

**University Institute of Information Technology,**

**PMAS-Arid Agriculture University,**

**Rawalpindi Pakistan**

**Project Name**

**Malware Behavior Classification Using Deep Learning**

***By***

**Mudassar Riaz 17-ARID-1493**

**Bilal Rafique 17-ARID-1462**

**Ali Hamza 16-ARID-846**

***Supervisor*Mr. Zeeshan Javed**

***Bachelor of Science in Computer Science (2017-2021)***

**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.

Mudassar Riaz Ali hamza Bilal Rafique

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

**CERTIFICATE OF APPROVAL**

It is to certify that the final year project of BS (CS) “Malware Behavior Classification Using Deep Learning” was developed by “Mudassar Riaz**, 17-ARID-1493”**, “Bilal Rafique**, 17-ARID-1462”** and “**Ali Hamza, 16-ARID-846”** 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 Computer Science.

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

(Mr. Zeeshan Javed)

**Supervisor**

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

(Mr. Imran Khurram)

**External Examiner**

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

**Administrator UIIT**

**Executive Summary**

In public places, there is often a need for monitoring people and different activities going on, which can be referred later for many reasons including security. Appointing humans for this task involves many problems such as increased employee hiring, accuracy problem, trust, no proof for later use, and also the fact that a human can remember things till a certain time limit. Talking about the current security system, they use dumb still cameras with a continuous recording facility ir-respective of the fact that any event may happen or not. Moreover they are usually pointing at a specific user defined locations so more than one cameras are required to cover the entire region.

To prevent all these problems from prevailing, the CSCS is developed. It is a surveillance system, which provides solution to many of these problems. It is a stand-alone application which doesn’t require any computer to operate. It monitors different situations using a camera which is able to rotate intelligently based on sensor messages and captures the scene in the form of video or photos later reference as well.

**C**ustomizable **S**urveillance **C**ontrol **S**ystem **(CSCS)** is a surveillance system that can be assigned a sensor type as in our case a heat sensor is used, it works accordingly, rotates the camera upon event detection and perform user defined actions like capturing video and stores them, for the future use.

It is an embedded system consisting of Linux fox kit with embedded a running server application also a camera, USB storage device and a sensor node base station is attached with fox kit. LAN communication is used by user to download the videos and to operate the system manually.

**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.

Mudassar Riaz Ali Hamza Bilal Rafique

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

**Abbreviations**

|  |  |
| --- | --- |
| **SRS** | Software Requirement Specification |
| **PC** | Personal Computer |
|  |  |
|  |  |
|  |  |

**Table of Contents**

**Introduction 1**

[1.1 Brief](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523777) 2

[1.2 Relevance to Course Modules](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523779) 2

[1.3 Project Background](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523780) 3

[1.4 Literature Review](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523780) 3

[1.5 Methodology and Software Life Cycle](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523782) 5

**Problem Definition** 6

[2.1 Purpose](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523795) 7

[2.2 Product Functions](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523796) 7

[2.3 Proposed Architecture](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523804) 7

[2.4 Project Deliverables](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523806) 8

[2.5 Operating Environment](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523807) 8

[2.6 Assumptions and Dependencies](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523808) 8

**Requirement Analysis** 9

[3.1 Functional Requirments](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523823) 10

[3.2 Non – Functional Requirments](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523825) 11

[3.2.1 Usability](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523787) 11

[3.2.2 Reliability](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523787) 11

[3.2.3 Performance](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523787) 11

[3.2.4 Supportability](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523787) 11

[3.2.5 Design Constraints](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523787) 11

[3.2.6 Licensing Requirements](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523787) 11

[3.3 Use case Model](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523823) 12

[3.3.1 Use Case Diagarm](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523787) 12

[3.3.2 Actors Discription](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523787) 16

[3.3.3 Use Case Discription](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523787) 17

**The Design** 21

[4.1 UML Structural Diagrams](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523830) 22

[4.1.1 Component Diagram](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523787) 22

[4.1.2 System Component Diagram](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523787) 24

[4.1.3 Package Diagram](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523787) 25

[4.1.4 Deployment Diagram](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523787) 26

[4.2 UML Behavioral Diagrams](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523830) 27

[4.2.1 Activity Diagrams](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523787) 27

[4.2.2 State Machine Diagrams](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523787) 29

[4.3 UML Interaction Diagrams](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523830) 30

[4.3.1 Sequence Diagrams](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523787) 30

[4.4 Node Structure](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523830) 31

[4.5 Communication Design Protocol](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523830) 32

**Implementation** 33

[5.1 Communication Protocol Implementation](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523830) 34

[5.2 PC Application Implementation](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523830) 36

[5.3 Embedded Application Implementation](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523830) 38

[5.4 Wireless Sensor Application Implementation](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523830) 51

**Testing and Evaluation** 52

[6.1 Verification](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523830) 53

[6.1.1 Functional Testing](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523787) 53

[6.1.2 Static Testing](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523787) 57

[6.2 Validation](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523830) 57

[6.3 Usability Testing](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523830) 57

[6.4 Unit Testing](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523830) 57

* 1. [Integration Testing](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523830) 57

6.6 [System Testing](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523830) 57

**GUI** 58

**Future Work** 70

**References** 72

**List of Figures**

Fig 1.1 Block Diagram 8

Fig 2.1 Use Case Diagram 9

# Chapter 1: Introduction

# Brief

The aim of this project is to help cyber security experts to understand the behavior of malware based on API’s. In this project we have performed classification of malware families according to their behavior using API calls.

# Relevance to Course Modules

Our project is related to various courses we have studied in our degree which are mentioned below:

* Object oriented Programming
* Artificial Intelligence
* Software Engineering-I
* Software Engineering-II

Above mentioned courses not only helped us in developing the UML and Class diagrams as well as Use Cases of the model.

# Project Background

The struggle between security analysts and malware developers is a never-ending battle with the complexity of malware changing as quickly as innovation grows. Current state-of-the-art research focuses on the development and application of machine learning techniques for malware detection due to its ability to keep pace with malware evolution. This survey aims at providing a systematic and detailed overview of machine learning techniques for malware detection and in particular, deep learning techniques.

# Literature Review

The system that we are going to develop will classify the malware of different types based on its behavior using deep learning techniques. So, to develop this system we have used following research papers or literature:

The struggle between security analysts and experts and malware developers is a never ending battle with the complications of malware changing as quickly as innovation grows. Current state-of-the-art research focuses on the development and application of machine learning techniques for malware detection due to its ability to keep pace with malware evolution. This survey aims at providing an organized and detailed overview of machine learning techniques for malware detection and in particular deep learning techniques. It provides absolute explanation of the methods and features in a traditional machine learning work ﬂow for malware detection and classiﬁcation it explores the challenges and limitations of traditional machine learning and it analyzes recent trends and developments in the ﬁeld with special importance on deep learning approaches. [1] (Gibert, Mateu, & Planes, 2010)

With the development of the Internet, malwares have also been expanded on the network systems rapidly. In order to deal with the diversity and amount of the variants, a number of automated behavior analysis tools have emerged as the time requires. Yet these tools produce detailed behavior reports of the malwares, it still needs to specify its category and judge its criticality manually. In this paper, we propose an automated malware classification approach based on the behavior analysis. We firstly perform dynamic analyses to obtain the detailed behavior profiles of the malwares, which are then used to abstract the main features of the malwares and serve as the inputs of the Back Propagation (BP) Neural Network model. The experimental results demonstrate that our classification technique is able to classify the malware variants effectively and detect malware accurately. [2]( Zhi-Peng PAN1, a, Chao FENG1 and Chao-Jing TANG1)

The use of operating system API calls is a promising task in detecting PE-type malware in the Windows operating system. This task is officially defined as running malware in an isolated sandbox environment, recording the Windows operating system’s API calls and sequentially analyzing these calls. Here, we have analyzed 7107 different malicious software belonging to various families such as virus, backdoor, trojan in an isolated sandbox environment and transformed these analysis results into a format where different classification algorithms and methods can be used. First, we’ll explain how we got the malware, and then we’ll explain how we’ve got this software bundled into families.

Finally, we will describe how to perform malware classification tasks using different computational methods for the researchers who will use the data set. [3]( Ferhat Ozgur Catak Cyber Security Institute TUBITAK-BILGE)

# Analysis from Literature Review (in the context of our project)

The literatures which are mentioned above help us to develop a classification system based on behavior of malware. So, the knowledge that we have gained from these literature are first we have understood the API calls sequence of different malwares that they have used to gain information from system. The API calls gives us source full information about malware that which type of malware uses which of API calls and what type of action it will perform in system. We have also learned that how to use different deep learning techniques or algorithms to perform classification.

# Methodology and Software Lifecycle for This Project

There are different types of methodologies are used to building a software or any of the project. We have studied all the types of methodologies that can be used but from all of them we select the method that best fit to our project is “Extreme Programming”.

We are selecting this model because it is:

Best suited for : Projects that require maintaining stringent stages and deadlines or projects that have been done various times over where chances of surprises during the development process are relatively high.

One more reason is that this method is applied where the requirements are not very much clear. So that will happen with our project too so that’s why we are selecting this Method.

In systems design, and particularly software design, a common methodology for the development of a new system is the Systems Development Life Cycle, or SDLC. The SDLC contains the following phases of systems development:

• **Planning**

Determine the purpose of the system.

• **Analysis**

Determine what the system needs to do, the goals for the system and how to determine if those goals have been met.

• **Design**

Determine how the system will work, what the overall architecture is, and determine what steps would need to be taken to construct an actual system.

• **Implementation**

Using the existing design, we will construct a system to meet the requirements of the project.

• **Testing**

Establish that the constructed system actually does meet the requirements detailed in the design.

• **Maintenance**

Fix bugs in the system, which are essentially differences between the design (requirements) and the constructed system (reality). As the design inevitably changes, update the actual system to match these changes.

**Chapter 2: Problem Definition**

# Problem Statement

The Malware threat has been with us since the dawn of computing. The earliest documented virus appeared during the 1970s. It was known as the Creeper Worm and was an experimental self-replicating program that copied itself to remote systems and displayed the message: “I’m the creeper, catch me if you can”. Later, in the early 80s, appeared Elk Cloner, a boot-sector virus that targeted Apply II computers. From these simple beginnings, a massive industry was born and, since then, the fight against malware has never stopped.

# Deliverables and Development Requirements

* Documentation
* Sandbox Environment
* Features Extraction
* Desktop Application
* Classification System

# Proposed Architecture

We are going to develop deep learning model for malware classification based on malware behavior. Security is the thing that every system requires that wants to communicate over the internet. Many of the malicious codes or files attached to your files when you are going to download anything from the internet. Different type of malware are present on the internet world, every specie has its own purpose and method that how to attack on any computer. So that’s why we are going to build classification model that can classify any type of malware based on its behavior.

**2.4 Product Functions**

1. Run malware sample in sandbox environment.
2. Extract Features from malware reports.
3. Test and train model
4. Perform Classification
5. After classification perform prediction

**2.5 Operating Environment**

* VMware
* Kali Linux
* Cuckoo Sandbox
* Virtual Box
* Windows 7 or Windows 10(Depends on user)

# Chapter 3: Requirement Analysis

Software Requirements Specification (SRS) report should be included in this chapter.

# Use Cases

Use cases are a widely used and highly regarded format for capturing requirements. Before writing functional requirement use cases can help you to understand the requirements in the way user expect. Following table presents you not only the template to write use case(s) as well as guides you to write each section with example.

|  |  |
| --- | --- |
| Use case ID | U\_id:1 |
| Use case Name | Submit Malware Samples |
| Actor Name | Direct User |
| Description | By selecting the files from the system after processing it will generate report about selected malware sample |
| Trigger | To accomplish this task we Click on button (Select File) |
| Precondition | NO info about samples |
| Post Condition | Report will provide complete information on the basis on Malware |
| Normal Flow | 1.Chose the malware samples from system  2.Submit malware samples to sandbox  4. Then select action report generate. |
| Alternative Flow | If success fully then move to the next process else request user to try again |
| Expectation | During taking action some errors can be occurs  1.No file selected  2.Folder having no malware samples  3.Uploading Error  In such type of condition user request to restart from initial stage. |
| Includes: | Null |

|  |  |
| --- | --- |
| Use case ID: | U\_id:2 |
| Use case Name | Sandbox Environment |
| Actor Name | System |
| Description: | After submitting the malwares in sandbox environment like cuckoo sandbox. Then cuckoo will run those samples in isolated virtual environment like on windows 7 installed on virtual box and connected with cuckoo sandbox |
| Trigger | For this process cuckoo sandbox will run samples on isolated environment using connected network. |
| Precondition | Isolated environment must be configured with cuckoo sandbox and in restore condition. |
| Post Condition | Generate report and save in database. |
| Normal Flow: | 1. Run malware samples in isolated environment.  2. Report generation based on executed malware samples  3.Check Successful or not |
| Alternative Flow: | If save successfully then move to the next step else again request save. |
| Exception: | During the performing action some error can be take place:   1. Cuckoo sandbox is not configured with isolated environment properly. 2. Isolated environment is not in restore mode. |
| Includes: | Report Generation based on executed malware sample. |

|  |  |
| --- | --- |
| Use case ID: | U\_id:3 |
| Use case Name | Feature Extraction |
| Actor Name | User |
| Description: | After the report is generated based on executed malware sample. Next step is to extract features from report in JSON form. |
| Trigger: | To complete this all reports should be collected from cuckoo sandbox. |
| Precondition | No feature is extracted. |
| Post condition | After extracting features save in CSV file. |
| Normal Flow | 1. Get report from Cuckoo Sandbox.  2. Extract features from reports.  3. Prepare dataset from these extracted features. |
| Alternation | Successfully move next else again try to extract features from report. |
| Exception: | During the performing action some error can be take place  1. Index number of desired feature is not defined or correct.  2. Desired feature may not exist.  3.Fail to Extract features |
| Includes: | Dataset |

|  |  |
| --- | --- |
| Use case ID: | U\_id:4 |
| Use case Name | Classification |
| Actor Name | Direct user |
| Description | For Classification of Malware first extract features then prepare dataset based on these extracted features. After dataset perform classification using deep learning model. |
| Trigger | Dataset is prepared based on extracted features. |
| Precondition | Non Classified dataset |
| Post Condition | Classified |
| Normal Flow | 1)Get Dataset  2) Classify the malware in families  3) Then save trained model in H5 file for prediction. |
| Alternative | If success then movie to the final steps else go back for classification |
| Exception | Dataset is not prepared properly. |
| Includes | Null |

**3.2 Functional Requirements:**

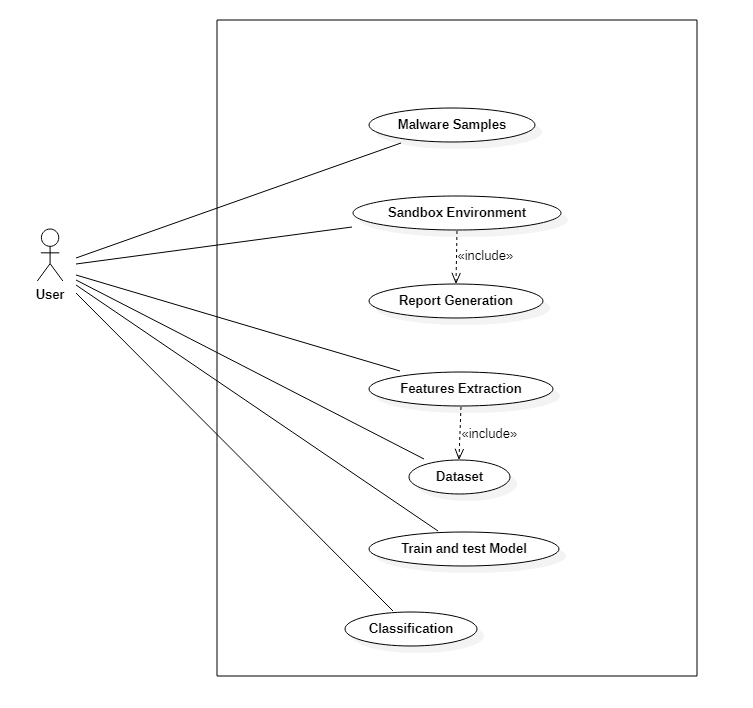
|  |  |
| --- | --- |
| **Functional Requirement No.** | **Functional Requirement Description** |
| Req.1 | User Collect Malware Samples. |
| Req.2 | Submit Malware Samples in sandbox environment. |
| Req.3 | Get generated reports from sandbox. |
| Req.4 | Extract features from these reports and prepare dataset |
| Req.5 | Now test and train dataset. |
| Req.6 | After testing and training prepare model for classification |

**3.3 Non-Functional Requirements**

|  |  |
| --- | --- |
| **Non-Functional Requirement No.** | **Non-Functional Requirement Description** |
| NFR1 | System will take sandbox generated malware reports from user as input and process it. |
| NFR2 | System will extract features from malware reports and prepare CSV File. |
| NFR3 | System will provide interface by using Tkinter. |
| NFR4 | MD5 will be used in for prediction. |

1. **Usability:** System should be easy to extend. The code should be written in a way that it favors implementation of new functions. It will provide the up to date information with good performance to satisfy user needs.
2. **Reliability:** This app should provide appropriate answers to the user. This app should be able to interact efficiently with the user.
3. **Integrity:** This desktop application will requires specific android version to run. It also requires an active internet connection to work and to exchange queries to provide information to the user.

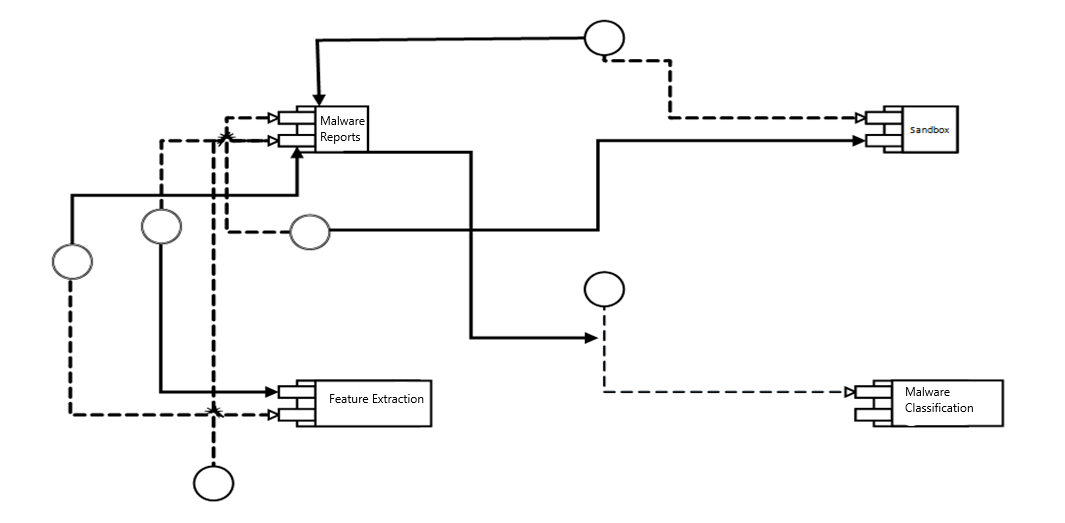
**3.4 Use Case Diagram**

****

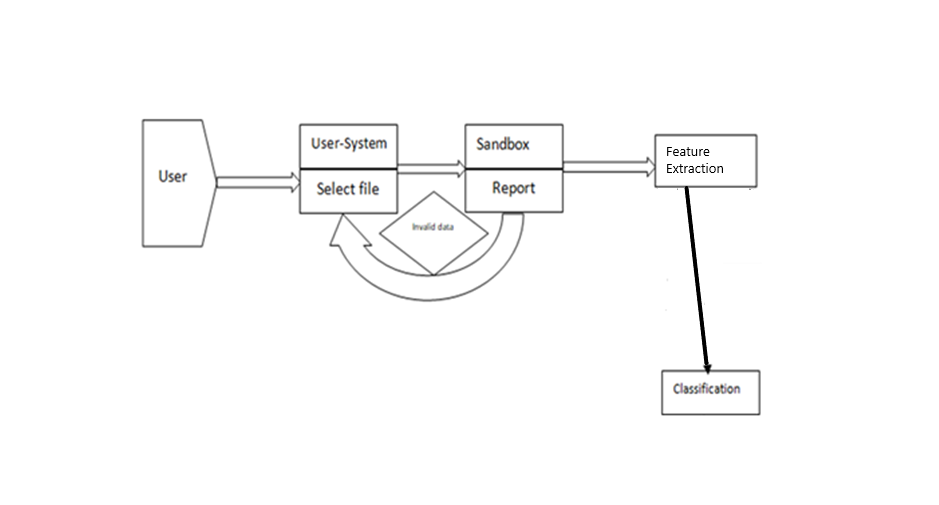
Chapter 4: Design and Architecture

**4.1 UML Structural Diagrams**

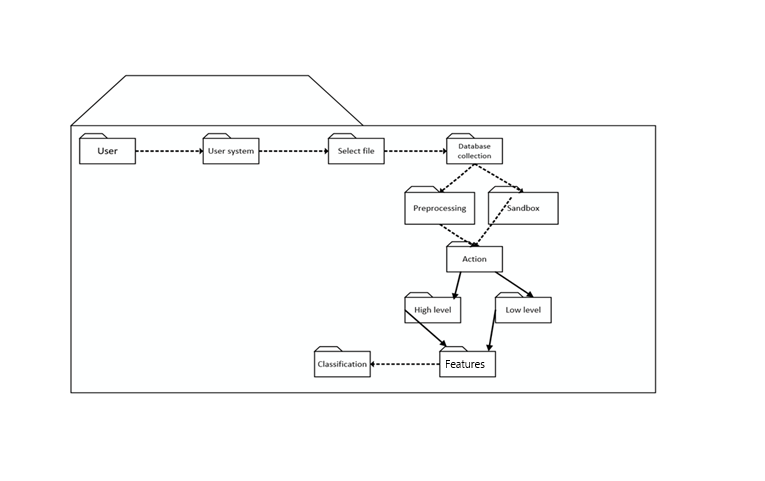
[4.1.1 Component Diagram](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report.docx#_Toc268523787)

****

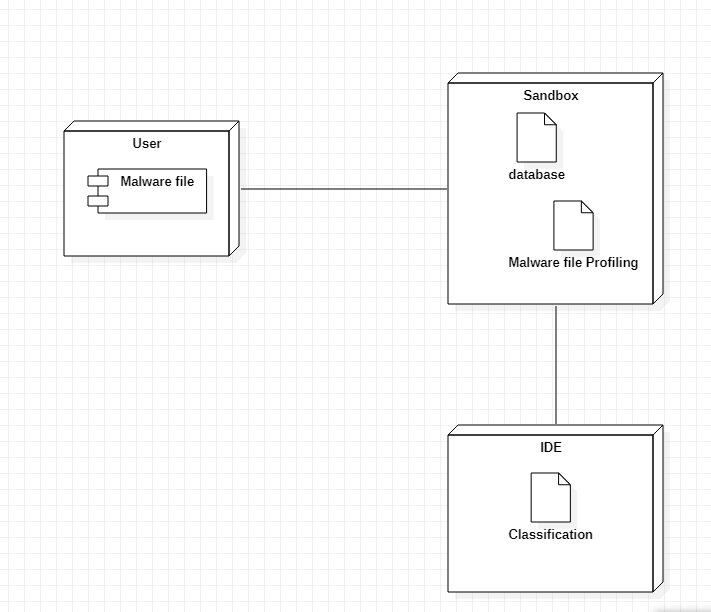
[**4.1.2 System Component Diagram**](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report.docx#_Toc268523787)

****

[**4.1.3 Package Diagram**](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report.docx#_Toc268523787)

****

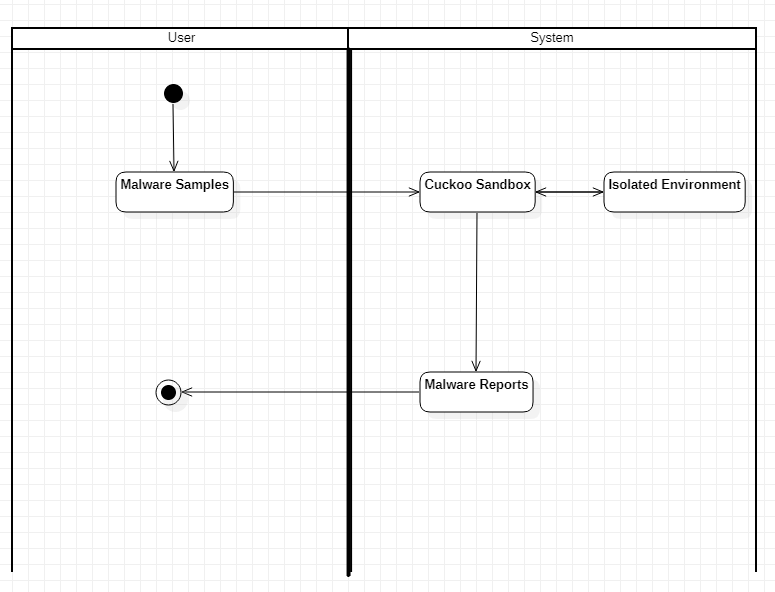
[**4.1.4 Deployment Diagram**](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report.docx#_Toc268523787)

****

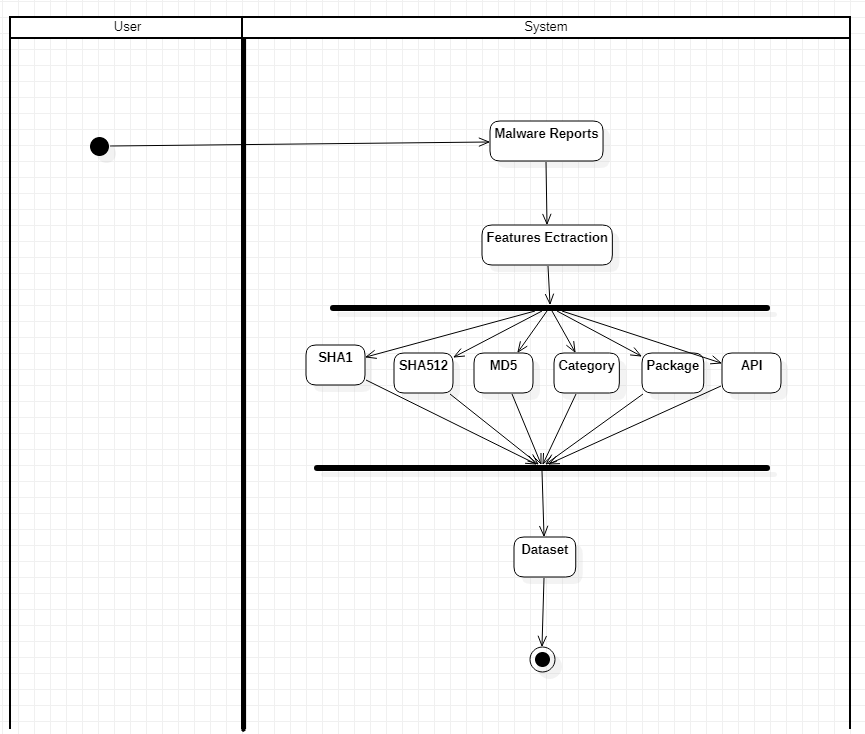
[**4.2 UML Behavioral Diagrams**](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report.docx#_Toc268523830)

[**4.2.1 Activity Diagrams**](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report.docx#_Toc268523787)

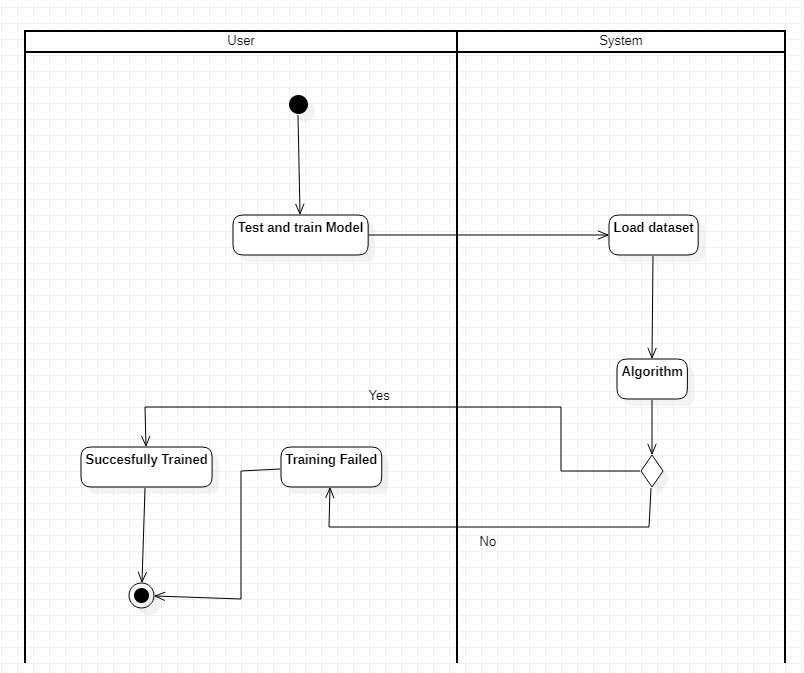
**Run Malware Samples:**

****

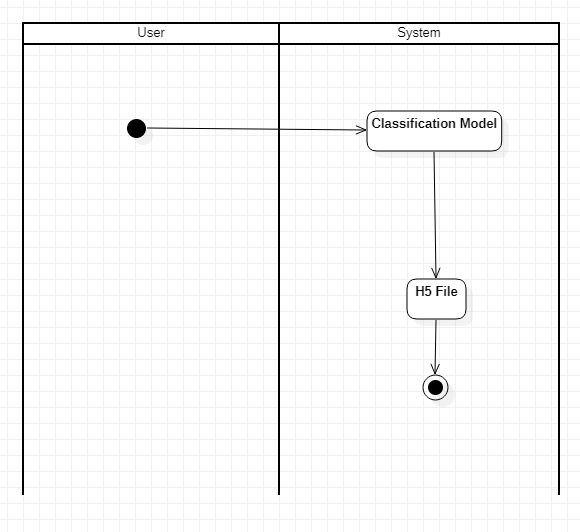
**Feature Extraction:**

****

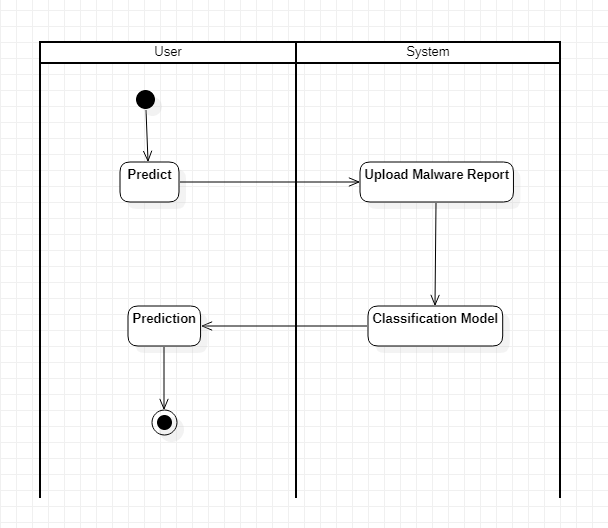
**Test and train model:**

****

**Saving Classification Model in H5 File:**

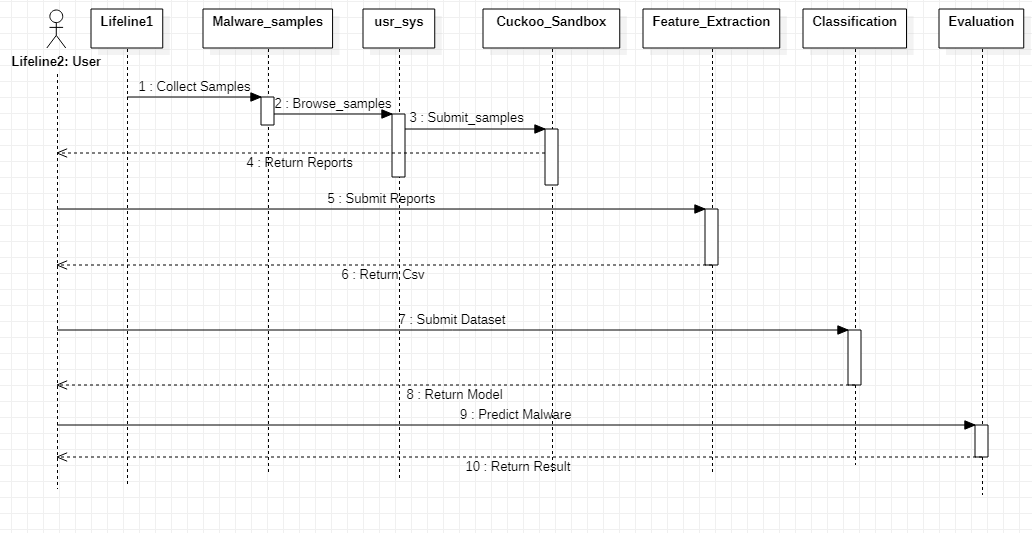
****

**Predict Malware Using Model:**

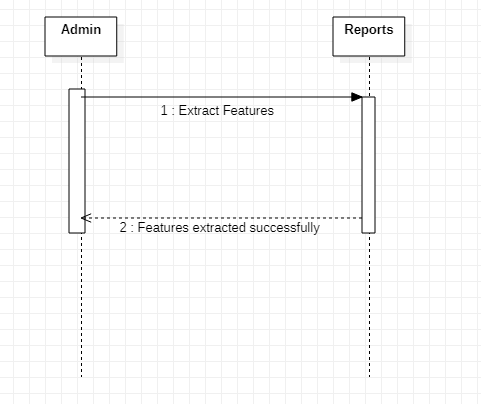
****

[**4.3 UML Interaction Diagrams**](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report.docx#_Toc268523830)

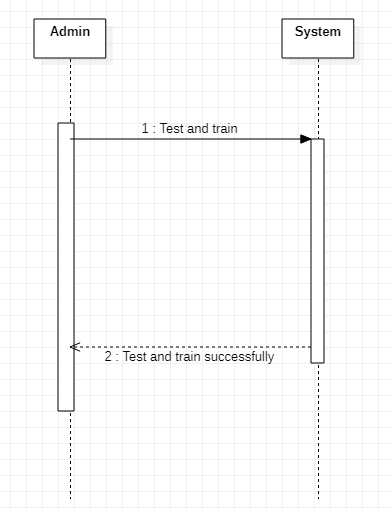
4.3.1 Sequence Diagram



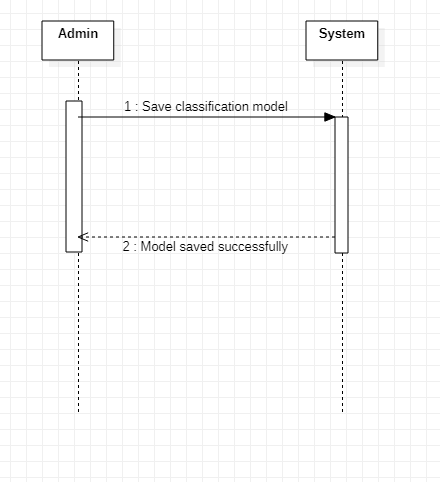
Extract Features:



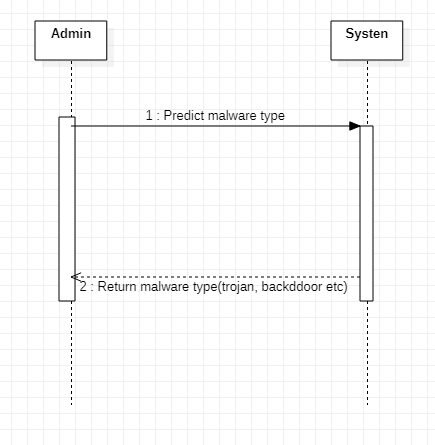
Test and train model:



Save model:



Predict Malware:



Chapter 5: Implementation

# Tools and Technologies

5.1.1 Python:

Python is a general-purpose programming language. Unlike HTML, CSS, and JavaScript, it can be used for other types of programming and software development besides web development.

We used Python because it is the best fit for machine learning, deep learning and AI-based projects.

We used it because of consistency and access to great libraries and frameworks for machine learning.

5.1.2 Tkinter:

Tkinter is a Python binding to the Tk GUI toolkit. It is the standard Python interface to the Tk GUI toolkit, and is Python's de facto standard GUI. Tkinter is included with standard Linux, Microsoft Windows and Mac OS X installs of Python. The name tkinter comes from Tk interface.

5.1.3 Cuckoo Sandbox:

Cuckoo Sandbox is the leading open source automated malware analysis system. You can throw any suspicious file at it and in a matter of minutes Cuckoo will provide a detailed report outlining the behavior of the file when executed inside a realistic but isolated environment.

5.1.4 Tensorflow:

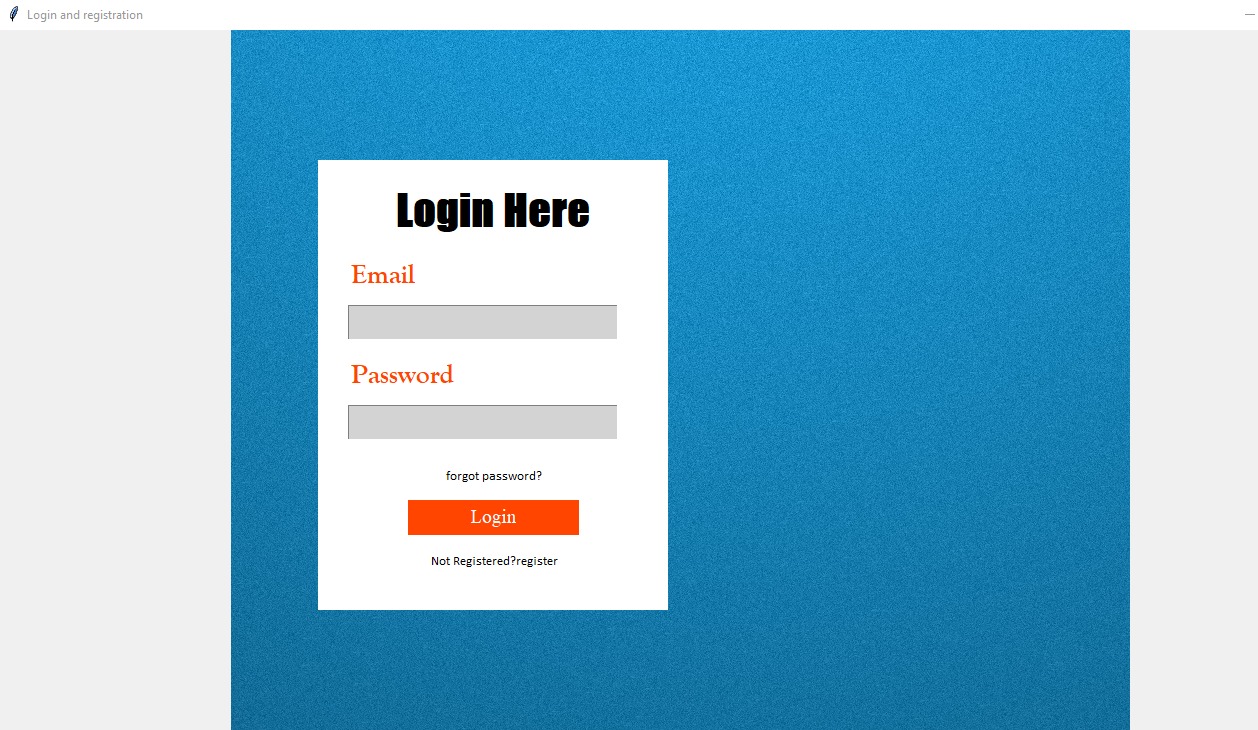
It is an open source artificial intelligence library, using data flow graphs to build models.

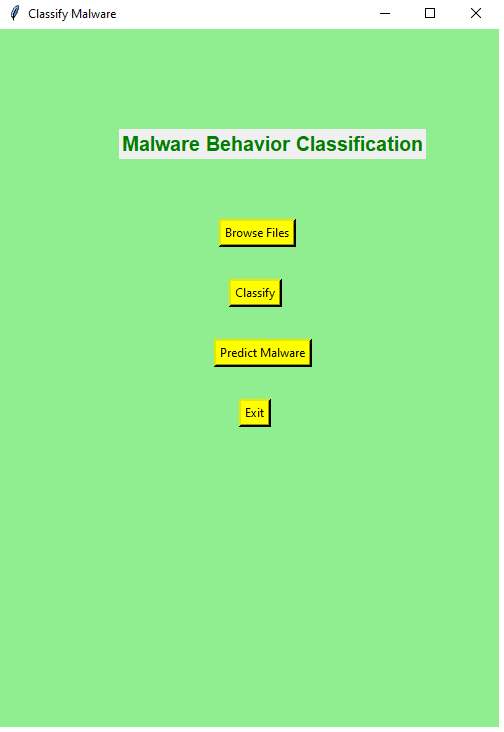
We used TensorFlow mainly for: Classification, Perception, Understanding, Discovering, Prediction and Creation

# Network and Protocol Choice

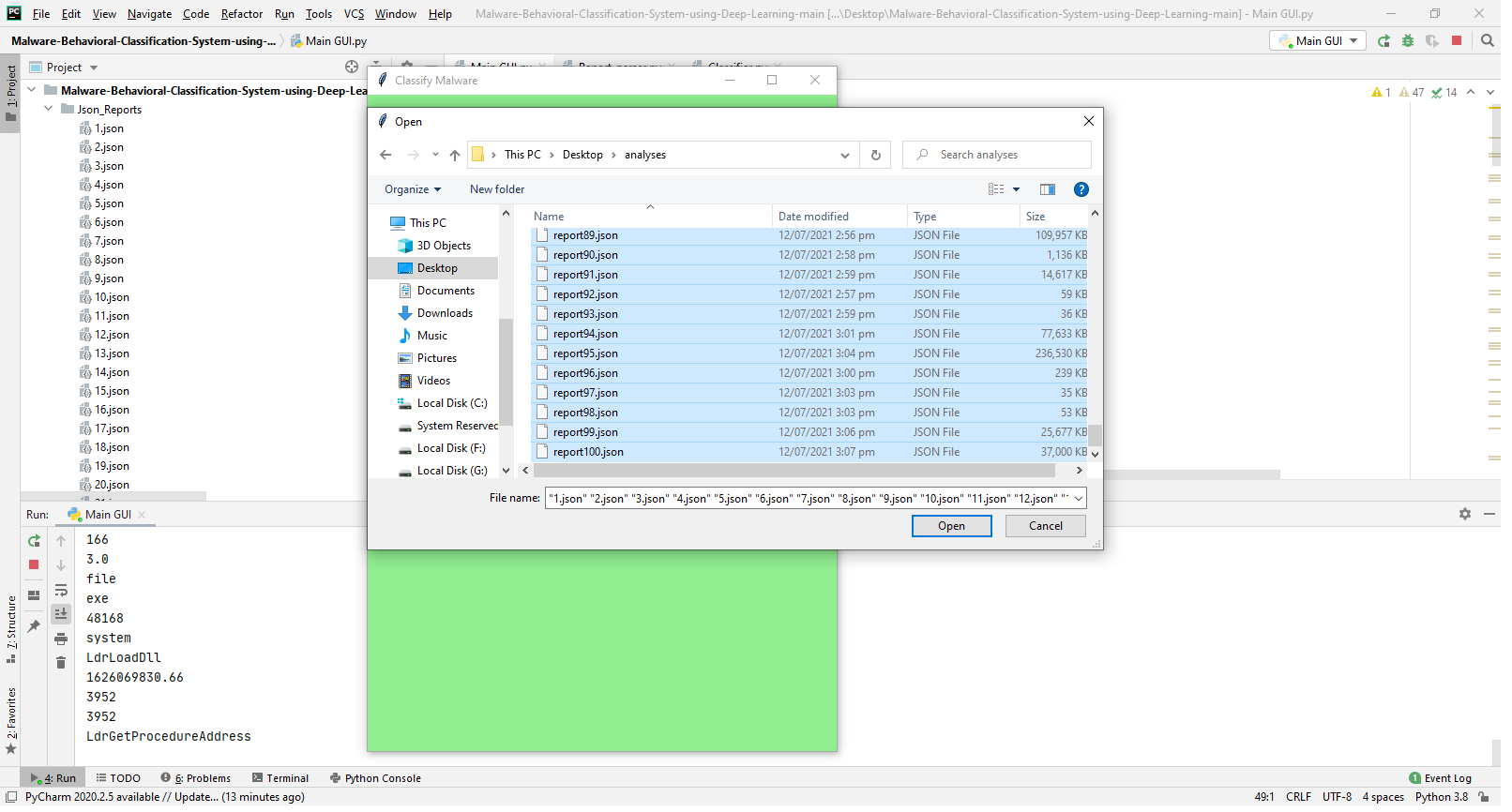
* **Design:** The designing of this malware classifier and the diagram have been done through Visio and StarUML.
* **MySQL:** It is a database system.
* Here the libraries of pycharm and tensorflow are mentioned that have been used in our project
* **Tkinter:** Tkinter is a Python binding to the Tk GUI toolkit. It is the standard Python interface to the Tk GUI toolkit,[1] and is Python's de facto standard GUI.
* **Tensorflow:** TensorFlow is a free and open-source software library for machine learning. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks. Tensorflow is a symbolic math library based on dataflow and differentiable programming.

# User Interface

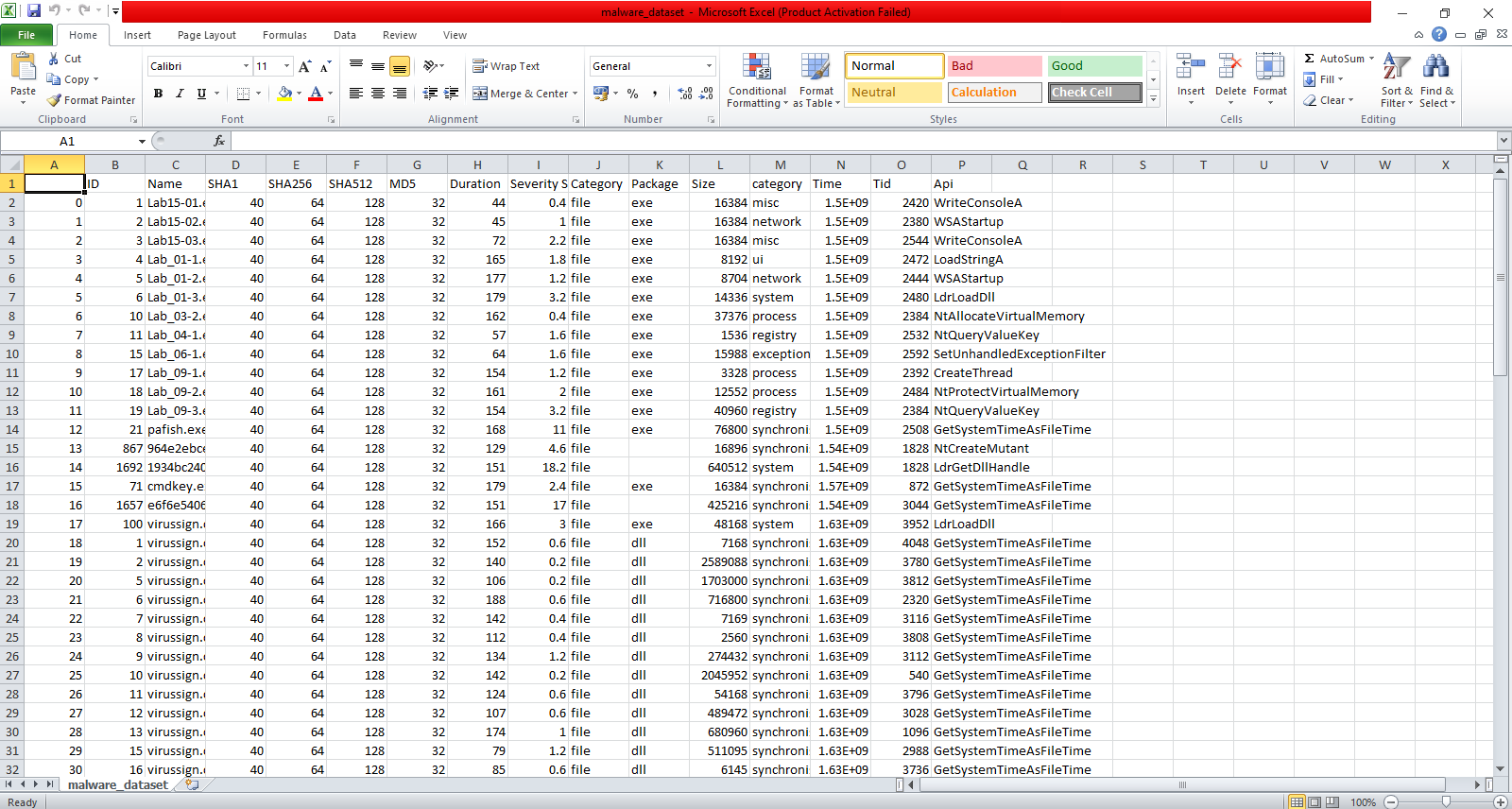




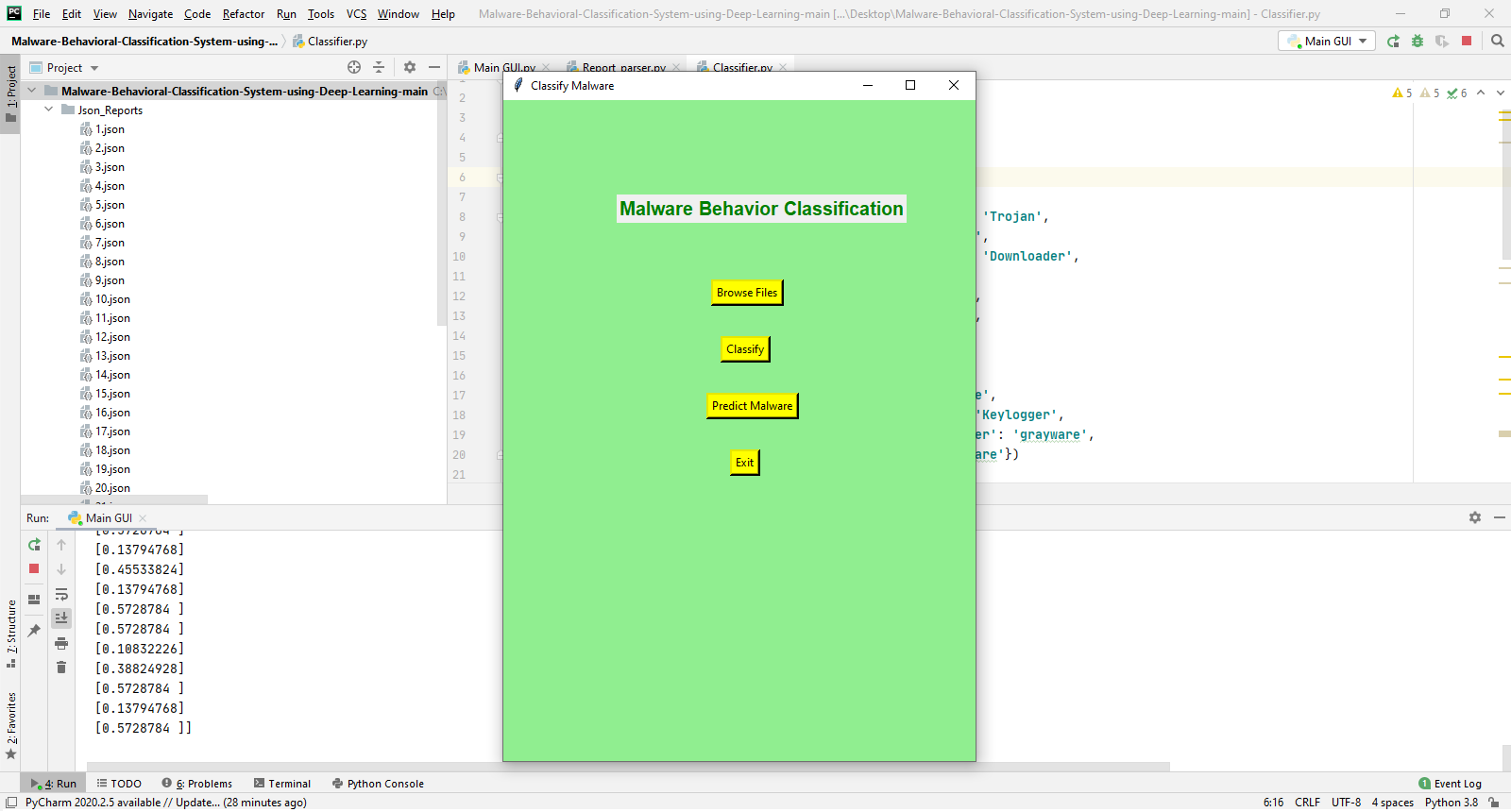
**Select 110 JSON files to extract features:**

****

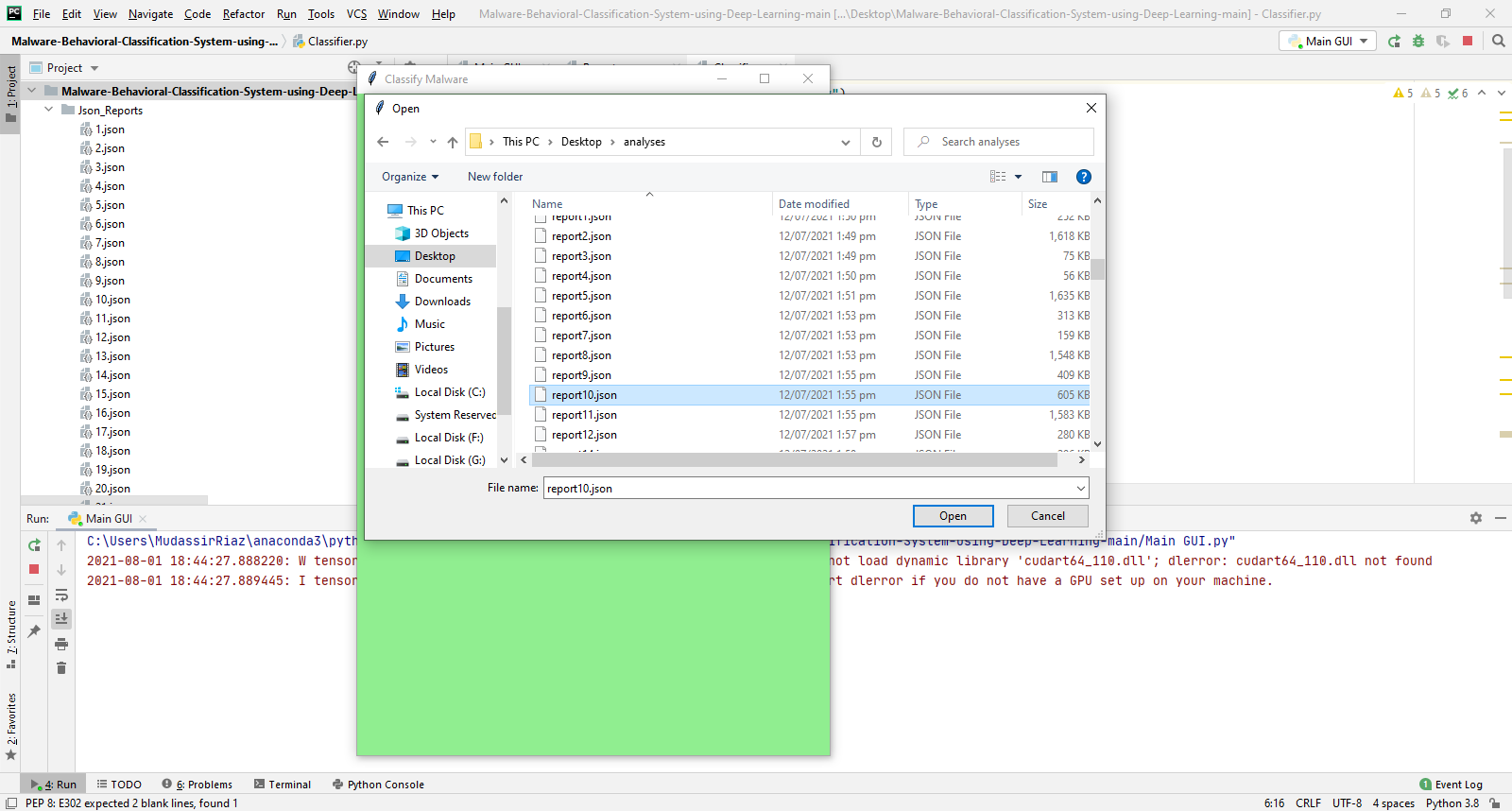
**Extracted features in CSV file:**

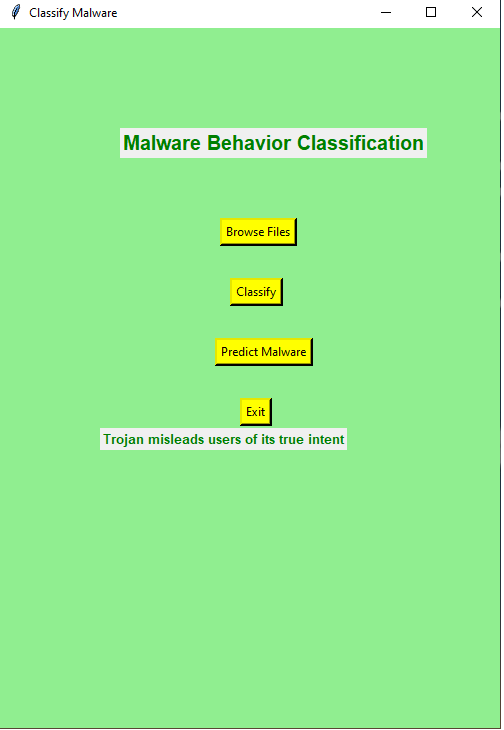
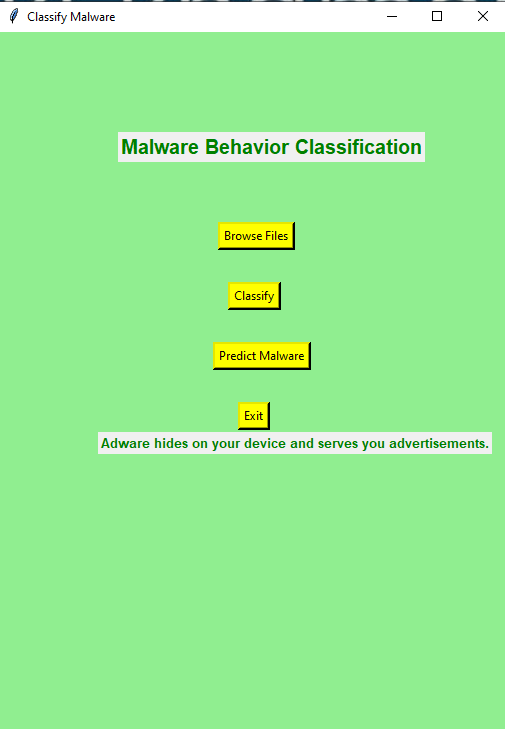
****

**Perform Classification:**

****

**Predict Malware:**

****

****

# Chapter 6: Testing and Evaluation

# Module/Unit testing

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Project Name: Malware Behavior Classification | | | | | |
| Test Case: 1 | | | | | |
| Test Case ID | | | T\_id:01 | | |
| Test priority (low, medium, high) | | | High | | |
| Module Name | | | Check Cuckoo conf. | | |
| Description | | | Admin Run command of cuckoo to check cuckoo is configure properly. | | |
| Test Title | | | Test configuration | | |
| Precondition | | | Cuckoo configure or not. | | |
| Dependency | | | Running cuckoo sandbox.  Mongo dB should start.  Virtual machine starts or restore mode. | | |
| S. | Test steps | Expected result | Actual result | Status fail or pass | Note |
| 1 | * Open terminal * Run command cuckoo -d | At the end of console  The message will appear  “waiting for analysis”. | Waiting for analysis. | Pass | If cuckoo not configure it will give message cuckoo could not find client. |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Project Name: Malware Behavior Classification | | | | | |
| Test Case: 2 | | | | | |
| Test Case ID | | | T\_id:02 | | |
| Test priority (low, medium, high) | | | High | | |
| Module Name | | | Upload malware sample | | |
| Description | | | Upload sample to cuckoo sandbox for malware analysis. | | |
| Test Title | | | Submit malware sample to cuckoo. | | |
| Precondition | | | No Task is running. | | |
| Despondency | | | Running cuckoo sandbox.  Mongo dB should start.  Virtual machine starts or restore mode. | | |
| S. | Test steps | Expected result | Actual result | Status fail or pass | Note |
| 1 | * Open cuckoo web interface * Browse file * Upload file | It should be submitted to cuckoo. Cuckoo Perform complete analysis.  The message will be shown like  “Successfully completed”. | It shows Message successfully completed analysis. | Pass | If no internet it might not show any result. |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Project Name: Malware Behavior Classification | | | | | | |
| Test Case: 3 | | | | | | |
| Test Case ID | | | T\_id:03 | | | |
| Test priority(low, medium, high) | | | High | | | |
| Module Name | | | Check Cuckoo Result and collect report. | | | |
| Description | | | Check the cuckoo sandbox result. After successful malware analysis it will generate malware report in JSON form. Then collect these reports. | | | |
| Test Title | | | Check result and collect reports. | | | |
| Precondition | | | Cuckoo produce result or not | | | |
| Despondency | | | Mongo dB started  VM in restore mode | | | |
| s. | Test steps | Expected result | | Actual result | Status fail or pass | Note |
| 1 | * Open terminal * Run command   Cuckoo –d   * Cuckoo web * Open browser * Type localhost:8000 * Submit upload file * Submit file | At the end of cuckoo console show message  “Analysis is completed”  And browser it will show All detail  If score is 0.0 this means cuckoo is not configure properly  It the score between  1-10  This mean cuckoo is producing accurate result. And after it will generate JSON report based on this malware analysis. | | Score 3.5 | Pass | If score is 0.0 it means we have to review cuckoo configuration |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Project Name: Malware Behavior Classification | | | | | |
| Test Case: 4 | | | | | |
| Test Case ID | | | T\_id:04 | | |
| Test priority (low, medium, high) | | | High | | |
| Module Name | | | Extract Features | | |
| Description | | | After uploading file and completion of malware analysis it will generate a malware JSON reports. Extract Features from these reports | | |
| Test Title | | | Features Extraction | | |
| Precondition | | | Raw JSON reports. | | |
| Despondency | | | Desktop App is open | | |
| S. | Test steps | Expected result | Actual result | Status fail or pass | Note |
| 1 | * Open Desktop App * Browse file * Upload file | Return extracted features from JSON reports. | It will generate a CSV file based on these extracted features. | Pass |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Project Name: Malware Behavior Classification | | | | | |
| Test Case: 5 | | | | | |
| Test Case ID | | | T\_id:05 | | |
| Test priority (low, medium, high) | | | High | | |
| Module Name | | | Test and Train Model | | |
| Description | | | After the features are extracted and saved in CSV file. The next step is training model based on these extracted features. The most import feature on which model is trained is API. After the model is trained it will save model in H5 file. | | |
| Test Title | | | Model Training on Extracted Features. | | |
| Precondition | | | Raw JSON reports. | | |
| Despondency | | | Desktop App is open | | |
| S. | Test steps | Expected result | Actual result | Status fail or pass | Note |
| 1 | * Open Desktop App * Pass CSV file | Based on CSV file the system will test and train model using deep learning techniques | It will test and train model based on extracted features most importantly on API feature and save model in H5 file. | Pass |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Project Name: Malware Behavior Classification | | | | | |
| Test Case: 6 | | | | | |
| Test Case ID | | | T\_id:06 | | |
| Test priority (low, medium, high) | | | High | | |
| Module Name | | | Use model to for final result | | |
| Description | | | Use model saved in H5 file to check final result. The system will return final result. The result shows the malware using this trained model. | | |
| Test Title | | | Use model to check type of malware. | | |
| Precondition | | | Desktop app running. | | |
| Despondency | | | Desktop App is open | | |
| S. | Test steps | Expected result | Actual result | Status fail or pass | Note |
| 1 | * Open Desktop App * Browse file * Upload JSON file | The system will extract the API from submitted JSON file and return the final result. | It will print message that it is Trojan or backdoor or any other type of malware based on extracted API. | Pass |  |

# Integration Testing

**Integration testing** is a level of software testing where individual units or components are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. Test drivers and test stubs are used to assist in Integration Testing. The tests are conducted to ensure that the components are working properly after interfacing.

Integration testing of the system is performed as follows:

* Integrate JSON report parser for extracting features.
* Integrate Malware classification model for classification.

# Chapter 7: Conclusion and Future Work

This chapter concludes the project and highlights future work.

# Conclusion

In our developed model we can classify the malware according to its behavior. First it extract's the features from malware reports and generate a CSV file with extracted features. Then after the CSV file is generated we classify the API's of different malware's according to its type. Then we train our model based on these extracted API's. When all the above processes are complete then our trained model shows the type of malware based on its API.

# Future Work

The system required some improvements' which will help the system to work in more efficient way. The system that we have developed can only classify the malware based on API feature. So, in future we will train our model on more features like networks, signature, category and other important features.

References

References to any book, journal paper or website should properly be acknowledged. Please consistently follow the style. The following are few examples of different resources i.e. journal article, book, and website.

1 Lyda M.S. Lau, Jayne Curson, Richard Drew, Peter Dew and Christine Leigh, (1999), Use Of VSP Resource Rooms to Support Group Work in a Learning Environment, ACM 99, pp-2. (Journal paper example)

2 Hideyuki Nakanishi, Chikara Yoshida, Toshikazu Nishmora and TuruIshada, (1996), FreeWalk: Supporting Casual Meetings in a Network, pp 308-314 (paper on web) http://www.acm.org/pubs/articles/proceedings/cscw/240080/p308-nakanishi.pdf

3 Ali Behforooz& Frederick J.Hudson, (1996), Software Engineering Fundamentals, Oxford University Press. Chapter 8, pp255-235. (book reference example)

4 Page Author, Page Title, http://www.bt.com/bttj/archive.htm, Last date accessed. (web site)