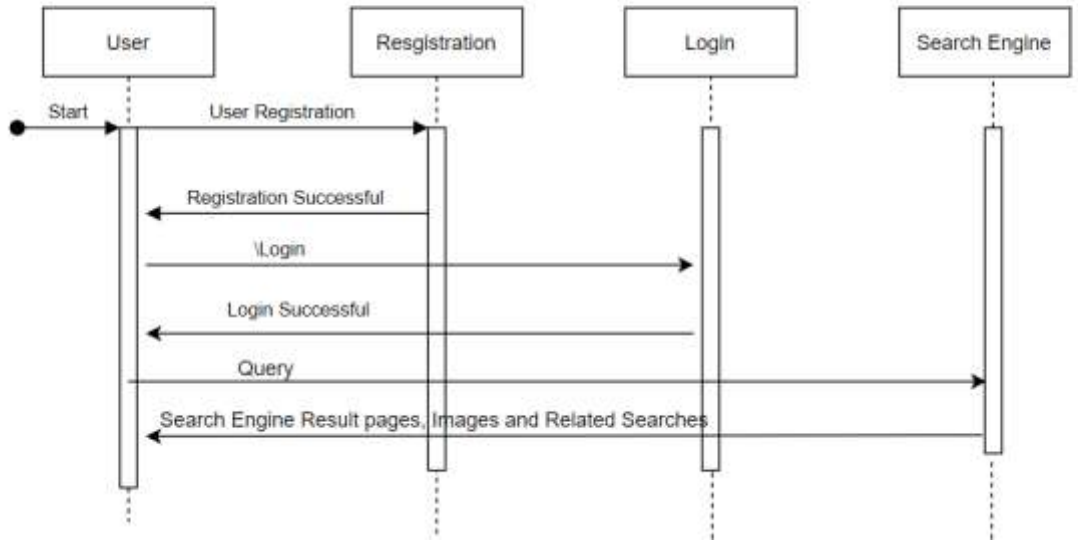


Fig 5.1.1.4 Sequence Diagram

5.1.1Functional Model: Functional Model focuses on the how data is flowing, where data is stored and different processes. Main concepts involved in Functional Model are data, data flow, data store, process and actors. Functional Model in OMT describes the whole processes and actions with the help of data flow diagram (DFD).

Data Flow Diagrams

Level 0:

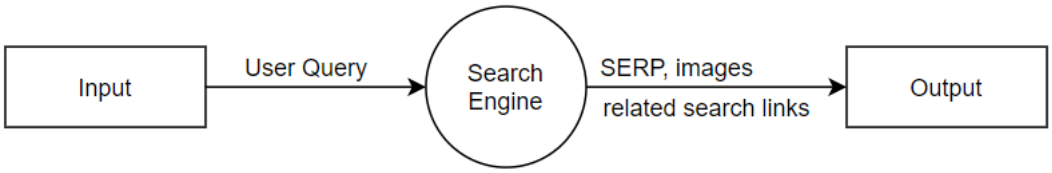


Fig 5.1.2.1 Data Flow Diagram Level 0

Level 1:

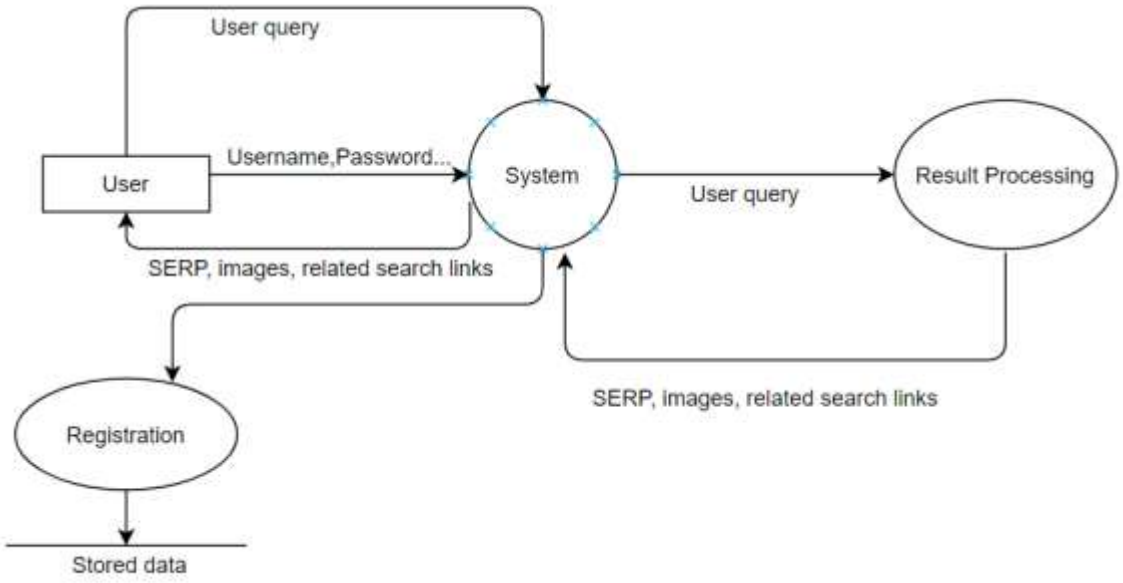


Fig 5.1.2.2 Data Flow Diagram Level

Chapter 6: Implementation

6.1 Code

Project Structure :

urls.py

Django runs through each URL pattern, in order, and stops at the first one that matches the requested URL, matching against path_info . Once one of the URL patterns matches, Django imports and calls the given view, which is a Python function.

```
from django.contrib import admin
from django.urls import path, include
from django.conf import settings
from django.conf.urls.static import static

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('search.url')),

]+ static(settings.STATIC_URL, document_root=settings.STATIC_ROOT)
```

url.py

Django runs through each URL pattern, in order, and stops at the first one that matches the requested URL, matching against path_info . Once one of the URL patterns matches, Django imports and calls the given view, which is a Python function.

```
from django.contrib import admin
from django.urls import path, include
from django.conf import settings
from django.conf.urls.static import static
urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('search.url')),

]+ static(settings.STATIC_URL, document_root=settings.STATIC_ROOT)
```

```
from django.urls import path
from . import views

urlpatterns = [
    path('', views.index, name='index'),
    path('search', views.search, name='search'),
    path('login', views.login, name='login'),
    path('register', views.register, name='register'),
    path('logout', views.logout, name='logout'),
    path('images', views.images, name='images'),
]
```

views.py

It will take a web request and returns a web response, this response can be the HTML page, or a redirect, or an image etc,..

```
from django.shortcuts import render, redirect
from django.contrib import messages
from django.contrib.auth.models import User, auth
import requests
from bs4 import BeautifulSoup as bs
from spellchecker import SpellChecker
# Create your views here.

def index(request):
    return render(request, 'index.html')

def search(request):
    if request.method == 'POST':
        search1 = request.POST['search']
        spell = SpellChecker()
        misspelled = spell.unknown([search1])
        if misspelled:
            print(misspelled)
            for word in misspelled:
                # Get the one `most likely` answer
                print(spell.correction(word))
                search = spell.correction(word)

            print(spell.candidates(word))
                url = 'https://www.ask.com/web?q='+search
```

```
    else:
    url = 'https://www.ask.com/web?q=' + search1
res = requests.get(url)
print("ask = ",res)
soup = bs(res.text, 'lxml')

top_first_related = soup.find_all('ul', {'class': 'PartialRelatedSearch-first-column'})
#print(top)
if not top_first_related:
    final_result1 = []
else:
    for d in top_first_related:
        result_listings1 = d.find_all('li', {'class': 'PartialRelatedSearch-item'})

    #print(result_listings1)

    final_result1 = []

    for result in result_listings1:
        result_text = result.find(class_='PartialRelatedSearch-item-link-text').text
        #result_title = result.find('span').text
        result_url = result.find('a').get('href')
        final_result1.append((result_text, result_url))

    #print(final_result1)

top_second_related = soup.find_all('ul', {'class': 'PartialRelatedSearch-second-column'})
#print
if not top_second_related:
    final_result2 = []
else:
    for d in top_second_related:
        result_listings2 = d.find_all('li', {'class': 'PartialRelatedSearch-item'})

#print(result_listings1)

    final_result2 = []

    for result in result_listings2:
        result_text = result.find(class_='PartialRelatedSearch-item-link-text').text
        #result_title = result.find('span').text
        result_url = result.find('a').get('href')
        final_result2.append((result_text, result_url))

result_listings3 = soup.find_all('div', {'class': 'PartialSearchResults-item'})

final_result3 = []

for result in result_listings3:
    result_title = result.find(class_='PartialSearchResults-item-title').text
    result_url = result.find('a').get('href')
    result_desc = result.find(class_='PartialSearchResults-item-abstract').text
    final_result3.append((result_title, result_url, result_desc))

#print(final_result2)
if misspelled:
```

```
    url1 = 'https://www.bing.com/search?q=' + search
else:
    url1 = 'https://www.bing.com/search?q=' + search1
res1 = requests.get(url1)
print("bing = ",res1)
soup1 = bs(res1.text, 'lxml')

result_listings4 = soup1.find_all('li', {'class': 'b_algo'})
final_result4 = []

for result in result_listings4:
    result_title = result.find('h2').text
    result_url = result.find('a').get('href')
    result_desc = result.find('p').text

    final_result4.append((result_title, result_url, result_desc))
#print(final_result4)

if misspelled:
    res2 = requests.get('https://www.bing.com/images/search?q='+search)
else:
    res2 = requests.get('https://www.bing.com/images/search?q=' + search1)
print("bing = ", res2)
soup2 = bs(res2.text, 'lxml')
img = soup2.find_all('div', {'class': 'imgpt'})

imgres2 = []
for result in img:
    if result is not None:
        url = result.find('a').get('href')
        src = result.find('img').get('src')
        imgres2.append((url,src))


if misspelled:
    context  = {
        'search1': search1,
        'search': search,
        'final_result1': final_result1,
        'final_result2': final_result2,
        'final_result3': final_result3,
        'final_result4': final_result4,
        'imgres':imgres2,
    }
else:
    context = {
        'final_result1': final_result1,
        'final_result2': final_result2,
        'final_result3': final_result3,
        'final_result4': final_result4,
        'imgres': imgres2,
    }

#def images(img):
    #context = {
        #'imgres': img,
    #}
    #return render('images.html', context)
#images=images(imgres2)
```

```
        return render(request, 'search.html', context)


    else:
        return render(request, 'search.html')

def login(request):
    if request.method == 'POST':
        username = request.POST['username']
        password = request.POST['password']

        user = auth.authenticate(username=username, password=password)
        if user is not None:
            auth.login(request, user)
            return redirect('/')
        else:
            messages.info(request, 'invalid Credentials')
            return redirect('login')
    else:
        return render(request, 'login.html')

def register(request):
    if request.method == 'POST':
        first_name = request.POST['first_name']
        last_name = request.POST['last_name']
        username = request.POST['username']
        password1 = request.POST['password1']
        password2 = request.POST['password2']
        email = request.POST['email']

        if password1 == password2:
            if  User.objects.filter(username=username).exists():
                messages.info(request,'Sorry,Username Taken!')
                return redirect('register')
            elif User.objects.filter(email=email).exists():
                messages.info(request, 'Sorry,Email Taken!')
                return redirect('register')
            else:
                newuser = User.objects.create_user(username=username, password=password1, email=email,
first_name=first_name,last_name=last_name);
                newuser.save();
                messages.info(request, 'User created :) ')
                return redirect('login')

        else:
            messages.info(request,'Sorry, passwords are not matching.. ')
            return redirect('register')
        return redirect('/')
    else:
        return render(request,'register.html')

def logout(request):
    auth.logout(request)
    return redirect('/')

def images(request):
    if request.method == 'POST':
        search1 = request.POST['search']
```

```
    spell = SpellChecker()
    misspelled = spell.unknown([search1])
    if misspelled:
        print(misspelled)
        for word in misspelled:
            # Get the one `most likely` answer
            print(spell.correction(word))
            search = spell.correction(word)
            # Get a list of `likely` options
            print(spell.candidates(word))
            url = 'https://www.bing.com/images/search?q=' + search
    else:
        url = 'https://www.bing.com/images/search?q='+search1
    res2 = requests.get(url)
    print("bing = ", res2)
    soup2 = bs(res2.text, 'lxml')
    img = soup2.find_all('div', {'class': 'imgpt'})
    imgres1 = []
    for result in img:
        url = result.find('a').get('href')
        src = result.find('img').get('src')
        imgres1.append((url,src))

    if misspelled:
        url = 'https://www.picsearch.com/index.cgi?q='+search
    else:
        url = 'https://www.picsearch.com/index.cgi?q='+search1
    res3 = requests.get(url)
    print("picsearch = ", res3)
    soup3 = bs(res3.text, 'lxml')

    img3 = soup3.find_all('span', {'class': 'result'})

    imgres3 = []
    for result in img3:
        url = result.find('a').get('href')
        src = result.find('img').get('src')
        imgres3.append((url,src))

    if misspelled:
        context = {
            'search1': search1,
            'search': search,
            'imgres1':imgres1,
            'imgres3' : imgres3,
        }
    else:
        context = {
            'imgres1':imgres1,
            'imgres3' : imgres3,
        }

    return render(request, 'images.html', context)
else:
    return render(request,'images.html')
```

index.html

Here user can write the query inside the search button and also can login, register or logout from the website

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
      <style>
body {
font-size: 10pt;
font-family: arial, sans-serif;
background-color: lavender;
}

a {
text-decoration: none;
}

a:hover {
text-decoration: underline;
}

button {
font-weight: bold;
font-family: arial;
}


/* Top Toolbar */

.top-toolbar {
height: 50px;
float:right;
margin: 7px 21px;
}

.top-toolbar nav a {
margin: 3px 6px;
color: #404040;
}

.top-toolbar nav a:hover {
color: #111111;
}

.top-toolbar nav button {
padding: 7px 12px;
border-radius: 2px;
background-color: #4585F1;
color: white;
border: 1px darkblue;
font-size: 10pt;
}

.top-toolbar nav button:hover {
```

```
.top-toolbar nav img {
margin: 0 7.5px;
height:  25px;
position: relative;
top: 7px;}

/* top left toolbar */

.top-toolbar-left {
height: 50px;
float:left;
margin: 7px 21px;
}

.top-toolbar-left nav a {
margin: 3px 6px;
color: #404040;
font-size: 18px;
}

/* End of Top Toolbar */
/* Search */

.search {
background-color: #DBDBF4;
margin: auto;
width: 700px;
height: 300px;
text-align: center;
clear: both;


}

.logo {
max-width: 21%;
margin: 0 auto;
}

.search img {
margin-bottom: 3%;
max-width: 100%;
}
.rv{
	color: #242F94;
	}

.search input {
height:  25px;
width: 570px;
border-radius:20px;
border: 1px solid #D8D8D8;
padding: 5px;
font-size: 15pt;
/* background: url('images/microphone.png') no-repeat;
background-position: right;
background-size: 4.25%;*/
}

.search-bar {
```

```
max-width: 80%;
}

.search input:focus {
outline: none;
border: 1px solid #4285F4;
}

.search button {
margin: 14px 7px 0 5px;
padding: 7px 8px;
border-radius: 20px;
border: 1px solid #D0D0D0;
color: #444444;
font-size:15px;
height: 35px;
width: 100px;
}

.search button:hover {
box-shadow: 1px 1px 1px rgba(.2,.2,.2,.5);
}

/* End of Search */
/* Footer */

footer {
position: fixed;
bottom: 0;
height: 42px;
width: 100%;
font-size: 9.5pt;
background-color:lightgrey;
}

.footer-left {
float: left;
margin-left: 16px;
}

.footer-right {
float: right;
margin-right: 16px;
}

footer nav {
display: inline-block;
padding: 15px 0 0 0;
}

footer nav a {
padding: 3px 14px;
color: #646464;
}

footer nav a:hover {
color: #000000;
}

/* End of Footer */
```

```
@media screen and (max-width: 510px) {
.top-toolbar nav a:nth-child(2), a:nth-child(3), footer {
display:none;
}

.logo {
max-width: 40%;
}
}
</style>
</head>
<body>
<div class="top-toolbar">
	<nav>
		<img src="" />
		{% if user.is_authenticated %}
		<h5>Hello,{{user.first_name}} {{user.last_name}}</h5>
		<a href="logout"><button>Logout</button></a>
		{% else %}
		<a href="register"><button class="signin" onclick="sign" >Sign Up</button></a>
		<a href="login"><button class="login">Login</button></a>
		{% endif %}
	</nav>
</div>
<div class="top-toolbar-left">
	<nav>
	  <a href="#">Video</a>
	  <a href="images">Images</a>
	</nav>
</div>
<div class="search">
	<div class="logo">
		<img src=""/>
	</div>
	<form method="POST" action="search">
		{% csrf_token %}
		<img src="" />
		<h1 class="rv"> R V College of Engineering </h1>
		<input class="search-bar" type="text" name="search" placeholder="     type here to search" />
		<div class="search-nav">
			<div class="search-button">
				<a href="#"><button type=" submit ">Search</button></a>
			</div>
		</div>
	</form>
</div>
<div class="footer">
	<footer class="bottom-toolbar">
		<nav class="footer-left">
			<a href="#">Advertising</a>
			<a href="#">Business</a>
			<a href="#">About</a>
			<a href="#">Settings</a>
		</nav>

		<nav class="footer-right">
			<a href="#"><B>Developed by Veena Reddy and Vijaylaxmi Patil <B> </a>
		</nav>
```

```
    </footer>
</div>
</body>
</html>
```

search.html

This will fetch all the web links, images and related search links and present it to the user based on keywords.

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
<style>
   body {
   font-size: 10pt;
   font-family: arial, sans-serif;
   background-color: lavender ;
   }

   a {
   text-decoration: none;
   }

   a:hover {
   text-decoration: underline;
   }

   button {
   font-weight: bold;
   font-family: arial;
   }


   /* Top Toolbar */

   .top-toolbar {
   height: 50px;
   float:right;
   margin: 7px 21px;
   }
   .top-toolbar nav a {
   margin: 3px 6px;
   color: #404040;
   }

   .top-toolbar nav a:hover {
   color: #111111;
   }

   .top-toolbar nav button {
   padding: 7px 12px;
   border-radius: 2px;
```

```
    }

    .top-toolbar nav button:hover {
    box-shadow: 1px 1px 0 rgba(0,0,0,.5);
    }

    .top-toolbar nav img {
    margin: 0 7.5px;
    height:  25px;
    position: relative;
    top: 7px;}

    /* top left toolbar */

.top-toolbar-left {
height: 50px;
float:left;
margin: 7px 21px;
}

.top-toolbar-left nav a {
margin: 3px 6px;
color: #404040;
font-size: 18px;
}

    /* End of Top Toolbar */
    /* Search */

    .search {
    margin-left:200px;
    clear: both;
    }

    .logo {
    max-width: 21%;
    margin: 0 auto;
    }

.search input {
margin-top: 1%;
height:  25px;
width: 570px;
border-radius:20px;
border: 1px solid #D8D8D8;
padding: 5px;
font-size: 15pt;
/* background: url('images/microphone.png') no-repeat;
background-position: right;
background-size: 4.25%;*/
}

    .search-bar {
    max-width: 80%;
    }

    .search input:focus {
    outline: none;
    border: 1px solid #4285F4;
```

```
}

.search button {
margin-left:250px;
padding: 7px 8px;
border-radius: 20px;
border: 1px solid #D0D0D0;
color: #444444;
font-size:15px;
height: 35px;
width: 100px;
}

.search button:hover {
box-shadow: 1px 1px 1px rgba(.2,.2,.2,.5);
}

/* End of Search */
/* Footer */

footer {
position: fixed;
bottom: 0;
height: 42px;
width: 100%;
font-size: 9.5pt;
background-color:lightgrey;
}

.footer-left {
float: left;
margin-left: 16px;
}

.footer-right {
float: right;
margin-right: 16px;
}

footer nav {
display: inline-block;
padding: 15px 0 0 0;
}

footer nav a {
padding: 3px 14px;
color: #646464;
}

footer nav a:hover {
color: #000000;
}

/* End of Footer */
/*Images*/
.row::after{
    clear:both;
    display:table;
    }
```

```
    .column{
        float:left;
        padding:5px;}

    @media screen and (max-width: 510px) {
    .top-toolbar nav a:nth-child(2), a:nth-child(3), footer {
    display:none;
    }

    .logo {
    max-width: 40%;
    }
    }
    </style>
</head>
<body>
    <div class="top-toolbar">
        <nav>
            <img src="" />
                {% if user.is_authenticated %}
                <h5>Hello,{{user.first_name}} {{user.last_name}}</h5>
                <a href="logout"><button>Logout</button></a>
                {% else %}
                <a href="register"><button class="signin" onclick="sign" >Sign Up</button></a>
                <a href="login"><button class="login">Login</button></a>
                {% endif %}
        </nav>
    </div>
<div class="top-toolbar-left" >
        <nav>
            <a href="#">Video</a>
            <a href="images">Images</a>
        </nav>
</div>
    <div class="search">
        <div class="logo">
            <img src="" />
        </div>
        <form method="POST" action="search">
            {% csrf_token %}
            <input class="search-bar" type="text" name="search" placeholder="     type here to search" />
            {% if search %}
                <h4> Showing results for {{ search }}</h4>
            {% endif %}
            {% if search1 %}
                <h7> Search instead for {{ search1 }}</h7>
            {% endif %}
        </form>
    </div>
    <br>
    <div>
        <div>
        {% if imgres %}
        <h3 style="padding-left: 200px; color: #646464;">Images</h3>
            {% for result in imgres %}
            <div style="padding-left: 200px;" class="row">
            <div class="column">
                <a href="{{result.0}}">
                    <img alt="image" src="{{result.1}}" height="120" width="120" >
                </a>
```

```
        </div>
        </div>
    {% endfor %}
    <br><br><br><br><br><br><br><br><br>
    {% endif %}
    </div>

{% if final_result3 %}
<h3><b style="padding-left: 200px; padding-right: 50px; color: #646464;">Web Results</b></h3>
{% for result in final_result3 %}
    <div style="padding-left: 200px; padding-right: 50px;">
        <h2 style="color:"><a href="{{ result.1 }}">{{ result.0 }}</a></h2>
        <h4 style="color:">{{ result.1 }}</h4>
        <p>{{ result.2 }}</p> </div>
{% endfor %}
{% else %}
    <div style="padding-left: 200px; padding-right: 50px;">
        <h2 style="color:">Sorry! there are no results for {{search}}</h2>
        <h4 style="color:">Please check your spelling or try different keywords</h4>
    </div><br><br>
{% endif %}


{% if final_result4 %}
{% for result in final_result4 %}
    <div style="padding-left: 200px; padding-right: 50px;">
        <h2 style="color:"><a href="{{ result.1 }}">{{ result.0 }}</a></h2>
        <h4 style="color:">{{ result.1 }}</h4>
        <p>{{ result.2 }}</p></div>
{% endfor %}
{% endif %}

{% if final_result2 %}
    <h3 style="padding-left: 200px; color: #646464;">Related Searchs</h3>
{% for result in final_result2 %}
    <div style="padding-left: 180px;">
    <ul>
        <li style="color:blue;"><a href="{{ result.1 }}">{{ result.0 }}</a></li>
    </ul>
    </div>
{% endfor %}
{% endif %}

    {% if final_result1 %}
{% for result in final_result1 %}
    <div style="padding-left: 180px; padding-right: 50px;">
    <ul>
        <li style="color:blue;"><a href="{{ result.1 }}">{{ result.0 }}</a></li>
    </ul>
    </div>
{% endfor %}
{% endif %}
</div>

<div class="footer">
    <footer class="bottom-toolbar">

    <nav class="footer-left">
        <a href="#">Advertising</a>
```

```
        <a href="#">Business</a>
        <a href="#">About</a>
      </nav>
      <nav class="footer-right">
        <a href="#">New Privacy &amp;amp; Terms</a>
        <a href="#">Settings</a>
      </nav>
      </footer>
   </div>
</body>
</html>
```

register.html

User can enter all details like name, username password etc and register themselves

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <title>Register</title>
   <style>
      html,body{
         background-color: lavender ;
      }
      form{
         height:500px;
         width:500px;
         background-color:#8A4B69;
      }
      input{
         background-color:#D39FB7;
         margin:3px;
         font-size:20px;
      }
      a {
         margin:2px;
         color: black;
      }
   </style>
</head>
<body>
   <center>
      <br><br>
   <form action="register" method="post" >
      <br>
      <h2>REGISTRATION</h2>
      {% csrf_token %}
      <br>
      <input type="text" name="first_name" placeholder="First Name"><br>
      <input type="text" name="last_name" placeholder="Last Name"><br>
      <input type="text" name="username" placeholder="Username"><br>
      <input type="email" name="email" placeholder="Email"><br>
      <input type="password" name="password1" placeholder="Password"><br>
      <input type="password" name="password2" placeholder="Confirm Password"><br><br>
      <input type="submit"><br><br>
      <a href="login"><B>LOGIN</B></a>
      <div>
      {% for message in messages %}
      <h3>{{message}}</h3>
```

```
    {% endfor%}
  </div>
  </form>

  </center>
</body>
</html>
```

login.html

Here user can enter registered username and password to login into the website

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Login</title>
  <style>
    body{
      background-color: lavender ;
    }
    form{
      height:350px;
      width:500px;
      background-color:#8A4B69;
    }
    input{
      background-color:#D39FB7;
      margin:3px;
      font-size:20px;
    }
    a {
      margin:2px;
      color: black;
    }
  </style>
</head>
<body>
  <center>
    <br><br>
  <form action="login" method="post">
    <br>
    <h2>LOGIN</h2>
    {% csrf_token %}
    <br>
    <input type="text" name="username" placeholder="Username"><br>
    <input type="password" name="password" placeholder="Password"><br><br>
    <input type="submit"><br><br/>
    <a href="register"> <B> New User </B> </a>
    <div>
    {% for message in messages %}
    <h3>{{message}}</h3>
    {% endfor%}
  </div>
  </form>
```

images.html

This will present the images to the user based on the keywords present in the query

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <title>Images</title>
   <style>
   body {
   font-size: 10pt;
   font-family: arial, sans-serif;
   background-color: lavender ;
   }

a {
   text-decoration: none;
   }

   a:hover {
   text-decoration: underline;
   }

   button {
   font-weight: bold;
   font-family: arial;
   }


   /* Top Toolbar */

   .top-toolbar {
   height: 50px;
   float:right;
   margin: 7px 21px;
   }
   .top-toolbar nav a {
   margin: 3px 6px;
   color: #404040;
   }

   .top-toolbar nav a:hover {
   color: #111111;
   }

   .top-toolbar nav button {
   padding: 7px 12px;
   border-radius: 2px;
   background-color: #4585F1;
   color: white;
   border: 1px darkblue;
   font-size: 9.5pt;
   }

   .top-toolbar nav button:hover {
   box-shadow: 1px 1px 0 rgba(0,0,0,.5);
   }
```

```
    .top-toolbar nav img {
    margin: 0 7.5px;
    height:  25px;
    position: relative;
    top: 7px;}

    /* top left toolbar */

.top-toolbar-left {
height: 50px;
float:left;
margin: 7px 21px;
}

.top-toolbar-left nav a {
margin: 3px 6px;
color: #404040;
font-size: 18px;
}

    /* End of Top Toolbar */
    /* Search */

    .search {
    margin-left:200px;
    clear: both;
    }

    .logo {
    max-width: 21%;
    margin: 0 auto;
    }

.search input {
margin-top: 1%;
height:  25px;
width: 570px;
border-radius:20px;
border: 1px solid #D8D8D8;
padding: 5px;
font-size: 15pt;
/* background: url('images/microphone.png') no-repeat;
background-position: right;
background-size: 4.25%;*/
}

    .search-bar {
    max-width: 80%;
    }

    .search input:focus {
    outline: none;
    border: 1px solid #4285F4;
    }

    .search button {
    padding: 7px 8px;
    border-radius: 20px;
    border: 1px solid #D0D0D0;
```

```
    color: #444444;
    font-size:15px;
    height: 35px;
    width: 100px;
    }

    .search button:hover {
    box-shadow: 1px 1px 1px rgba(.2,.2,.2,.5);
    }

    /* End of Search */
    /*Images*/
    .row::after{
        clear:both;
        display:table;
        }
    .column{
        float:left;
        padding:5px;}
    </style>
</head>
<body>
<div class="top-toolbar">
        <nav>
                <img src="" />
                {% if user.is_authenticated %}
                <h5>Hello,{{user.first_name}} {{user.last_name}}</h5>
                <a href="logout"><button>Logout</button></a>
                {% else %}
                <a href="register"><button class="signin" onclick="sign" >Sign In</button></a>
                <a href="login"><button class="login">Login</button></a>
                {% endif %}
        </nav>
</div>
<div class="top-toolbar-left">
        <nav>
            <a href="#">Video</a>
            <a href="images">Images</a>
        </nav>
</div>
<div class="search">
        <div class="logo">
                <img src="" />
        </div>
        <form method="POST" action="images">
            {% csrf_token %}
            <input class="search-bar" type="text" name="search" placeholder="     type here to search" />
            <a href="#"><button type=" submit ">Search</button></a>
            {% if search %}
                <h4> Showing results for {{ search }}</h4>
            {% endif %}
            {% if search1 %}
                <h7> Search instead for {{ search1 }}</h7>
            {% endif %}
        </form>
</div>
    <h3 style="padding-left: 200px; color: #646464;">Images</h3>
        <div>
        {% if imgres3 %}
            {% for result in imgres3 %}
```

```
        <div style="padding-left: 180px;" class="row">
        <div class="column">
            <a href="{{result.0}}">
                <img alt="image" src="{{result.1}}" height="170" width="200" >
            </a>
        </div>
        </div>
    {% endfor %}
    {% else %}
        <div style="padding-left: 200px; padding-right: 50px;">
            <h2 style="color:">Sorry! there are no results for {{search}}</h2>
            <h4 style="color:">Please check your spelling or try different keywords</h4>
        </div><br><br>
    {% endif %}

    {% if imgres1 %}
        {% for result in imgres1 %}
        <div style="padding-left: 180px;" class="row">
        <div class="column">
            <a href="{{result.0}}">
                <img src="{{result.1}}" height="170" width="200" >
            </a>
        </div>
        </div>
        {% endfor %}
    {% endif %}
    </div>
</body>
</html>
```

6.2 Implementation

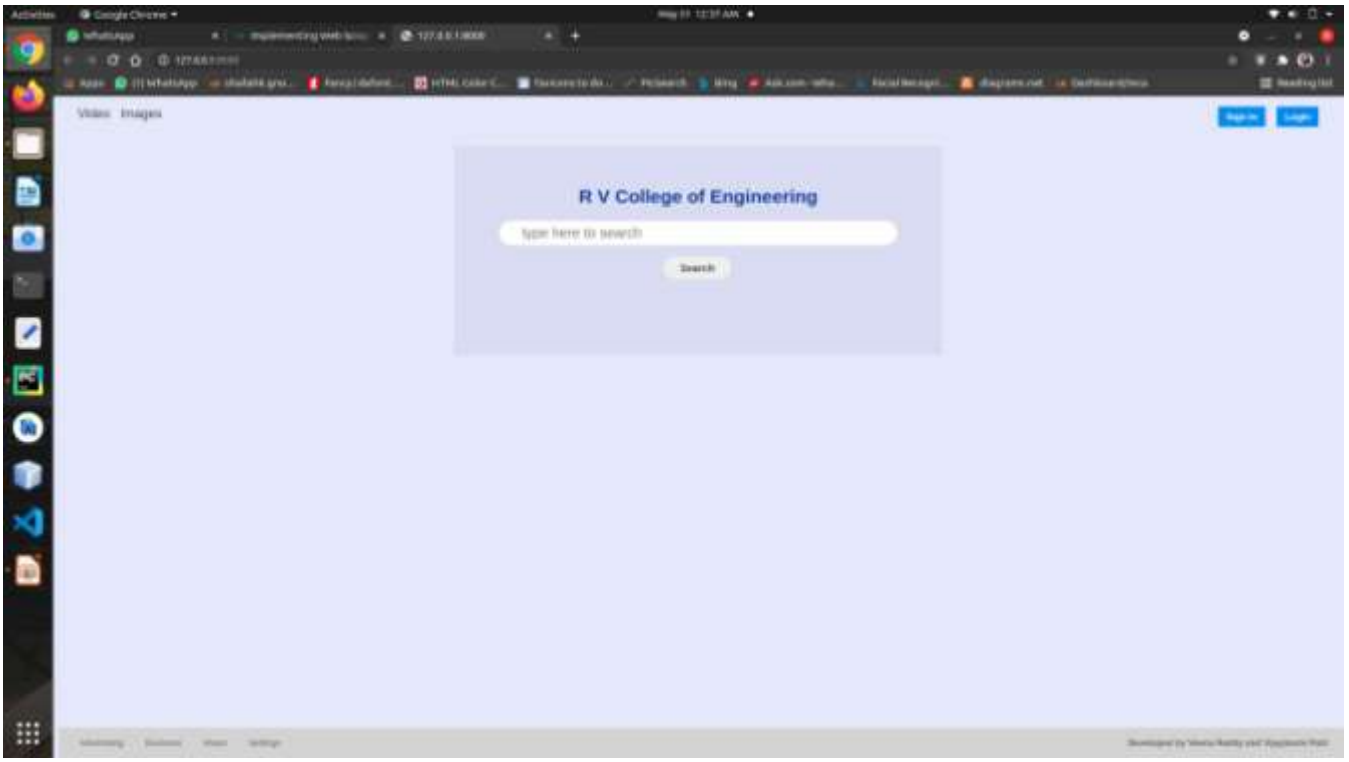


Fig 6.2.1 Home Page

Here user can write the query inside the search button and also can login, register or logout from the website

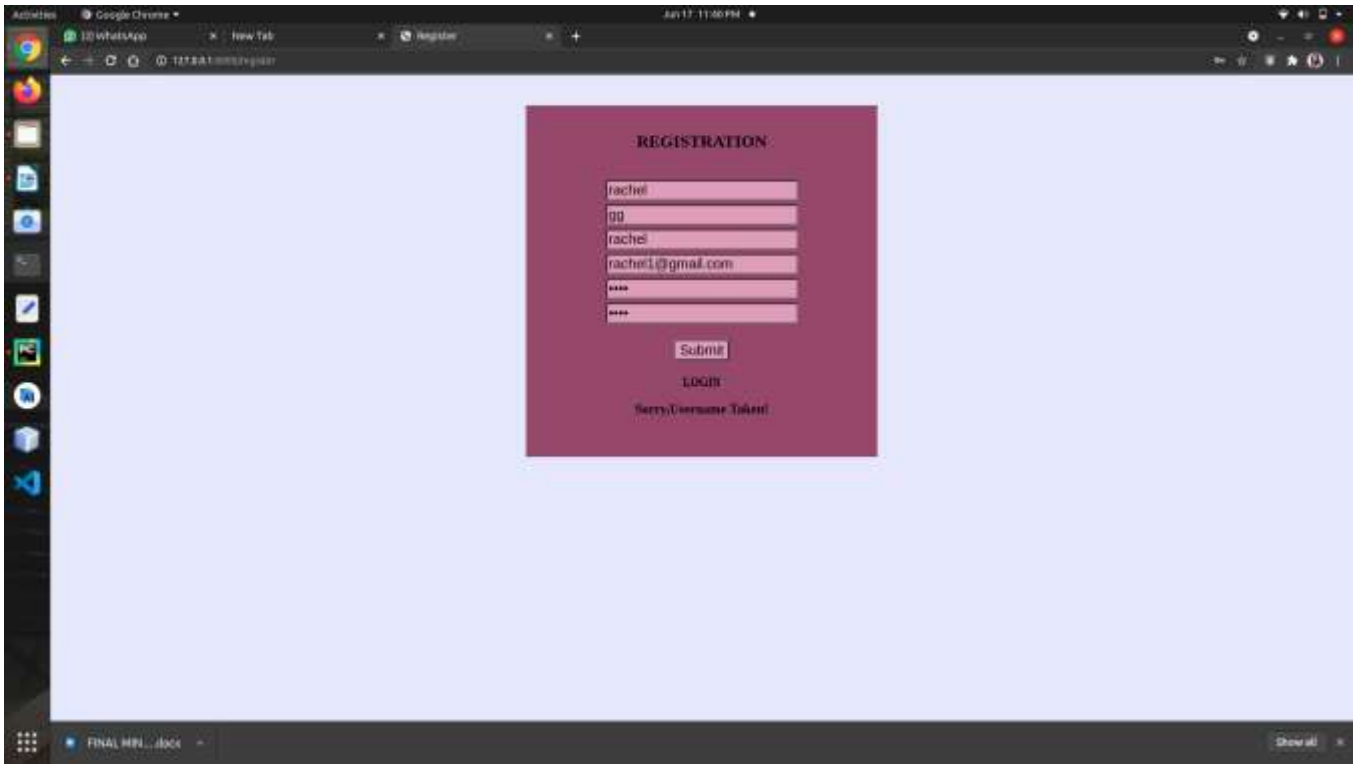


Fig 6.2.2 Registration with wrong username

Registration with wrong username

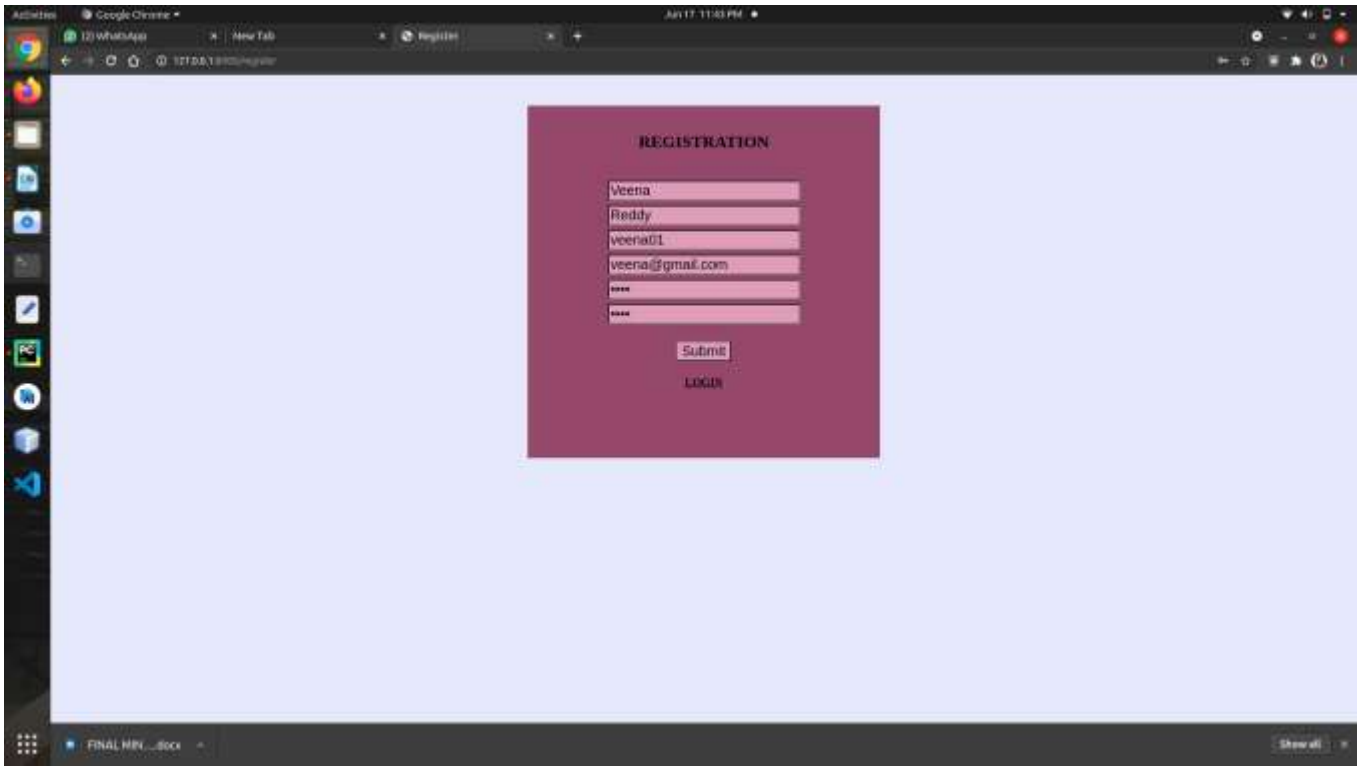


Fig 6.2.3 Registration Page

Here New user can register by entering all the correct credentials.

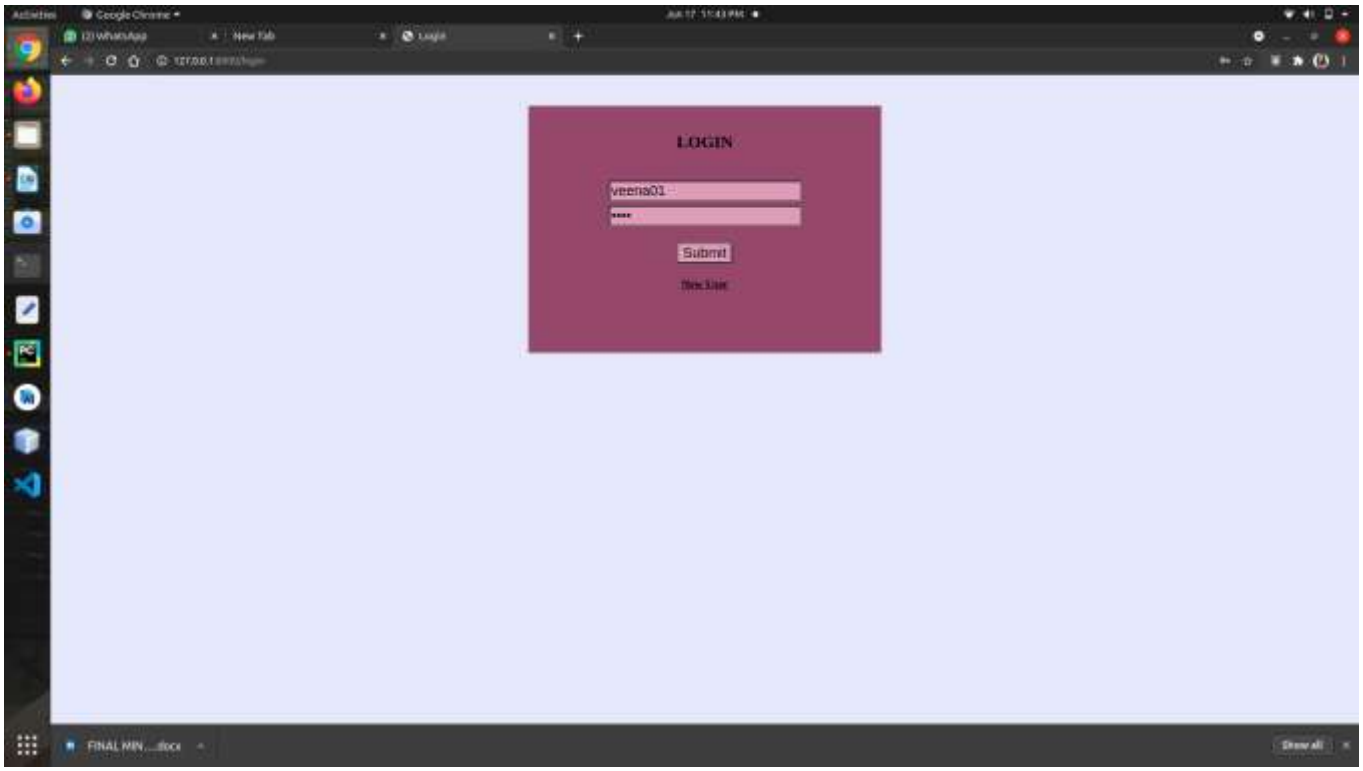


Fig 6.2.4 Login Page

Here user can enter registered username and password to login into the website

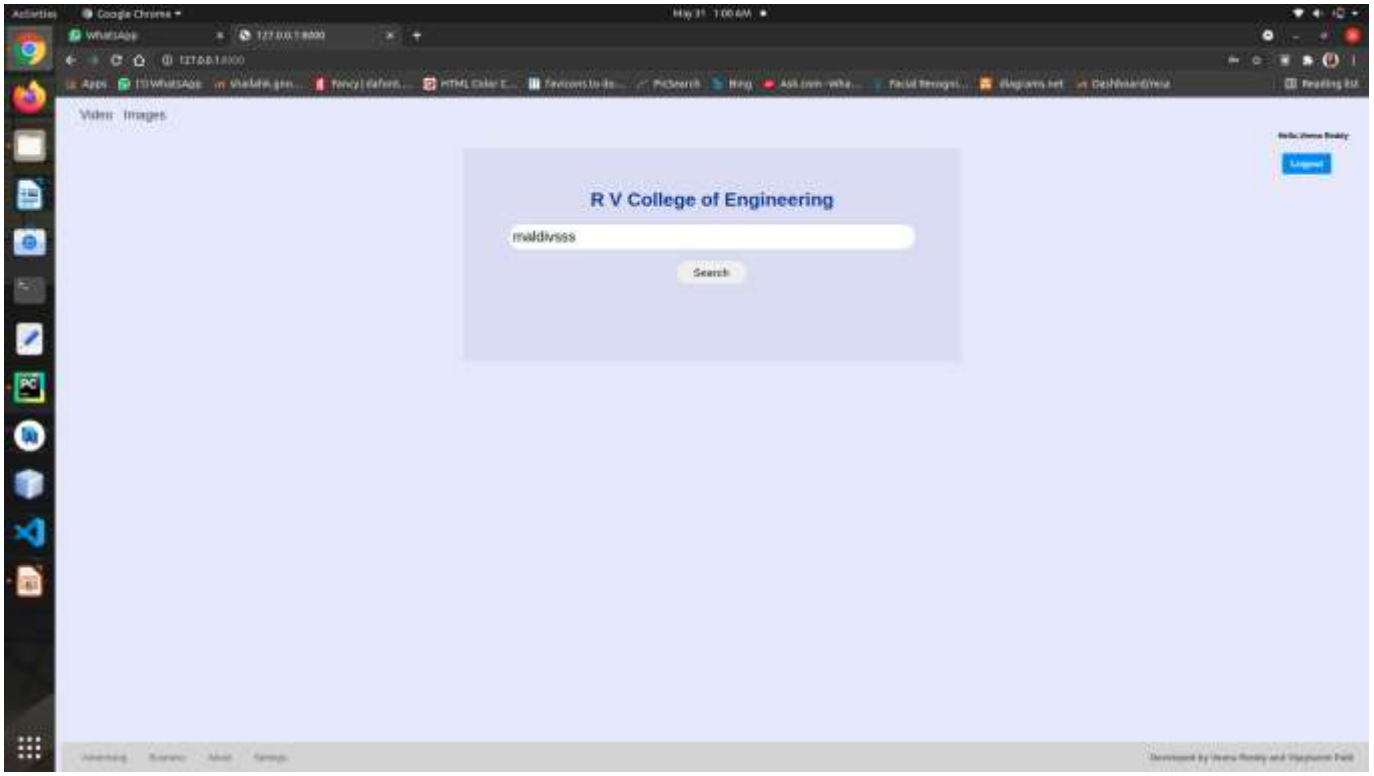


Fig 6.2.5 Search Page

This will fetch all the web links, images and related search links and present it to the user based on words

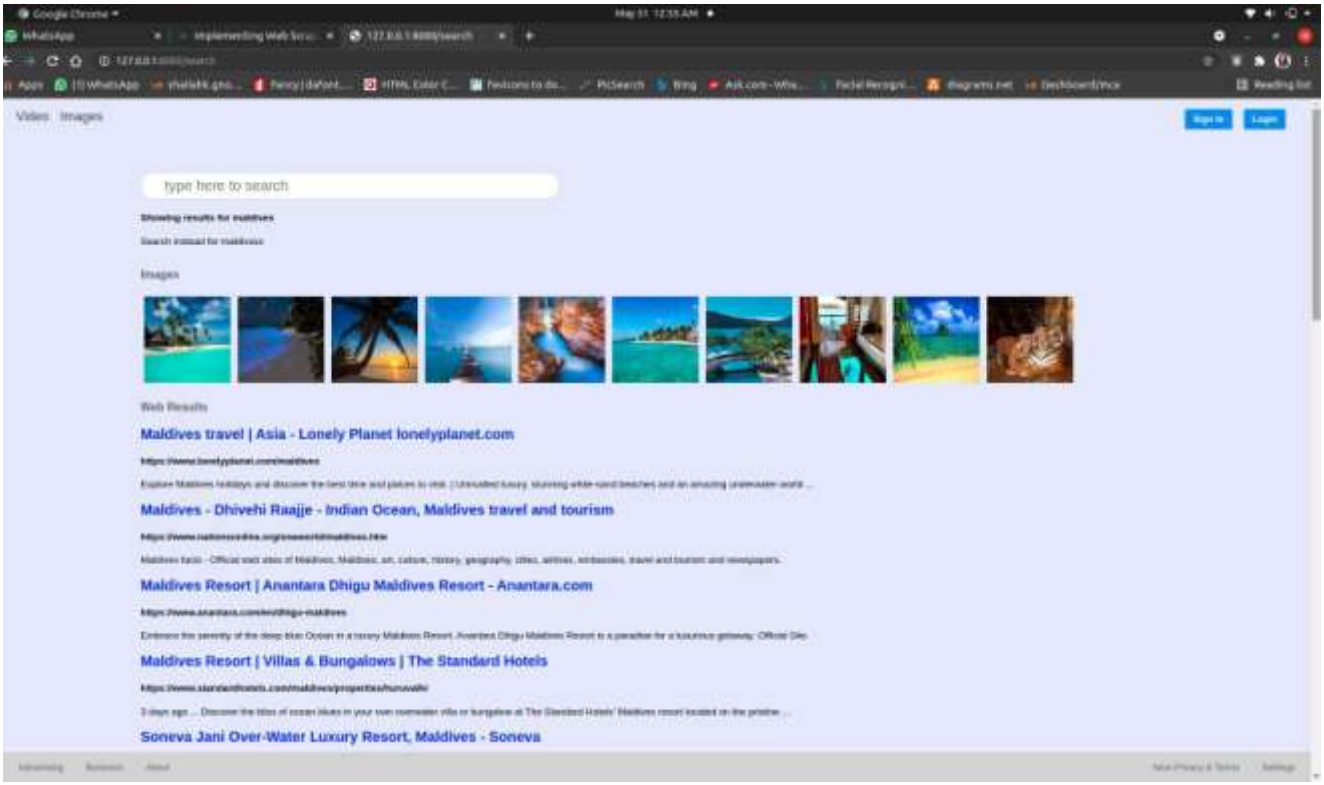


Fig 6.2.6 Search engine result pages and images

After entering a user query, the website fetches the data and results the search engine result pages, images and related links.

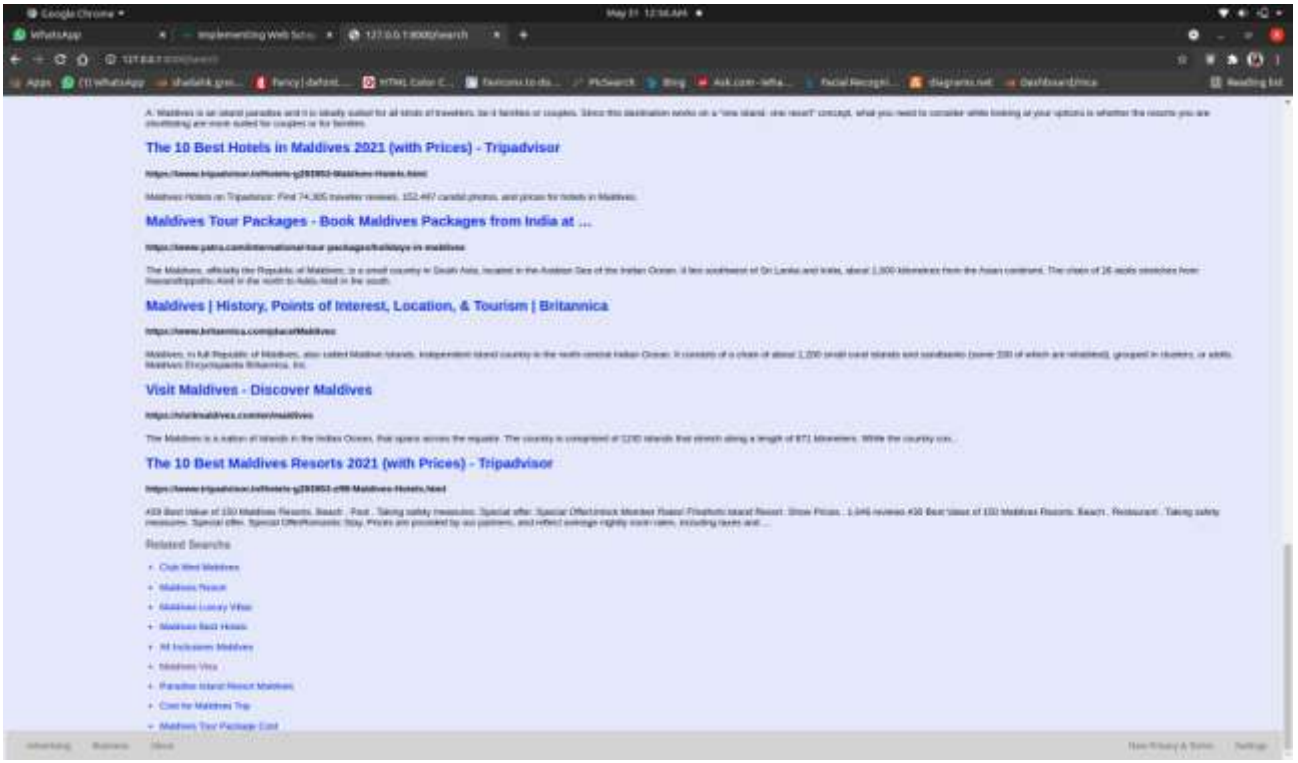


Fig 6.2.7 SERP, related search links

These are the search engine related pages, and related links searched by the user.

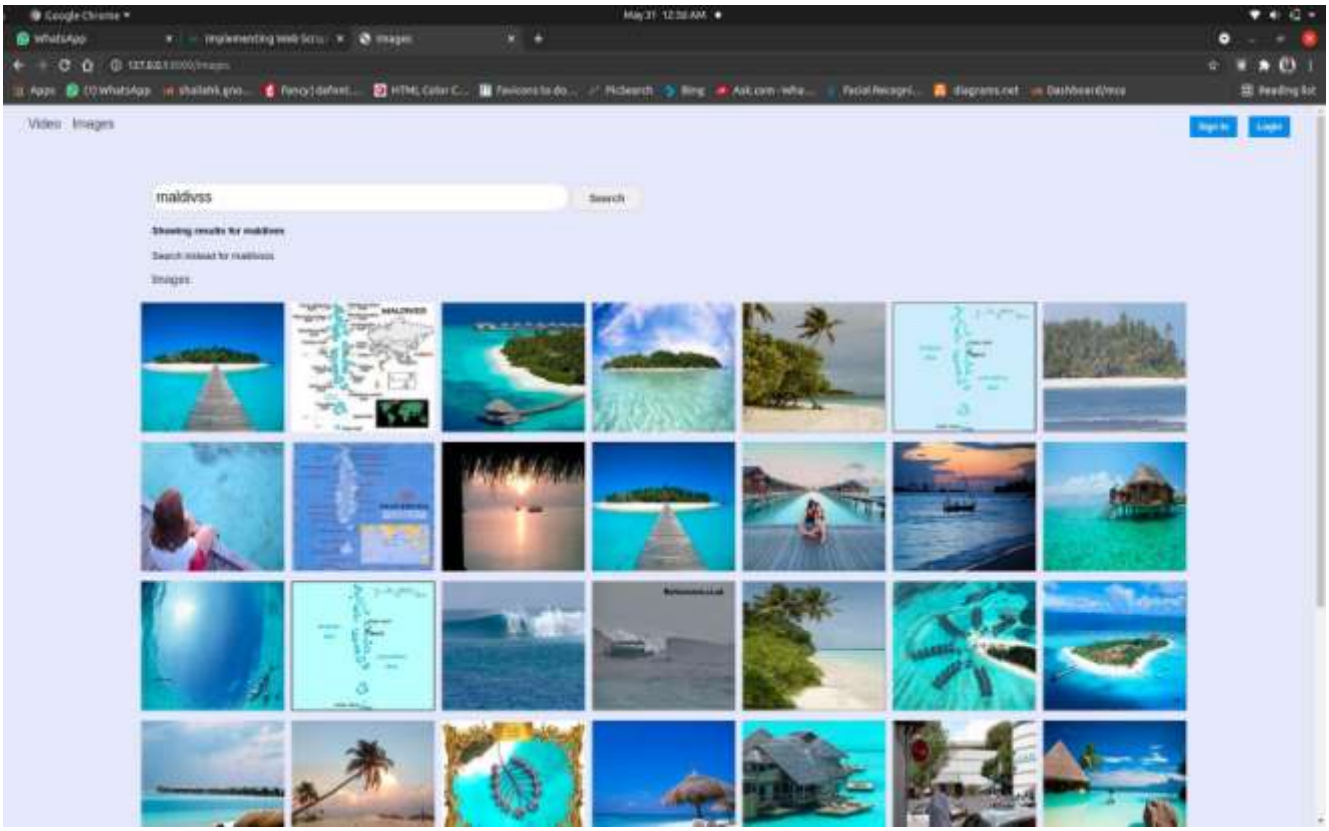


Fig 6.2.8 Result images

This will present the images to the user based on the keywords present in the query

Fig 6.2.9 spell check

If the user enters the query with wrong spellings, the website debug the spelling and shows the result fo the correct one.

Chapter 7: Software Testing

White Box Testing:

Here we are testing internal operations of system, it involves testing of software code for flow of specific inputs through the code.

Unit Testing: Here individual units or components of software are tested the purpose is to validate that each unit of software code performs as expected .It is done in development phase of an application by developers.

Test Case ID	Test Description	Expected result	Actual result	Remark
T01	Check SignUp and login button working .	Working fine	Working fine	Pass.
T02	Check for unique username , emailid and password & confirm password are same	Registration Successful	Registration Unsuccessful	Fail
T03	Check for unique username , emailid and password & confirm password are same	Registration Successful	Registration Successful	Pass
T04	Check username and password if they are not matching	Login Unsuccessful	Login Successful	Fail
T05	Check username and password are matching	Login Successful	Login Successful	Pass
T06	Search button working.	Working fine	Working fine	Pass.
T07	Check whether user able to input text.	Able to input text	Able to input text	Pass.
T08	User query.	Web search engine result page and Related searches	Search engine result pages and Related seaches	Pass.

T09	User query for Images	Images.	Images.	Pass.
T10	Check whether related information provided or not.	Related information provided	Related information is not provided	Fail.
T11	Check whether related information provided or not.	Related information provided	Related information provided	Pass.
T12	If query is misspelled	Correct the query	It doesn't correct the query	Fail.
T13	If query is misspelled	Correct the query	Correct the query	Pass.

T14	Images button.	Displaying images.	Displaying images.	Pass.
T15	After login.	User name and logout button should display.	Username and logout button is displaying.	Pass.
T16	Admin login.	Only admin can login.	Only admin can login.	Pass.
T17	Admin functionally.	Able to insert, delete and update a user information	Able to insert, delete and update a user information.	Pass.

Integration Testing: Here individual units or components of the application's source codes are combined and tested as a group. The purpose is to expose errors in the interactions of different interfaces with one another. It is done after unit testing.

Test Case ID	Test Description	Execute result	Actual result	Remark
IT01	Web page opening	Opened	Opened	Pass.
IT02	After Successful Registration	Open login page	It doesn't open the Login Page	Fail.
IT03	After Successful Registration	Open login page	Open login page	Pass.
IT04	After Successful login	Open Home page	It doesn't opens the home page	Fail.
IT05	After Successful login	Open Home page	Opens the Home page	Pass.
IT06	Only after login logout button should work	Working fine	Working fine	Pass.

IT07	Check the elements are aligned properly	Aligned properly	Aligned properly	Pass.
IT08	After searching a query	Render search page and display relevant results	Render search page and display relevant results	Pass.
IT09	If query is misspelled	Correct the query and present relevant results	It doesn't correct the query andpresent relevant results	Fail.
IT10	If query is misspelled	Correct the query and present relevant results	Correct the query and present relevant results	Pass.
IT11	Admin login successful	Open Django administration page	Open Django administration page	Pass.
IT12	User query for Images	Open images page and display relevant images	Open images page and display relevant images	Pass

System Testing :

Test Case ID	Test Description	Execute result	Actual result	Remark
ST01	Verify that the response fetched for a keyword is correct	Relevant results are fetched	Relevant results are not fetched	Fail.
ST02	Verify that the response fetched for a keyword is correct	Relevant results are fetched	Relevant results are fetched	Pass.
ST03	Verify that response for multi word keywords is correct	Relevant results are displayed	Relevant results are displayed	Pass.
ST04	Verify whether the link of search engine result pages are working	Working properly	Not working	Fail.
ST05	Verify whether the link of search engine result pages are working	Working properly	Working properly	Pass.
ST06	If incorrect keywords which do not have relevant results are provided then display.	Sorry! Thereare no resultsfor keyword. Please checkyour spellingor try different keywords.	Sorry! Thereare no resultsfor keyword. Please checkyour spellingor try different keywords.	Pass.

Chapter – 8 : Conclusion

Today's internet users are very positive about what search engines already do, and they feel good about their experiences when searching the internet. They say they are comfortable and confident as searchers and are satisfied with the results they find. They trust search engines to be fair and unbiased in returning results. The idea behind choosing this project was to know exactly how search engine works and how user gets the relevant information and of all the type.

This project determines how search engine works through web scraping. Here basically user enters the query or the keyword, depending upon the keyword the search engine extract the data and provides user in getting the relevant information and also it displays the information in the form of images, Search Engine Result Pages(SERP) and related links. The main focus here is to show the web scraping technology which helps to solve the real world problems.

Chapter - 9

Future Enhancement

In Future, it will seem routine to be able to search the contents of vast libraries of books, to find selected portions of video streams or audio recordings, to benefit from personalized searches that remember a user's preferences and keep track of changing geographical locations. Audio searching and search results will be available for the blind; "implicit searching" will anticipate users' queries and have answers ready.

Bibliography

[1] D.Pratibha, Abhay.M.S., Akhil Dua, Giridhar K.Shanbhag, Neel Bhandari, Utkarsh Singh, "Web Scraping And Data Acquisition Using Google Scholar", 2018 3rd International Conference on Computational Systems and Information Technology for Sustainable Solutions (CSITSS).

[2] ERDINÇ UZUNA, "Novel Web Scraping using the additional information obtained from web pages" IEEE ACCESS, Date of publication March 31, 2020, Department of Computer Engineering, Çorlu Faculty of Engineering, Tekirdag Namik Kemal University, 59860 Tekirdag, Turkey.

[3] Akash Kumar Singh, "Scraping the web data from google using python", Geeks for Geeks, 27 May 2020, 5th Floor, A-118,Sector-136, Noida, Uttar Pradesh – 201305.

[4] DI Dr. Gerd Holweg ,"Web scraping data extraction from websites", Published on 04.02.2018 Bachelor of Science in Engineering at the University of Applied Sciences Technikum Wien – Degree Program Business Informatics.

[5] Vidhi Singrodia, Anirban Mitra, Subrata Paul , "A Review on Web Scrapping and its Application" 2019 International Conference on Computer Communication and Informatics (ICCCI) Coimbatore, India.

[6] Yi Jin, Zhuying Lin, Hongwei Lin, The Research of Search Engine Based on Semantic Web, 21-22 Dec. 2008, Published in: 2008 International Symposium on Intelligent Information Technology Application Workshops, Shanghai, China.

[7] Meng Cui, Songyun Hu, "Search Engine Optimization Research for Website Promotion", 24-25 Sept. 2011, Published in: 2011 International Conference of Information Technology, Computer Engineering and Management Sciences, Nanjing, China.

[8] Farzaneh Shoele, Mohammad Sadegh Zahedi, Mojgan Farhoodi, "Search engine pictures: Empirical analysis of a web search engine", 19-20 April 2017, Published in: 2017 3th International Conference on Web Research (ICWR), Tehran, Iran.

[9] Dushyant Sharma, Rishabh Shukla, Anil Kumar Giri, Sumit Kumar, "A Brief Review on Search Engine Optimization", 10-11 Jan. 2019, Published in: 2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence), Noida, India.

[10] Zhou Hui, Qin Shigang, Liu Jinhua, Chen Jianl, "Study on Website Search Engine Optimization", 11-13 Aug. 2012, Published in: 2012 International Conference on Computer Science and Service System, Nanjing, China.

[11] Payal A. Jadhav, Prashant N. Chatur, Kishor P. Wagh, "Integrating performance of web search engine", 27-28 Feb. 2016, Published in: 2016 2nd International Conference on Advances in Electrical, Electronics, Information, Communication and Bio-Informatics (AEEICB), Chennai, India.

[12] Željko Knok, Mark Marčec, "Universtiy search engine", 30 May-3 June 2016, Published in: 2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, Croatia.

[13] Petrović, Ilja Stanišević Dorde ,"Web scrapping and storing data in a database", 2017 25th Telecommunication Forum (TELFOR).

[14] Sneha Nain, Bhumika Lall, "Web Data Scraper Tools: Survey", 31 May 2014, 3rd International Conference on Computational Systems and Information Technology for Sustainable Solutions (CSITSS).

[15] Erdinç Uzun, "A Novel Web Scraping Approach Using the Additional Information Obtained From Web Pages", IEEE Access (Volume: 8)

[16] Zhang, "Research a New Method of Search Engine Optimization" 2010 International Conference on Internet Technology and Applications.

[17] DI Dr. Gerd Holweg , “Web scraping data extraction from websites”, Published on 04.02.2018 Bachelor of Science in Engineering at the University of Applied Sciences Technikum Wien – Degree Program Business Informatics..

[18] Sadegh Kharazmi, Tehran, Iran, Ali Farahmand Nejad; Hassan Abolhassani, “Improving performance of Web search engines”, 9-12 Nov. 2009, Published in: 2009 International Conference for Internet Technology and Secured Transactions, (ICITST), London, UK.

[19] R. Aravindhan; R. Shanmugalakshmi, “ Comparative analysis of Web search engines”, 19-21 Dec. 2013, Published in: 2013 International Conference on Advanced Computing and Communication Systems, Coimbatore, India.

[20] Huanwei Wu, “ Search Engine Optimization”, 12-14 Aug. 2011, Published in: 2011 International Conference on Management and Service Science, Wuhan, China.