A project report on

"WEB APP TESTER"

Submitted in partial fulfilment for the award of the degree of

B. Tech Computer Science and Engineering

Submitted to:
Dr. Swathi J.N.

by

NAME		REG.NO
1. Ritik Pratap Singh	-	18BCE0631
2. Ayush Jain	-	18BCE2245
3. Ritvik Tyagi	-	19BCI0180

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and Engineering

Department of Computer Science and engineering

June 2021

DECLARATION

I hereby declare that the thesis entitled **"Web App Tester"** submitted by 1. Ritik Pratap Singh (18BCE0631), Ayush Jain (18BCE2245) and Ritvik Tyagi (19BCI0180), for the award of the degree of Specify the name of the degree VIT is a record of bonafide work carried out by me under the supervision of Dr. Swathi J N.

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place: Vellore

Date: 5/6/21

Signature of the Candidate

RITIK PRATAP SINGH

AYUSH JAIN

RITVIK TYAGI

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENT

It is my pleasure to express with deep sense of gratitude to **Dr. SWATHI J N**, Associate Professor, School of Computer Science and Engineering, Vellore Institute of Technology, for her constant guidance, continual encouragement, understanding; more than all, she taught me patience in my endeavor. Our association with her is confined to academics only, but it is a great opportunity on our part of work with an intellectual and expert in the field of Software Engineering

We would like to express our gratitude to **G VISWANATHAN(Chancellor)**, **SEKAR VISWANATHAN (Vice President)**, **DR. ANAND A. SAMUEL(Vice-Chancellor)**, **S NARAYANAN (Pro-Vice Chancellor), Dr. RAMESH BABU K (DEAN, School of Computer Science and Engineering) and special thanks to Dr. VAIRAMUTHU S (Head of Department, School of Computer Science and Engineering).** for providing with an environment to work in and for his inspiration during the tenure of the course.

It is indeed a pleasure to thank my friends who persuaded and encouraged me to take up and complete this task. At last, but not least, we express my gratitude and appreciation to all those who have helped me directly or indirectly toward the successful completion of this project.

EXECUTIVE SUMMARY

We are now living in a time where users expect as much functionality, reliability and flexibility from Web apps as desktop programs. All our data is slowly but surely moving into the cloud, and businesses are looking to the Web for software to easily implement and deploy across the board – whether it is accounting, CRM or inventory management. That puts a lot of pressure on developers to deliver rock-solid Web apps that users can genuinely depend on for work and play.

That is why it is so important to thoroughly test your Web application before launch. Presenting a quality web app that does not break, works efficiently and delights users naturally builds a foundation of trust between you and your customers, and they'll be happy to use it more often and even refer it to peers.

So, we aim to build a web app tester that will enable a user to check for broken links, difficult navigation, web security and other potential risks or bugs. The app aims to have three main modules which will focus three main types of testing in a web app. The modules are Functionality test which aim to test broken links, test forms, cookies and test HTML and CSS syntax for errors. Usability testing module aims to test navigation in web app such as if buttons are visible, checkboxes are clickable etc. and test content in the web app. The third module is Security testing which aims to test the security of the web app by checking if there any broken or invalid links that might creep into the web page.

The Scope of this project is to help the testers to test their web app In development phase before launching it to general public.
This Software aims to build a web app tester that will perform the basic and necessary tasks on the web app such as Syntax errors and hyperlink testing which are very crucial for proper functioning of a web app.
The Software can be further extended in future versions to inculcate more functionalities such as Interface testing and ease of use by users which is very much demanded nowadays in Web applications. This will help the web app developers to keep their app error free and also introduce ease of use.

1. INTRODUCTION

1.1 OBJECTIVE

The main aim of the project is to provide a tool which can be helpful in testing other software and projects. The main objectives are to test the HTML errors in the web pages, test the CSS errors in the pages and to find the broken links by using the method of web crawler that searches all the links. We have also achieved in providing reports to our users which help them locate the error and correct them so that their software will function better.

Our objective was to provide an efficient login and sign -up system for our users that will help them visualizing all these errors and provide smooth login for their account.

Another objective of our project was to implement any Testing application because Testing is one of the more important activities in the umbrella activities of software engineering. It is also to test the software before providing to the users as the system is risk oriented and any mis-leading scenarios may cause the system to crash or the user might be unhappy and dissatisfied with the software.

1.2 MOTIVATION

The main motivation to begin working on such a project started from the basic requirement of any software developer that is Testing. As we have mentioned in the objective, we are motivated to provide users faster methods to find errors in their projects and quickly locate and correct them.

Also, the other part of our motive was to provide a tool in study of Testing as it is an integral part of any software developed under any scope.

1.3 BACKGROUND

Web Testing, or website testing is checking your web application or website for potential bugs before it is made live and is accessible to general public. Web Testing checks for functionality, usability, security, compatibility, performance of the web application or website.

During this stage issues such as that of web application security, the functioning of the site, its access to handicapped as well as regular users and its ability to handle traffic is checked.

Web based Testing Activities includes:

Test all links in your webpages are working correctly and make sure there are no broken links. Links to be checked will include -

- Outgoing links
- Internal links
- Anchor Links
- MailTo Links

Test HTML and CSS to ensure that search engines can crawl your site easily. This will include

- Checking for Syntax Errors
- Readable Color Schemas
- Standard Compliance. Ensure standards such W3C, OASIS, IETF, ISO, ECMA, or WS-I are followed.

Also, we tried to find a software Selenium which is using these features and providing users the application of testing Web Application. We then tried to take motivation from this tool and tried to implement some parts in our project.

About Selenium:

Selenium is a portable framework for testing web applications. Selenium provides a playback tool for authoring functional tests without the need to learn a test scripting language (Selenium IDE). It also provides a test domain-specific language (Selenese) to write tests in a number of popular programming languages, including C#, Groovy, Java, Perl, PHP, Python, Ruby and Scala. The tests can then run against most modern web browsers. Selenium runs on Windows, Linux, and macOS.

2. PROJECT DESCRIPTION AND GOALS

2.1 Stake holders:

1) Web App developer team and managers – They are responsible for implementing the tests, receiving the test results and take actions like fixing bugs based on the results.
2) Technical Experts – They are usability exerts, security consultants, hardware people, experts in the technologies, or experts from any technical field that the web app could use.
3) Business and marketing analysts – They define the product features and their expected quality. They also contribute in defining test coverage, analyzing test results and taking decisions on the basis of those results.
4) Direct and Indirect Users – They are the users who use the web app directly, receive output from the web app or get support from the web app.

2.2 Selection of Software process model:

We have selected the Spiral Model because of the following reasons:

- As this is project involves new Technology which is new for us too and completely different than the other projects we did, therefore this model gives the time to explore each aspect our project correctly before moving on the other phase.
- As our project intends to check the complete functional testing and other tests that will be checking how user friendly is the web app therefore, we intend to understand each requirement.
- This model will enable our project to go through iterations necessary to ensure that our software performs properly and any bug is not left out and the client can identify and convey the changes required in our project.
- Also, we do not have enough expertise and resources available so we need time which this model of planning will provide.

2.3 WORK BREAKDOWN STRUCTURE:

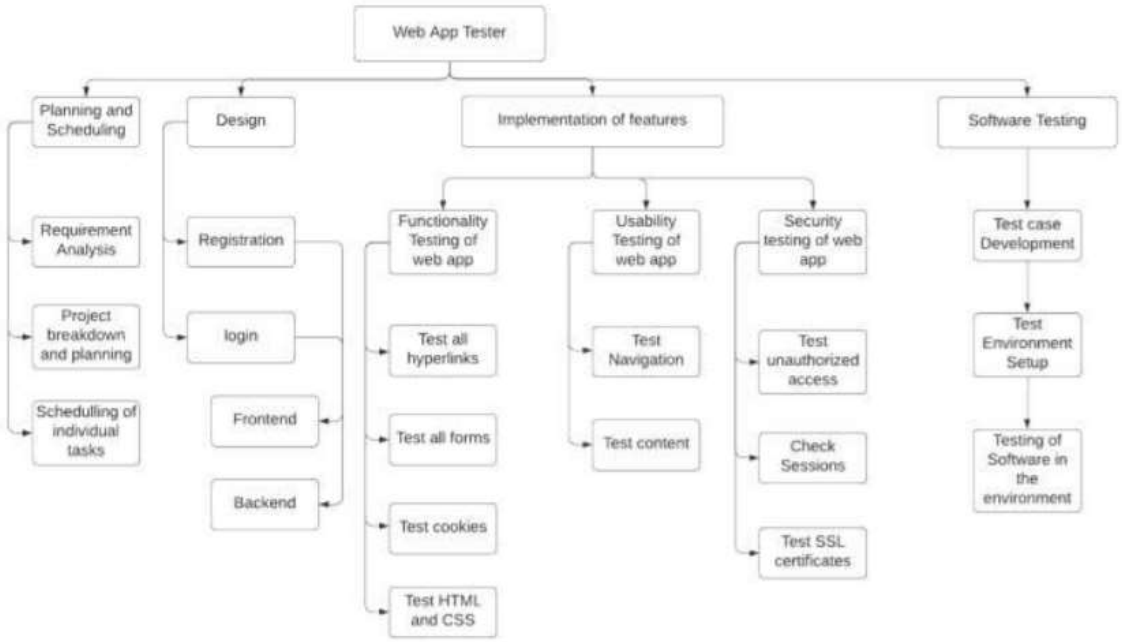


Figure 1: Work Breakdown Structure

3. TECHNICAL SPECIFICATION

3.1 PRODUCT PERSPECTIVE

Web app Tester Software is a new Web app testing software, designed with parallelization in mind. It was built to dramatically increase The web app testing phase of development.

The main distinguishing factor of Web App tester is its uniqueness which is not following the traditional ways of App testing and still getting the work done using the same traditional algorithms ensuring correct standpoints.

Web app tester will allow you to perform several tests at the same platform and hence saving your time and effort to do those tests separately on separate software.

3.2 PRODUCT FUNCTIONS

The App before being launched needs to be checked if it contains any error or logic problem which can cause error to users. This is done by running various tests on the Web Application. This is called Web App testing.

Many tests need to be performed on the web application to ensure that the web app performs well and without any errors. This is done with the help of various platforms which employ various tests which can be hectic as well as cumbersome for the tester.

Web App tester solves this problem by performing various tests on the web app at a single platform and hence saves time for the tester and helps ensure that the web app is bug free and performing all the functions as it should perform during the running phase. It employs tests such as Security tests, functionality tests and usability tests.

3.3 USER CHARACTERISTICS

Web app tester is a simple Web app testing software which can be used by any web app developer to use test cases for his web application.

The user needs to type the link of his web app software and a simple mouse click will enable the user to perform various tests on his web app at a single location and save his time too.

The user needs to have some information about the type of errors in order to rectify them as the software will be able to just detect the errors and not suggest methods to rectify those errors.

3.4 SPECIFIC REQUIREMENTS

3.4.1 System Features - Login and Sign-up

3.4.1.1 Introduction

The automated testing should only be available to verified end users. So, their must be an appropriate facility for users to login.

3.4.1.2 Functional Requirements

Purpose: Verifying whether user has authorization to use the software or not.
Input: Username and password. Additionally, login using google account or GitHub account can also be provided.
Processing: Taking input of username and password and comparing them with the credentials in database to validate.
Output: User can either use the software or not.

3.4.1.3 Stimulus Response

A) User Sends credentials for verification.

User Actions	System Actions
(1) Enter user credentials	
(2) Initiate the Login function	
	(3) Compare the user sent credentials to the ones in the database.
	(4) If credentials don't match then display invalid credentials page
(5) On unsuccessful login attempt user can try again.	
	(6) If credentials match, then allow user to use the software

3.4.2 Checking Syntax

3.4.2.1 Introduction

This feature allows users to check for any syntactical error in their web-app. Most of the web-apps are written using HTML, CSS and JavaScript. The software checks for basic syntactical errors in the whole document. E.g., closing all elements tag in the HTML document.

3.4.2.2 Functional Requirements

Purpose: Checking syntax of the web app.
Input: URL of the webapp to be tested for syntax errors.
Processing: Software reads throughout the whole code and check for syntactical errors with the errors in the database.
Output: List of all the syntactical errors found in the code

3.4.2.3 Stimulus Response

User Actions	System Actions
(1) User provides URL for their webapp	
	(2) System Checks for presence of any error
	(3) Errors found are appended into a list
	(4) Display the errors and their location, if any

3.5 Performance Requirements

The following tables list the performance requirements of the Web App Tester software.

Performance Requirement	Description
Response Time	The software should be able to iterate through the entire webapp and find the bugs in very low amount of time.
Bugs detected	The Web tester should be able to find most of the potential bugs and exploits in the web app.

3.6 Software System Attributes

3.6.1 Reliability

Reliability in the Web App tester will be ensured by thorough unit, milestone, and release testing.
Comprehensive test scenarios and acceptance criteria will be established to reflect the necessary level reliability required of the Web App tester. The all delivered source code will be thoroughly tested using the established test scenarios until the acceptance criteria are satisfied by the Web App tester.

3.6.2 Security

The Web App tester will have login feature which will ensure that the privacy and data of the user is safe at all times. As the programming in which it is written is PYTHON, it ensures high security as PYTHON is famous for its secure code writing practices.

3.6.3 Maintainability

The Web App tester is written in Python programming language. PYTHON promotes good design practices due to the inherent structure of a PYTHON program.
Along with the well-formed programming enforced by PYTHON, best practice development conventions will be enforced for the construction of the Web App tester. Consistent variable naming conventions will be used by all the programmers. Consistent spacing will be used in the source code by all the programmers. The design of the source code will use the principles of Object-Oriented Design and the source code will be programmed using Object Oriented Programming. Object-Oriented Design and Object-Oriented programming will make the code easier to understand.

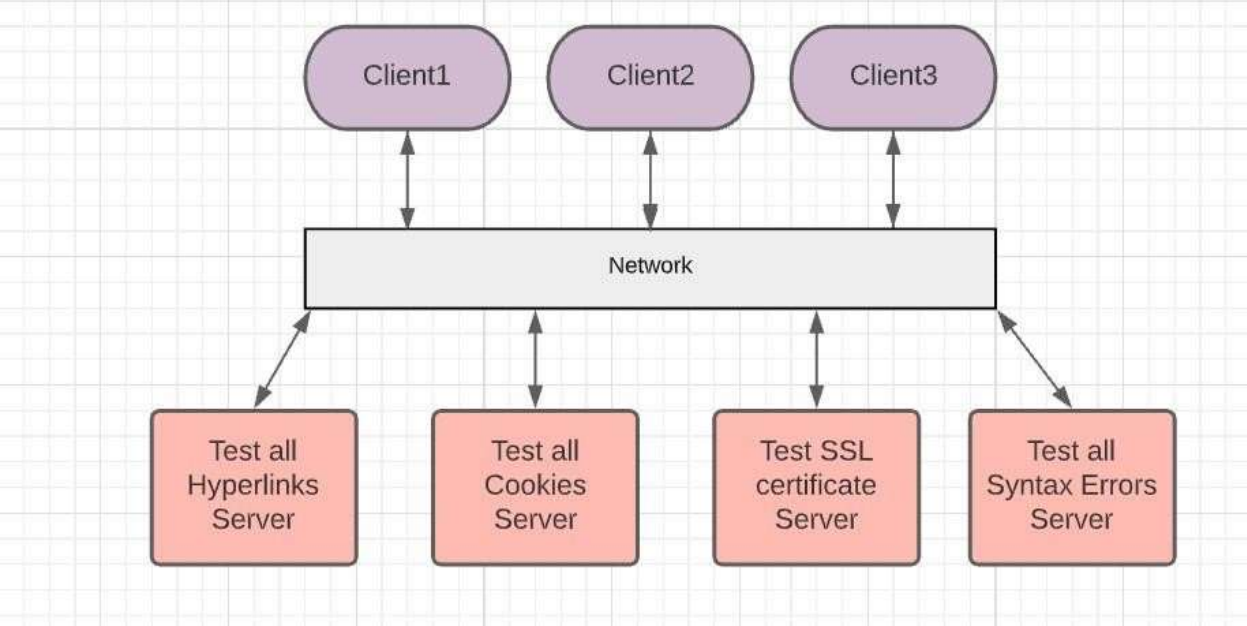
3.6.4 Portability

It is safe to say that the implementation of the Web App tester will be able to be ported to other system platforms that accept PYTHON applications with little to no changes required. It is not safe to say that the Web App tester will execute properly on the other system platforms with little or no change. Significant changes to the Web App tester may be required to ensure proper execution on other system platform.

4. DESIGN APPROACH AND DETAILS:

4.1 DESIGN APPROACH / MATERIALS & METHODS:

4.1.1 Architectural Diagram:

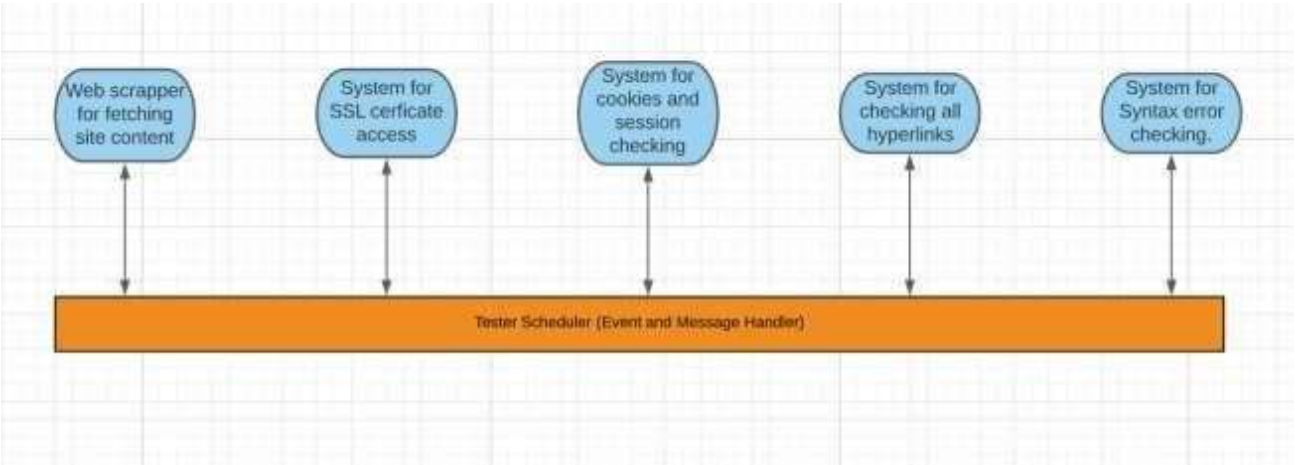


We are choosing Client-Server model for designing architecture for our project Web App Tester.

Reasons for choosing Client-Server Context Model:

- Since our project aims at providing our users an application that tests their web applications and get an output from us validation their site this model will be efficient way to get the required output.
- We can divide our requirements into modules like Test All Hyperlinks as one server function, Test navigation as another server, test all forms and cookies as another server function and other requirements can also be handled in a similar way.
- This will give the client the option to individually test all these functions and validate their product.
- Another reason for choosing this model is that later we also additional test features which the client may ask for in future as for adding these features it will not impact any existing subsystem. • There is no huge database involved and not much of data flowing between the individual sub-systems so this is an appropriate way for designing our project.

4.1.2 Control Model

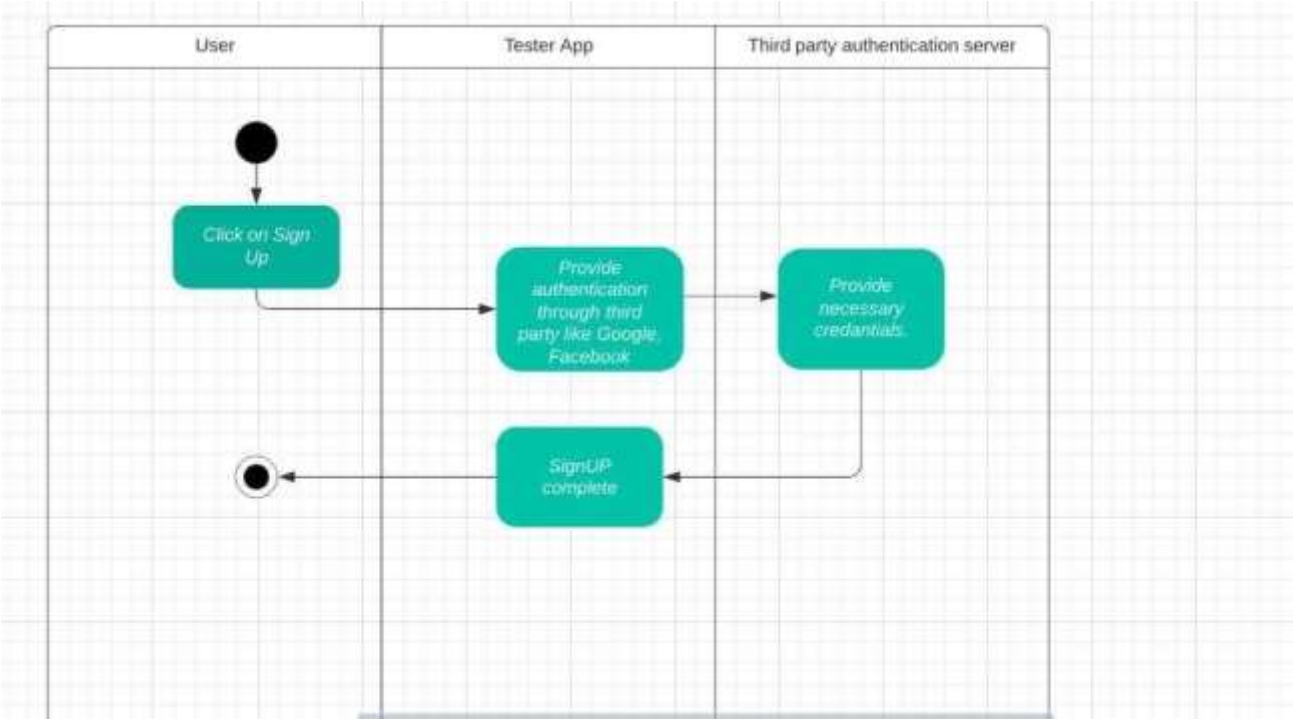


We have chosen broadcast model because event-based model is driven by externally generated events where the timing of the event is outside the control of the sub-systems which process the event

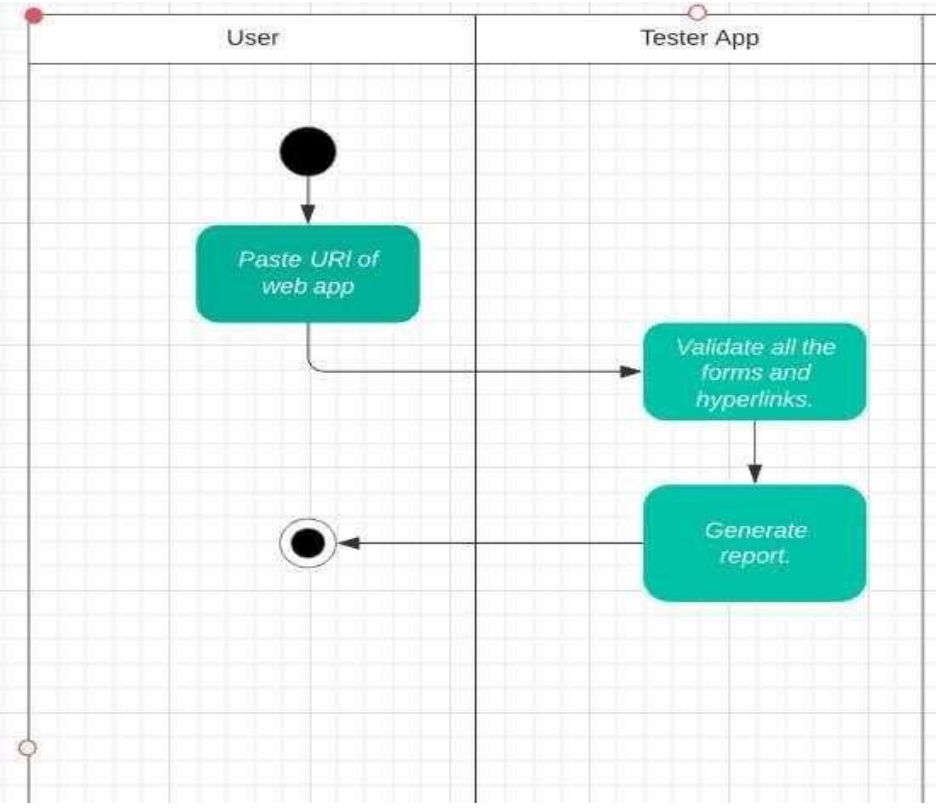
- Since our Software will be running many tests on the web application broadcast model will help us to broadcast an event to all the sub-systems, i.e., running tests on the web application.
- As in broadcast model any subsystem which can handle the broadcasting event may behave as a broadcast model it will be appropriate for our software.
- As we do not need a control policy for our software, Broadcast model is suitable as then sub systems can decide on event which are of interest to them.

4.1.3 Swimlane Activity Diagram:

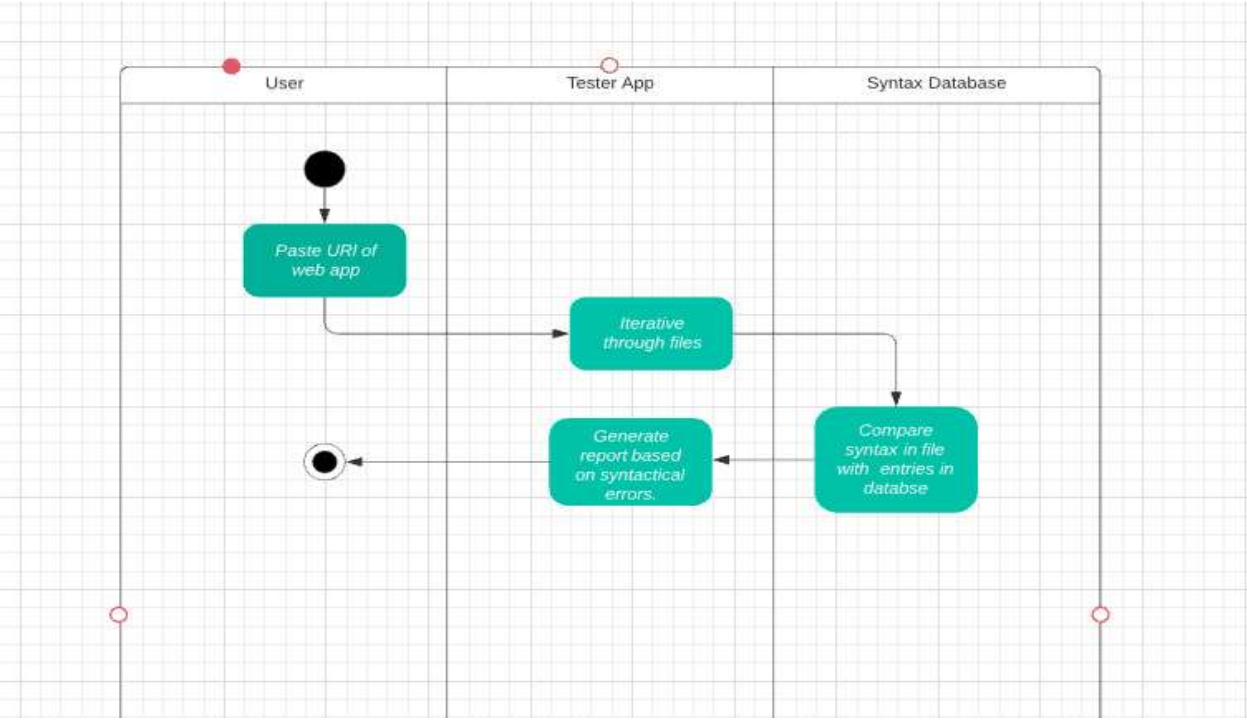
a. Login Activity



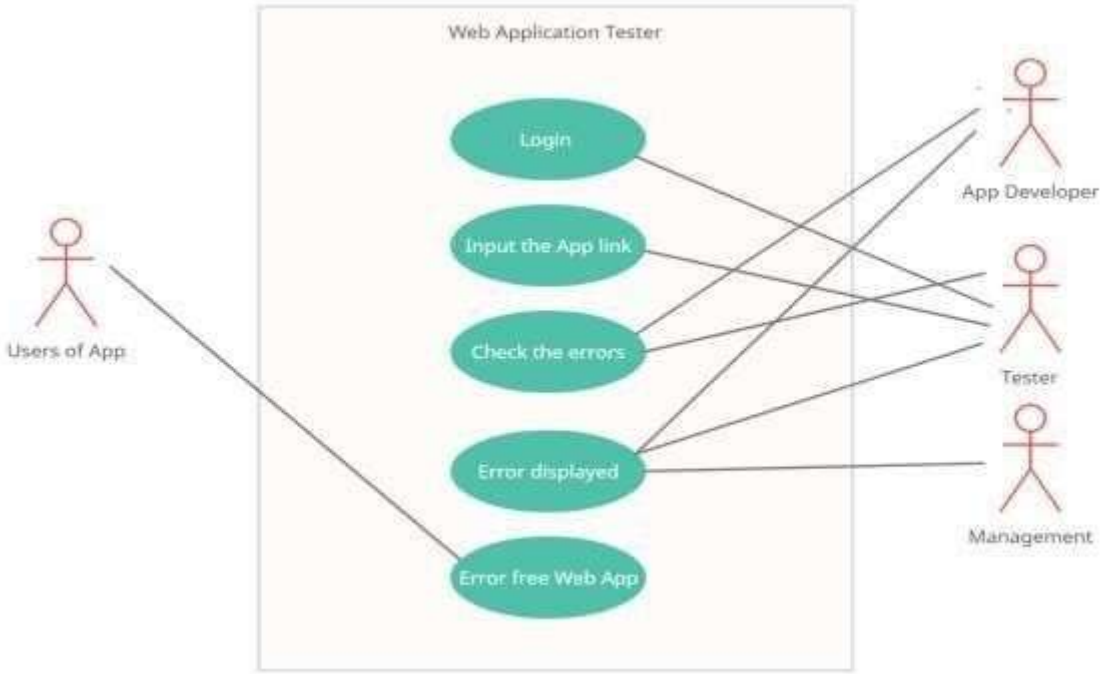
b. Hyperlink Validation



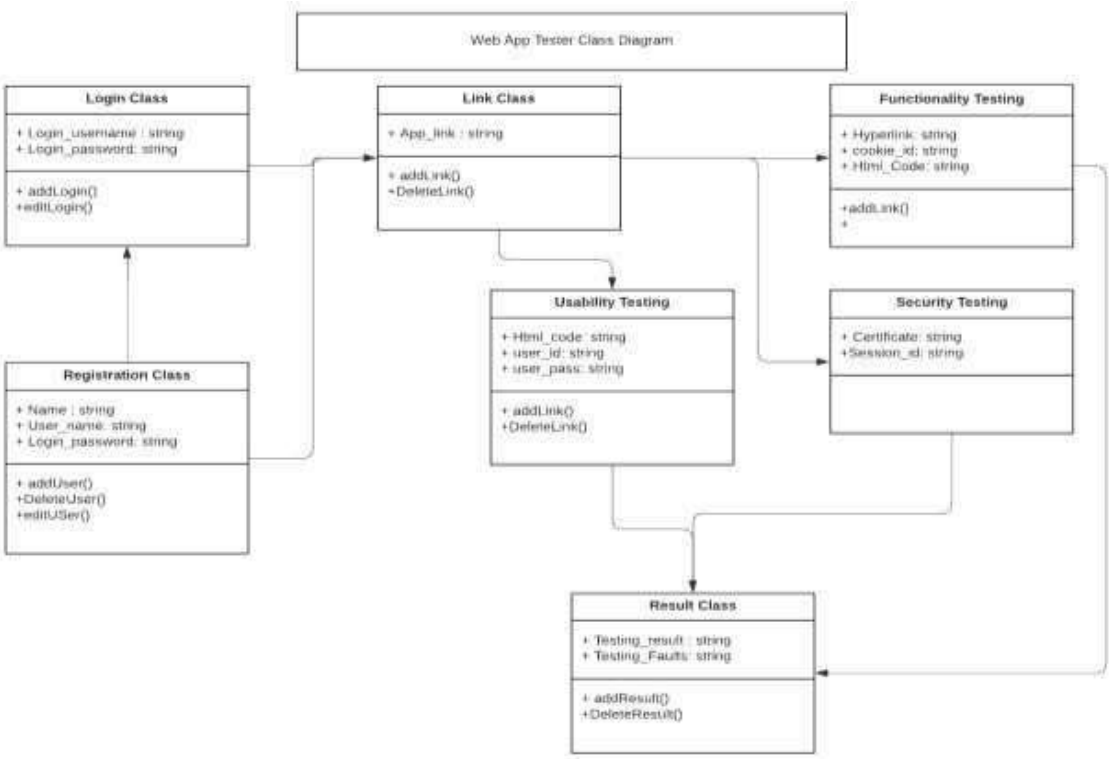
c. Checking Errors



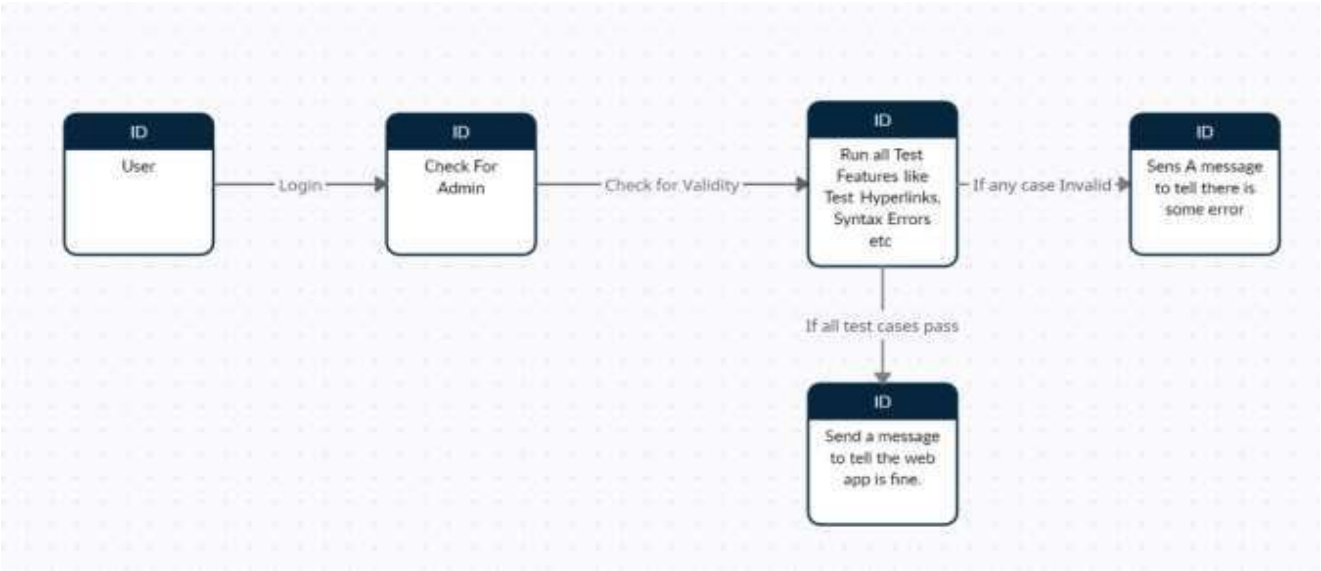
4.1.4 Use Case Diagram:



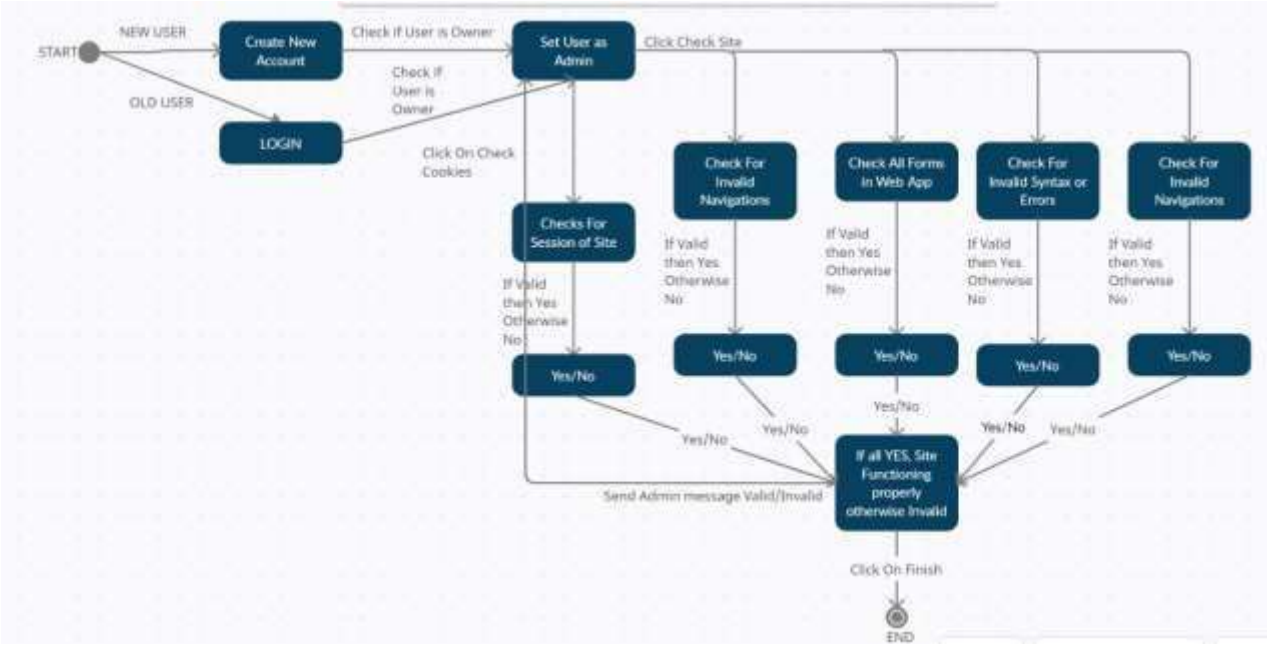
4.1.5 Class Diagram:



4.1.6 Data Flow Diagram:



4.1.7 State Transition Diagram:



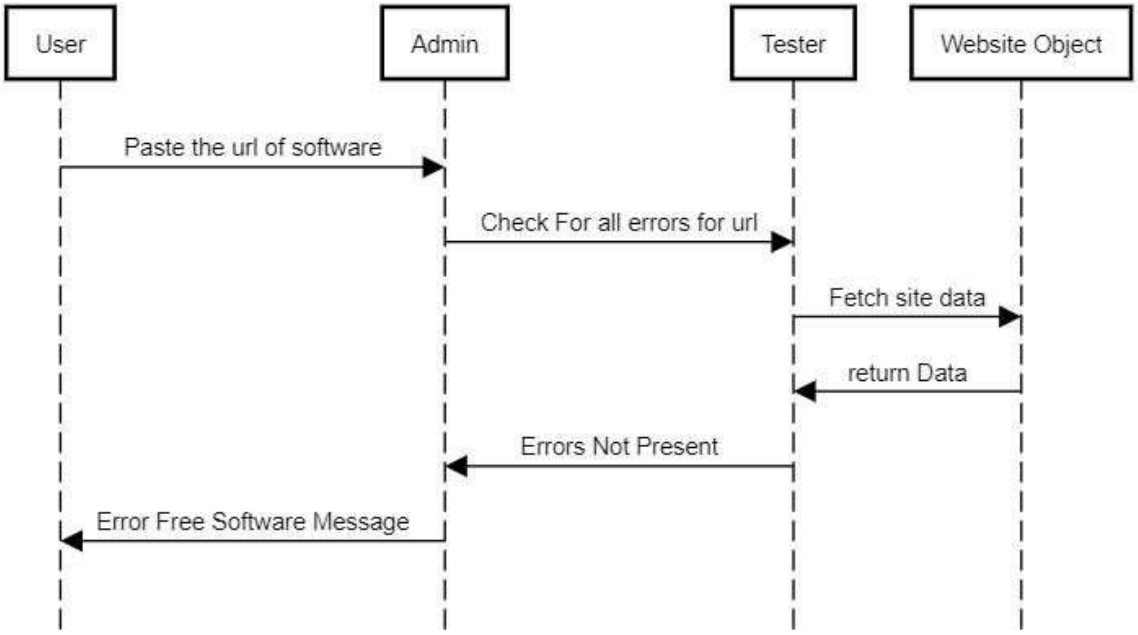
4.1.8 Sequence & Collaboration Diagrams:

4.1.7.1 Configuration Module:

SEQUENCE DIAGRAM-

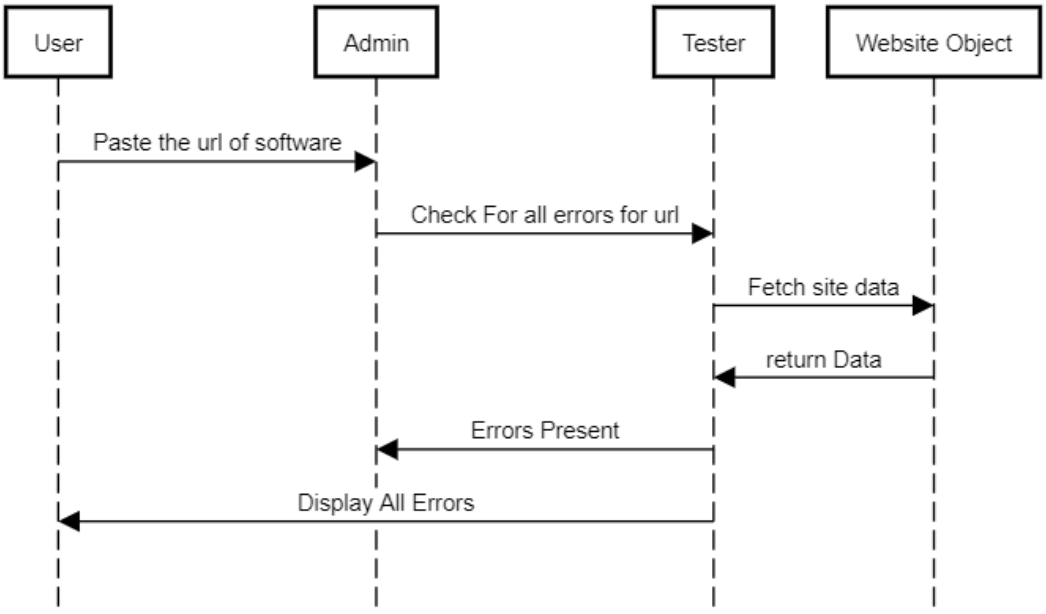
Bug Free Sequence Diagram

This sequence diagram shows the data flow of the Software when no error is detected in the web app.

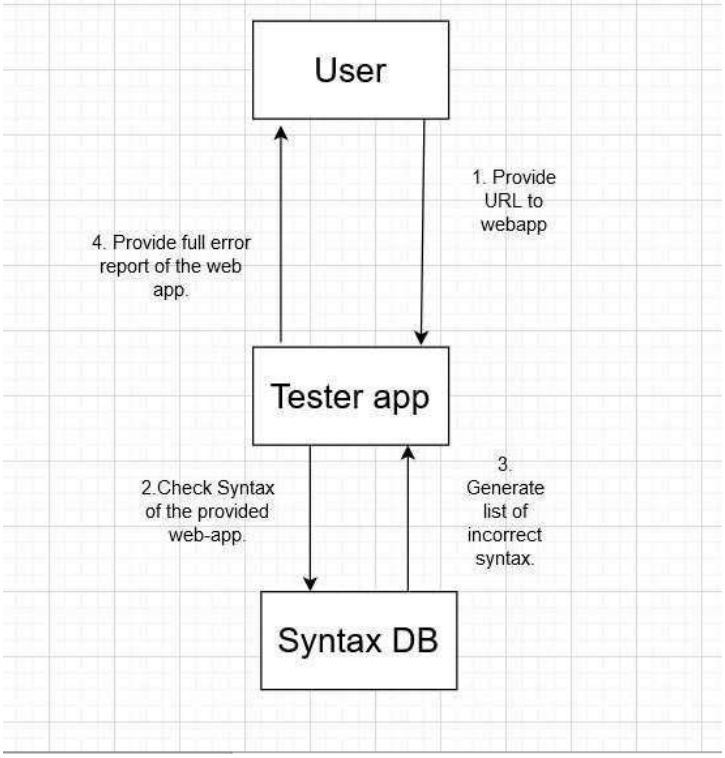


Error Sequence Diagram

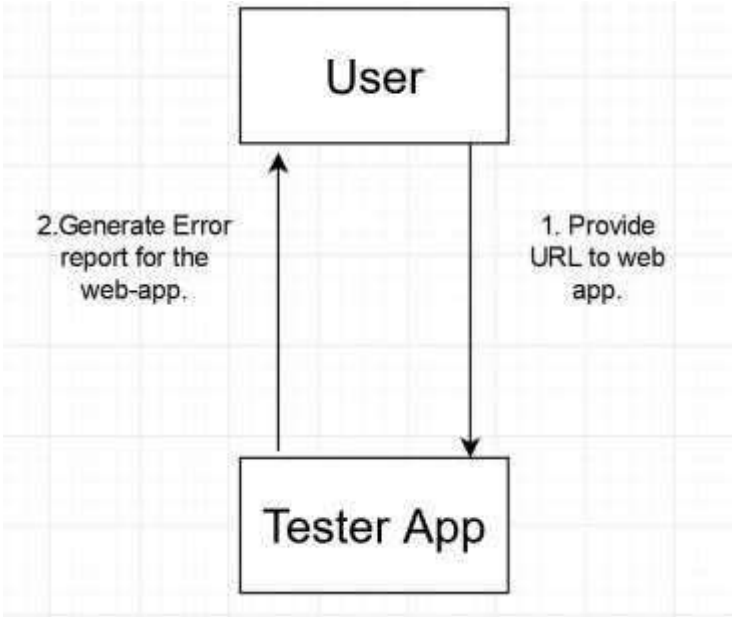
This sequence diagram shows the data flow of the Software when no error is detected in the web app.



Collaboration diagram for checking syntax errors



Collaboration diagram for generating test report



4.2 CODES AND STANDARDS:

4.3 CONSTRAINTS AND TRADEOFFS:

4.3.1 CONSTRAINTS

Table for Constraints –

Constraints	Description
Programming Language	HTML, CSS, JavaScript and Python Django
OS Support	This project cannot be considered as a very platform friendly software due to its preliminary Mac/Linux OS support.
Interruptions and errors	If input is an non-existent link then

4.3.2 ALTERNATIVES-

As a whole tool, there might not be a proper alternative, the only thing that comes to this is Selenium but Selenium is a web driver not a stand-alone tool. Web App tester is a one for all testing platform for all your web apps.

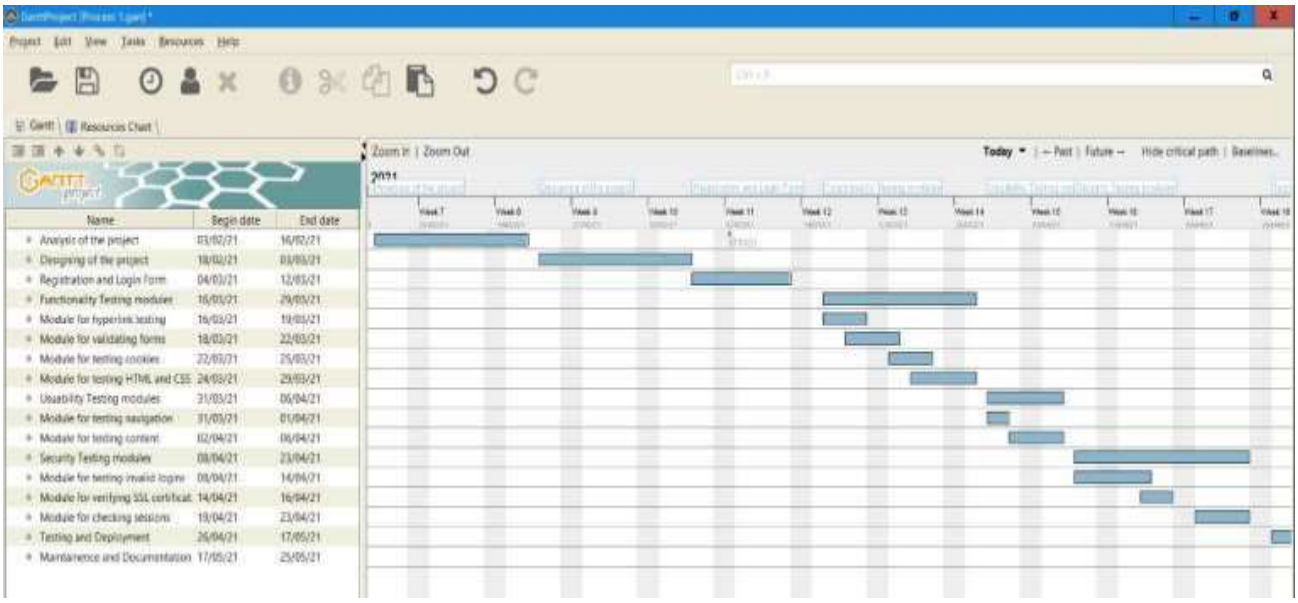
4.3.3 TRADEOFFS-

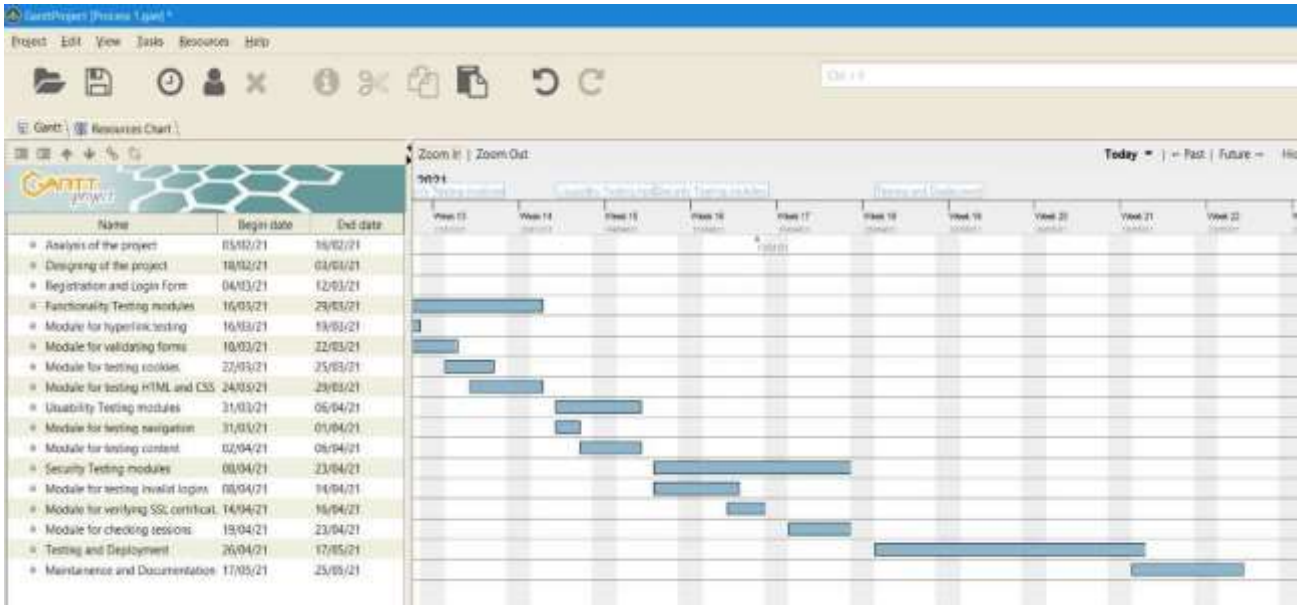
The idea for Web App Tester is inspired by Selenium web driver but not based on it. The biggest trade-off that comes from this is that developers/maintainers do not have the access wide range of facilities provided by Selenium. For instance, Selenium allows to emulate user's interaction with the widgets and buttons on the site and by that way developer can better test abilities like useability of web application. But Selenium is a web driver which can only be used to build upon. While Web App tester is a stand-alone tool for the web developers.

5. SCHEDULES, TASKS AND MILESTONES

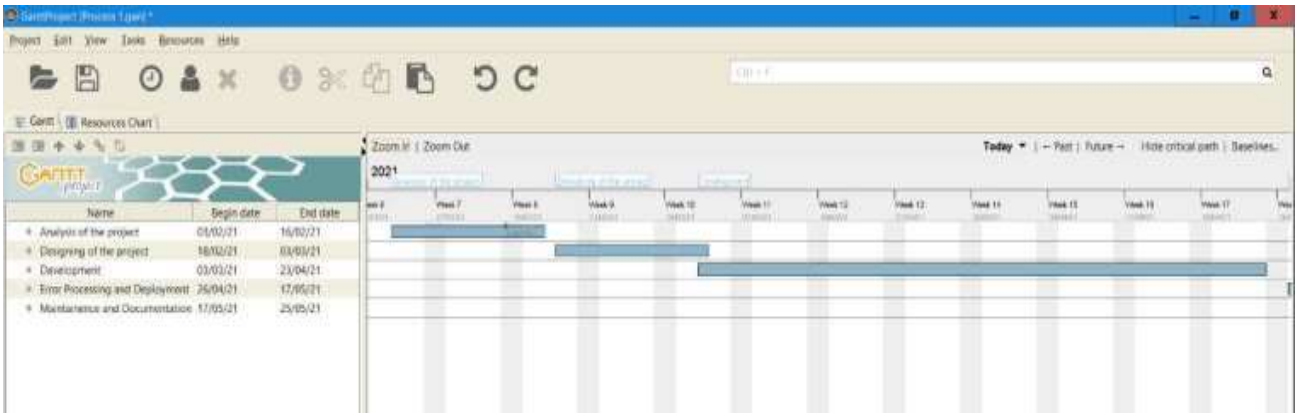
5.1 GANTT CHART:

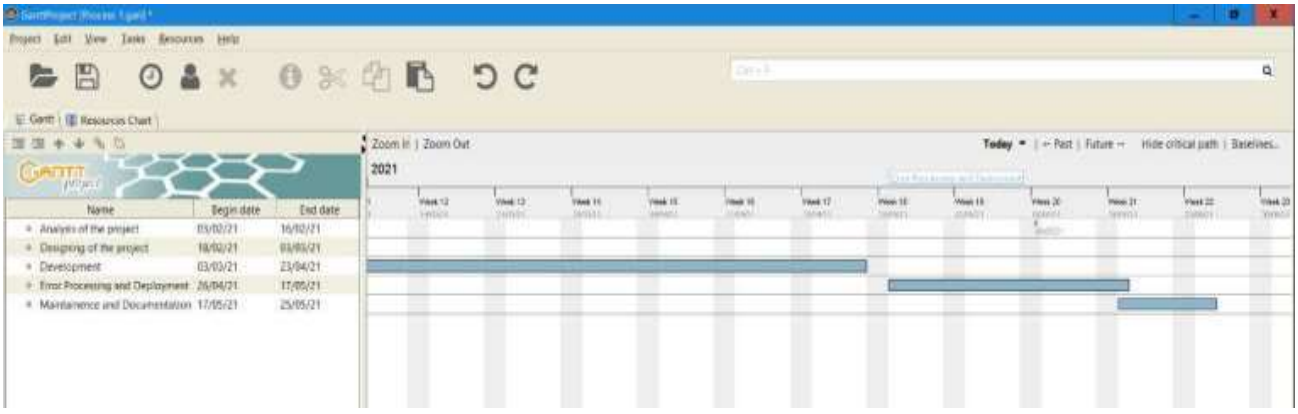
Product Gantt Chart:





Process Gantt Chart:

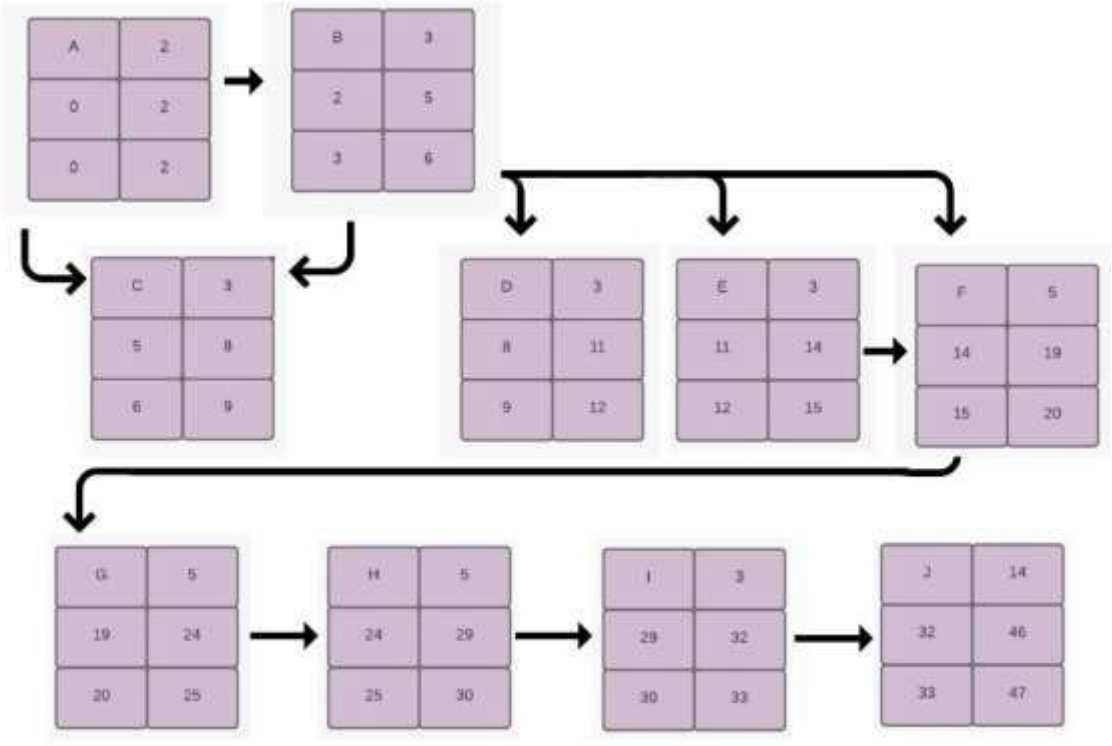




5.2 TIMELINE NETWORK:

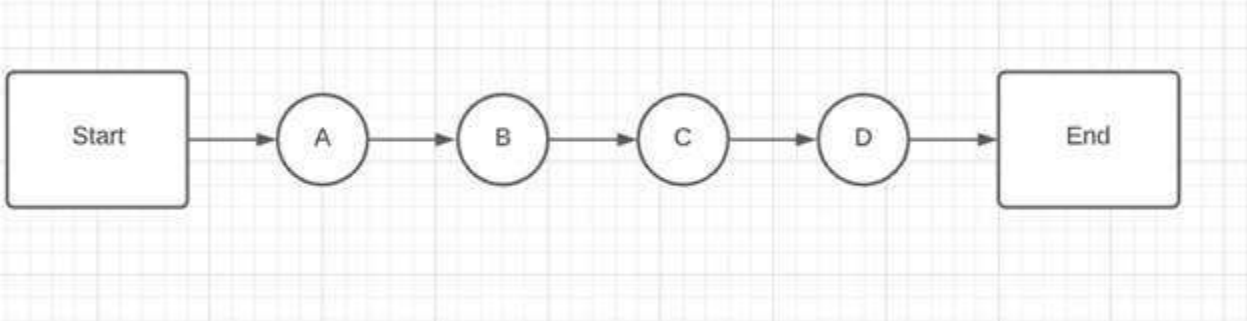
5.3 ACTIVITY NETWORK DIAGRAM:

Product Based:



TASK	LABEL	PREDECESSOR	DURATION
Designing the BASIC website	A	.....	2 DAYS
Implementing Login and Registration Modules	B	A	3 DAYS
Testing All Hyperlinks	C	A,B	3 DAYS
Test all forms and Cookies	D	A,B	3 DAYS
Test the frontend of system(HTML and CSS)	E	A,B	3 DAYS
Test Navigation AND site content	F	A,B,E	5 DAYS
Test unauthorised access	G	A,B,C,D,E,F	5 DAYS
Check for validity of Sessions	H	A,B,C,D,E,F,G	5 DAYS
Test SSL certificates	I	A,B,C,D,E,F,G,H	3 DAYS
Testing AND Integrating all our Modules	J	A,B,C,D,E,F,G,H,I	14 DAYS

Process Based:



TASK	LABEL	PREDECESSOR	DURATION
Planning and Scheduling	A	....	5 DAYS
Design	B	A	5 DAYS
Implementation of features	C	B	27 DAYS
Software Testing	D	C	14 DAYS

6. PROJECT CODE:

Script.py

```
from scrapy.spiders import CrawlSpider, Rule
from scrapy.linkextractors import LinkExtractor
from scrapy.selector import Selector
from scrapy.item import Item, Field
import os
from scrapyscript import Job, Processor
import scrapy
from scrapy.crawler import CrawlerProcess
import sys



print(sys.argv,'-----------------------')
start_urls = [sys.argv[1].split('_')[0]]
file_path=sys.argv[1].split('_')[1]
target_domains = [start_urls[0].split('//')[1]]
class MyItems(Item):
        referer =Field() # where the link is extracted
        response= Field() # url that was requested
        status = Field() # status code received


class MySpider(CrawlSpider):
        name = "test-crawler"
        start_urls=start_urls
        handle_httpstatus_list = [404,410,301,500]

        custom_settings = {
                'CONCURRENT_REQUESTS': 2, # only 2 requests at the same time
                'DOWNLOAD_DELAY': 0.5 # delay between requests
        }

        rules = [
                Rule(
                        LinkExtractor( allow_domains=target_domains,
deny=('patterToBeExcluded'), unique=('Yes')),
                        callback='parse_my_url', # method that will be called for each
request
```

```
                follow=True),
            # crawl external links but don't follow them
            Rule(
                LinkExtractor(
allow=(''),deny=("patterToBeExcluded"),unique=('Yes')),
                callback='parse_my_url',
                follow=False
            )
        ]


        def parse_my_url(self, response):

          report_if = [404,500,410]
          if response.status in report_if:
                item = MyItems()
                item['referer'] = response.request.headers.get('Referer', None)
                item['status'] = response.status
                item['response']= response.url
                yield item
          yield None # if the response did not match return empty



def spider_results():


        process = CrawlerProcess({
            'USER_AGENT': 'Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1)',
            'FEED_FORMAT': 'csv',
            'FEED_URI': 'media/links/data.csv'
            })
        process.crawl(MySpider)
        process.start(stop_after_crawl=True)  # the script will block here until the crawling
is finished

        print('------------------------------------------------------------')
        os.rename('media/links/data.csv',file_path)
        return True



spider_results()
```

Script.py

```
#!/usr/bin/env python
"""Django's command-line utility for administrative tasks."""
import os
import sys


def main():
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'web_tester.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)


if __name__ == '__main__':
    main()
```

app.py

```
import watchdog.events
import watchdog.observers
import time
from pathlib import Path
from test import processor
from watchdog.events import LoggingEventHandler
class Handler(LoggingEventHandler):

    def on_created(self, event):
        print("Watchdog received created event - % s." % event.src_path)

        processor(event.src_path)

        #self.observer.stop() # stop watching
        #self.callback()
        # Event is created, Now process for model
```

```
if __name__ == "__main__":
    src_path = 'media/master'
    event_handler = Handler()
    observer = watchdog.observers.Observer()
    observer.schedule(event_handler, path=src_path, recursive=True)
    observer.start()
    print('observer initiated')
    try:
        while True:
            time.sleep(1)
    except KeyboardInterrupt:
        observer.stop()
        print('KeyboardInterrupt')
    observer.join()
```

settings.py

```
"""
Django settings for web_tester project.

Generated by 'django-admin startproject' using Django 2.2.7.

For more information on this file, see
https://docs.djangoproject.com/en/2.2/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/2.2/ref/settings/
"""

import os

# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/2.2/howto/deployment/checklist/
```

```
# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'akdfmyblk0qp%qzu_0t42kl4#n@_$=_@0sqq02*agpnqeuc1u+'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []


# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'validator',
    'django_libsass_compass_mixins'
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'web_tester.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': ['templates'],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
```

```
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'web_tester.wsgi.application'


# Database
# https://docs.djangoproject.com/en/2.2/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}


# Password validation
# https://docs.djangoproject.com/en/2.2/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]


# Internationalization
# https://docs.djangoproject.com/en/2.2/topics/i18n/
```

```
LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_L10N = True

USE_TZ = True


# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/2.2/howto/static-files/

STATIC_URL = '/static/'

STATICFILES_DIRS = (
   os.path.join(BASE_DIR, "static"),
)

MEDIA_ROOT  = os.path.join(BASE_DIR, 'media')
MEDIA_URL = '/media/'
```

urls.py

```
"""web_tester URL Configuration

The `urlpatterns` list routes URLs to views. For more information please see:
   https://docs.djangoproject.com/en/2.2/topics/http/urls/
Examples:
Function views
   1. Add an import:  from my_app import views
   2. Add a URL to urlpatterns:  path('', views.home, name='home')
Class-based views
   1. Add an import:  from other_app.views import Home
   2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
Including another URLconf
   1. Import the include() function: from django.urls import include, path
   2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
"""
from django.contrib import admin
from django.urls import path
from validator import views
```

```
from django.conf import settings
from django.conf.urls.static import static
app_name = "validator"
urlpatterns = [
    path('admin/', admin.site.urls),
    path('validate/', views.validate ,name='validate'),
    path('validate_1/', views.validate_1,name='validate_1'),
    path('', views.index,name='index'),
    path('signup/', views.signup,name='signup'),
    path('login/', views.login,name='login'),
    path('logout/', views.logout_view,name='logout'),
] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

wsgi.py

```
"""
WSGI config for web_tester project.

It exposes the WSGI callable as a module-level variable named ``application``.

For more information on this file, see
https://docs.djangoproject.com/en/2.2/howto/deployment/wsgi/
"""

import os

from django.core.wsgi import get_wsgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'web_tester.settings')

application = get_wsgi_application()
```

views.py

```
from django.shortcuts import render
import requests,json
from django import template
from scrapy.spiders import CrawlSpider, Rule
from scrapy.linkextractors import LinkExtractor
from scrapy.selector import Selector
from scrapy.item import Item, Field
import os
from django.contrib.auth.models import User
import scrapy
```

```
from scrapy.crawler import CrawlerProcess
# Create your views here.
import subprocess
import os
from django.contrib.auth import login as auth_login
from django.contrib.auth import login, authenticate, logout
from random import randint
from django.http import HttpResponseRedirect, HttpResponse
from django.shortcuts import render, reverse, redirect
register = template.Library()


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

def login(request):
	if not request.user.is_authenticated:
		if request.method == 'POST':

			username = request.POST['username']
			password = request.POST['password']
			user = authenticate(request, username=username,
password=password)
			if user:
				auth_login(request,user)
				return render(request, 'search.html')
	return render(request, 'login.html')
def signup(request):
	if not request.user.is_authenticated:
		if request.method == 'POST':

			username = request.POST['username']
			first_name = request.POST['fname']
			last_name = request.POST['lname']
			email = request.POST['email']
			password=request.POST['password']

	user_obj=User.objects.create_user(username=username,first_name=first_name,last
_name=last_name,email=email,password=password)
			user_obj.save()

			auth_login(request, user_obj)
			return render(request,'search.html')
```

```
                context=dict()
                auser=User.objects.all()
                context['auser']=auser
                return render(request,'login.html',context)
        return render(request,'search.html')
def logout_view(request):
        logout(request)
        return HttpResponseRedirect(reverse('index'))

def validate(request):
        if request.method=='POST':
                if request.user.is_authenticated:
                        requested_url=request.POST.get('url')

        #requested_url='http://rinkworks.com/slapdash/broken.shtml'#'w3schools.com'
                        payload = {'uri': requested_url,'output':'json'}
                        r=requests.get('http://validator.w3.org/check',payload)
                        context=dict(r.json())
                        r2=requests.get('http://jigsaw.w3.org/css-
validator/validator',payload)

                        if 'http' not in requested_url:
                                link='http://'+requested_url
                        else:
                                link=requested_url

                        #link='http://rinkworks.com/slapdash/broken.shtml'
                        try:
                                os.remove("media/master/master_link.txt")
                        except OSError:
                                pass
                        f = open("media/master/master_link.txt", "w")
                        val=randint(1000,9999)
                        f.write("{0}_media/links/{1}.csv".format(link,str(val)))
                        f.close()
                        context['links']="/media/links/{0}.csv".format(str(val))
                        #context['loop_cycle']=range(0,10)
                        if 'errors' in dict(r2.json())['cssvalidation']:
                                context['css_error']=dict(r2.json())['cssvalidation']['errors']

        context['css_source']=dict(r2.json())['cssvalidation']['errors'][0]['source']
                        context['qlink']=link
                        request.session['val_data']=context
```

```
                return render(request,'validate_new.html',context)
        else:
                return HttpResponseRedirect(reverse('login'))


def validate_1(request):

        context=request.session['val_data'] #"media/links/{0}.csv".format(str(val))
        return render(request,'validate.html',context)


class MyItems(Item):
        referer =Field() # where the link is extracted
        response= Field() # url that was requested
        status = Field() # status code received


class MySpider(CrawlSpider):
        name = "test-crawler"
        target_domains = ["rinkworks.com/slapdash/broken.shtml"] # list of domains that
will be allowed to be crawled
        start_urls = ["http://rinkworks.com/slapdash/broken.shtml"] # list of starting urls
for the crawler
        handle_httpstatus_list = [404,410,301,500] # only 200 by default. you can add
more status to list

        # Throttle crawl speed to prevent hitting site too hard
        custom_settings = {
                'CONCURRENT_REQUESTS': 2, # only 2 requests at the same time
                'DOWNLOAD_DELAY': 0.5 # delay between requests
        }

        rules = [
                Rule(
                        LinkExtractor( allow_domains=target_domains,
deny=('patterToBeExcluded'), unique=('Yes')),
                        callback='parse_my_url', # method that will be called for each
request
                        follow=True),
                # crawl external links but don't follow them
                Rule(
```

```
                    LinkExtractor(
allow=(''),deny=(''patterToBeExcluded''),unique=('Yes')),
                    callback='parse_my_url',
                    follow=False
                )
        ]


        def parse_my_url(self, response):
          # list of response codes that we want to include on the report, we know that 404
          report_if = [404,500,410,301]
          if response.status in report_if: # if the response matches then creates a MyItem
                item = MyItems()
                item['referer'] = response.request.headers.get('Referer', None)
                item['status'] = response.status
                item['response']= response.url
                yield item
          yield None # if the response did not match return empty
```

Frontend

login.html

```
<html>
<head>
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
  <link rel="stylesheet" href="/static/login.css">

</head>
<body>
<div class="form-wrap">
   <div class="tabs">
     <h3 class="signup-tab"><a class="active" href="#signup-tab-content">Sign
Up</a></h3>
     <h3 class="login-tab"><a href="#login-tab-content">Login</a></h3>
   </div><!--.tabs-->

   <div class="tabs-content">
     <div id="signup-tab-content" class="active">
       <form class="signup-form" action="/signup/" method="post">
         {% csrf_token %}
         <input type="email" name="email" class="input" id="user_email"
autocomplete="off" placeholder="Email">
```

```
        <input type="text" class="input" name="username" id="user_name"
autocomplete="off" placeholder="Username">
        <input type="text" class="input" name="fname" id="user_name"
autocomplete="off" placeholder="First Name">
        <input type="text" class="input" name="lname" id="user_name"
autocomplete="off" placeholder="Last Name">
        <input type="password" class="input" name="password" id="user_pass"
autocomplete="off" placeholder="Password">
        <input type="submit" class="button" value="Sign Up">
       </form><!--.login-form-->
       <div class="help-text">
        <p>By signing up, you agree to our</p>
        <p><a href="#">Terms of service</a></p>
       </div><!--.help-text-->
     </div><!--.signup-tab-content-->

     <div id="login-tab-content">
       <form class="login-form" action="/login/" method="post">
        {% csrf_token %}
        <input type="text" class="input" name="username" id="user_login"
autocomplete="off" placeholder="Username">
        <input type="password" class="input" name="password" id="user_pass"
autocomplete="off" placeholder="Password">

        <label for="remember_me">Remember me</label>

        <input type="submit" class="button" value="Login">
       </form><!--.login-form-->
       <div class="help-text">
        <p><a href="#">Forget your password?</a></p>
       </div><!--.help-text-->
     </div><!--.login-tab-content-->
   </div><!--.tabs-content-->
  </div><!--.form-wrap-->

</body>

<script type="text/javascript">
jQuery(document).ready(function($) {
  tab = $('.tabs h3 a');

  tab.on('click', function(event) {
    event.preventDefault();
```

```
    tab.removeClass('active');
    $(this).addClass('active');

    tab_content = $(this).attr('href');
    $('div[id$="tab-content"]').removeClass('active');
    $(tab_content).addClass('active');
  });
});
</script>
</html>
```

search.html

```
<html>
 <head>

 <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.3/css/all.min.css"/>
 <style>


     * {
  box-sizing: border-box;
}

.body {
  background-image: linear-gradient(90deg, #ff5fa2, #be20dd);
  display: flex;
  align-items: center;
  justify-content: center;
  height: 100vh;
  margin: 0;

}

.search {
  position: relative;
  height: 50px;
}

.search .input {
  background-color: rgba(241, 197, 227, 0.986);
  border: 0;
```

```
  font-size: 18px;
  padding: 15px;
  height: 50px;
  width: 50px;
  transition: width 0.3s ease;
}

.btn {
  background-color: rgba(241, 197, 227, 0.986);
  border: 0;
  cursor: pointer;
  font-size: 24px;
  position: absolute;
  top: 0;
  left: 0;
  height: 50px;
  width: 50px;
  transition: transform 0.3s ease;
}

.btn:focus,
.input:focus {
  outline: none;
}

.search.active .input {
  width: 300px;
}

.search.active .btn {
  transform: translateX(298px);
}

.wrapper {
    margin: 0 auto;
    width: 90%;
}

header {
    width: 100%;
    max-width: 100%;
    position: fixed;
    z-index: 999;
```

```
    background: rgba(50,50,50,.95);
    box-shadow: 0 10px 20px rgba(50,50,50,.95);
}

nav {
    margin-top: 10px;
    -webkit-border-radius: 5px;
    -moz-border-radius: 5px;
    border-radius: 5px;
}

nav a {
    font: 1.2em/1em sans-serif;
    display: inline-block;
    padding: 10px 15px;
    text-decoration: none;
    color: white;
}

 </style>
    </head>
<body>
        <header>
      <div class="wrapper">
          <nav>
            <a href="#">Home</a>

            {% if not request.user.is_authenticated %}
            <a href="login">Login</a>
            <a href="signup">Sign Up</a>
            {% endif %}
            {% if request.user.is_authenticated %}
            <a href="/logout/">Logout</a>
            <a href="#" style="float:right">{{request.user.first_name}}
{{request.user.last_name}}</a>
            {% endif %}
          </nav>
      </div>
    </header>
    <div class="body">
    <div class="search active">
        <form action="{% url 'validate'  %}" method="post">
            {% csrf_token %}
```

```
    <input type="text" name="url" class="input" placeholder="Search Here ..." />
    <button type="submit" class="btn" >
      <i class="fas fa-search"></i>
    </button>
  </form>
   </div>
</div>
   <script src="./script.js"></script>
  </body>
  <script>


const search = document.querySelector(".search");
const btn = document.querySelector(".btn");
const input = document.querySelector(".input");

btn.addEventListener("click", () => {
search.classList.toggle("active");
input.focus();
});

  </script>
</html>
```

validate.html

```
<!DOCTYPE html>
{%load static%}
{% load my_filter %}
<html lang="en">
  <head>
    <link href="/static/icon.png" rel="icon">
    <link href="/static/style.css" rel="stylesheet">
    <title>Showing results for https://www.w3schools.com/</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">


    <style>
        table {
            border-collapse: collapse;
            border: 2px black solid;
            font: 12px sans-serif;
        }
```

```
        td {
            border: 1px black solid;
            padding: 5px;
        }
    </style>
</head>



<div id="console" >
            <div id="errors">
    <h3>We found the following Broken Links </h3>

    <script type="text/javascript"charset="utf-8">
        var link="{{links}}"

        d3.text(link ,    function(data) {
            var parsedCSV = d3.csv.parseRows(data);

            var container = d3.select("body")
                .append("table")

                .selectAll("tr")
                    .data(parsedCSV).enter()
                    .append("tr")

                .selectAll("td")
                    .data(function(d) { return d; }).enter()
                    .append("td")
                    .text(function(d) { return d; });
        });

        $.ajax({
                                    url:'{{links}}',
                                    type:'HEAD',
                                    error: function()
                                    {

                                    },
                                    success: function()
                        {clearInterval(myTimer);
```

```
                }
                });

        </script>
         </div>


        </div>

    </body>
</html>
```

Validate_new.html

```
<!DOCTYPE html>
{%load static%}
{% load my_filter %}
<html lang="en">
  <head>
    <link href="/static/icon.png" rel="icon">
    <link href="/static/style.css" rel="stylesheet">
    <link type="text/css" rel="stylesheet" href="/static/base.css" />
     <link type="text/css" rel="stylesheet" href="/static/results.css" />
    <title>Showing results for https://www.w3schools.com/</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">


    <style>
        table {
            border-collapse: collapse;
            border: 2px black solid;
            font: 12px sans-serif;
        }

        td {
            border: 1px black solid;
            padding: 5px;
        }


.wrapper {
   margin: 0 auto;
```

```
    width: 90%;
}

header {
    width: 100%;
    max-width: 100%;
    position: fixed;
    z-index: 999;
    background: rgba(50,50,50,.95);
    box-shadow: 0 10px 20px rgba(50,50,50,.95);
}

nav {
    margin-top: 10px;
    -webkit-border-radius: 5px;
    -moz-border-radius: 5px;
    border-radius: 5px;
}

nav a {
    font: 1.2em/1em sans-serif;
    display: inline-block;
    padding: 10px 15px;
    text-decoration: none;
    color: white;
}
      </style>
  </head>
  <body>
    <header>
    <div class="wrapper">
       <nav>
         <a href="#">Home</a>

         {% if not request.user.is_authenticated %}
         <a href="login">Login</a>
         <a href="signup">Sign Up</a>
         {% endif %}
         {% if request.user.is_authenticated %}
         <a href="/logout/">Logout</a>
         <a href="#" style="float:right">{{request.user.first_name}}
{{request.user.last_name}}</a>
         {% endif %}
```

```
        </nav>
      </div>
    </header>


      <h2 id="top">Showing results for {{qlink}}</h2>
      <script src="{% static 'script.js' %}"></script><script
src="data:text/javascript,var%20_paq%3Dwindow._paq%7C%7C%5B%5D%3B_paq.pus
h%28%5B%22trackPageView%22%5D%29%2C_paq.push%28%5B%22enableLinkTrac
king%22%5D%29%2Cfunction%28%29%7Bvar%20e%3D%22https%3A%2F%2Fwww.
w3.org%2Fanalytics%2Fpiwik%2F%22%3B_paq.push%28%5B%22setTrackerUrl%22%
2Ce%2B%22matomo.php%22%5D%29%2C_paq.push%28%5B%22setSiteId%22%2C%
22444%22%5D%29%3Bvar%20a%3Ddocument%2Ct%3Da.createElement%28%22script
%22%29%2Cp%3Da.getElementsByTagName%28%22script%22%29%5B0%5D%3Bt.ty
pe%3D%22text%2Fjavascript%22%2Ct.async%3D%210%2Ct.defer%3D%210%2Ct.src
%3De%2B%22matomo.js%22%2Cp.parentNode.insertBefore%28t%2Cp%29%7D%28%
29%3B"></script>
      <div id="errors" style="padding: 3%">
      <h3>We found the following errors In HTML </h3>
   </div>
      <div id="results">

        <ol>
          {% for res in messages%}
          {% if res.type == 'error' %}
                  <li class="error">
                    <p><strong>Error</strong>: <span>{{res.message}}</span></p>
                    <p class="location">At line <span class="last-
line">{{res.lastLine}}</span>, column <span class="last-
col">{{res.lastColumn}}</span></p>
                    <p class="extract" ><code><span class="lf" title="Line break"
></span>{{res.extract|replace_newline}}</span></code></p>
                  </li>
          {% endif %}
          {% if res.type == 'info' %}
                  <li class="info warning">
                    <p><strong>Warning</strong>: <span>{{res.message}}</span></p>
                    <p class="location">From line <span class="first-
line">{{res.lastLine}}</span>, column <span class="first-
col">{{res.firstColumn}}</span>; to line <span class="last-
line">{{res.lastLine}}</span>, column <span class="last-
col">{{res.lastColumn}}</span></p>
```

```
                    <p class="extract" ><code><span class="lf" title="Line break"
></span>{{res.extract|replace_newline}}</span></code></p>
                </li>
          {% endif %}

          {% endfor %}

          <script src="http://d3js.org/d3.v3.min.js"></script>
                <script type="text/javascript" src="{% static 'js/jquery.js' %}"></script>


        </ol>
        {% if css_error %}
<!-------------------------------------------------------------------------------->
          <div id="errors">
          <h3>We found the following errors In CSS </h3>
               <div class="error-section-all">

                                                                  <div class="error-section">
                                    <h4>URI : <a
href="{{css_source}}">{{css_source}}</a></h4>
                                    <table>
                                       {% for css_err in css_error%}
                                          <tr class="error" width="20">
                         <td class="linenumber" title="Line {{line}}">{{css_err.line}}</td>
                         <td class="codeContext"> {{css_err.type}} </td>
                         <td class="parse-error">

                                    <span class="exp">
                              {{css_err.message|replace_newline}}
                            </span>
                          </td>
                         </tr>
                         {% endfor %}
          </table>
               <!--end of individual error section-->
               </div>

               </div>
       </div>
<!-------------------------------------------------------------------------------------------->
      {% endif %}
```

```
        <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>

   <script>

function autorefresh() {
       // auto refresh page after 1 second
       myTimer =setInterval('refreshPage()', 5000);
}

function refreshPage() {
       $.ajax({
       url: '{% url 'validate_1' %}',
       success: function(data) {
       $('#console').html(data);
       }
    });
}
</script>
.
.
.
<script>autorefresh()


</script>

<div id="console" >

       <script type="text/javascript"charset="utf-8">
          var link="{{links}}"

          d3.text(link ,     function(data) {
             var parsedCSV = d3.csv.parseRows(data);

             var container = d3.select("body")
                .append("table")

                .selectAll("tr")
                   .data(parsedCSV).enter()
                   .append("tr")

                .selectAll("td")
```

```
                .data(function(d) { return d; }).enter()
                .append("td")
                .text(function(d) { return d; });
                });




        </script>
        </div>
      <img style="text-align: center;"
src="https://miro.medium.com/max/1600/1*CsJ05WEGfunYMLGfsT2sXA.gif"
class="center"></img>
   </body>
</html>
```

6. PROJECT DEMONSTRATION:

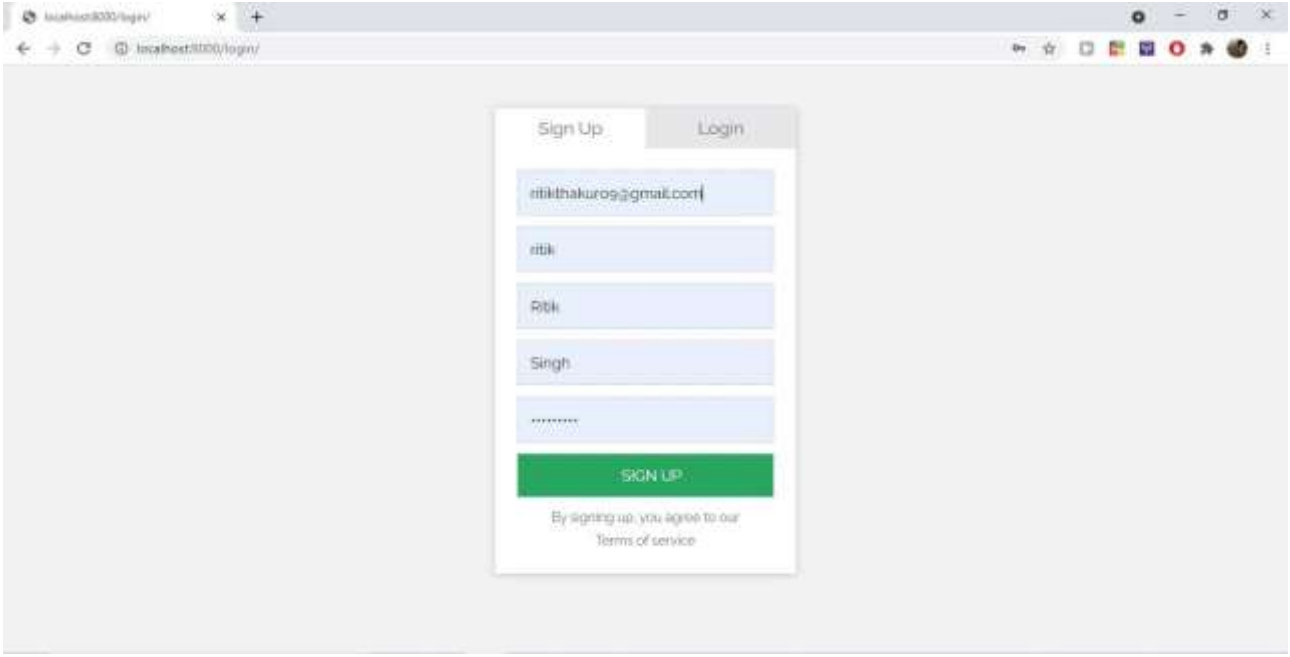


Figure 21: Signup Page

The new user needs to create an account before testing his Web application.

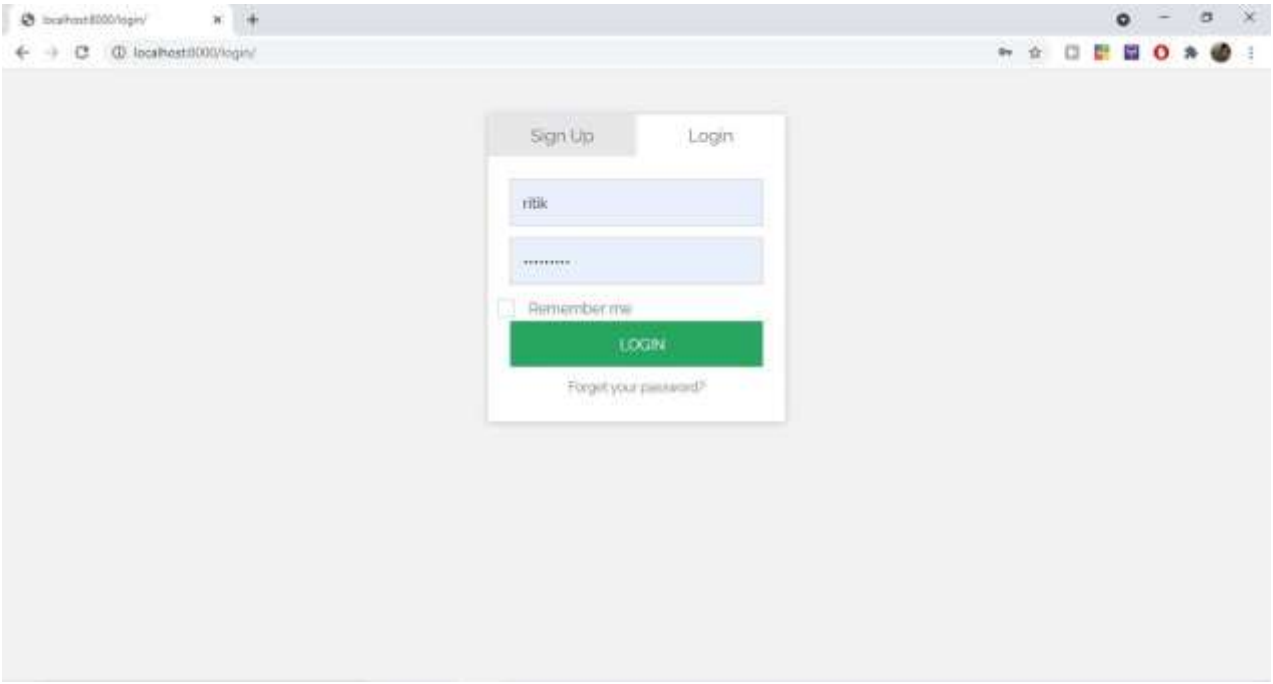


Figure 22: Login Page

Old or newly signed users need to sign up before testing their Web application.

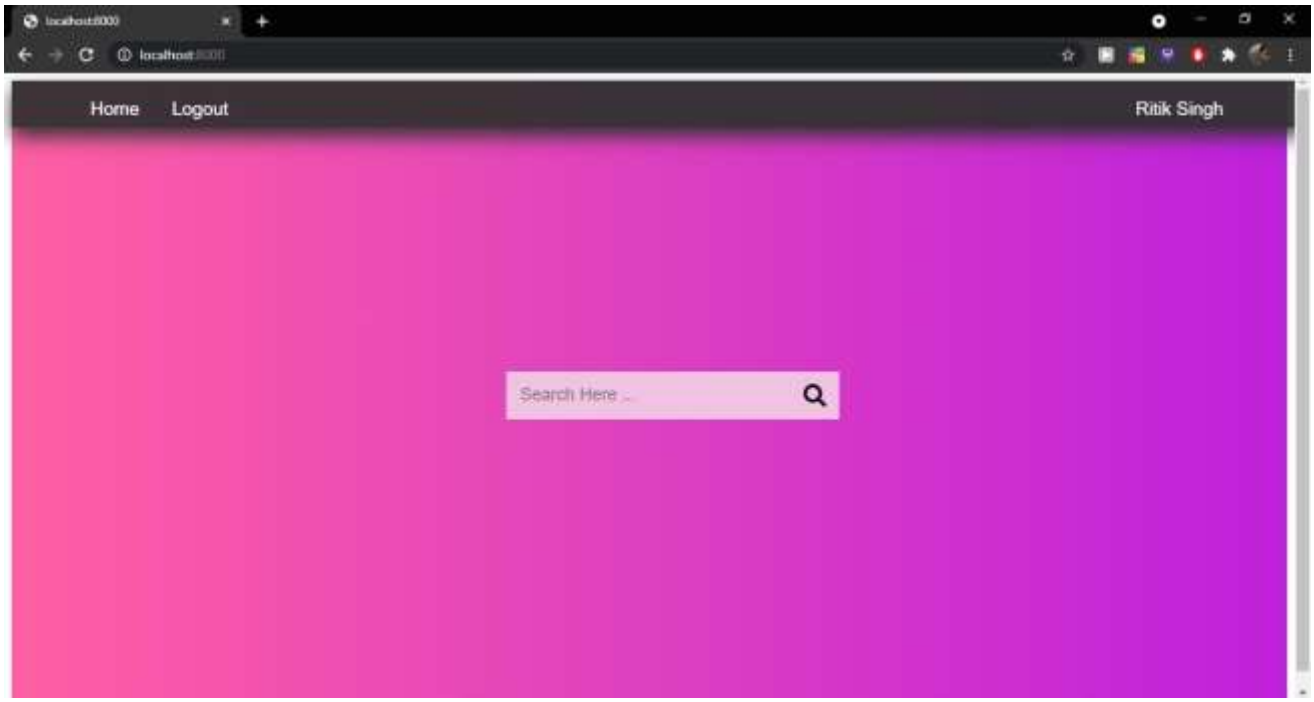


Figure 23: Homepage

The user needs to enter the link of his Web application in order to test it.

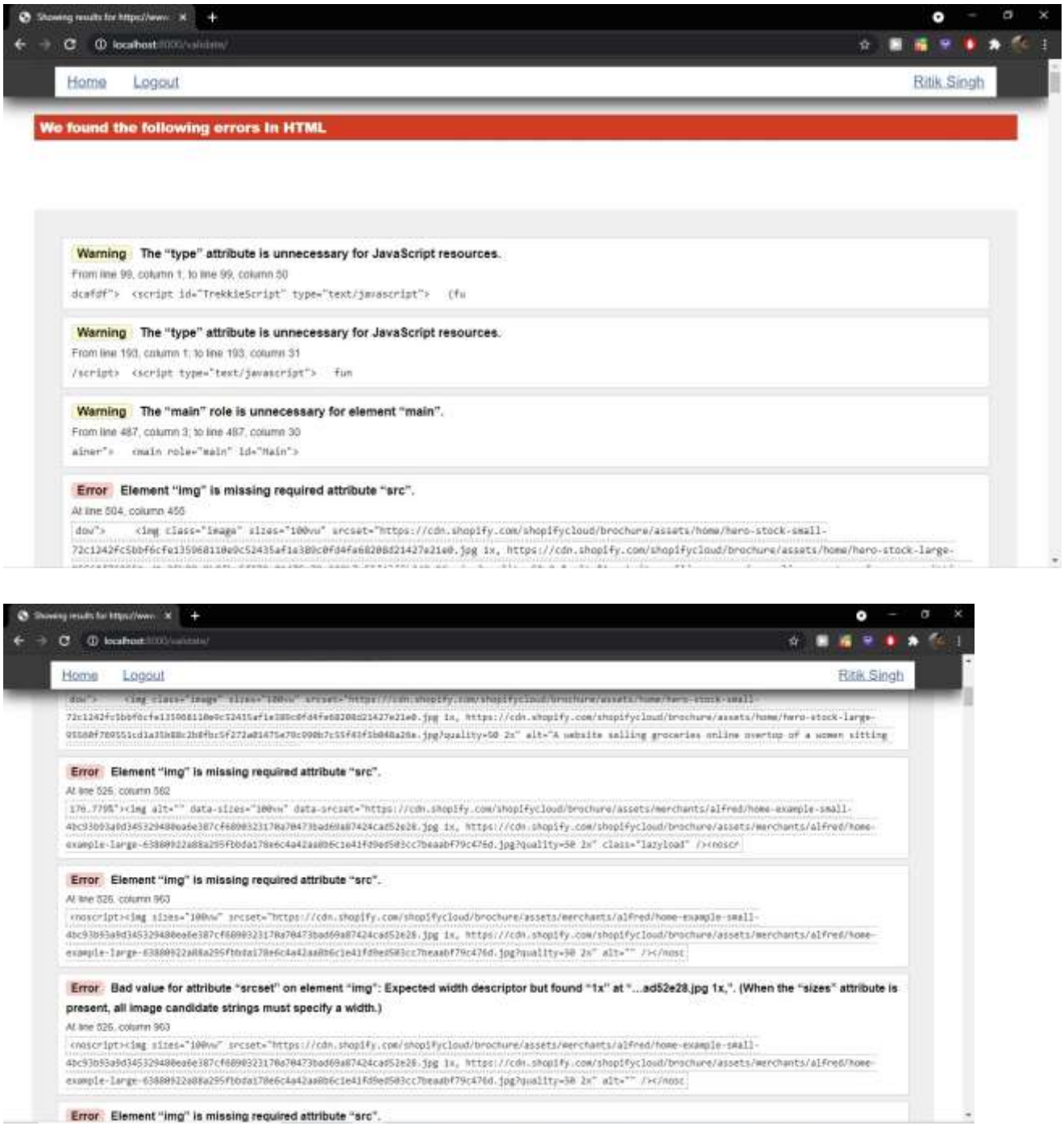


Figure 24: HTML Errors Page

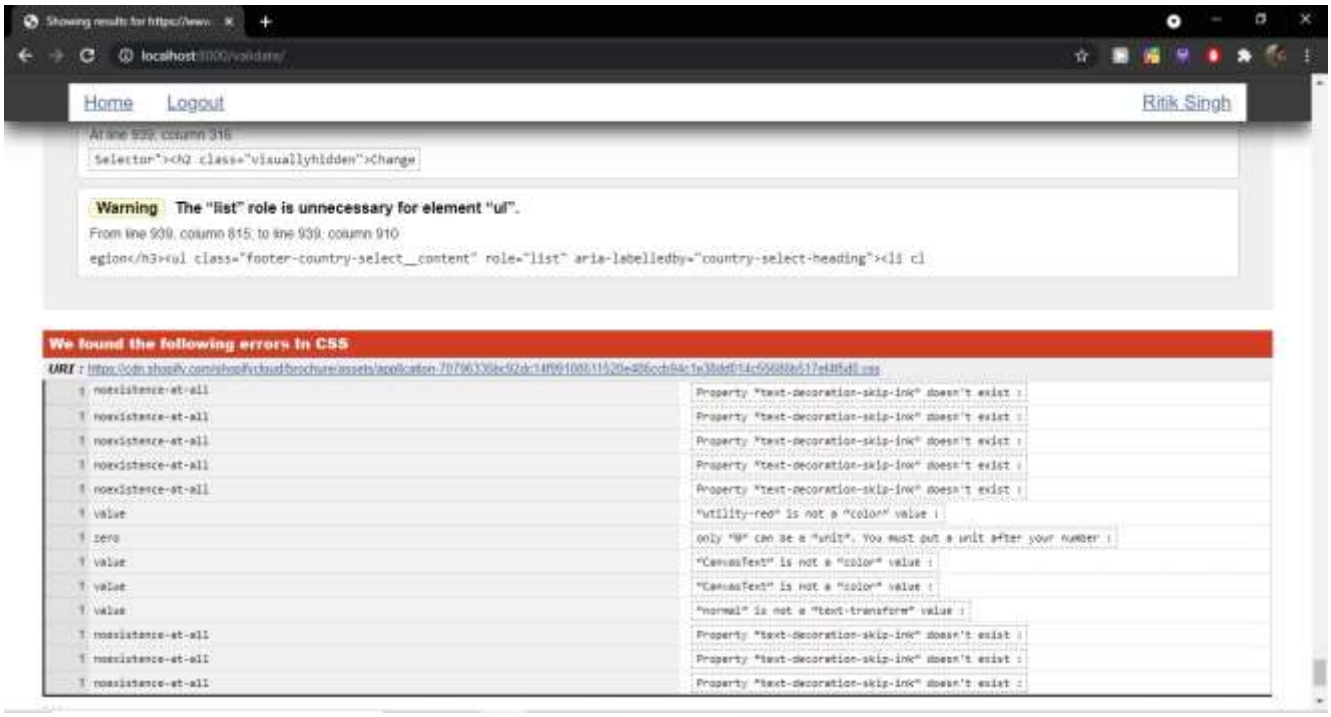


Figure 25: CSS Errors Page

Figure 26: File containing the CSS Error

7. RESULTS AND DISCUSSION:

We have successfully created an autonomous Web App Tester to help the web App developers. We were successful in imparting a lot of functionalities into our project. Web App Tester will save web developers a lot of time as some of the tedious tasks like testing for functioning links will be automated.

We were not able to make an enterprise level tool because our limited resources at our hands and our inexperience in this technology. Future work includes adding Artificial Intelligence in to our project to catching not so obvious errors in the code and also to test navigation of the web app.