

# UIAnalyzer

---

A WEB BASED UI SMELL FINDER TOOL FOR WEB APPLICATION

Md. Aquib Azmain  
INSTITUTE OF INFORMATION TECHNOLOGY

# A web based UI smell finder tool for web application

## Final Report of the Project

Submitted by

**Md. Aquib Azmain**

BSSE 0718

BSSE Session: 2014-2015

Supervised by

**Md. Saeed Siddik**

Lecturer

Institute of Information Technology

University of Dhaka



**Institute of Information Technology  
University of Dhaka**

25-11-2018

## LETTER OF TRANSMITTAL

25<sup>th</sup> November 2018

The Coordinator  
Software Project Lab 3  
Institute of Information Technology  
University of Dhaka

**Subject: Submission of final report of Software Project Lab 3.**

Dear Sir,

With due respect, I am pleased to submit the report on UIAnalyzer, A web based UI smell finder tool for web application. Although this report may have shortcomings, I have tried my level best to produce an acceptable report. This documentation includes Software Requirement Specifications, Design, Test plan for implementing the tool as well as a user manual for the users of it. I would be highly obliged if you overlooked the mistakes and accepted the effort that has been put in this report.

Sincerely yours,

Md. Aquib Azmain

Roll: BSSE 0718  
BSSE 7<sup>th</sup> batch  
Institute of Information Technology  
University of Dhaka

## DOCUMENT AUTHENTICATION

This project document has been approved by the following persons.

Prepared by,

Approved by,

-----  
Md. Aquib Azmain  
BSSE-0718  
Institute of Information Technology  
University of Dhaka

-----  
Md. Saeed Siddik  
Lecturer  
Institute of Information Technology  
University of Dhaka

## LETTER OF ENDORSEMENT

**Subject:** Approval of the Report

This letter is to certify that, Md. Aquib Azmain, BSSE0718, student of Institute of Information Technology, University of Dhaka, has done the project “UIAnalyzer” under my supervision. I have gone through the report. All the information mentioned in this document is true.

I wish him all the best and hope that he will lead a successful career.

Md. Saeed Siddik

Lecturer

Institute of Information Technology

University of Dhaka

## ACKNOWLEDGMENT

At first, I would like to thank almighty for helping to complete the project 'UIAnalyzer'.

I would like to express my deepest gratitude to all those who provided me the support and encouragement to start this project. Thanks to my supervisor Md. Saeed Siddik, Lecturer, Institute of Information Technology, University of Dhaka, whose continuous suggestions and guidance has been invaluable to me.

I am grateful to the Institute of Information Technology for giving me the opportunity to conduct such a project.

Lastly, I would like to thank my classmates. They have always been helpful and provided valuable insights from time to time.

## ABSTRACT

The project 'UIAnalyzer' is about building a software that will analyze the user interface of a web application. Usability assessment of web applications continues to be an expensive and often neglected practice. While large companies are able to spare resources for studying and improving usability in their products, smaller businesses often divert theirs in other aspects. This software finds the usability smells from user interface of the application. This is a web based tool that works with web application. The automated strategy to usability smell identification is based on a process consisting of three steps: Events Logging (Records each event by parsing HTML elements and web crawling), Usability Smells Detection (Five usability smells can be found using this tool), and Reporting (Shows results in CSV files). After inserting the website URL, UIAnalyzer can analyze a website, find specific usability smells without any human interaction.

## Table of Contents

1	Introduction.....	1
2	Scope of the project .....	1
3	Project Description .....	2
3.1	Specific Requirements.....	2
3.1.1	Functionality .....	2
3.1.2	Usability .....	2
3.1.3	Performance .....	2
3.1.4	Web Based Requirements .....	2
3.2	Usage scenario of UIAnalyzer.....	3
3.2.1	Events logging .....	3
3.2.2	Smell Detection.....	4
3.2.3	Reporting .....	5
4	Scenario-Based Modeling.....	6
4.1	Use-case Diagram.....	6
4.2	Activity Diagram .....	8
5	Class-Based Modeling.....	9
5.1	Final Classes .....	9
5.2	Class Diagram .....	11
6	Architectural Design .....	12
6.1	Overview .....	12
6.2	Instantiations of the System .....	13
6.3	Elaborated Deployment .....	13
7	User Interface Design .....	14
8	Implementation Overview.....	18
8.1	Technology Used in implementation .....	18
8.1.1	Client-side Technology.....	18



8.1.2	Server-side Technology.....	19
8.1.3	Implementation Tools.....	20
8.2	System Configuration.....	21
8.2.1	Server Requirements .....	21
8.2.2	Supported Browsers .....	21
8.3	Source Code Description.....	22
8.3.1	WebExplorer class.....	22
8.3.2	EventPreprocessor class .....	23
8.3.3	SmellIdentifier class.....	23
8.3.4	ReportGenerator class.....	24
8.3.5	CSVGenerator class.....	25
8.3.6	ToolManager class .....	25
9	Test Plan, Test Cases and Automated Testing.....	26
9.1	Test Item to be tested.....	26
9.2	Features to be tested .....	26
9.3	Features not to be tested.....	26
9.4	Approach.....	26
9.5	Item Pass Fail Criteria.....	26
9.6	Test Deliverables.....	26
9.7	Testing Tasks .....	27
9.8	Testing cost .....	27
9.9	Test design specification .....	27
9.10	Test case specifications.....	28
9.11	Test case execution using Selenium.....	29
10	User Manual .....	32
11	Conclusion .....	41

## List of figures

Figure 1: The three steps of the UIAnalyzer System.....	4
Figure 2: Level-0 Usecase Diagram .....	6
Figure 3: Level-1 Usecase Diagram .....	6
Figure 4: Activity Diagram of Whole System .....	8
Figure 5: Class Diagram of UIAnalyzer .....	11
Figure 6: 2-tier architecture of UIAnalyzer .....	12
Figure 7: Archetectural overview of UIAnalyzer .....	12
Figure 8: Instantiations of UIAnalyzer.....	13
Figure 9: Elaborated Deployment of UIAnalyzer .....	13
Figure 10: Home page of UIAnalyzer .....	14
Figure 11: Application Under Analysis View .....	15
Figure 12: Smell Specific View .....	16
Figure 13: Summary Report View .....	17
Figure 14: Level-1 Usecase Diagram .....	27
Figure 15: Selenium test case 1 and output .....	30
Figure 16: Selenium test case 2 and output .....	31
Figure 17: Selenium test case 3 .....	31
Figure 18: Enter url of UIAnalyzer.....	33
Figure 19: Homepage of UIAnalyzer .....	33
Figure 20: Error page .....	34
Figure 21: Analysis result .....	34
Figure 22: 5 sections of the analysis page .....	35
Figure 23: Overall analysis view .....	36
Figure 24: Smell Summary View .....	37
Figure 25: No link, Broken link report.....	38
Figure 26: Flash scrolling report.....	38
Figure 27: Smell meter view .....	39
Figure 28: Screenshot view .....	39
Figure 29: List of pages view .....	40

## List of Tables

Table 1: Usability Smells [1] .....	3
Table 2: Class card of WebExplorer Class .....	9
Table 3: Class card of EventProcessor class .....	9
Table 4: Class card of SmellIdentifier class .....	10
Table 5: Class card of ReportGenerator Class.....	10
Table 6: Class card of CSVGenerator Class.....	10
Table 7: Class card of ToolManager Class .....	10
Table 8: Server Requirements for UI Analyzer.....	21
Table 9: Supported Browsers for different OS.....	21
Table 10: Function description of WebExplorer class.....	22
Table 11: Function description of EventPreprocessor class .....	23
Table 12: Function description of SmellIdentifier class .....	24
Table 13: Function description of ReportGenerator class .....	24
Table 14: Function description of CSVGenerator class.....	25
Table 15: Function description of ToolManager class .....	25
Table 16: Test specifications of T1 .....	28
Table 17: Test specifications of T2 .....	28
Table 18: Table 8: Test specifications of T2 .....	29

## 1 Introduction

Web applications help us in many of our daily life activities, like shopping, news reading, social interaction, home banking, trip planning or requesting a doctor's appointment. Every day new websites appear broadening our possibilities to accomplish tasks comfortably from home, and yet many times they suffer from usability problems that make them awkward and hard to use. One of the most popular ways of evaluating usability is by conducting usability tests, particularly, user tests. The benefit of user testing over inspection methods like heuristic evaluations is that it captures real usage data and users' experiences. The down-side, however, is that it requires recruiting users and spending time and resources for experts first to design the tests and afterwards to analyze the results, discover the problems and find solutions for those problems.

[1]

UIAnalyzer tool can provide automatic advice about usability smells of user interaction for deployed web applications. The automated strategy to usability smell recognition is based on the analysis of user interaction (UI) events, linking specific UI events to usability smells, and reporting usability smells which makes it possible to suggest solutions for them in terms of refactoring.

## 2 Scope of the Project

- This tool will not analyze all types of website. Only static websites can be analyzed using UIAnalyzer.
- This tool also will not crawl pages which needed authentication.
- Only defined usability smell can be found using this tool.
- The highest limit page limit for a single website is 50.

## 3 Project Description

### 3.1 Specific Requirements

The specific requirements of this project are –

#### 3.1.1 Functionality

- The system will analyze the user interface of a web application.
- This will identify the usability smell from the user interface.
- This will provide the details of identified smell with certain parameters that determine the number, proportion of elements that trigger a specific smell.
- The system will also provide suggestion to refactor the UI code that will solve the usability smell.
- The system will provide a complete report of the findings which can be downloaded as CSV files.

#### 3.1.2 Usability

- The system shall provide a uniform look and feel between all the web pages.
- The graphical user interface will contain interactive charts.

#### 3.1.3 Performance

- The product shall be based on web and has to be run from a web server.
- The tool shall take initial load time depending on internet connection strength which also depends on the media from which the tool is run.
- The performance shall depend upon hardware components of the client/customer.

#### 3.1.4 Web Based Requirements

- There are no memory requirements
- The computers must be equipped with web browsers such as Internet explorer.
- The product must be stored in such a way that allows the client easy access to it.
- A general knowledge of basic computer skills is required to use the product.

## 3.2 Usage scenario of UIAnalyzer

UIAnalyzer will be a web based tool that will work with web application. This tool will find usability smells from the user interface of a web application and decide whether the web interface follows standard user interface pattern or not. [3] The automated strategy to usability smell identification is based on a process consisting of three steps: Events Logging, Usability Smells Detection, and Reporting.

### 3.2.1 Events logging

First the user will enter the URL of the application which will be analyzed. Then the events logging step will be started. The Events Logging step will be implemented by an automated selenium script that intercepts selected low-level UI events and parses the html raw code. Whenever the script loads a page, it will start analyzing low-level events like page scrolling, form submission. It will then processes these events according to different criteria and generates higher-level usability events for further analysis. The script will be able to capture usability events when it crawls across different pages, such as the DOM load time, page height, input fields, form validation, response time of a request.

The usability events and their related usability smells are given below-

Table 1: Usability Smells [1]

<b>Event/Elements</b>	<b>Usability Smell</b>
Clickable elements (anchor tag, button)	Undescriptive Element
Flash Scrolling	Overlooked Contents
Link	No link, Broken link
Text Input	Unformatted Input
Request	Long request

1. No link, broken link: This smell triggers when an anchor tag has no href attribute. Also if the link does not exist, the tool marked it as a broken link [1].
2. Flash Scrolling: This smell triggers when a page height is more than 1556px.
3. Long Request: If any server request takes longer than 3 ms, this will occur a long request.
4. Undescriptive Element: If any anchor tag or button has not any descriptive text, image or icon on it, it will be marked as an undescriptive element [1].
5. Unformatted Input: When any input field has no formatting attributes like maxlength, type etc. it will be marked as an unformatted input [1].

### 3.2.2 Smell Detection

The Usability Smell Detection step will take place at the server, using the different kinds of usability smell finders. Each kind of finder will detect a specific type of usability smell, and each application under analysis will have its own set of finders with potentially different configurations. A finder classifies the detected usability smells by a common criterion, generally by the affected DOM element, but also by URL or URL sequence. When smells are classified by DOM element, it's important to note that a single HTML template may be used to generate different but equivalent DOM elements [1].

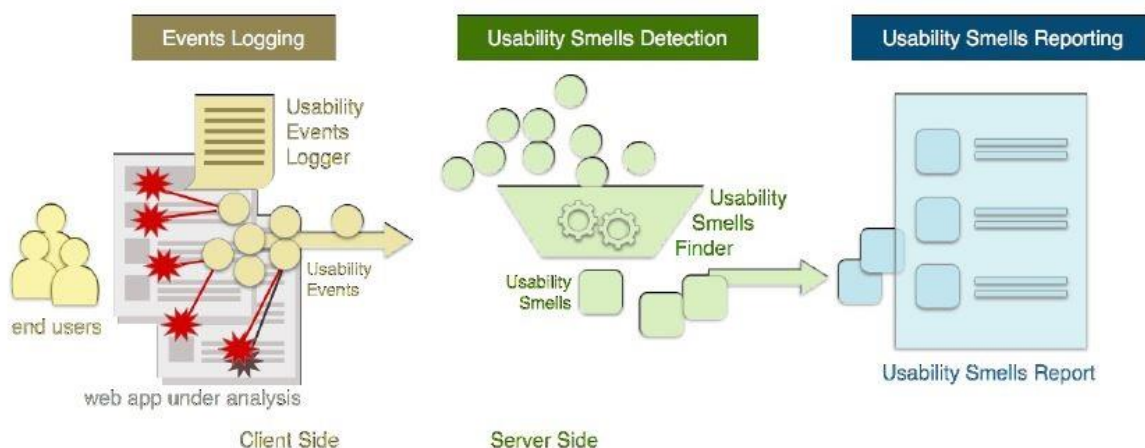


Figure 1: The three steps of the UIAnalyzer System

### 3.2.3 Reporting

This tool will report bad smells as they appear. It will show detailed information on every bad smell, including when possible a live view of the affected widget. Each usability smell shows specific data for a better understanding of the problem. The user can also be able to download the report in CSV format. This report will evaluate the application whether the UI patterns are followed properly or not.

The steps of the whole system is shown in figure 1.



## 4 Scenario-Based Modeling

### 4.1 Use-case Diagram

Use Case diagrams give the non-technical view of the overall system.

Level-0 Use case Diagram -UIAnalyzer is shown in figure 2.

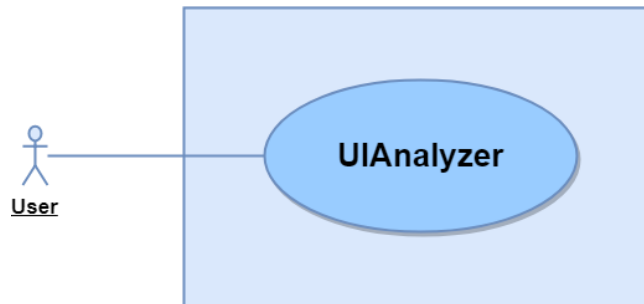


Figure 2: Level-0 Usecase Diagram

Name	UIAnalyzer
ID	UIANALYZER-L-0
Primary Actors	User

Level-1 Use case Diagram -UIAnalyzer is shown in figure 3.

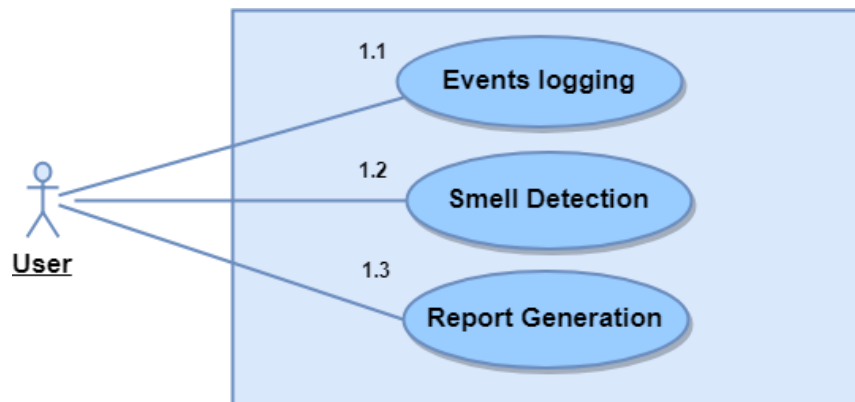


Figure 3: Level-1 Usecase Diagram

Name	Subsystems of UIAnalyzer
ID	UIANALYZER-L-1
Primary Actors	User

### **Description of Use Case Diagram Level 1:**

UIAnalyzer will be a web based tool that will work with web application. This tool will find usability smells from the user interface of a web application and decide whether the web interface follows standard user interface pattern or not. The automated strategy to usability smell identification is based on a process consisting of three steps:

1. Events Logging,
2. Usability Smells Detection,
3. Reporting.

### **Action-Reply of Use Case Diagram Level 1:**

Action 1: User will invoke the software

Reply 1: System will start the software

Action 2: User will enter the site URL of the web application

Reply 2: System will start event logging. If the event is found, it will record it as a smell.

Action 3: User will choose option for report.

Reply 3: The system will generate the refactoring suggestion and statistical report.

Action 4: User will choose option for CSV

Reply 4: The CSV will be generated and downloaded.

## 4.2 Activity Diagram

The activity diagram of the total system is shown in figure 4. In this diagram, the basic activity flow of the software is shown in brief.

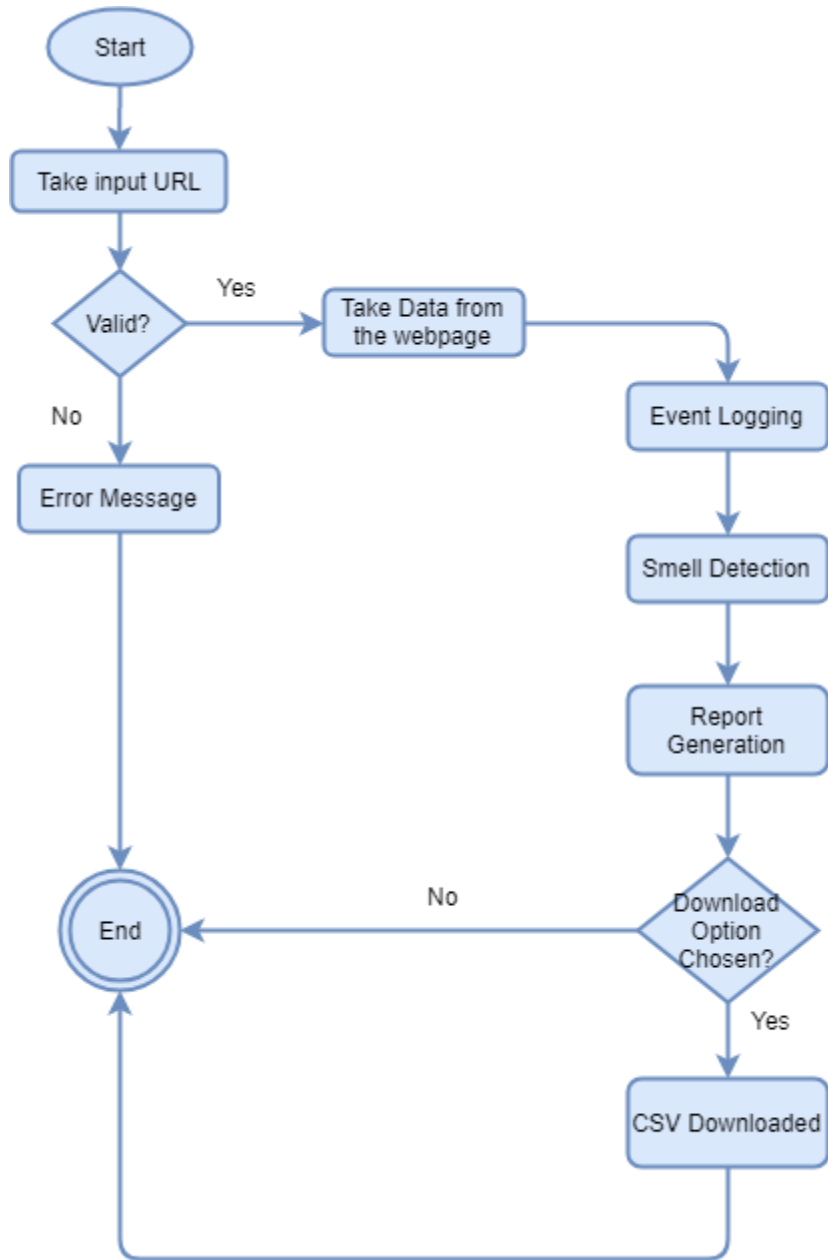


Figure 4: Activity Diagram of Whole System

## 5 Class-Based Modeling

### 5.1 Final Classes

The final classes are identified from the scenario of this project. Those are:

1. WebExplorer
2. EventPreprocessor
3. SmellIdentifier
4. ReportGenerator
5. CSVGenerator
6. ToolManager

The class cards of these classes are shown in tables below:

Table 2: Class card of WebExplorer Class

1.WebExplorer	
Attributes	Methods
-	getURL(), fetchingData(), parseData()

Table 3: Class card of EventProcessor class

2.EventPreprocessor	
Attributes	Methods
actionList	getAction(), parseAction(), ExtractEvent()

Table 4: Class card of SmellIdentifier class

3.SmellIdentifier	
Attributes	Methods
smellName, targetElement	getEvent(), parseEvent(), CalculateDistanceBetweenElement(), getAffectedElement(), ExtractSmell()

Table 5: Class card of ReportGenerator Class

4.ReportGenerator	
Attributes	Methods
smellList, urlList	readSmellList(), generateRefactoringSuggestion(), generateChart()

Table 6: Class card of CSVGenerator Class

5.CSVGenerator	
Attributes	Methods
fileName filePath	getSummaryReport(), generateCSV()

Table 7: Class card of ToolManager Class

6.ToolManager	
Attributes	Methods
-	initiateWebExplorer(), initiateSmellIdentifier(), initiateReportGenerator()

## 5.2 Class Diagram

The class diagram of the project “UIAnalyzer” is shown in figure 5.

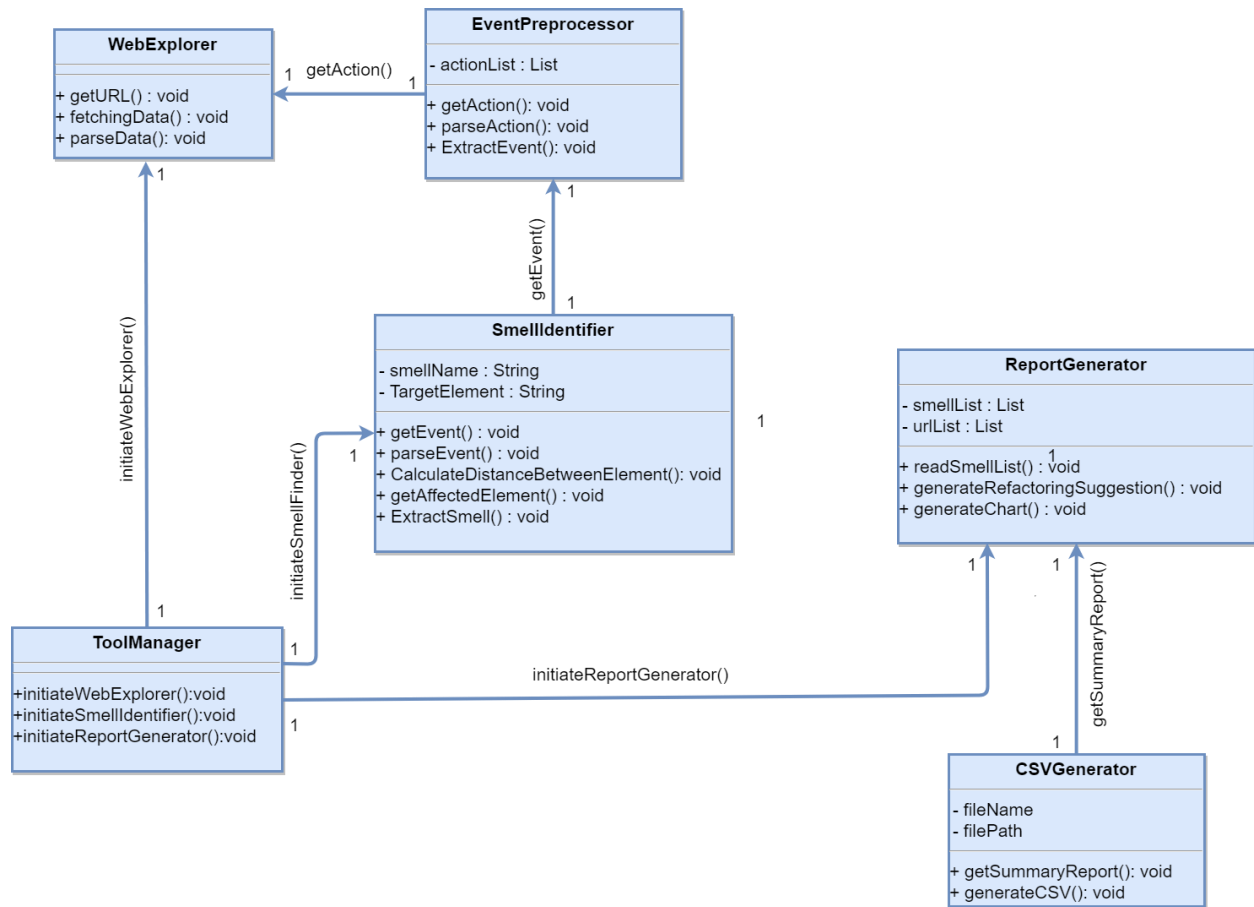


Figure 5: Class Diagram of UIAnalyzer

## 6 Architectural Design

### 6.1 Overview

The application is based on 2-tier architecture. The software is divided into a presentation layer and a logic layer (figure 6). There is no persistence layer because no database is needed.

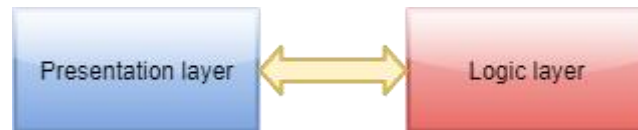


Figure 6: 2-tier architecture of UIAnalyzer

The presentation layer is where all the user interactions take place. The presentation layer communicates with the logic layer. Here the logic layer will be a REST API that provides URL endpoints for the presentation layer to communicate. Through the logic layer the inputs are processed and information are returned to the client which is shown in figure 7.

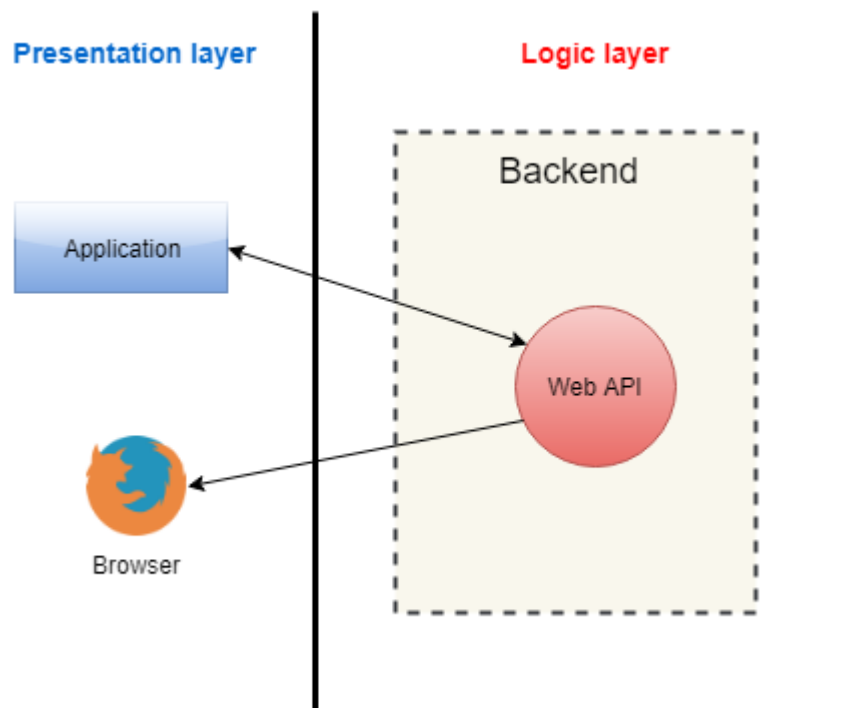


Figure 7: Archetctural overview of UIAnalyzer

When the client requests the server for the application a webpage eill be loaded. Later on the request to analyze a web application will be AJAX calls. The server will reply in JSON.

## 6.2 Instantiations of the System

The instantiations of the whole system is given below:

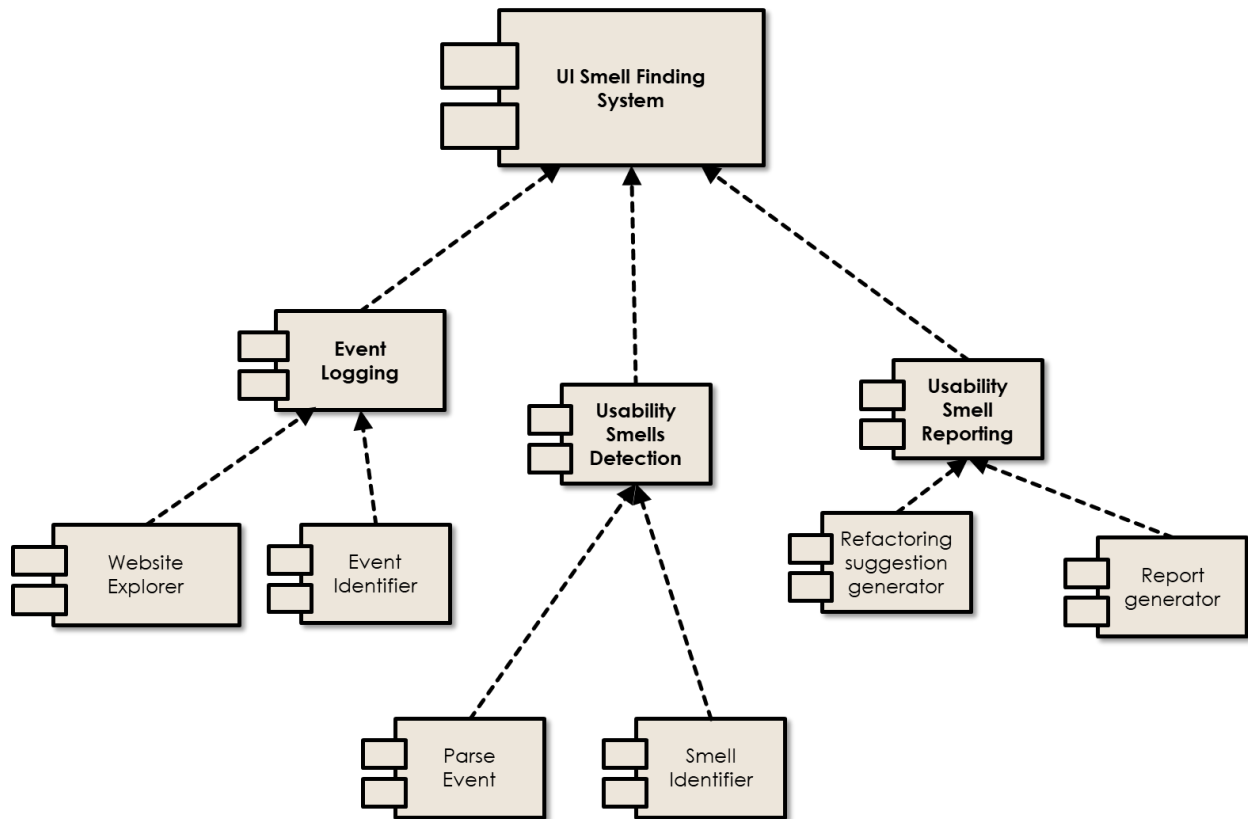


Figure 8: Instantiations of UIAnalyzer

## 6.3 Elaborated Deployment

The elaborated deployment of the system is given below:

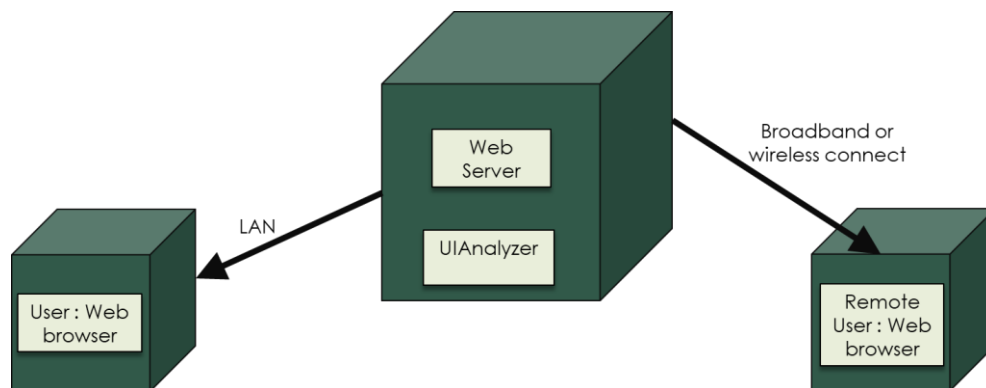


Figure 9: Elaborated Deployment of UIAnalyzer



## 7 User Interface Design

The mock graphical interface design is provided in the following part. When the user will initiate the tool, s/he will see the first page shown below.

The image shows a mock graphical interface design for the home page of UIAnalyzer. It consists of four main sections arranged vertically:

- Welcome**: A light green rectangular box at the top.
- Header**: A light green rectangular box below the Welcome section.
- Main Content Area**: A large light green rectangular box containing:
  - An input field with the placeholder text "Enter URL".
  - A rounded rectangular button labeled "Submit" below the input field.
- Footer**: A light green rectangular box at the bottom.

Figure 10: Home page of UIAnalyzer

First the user will enter the URL of the application which will be analyzed. Then the events logging step will be started.

Then the application will be under analysis state. The actions performed by the user will be recorded for event identification.



Figure 11: Application Under Analysis View

After the application under analysis step, the system will detect the usability smell if there is any smell. Each smell will contain some information like smell name, specific element and refactoring suggestion.

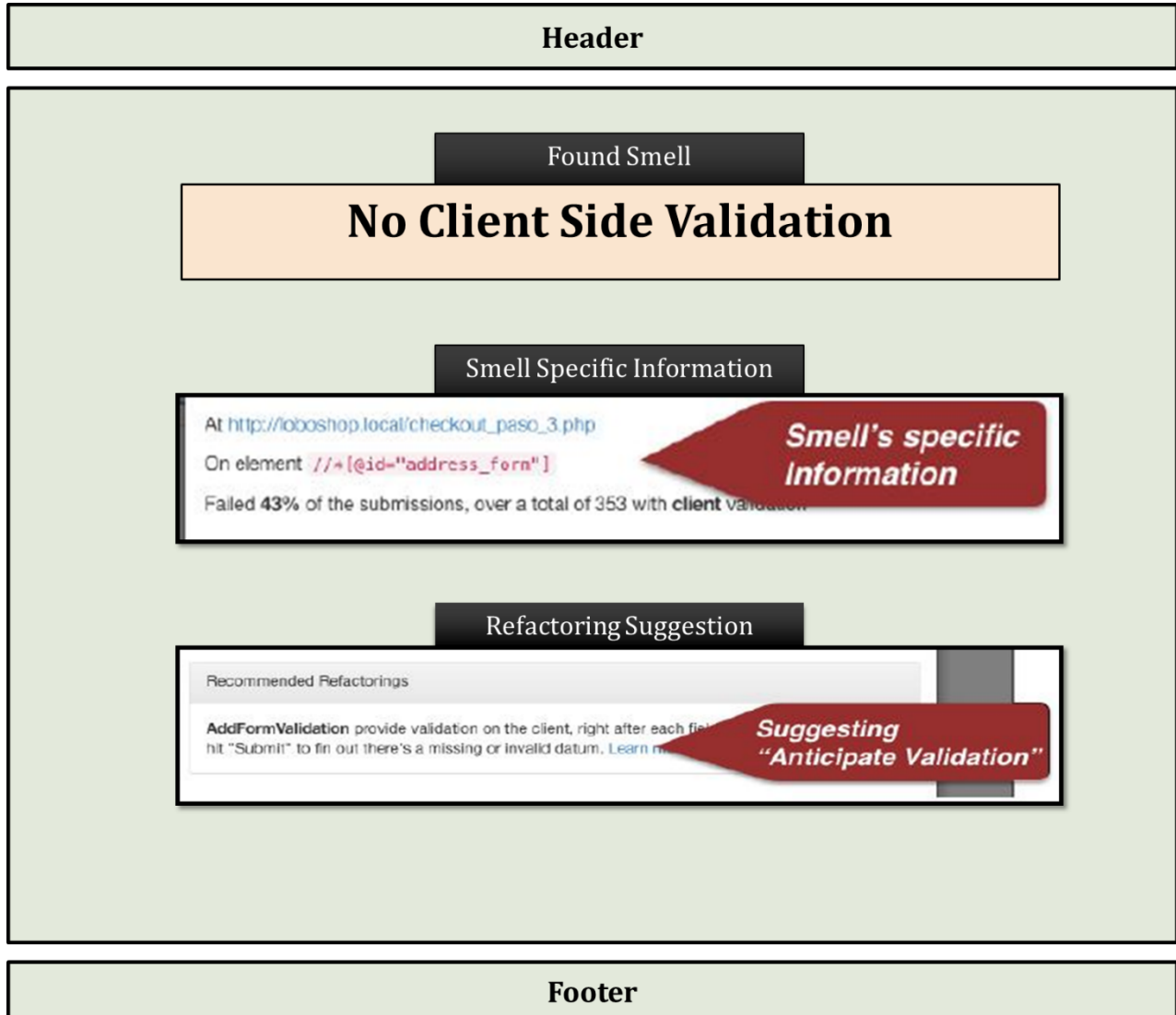


Figure 12: Smell Specific View

Then the user can see a summary report of the application h/she has wanted. The summary report will contain different types of usability smells found on that system. The use can download the report in CSVformat if he/she wants to.

**Header**

Summary Report

### Usability Smells

Found 10 smells.

[Recalculate](#)   [Reveal 3 Ignored Smells](#)

<div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 5px;"> <b>Late Validation (client side)</b> ⓘ <small>http://loboshop.local/checkout_paso_3.php</small> <code>//*[@id="address_form"]</code> <small>Ignore   Reset stats</small></div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 5px;"> <b>Late Validation (client side)</b> ⓘ <small>http://loboshop.local/registro.php</small> <code>//*[@id="registration_form"]</code> <small>Ignore   Reset stats</small></div> <div style="border: 1px solid #ccc; padding: 5px;"> <b>Free Input for Limited Values</b> ⓘ <small>http://loboshop.local/checkout_paso_3.php</small></div>	<div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 5px;"> <b>Short Input</b> ⓘ <small>http://loboshop.local/checkout_paso_2.php</small> <code>//*[@id="email"]</code> <small>Ignore   Reset stats</small></div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 5px;"> <b>Unresponsive Element</b> ⓘ <small>http://loboshop.local/productos.php</small> <code>//*[@id="categories-header"]</code> <small>Ignore   Reset stats</small></div> <div style="border: 1px solid #ccc; padding: 5px;"> <b>Unresponsive Element</b> ⓘ <small>http://loboshop.local/carrito.php</small></div>
---	---

Download

**Footer**

Figure 13: Summary Report View

## 8 Implementation Overview

This chapter aims to describe the implementation process of “UIAnalyzer”. Here the technologies that have been used to develop this system will be described in brief. Implementation is the stage in the project where the theoretical design is turned into a working system.

### 8.1 Technology Used in implementation

Development technologies are growing very rapidly with the increase of requirements. The technologies that have been used to develop this system is the most recent technologies and also very much appropriate to it.

The whole implementation has two parts- the web api for data communication and the UI for presentation. For serving the client I made a REST API that receives client requests and serves JSON data (webpage analysis data, smell specific data). This API is written in Python. Python is very suitable for designing prototypes as a working implementation can be produced in minimal code. Not only is it concise it's also easy for others to read. I also used Flask framework for making the web API. Flask is a micro-framework written in Python. I considered Django but realized it would be too heavy for my project. Flask is a lot more lightweight framework and is easier to use.

#### 8.1.1 Client-side Technology

The user interface is the working environment for the user. Various languages and libraries have been used to develop this project. They are:

- **Hyper Text Markup Language (HTML)**

Hyper Text Markup Language (HTML) is the main markup language for web pages. HTML elements are the basic building-blocks of a webpage. The latest HTML5 has been used for developing this system.

- **Cascading Style Sheets (CSS)**

Cascading Style Sheets (CSS) is a stylesheet language used for describing the presentation of a document written in a markup language. Although most often used to set the visual style of web pages and user interfaces written in HTML. CSS3 have been used for developing this system.

- **JavaScript (JS)**

JavaScript is a high-level, dynamic and interpreted run-time language. Alongside HTML and CSS, JavaScript is one of the three core technologies of World Wide Web content production. The majority of websites employ it and all modern Web browsers support it without the need for plug-ins.

- **Bootstrap (front-end framework)**

Bootstrap is a free and open-source collection of tools for creating websites and web applications. It contains HTML- and CSS-based design templates for typography, forms, buttons, navigation and other interface components, as well as optional JavaScript extensions.

### 8.1.2 Server-side Technology

For back-end coding, various languages and libraries have been used-

- **Python**

Python is an interpreted high-level programming language for general-purpose programming. Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

In this project, I have used **Python 3.6.4**

- **Flask**

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools

In this project, **Flask 0.12.2** is used.

- **Beautiful Soup**

Beautiful Soup is a Python package for parsing HTML and XML documents (including having malformed markup, i.e. non-closed tags, so named after tag soup). It creates a parse tree for parsed pages that can be used to extract data from HTML, which is useful for web scraping.

In this project, I have used **beautifulsoup 4.6.3**

- **Requests**

Requests is a Python HTTP library, released under the Apache2 License. The goal of the project is to make HTTP requests simpler and more human-friendly.

For UIAnalyzer, I have used **requests 2.18.4**

- **Selenium WebDriver**

Selenium is a portable software-testing framework for web applications. Selenium WebDriver accepts commands and sends them to a browser. This is implemented through a browser-specific browser driver, which sends commands to a browser and retrieves results. Most browser drivers actually launch and access a browser application (such as Firefox, Chrome, Internet Explorer, Safari, or Microsoft Edge).

In this project I have used **Selenium ChromeDriver 2.44**

### 8.1.3 Implementation Tools

As types of software are increasing day by day, new implementation tools are also needed for their implementation. Nowadays, there are many implementation tools. Developers have to choose right tools for each part of their application. If they can utilize tools perfectly, their labor can be reduced.

- **Visual Studio Code**

Visual Studio Code is a source code editor developed by Microsoft for Windows, Linux and macOS. It includes support for debugging, embedded Git control, syntax highlighting, intelligent

code completion, snippets, and code refactoring. It is also customizable, so users can change the editor's theme, keyboard shortcuts, and preferences. It is free and open-source.

For development purpose, I have used Visual Studio Code 1.29

- **SourceTree**

SourceTree is a Git and Mercurial desktop client for developers on Mac or Windows. To conduct the project I have used Git. To use Git easily, I have used SourceTree.

## 8.2 System Configuration

This section has shown the minimum and recommended requirement for UIAnalyzer both server and client.

### 8.2.1 Server Requirements

Table 8: Server Requirements for UI Analyzer

<b>Windows 10/8/7 (incl.64-bit)</b>	<b>Linux</b>
1 GB RAM minimum, 2 GB RAM recommended	512 MB RAM minimum, 1 GB RAM recommended
Python 2.7 or higher, Python 3.6 recommended	Python 2.7 or higher, Python 3.6 recommended

### 8.2.2 Supported Browsers

Table 9: Supported Browsers for different OS

	<b>Chrome</b>	<b>Firefox</b>	<b>Internet Explorer</b>	<b>Microsoft Edge</b>
<b>Mac</b>	Supported	Supported	N/A	N/A
<b>Windows</b>	Supported	Supported	Supported, IE10+	Supported



### 8.3 Source Code Description

There were six analytic classes from the SRS document. Those are-

1. WebExplorer
2. EventPreprocessor
3. SmellIdentifier
4. ReportGenerator
5. CSVGenerator
6. ToolManager

#### 8.3.1 WebExplorer class

WebExplorer class is a class where a website link is taken as input, crawls the link, parse the information.

**Attributes:** This class has three attributes. They are:

1. homeurl,
2. scanned\_urls[],
3. nonscanned\_urls[]

**Functions:** The functions are given in the following Table 10

Table 10: Function description of WebExplorer class

Functions	Description
getURL(url),	This function capture the given url. Initiates parsing
checkEligibile(url)	This function checks eligibility of website url
getParse()	This function extracts all links from a webpage. Also decides the link should be in scanned_urls or not by checking the type of a link (e.g #, http, https)
crawlSite()	The function visits all possible way into a website using the getParse() and checkEligibile() method

### 8.3.2 EventPreprocessor class

EventPreprocessor class initiates selenium driver, checks the attributes of html elements, captures request time and page height of a webpage.

**Attributes:** This class has three attributes. They are:

1. elements[]
2. pageHeights[],
3. responseTimes[]

**Functions:** The functions are given in the following Table 11

Table 11: Function description of EventPreprocessor class

Functions	Description
initiateSelenium()	This function initiates selenium driver for start event execution
parseAction(event)	This function takes event as parameter, parse the action and returns parsed elements[]
getPageHeight(url)	This function gets the page height of a webpage
getResponseTime()	This function extracts response time for a server request event

### 8.3.3 SmellIdentifier class

SmellIdentifier class initiates smell identification, get all parsed information about pages, elements and events and decides what type of smell is found..

**Attributes:** This class has two attributes. They are:

1. smells[]
2. pageUrls[],

**Functions:** The functions are given in the following Table 12

Table 12: Function description of SmellIdentifier class

Functions	Description
getParsedAction(event)	This function gets the parsed action and records the extracted information
getParsedElement(element)	This function gets the parsed element record the extracted information
IdentifySmell(element, event)	It takes parsed element and event and decides what type of smell it contains

#### 8.3.4 ReportGenerator class

ReportGenerator class initiates report generation, generate refactoring suggestion and generate summary report.

**Attributes:** This class has two attributes. They are:

1. smells[]
2. pageUrls[],

**Functions:** The functions are given in the following **Error! Reference source not found.**

Table 13: Function description of ReportGenerator class

Functions	Description
getSmellList()	From SmellIdentifier class this function gets all smell found in a specific website with page url, element list and event list
generateRefactoringSuggestion(smell)	This function generates refactoring suggestion for each smell found
generateAverage(smells[], url)	This function gives the average smell found per page

### 8.3.5 CSVGenerator class

CSVGenerator class serves the purpose of generating CSV files for all specific smell and download the files.

**Attributes:** This class has two attributes. They are:

1. filePath
2. fileName

**Functions:** The functions are given in the following Table 14

Table 14: Function description of CSVGenerator class

Functions	Description
generateCSV()	This function gets the result from ReportGenerator class and generates CSV
downloadCSV(filename, filePath)	This function downloads the CSV

### 8.3.6 ToolManager class

ToolManager class initiates the tool, starts the server and serves different routes.

**Attributes:** This class has no attribute.

**Functions:** The functions are given in the following Table 12

Table 15: Function description of ToolManager class

Functions	Description
initiateServer()	This function starts the server
serveRoute(routes)	This function serves specific routes for different purposes

## 9 Test Plan, Test Cases and Automated Testing

### 9.1 Test Item to be tested

**Name:** UIAnalyzer

**Type:** Web Application

**Version:** 1.0

### 9.2 Features to be tested

- UI Smell identification
- Provide the details of identified smell with certain parameters that determine the number, proportion of elements that trigger a specific smell.
- Report generation of the findings which can be downloaded as CSV files.

### 9.3 Features not to be tested

There is no such feature. All the features will be tested.

### 9.4 Approach

Since the test item is a website, I have tested it locally using the Python Flask server. I have divided the functionalities into modules and test each feature using **Black Box Testing**. This testing will be done using Selenium (A portable software-testing framework for web applications). The test cases will be prepared with **Robustness Testing**. The input specifications will be prepared using the specifications acquired from the SRS.

### 9.5 Item Pass Fail Criteria

If 90% of the prepared tested cases pass, then it is considered the project as a whole to pass.

### 9.6 Test Deliverables

The deliverable documents are:

- 1) Test plan
- 2) Test case specifications

## 9.7 Testing Tasks

**Preparing and setting up system environment:** Before executing the test cases I have to prepare the server, which is a local Python Flask server.

**Tasks for test case:** The features will be divided into three modules of testing.

- 1) Testing Events Logging
- 2) Testing Smell identification
- 3) Testing Report Generation

## 9.8 Testing cost

**Cost to support the environment:** Since it is a web-based software, I need internet connection to support the environment. So, the cost to maintain internet connection in order to continue the testing.

**Cost of executing the tests:** I did not need any executing test cost.

## 9.9 Test design specification

S. No	Functionality	Use case no in SRS	Test cases
1	Check valid website url	1	T1
2	Smell identification	1	T2
3	Report generation	1	T3

Reference Usecase:

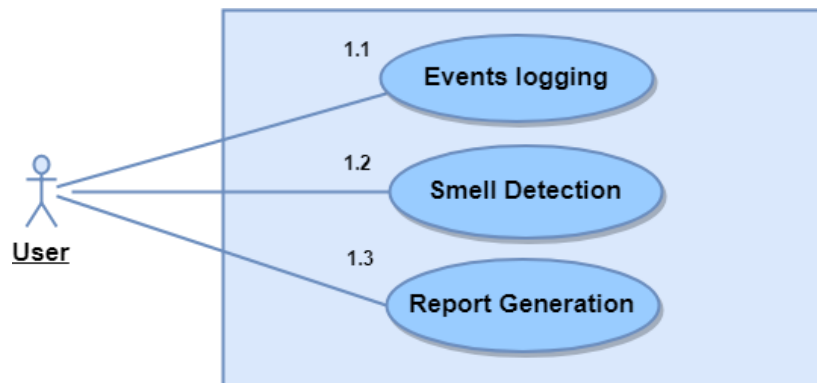


Figure 14: Level-1 Usecase Diagram

## 9.10 Test case specifications

Table 16: Test specifications of T1

<b>Test case Specification Identifier</b>	T1
<b>Purpose</b>	To check the functionality of 'Check valid website url'
<b>Test Items Needed</b>	Refer Usecase-1 in SRS
<b>Special Environmental</b>	Internet should be in working condition.
<b>Special Procedural Requirements</b>	-
<b>Inter-case Dependencies</b>	-
<b>Input Specifications</b>	Enter URL as given below: data.gov.abc www.gov
<b>Test Procedure</b>	Press 'Enter' button
<b>Expected system response</b>	Redirect to error page
<b>Pass/Fail</b>	Pass

Table 17: Test specifications of T2

<b>Test case Specification Identifier</b>	T2
<b>Purpose</b>	To check the functionality of 'Smell identification'
<b>Test Items Needed</b>	Refer Usecase-1 in SRS
<b>Special Environmental</b>	Internet should be in working condition.
<b>Special Procedural Requirements</b>	-
<b>Inter-case Dependencies</b>	-
<b>Input Specifications</b>	Enter URL as given below: data.gov.bd
<b>Test Procedure</b>	Press 'Enter' button
<b>Expected system response</b>	Redirect to analysis page and show smell report
<b>Pass/Fail</b>	Pass

Table 18: Table 8: Test specifications of T2

<b>Test case Specification Identifier</b>	T3
<b>Purpose</b>	To check the functionality of 'Report generation'
<b>Test Items Needed</b>	Refer Usecase-1 in SRS
<b>Special Environmental</b>	Internet should be in working condition.
<b>Special Procedural Requirements</b>	-
<b>Inter-case Dependencies</b>	T2 test case must be executed prior to the current test case execution.
<b>Input Specifications</b>	-
<b>Test Procedure</b>	Press 'Download CSV button
<b>Expected system response</b>	A CSV file will be download with specific smell result
<b>Pass/Fail</b>	Pass

### 9.11 Test case execution using Selenium

I have used Selenium IDE to execute the test cases. Selenium is a portable software-testing framework for web applications. Steps to follow how I have executed the test cases using Selenium IDE (Chrome Extension).

#### 1. Localhost Server Setup Steps:

- a. First you have to clone the UIAnalyzer project. The github link of the project is given below: <https://github.com/AquibAzmain/UIAnalyzer.git>
- b. You need python 2.7 or higher install in your machine. Windows 7+, Linux is recommended.
- c. Go to the root folder of the project and run the command 'pip install --r requirements.txt'. It will install all dependencies of this project.
- d. Finally you have to run the command 'python run.py'. It will start a server on the url 'localhost:9000'. Just go to browser, enter the url and hit enter.



2. Test Environment:

- Operating System: Windows 10
- Browser: Google Chrome Version 70.0.3538.102 (Official Build) (64-bit)
- Selenium IDE: Version 3.4.4
- Proper internet connection

3. Download and add Selenium IDE into Google Chrome if it is not added. Download link is given below:

[chrome.google.com/webstore/detail/selenium-ide/mooikfahbdckldjndioackbalphokd?](https://chrome.google.com/webstore/detail/selenium-ide/mooikfahbdckldjndioackbalphokd?)

4. Run Selenium IDE.

5. Add a new project named 'UIAnalyzer'. Enter home url: localhost:9000

6. Add a new test case named 'T1'

7. Add new commands and execute the test case (Figure 15) and get the error page.

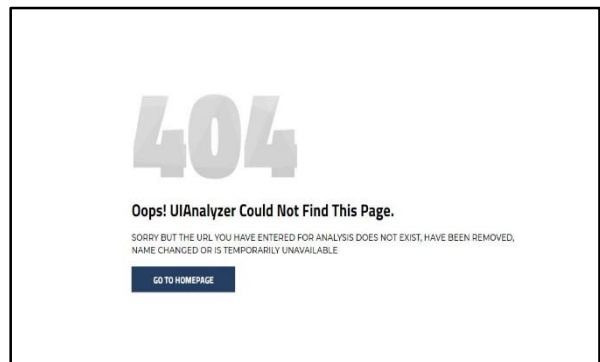
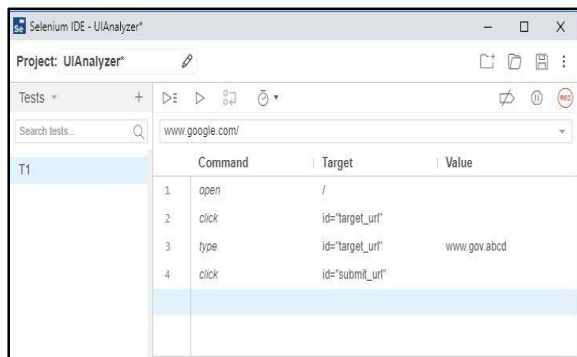


Figure 15: Selenium test case 1 and output

8. Then create new test case T2 and execute it. It will redirect to the analysis page (Figure 16).

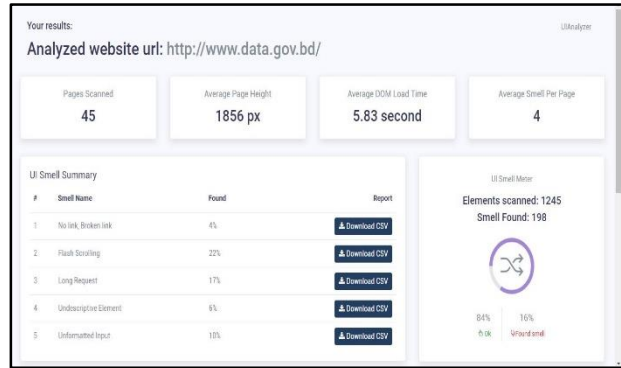
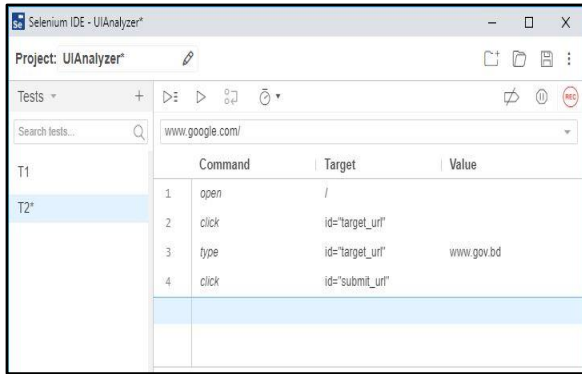


Figure 16: Selenium test case 2 and output

9. And lastly create and execute test case T3. For this you have to run the test case T2 first. A CSV file will be downloaded.

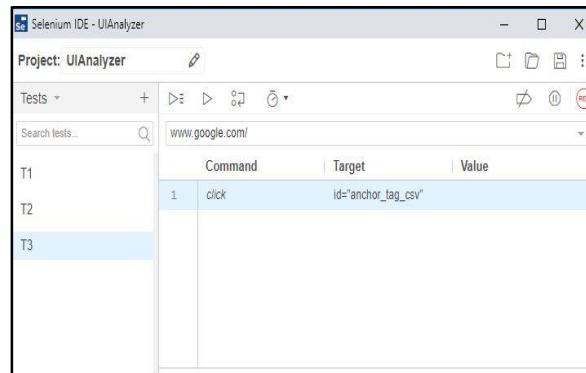
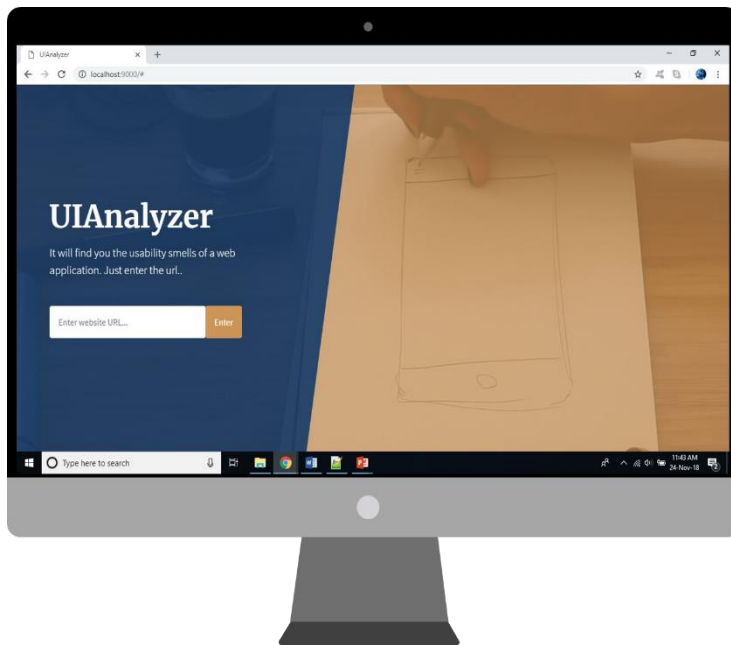


Figure 17: Selenium test case 3

## 10 User Manual



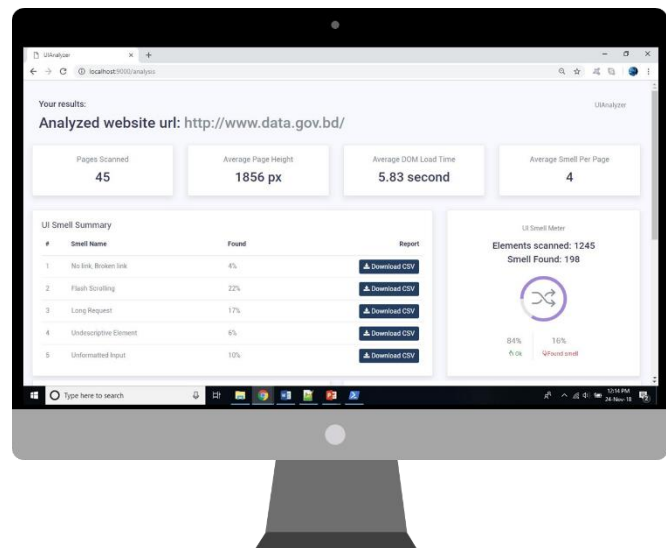
### What is UIAnalyzer?

UIAnalyzer is a web-based analytics tool that identifies and reports website UI smell. We can use these reports to get actionable insights and use them to take steps to improve our website.

### Why should we use it?

UIAnalyzer helps you analyze front end code and performance of a web application and paint a complete picture of the user interface. In a nutshell, UIAnalyzer provides information about:

- What kind of usability smell found in your website
- Page height
- DOM load time
- How many pages scanned
- How many html elements contain smell



## How to use UIAnalyzer?

1. Open your browser, type the url: **localhost:9000** and hit enter (Figure 18) . If the server is running on a different machine, you have to enter the ip address of that machine (e.g 10.100.102.145:9000).



Figure 18: Enter url of UIAnalyzer

2. You will get the homepage of UIAnalyzer (Figure 19). You will have to enter the desired website url (e.g data.gov.bd) in the marked text filed and hit enter.

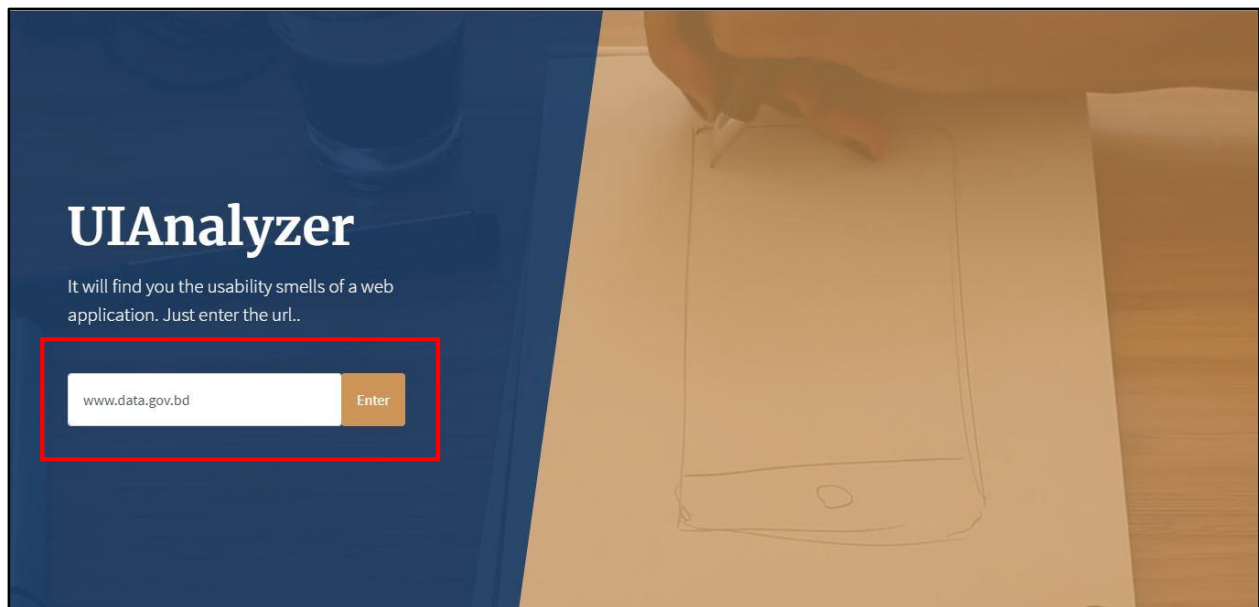


Figure 19: Homepage of UIAnalyzer

3. If you enter an invalid website url which does not exist, have been removed, name changed or temporarily unavailable you will get error message (Figure 20).

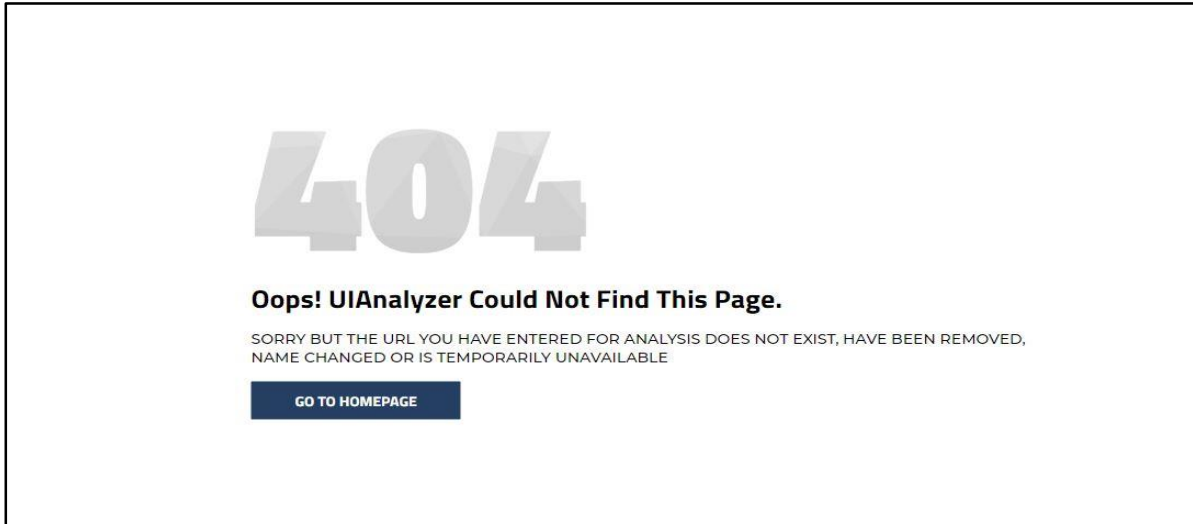


Figure 20: Error page

4. If you enter a valid url you will get the analysis result page (Figure 21).

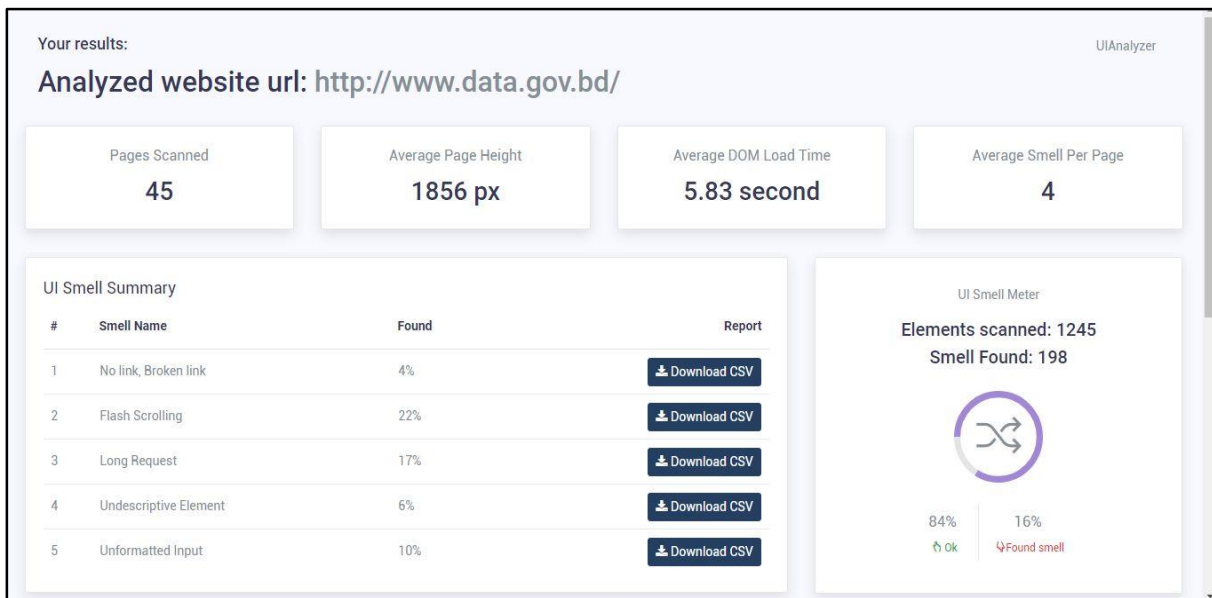


Figure 21: Analysis result

## Analysis page Interface

Analysis result page is divided into 5 sections. They are:

1. Overall analysis
2. UI Smell Summary
3. UI Smell Meter
4. Screenshot of analyzed website homepage
5. List of pages scanned

The screenshot shows the UIAnalyzer interface for the website <http://www.data.gov.bd/>. The interface is divided into five numbered sections:

- Section 1: Overall analysis** - A summary of key metrics:

Metric	Value
Pages Scanned	45
Average Page Height	1856 px
Average DOM Load Time	5.83 second
Average Smell Per Page	4
- Section 2: UI Smell Summary** - A table listing detected UI smells:

#	Smell Name	Found	Report
1	No link, Broken link	4%	<a href="#">Download CSV</a>
2	Flash Scrolling	22%	<a href="#">Download CSV</a>
3	Long Request	17%	<a href="#">Download CSV</a>
4	Undescriptive Element	6%	<a href="#">Download CSV</a>
5	Unformatted Input	10%	<a href="#">Download CSV</a>
- Section 3: UI Smell Meter** - A gauge showing the overall quality score:
  - Elements scanned: 1245
  - Smell Found: 198
  - 84% OK (Green)
  - 16% Found smell (Red)
- Section 4: Screenshot of analyzed website homepage** - A screenshot of the Bangladesh Open Data website homepage.
- Section 5: List of page visited** - A table listing the pages scanned during the analysis:

#	URL	DOM Load Time	Page Height
1	<a href="http://data.gov.bd/">http://data.gov.bd/</a>	4.5s	1456 px
2	<a href="http://data.gov.bd/dataset/">http://data.gov.bd/dataset/</a>	5.52s	1956 px
3	<a href="http://data.gov.bd/stories">http://data.gov.bd/stories</a>	4.24s	1240 px
4	<a href="http://data.gov.bd/about-us">http://data.gov.bd/about-us</a>	5.7s	1010 px
5	<a href="http://data.gov.bd/">http://data.gov.bd/</a>	4.5s	1456 px
6	<a href="http://data.gov.bd/search-type">http://data.gov.bd/search-type</a>	5.96s	778 px
7	<a href="http://data.gov.bd/dataset/top?">http://data.gov.bd/dataset/top?</a>	7.6s	1920 px

Developed by Aquib Azmain

Figure 22: 5 sections of the analysis page

## Overall analysis view

From this section you can get overall analysis information about the target website. Like:

1. Website url: The target website url you have entered for analysis
2. Pages scanned: The number of pages scanned. The url of these pages has been extracted from the target website by UIAnalyzer.
3. Average Page Height: This shows you the average height of all the pages scanned of the website.
4. Average DOM Load Time: This will give you the average DOM processing time of all pages of the website. DOM processing is the time it takes to parse the HTML into a DOM and retrieve or execute synchronous scripts. [6]
5. Average Smell: This shows average smell number found in a particular page. It indicates the quality of the pages of the website.



Figure 23: Overall analysis view

## Smell Summary View

This section (Figure 24) will show you what type of UI smell is found in the website and also show the found percentage. The “Download CSV” button is available to download the detailed result of the analysis. The smells that can be found by UIAnalyzer is given below:

6. No link, broken link: This smell triggers when an anchor tag has no href attribute. Also if the link does not exist, the tool marked it as a broken link. The found % of this smell follows the formula:

$$\frac{\text{no link count} + \text{broken link count}}{\text{all links found}} \times 100$$

7. Flash Scrolling: This smell triggers when a page height is more than 1556px. The found % of this smell is measured by the formula:

$$\frac{\textit{flash scroll count}}{\textit{page count}} \times 100$$

8. Long Request: If any server request takes longer than 3 ms, this will occur a long request. The found % of this smell is measured by the formula:

$$\frac{\textit{long request count}}{\textit{scanned request count}} \times 100$$

9. Undescriptive Element: If any anchor tag or button has not any descriptive text, image or icon on it, it will be marked as an undescriptive element. The found % of this smell is measured by the formula:

$$\frac{\textit{undescriptive element count}}{\textit{anchor element count} + \textit{button count}} \times 100$$

10. Unformatted Input: When any input field has no formatting attributes like maxlength, type etc. it will be marked as an unformatted input. The found % of this smell is measured by the formula:

$$\frac{\textit{unformatted input field count}}{\textit{total input field count}} \times 100$$

UI Smell Summary			
#	Smell Name	Found	Report
1	No link, Broken link	4%	<a href="#">Download CSV</a>
2	Flash Scrolling	22%	<a href="#">Download CSV</a>
3	Long Request	17%	<a href="#">Download CSV</a>
4	Undescriptive Element	6%	<a href="#">Download CSV</a>
5	Unformatted Input	10%	<a href="#">Download CSV</a>

Figure 24: Smell Summary View



Some sample CSV results are shown below. Figure 25 is showing the No link and broken link found in the website. It shows the page link, html line number of the anchor tag, smell status and refactoring suggestion also.

	A	B	C	D	E	F
1	Page	Start_Tag_Location	End_Tag_Location	Link	Status	Suggestion
2	http://data.gov.bd/	131	131	#main-content	Ok	None
3	http://data.gov.bd/	135	137	/	Ok	None
4	http://data.gov.bd/	140	140	/	Ok	None
5	http://data.gov.bd/	141	141		No link	Add href to this anchor tag
6	http://data.gov.bd/	142	142	/dataset	Ok	None
7	http://data.gov.bd/	143	143	/stories	Ok	None
8	http://data.gov.bd/	144	144	http://data.gov.bd/about-us	Ok	None
9	http://data.gov.bd/	151	151	/	Ok	None
10	http://data.gov.bd/	152	152	/bn	Ok	None
11	http://data.gov.bd/bn/about-us	196	196	/dataset?topics=society	Ok	None
12	http://data.gov.bd/bn/about-us	198	198	/dataset?topics=technology	Ok	None
13	http://data.gov.bd/bn/about-us	199	199	/dataset?topics=technology	Ok	None
14	http://data.gov.bd/bn/about-us	201	201	/dataset?topics=transport	Ok	None
15	http://data.gov.bd/bn/about-us	202	202	/dataset?topics=transport	Ok	None
16	http://data.gov.bd/bn/about-us	215	215		No link	Add href to this anchor tag
17	http://data.gov.bd/bn/about-us	219	221	/	Ok	None
18	http://data.gov.bd/bn/about-us	258	258	/group/agriculture	Ok	None

Figure 25: No link, Broken link report

Figure 26 shows the flash scrolling smell report of the website. This CSV will you provide you the page heights of the all the pages scanned of the website, flash scrolling status and refactoring suggestion.

	A	B	C	D
1	Page	Height(px)	Status	Suggestion
2	http://data.gov.bd/	2269	Flash Scrolling	Page height =< 1556 px
3	http://data.gov.bd//	2269	Flash Scrolling	Page height =< 1556 px
4	http://data.gov.bd//dataset	2362	Flash Scrolling	Page height =< 1556 px
5	http://data.gov.bd//stories	1346	Ok	None
6	http://data.gov.bd/about-us	1421	Ok	None
7	http://data.gov.bd//about-us	1421	Ok	None
8	http://data.gov.bd//bn/about-us	1421	Ok	None
9	http://data.gov.bd//bn	2266	Flash Scrolling	Page height =< 1556 px
10	http://data.gov.bd/bn	2266	Flash Scrolling	Page height =< 1556 px
11	http://data.gov.bd/bn/search/type/dataset	2362	Flash Scrolling	Page height =< 1556 px
12	http://data.gov.bd/bn/stories	1346	Ok	None
13	http://data.gov.bd/bn/about-us	1421	Ok	None
14	http://data.gov.bd//bn/dataset?topics=agriculture	2717	Flash Scrolling	Page height =< 1556 px
15	http://data.gov.bd//bn/dataset	2362	Flash Scrolling	Page height =< 1556 px
16	http://data.gov.bd//bn/dataset?topics=business	2562	Flash Scrolling	Page height =< 1556 px

Figure 26: Flash scrolling report

## UI Smell Meter View

The section (Figure 27) show the total elements, pages scanned and total number of smell found in a website. The percentages are showing the quality of the website.

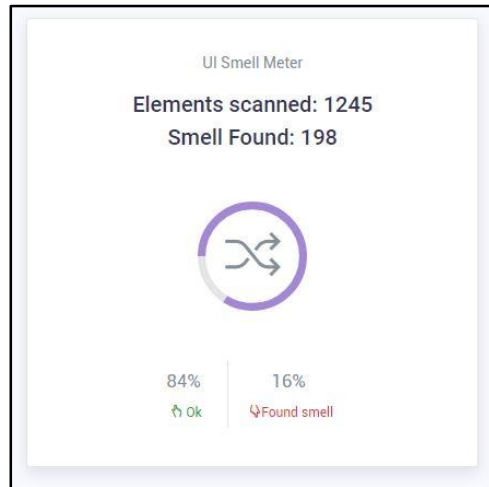


Figure 27: Smell meter view

## Homepage Screenshot

This section (Figure 28) shows the screenshot of the homepage of the website which has to be analyzed.

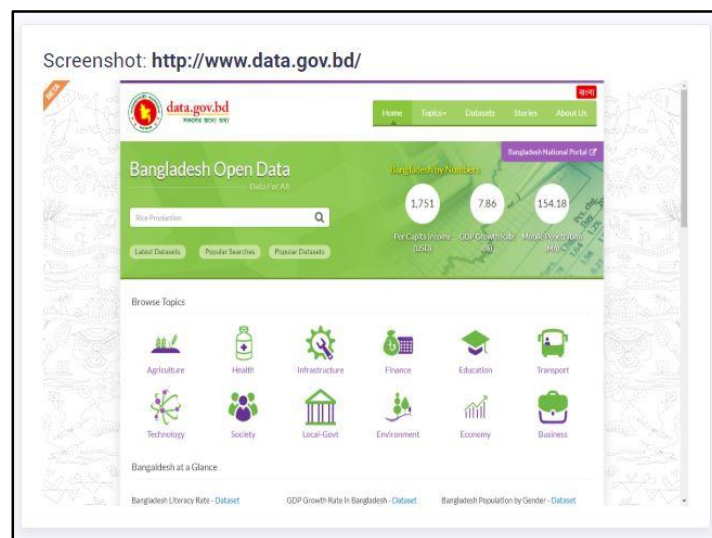
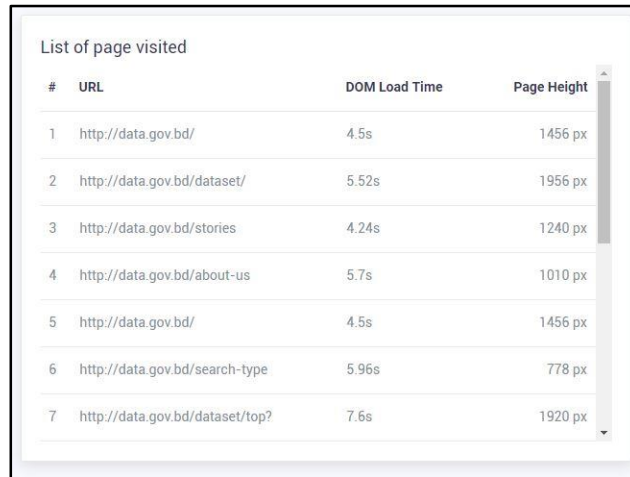


Figure 28: Screenshot view

## List of Pages View

This section (Figure 29) contains a table of all the pages scanned from the website along with their DOM load time and page height.



#	URL	DOM Load Time	Page Height
1	<a href="http://data.gov.bd/">http://data.gov.bd/</a>	4.5s	1456 px
2	<a href="http://data.gov.bd/dataset/">http://data.gov.bd/dataset/</a>	5.52s	1956 px
3	<a href="http://data.gov.bd/stories">http://data.gov.bd/stories</a>	4.24s	1240 px
4	<a href="http://data.gov.bd/about-us">http://data.gov.bd/about-us</a>	5.7s	1010 px
5	<a href="http://data.gov.bd/">http://data.gov.bd/</a>	4.5s	1456 px
6	<a href="http://data.gov.bd/search-type">http://data.gov.bd/search-type</a>	5.96s	778 px
7	<a href="http://data.gov.bd/dataset/top?">http://data.gov.bd/dataset/top?</a>	7.6s	1920 px

Figure 29: List of pages view

## 11 Conclusion

With all requirements satisfied I consider this project to be a success. UIAnalyzer is a web-based analytics tool that identifies and reports website UI smell. It tool can provide automatic advice about usability smells of user interaction for deployed web applications. The automated strategy to usability smell recognition is based on the analysis of user interaction (UI) events, linking specific UI events to usability smells, reporting usability smells which makes it possible to suggest solutions for them in terms of refactoring and generating CSV reports.

The work carried out on this project opens itself to a number of future uses. Of course, many of the classes and structures in the project were designed with code reuse in mind. The event parser and smell identifier are the examples of that. I myself fully intend to continue work in future on this topic about usability smell findings without human interaction.

## References

- [1] Julián Grigera, A. G. (January 2017). Automatic detection of usability smells in web applications. *International Journal of Human-Computer Studies*, Pages 129-148.
- [2] Arnaud Blouina, V. L. ( October 2018). User interface design smell: Automatic detection and refactoring of Blob listeners. *Information and Software Technology*, Pages 49-64.
- [3] Clara Sacramento, A. C. (2014). Web Application Model Generation through Reverse Engineering and UI Pattern Inferring. *9th International Conference on the Quality of Information and Communications Technology*. Guimarães, Portugal.
- [4] Faria, M. N. (2014). Inferring User Interface Patterns from Execution Traces of Web Applications. *International Conference on Computational Science and Its Applications*.
- [5] Patrick Harms, J. G. (29 Nov 2016). Usage-Based Automatic Detection of Usability Smells. *International Conference on Human-Centred Software Engineering*, 217-234.
- [6] Xiao Sophia Wang, A. B. (2013). Demystifying Page Load Performance with WProf. *10th USENIX Symposium on Networked Systems Design and Implementation* , 473-485.