

**University Institute of Information Technology,**

**PMAS-Arid Agriculture University,**

**Rawalpindi Pakistan**

**Analysis of Cross Platform Binaries**

***By***

**Anas bin Anwar 18-Arid-3007**

**Muhammad Mohsin Ali Babar 18-Arid-3028**

***Supervisor*Mr. Zeeshan Javed**

***Bachelor of Science in Software Engineering (2018-2022)***

**The candidate confirms that the work submitted is their own and appropriate  
 credit has been given where reference has been made to the work of others**.

**DECLARATION**

We hereby declare that this software, neither whole nor as a part has been copied out from any source. It is further declared that we have developed this software documentation and accompanied report entirely on the basis of our personal efforts. If any part of this project is proved to be copied out from any source or found to be reproduction of some other. We will stand by the consequences. No Portion of the work presented has been submitted of any application for any other degree or qualification of this or any other university or institute of learning.

Anas bin Anwar Muhammad Mohsin Ali Babar

**CERTIFICATE OF APPROVAL**

It is to certify that the final year project of BS (SE) “Analysis of Cross Platform Binaries” was developed by **“Anas bin Anwar (18-Arid-3007)” and “Muhammad Mohsin Ali Babar (18-Arid-3028)”** under the supervision of “Mr. Zeeshan Javed” and that in their opinion; it is fully adequate, in scope and quality for the degree of Bachelors of Science in Software Engineering.

---------------------------------------

**Supervisor**

---------------------------------------

**(Mr. Zeeshan Javed)**

**External Examiner**

---------------------------------------

**Administrator UIIT**

**Executive Summary**

We will develop an advanced and efficient dedicated Analysis of cross platform binaries using latest technologies; these technologies will be a web-based. This system will be introduced as a solution of these issues as well as a Some binaries use malware which is used by Attackers to perform malicious actions and gain control over their targets or spying on them. The users in this project will be analysis the malware and detect it. Cross platform analysis is really important because by analyzing how the malware functions, how it spreads, and the type of malware used. It helps us defend against future attacks. Remember that security of user or privacy of user is main important for website.

**Acknowledgement**

All praise is to Almighty Allah who bestowed upon us a minute portion of His boundless knowledge by virtue of which we were able to accomplish this challenging task.

We are greatly indebted to our project supervisor “Mr. Zeeshan Javed” for personal supervision, advice, valuable guidance and completion of this project. We are deeply indebted to him for encouragement and continual help during this work.

And we are also thankful to our parents and family who have been a constant source of encouragement for us and brought us the values of honesty & hard work.

Anas bin Anwar Muhammad Mohsin Ali Babar

**Table of Contents**

**Introduction 1**

[1.1 Brief](#_Toc268523777) 1

[1.2 Relevance to Course Modules](#_Toc268523779) 2

[1.3 Project Background](#_Toc268523780) 2

[1.4 Literature Review](#_Toc268523780) 3

[1.5 Methodology and Software Life Cycle](#_Toc268523782) 4

**Problem Definition** 6

[2.1 Purpose](#_Toc268523795) 6

[2.2 Product Functions](#_Toc268523796) 6

[2.3 Proposed Architecture](#_Toc268523804) 7

[2.4 Project Deliverables](#_Toc268523806) 8

[2.5 Operating Environment](#_Toc268523807) 8

[2.6 Assumptions and Dependencies](#_Toc268523808) 9

**Requirement Analysis** 10

[3.1 Functional Requirments](#_Toc268523823) 10

[3.2 Non – Functional Requirments](#_Toc268523825) 10

[3.3 Use case Model](#_Toc268523823) 11

[3.3.1 Use Case Diagarm](#_Toc268523787) 11

[3.3.2 Actors Discription](#_Toc268523787) 12

[3.3.3 Use Case Discription](#_Toc268523787) 12

**The Design** 15

[4.1 System Architecture](#_Toc268523830) 15

[4.2 System Design](#_Toc268523830) 15

[4.2.1 UML Structural Diagrams](#_Toc268523787) 16

[4.2.1.1 Component Diagram](#_Toc268523787) 16

[4.2.1.2 Package Diagram](#_Toc268523830) 17

[4.2.1.3 Deployment Diagram](#_Toc268523830)

[4.2.2.2 State Machine Diagram](#_Toc268523830) 20

[4.2.3 UML Interaction Diagrams](#_Toc268523787) 21

[4.2.3.1 Sequence Diagrams](#_Toc268523830) 21

[4.3 Node Structure](#_Toc268523830) 24

[4.4 Class Diagram](#_Toc268523830) 25

[4.5 Communication Design Protocol](#_Toc268523830) 26

**List of Figures**

Fig 3.1 Use Case Diagram 11

Fig 4.1 System Architecture 15

Fig 4.2 Component Diagram 16

Fig 4.3 Package Diagram 17

Fig 4.4 Deployment Diagram 18

Fig 4.5 Activity Diagram 19

Fig 4.6 State Machine Diagram 20

Fig 4.9 Sequence Diagram 21

Fig 4.14 Node Structure 24

Fig 4.15 Class Diagram 25

Fig 4.16 Communication Diagram 26

# Chapter 1:

In this chapter we will discuss over view of our whole project, its brief introduction, and how it is relevant to the courses which we have studied during our degree. We will also discuss project literature review and its analysis and methodology that we will use in project.

**Introduction**

Binaries is a term used to refer to a particular set of executables or compiled computer programs files that will perform particular functions. Some binaries use malware which is used by Attackers to perform malicious actions and gain control over their targets or spying on them e.g., Virus, Trojan, spyware. These are all examples of malware that are widely common and used.

Cross platform analysis is the process of analyzing a malware sample and be able to infiltrate as much as information as possible, Cross platform analysis is really important because by analyzing how the malware functions, how it spreads, and the type of malware used. It helps us defend against future attacks.

# Brief

Binaries is a term used to refer to a particular set of executables or compiled computer programs files that will perform particular functions. Some binaries are malware which is used by attackers to perform malicious actions. Cross platform analysis is the process of analyzing a malware sample and be able to infiltrate as much information as possible. Cross platform analysis is really important because by analyzing how the malware functions, how it spreads and the type of malware used. It helps us defend against future attacks. Platform in which malware design to execute such as Windows, Mac OS and Android etc. The expansion of malware is not only towards Windows OS but also towards other platforms such as Android, Linux and OS X

Most of the scanning methods used specific signatures to detect malware. Methods are based on various techniques such as static and dynamic analysis.

**Static Analysis**

Static analysis works by disassembling and executing the code without a virtual environment. Static analysis techniques used signatures for malware identification.

**Dynamic Analysis**

Dynamic analysis works by running the code in a virtual environment. In dynamic analysis malware must be executed for a specific time duration.

**Relevance to Course Modules**

Almost everything which we are using in our project “Analysis of Cross Platform Binaries” is totally relevant to our course materials.

* Programming fundamental
* Object Oriented Programming
* Data structure and analysis
* Database
* The malware which we are developing in our project is relevant to the subject “Python Programming Language" and “Flask”.
* The data of the users which will be stored in cryptography and status will be shown on screen, is relevant to the subject Beautiful soup which is library file of python.

# Project Background

Binary code analysis is a key requirement for many software engineering tasks that include:

Especially with the rapid growth of IT devices and the complexity of software applications, Program binaries are often shared among multiple platforms. Imagine a single bug is injected at source code level, it may spread across thousands or more devices that have diverse hardware architectures and software platforms. Thus, binary analysis is particularly critical and more crucial than ever. Binary Code analysis is used to provide information about a program’s content (instructions, basic blocks, functions, and modules), structure (control and data flow), and data structures (global and stack variables), and is, therefore, a foundation of many security applications. Code Clone Detection in binaries seeks to find code sequence used more than once, or same source code but compiled under different Instruction Set Architectures. Malware detection detects malicious programs which have vulnerabilities inside and will cause damage to systems or programs crashes. Code obfuscation translates the original program into another one preserving its function but making it hard for analysis. Binary reverse engineering translates the binary program into high level readable language, such as converting binaries back to source code. Binary customization directly changing the binary program through binary rewriting, removing unused or vulnerable codes/functions in program binaries. Software can manipulate hardware features as well to do malicious activities. Such studies call for a more thorough understanding of software binaries and assessing their potential to be exploited in security attacks

# Literature Review

Data security is challenge all over the world nowadays among the Internet use. The primary function of Cross platform analysis system is to analyze, reverse engineering and dissect malicious code as well as potentially unwanted software. Hopefully by using this system, users can avoid the loss of data. It is difficult for user to understand the complex tool, so we decided to make a system that user can use easily and understand the functionalities. Virtual environment is provided for execution where possible.

Here is brief review of some existing systems which are related to our project.

**PeStudio**

**Features**

The PeStudio software is safe to use.

The support of PeStudio is good.

It retrieves maximum detail from any executable file.

It can retrieve the referenced libraries and functions.

It can inspect malicious executable file, Trojan, ransomware, and APT samples.

**Limitations**

Not easy to use.

**Process Hacker**

**Features**

It allows a malware analyst to see what processes are running on device.

It useful when to detect new process detonated by malware.

It makes easy to identify how the malware is attempting to hide.

This tool is also useful for pulling information from the memory of a process.

**Limitations**

Not easy to use.

Performance is not good.

Detects limited malware and malicious files.

**Process monitor**

**Features**

It record live file system activity system such as process creations and registry changes.

It detects malware when the process start.

It is also useful for analyzing malicious documents.

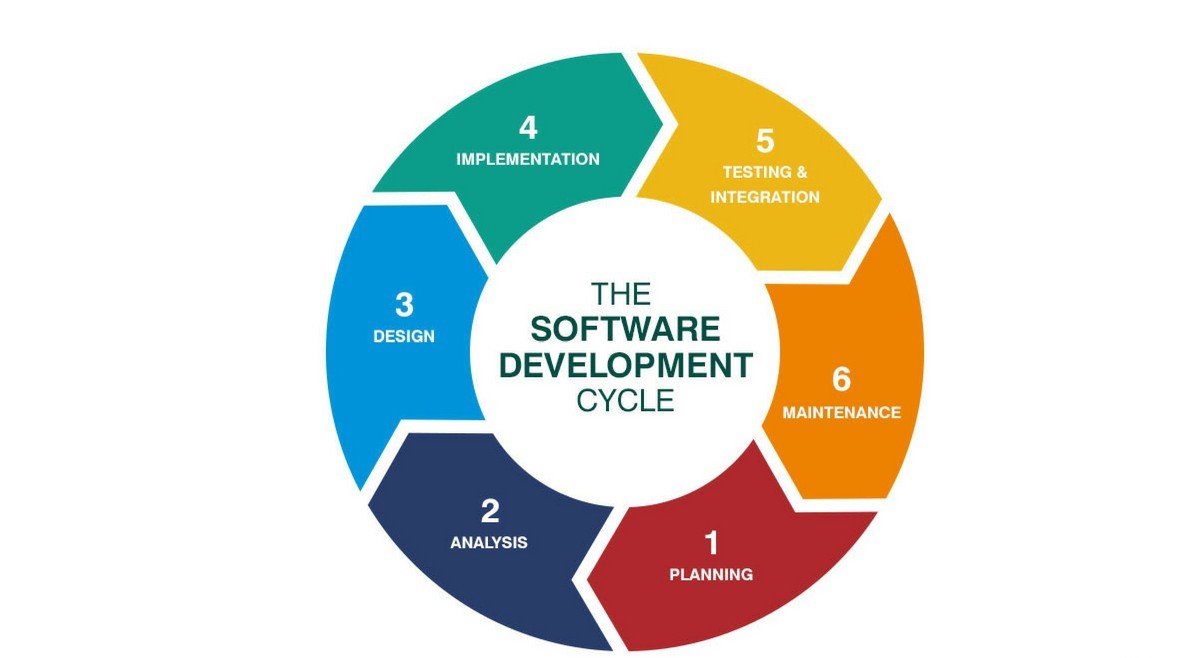
**Limitations**

Slow in analyzing malware.

# Methodology and Software Lifecycle for This Project

The software methodology used in our Project is agile method. We are using this technique because Agile methods can help us to manage work moreefficientlyand do the work more effectively and it will help us to delivering highest quality product within the constraint of time limit.

* + 1. Rationale behind Selected Methodology
* Easy to use
* User friendly
* Take action in time
* Well managed Problems and their solution
* Give tips and tricks against attacker

****

**Chapter 2: Problem Definition**

# Purpose:

The purpose of this analysis is to better understand the capabilities and behaviors of cross platform binaries, also known as cross platform executables. We plan to examine the reverse engineering techniques that were used to generate the cross platform binaries, and will explore the different ways they can be leveraged and the risks that may be associated with their use. The analysis will also provide background information on cross platform binaries and will discuss the technical challenges that are associated with the analysis of these types of binaries. In the past, malware was primarily delivered through email or other web-based methods. The majority of malware today, however, is delivered through cross-platform binaries such as exploit kits and is therefore harder to detect. The purpose of this analysis is to help detect this type of malware. This analysis can be used to proactively protect users from zero-day threats, or to aid security analysts during incident response or triage. Malware analysis is the practice of analyzing malicious software for the purposes of understanding, detecting, and proactively protecting against threats such as spyware and ransomware. One use case for malware analysis is the detection of cross platform binaries, which are executables that are able to run on several operating systems such as Windows, Linux, and Mac OS X. This is especially useful for defenders who want to be able to detect malicious activity regardless of the platform it is executed on. Another use case is threat hunting, which is the process of searching for threats in order to proactively prevent them from causing harm. One way to hunt for cross platform binaries is to run them through a sandbox, which is a controlled environment that allows for the observation of a program's behavior

Malware often takes the form of cross platform binaries, which are programs that can be executed on more than one operating system. The ability to execute a single malicious program on multiple operating systems is a source of convenience for users, but it also enables malware developers to evade security mechanisms that are platform-specific. Cross platform binaries are often used to evade detection by traditional anti-malware mechanisms, which are often designed to detect binaries that are only designed to run on a single operating system. This ability to evade detection makes the analysis of cross platform binaries a critical component in modern security defenses.

# Product Functions:

The analysis of cross platform binaries is critical, as attackers often use them to distribute malware. Malware has become one of the most destructive, costly and complex threats facing today’s society. The increasing number of malware strains, coupled with a strong desire on the part of hackers, criminals, and nation-states to acquire and sell them has created a lucrative and growing business.

The product function of Analysis of cross platform binaries is to provide developers and analysts with a set of ready to use, comprehensive and extensible analysis tools for binary and source code inspection, bug detection and data exploration. The product is composed of a set of extensible and customizable analysis engines, all of which are written in Python and can be extended via Python modules. The product is primarily focused on the analysis of binaries and source code, though there are plans to expand into other areas such as network traffic.

* 1. **Proposed Architecture:**

Architecture has been used in our project.

The tools will provide the following functionalities:

* monitoring communications on C&C channel to detect harmful activities,
* monitoring traffic for detecting and confining DDoS attacks,
* detecting zombie computers on the network,
* filtering botnet-related traffic out of networks,
* confining infected parts of the network and limiting the propagation of the malware,
* finding methods for disabling botnets.

**2.3. Project Deliverables:**

Following are the deliverables and development requirements:

**2.3.1 Projects Deliverables:**

In this project the deliverables will consist of the input as equipment (i.e., software components) then there will be a process (i.e., development phases in which development will be completed) applied on the input and then as a result there will be an output (Project being completed “Rise Real ”). In this case product deliverables are the completed parts or modules of the project. Input will be hardware and software-based components. The project is divided into in different modules and each module is major milestone in the project.

**2.3.2 Development Requirements:**

Development requirements are the requirements needed for the development purposes without which the development is not possible. It can be hardware, software or any kind of requirements. These include the software and hardware equipment’s, time and date constraints, budget, planning, following the SDLC etc. development requirements are met accordingly to make sure that the end result does not differ from what is expected and that it can perform its functionality accurately and perfectly without any glitches.

**2.3.3 Operating Environment:**

Operating environment for the Analysis of cross platform binaries is as listed below:

**Operating System**: Android and Ios Operating system

**Database**: Firebase

**Platform:** Visual Studio code in which web app will be developed using python. We will use python interpreter.

**2.4. Assumptions and Dependencies:**

**Assumptions:**

The website from this technique is more efficient than others. This website gives prefect result in to detect the malware from the analyze the cross binaries. The customer will find every kind of properties there.

**Dependencies:**

This software is dependent on internet connectivity and running device.

# Chapter 3: Requirement Analysis

In this chapter we will define all the requirements of proposed system that include functional and non-functional requirements. We will also discuss about use cases of the system and see how our system will respond to various use cases.

# Functional Requirements

A number of functional requirements are necessary to make our system work and facilitate the user's experience. Since all of these functional specifications are interface-less, they can also be assumed to be interface requirements.

* System provides a link for the Users/Client Registration.
* Administrator and Client can log in by entering user name and password and control the functionalities of system.
* System analysis the type of malware and its functionalities.
* System determines how the malware can affect system and which type of malware it is.
* System tells user that how to protect from malware attacks by giving tips to user.
* Malware analysis can expose behavior and artifacts that hunters can use to find activity.
* Malware analysis system can find how malware is communicating with attackers.
* Clients can provide feedback about services through feedback option.

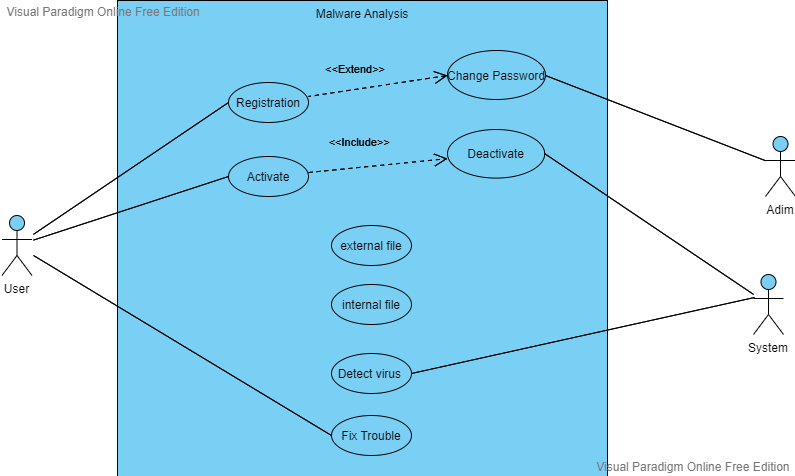
# Non-Functional Requirements

* The usability of system should be excellent. The system should be easy to use and should provide guide to user.
* System should provide all functionalities without failure.
* The performance of system be fast.

# Use Case Model

In the Unified Modeling Language (UML), a use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system. Following are the use cases of the Cross Platform malware analysis system.

**3.3.1 Use Case Diagram:**



* + 1. **Actors Description:**

We have two types of actors. One is Primary Actors and another one will act as a secondary actors.

* **User:**

User is a primary actor of our system which will directly interacts with the system. User will register himself, can change his/her password, User have two options he can check his system or any outsource file e.g.; for virus.

* **System:**

System is primary actor of our system. System detects the virus and provides the facility to fix the trouble.

* **Admin:**

Admin is an secondary actor of our system. Admin will maintains the authorization records of users and maintain other check and balance of system working.

* + 1. **Use Case Description:**

|  |  |
| --- | --- |
| **Use Case ID:** | ID-01 |
| **Use Case Name:** | Register |
| **Actors:** | User, Admin |
| **Description:** | User will register and authenticated by the system. |
| **Trigger:** | When user click on register system will provide him registration process. |
| **Pre-conditions:** | Username and password must be provided by the users. |
| **Post-conditions:** | User will login successfully. |
| **Normal Flow:** | * Credentials will be entered by users * Credentials will be authenticated from system * User will be successfully login |
| **Alternative Flows:** | * If user is not already registered, then user will first sign up * User will then provide credentials |
| **Exceptions:** | None. |
| **Special Requirements:** | None |
| **Assumptions:** | User know about registration information. i.e.; name etc |
| **Notes and Issues:** | User will have only 3 attempts for login. After 3 attempts user will have to wait for 30 seconds to retry. |

**Table 3.1: Register**

|  |  |
| --- | --- |
| **Use Case ID:** | ID-02 |
| **Use Case Name:** | Change password |
| **Actors:** | Primary Actor: user, admin |
| **Description:** | User click the change password button. |
| **Trigger:** | User is intended to change password |
| **Preconditions:** | User must change the password |
| **Postconditions:** | The user password is change successfully |
| **Normal Flow:** | **User Actions:**   1. User clicks the change password button 2. User change the password.   **System Responses:**   1. System stores the new password 2. Message is shown, “Success” or “Failure” |
|  | **3a.** In step 3 of the normal flow, if the System stores new password  System verifies the query and entities in the  Use Case resumes on step 4 of the normal flow  **3b.** In step 3 of the normal flow, if the System stores new password  into the database  System finds ambiguity and is not able to store new password  Use Case stays on step 3 of normal flow |
|  | **4a.** In step 4 of the normal flow, if the System shows Success  Message  New password is updated.  And stays onto the same page  **4b.** In step 4 of the normal flow, System shows failure Message,  System prompts the User to again change the password it will  stay on the same page again. |
| **Exceptions:** | **2a.** In step 2 of the normal flow, if a user uploads an invalid format  Changing of new password is disapproved  Message to user to re-enter new password  User again put new password with correct format  Use Case resumes on step 4 of normal flow |
| **Includes:** | None |
| **Special Requirements:** | There are no special requirements associated with this use case. |
| **Assumptions:** | User is aware of how to change the password and select the new one. |
| **Notes and Issues:** | User must have to enter the new password in appropriate format |

**Table 3.2: Change Password**

|  |  |
| --- | --- |
| **Use Case ID:** | ID-03 |
| **Use Case Name:** | Activate |
| **Actors:** | User, System |
| **Description:** | Registered user will activate the system to detect the virus. |
| **Trigger:** | User have to detect the virus. |
| **Preconditions:** | System is functional |
| **Post-conditions:** | System activated. |
| **Normal Flow:** | User activate the system.  System activated |
| **Special Requirements:** | None |
| **Assumptions:** | None |

**Table 3.3: Activate**

|  |  |
| --- | --- |
| **Use Case ID:** | ID-04 |
| **Use Case Name:** | deactivate |
| **Actors:** | User, System |
| **Description:** | Registered user will deactivate the system if he had completed his process. |
| **Trigger:** | User have to exit. |
| **Preconditions:** | System is functional |
| **Post-conditions:** | System deactivated. |
| **Normal Flow:** | User deactivate the system,  System deactivated. |
| **Special Requirements:** | None |
| **Assumptions:** | None |

**Table 3.4: deactivate**

|  |  |
| --- | --- |
| **Use Case ID:** | ID-04 |
| **Use Case Name:** | Detect virus |
| **Actors:** | system |
| **Description:** | System detects the virus as per user command. |
| **Trigger:** | User want to detect the virus. |
| **Preconditions:** | System functional. |
| **Post-conditions:** | System detects the virus. |
| **Normal Flow:** | System detects the virus |
| **Special Requirements:** | None |
| **Assumptions:** | None |

**Table 3.5: Create New Project**

|  |  |
| --- | --- |
| **Use Case ID:** | ID-05 |
| **Use Case Name:** | Fix trouble |
| **Actors:** | user |
| **Description:** | User want to fix the errors detected by the system. |
| **Trigger:** | User want resolve the troubleshoots. |
| **Preconditions:** | Errors detected. |
| **Post-conditions:** | Errors resolved. |
| **Normal Flow:** | User select the fix the error  Error fixed. |
| **Special Requirements:** | None |
| **Assumptions:** | None |

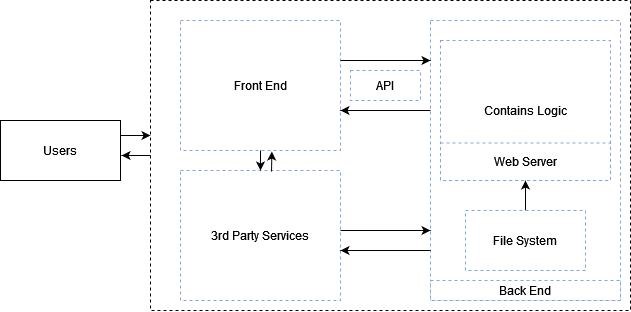
**Table 3.5: Create New Project**

# Chapter 4: Design and Architecture

In this chapter we will discuss the design and architecture of our system.

**4.1. System Architecture**

As system design varies from system to system, therefore user need to have the architecture view of the whole system.



**4.2. System Design**

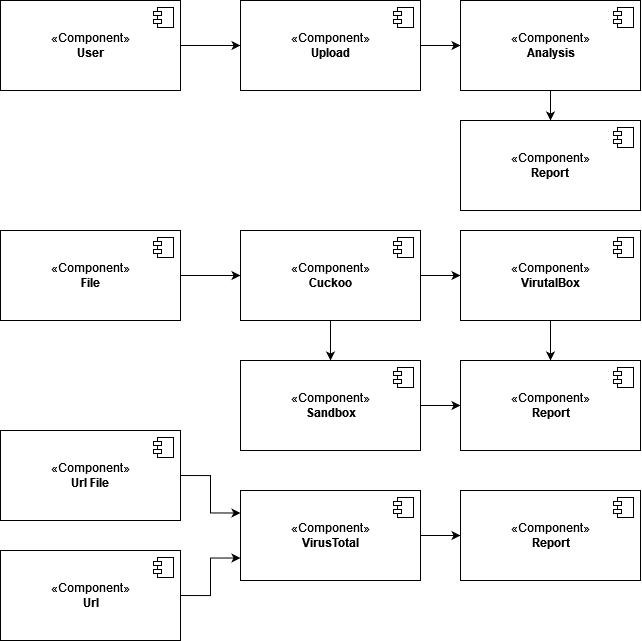
Systems design is the process of defining elements of a system like components, modules, architecture and their interfaces and data for a system based on the specified requirements. The purpose of the System Design process is to provide sufficient detailed data and information about the system. Following is the system design of the EzParking.

**System Design**

**UML Structural Diagrams**

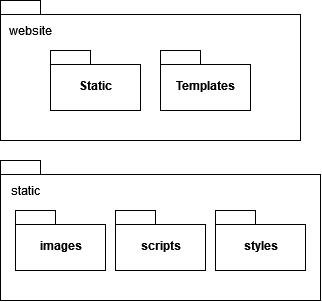
Following are the UML structural diagrams of our system:

**Component Diagram**

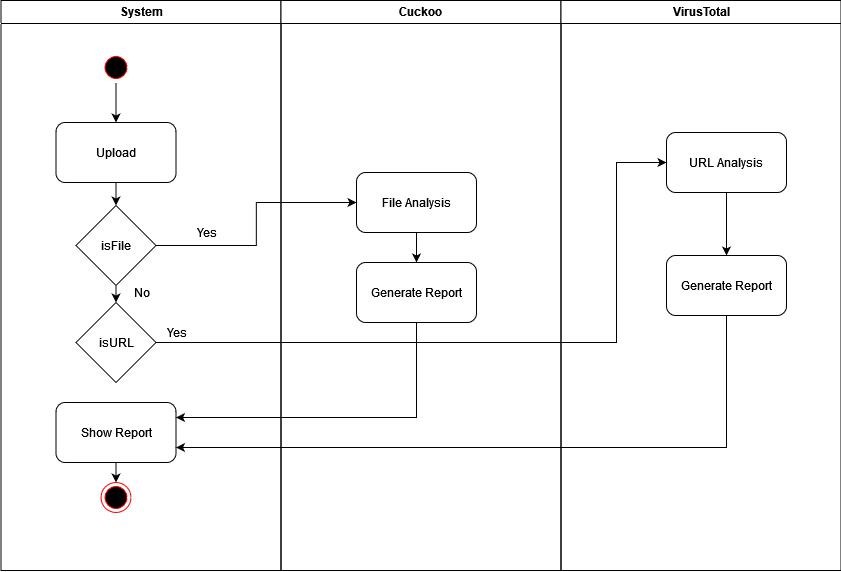


**Fig 4.2: Component Diagram**

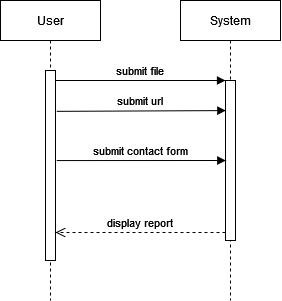
**4.2.1.2. Package Diagram**

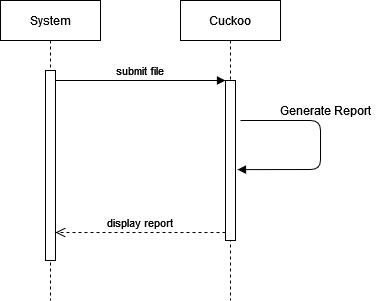


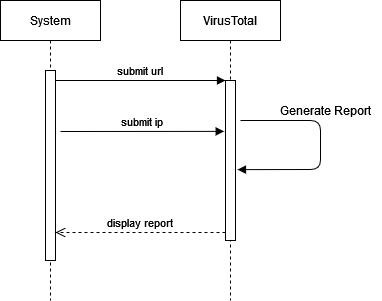
**Fig 4.3: Package Diagram**

**UML Behavioral Diagrams Activity Diagra****m**

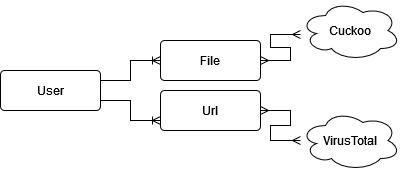
# UML Interaction Diagrams Sequence Diagrams







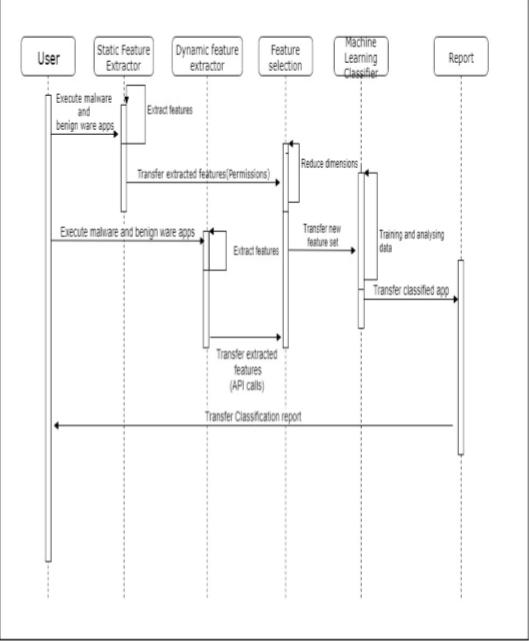
# Entity Relationship Diagram





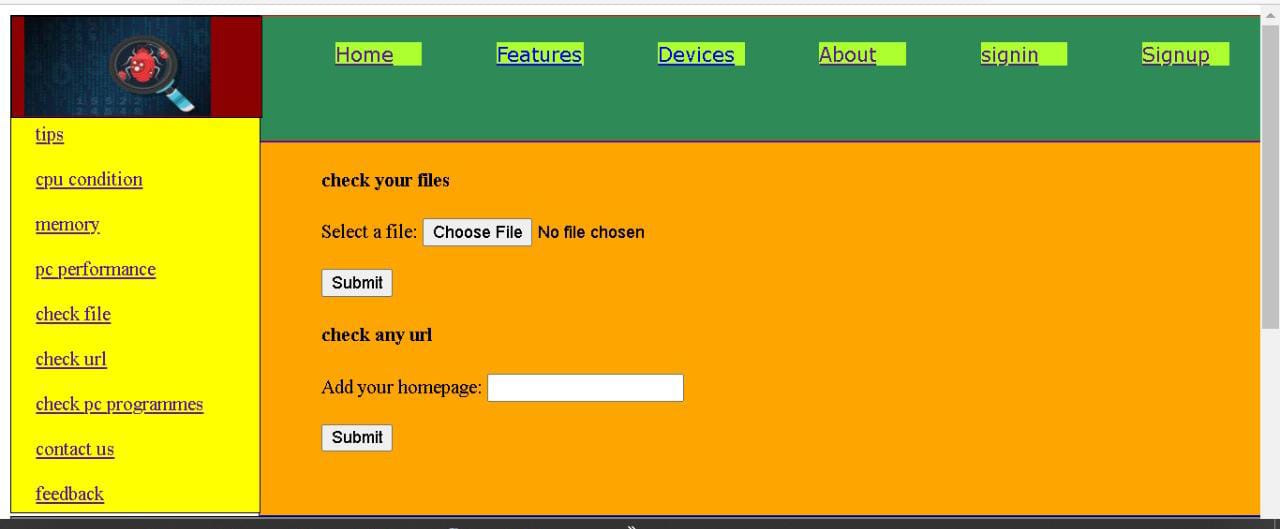
**4.2.3.1 Sequence Diagrams**

A sequence diagram is a type of interaction diagram because it describes how and in what order a group of objects works together

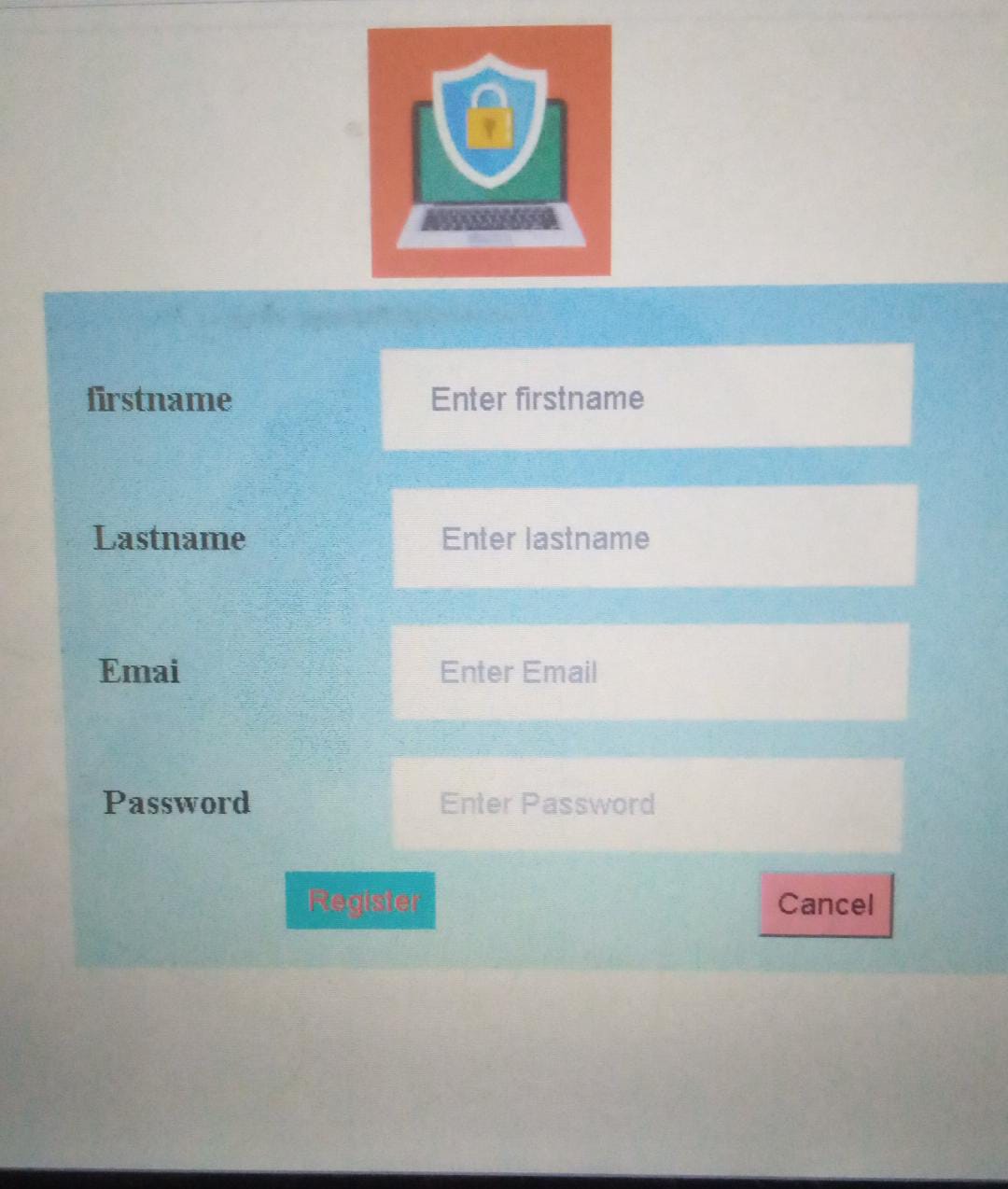


**4.6 Interface of Analysis of cross platform binaries:**

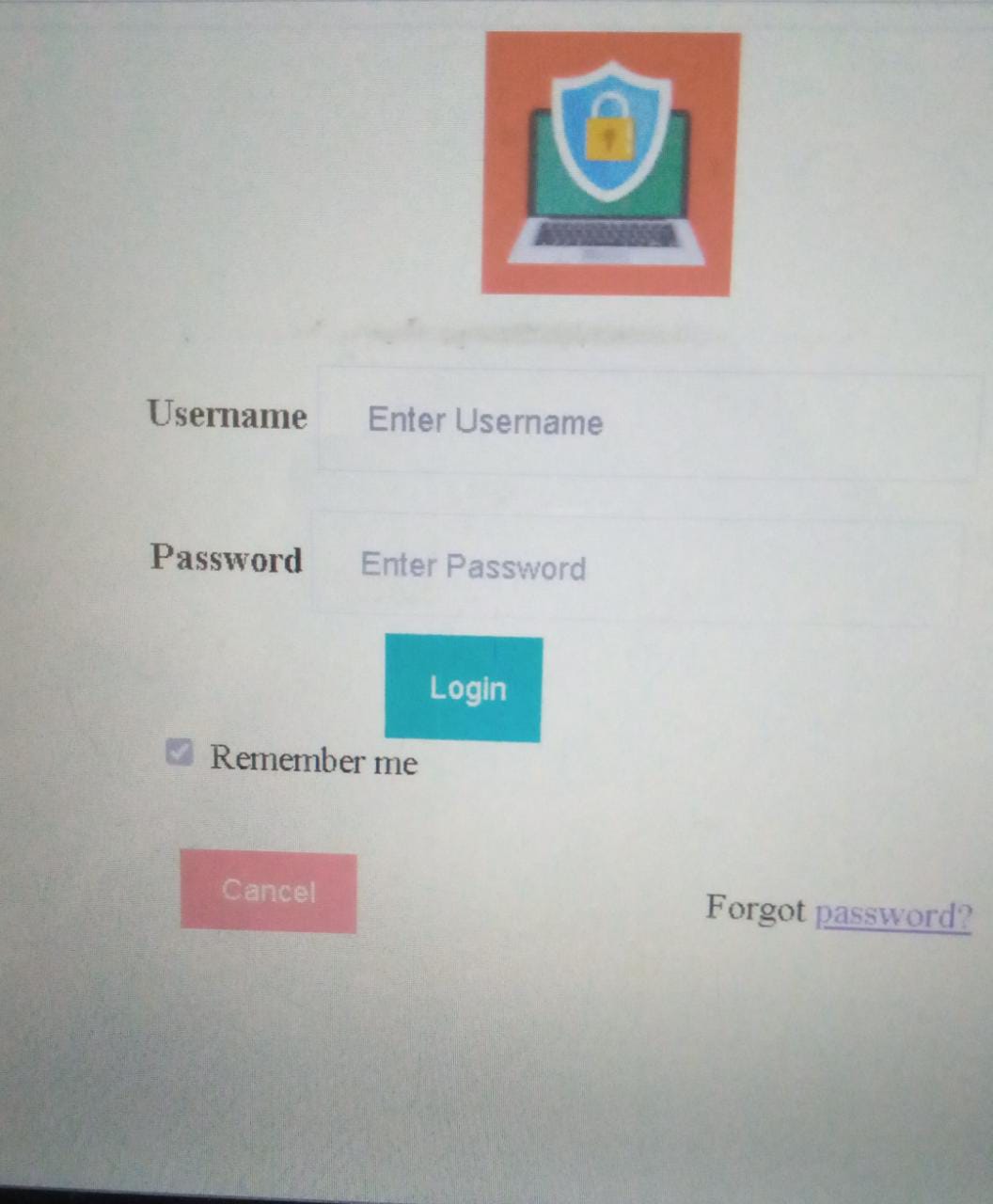
* **Splash Screen:**

****

* **Signup:**

****

* **Login:**

****