Md. Aquib Azmain

Institute of Information Technology

UIAnalyzer

A web based UI smell finder tool for web application

A web based UI smell finder tool for web application

Technical Report of the Project

Supervised by

**Md. Saeed Siddik**

Lecturer

Institute of Information Technology

University of Dhaka

Submitted by **Md. Aquib Azmain (BSSE 0718)**

BSSE Session: 2014-2015



**Institute of Information Technology  
University of Dhaka**

29-09-2018

LETTER OF TRANSMITTAL

29th September 2018

The Coordinator

Software Project Lab 3

Institute of Information Technology

University of Dhaka

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

Dear Sir,

With due respect, I am pleased to submit the technical 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 technical report. 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 7th batch   
Institute of Information Technology   
University of Dhaka

ACKNOWLEDGMENT

At first, I would like to thank almighty for helping to prepare the technical report of this project.

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 start 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 wil 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. To help these cases, I have chosen to work with a project based on usability smells. This software will find the usibility smells from user interface of the application. This will be a web based tool that will work with web application. This tool will find also whether the web interface follows standard user interface pattern or not, based on usability. The automated strategy to usability smell identification is based on a process consisting of three steps: Events Logging, Usability Smells Detection, and Reporting.

Table of Contents

[1 Introduction 1](#_Toc525996122)

[2 Project Description 2](#_Toc525996123)

[2.1 Specific Requirements 2](#_Toc525996124)

[2.1.1 Functionality 2](#_Toc525996125)

[2.1.2 Usability 2](#_Toc525996126)

[2.1.3 Performance 2](#_Toc525996127)

[2.1.4 Web Based Requirements 2](#_Toc525996128)

[2.2 Usage scenario of UIAnalyzer 3](#_Toc525996129)

[2.2.1 Events logging 3](#_Toc525996130)

[2.2.2 Smell Detection 4](#_Toc525996131)

[2.2.3 Reporting 4](#_Toc525996132)

[3 Scenario-Based Modeling 5](#_Toc525996133)

[3.1 Use-case Diagram 5](#_Toc525996134)

[3.2 Activity Diagram 7](#_Toc525996135)

[4 Class-Based Modeling 8](#_Toc525996136)

[4.1 Final Classes 8](#_Toc525996137)

[4.2 Class Diagram 10](#_Toc525996138)

[5 Architectural Design 11](#_Toc525996139)

[5.1 Overview 11](#_Toc525996140)

[5.2 Instantiations of the System 12](#_Toc525996141)

[5.3 Elaborated Deployment 12](#_Toc525996142)

[6 User Interface Design 13](#_Toc525996143)

[7 Conclusion 17](#_Toc525996144)

# List of figures

[Figure 1: The three steps of the UIAnalyzer System 4](#_Toc525996145)

[Figure 2: Level-0 Usecase Diagram 5](#_Toc525996146)

[Figure 3: Level-1 Usecase Diagram 5](#_Toc525996147)

[Figure 4: Activity Diagram of Whole System 7](#_Toc525996148)

[Figure 5:Class Diagram of UIAnalyzer 10](#_Toc525996149)

[Figure 6: 2-tier architecture of UIAnalyzer 11](#_Toc525996150)

[Figure 7: Archetectural overview of UIAnalyzer 11](#_Toc525996151)

[Figure 8: Instantiations of UIAnalyzer 12](#_Toc525996152)

[Figure 9: Elaborated Deployment of UIAnalyzer 12](#_Toc525996153)

[Figure 10: Home page of UIAnalyzer 13](#_Toc525996154)

[Figure 11: Application Under Analysis View 14](#_Toc525996155)

[Figure 12: Smell Specific View 15](#_Toc525996156)

[Figure 13: Summary Report View 16](#_Toc525996157)

# List of tables

[Table 1: Usability Smells 3](#_Toc525996199)

[Table 2: Class card of WebExplorer Class 8](#_Toc525996200)

[Table 3: Class card of EventProcessor class 8](#_Toc525996201)

[Table 4: Class card of SmellIdentifier class 9](#_Toc525996202)

[Table 5: Class card of PatternCategorizer Class 9](#_Toc525996203)

[Table 6: Class card of ReportGenerator Class 9](#_Toc525996204)

[Table 7: Class card of PDFGenerator Class 9](#_Toc525996205)

# 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 will 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, defining new usability smells, and reporting usability smells which makes it possible to suggest concrete solutions for them in terms of refactoring.

# Project Description

## Specific Requirements

The specific requirements of this project are –

### 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, or combination of usability events that trigger a specific smell.
* The system will also provide suggestion to refactor the UI code the will solve the usability smell.
* By finding usability smell the system can also conclude whether a web application has followed standard user interface pattern or not.
* The system will provide a complete report of the findings with statistical visualization and can be downloaded as PDF.

### Usability

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

### 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.

### 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.

## 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.

### 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 a client-side script that intercepts selected low-level UI events. Whenever an end user loads a page, the script will start analyzing low-level events like single keystrokes or mouse moves. 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 even when they require information across different pages (or requests), such as the Navigation Path event, which keeps track of quick navigations sequences.

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

Table : Usability Smells [1]

|  |  |
| --- | --- |
| Event | Usability Smell |
| Click Attempt | Unresponsive Element |
| Flash Scrolling | Overlooked Contents |
| Flash Navigation | Misleading Link |
| Navigation Path | Distant Contents |
| Bulk Action | Unnecessary Bulk Action |
| Form Submission | No validation |
| Text Input | Unformatted Input, Short Input |
| Long Request | No Processing Page |

### 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.

### 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 PDF 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 [1].

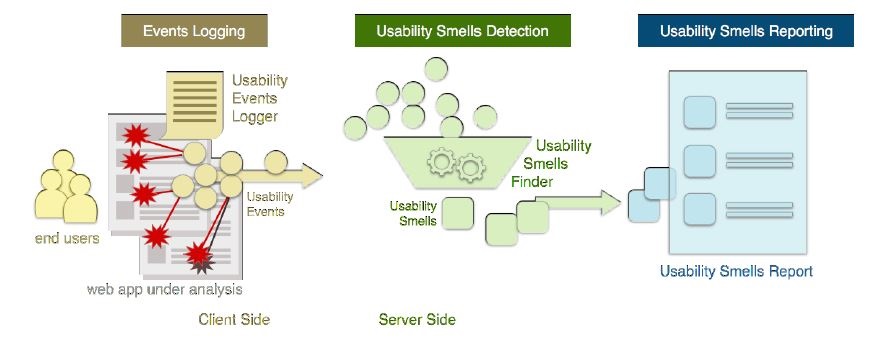


Figure : The three steps of the UIAnalyzer System

# Scenario-Based Modeling

## 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.

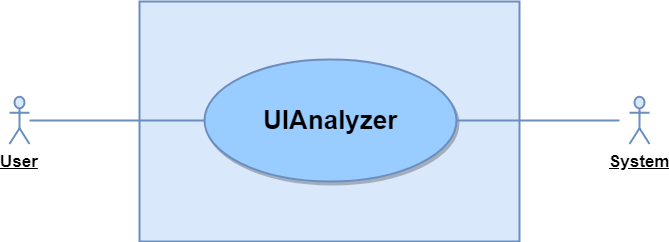


Figure : Level-0 Usecase Diagram

|  |  |
| --- | --- |
| Name  ID  Primary Actors | UIAnalyzer  UIANALYZER-L-0  User, System |

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

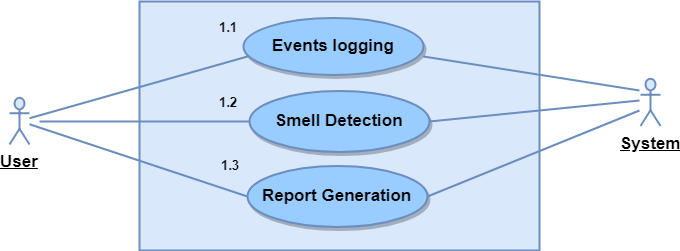


Figure : Level-1 Usecase Diagram

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

**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 PDF

Reply 4: The PDF will be generated and downloaded.

## 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.

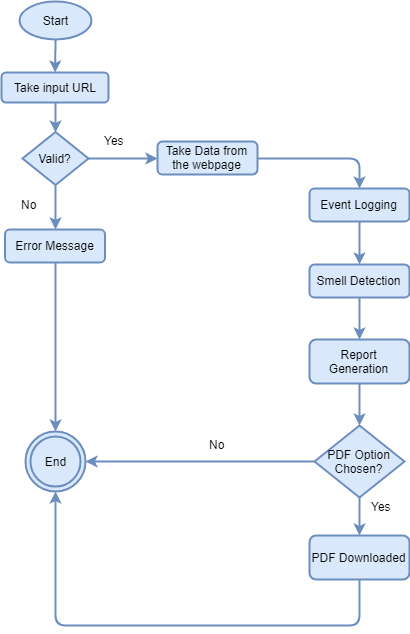


Figure : Activity Diagram of Whole System

# Class-Based Modeling

## Final Classes

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

1. WebExplorer
2. EventPreprocessor
3. SmellIdentifier
4. PatternCategorizer
5. ReportGenerator
6. PDFGenerator
7. 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 PatternCategorizer Class

|  |  |
| --- | --- |
| 4.PatternCategorizer | |
| Attributes | **Methods** |
| smellList | getIdentifiedSmell(),  InferUIPattern() |

Table 6: Class card of ReportGenerator Class

|  |  |
| --- | --- |
| 5.ReportGenerator | |
| Attributes | **Methods** |
| smellList,  urlList | readSmellList(),  generateRefactoringSuggestion(),  generateChart() |

Table 7: Class card of PDFGenerator Class

|  |  |
| --- | --- |
| 6.PDFGenerator | |
| Attributes | **Methods** |
| fileName  filePath | getSummaryReport(),  generatePDF() |

|  |  |
| --- | --- |
| 7.ToolManager | |
| Attributes | **Methods** |
| - | initiateWebExplorer(),  initiateSmellIdentifier(),  initiateReportGenerator() |

## Class Diagram

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

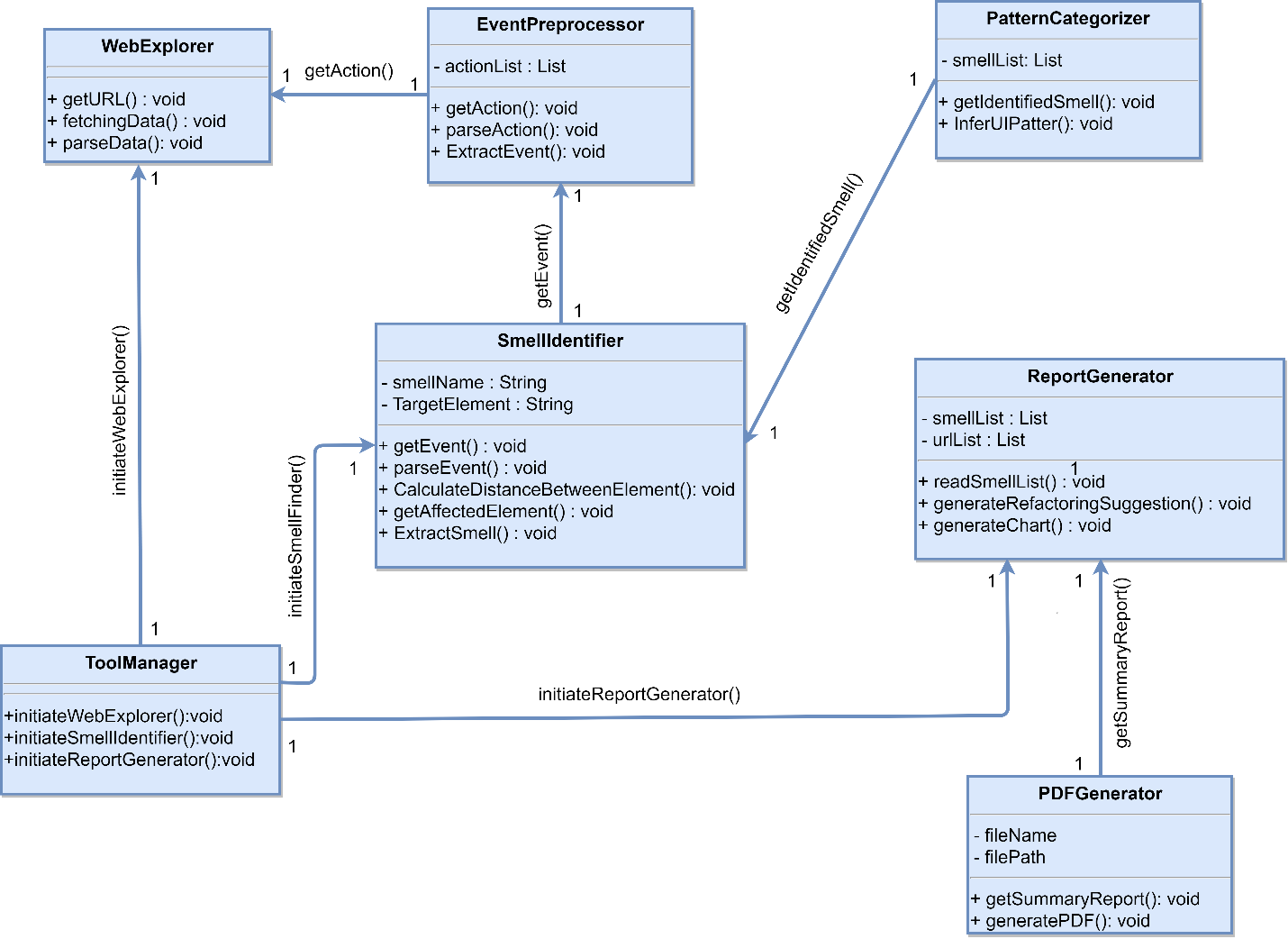


Figure :Class Diagram of UIAnalyzer

# Architectural Design

## 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.

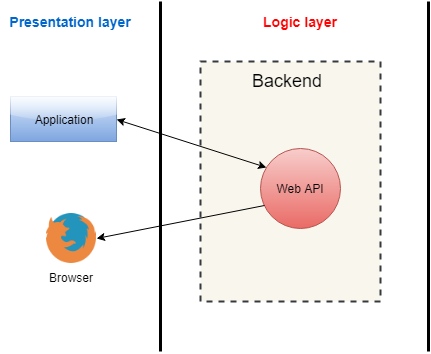


Figure 7: Archetectural overview of UIAnalyzer

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.

## Instantiations of the System

The instantiations of the whole system is given below:

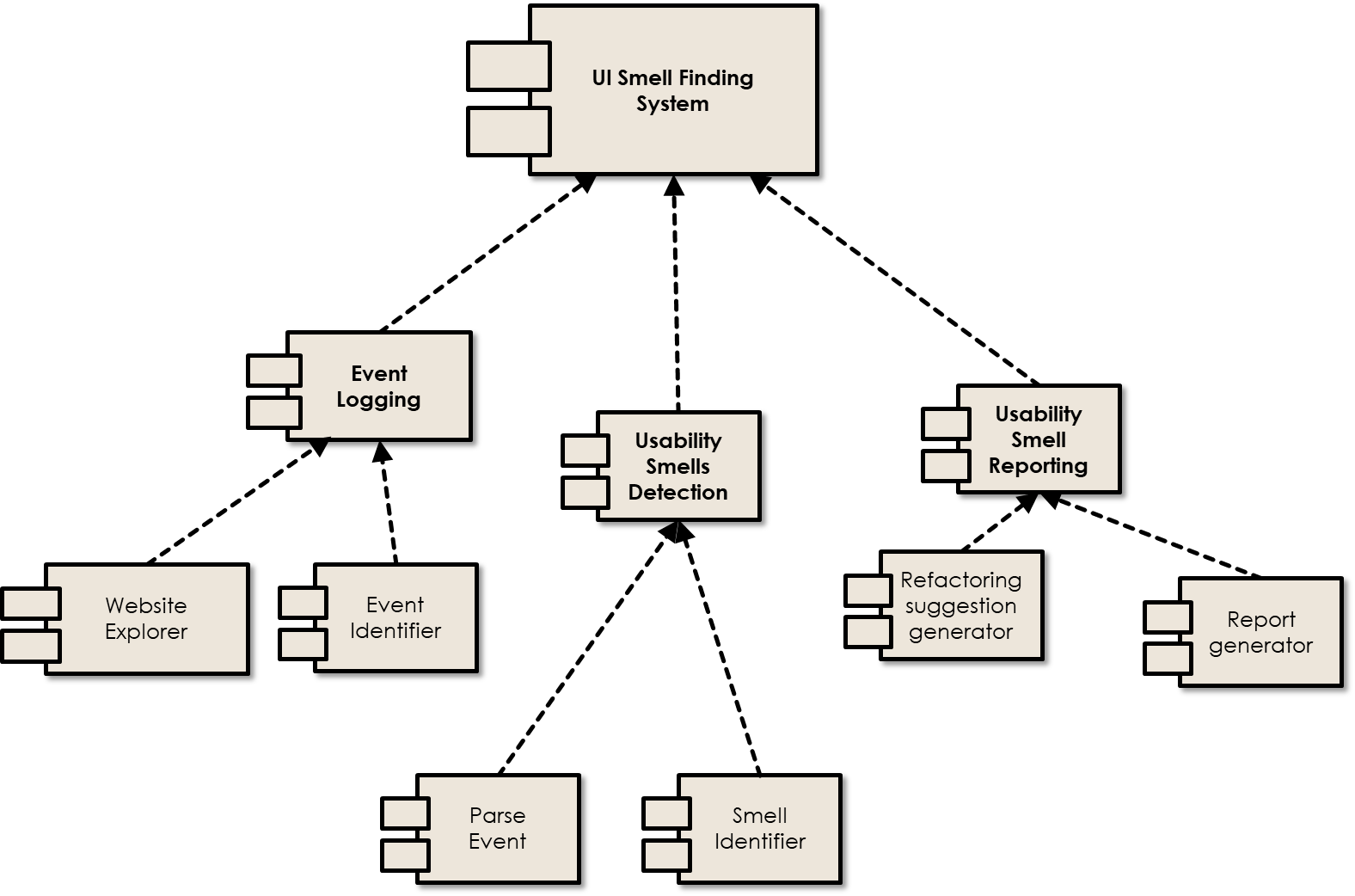


Figure : Instantiations of UIAnalyzer

## Elaborated Deployment

The elaborated deployment of the system is given below:

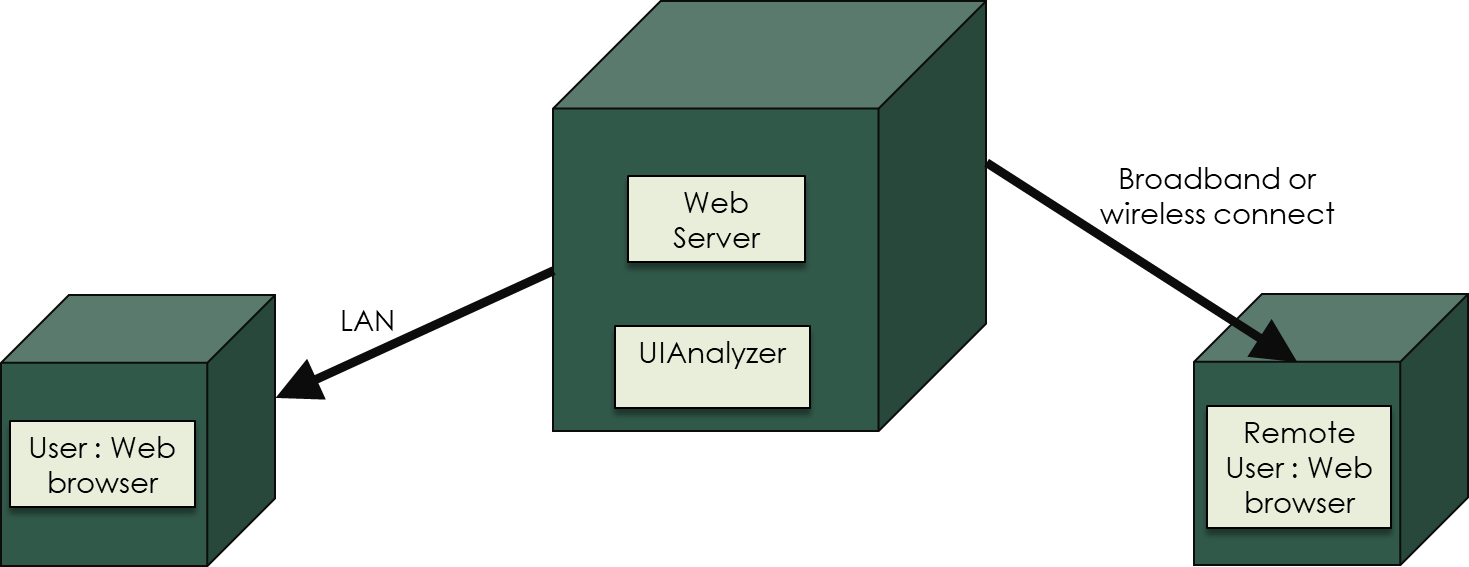


Figure : Elaborated Deployment of UIAnalyzer

# 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.

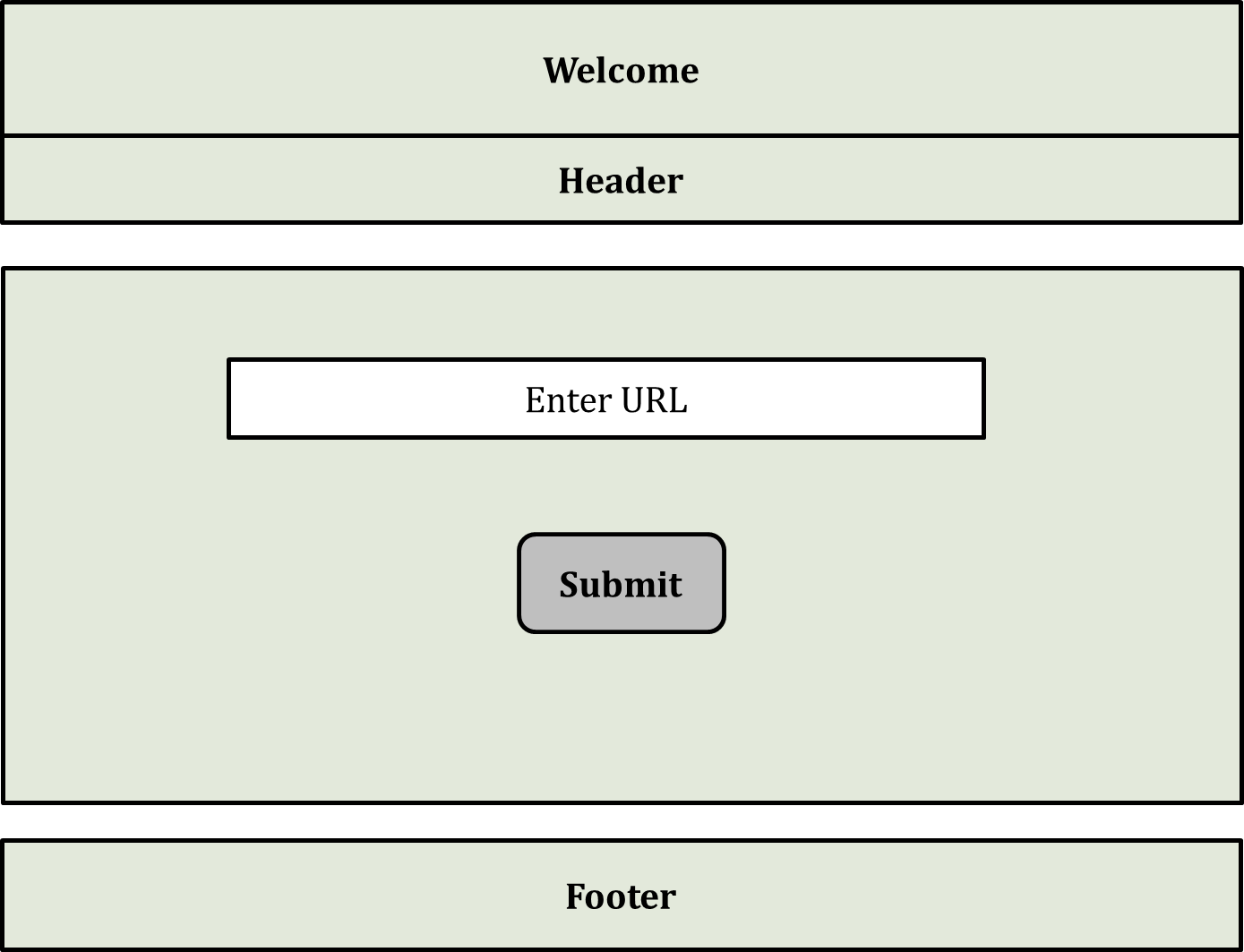


Figure : Home page of UIAnalyzer

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.

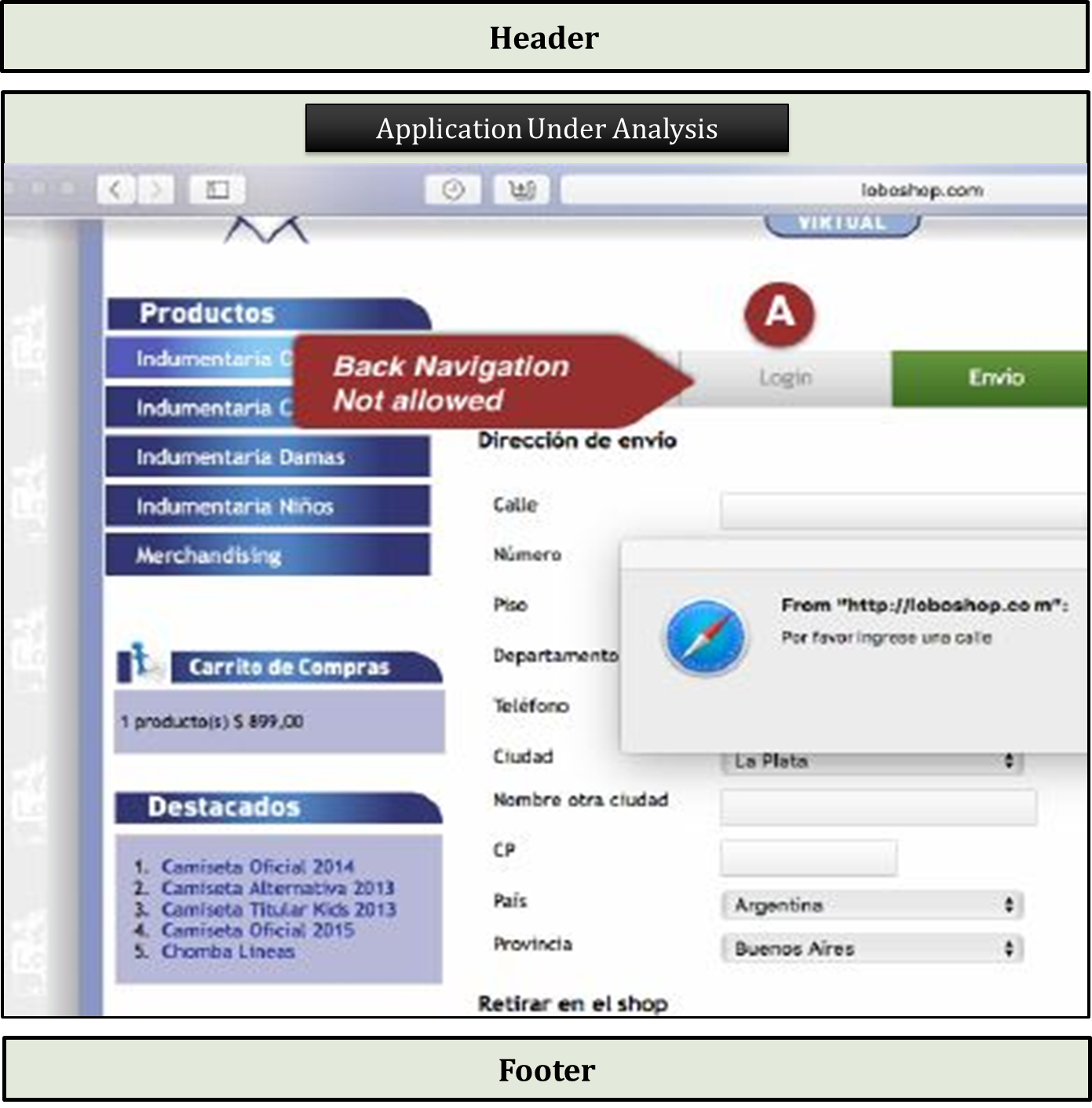


Figure : Application Under Analysis View

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

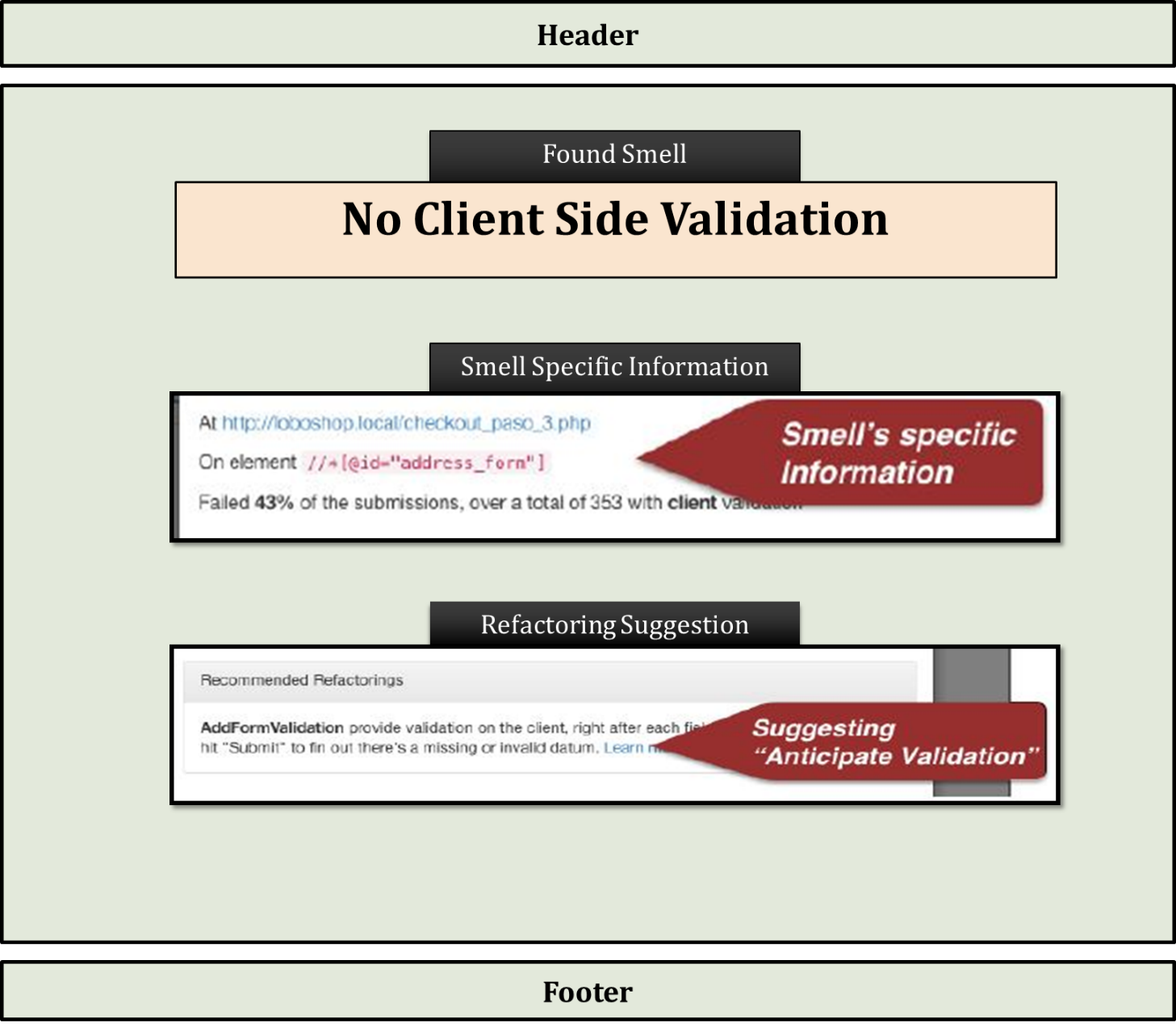


Figure : Smell Specific View

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 PDF format if he/she wants to.

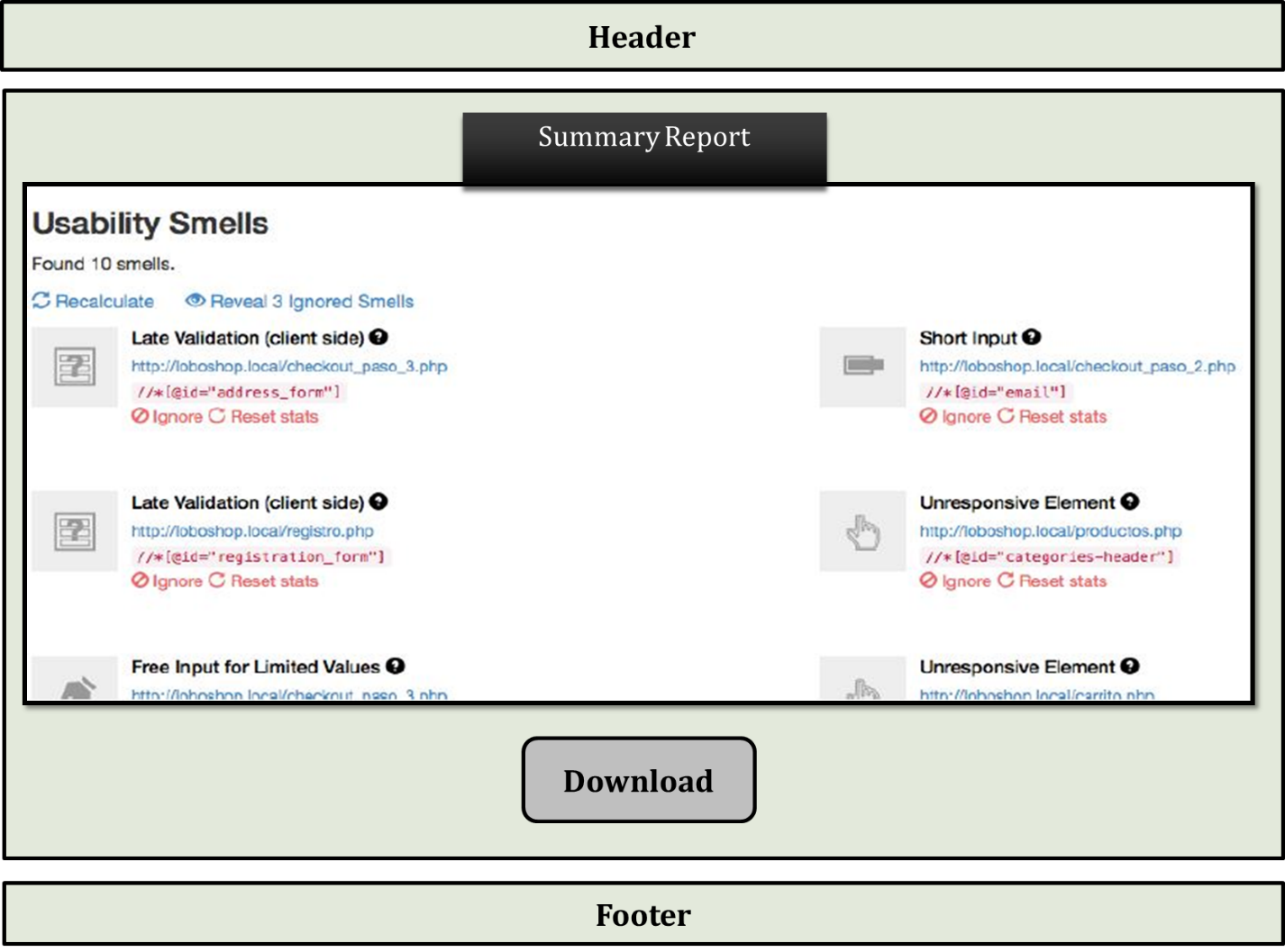


Figure : Summary Report View

# Conclusion

This document has pointed out every necessary point which is necessary for the next development phase. For better understanding, different figure and the table have shown in this document. All the steps for developing the software and both high-level and low-level categories have been shown in brief in the usage scenario point.

The mockup UI design for the tool has also been shown for the front-end development. Besides, this will help to understand the workflow of the total system.

Finally, with this document, I have tried to minimize all the ambiguity of the development. I hope this report can be used effectively to maintain the software development cycle. It will be very easy to conduct the whole project using this technical report.

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.