Mini Project Report on

OneLyf

Gaurav U D (16IT113)

Abhilash V (16IT201)

Shrinivas V Shanbhag (16IT244)

Under the Guidance of,

Dr Biju R Mohan

Ms Raksha Nadgir

Department of Information Technology, NITK Surathkal

Date of Submission: April 9, 2019

in partial fulfillment for the award of the degree

of

Bachelor of Technology

In

Information Technology

At

Department of Information Technology

National Institute of Technology Karnataka, Surathkal

April 2019

Department of Information Technology, NITK Surathkal
Mini Project
End Semester Evaluation Report (April 2019)

Course Code : IT 350

Course Title: Software Engineering

Project Title: *OneLyf*

Project Group:

Name of the Student	Register No.	Signature with Date
Gaurav U D	16IT113	
Abhilash V	16IT201	
Shrinivas V Shanbhag	16IT244	

Place:

Date: *(Name and Signature of Mini Project Guide)*

Abstract

OneLyf is a natural disaster management application. Here all the disasters are listed. Each of this has a map associated with it, where the other users would have marked which locations are safe, danger, or in which location help is needed the most.

This application is also linked with News API, Weather API, and Google Maps API for getting relevant data about each disaster.

The application can also capture audio files from radio communication and mark the relevant location on the corresponding map.

Keywords: *Django, API (Application Programming Interface)*

Contents

1 Introduction

OneLyf is a disaster management application which is built using Python Django framework. The main objective of this application is to facilitate rescue operations. This application gathers all relevant information about a disaster and compiles it into a single page for easier use. The different modules used in this application are Landing, Event, Donation, Audio.

1.1 *Landing*

This module handles the Login and Sign up operations of the application. It has databases to store information of the users of this applications. New users are prompted to Sign up. Existing users can sign in with their username and password. Django Sessions library is also used to maintain each login session.

1.2 *Event*

This is the main module in this application. This fetches data from different sources through APIs, like maps, news, weather and more and puts them in a single page. It also has the functionality of marking disaster locations on map. It has a comment functionality for users to post on particular events.

1.3 *Donations*

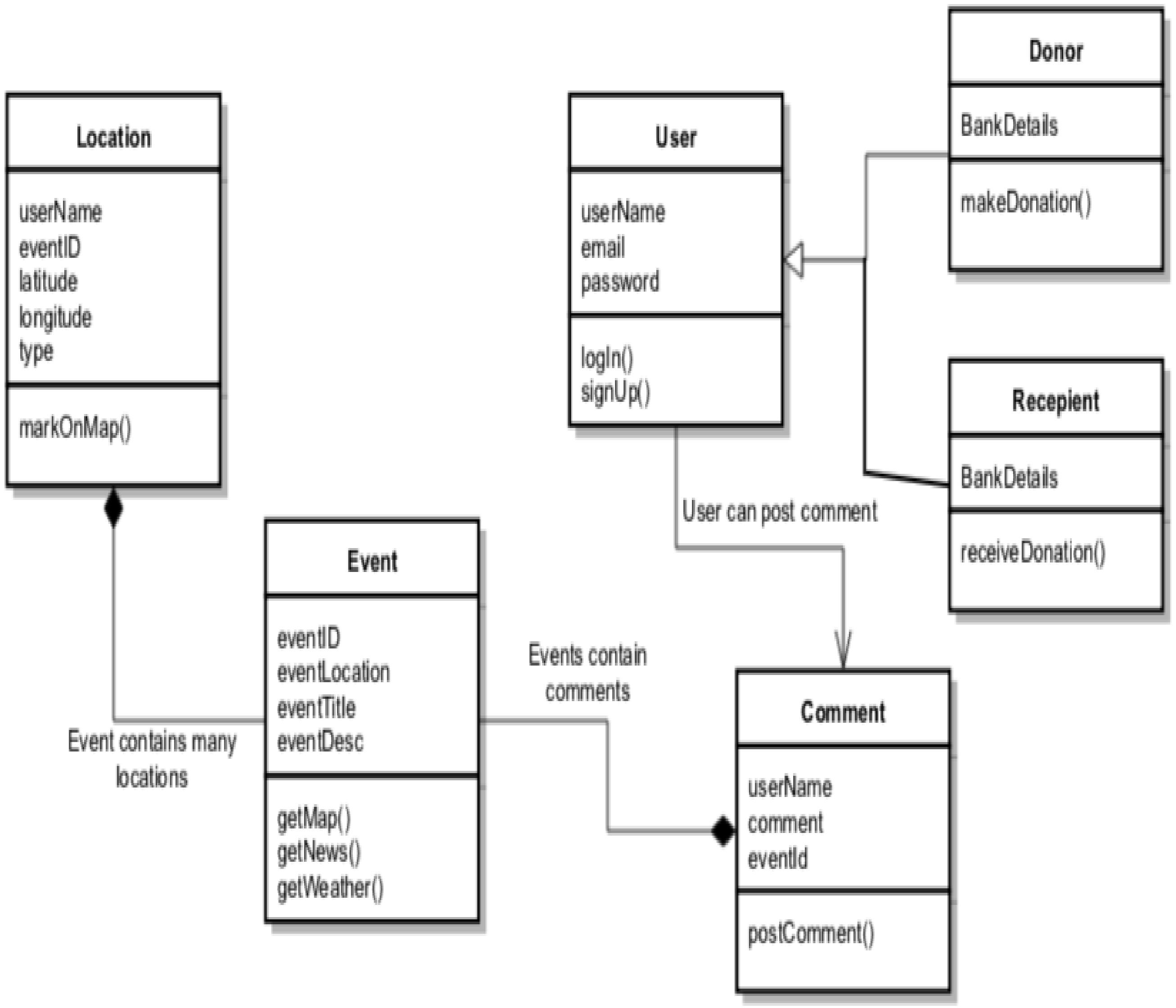
This module provides the users the functionality of donating or receiving donations. Verified Users can register as authentic victims on this module. Donors can donate to particular receivers.

1.4 *Audio*

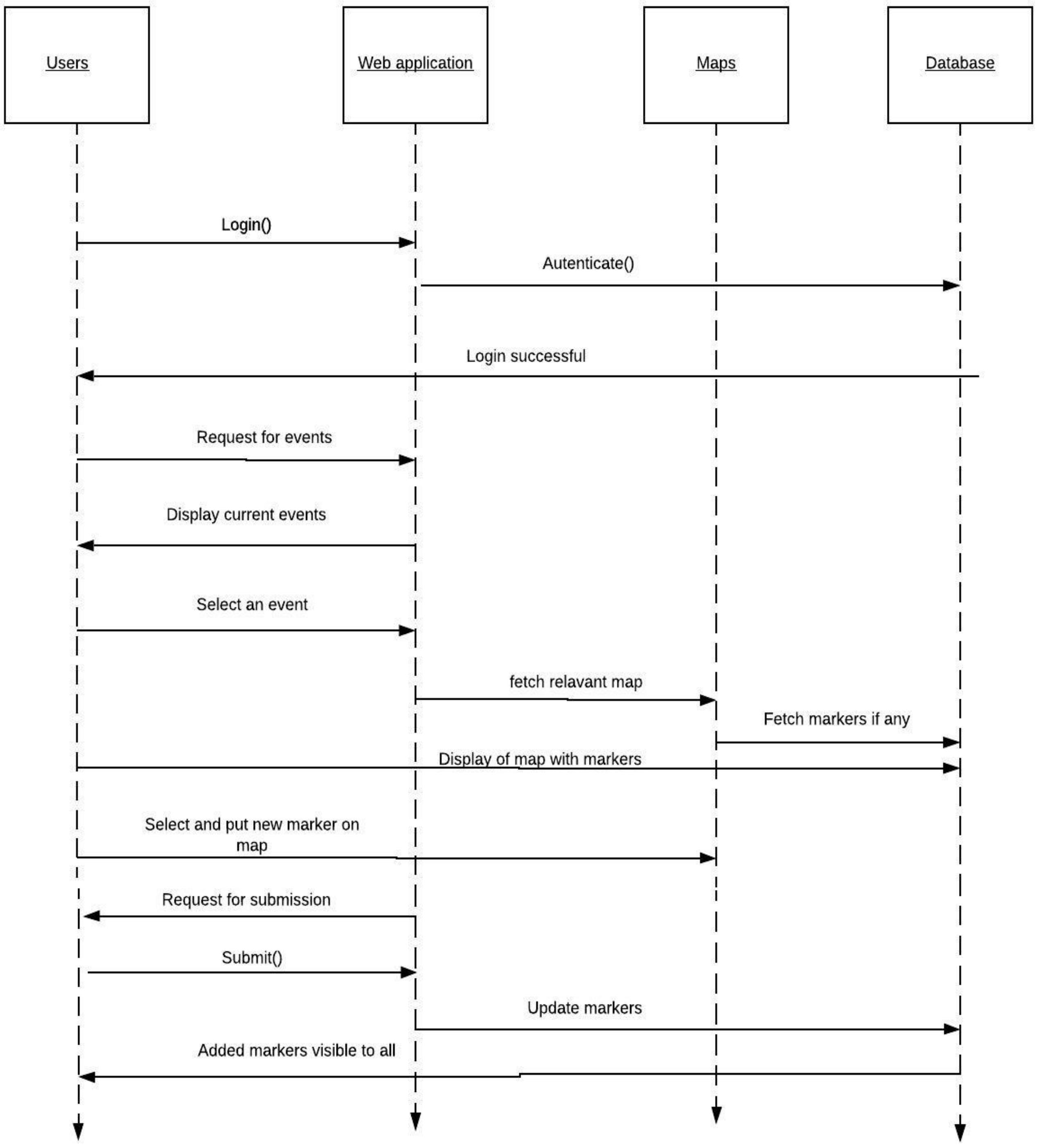
Here a audio file is uploaded which is then converted into text using Azure Speech to Text API. Exact context of this text is obtained through Azure Text services. After this that particular location is marked on the map.

2 Design Diagrams

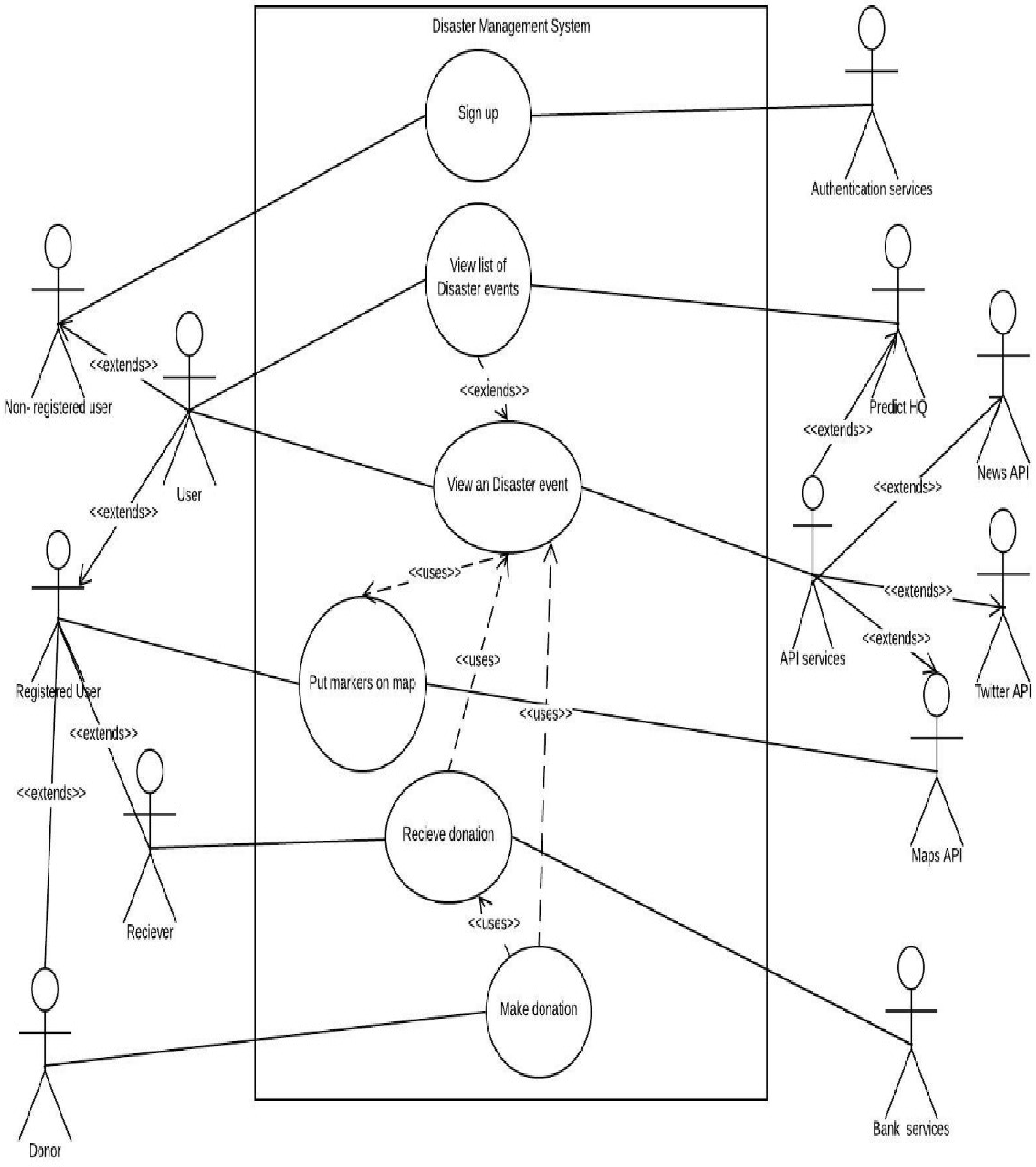
2.1 *UML Class diagram*



2.2 *UML Sequence diagram*

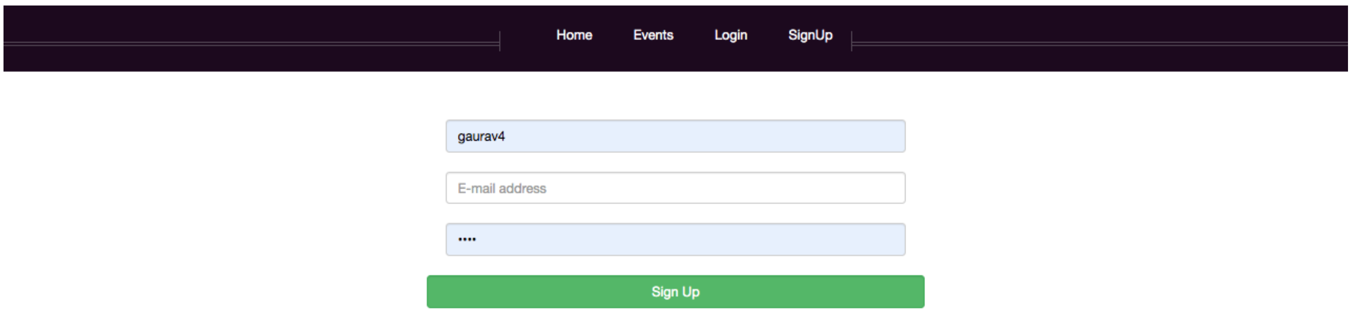


2.3 *UML Use Case diagram*

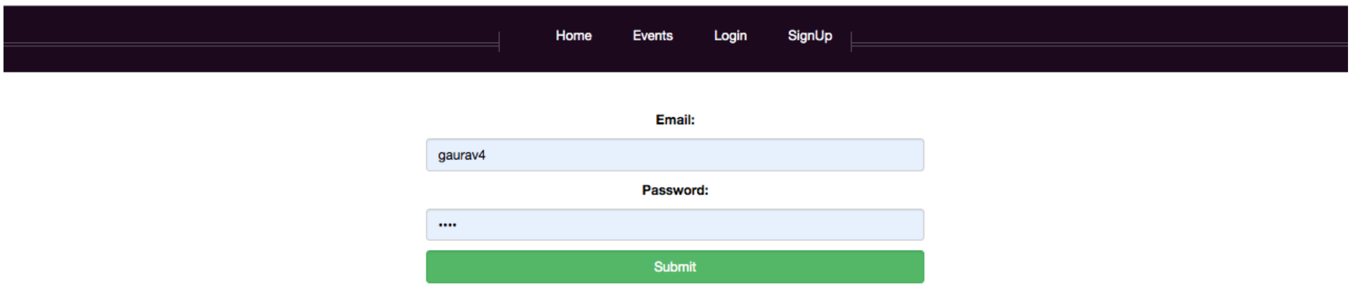


3 Work Done

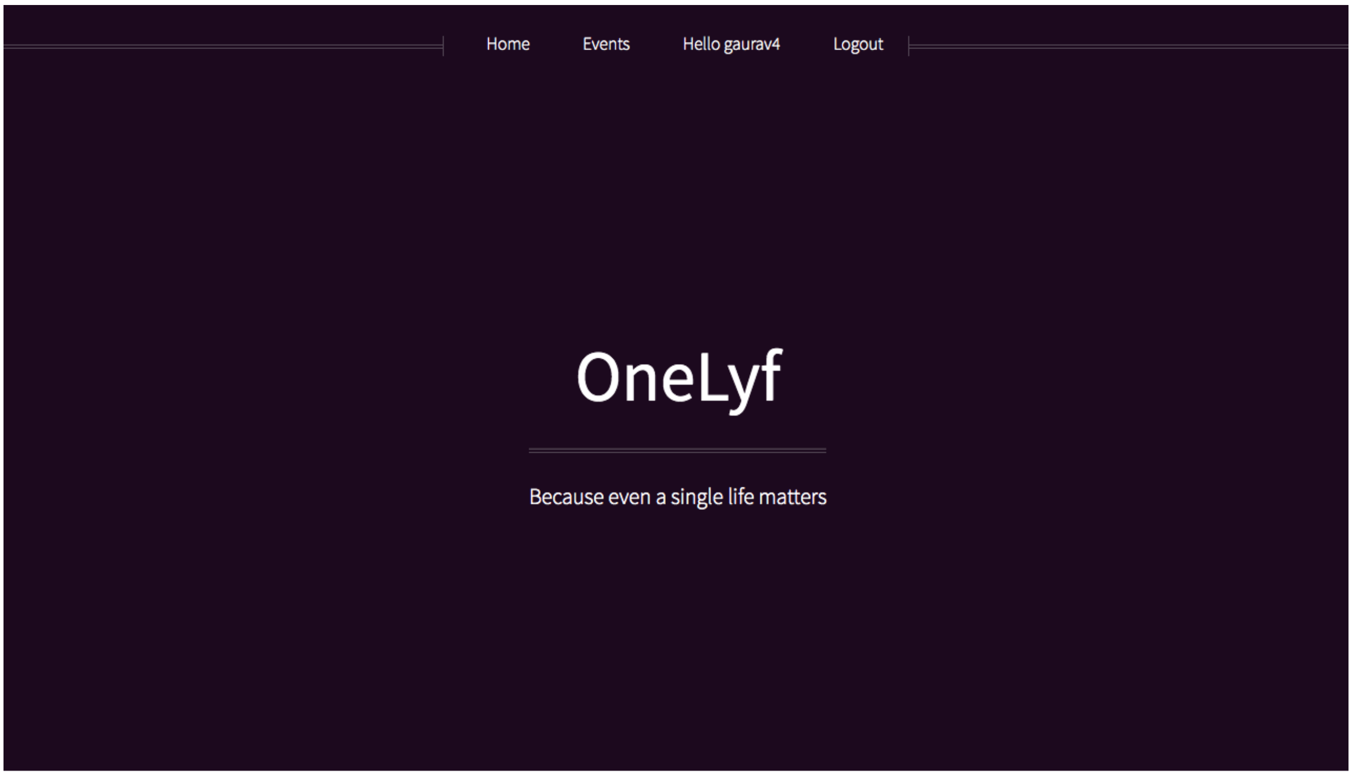
3.1 *Sign Up page*



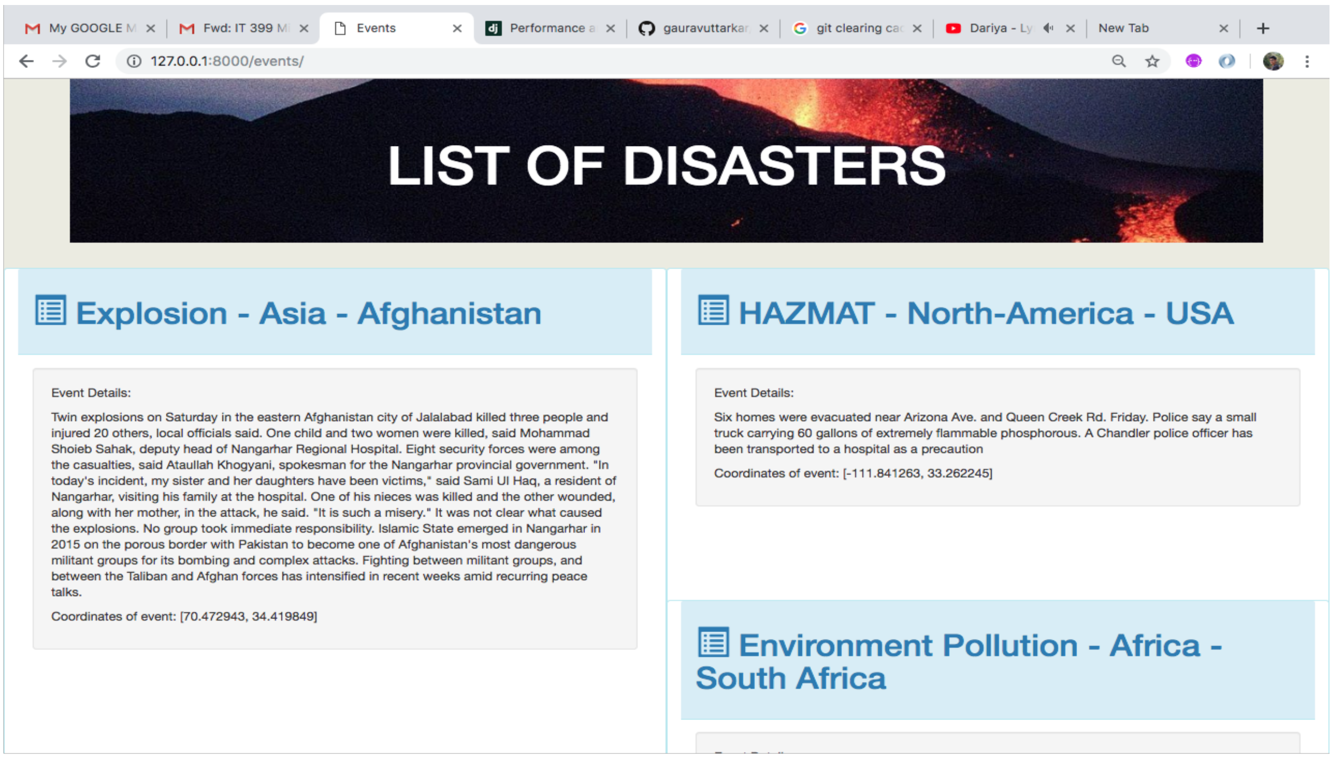
3.2 *Log in page*



3.3 *Home page*



3.4 *List of events*



Explosion - Asia - Afghanistan

Event Details:

Twin explosions on Saturday in the eastern Afghanistan city of Jalalabad killed three people and injured 20 others, local officials said. One child and two women were killed, said Mohammad Shoieb Sahak, deputy head of Nangarhar Regional Hospital. Eight security forces were among the casualties, said Ataullah Khogyani, spokesman for the Nangarhar provincial government. "In today's incident, my sister and her daughters have been victims," said Sami Ul Haq, a resident of Nangarhar, visiting his family at the hospital. One of his nieces was killed and the other wounded, along with her mother, in the attack, he said. "It is such a misery." It was not clear what caused the explosions. No group took immediate responsibility. Islamic State emerged in Nangarhar in 2015 on the porous border with Pakistan to become one of Afghanistan's most dangerous militant groups for its bombing and complex attacks. Fighting between militant groups, and between the Taliban and Afghan forces has intensified in recent weeks amid recurring peace talks.

Coordinates of event: [70.472943, 34.419849]

HAZMAT - North-America - USA

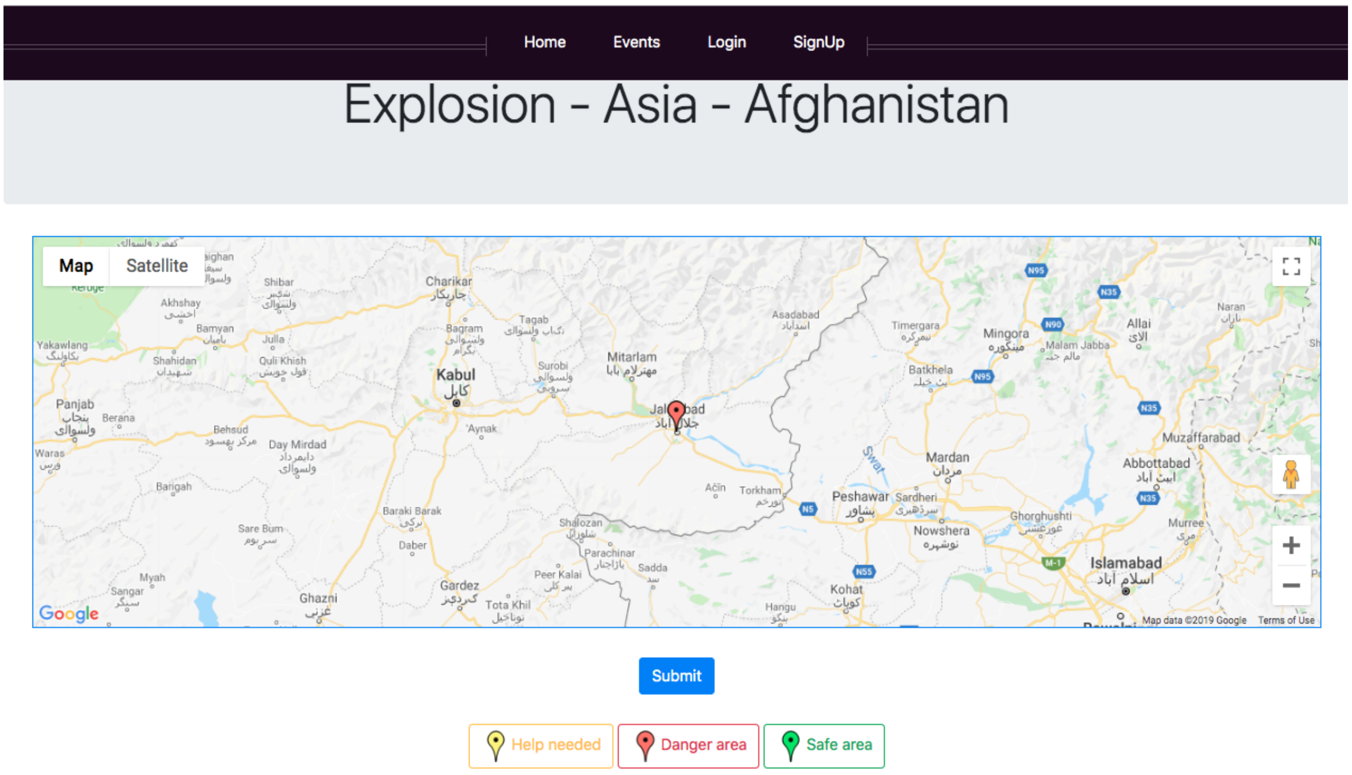
Event Details:

Six homes were evacuated near Arizona Ave. and Queen Creek Rd. Friday. Police say a small truck carrying 60 gallons of extremely flammable phosphorous. A Chandler police officer has been transported to a hospital as a precaution

Coordinates of event: [-111.841263, 33.262245]

Environment Pollution - Africa - South Africa

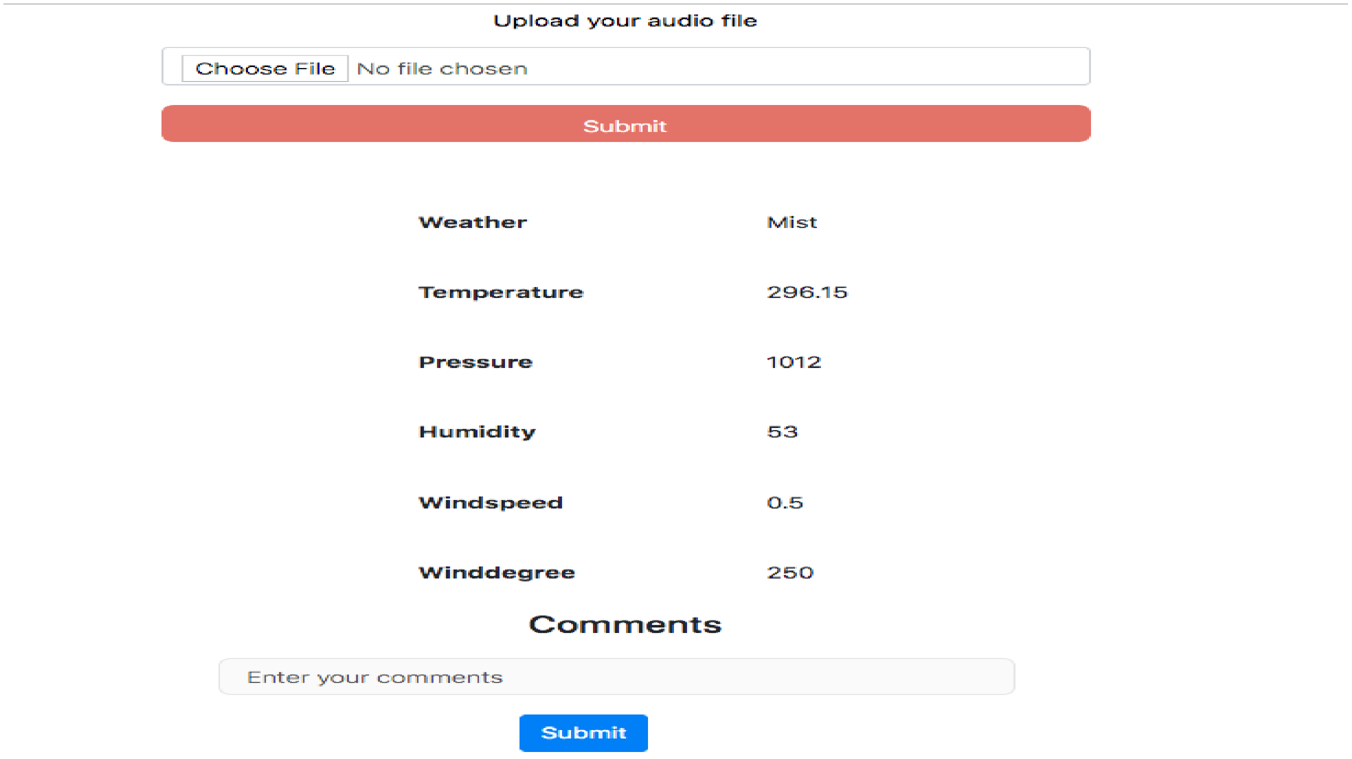
3.5 *Event map*



Explosion - Asia - Afghanistan

Submit

3.6 *Event detail*



Upload your audio file

Choose File No file chosen

Submit

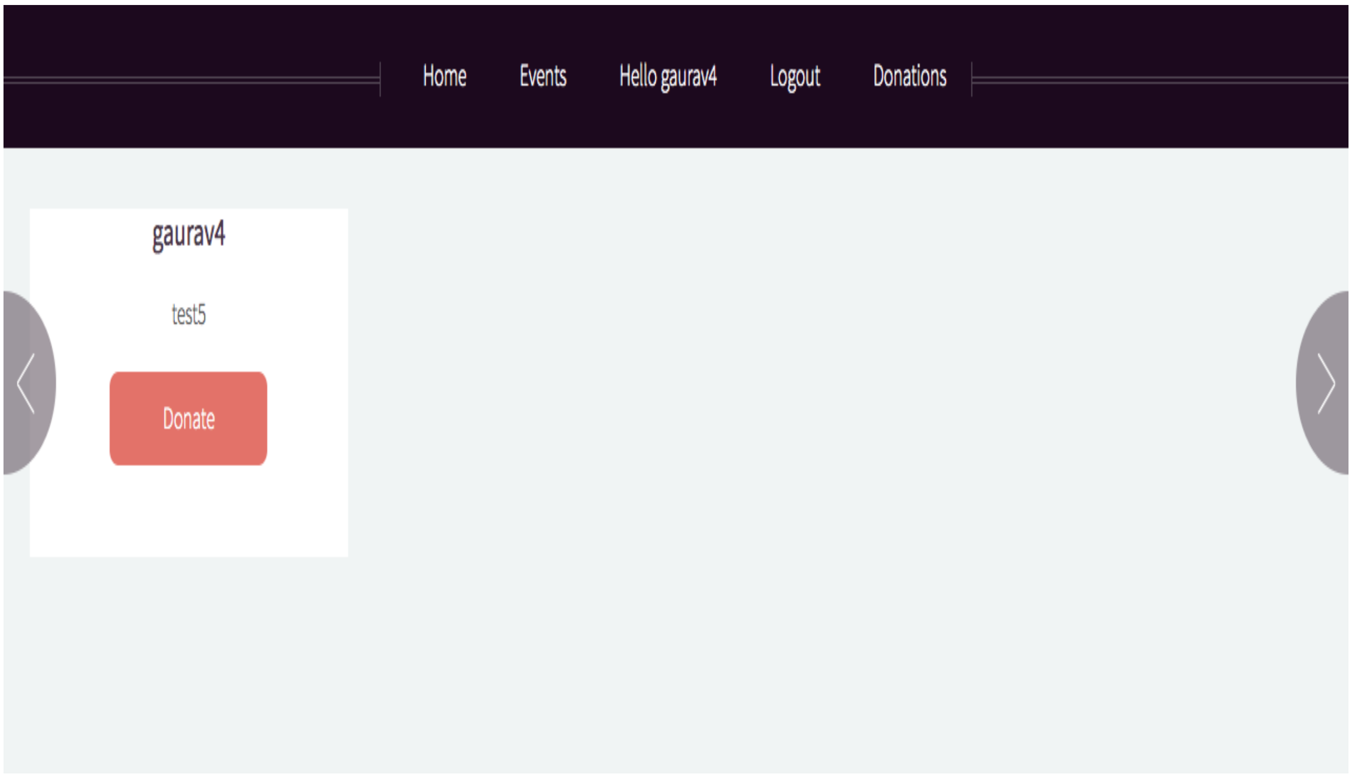
Weather	Mist
Temperature	296.15
Pressure	1012
Humidity	53
Windspeed	0.5
Winddegree	250

Comments

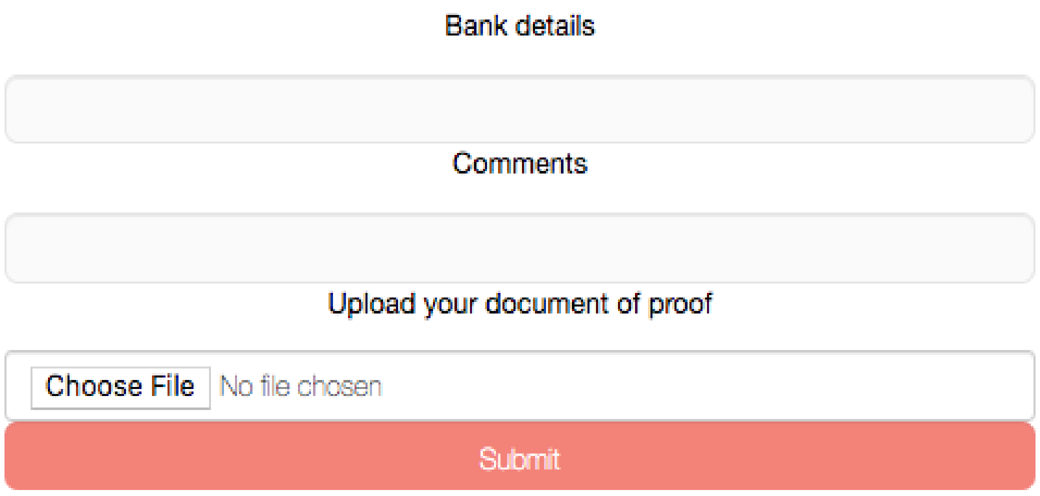
Enter your comments

Submit

3.7 *Donation*



3.8 *Recepient*



4 Tools used

4.1 *Selenium-Python*

Selenium Python bindings provides a simple API to write functional/acceptance tests using Selenium WebDriver. Through Selenium Python API you can access all functionalities of Selenium WebDriver in an intuitive way.

- Used in Automation testing

4.2 *Lucid Chart*

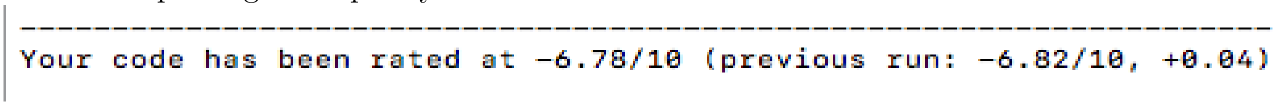
Lucidchart is a web-based proprietary platform that is used to allow users who are located in multiple locations to collaborate with their colleagues in drawing, revising and sharing charts and diagrams.

- Used in creating Sequence diagram
- Used in creating Class diagram
- Used in creating Usecase diagram

4.3 *Pylint*

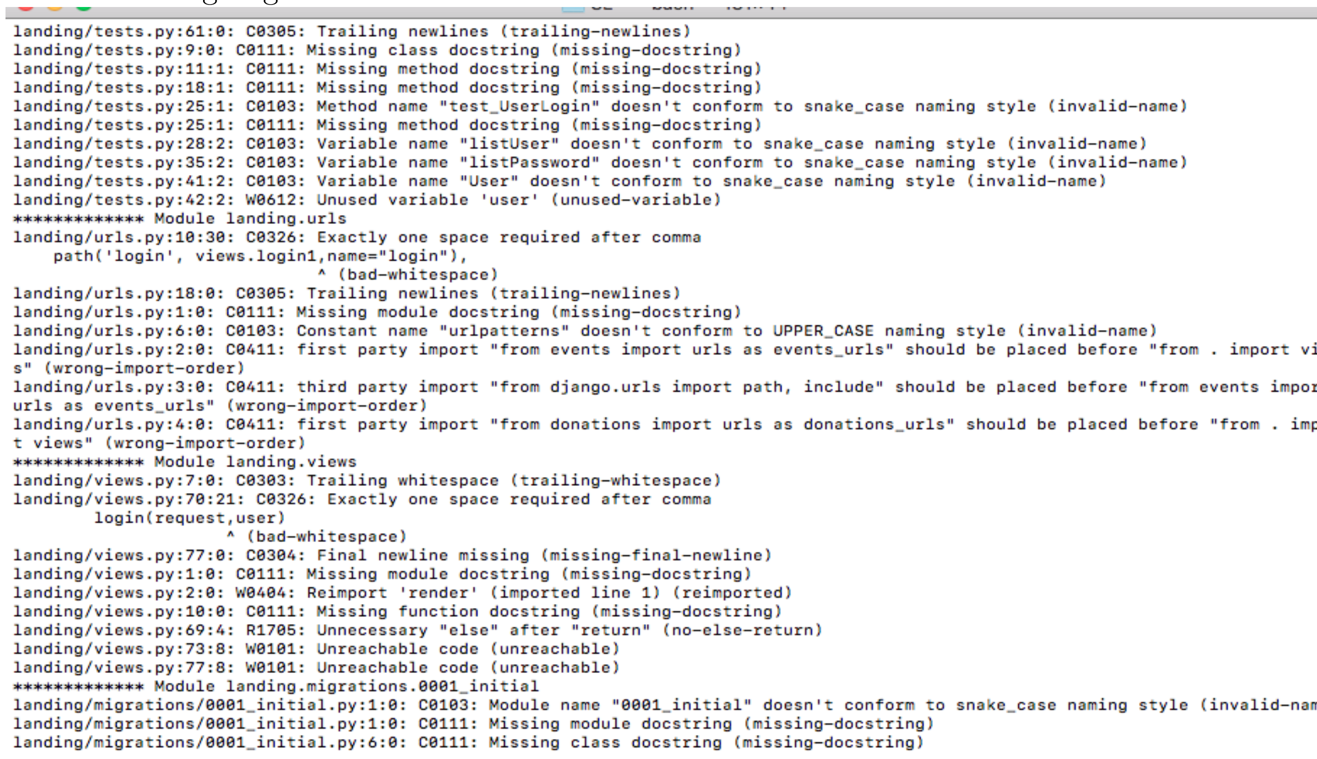
Pylint is a source-code, bug and quality checker for the Python programming language. It follows the style recommended by PEP 8, the Python style guide.

- Used for improving code quality



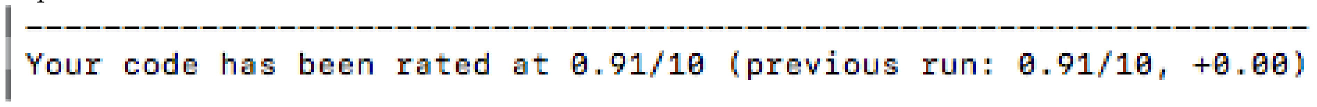
```
--------------------------------------------------------------------
Your code has been rated at -6.78/10 (previous run: -6.82/10, +0.04)
```

- Used for finding bugs



```
landing/tests.py:61:0: C0305: Trailing newlines (trailing-newlines)
landing/tests.py:9:0: C0111: Missing class docstring (missing-docstring)
landing/tests.py:11:1: C0111: Missing method docstring (missing-docstring)
landing/tests.py:18:1: C0111: Missing method docstring (missing-docstring)
landing/tests.py:25:1: C0103: Method name "test_UserLogin" doesn't conform to snake_case naming style (invalid-name)
landing/tests.py:25:1: C0111: Missing method docstring (missing-docstring)
landing/tests.py:28:2: C0103: Variable name "listUser" doesn't conform to snake_case naming style (invalid-name)
landing/tests.py:35:2: C0103: Variable name "listPassword" doesn't conform to snake_case naming style (invalid-name)
landing/tests.py:41:2: C0103: Variable name "User" doesn't conform to snake_case naming style (invalid-name)
landing/tests.py:42:2: W0612: Unused variable 'user' (unused-variable)
************* Module landing.urls
landing/urls.py:10:30: C0326: Exactly one space required after comma
    path('login', views.login1,name="login"),
                              ^ (bad-whitespace)
landing/urls.py:18:0: C0305: Trailing newlines (trailing-newlines)
landing/urls.py:1:0: C0111: Missing module docstring (missing-docstring)
landing/urls.py:6:0: C0103: Constant name "urlpatterns" doesn't conform to UPPER_CASE naming style (invalid-name)
landing/urls.py:2:0: C0411: first party import "from events import urls as events_urls" should be placed before "from . import vi
s" (wrong-import-order)
landing/urls.py:3:0: C0411: third party import "from django.urls import path, include" should be placed before "from events impor
urls as events_urls" (wrong-import-order)
landing/urls.py:4:0: C0411: first party import "from donations import urls as donations_urls" should be placed before "from . imp
t views" (wrong-import-order)
************* Module landing.views
landing/views.py:7:0: C0303: Trailing whitespace (trailing-whitespace)
landing/views.py:70:21: C0326: Exactly one space required after comma
        login(request,user)
                     ^ (bad-whitespace)
landing/views.py:77:0: C0304: Final newline missing (missing-final-newline)
landing/views.py:1:0: C0111: Missing module docstring (missing-docstring)
landing/views.py:2:0: W0404: Reimport 'render' (imported line 1) (reimported)
landing/views.py:10:0: C0111: Missing function docstring (missing-docstring)
landing/views.py:69:4: R1705: Unnecessary "else" after "return" (no-else-return)
landing/views.py:73:8: W0101: Unreachable code (unreachable)
landing/views.py:77:8: W0101: Unreachable code (unreachable)
************* Module landing.migrations.0001_initial
landing/migrations/0001_initial.py:1:0: C0103: Module name "0001_initial" doesn't conform to snake_case naming style (invalid-nam
landing/migrations/0001_initial.py:1:0: C0111: Missing module docstring (missing-docstring)
landing/migrations/0001_initial.py:6:0: C0111: Missing class docstring (missing-docstring)
```

- Improved score



```
------------------------------------------------------------------
Your code has been rated at 0.91/10 (previous run: 0.91/10, +0.00)
```

4.4 *Evernote*

Evernote is a mobile app designed for note taking, organizing, task lists, and archiving.

- Used for weekly project reviews.

- Used for task distribution

- Used for brainstorming ideas.

4.5 ***Google Lighthouse***

Lighthouse is an open-source, automated tool for improving the quality of web pages. It can run against any web page, public or requiring authentication. It has audits for performance, accessibility, progressive web apps, and more.

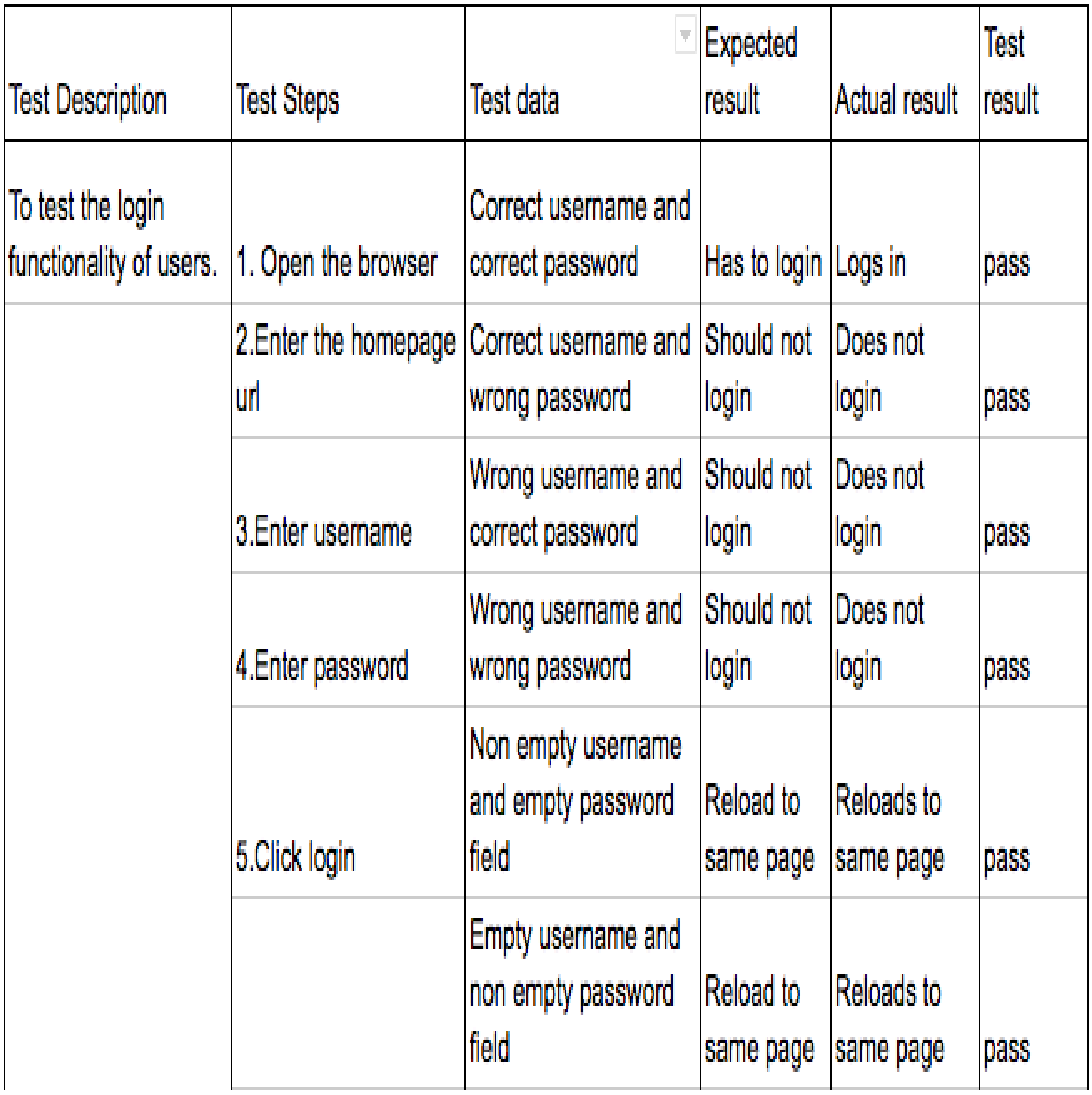
- Used for performance testing

4.6 ***Python Virtual Environment***

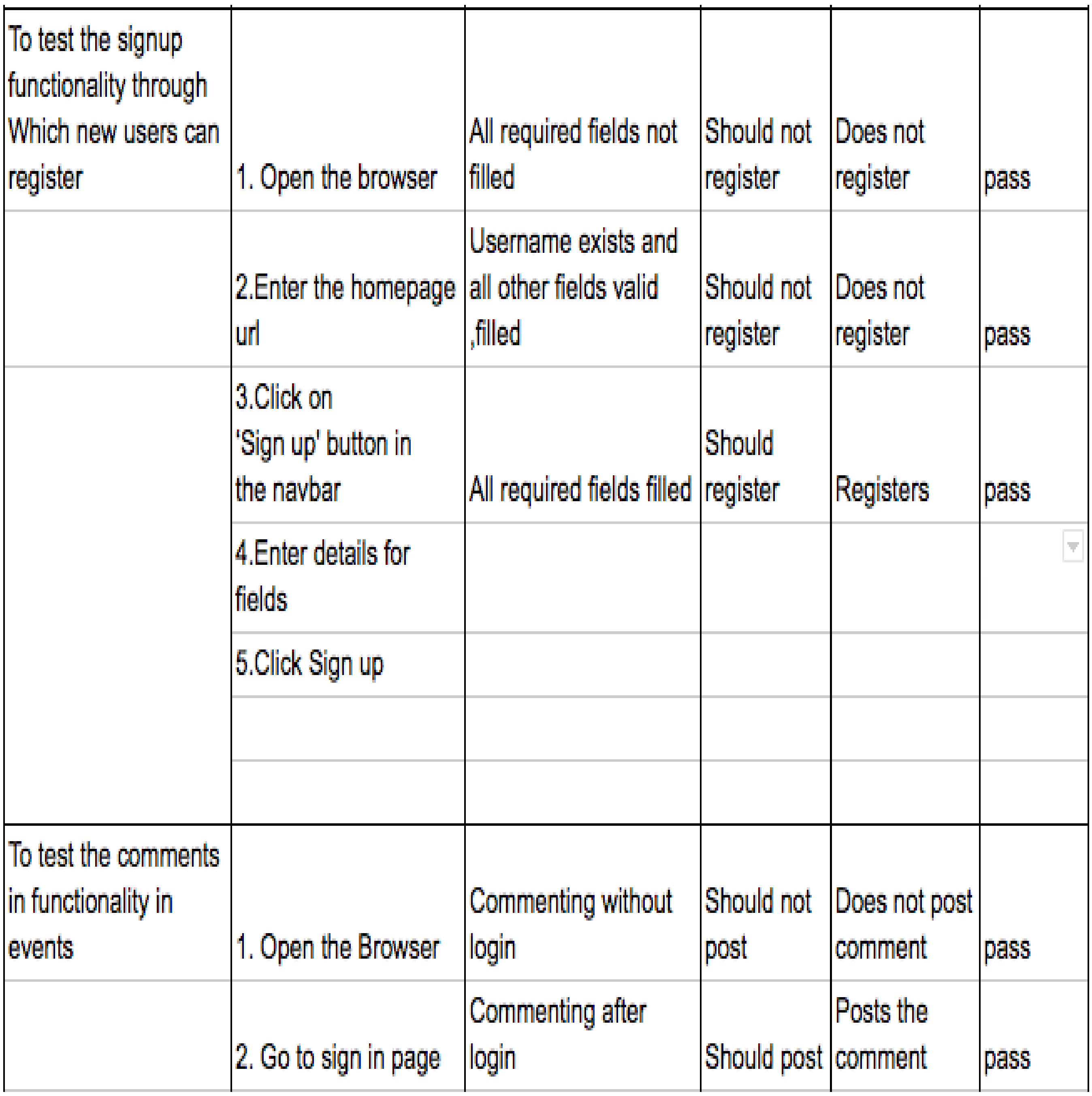
A virtual environment is a tool that helps to keep dependencies required by different projects separate by creating isolated python virtual environments for them.

- Used for creating a separate environment for the project.

5 Manual Test case Design

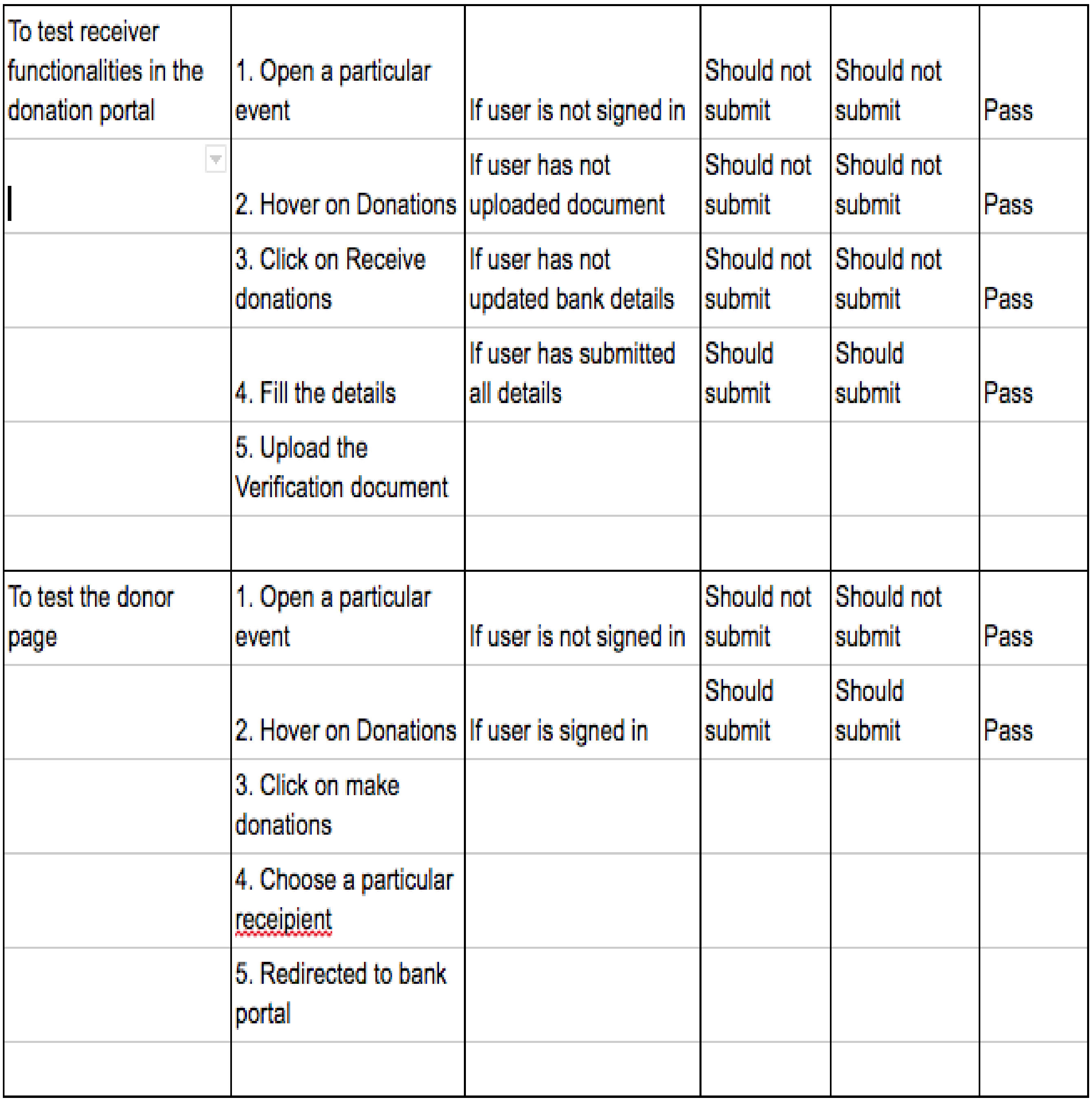


Test Description	Test Steps	Test data	Expected result	Actual result	Test result
To test the login functionality of users.	1. Open the browser	Correct username and correct password	Has to login	Logs in	pass
	2.Enter the homepage url	Correct username and wrong password	Should not login	Does not login	pass
	3.Enter username	Wrong username and correct password	Should not login	Does not login	pass
	4.Enter password	Wrong username and wrong password	Should not login	Does not login	pass
	5.Click login	Non empty username and empty password field	Reload to same page	Reloads to same page	pass
		Empty username and non empty password field	Reload to same page	Reloads to same page	pass



To test the signup functionality through Which new users can register	1. Open the browser	All required fields not filled	Should not register	Does not register	pass
	2.Enter the homepage url	Username exists and all other fields valid ,filled	Should not register	Does not register	pass
	3.Click on 'Sign up' button in the navbar	All required fields filled	Should register	Registers	pass
	4.Enter details for fields				
	5.Click Sign up				
To test the comments in functionality in events	1. Open the Browser	Commenting without login	Should not post	Does not post comment	pass
	2. Go to sign in page	Commenting after login	Should post	Posts the comment	pass

	3.Sign in				
	4. Go to an event				
	5. Write Comment				
	6. Submit comment				
To test various markers of google maps	1. Open the Browser	Add help marker without login	Should not post	Does not post	pass
	2. Go to sign in page	Add save marker without login	Should not post	Does not post	pass
	3.Sign in	Add danger marker without login	Should not post	Does not post	pass
	4. Go to an event	Add markers with login	Should post	Posts the marker	pass
	5. Select and put a marker				
	6. Submit				



To test receiver functionalities in the donation portal	1. Open a particular event	If user is not signed in	Should not submit	Should not submit	Pass
	2. Hover on Donations	If user has not uploaded document	Should not submit	Should not submit	Pass
	3. Click on Receive donations	If user has not updated bank details	Should not submit	Should not submit	Pass
	4. Fill the details	If user has submitted all details	Should submit	Should submit	Pass
	5. Upload the Verification document				
To test the donor page	1. Open a particular event	If user is not signed in	Should not submit	Should not submit	Pass
	2. Hover on Donations	If user is signed in	Should submit	Should submit	Pass
	3. Click on make donations				
	4. Choose a particular receipient				
	5. Redirected to bank portal				

6 Automation Testing

Automation testing has been done using in built Django Test framework and Selenium web driver for python.

6.1 *Sign up*

Sign up Automation is run by Selenium web driver by inputting various values for different fields and testing if each case passes or fails.

6.2 *Log in*

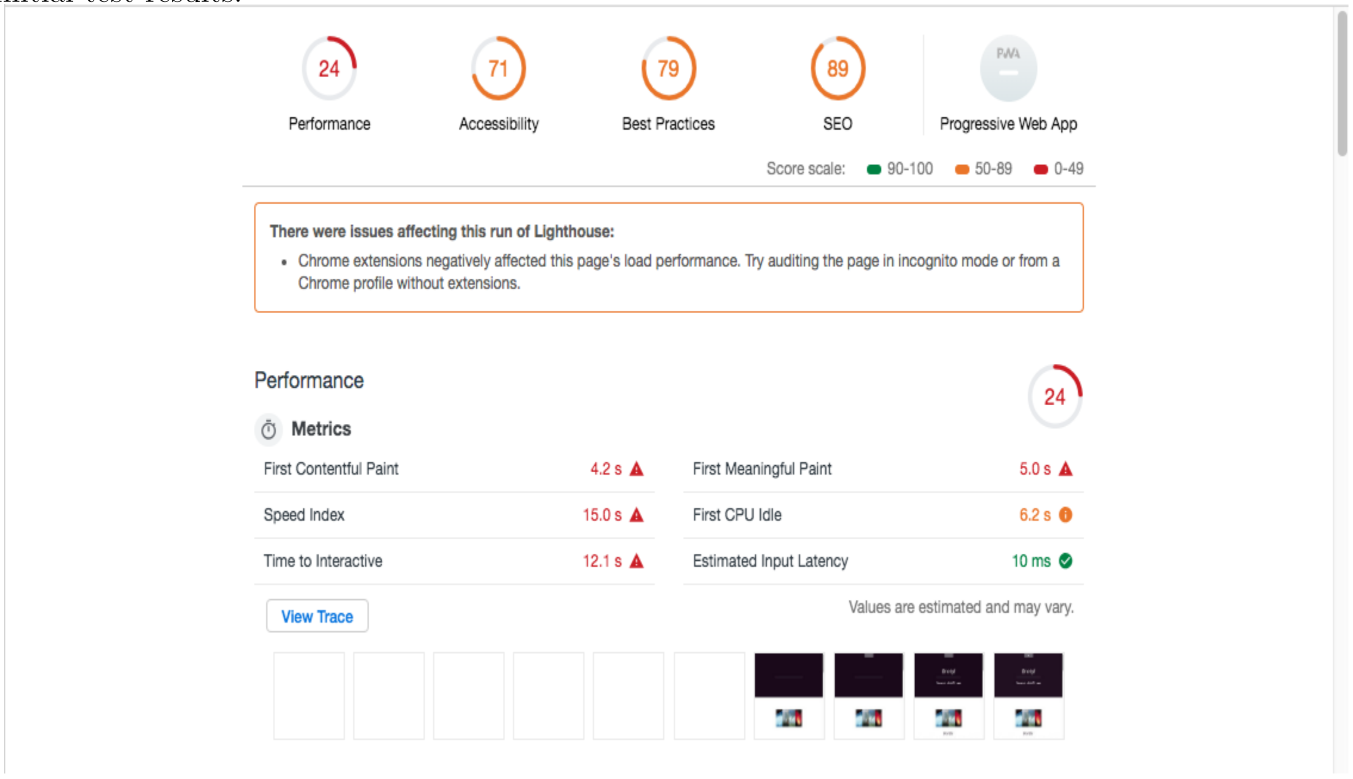
Log In Automation is run by Selenium web driver by inputting different combinations of right and wrong credentials and testing if each case passes or fails.

6.3 *User comments*

Comment Automation is run by Selenium web driver by inputting comments for different events and testing if each case passes or fails.

7 Performance testing

Initial test results.



24 Performance | 71 Accessibility | 79 Best Practices | 89 SEO | Progressive Web App

Score scale: 🟩 90-100 🟧 50-89 🟥 0-49

There were issues affecting this run of Lighthouse:

- Chrome extensions negatively affected this page's load performance. Try auditing the page in incognito mode or from a Chrome profile without extensions.

Performance 24

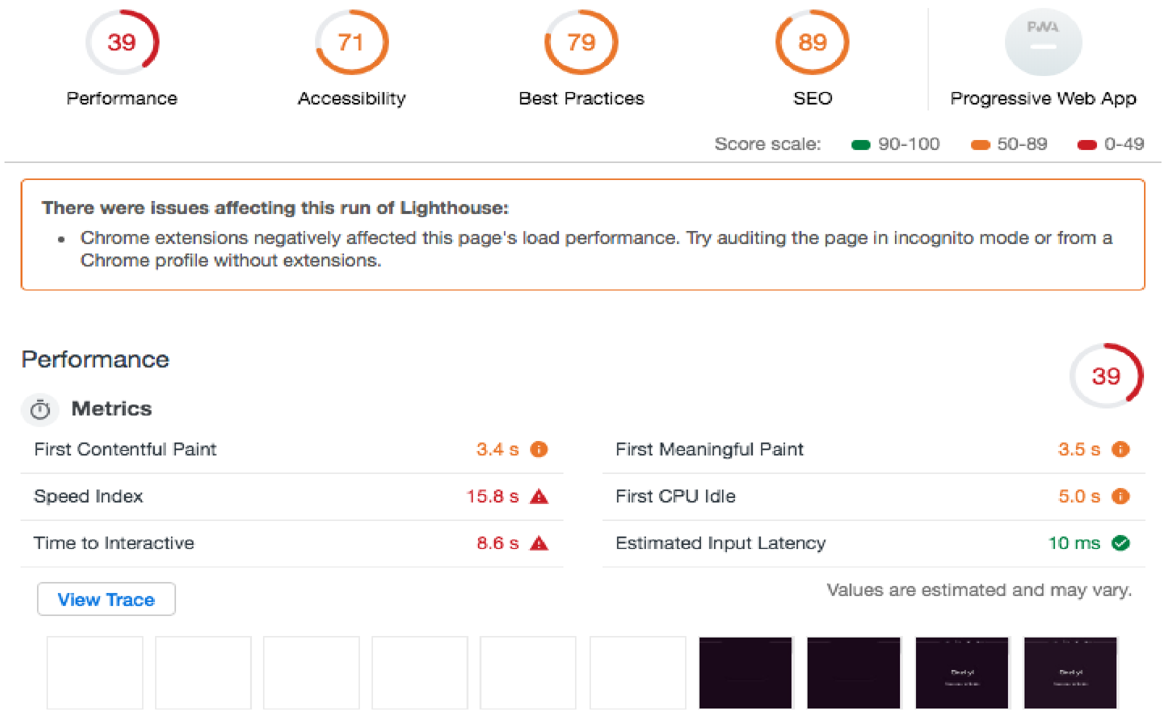
Metrics

First Contentful Paint	4.2 s ▲	First Meaningful Paint	5.0 s ▲
Speed Index	15.0 s ▲	First CPU Idle	6.2 s ●
Time to Interactive	12.1 s ▲	Estimated Input Latency	10 ms ✓

View Trace

Values are estimated and may vary.

After using compression middleware.



39 Performance | 71 Accessibility | 79 Best Practices | 89 SEO | Progressive Web App

Score scale: 🟩 90-100 🟧 50-89 🟥 0-49

There were issues affecting this run of Lighthouse:

- Chrome extensions negatively affected this page's load performance. Try auditing the page in incognito mode or from a Chrome profile without extensions.

Performance 39

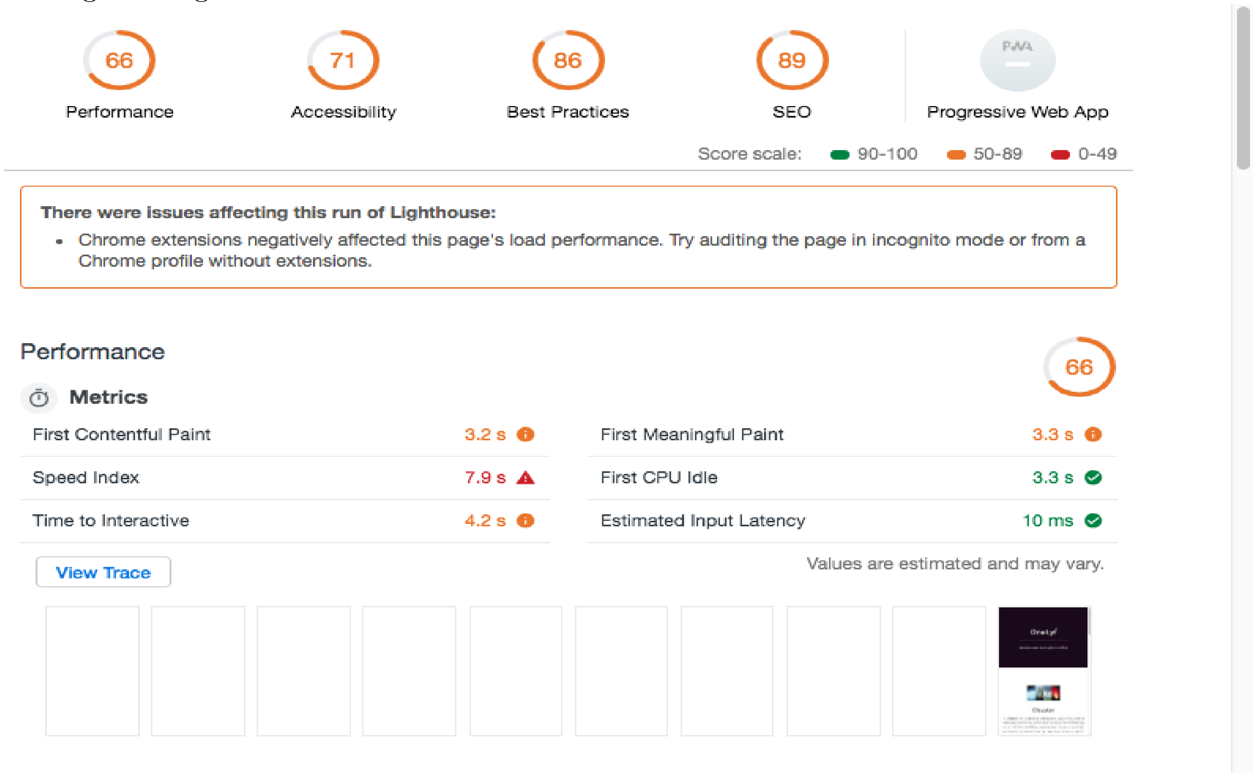
Metrics

First Contentful Paint	3.4 s ●	First Meaningful Paint	3.5 s ●
Speed Index	15.8 s ▲	First CPU Idle	5.0 s ●
Time to Interactive	8.6 s ▲	Estimated Input Latency	10 ms ✓

View Trace

Values are estimated and may vary.

After enabling caching



66 Performance | 71 Accessibility | 86 Best Practices | 89 SEO | PWA Progressive Web App

Score scale: 🟩 90-100 🟧 50-89 🟥 0-49

There were issues affecting this run of Lighthouse:

- Chrome extensions negatively affected this page's load performance. Try auditing the page in incognito mode or from a Chrome profile without extensions.

Performance 66

Metrics

First Contentful Paint	3.2 s	First Meaningful Paint	3.3 s
Speed Index	7.9 s	First CPU Idle	3.3 s
Time to Interactive	4.2 s	Estimated Input Latency	10 ms

View Trace

Values are estimated and may vary.

8 Cost estimation

Cocomo (Constructive Cost Model) is a regression model based on LOC, i.e number of Lines of Code in thousands(KLOC). It is a procedural cost estimate model for software projects ,the key parameters which define the quality of any software products, which are also an outcome of the Cocomo are primarily:

- Effort: Amount of labor that will be required to complete a task. It is measured in person-months units.
- Schedule:: Simply means the amount of time required for the completion of the job, which is, of course, proportional to the effort put. It is measured in the units of time such as weeks, months.

COCOMO applies to three classes of software projects:

1. Organic: A software project is said to be an organic type if the team size required is adequately small, the problem is well understood and has been solved in the past and also the team members have a nominal experience regarding the problem. 2-50 KLOC, small and stable
2. Semi-detached: A software project is said to be a Semi-detached type if the vital characteristics such as team-size, experience, knowledge of the various programming environment lie in between that of organic and Embedded. 50-300 KLOC, medium-sized, average abilities, medium time-constraints.
3. Embedded: A software project with requiring the highest level of complexity, creativity, and experience requirement fall under this category. Greater than 300 KLOC, large project team, complex, innovative, severe constraints

$E = a(KLOC)^b$

$a = 1.25$

$b = 1.05,$

$KLOC = 1.5$

$E = 1.91$

$T = c(E)^d$

$c = 2.5$

$d = 0.38$

$T = 3.199$

$P = E/T$

$P = 2$

Here a, b, c, d are constants for organic.

E Effort applied by staff per month.

T is Time for development in months.

P is People for development.

9 Conclusion

This project has been implemented using proper Software development processes and it makes proper use of various software development tools. It has used code quality checker, bug checker, progress tracking and various other important tools. This project comprehensively demonstrates how a project is conceptualized and then implemented in a step by a step fashion following proper methodology.

References

1. Google Maps Documentation
2. API documentation
3. IEEE Software Requirements Specification Template