VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

School of computer science and Engineering

CSE-3001 Software Engineering

Personal Automation System

By-

Shelly Mohanty (19BCE0820)

Varsha S (19BCE2362)

Winter-Semester, 2020-2021

Under the guidance of

Prof. Meenakshi SP

CONTENTS

Page No

1. Problem Statement

Like many other methodological innovations, Automation has grown over the years with technological advancements, however very few people take advantage of the libraries and methods available to automate their daily tasks. This paper explores the avenue of automation along with scraping data from the internet.

Our project aims to encourage and explore the world of automation through the use of two major programming languages and interfaces – Python and Selenium. It explores the most popular libraries implemented by python to perform automation and file manipulation in recent years which include the OS, XML, urlparse and getpass libraries. It also explores selenium which is a free (open-source) automated testing framework used to validate web applications across different browsers and platforms. It intends to create a tool-set driven by automation and web scraping.

We intend to implement and provide a tool set to users with the purpose of saving multiple webpages, documents and means to access files offline without the need to individually go and save each and every required page manually. The tool set also includes scripts that can be used to scrape weather data, top daily news as well as open multiple links through a simple script. This documentation will also cover our workflow methodologies, explore previous work in the field, provide a Software requirement specification documentation along with the design of the proposed system.

The objectives of the implementation are as follows:

- Implementation of a multiple-link opener that automates opening links one-by-one on a browser.
- Scraping the top news headlines from the google news webpage
- Scraping weather information from Open Weather Map
- Automate form fill-ups and logins through Facebook login automation
- Automatically download the desired number of images corresponding to the keyword and number of images required entered by the user.
- Provide a working interface to the user that works on menu-based selection with proper user sanitized inputs.

Work for most people nowadays involves using the internet on a daily basis, however people sometimes need offline web pages and images. They may require to do trivial tasks repeatedly such as open multiple links, download multiple images and so on. They may not have the time required to manually perform multiple searches and operations over the internet. We intend to provide a toolset that can help such people and other students that need to scrape data off the internet and automate their work. This can help improve their efficiency along with lessening their burden. It can also be used by tech enthusiasts to explore the upcoming field of automation through programming.

2. Project Planning Chart

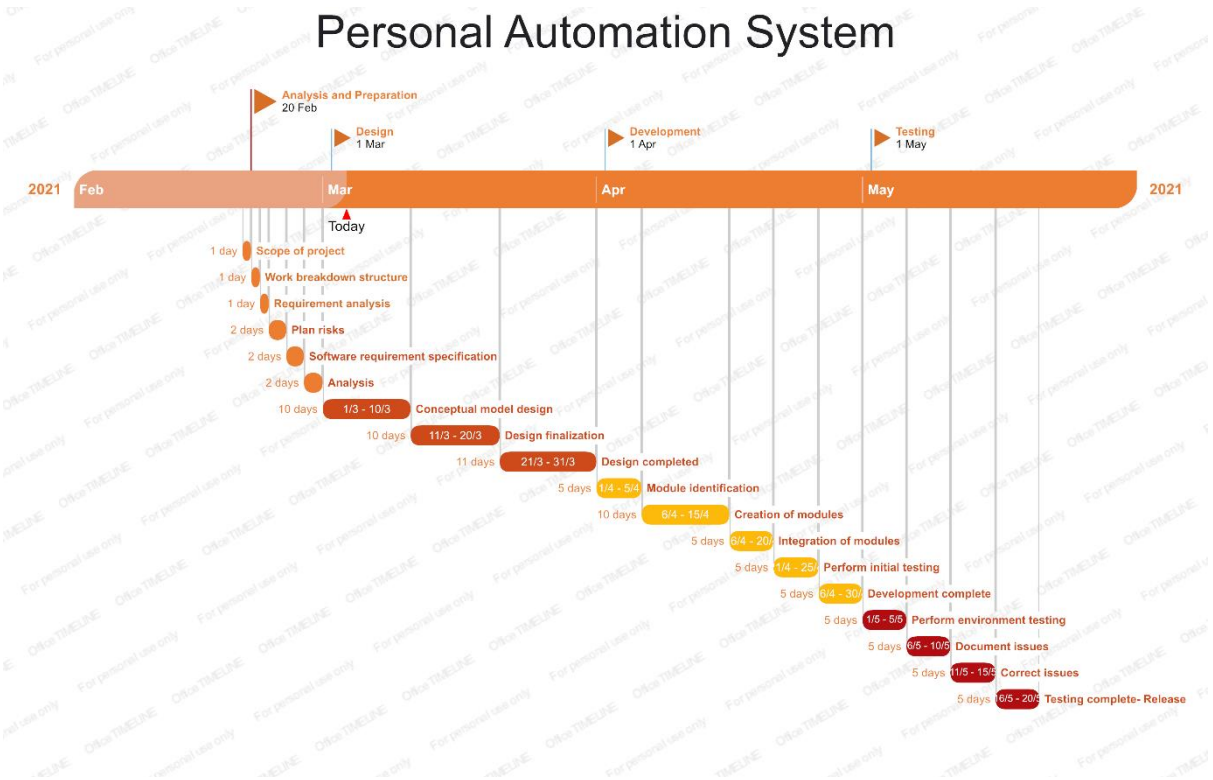


Fig: Timeline Chart

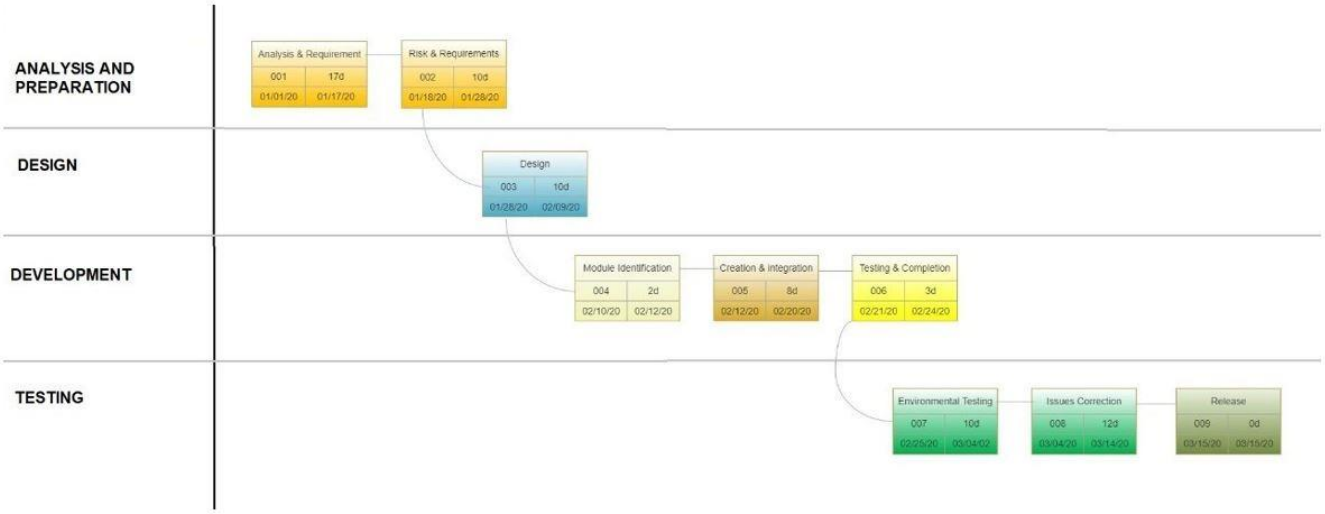


Fig: Pert Chart

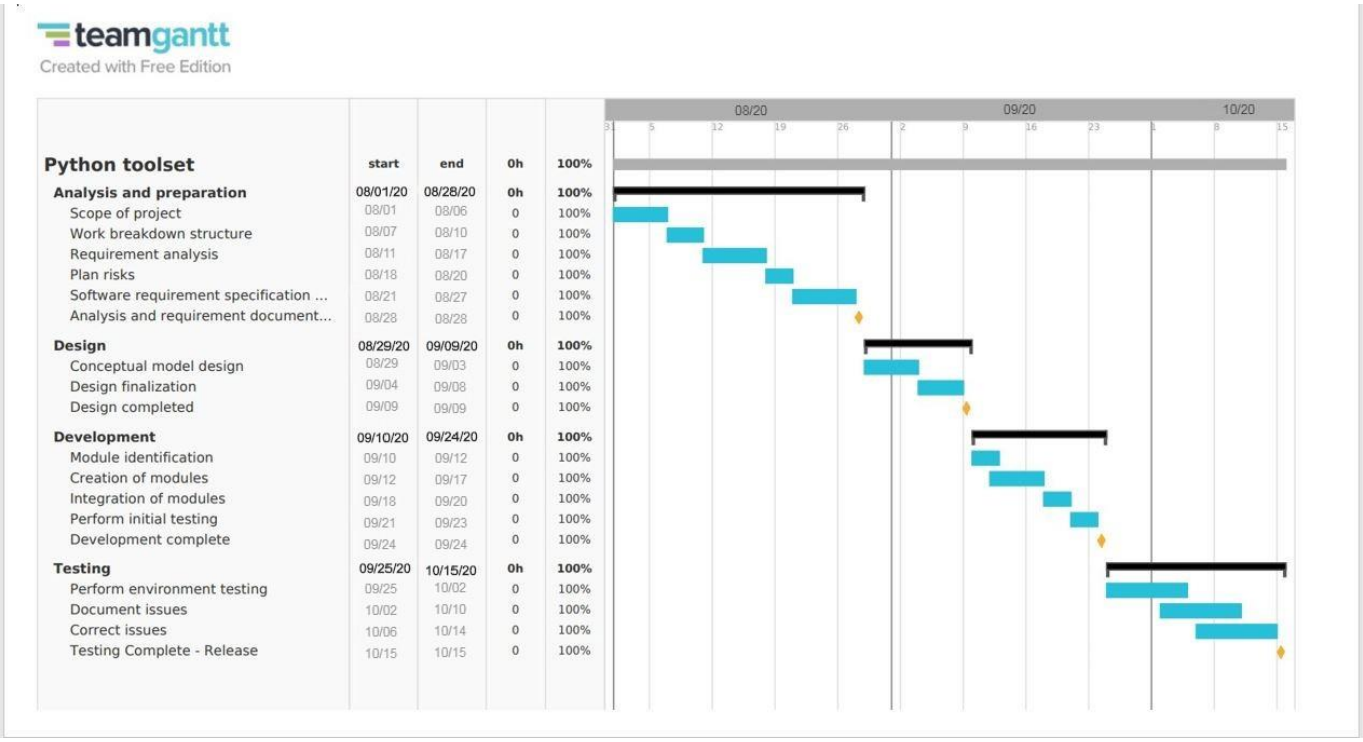


Fig: Gantt Chart

3. Proposed System Requirements Analysis

1.Introduction

1.1 Purpose

The purpose of this document is to build a set of scrapping tools that automate user functions such as filling forms, open multiple tabs with respective user input for search results, download multiple images from a website etc.

1.2 Document Conventions

The document uses the following convention

Sel – Selenium Module

SCB– Selenium Controlled Browser

1.3 Intended Audience and Reading suggestions

These set of tools can be used by any user to automate their daily tasks that require them multiple tab access and downloading / extracting data from the internet on a daily basis for professional or personal needs. It's useful for web programmers as well as graphic designers.

1.4 Project Scope

The programs can be used by any user to automate their daily tasks that require them multiple tab access and downloading / extracting data from the internet on a daily basis for professional or personal needs. The programs are coded in python or through Sel which is accessed through the Firefox browser. The main code can be changed to accommodate any form of data or website the user needs to access. It provides a much faster method to do tasks that would take up huge amounts of time to do manually. It can also do functions that would be too hectic to proceed with manually.

1.5 References

http://automatetheboringstuff.com/
https://hackernoon.com/building-a-web-scraper-from-start-to-finish-bb6b9538818

2. Overall Description

2.1 Product Perspective and features.

The system after running the program would perform the following features

- Store required number of images in a folder named after the keyword required.
- Provide weather information as per user input
- Display the top news on google news feed
- Automate user login in Facebook
- Open multiple links provided by the user (separated by space) on a new browser.

Filling out forms automatically.
The browser can automatically locate login fields and then login as a user as soon as the user loads the respective page.

Automated clicking and submitting input
The browser will automatically perform keystrokes depending on the code of the user. It will be achieved through Selenium.

2.2 User class and characteristics.

The main program will consist of only one major user class which would be the person who runs the tools for automation. They have the ability to access the main code and change the URLs as well as the given values for adjusting the changes they need to be made for their next scraping process.
The customer should be able to do the following functions:

Python functions

1. Crawl websites, extract data and store it offline if required.
2. Use weather APIs to scrape weather information
3. Provide a user-controlled menu interface

SCB functions

1. Automate text box and form fill-ups in websites
2. Open a new instance of a browser (or new tab if pre-opened)
3. Mimic user keystrokes and interactions
4. Navigate through the browser interface (tabs)

2.3 Operating Environment

Frontend: Command line interface (CLI)
Operating system: Windows
Browser: Firefox
External modules: Selenium
Scripting language: Python

2.4 Design and implementation constraints

The response procedure from downloading data such as images and plain text, their storage location, the format of the file saved and their grouping.

The browser-based scripting limits the use of an external front end and back end system as most of the functionalities involve opening other websites and loading browser modules which would be ineffective to be merged with a full stack system.

2.5 Assumption Dependencies

Let us assume the tool is being used for websites:

- The websites accessed for scraping should be a general one that does not consist of millions of sub sections and subdomains which would result in a huge overhead
- The scraping is done on a website that allows such a procedure to follow. Many websites do not permit scraping data through many guidelines and rules.
- The automation is done on websites that do not run bot checkers and crawlers. It would restrict the login functionality of the browser many times due to captcha checks and "I'm a robot" checks.

3. System features

3.1 Description and Priority

The program's high priority is properly downloading images and other data in high resolution without any skipping or skimming. This requires the code to be very efficient otherwise it will experience clogging due to huge traffic flow.

3.2 Stimulus/Response Sequences

- Open web pages according to the data in the search fields inputted.
- Download images of complete sites or crawling through websites
- Correctly input data in login fields of the data and automate keystrokes.

4. External Interface requirements

4.1 User Interfaces

- Python GUI for code execution

4.2 Hardware Interfaces

- Windows
- Browser which supports Sel like Firefox

4.3 Software Interfaces

Software	**Description**
Windows	It is the operating system we use as it is the most user friendly operating system with the highest number of compatible software.
BeautifulSoup (BS4)	It is a parsing library that can use different parsers. Advantage of BS4 is its ability to automatically detect encodings and navigate parsed documents to extract needed data.
Selenium module – Gecko	It's the most popular tool for automating browsers primarily for testing. It's one of the easiest testing and automation software to use with python.
Scrapy	An open source and collaborative framework for extracting the data you need from websites, We're using it as it is a fast, simple, yet extensible way to scrape data.
Firefox	Mozilla Firefox, or simply Firefox, is a free and open-source web browser developed by the Mozilla Foundation and its subsidiary, Mozilla Corporation. Firefox uses the Gecko layout engine to render web pages
URLparse, URLlib, requests, OS and re	A set of python libraries that manipulate HTTP requests and the system directories. Re works to identify patterns.

Specific versions used for the project:

Python: Version 3.7.7

Firefox: Firefox 77

Selenium: Version 3.141.59

4.4 Communication Interfaces

The SCB supports Firefox browser integration but the web scrapping functionalities and the combination of front-end and back-end of Electron supports all operating systems and browsers.

3.1.1 Functional Requirements

- Proper menu-based interface and proper tool-selection inputs.
- News scraper that extracts and presents the top 5 headlines, separated by user inputs
- Image scraper that creates a folder and stores images based on the keyword and number of images provided by the user.
- Weather scraper that extracts the weather conditions of the location specified by the user and presents it in a readable format.
- Multiple link opener that automates a user browser to open several links (separated by space) provided by the user.
- Automated Facebook login that opens a browser, logs in with the given credentials and presents the home page.

3.1.2 Non Functional Requirements

4.5 Performance Requirements

The program should not be clogged or experience bottleneck throughout the automation process. It could result in a deadlock and make the whole automated process obsolete.

4.6 Safety Requirements

The programs should in no way act as a suspicious bot on the web which can lead to the IP blocks. It should also not tamper with sensitive website data as it is a violation of webpage policies.

4.7 Security Requirements

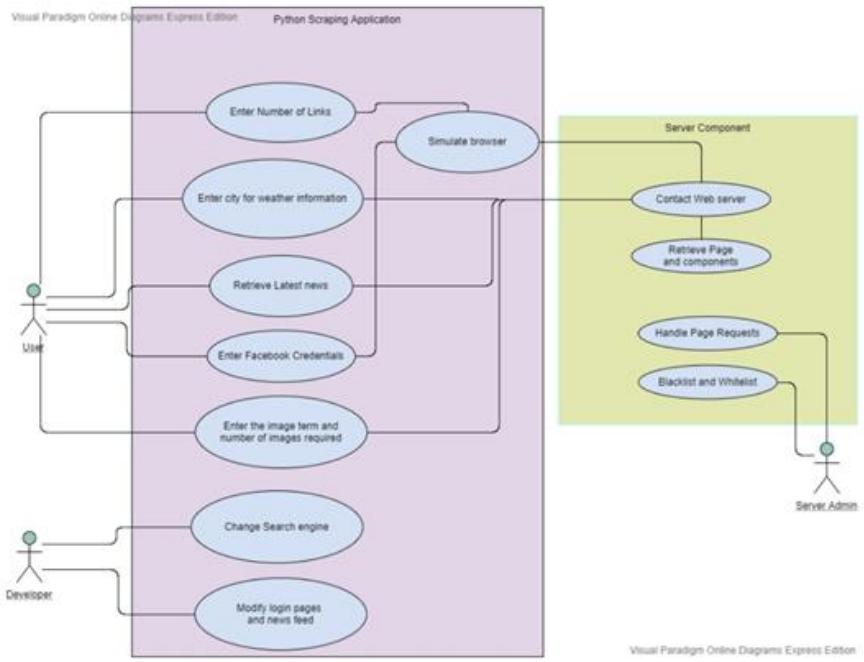
The data extracted from the website should not be visible to anyone else. There should be no form of personal information leakage or data leakage during the automated/scraping process.

4.8 Software Quality Attributes

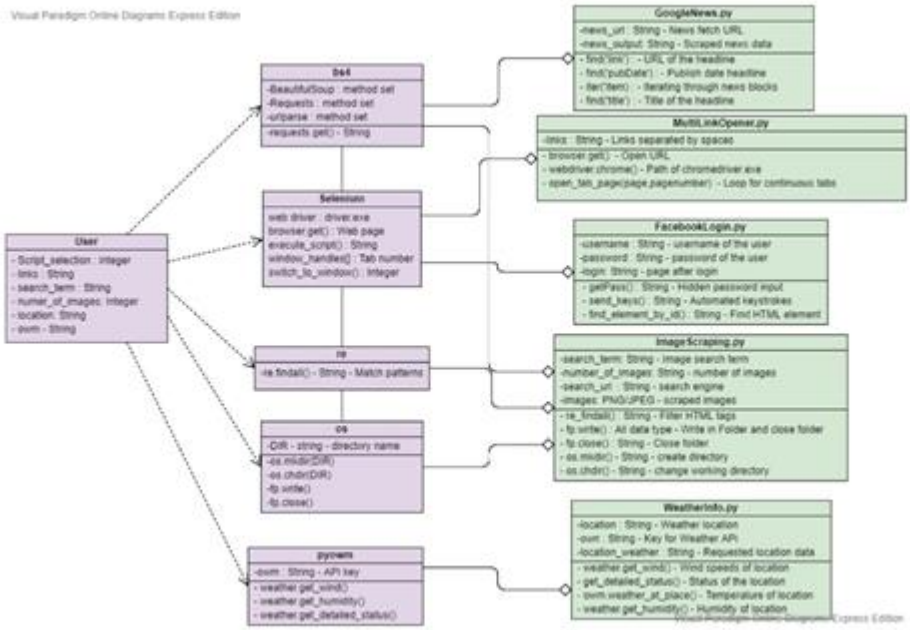
- Availability- The tools should run on most websites and be compatible with popular OS.
- Correctness- It should correctly scrap data relevant to the website.
- Maintainability- The scraper should use the latest and most optimised python libraries for minimum execution time.
- Usability: It should display proper logs in case of an error. It should also sanitize user input.

4. UML Diagrams

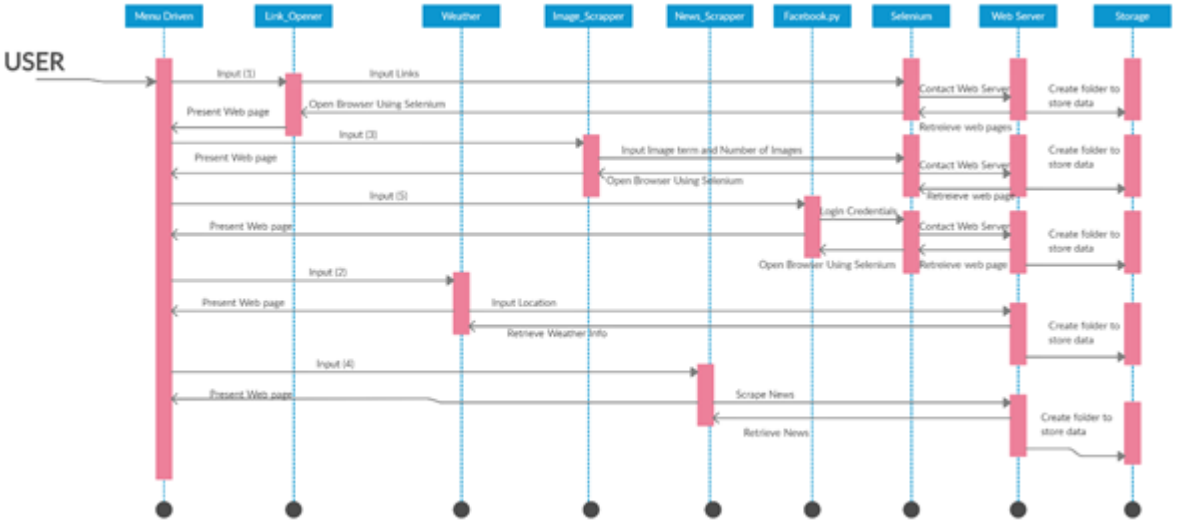
#1) USE-CASE DIAGRAM



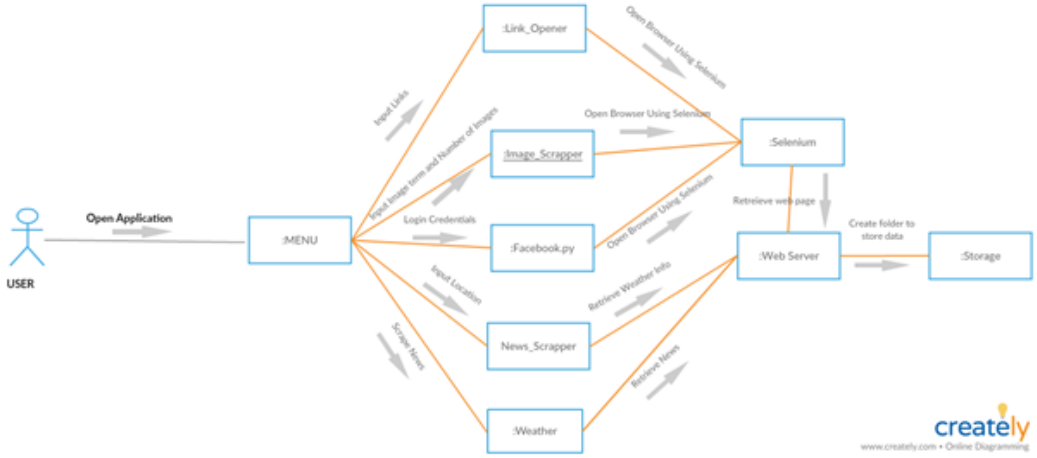
#2) CLASS DIAGRAM



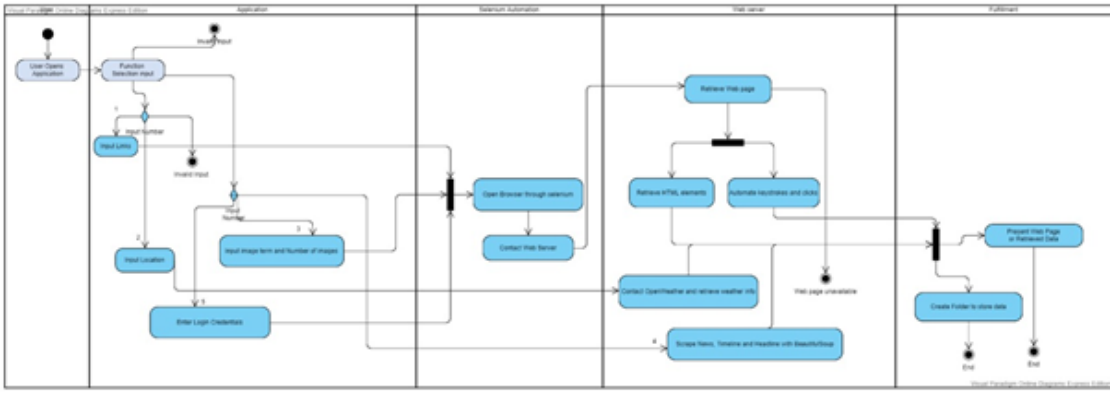
#3) SEQUENCE DIAGRAM



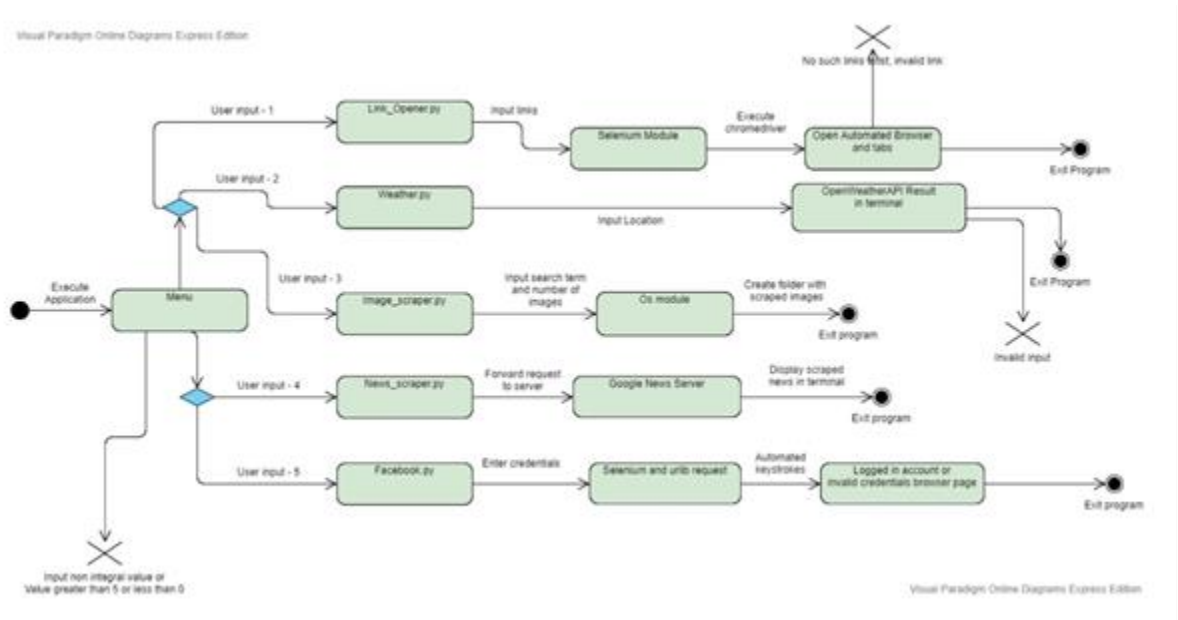
#4) COLLABORATION DIAGRAM



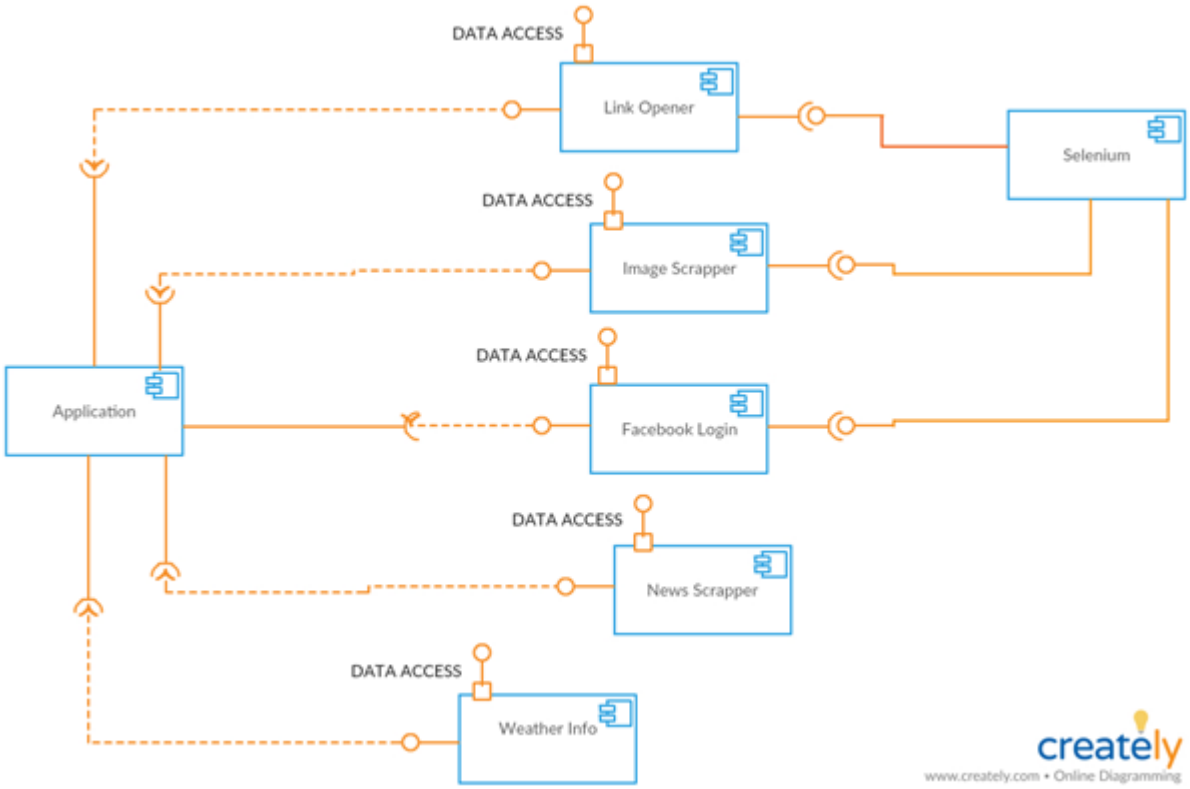
#5) ACTIVITY DIAGRAM



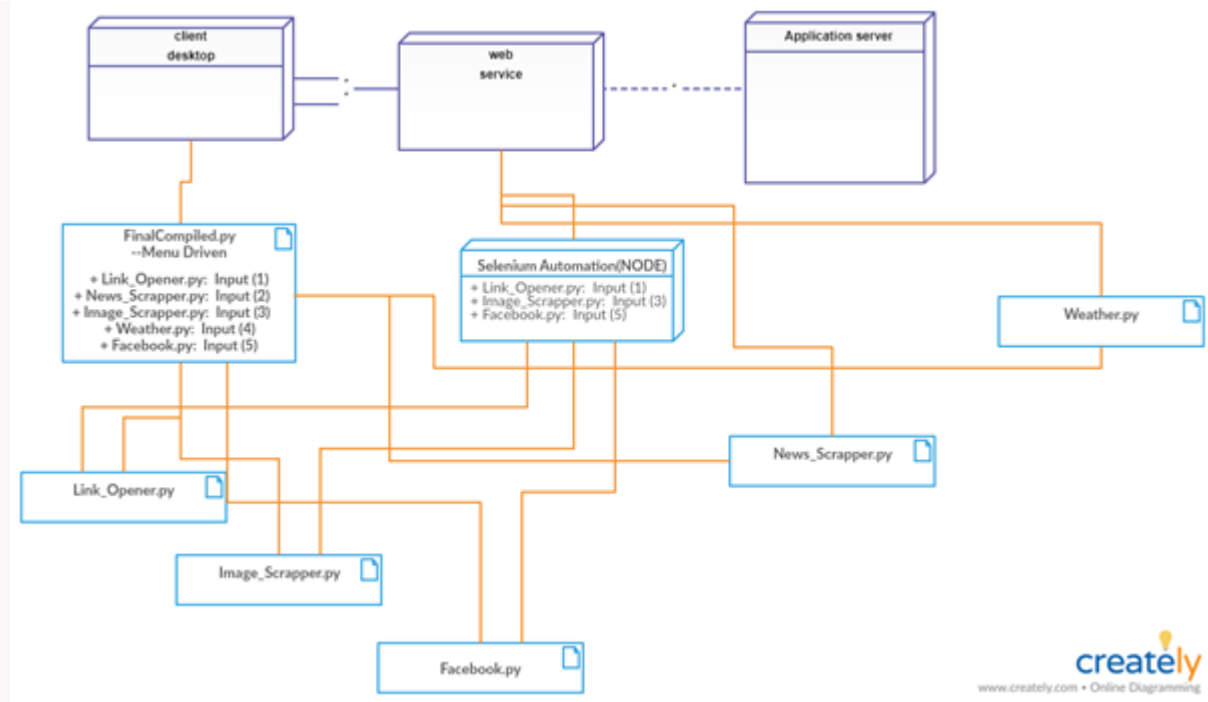
#6) STATE DIAGRAM



#7) COMPONENT DIAGRAM



#8) DEPLOYMENT DIAGRAM



5 DFD and ER

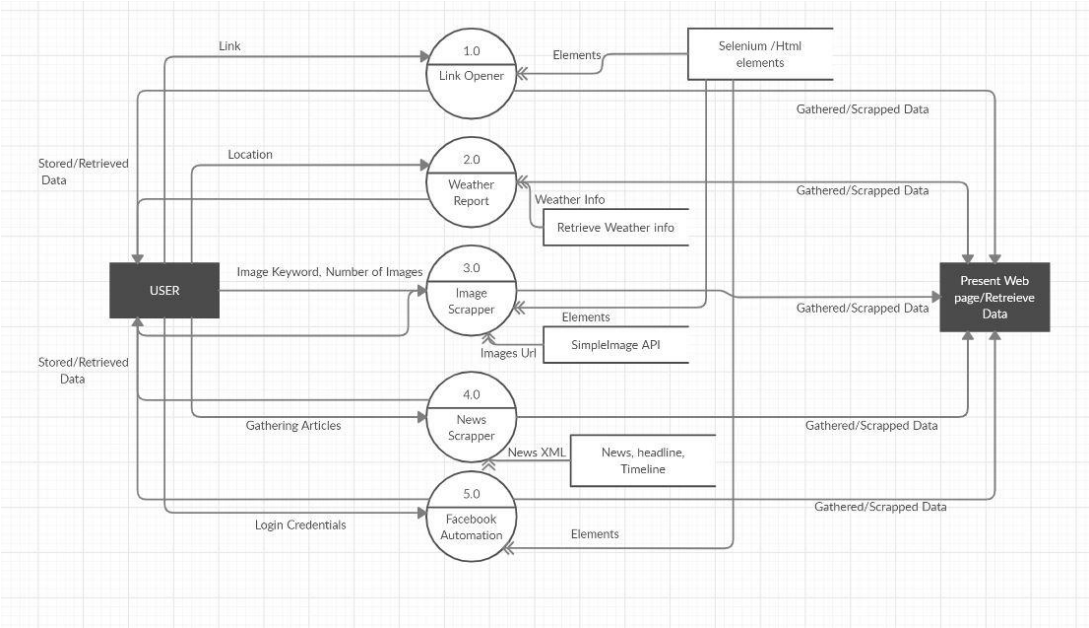


Fig: Level 2 DFD

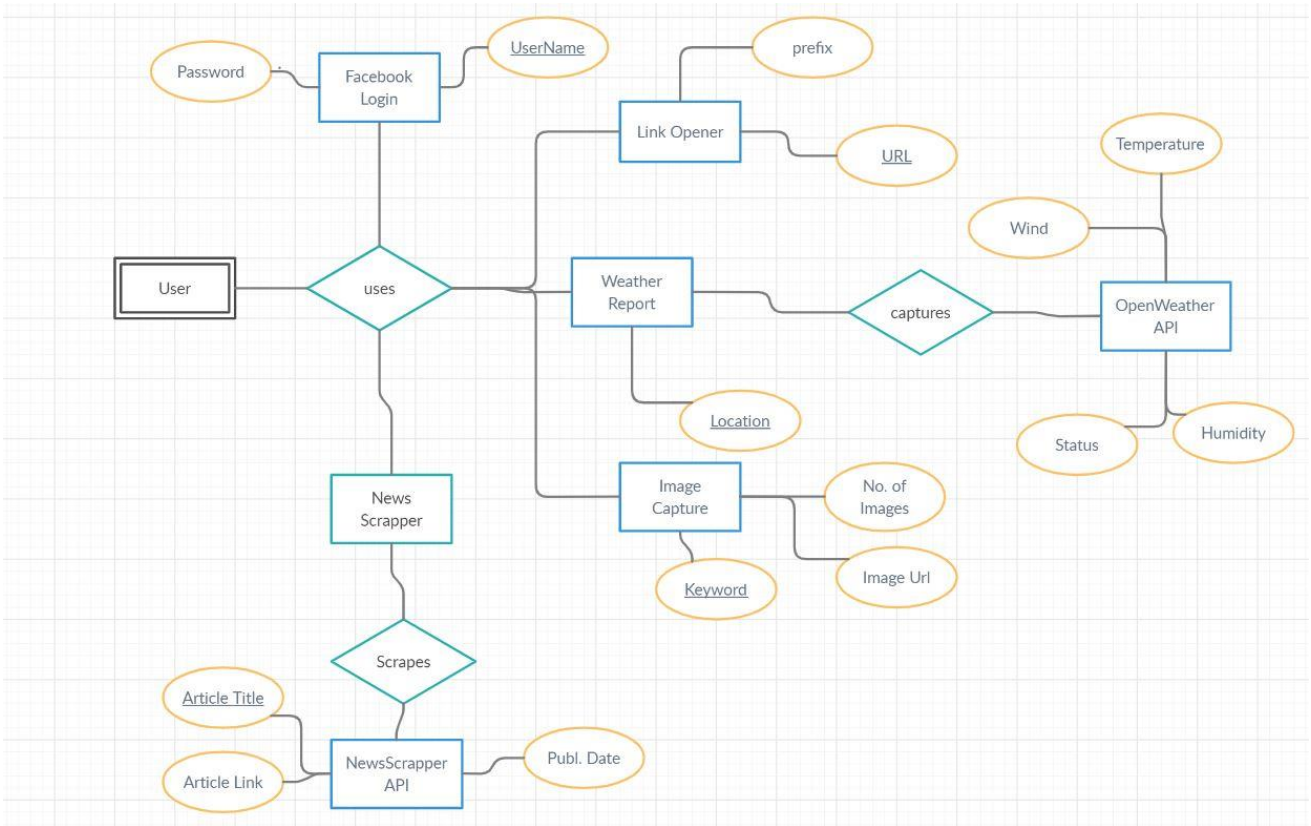
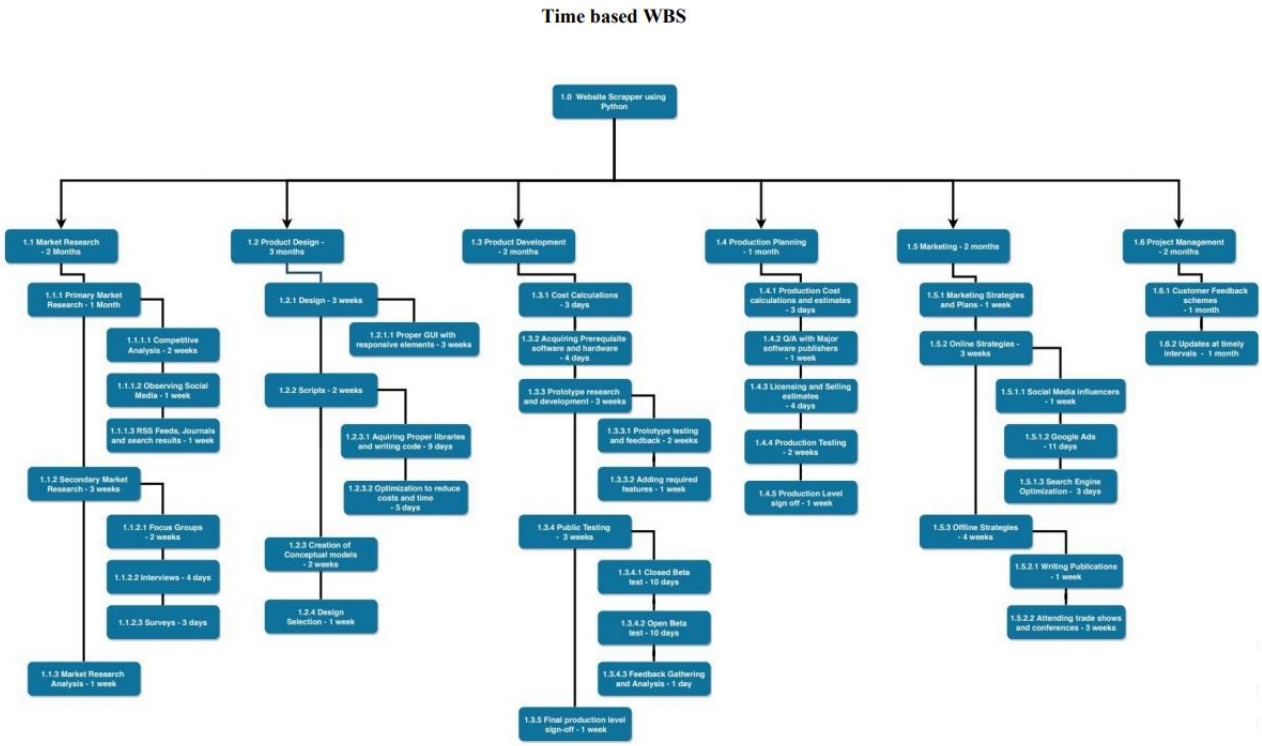


Fig: ER Diagram

6. Work Breakdown Structure



7. Algorithm and Architecture

This section elaborates the architecture used by the project. It will cover the high-level design along with the architectural pattern used in our project.

Architecture design

We will use the **Call and Return** architecture for our project. Our program involves a variety of functional calls and returns and we require a robust architecture that is easy to modify.

Reasons for **Call and return** architecture:

1. Has the main goal of modifiability and scalability
2. It has been a very dominant architecture since the start of software development.
3. The architecture is based on well-identified parts of a task (our tool functions) and can allow change of routines without affecting any part of the client side.
4. It allows for efficient reuse of individual operations and functions among different components and modules of a system.

This architecture is further classified into various other architectures such as:

1. Layered
2. Main program and subroutine
3. Remote procedural Call

Our project focuses on **Main program and subroutine** calls. This form of sub-architecture decomposes our main program structure into a number of subprograms or functions into a control hierarchy. Main program contains a number of other subprograms that in-turn invoke other components and modules of the system.

Working of our project architecture with respect to the **call and return** architecture:

1. The user executes the python application which is the main program
2. According to user input, a sub-program is executed. This order may differ from one user to another. The architecture is robust to difference in routine calls.
3. These sub-programs then access their specific libraries and invoke calls among themselves.
4. This process takes place continuously until the user decides to exit the program.

8. Code

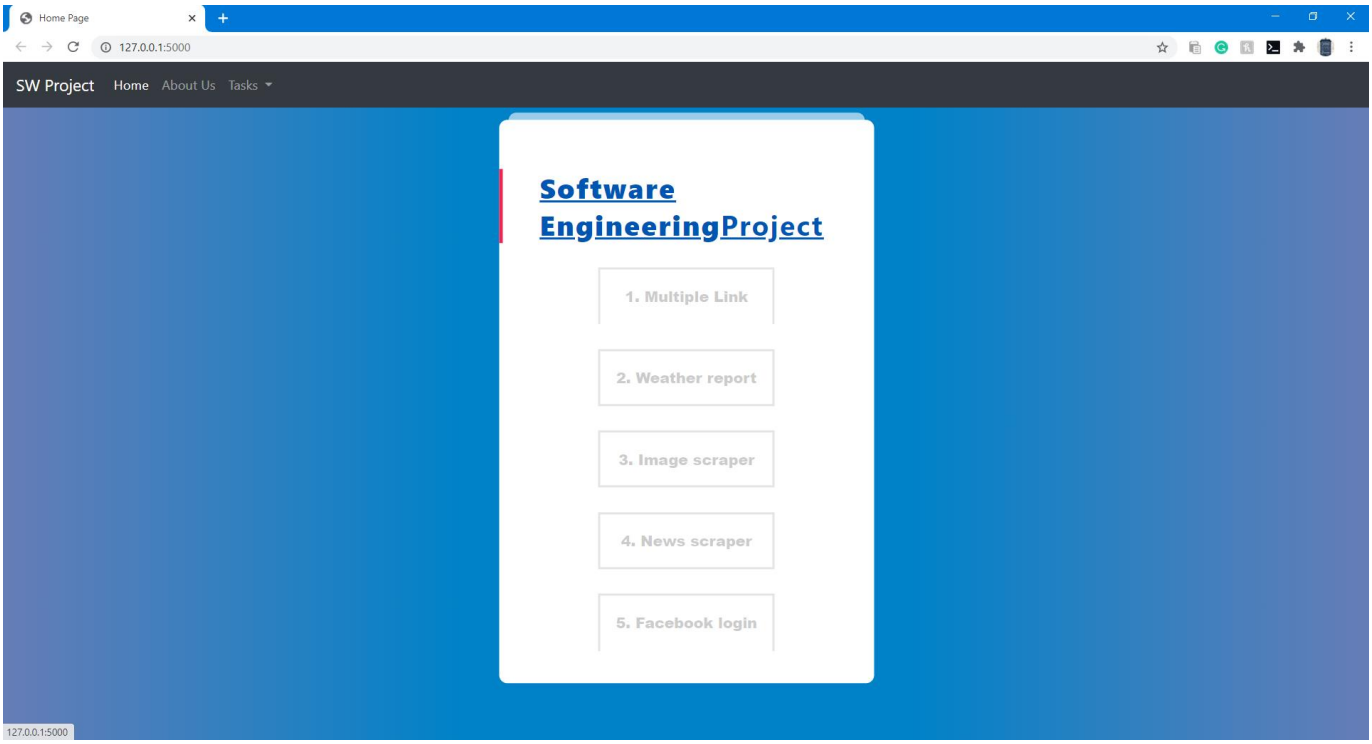
```
from flask import Flask, redirect, request, url_for, render_template
import pyowm
import requests
from selenium import webdriver
from getpass import getpass
import xml.etree.ElementTree
from urllib.parse import urlparse
 from bs4 import BeautifulSoup
from simple_image_download import simple_image_download as simp
import urllib3

app= Flask(__name__)
http = urllib3.PoolManager()
url = 'http://www.thefamouspeople.com/singers.php'
response = http.request('GET', url)
news={1:{'title':'','link':'','pubDate':''},2:{'title':'','link':'','pubDate':''},3:{'ti
tle':'','link':'','pubDate':''},4:{'title':'','link':'','pubDate':''},5:{'title':'','lin
k':'','pubDate':''}}
links={1:news[1]['link'],2:news[2]['link'],3:news[3]['link'],4:news[4]['link']}
soup = BeautifulSoup(response.data)

@app.route("/")
def home():
    return render_template("index.html")

@app.route('/task2')
def task2():
    return render_template('task2.html')

@app.route("/task2/result",methods=['POST'])
def task2_results():
    owm=pyowm.OWM('3282e54688306ae3571e778e51429885')
    s = request.form['location']
    observation = owm.weather_at_place(s)
    weather = observation.get_weather()
    temperature=weather.get_temperature('celsius')['temp']
    wind = weather.get_wind()['speed']
    humid = weather.get_humidity()
    status = weather.get_detailed_status()
    return render_template("task2_result.html",location=s,temp=str(temperature),wind=str
(wind),humid=str(humid),status=str(status))

@app.route('/task1')
```

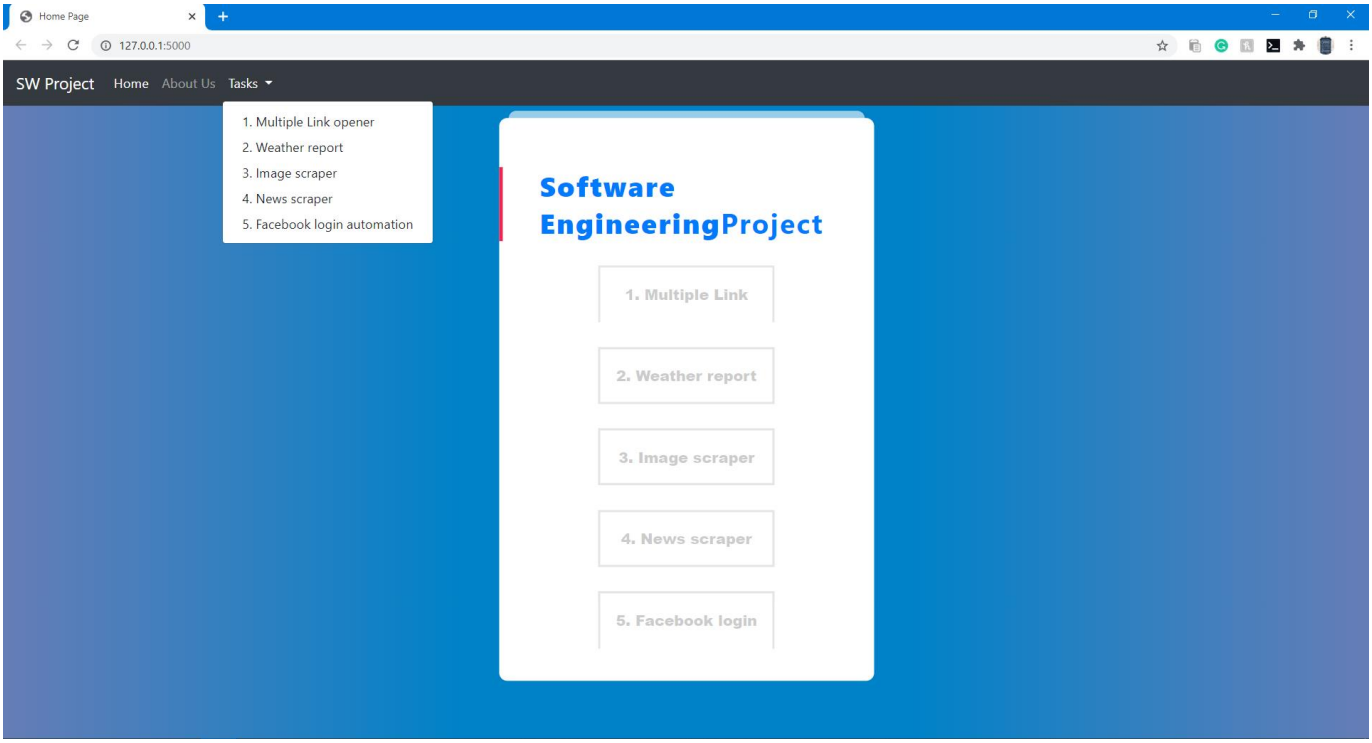
```
def task1():
    return render_template('task1.html')

@app.route('/task1',methods=['POST'])
def task1_option():
    option=request.form['option']
    link=request.form['link']
    l=link.split()
    t=len(l)

    if(option=="2"):
        for elements in l:
            elements=['https://www.' + elements + '' for elements in l]

    elif(option=="1"):
            for elements in l:
                elements=[elements for elements in l]

    driver = webdriver.Firefox()
    driver.get("https://www.google.com")
    page_number = 1
    for page in elements:
        driver.execute_script("window.open('');")
        driver.switch_to.window(driver.window_handles[page_number])
        driver.get(page)
        page_number +=1
    return render_template('task1_result.html')

@app.route('/task5')
def task5():
    return render_template('task5.html')

@app.route("/task5",methods=['POST'])
def task5_result():
    username=request.form['username']
    password=request.form['password']
    url='https://facebook.com'
    driver = webdriver.Firefox()
    driver.get(url)
    driver.find_element_by_id('email').send_keys(username)
    driver.find_element_by_id('pass').send_keys(password)
    return render_template('task5_result.html')

@app.route("/task3")
def task3():
    return render_template('task3.html')

@app.route("/task3",methods=['POST'])
```

```
def task3_result():
    key=request.form['key']
    num=int(request.form['num'])
    response = simp.simple_image_download
    imageurl1=response().urls(key,num)
    try:
        response().download(key,num)
        imageurl1=response().urls(key, num)
    except:
        pass
    return render_template('task3_result.html',imageurl=imageurl1)
@app.route('/task4')
def task4():
    news_url = "https://news.google.com/news/rss"
    xml_page=requests.get(news_url).content
    e= xml.etree.ElementTree.fromstring(xml_page)
    p = 1

    for it in e.iter('item'):
        if(p>4):
            break
        if(p==4):
            news[p]['title']=it.find('title').text
            news[p]['link']=it.find('link').text
            news[p]['pubDate']=it.find('pubDate').text
            p=p+1
        else:
            news[p]['title']=it.find('title').text
            news[p]['link']=it.find('link').text
            news[p]['pubDate']=it.find('pubDate').text

            p=p+1
    return render_template('task4.html',news=news,links=links)



if __name__ == "__main__":
    app.run(debug=True)
```

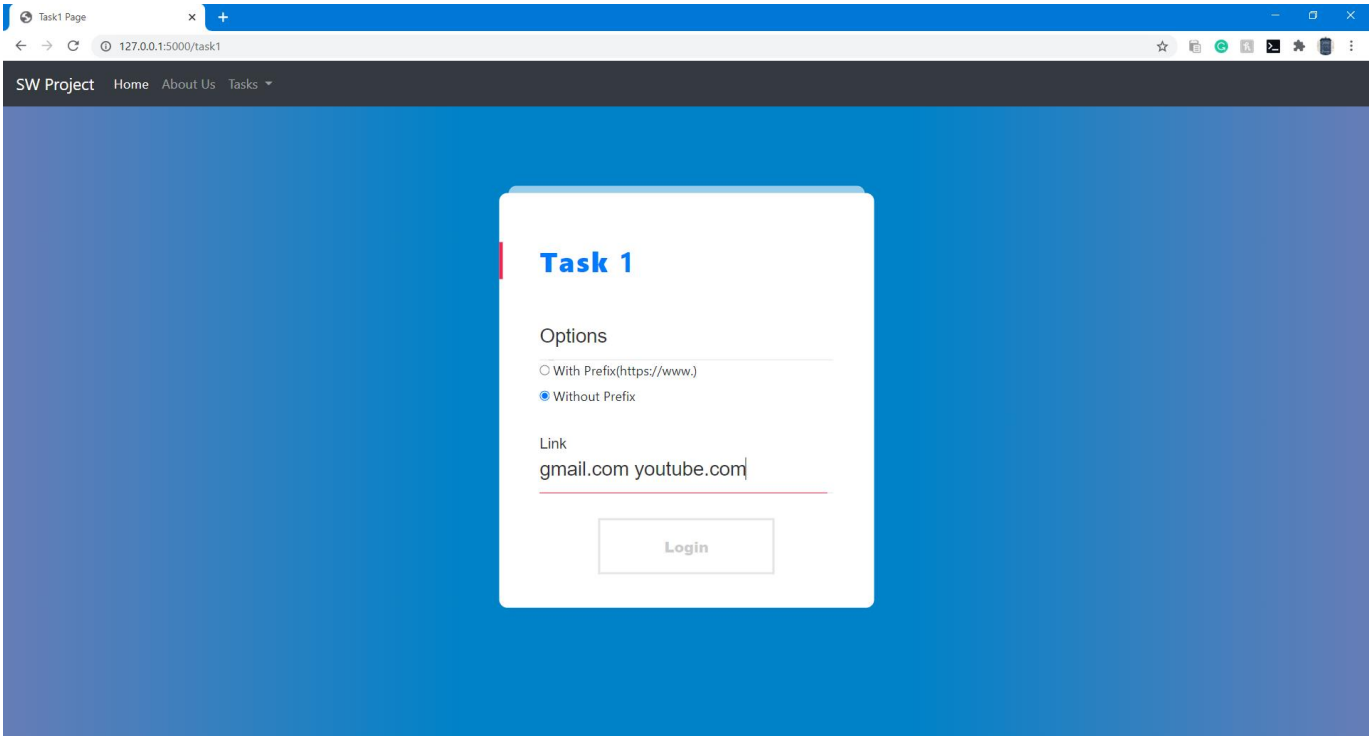
9. Complete Execution Screenshots

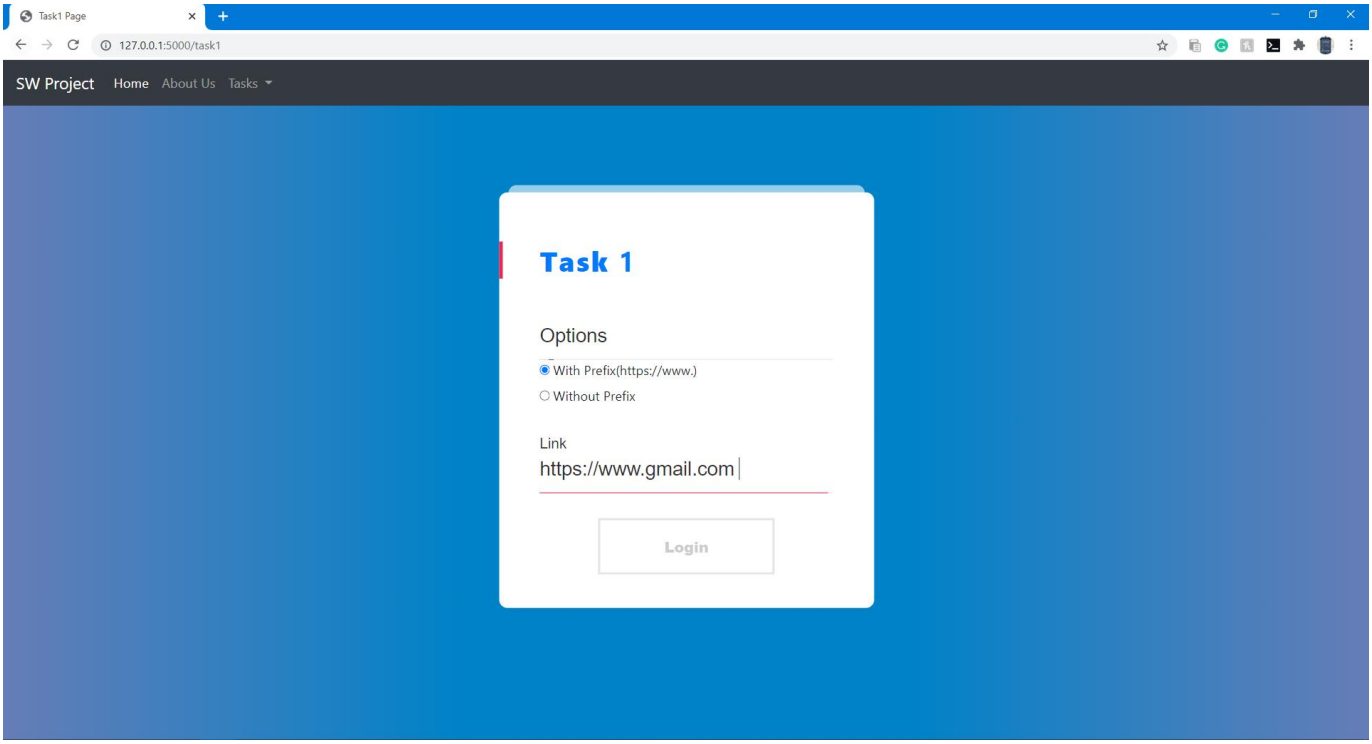
1) Home Page

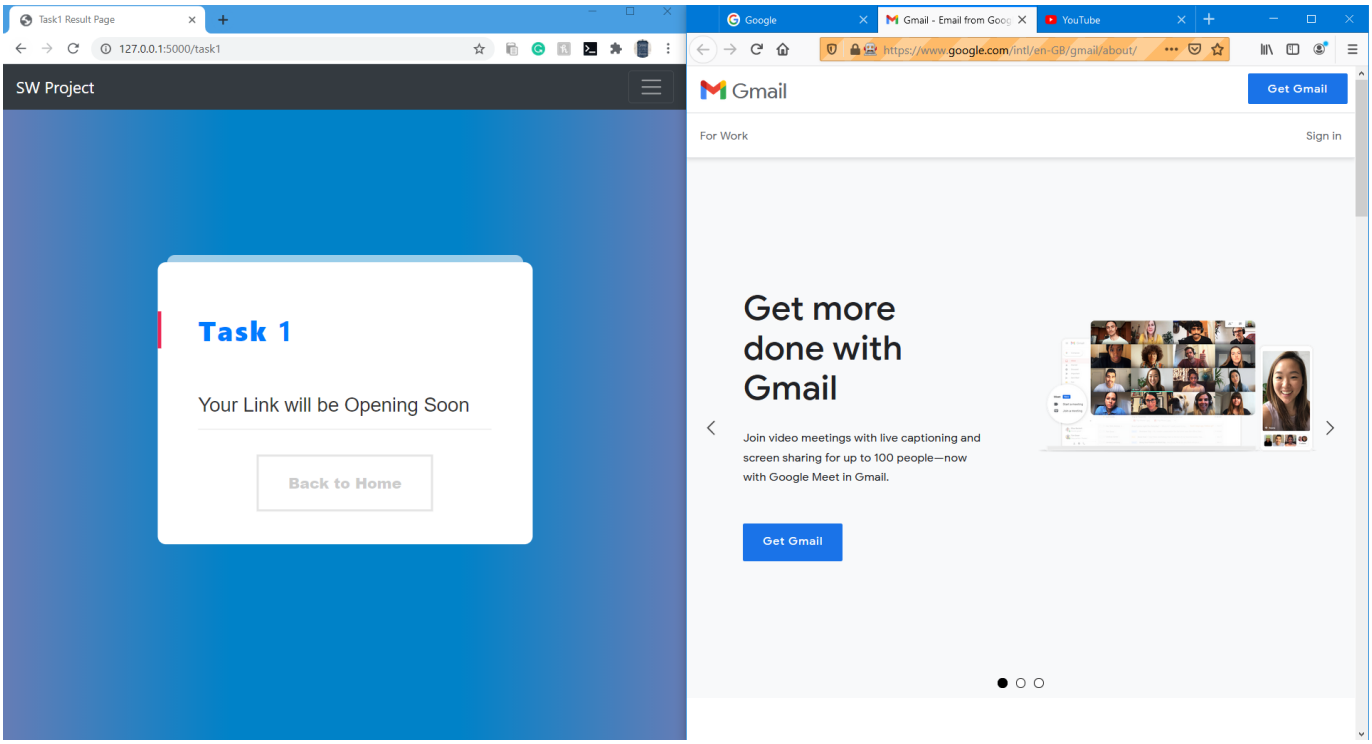




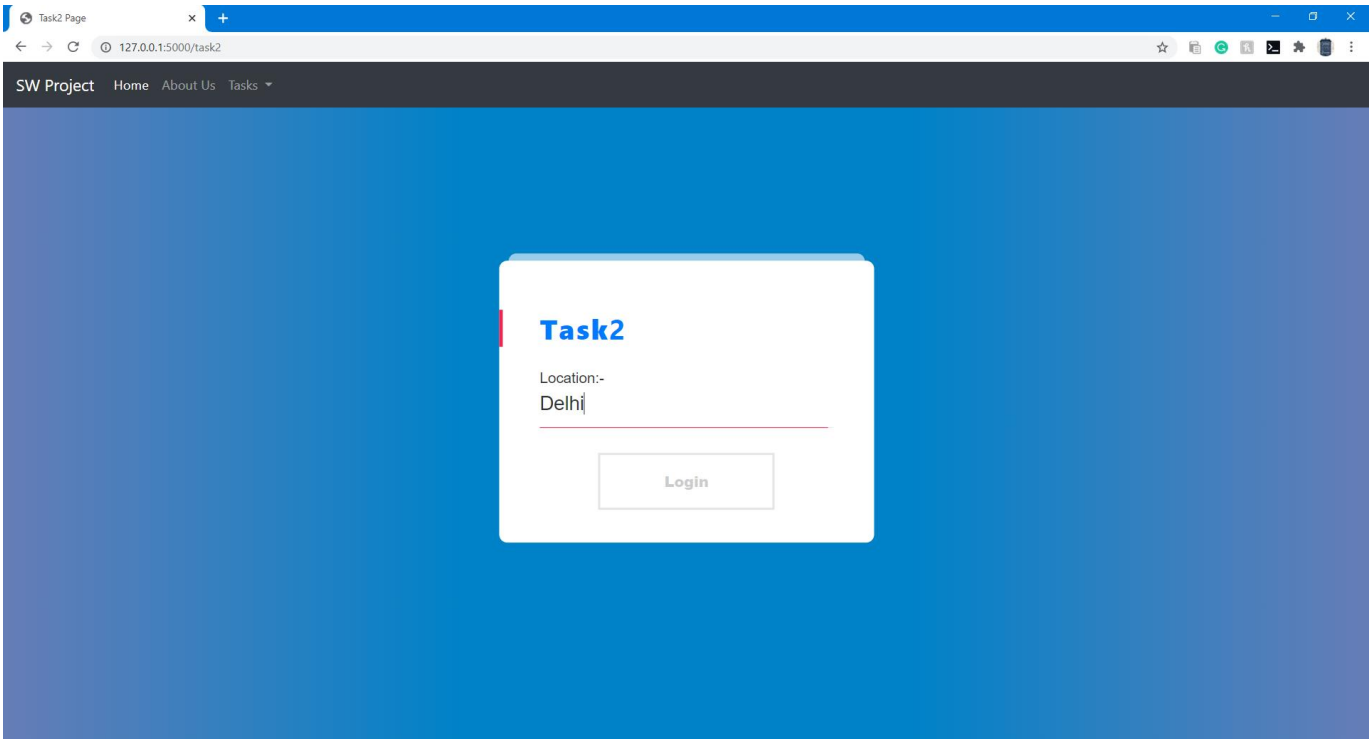
2) Multiple Link Opener

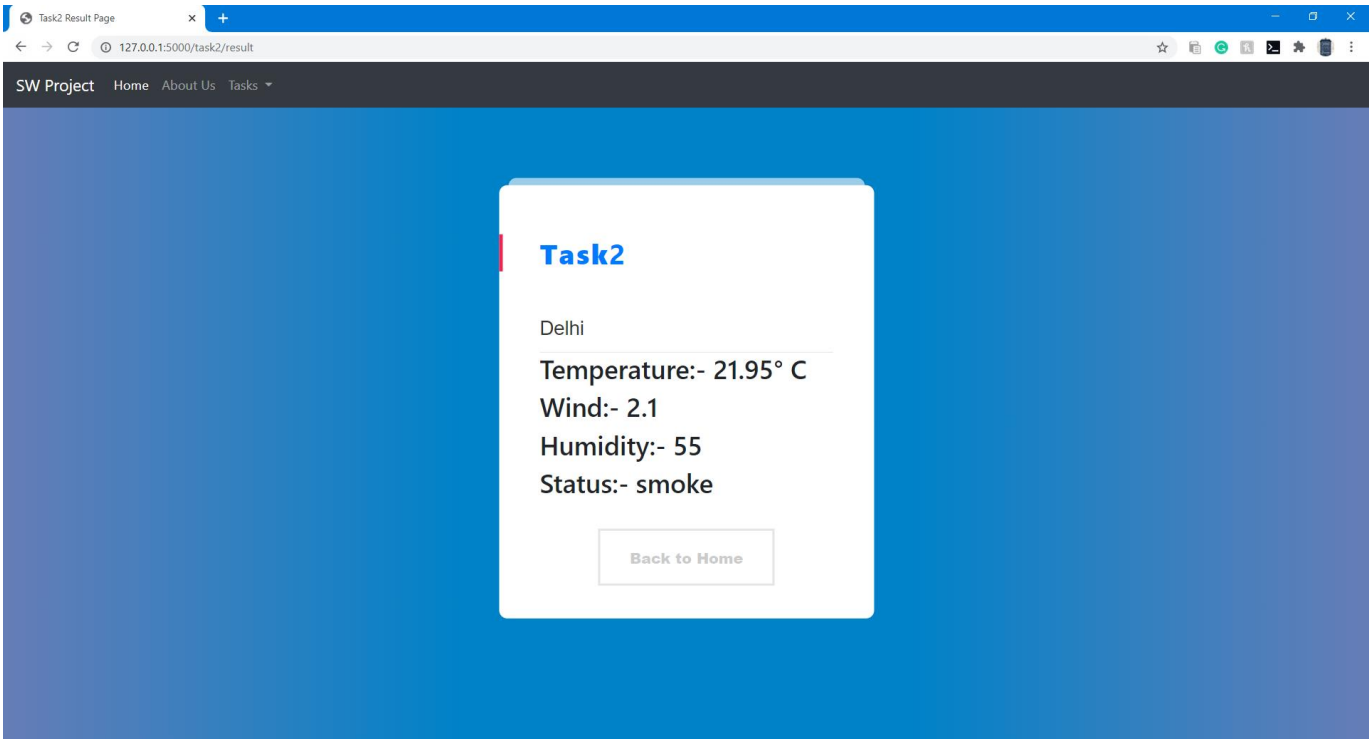




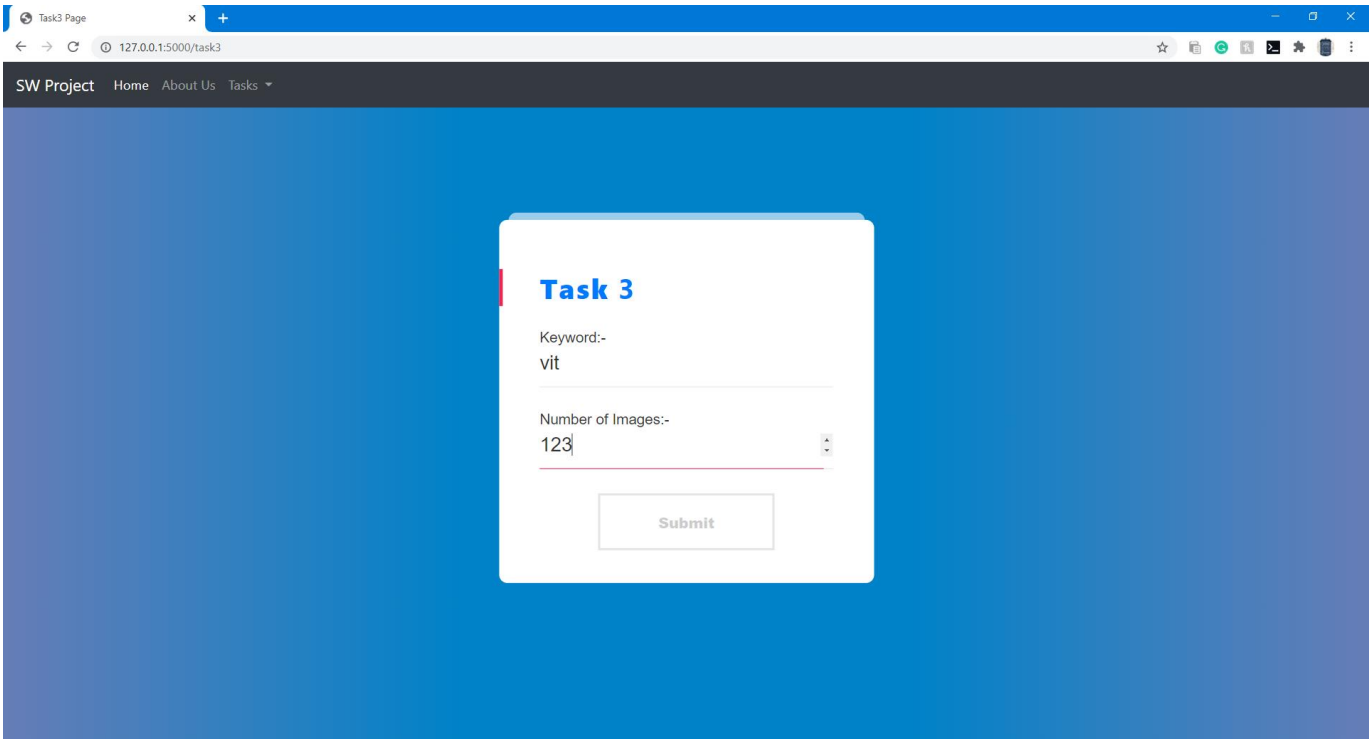


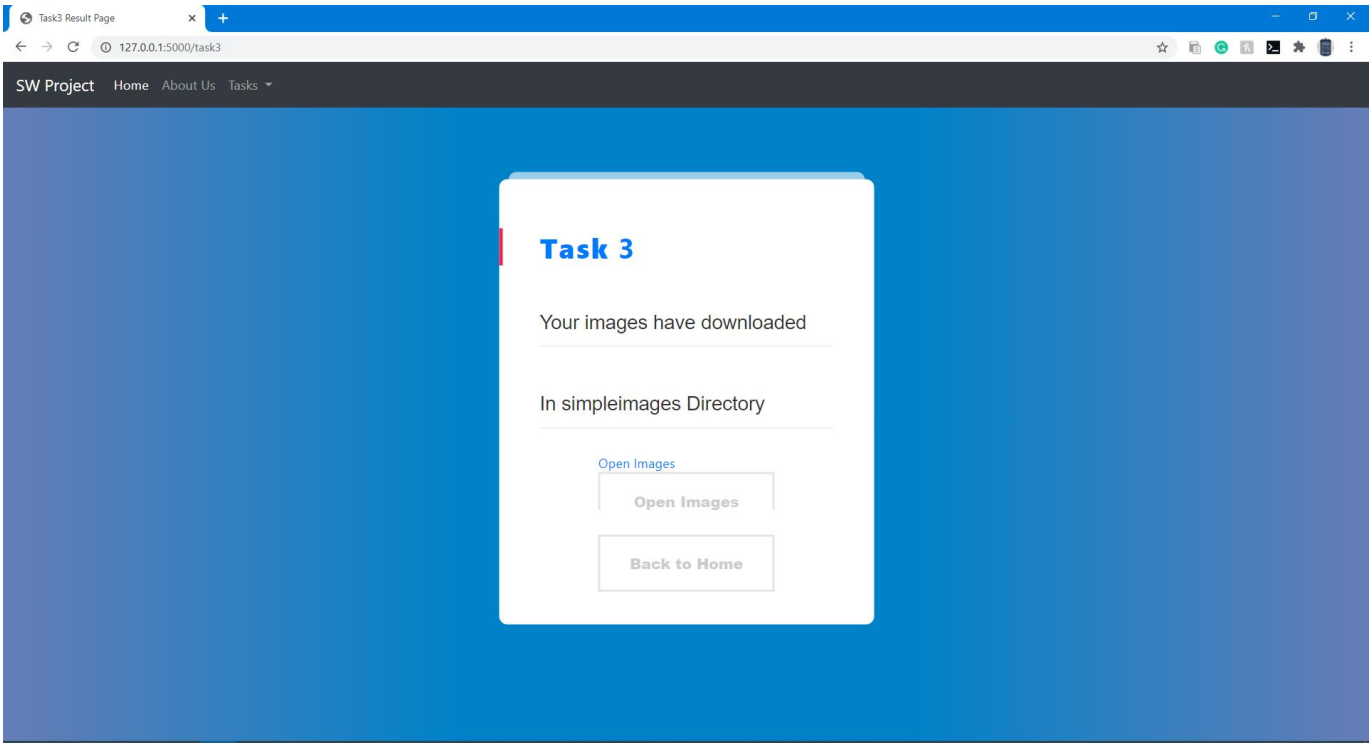
3) Weather Report

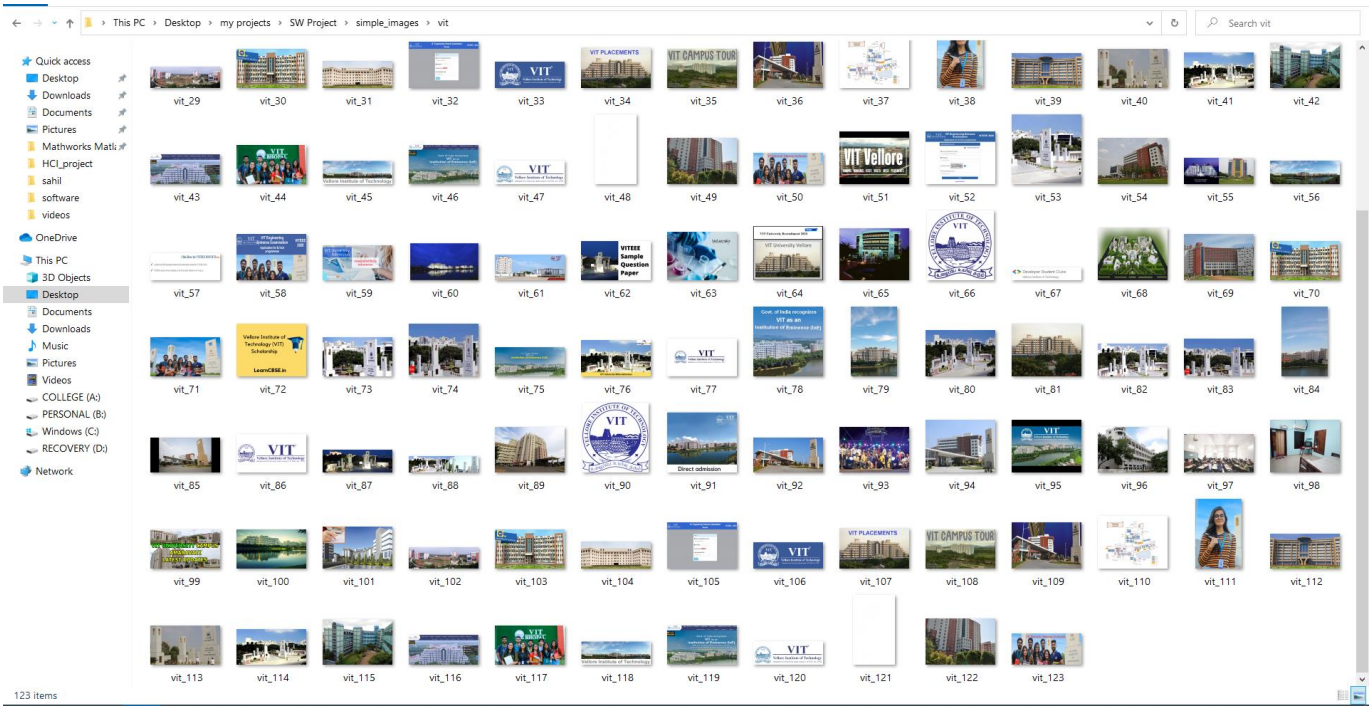




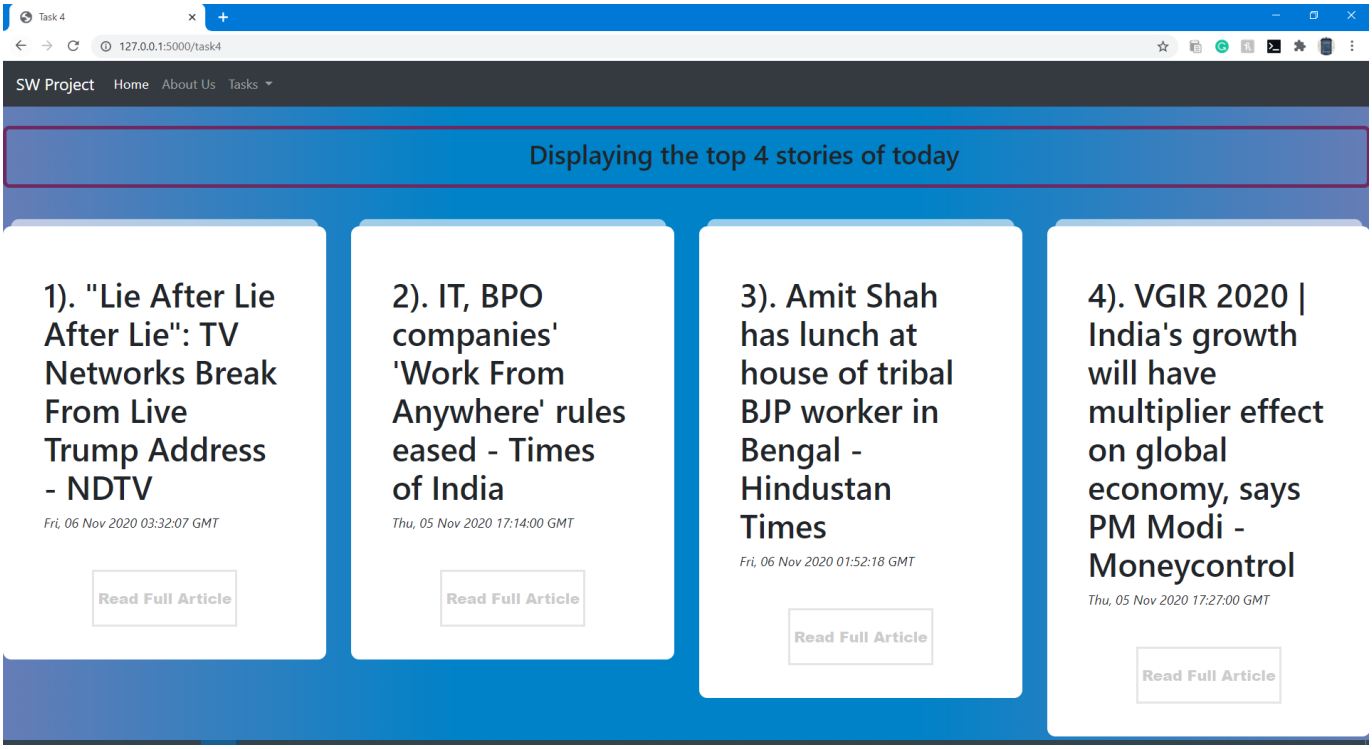
4) Image Scrapper



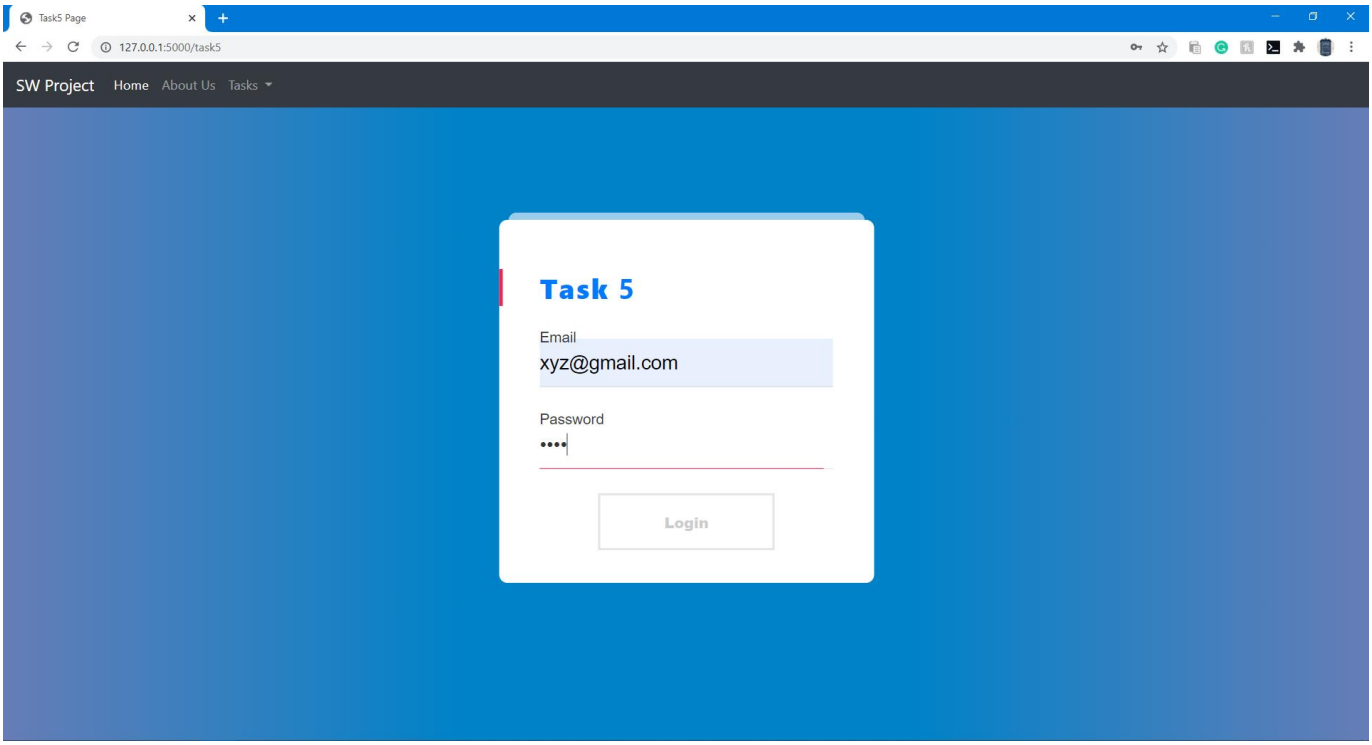

Task3 Result Page
127.0.0.1:5000/task3
SW Project
Home
About Us
Tasks
Task 3
Your images have downloaded
In simpleimages Directory
Open Images
Open Images
Back to Home

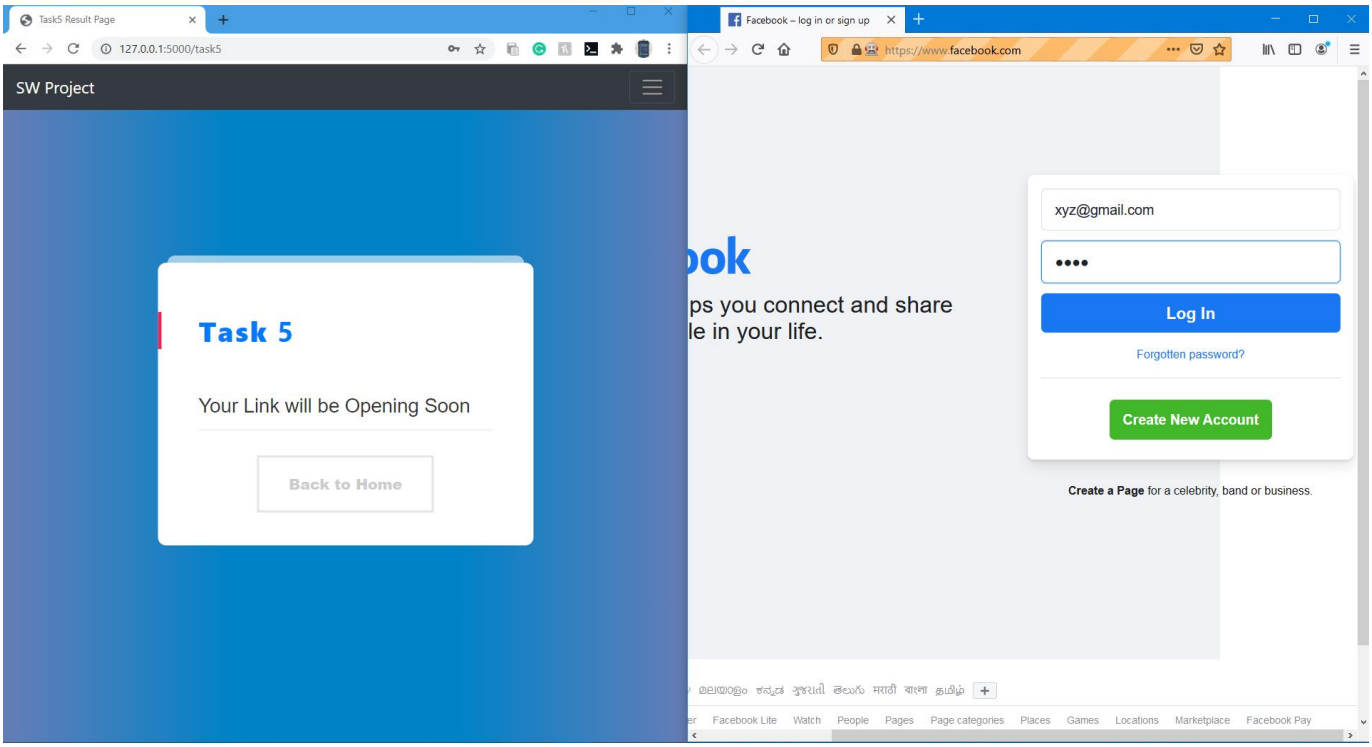

This PC
Desktop
my projects
SW Project
simple_images
vit
Search vit
123 items

5) News Scrapper



6) Facebook Login Automation




Task5 Result Page
127.0.0.1:5000/task5
SW Project
Task 5
Your Link will be Opening Soon
Back to Home
Facebook – log in or sign up
https://www.facebook.com
ps you connect and share
le in your life.
xyz@gmail.com
Log In
Forgotten password?
Create New Account
Create a Page for a celebrity, band or business.
Facebook Lite
Watch
People
Pages
Page categories
Places
Games
Locations
Marketplace
Facebook Pay

10. User Testing Report

Test Case ID	Test Case Objective	Pre-requistie	Input Data	Expected Output	Actual Output	Status
TC_01	Multiple Link Opener	Connection to Internet	Pressing button 1,1,youtube.com	A Firefox browser with the first tab as google search followed by the user input links	We get the error message that this url doesn't exist	Fail
TC_02	Multiple Link Opener	Connection to Internet	'Pressing button 1,2,youtuuuuube.com	A Firefox browser with the first tab as google search followed by the user input links	We get the error message that this url doesn't exist	Fail
TC_03	Program Termination from menu	Connection to Internet	Using Navbar to exit	The program exits and returns to the Main Menu	The program exits and returns to the Main Menu	Pass
TC_04	Multiple Link opener	Connection to Internet	Pressing Button 1, 1, https://www.youtube.com https://www.coursera.com https://www.gmail.com	A browser with 4 tabs, the first corresponding to google search followed by the links given as input	We get a Firefox browser (with a robot icon to denote the browser is automated) with all the correct links opened. We then get the prompt to exit and return to the main menu	Pass
TC_05	Multiple Link opener	Connection to Internet	Pressing Button 1,2,youtube.com gmail.com	A Firefox browser with the first tab as google search followed by the user input links	We get the browser (with the robot icon denoting automated) along with the desired links.	Pass
TC_06	Weather Report	Connection to Internet and access to weather API	Pressing Button 2, Delhi	The current temperature, Wind speeds, Humidity and status of the input location	It gives us the relevant weather information pertaining to the user input.	Pass
TC_07	Weather Report	Connection to Internet and access	Pressing Button 2, Dveuwgeuvhw	The current temperature, Wind speeds,	We get the error message along with a	Fail

		to weather API		Humidity and status of the input location	prompt to press any key to continue	
TC_08	Image Scraper	files should be available	pressing Button 3,dog,5	The CLI should output the image links of the keyword and a folder should be created with 5 dog images	The CLI should output the image links of the keyword and a folder should be created with 5 dog images	Pass
TC_09	Image Scraper	files should be available	Pressing Button 3,ddfwe,5	The CLI should output the image links of the keyword and a folder should be created with 5 dog images	The CLI should output the image links of the keyword and a folder should be created with 5 images related to keyword	Pass
TC_10	News Scraper	Connection to Internet	Pressing Button 4	News headlines, up-to 5 and then a redirect to the main-menu	We get our articles along with the date and time of their publishing.	Pass
TC_11	Facebook login automation	Connection to Internet, Facebook ID and Password	Pressing Button 5, (personal credentials)	Home screen of my logged in Facebook	It was able to open a browser and log in through my credentials	Pass

11. Tools Used

Software	Description
Windows	It is the operating system we use as it is the most user friendly operating system with the highest number of compatible software.
BeautifulSoup (BS4)	It is a parsing library that can use different parsers. Advantage of BS4 is its ability to automatically detect encodings and navigate parsed documents to extract needed data.
Selenium module – Gecko	It's the most popular tool for automating browsers primarily for testing. It's one of the easiest testing and automation software to use with python.
Scrapy	An open source and collaborative framework for extracting the data you need from websites, We're using it as it is a fast, simple, yet extensible way to scrape data.
Firefox	Mozilla Firefox, or simply Firefox, is a free and open-source web browser developed by the Mozilla Foundation and its subsidiary, Mozilla Corporation. Firefox uses the Gecko layout engine to render web pages
URLparse, URLlib, requests, OS and re	A set of python libraries that manipulate HTTP requests and the system directories. Re works to identify patterns.

12. Conclusion and Limitations

Given the right tools, automating computer operations can be surprisingly easy and can reap major benefits. Understanding these benefits—and some obstacles—can help one develop support for a project or task. Automation can lead to cost reduction, productivity, availability, reliability, and improved performance.

Automation technology, if used wisely and effectively, can yield substantial opportunities for the future. There is an opportunity for future automation technologies to provide a growing social and economic environment in which humans can enjoy a higher standard of living and a better way of life.

Through this project, we were successfully able to implement automation tools for common tasks done by users manually. We were able to use two popular programming interfaces: python and selenium to achieve the concept of automation in daily life. These two tools exponentially decrease the time required to do tasks when compared to their manual implementations.

Limitations:

The current tool-set has certain limitations as follows:

- It requires a new code for every different website scraped.
- It cannot perform automation and scraping together.
- The program may freeze or bottleneck when the input is too large.
- Multiple websites cannot be scraped together.
- It is heavily reliant on simple website architectures (light use of JavaScript and its frameworks)

Scope for future work:

With rapid development in artificial intelligence (AI) and robotics technology, automation is at a tipping point. Today, robots can perform a slew of functions without considerable human intervention. Automated technologies are not only executing iterative tasks, but also augmenting workforce capabilities significantly. In fact, automated machines are expected to replace almost half of the global workforce. Multiple industries, from manufacturing to banking, are adopting automation to drive productivity, safety, profitability, and quality.

Automation will bolster connectivity and reliability in a hyper-competitive ecosystem. The future of automation looks promising where everything will be made accessible and easily available.

13. References

- Volume 4 Issue VI, June 2016 IC Value: 13.98 ISSN: 2321-9653 International Journal for Research in Applied Science & Engineering Technology (IJRASET): Web Information Retrieval Using Python and BeautifulSoup: Pratiksha Ashiwal, S.R.Tandan, Priyanka Tripathi, Rohit Miri
- D. PRATIBA, A. M.S., A. DUA, G. K. SHANBHAG, N. BHANDARI and U. SINGH,"Web Scraping And Data Acquisition Using Google Scholar," 2018 3rd International Conference on Computational Systems and Information Technology for Sustainable Solutions (CSITSS), Bengaluru, India, 2018, pp. 277-281, doi: 10.1109/CSITSS.2018.8768777.
- 2nd International Symposium on Big Data and Cloud Computing (ISBCC'15) Analysis and Design of Selenium WebDriver Automation Testing Framework Satish Gojarea,Rahul Joshib ,Dhanashree Gaigawarec