**Detecting Malware Using Deep Learning**



***By***

**Muhammad Aamir Manzoor**

**17-ARID-1496**

**Abdul Rehman**

**16-ARID-839**

***Supervisor*Mr. Zeeshan Javed**

**University Institute of Information Technology,**

**PMAS-Arid Agriculture University,**

**Rawalpindi**

**2021**

***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.

Muhammad Aamir Manzoor Abdul Rehman

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

**CERTIFICATE OF APPROVAL**

It is to certify that the final year project of BS (CS) “Detecting Malware using Deep Learning” was developed by “**Muhammad Aamir Manzoor, (17-Arid-1496)”**, and “**Abdul Rehman, (16-Arid-839)”** 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.

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

**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 irrespective 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.

Muhammad Aamir Manzoor Abdul Rehman

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

**Abbreviations**

|  |  |
| --- | --- |
| **SRS** | Software Requirement Specification |
| **PC** | Personal Computer |
| **SDLC** | System Development Life Cycle |
| **FYP** | Final Year Project |
| **UAAR** | University of Arid Agriculture Rawalpindi |
| **UC** | Use Case |
| **FR** | Functional Requirements |
| **NFR** | Non Functional Requirements |
| **TC** | Test Case |
| **SDK** | System Development Kit |

**Table of Contents**

**Introduction 1**

[**Chapter 1: Introduction.** 1](#_Toc79618372)

[1.1.Brief 1](#_Toc79618373)

[1.2.Relevance to Course Modules 1](#_Toc79618374)

[1.3.Project Background 2](#_Toc79618375)

[1.4.Literature Review 2](#_Toc79618376)

[1.5.Analysis from Literature Review (in the context of your project) 3](#_Toc79618377)

[1.6.Methodology and Software Life cycle 3](#_Toc79618378)

[**Chapter 2: Problem Definition** 5](#_Toc79618379)

[2.1. Problem Statement 5](#_Toc79618380)

[2.2. Deliverables and Development Requirements 5](#_Toc79618381)

[2.3. Proposed Architecture 5](#_Toc79618382)

[2.4. Operating Environments 6](#_Toc79618383)

[2.5. Assumptions and Dependencies 6](#_Toc79618384)

[**Chapter 3: Requirement Analysis** 7](#_Toc79618385)

[3.1.Functional Requirements 7](#_Toc79618386)

[3.1. Non-Functional Requirements 17](#_Toc79618387)

[3.2.1. Usability 17](#_Toc79618388)

[3.2.2. Reliability 17](#_Toc79618389)

[3.2.3. Integrity 17](#_Toc79618390)

[3.2.4. Performance 17](#_Toc79618391)

[3.2.Use Case Model 18](#_Toc79618392)

[3.3.Actor Description 18](#_Toc79618393)

[3.4.Use Case Description 19](#_Toc79618394)

[**Chapter 4: Design and Architecture** 20](#_Toc79618395)

[4.1. UML Structural Diagrams 20](#_Toc79618396)

[4.1.1. Class Diagram 20](#_Toc79618397)

[4.1.2. Component Diagram 21](#_Toc79618398)

[4.1.3. Deployment Diagram 22](#_Toc79618399)

[4.2. UML Behavioral Diagrams 23](#_Toc79618400)

[4.1.3. Activity Diagrams 23](#_Toc79618401)

[4.3. UML Interaction Diagrams 29](#_Toc79618402)

[4.3. Sequence Diagrams 29](#_Toc79618403)

[**Chapter 5: Implementation** 33](#_Toc79618404)

[5.1. Tools and Technologies 33](#_Toc79618405)

[5.1.1. PyCharm 33](#_Toc79618406)

[5.1.2. Tkinter 33](#_Toc79618407)

[5.1.3. Wamp Server 33](#_Toc79618408)

[5.1.4. MS Word 33](#_Toc79618409)

[5.1.5. MS Power Point 33](#_Toc79618410)

[5.1.6. Python 3.9 33](#_Toc79618411)

[5.1.7. Matplotlib 34](#_Toc79618412)

[5.1.8. TensorFlow 34](#_Toc79618413)

[5.1.9. Pandas 34](#_Toc79618414)

[5.1.10. MySQL Connector 34](#_Toc79618415)

[5.1.11. NumPy 34](#_Toc79618416)

[5.1.12. Star UML 34](#_Toc79618417)

[5.1.13. Cuckoo Sandbox 34](#_Toc79618418)

[5.2. User Interface 35](#_Toc79618419)

[**Chapter 6: Testing and Evaluation** 38](#_Toc79618420)

[**Conclusion and Future Work**](#_Toc79618421)

[Conclusion 46](#_Toc79618422)

[.Future Work 46](#_Toc79618423)

[References 47](#_Toc79618424)

**List of Figures**

Fig 2.1 Proposed Architecture 6

Fig 3.1 System Use Case 17

Fig 4.1 Class Diagram 19

Fig 4.2 Component Diagram 20

Fig 4.3 Deployment Diagram 21

Fig 4.: Activity Diagram (Sign Up) 22

Fig 4.5: Activity Diagram (Login) 23

Fig 4.6: Activity Diagram (Logout) 24

Fig 4.7: Activity Diagram (Feature Extraction) 25

Fig 4.8: Activity Diagram (Malware Detection) 26

Fig 4.9: Activity Diagram (Classification) 27

Fig 4.10: Sequence Diagram (Sign up) 28

Fig 4.11: Sequence Diagram (Login) 29

Fig 4.12 Sequence Diagram (Logout) 30

Fig 4.13: Sequence Diagram (Feature Extraction) 31

# 

# Chapter 1: Introduction.

# Brief

Malware detection is a method for judging the security of computer software, it is a key part of software safety research. Many malware detection technologies, such as malware code structure Therefore, the advancement and completeness of the detection method will determine the effectiveness of the malware detection project and control scheme. Analysis, function analysis, which are all based on the detection and the classiﬁcation. Feature based malware protection schemes such as antivirus software are still the most universal network security products in the current application. Malware is a malicious software or files that are harmful when we are going to execute that type of files or software that can damage the system or may create any effect on the system. Therefore, malware affects our daily life and our system need continuous security to defend against that attack. To provide protection against these attacks many researchers and teams are working on that. Lot of people uses the machine learning to automatic detection and classify the malware to secure their systems.

# Relevance to Course Modules

In our whole degree studied following subjects that have to relate in our project

* Database management system (DBMS)
* Software engineering-1
* Software engineering-2
* Web development.

A database typically requires a comprehensive database software known as database system. A database system serves as an interface between the database and its end users or program allowing users to retrieve, update and manage how the information is organized and optimized.

Software engineering is the application of principles used in the field of engineering, which usually deals with physical systems, to the design, development, testing, deployment and management of software systems.

Web development is the coding or programming that enables website functionality, per the owner's requirements. It mainly deals with the non-design aspect of building websites and Applications which includes coding and writing markup.

# Project Background

In this project we performed malware detection, we selected a dataset and perform malware detection on that dataset. This project is developed for malware detection. This system can identify the malicious files on the system and classify the type of malware. We configured Cuckoo Sandbox on Linux then execute files on cuckoo and associate API calls with parent process, extracted from ‘calls’ elements of cuckoo sandbox reports.

# Literature Review

[1](Gibert, Mateu, & Planes, 2010)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 focus 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.

[2](M. Siddiqui et al, 2009) Used Data Mining for detection of Worms. They used variable length instruction sequence. Their Primary data set consists of 2,775 Windows PE files, in which in which 1,444 were worms and 1,330 were benign. They performed detection of compilers, common packers and crypto before disassembly of files. Sequence reduction was performed and 97% of the sequences were removed. They used Decision Tree, Bagging and Random Forest models using. Random forest performed slightly better than the others.

[3]([Shabtai et al, 2009)](#_bookmark141)Provide a taxonomy for malware detection using machine learning algorithms by reporting some feature types and feature selection techniques used in the literature. They mainly focus on the feature selection techniques(Gain ratio, Fishers core, document frequency, and hierarchical features election) and classiﬁcation algorithms (Artiﬁcial Neural Networks, Bayesian Networks, Natïve Bayes, K-Nearest Neighbor, etc). In addition, they review how ensemble algorithms can be used to combine a set of classiﬁers

[4](J. Z. Kolter et al, 2004.) Used n-gram analysis and data mining approaches to detect malicious executables in the wild. The authors used a hexdump utility to convert each executable to hexadecimal code in an ASCII format and produced n-gram features by combining each four-byte sequence into a single term. Their primary dataset consisted of 1971 clean and 1651 malicious programs They used different classifiers including Instance-based Learner, TFIDF, Naive-Bayes, Support vector machines, Decision tree, boosted Naive-Bayes, SVMs and boosted decision tree. They used information gain to select valued features which are provided as input to all classifiers. The area under an ROC curve (AUC) is a more complete measure compared with the detection accuracy as they reported that the boosted decision trees outperform rest of the classifiers for both classification problems.

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

We have studied the lot of systems and literatures about the malware detection. Some of them are on machine learning and some are on the deep learning techniques. Deep learning technique is pretty good for this system. Machine learning is good but in the machine learning lot of effort is required for the feature engineering. In the deep learning method Auto-Encoders are mostly used for the malware detection system. We are going to use the Deep Learning to build this model. In this model we will develop the system that will detect the malware using deep learning and will classify the malware. It will identify the malware and will also identify it according to their family.

# Methodology and Software Life cycle

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**

Software applications, including malware, naturally evolve over time due to changes resulting from adding features, ﬁxing bugs, porting to new environments and platforms. These changes are expected to be introduced relatively infrequently. Additionally, successive versions of the software are expected to be highly similar to previous versions, with few exceptions such as when the code base under goes signiﬁcant refactoring and there are changes in the compilers or libraries linked to the software. Moreover, the similarity between previous and future versions is expected to degrade slowly over time. In consequence, the prediction quality decays over time as malware evolves and new variants and families appear. Thus, in order to build high quality models for malware detection and classiﬁcation, it is important to identify when the model shows signs of degradation and there by it fails to recognize new malware. Existing solutions aim at periodically retrain the model with the hope that it will automatically adapt to changes in malware overtime. The process of retraining the model can be done from scratch, partially and incrementally, were incremental retraining refers to the process of retraining a given model with new labeled malware samples and all previous training samples without for getting the knowledge obtained from prior datasets.

## **Deliverables and Development Requirements**

* Documentation
* User Panel
* Loading Data Panel
* Desktop Application
* Detection and Classification System

## **Proposed Architecture**

This system is to accurately detect new malware (unknown malware) binaries using a number of data mining techniques. The data set consists of malware files that were collected from two sources in addition to a set of benign files. This proposed architecture is that first to deploy the cuckoo sandbox, which perform dynamic analysis. An malware detection environment is then setup, which consists of all the necessary components to provide a suitable atmosphere to execute the samples and generate the required malware detection reports. The feature identification is then performed, and pre-processing is applied. Finally, the data set is used to train and test the classification accuracy of several deep learning models.

If already Detected

Feature Extraction

Classify

Data Set

Pre-Processing

Detect Malware

**Figure 2.1: Proposed Architecture**

## **Operating Environments**

* Windows 7 or higher
* Core i3 3rd generation
* 8Gb Ram

## **Assumptions and Dependencies**

No assumptions and dependencies.

# Chapter 3: Requirement Analysis

# Functional Requirements

**Table 3.1: Sign up**

|  |  |
| --- | --- |
| **Use Case ID:** | **UC-1.2.1** |
| **Use Case Name:** | **Sign up** |
| **Actors:** | **Primary Actor:** Administrator  **Secondary Actor:** None |
| **Description:** | User enters all required information in order to get signed up. |
| **Trigger:** | User is intended to Sign-up |
| **Preconditions:** | User must be on the Sign-up page. |
| **Post conditions:** | The user goes to the Login page after the completion of this requirement. |
| **Normal Flow:** | **User Actions:**   1. User enters First Name 2. User enters Last Name 3. User enters Email 4. User enters Password 5. User enters Confirm-Password 6. User clicks the Sign-up Button   **System Responses:**   1. System validates Email 2. System validates Password 3. System validates Confirm-Password 4. System enters data into the Database 5. Message is shown, “Success” or “Failure” |
| **Alternative Flows:** | **3a.** In step 3 of the normal flow, if the user enters Email  System validates the Email and accepts the format  User moves to the next field  Use Case resumes on step 4 of the normal flow  **3b.** In step 3 of the normal flow, if the User enters Email  System validates Email, rejects the format. Prompts the user to  re-enter the Email  Use Case stays on step 3 of normal flow |
|  | **4a.** In step 4 of the normal flow, if the user enters Password  System validates Password and accepts the format  User moves to the next field  Use Case resumes on step 5 of the normal flow  **4b.** In step 4 of the normal flow, if the User enters Password  System validates Password, rejects the format. Prompts the  user to re-enter the Password  Use Case stays on step 4 of normal flow |
|  | **5a.** In step 5 of the normal flow, if the user enters Confirm-Password  System validates Confirm-Password and matches it with  previously entered Password  User clicks the Sign-up Button  Use Case resumes on step 6 of the normal flow  **5b.** In step 5 of the normal flow, if the user enters Confirm-Password  System validates Confirm-Password and finds it different from  the previously entered Password  Use Case stays on step 5 of the normal flow |
|  | **10a.** In step 10 of the normal flow, if the System stores data  in Database  System verifies the query and entities in the Database  Use Case resumes on step 11 of the normal flow  **10b.** In step 10 of the normal flow, if the System stores data  in Database  System finds ambiguity and is not able to store the data  Use Case stays on step 10 of normal flow |
|  | **11a.** In step 11 of the normal flow, if the System shows Success  Message  User is Signed up  And heads over to the Login Page  **11b.** In step 11 of the normal flow, System shows failure  message it will stay on the Sign-up page again. |
| **Exceptions:** | **3a.** In step 3 of the normal flow, if a user enters an invalid Email  Sign-up is disapproved  Message to user to re-enter a valid Email  Customer enters correct Email  Use Case resumes on step 4 of normal flow |
|  | **4a.** In step 4 of the normal flow, if a user enters an invalid Password  Sign-up is disapproved  Message to user to re-enter a valid Password  User enters correct Password  Use Case resumes on step 5 of normal flow |
|  | **5a.** In step 5 of the normal flow, if a user enters an invalid Password  in the Confirm-Password field  Sign-up is disapproved  Message to user to re-enter a valid Password  User enters correct Password in the Confirm Password field  Use Case resumes on step 6 of normal flow |
| **Includes:** | Login |
| **Special Requirements:** | There are no special requirements associated with this use case. |
| **Assumptions:** | User knows the personal information in order to Sign up |

**Table 3.2: Login**

|  |  |
| --- | --- |
| **Use Case ID:** | **UC-1.2.2** |
| **Use Case Name:** | **Login** |
| **Actors:** | **Primary Actor:** Administrator  **Secondary Actor:** None |
| **Description:** | User enters all required information in order to get Logged up. |
| **Trigger:** | User is intended to Login |
| **Preconditions:** | User must be on the Login page and already have signed up |
| **Post conditions:** | The user goes to the main page of application after the completion of this requirement. |
| **Normal Flow:** | **User Actions:**   1. User enters Email 2. User enters Password 3. User clicks the Login Button   **System Responses:**   1. System validates Email 2. System validates Password 3. System verifies data from the Database 4. Message is shown, “Success” or “Failure” |
| **Alternative Flows:** | **1a.** In step 1 of the normal flow, if the user enters Email  System validates the Email and accepts the format  User moves to the next field  Use Case resumes on step 2 of the normal flow  **1b.** In step 1 of the normal flow, if the User enters Email  System validates Email, rejects the format. Prompts the user to  re-enter the Email  Use Case stays on step 1 of normal flow |
|  | **2a.** In step 2 of the normal flow, if the user enters Password  System validates Password and accepts the format  User moves to the next field  Use Case resumes on step 3 of the normal flow  **2b.** In step 2 of the normal flow, if the User enters Password  System validates Password, rejects the format. Prompts the  user to re-enter the Password  Use Case stays on step 2 of normal flow |
|  | **6a.** In step 6 of the normal flow, if the System verifies data  from Database  System verifies the query and entities in the Database  Use Case resumes on step 7 of the normal flow  **6b.**  In step 6 of the normal flow, if the System verifies data  from Database  System finds ambiguity and is not able to match the data  Use Case stays on step 6 of normal flow |
|  | **7a.** In step 7 of the normal flow, if the System shows Success  Message  User is Logged in  And heads over to the Main Page  **7b.** In step 7 of the normal flow, System shows failure  message it will stay on the Login-up page again. |
| **Exceptions:** | **1a.** In step 1 of the normal flow, if a user enters an invalid Email  Login is disapproved  Message to user to re-enter a valid Email  Customer enters correct Email  Use Case resumes on step 2 of normal flow |
|  | **2a.** In step 2 of the normal flow, if a user enters an invalid Password  Login is disapproved  Message to user to re-enter a valid Password  User enters a valid Password  Use Case resumes on step 3 of normal flow |
| **Includes:** |  |
| **Special Requirements:** | The user must be registered or signed up |
| **Assumptions:** | User knows the Login credentials |
| **Notes and Issues:** | User has to put email in appropriate format. The length of Password must be greater than 4. Password has to be comprised of Lower case letters, Upper case letters and numeric |

**Table 3.3: Logout**

|  |  |
| --- | --- |
| **Use Case ID:** | **UC-1.2.3** |
| **Use Case Name:** | **Logout** |
| **Actors:** | **Primary Actor:** Administrator  **Secondary Actor:** None |
| **Description:** | User clicks the logout button |
| **Trigger:** | User is intended to Logout |
| **Preconditions:** | User must be already Logged in |
| **Post conditions:** | Successful logging out leads to desktop application’s Login page and failed logging out remains the user onto the main page. |
| **Normal Flow:** | **User Actions:**   1. User clicks the Logout button   **System Responses:**   1. Message is shown, “Success” or “Failure” |
|  | **2a.** In step 2 of the normal flow, if the System shows Success Message User is Logged out and returns back to the Login page of application  **2b.** In step 2 of the normal flow, System shows failure message, it will stay on the main page again. |
| **Includes:** |  |
| **Special Requirements:** | There are no special requirements associated with this use case. |
| **Assumptions:** | User is aware of Logging out through the Logout button |
| **Notes and Issues:** | None |

**Table 3.4: Preprocessing**

|  |  |
| --- | --- |
| **Use Case ID:** | **UC-1.2.4** |
| **Use Case Name:** | **Preprocessing** |
| **Actors:** | **Primary Actor:** User  **Secondary Actor:** None |
| **Description:** | User uploads a raw file into cuckoo sandbox |
| **Trigger:** | User is intended to load the binary dataset file |
| **Preconditions:** | User must upload the appropriate file into the required field |
| **Post conditions:** | The dataset must have loaded |
| **Normal Flow:** | **User Actions:**   1. User clicks the browse button to find the file 2. User clicks the upload button to upload the file into   Cuckoo sandbox  **System Responses:**   1. System starts malware analysis 2. Message is shown, “completed” or “Error” |
|  | **3a.** After analysis is completed the cuckoo will generate JSON report  **3b.** Parse the JSON report using cuckoo report parser and generate a useful dataset. |
| **Exceptions:** | **2a.** In step 2 of the normal flow, if a user uploads an invalid format  Loading of dataset is disapproved  Message to user to re-upload a valid file  User uploads the correct format |
| **Includes:** | None |
| **Special Requirements:** | There are no special requirements associated with this use case. |
| **Assumptions:** | User is aware of uploading the dataset file through the upload button |
| **Notes and Issues:** | User must have to upload the appropriate dataset file format |

**Table 3.5: Feature Extraction**

|  |  |  |
| --- | --- | --- |
| **Use Case ID:** | **UC-1.2.5** | |
| **Use Case Name:** | **Feature Extraction** | |
| **Actors:** | **Primary Actor:** User  **Secondary Actor:** None | |
| **Description:** | System Action:   1. Get Profiling detail 2. Perform Feature Generation 3. Success | |
| **Trigger:** | Get generated data from preprocessing | |
| **Preconditions:** | The dataset was generated using JSON report | |
| **Post conditions:** | The dataset must have loaded | |
| **Normal Flow:** | For performing this process there are following steps   1. Get Profiling detail 2. Perform Feature Extraction 3. Success 4. Feature are generated. |
| **Alternative Flow:** | Success full then move to next step else Move again to feature generation | |
| **Exceptions:** | During the performing action some error can be take place   1. Profiling detail not collected 2. Fail to Perform feature extraction | |
| **Includes:** | None | |
| **Special Requirements:** | There are no special requirements associated with this use case. | |
| **Assumptions:** | User is aware of uploading the dataset file through the upload button | |
| **Notes and Issues:** | User must have to upload the appropriate dataset file format | |

**Table 3.1: Malware Detection**

|  |  |  |
| --- | --- | --- |
| **Use Case ID:** | **UC-1.2.6** | |
| **Use Case Name:** | **Malware Detection** | |
| **Actors:** | **Primary Actor:** User  **Secondary Actor:** None | |
| **Description:** | 1. For Detection of Malware 2. System Action 3. Get Feature generated report database 4. detect the malware from report 5. Success | |
| **Trigger:** | Collect feature generation data from database | |
| **Preconditions:** | Non detected | |
| **Post conditions:** | Detected | |
| **Normal Flow:** | System Action   1. Get Feature generated report database 2. Detect the malware from report 3. Success |
| **Alternative Flow:** | Success full then move to next step else Move again to detect malware | |
| **Exceptions:** | Data collection fail from Feature generation report  Detection of malware fail | |
| **Includes:** | None | |
| **Special Requirements:** | There are no special requirements associated with this use case. | |
| **Assumptions:** | User is aware of uploading the dataset file through the upload button | |
| **Notes and Issues:** | User must have to upload the appropriate dataset file format | |

**Table 3.7: Malware Classification**

|  |  |  |
| --- | --- | --- |
| **Use Case ID:** | **UC-1.2.7** | |
| **Use Case Name:** | **Malware Classification** | |
| **Actors:** | **Primary Actor:** User  **Secondary Actor:** None | |
| **Description:** | 1. For Classification of Malware 2. System Action 3. Get data from detection 4. Classify the detected data in families 5. success | |
| **Trigger:** | Get detected malware file | |
| **Preconditions:** | Non classified dataset | |
| **Post conditions:** | Classified | |
| **Normal Flow:** | 1. Get data from detection 2. Classify the detected data in families 3. Classified data set 4. Complete detail of malware and their classes |
| **Alternative Flow:** | Successfully classified else Move again to classify malware | |
| **Exceptions:** | 1. Data not collected from detection 2. Failed to Classify | |
| **Includes:** | None | |
| **Special Requirements:** | Malware must be detected | |
| **Assumptions:** | User is aware of uploading the dataset file through the upload button | |
| **Notes and Issues:** | User must have to upload the appropriate dataset file format | |

## **Non-Functional Requirements**

### **Usability**

Our system basically defines a malware detection system which is very user friendly and any user can understand its functionality in just 4 to 5 minutes.

### **Reliability**

Our system is reliable and accurately detect the malware with 97% accuracy; however, it may need up gradation in future.

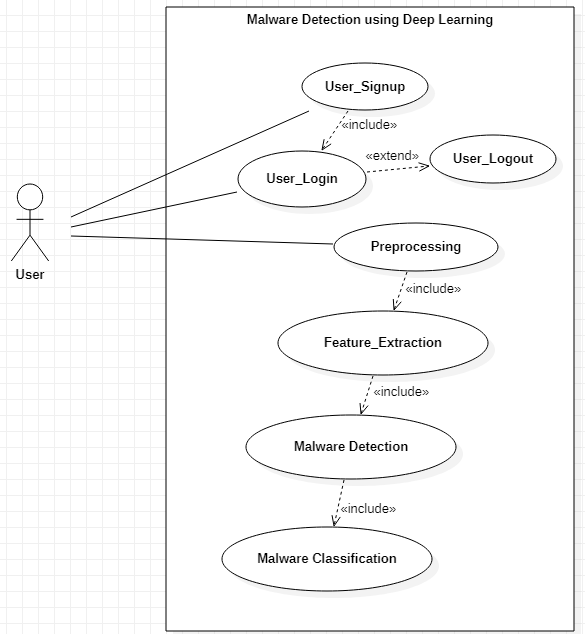
### **Integrity**

We use authentication and validation to the email and password of user account to make them secure. It can be tested, rendering error free results. It lacks security vulnerabilities which is the most important aspect of a system. It is easy to edit and upgrade without introducing new errors.

### **Performance**

Our system performs well without any lag. It is highly responsive, has an efficient throughput, execution time is less than 5 minutes.

# Use Case Model

****

**Figure 3.1: System use case**

# Actor Description

For the detection of malware Actor first perform report generate then data collection, Profiling, Feature generate, feature database, Detection, Classification and at the end perform Evaluation and then give output as classified malware file dat**a**

# Use Case Description

1. **Preprocessing**

* Upload dataset to cuckoo sandbox
* Generate a JSON report
* Generate dataset using parse JSON report and cuckoo report parser

1. **Feature Extraction**

* Feature extraction Report
* Save report
* Successfully save or fail

1. **Detection**

* Get data from feature extraction
* Features extracted properly
* Success or fail

1. **Classification**

* System Action
* Get data from detection
* Classified the detected data in families
* Success or fail

# Chapter 4: Design and Architecture

## **UML Structural Diagrams**

### **Class Diagram**

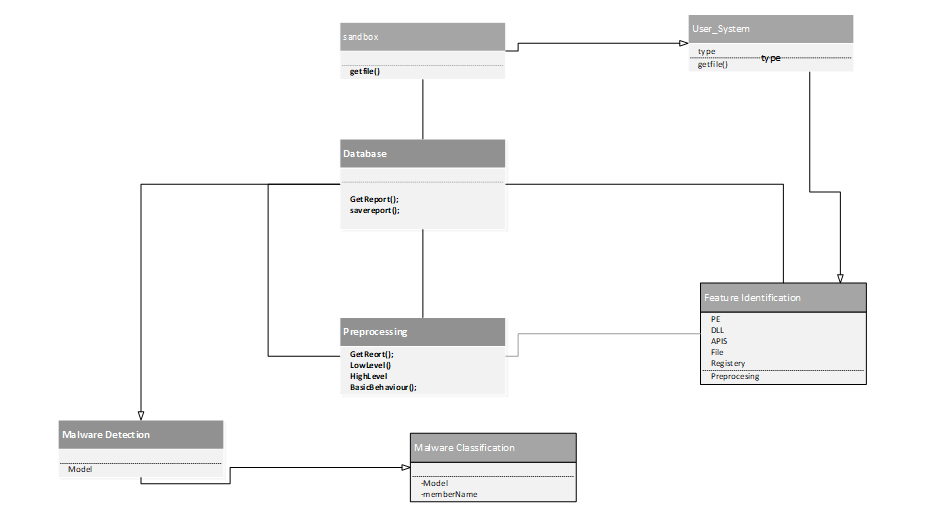
****

Figure 4. 2: Class Diagram

### **Component Diagram**

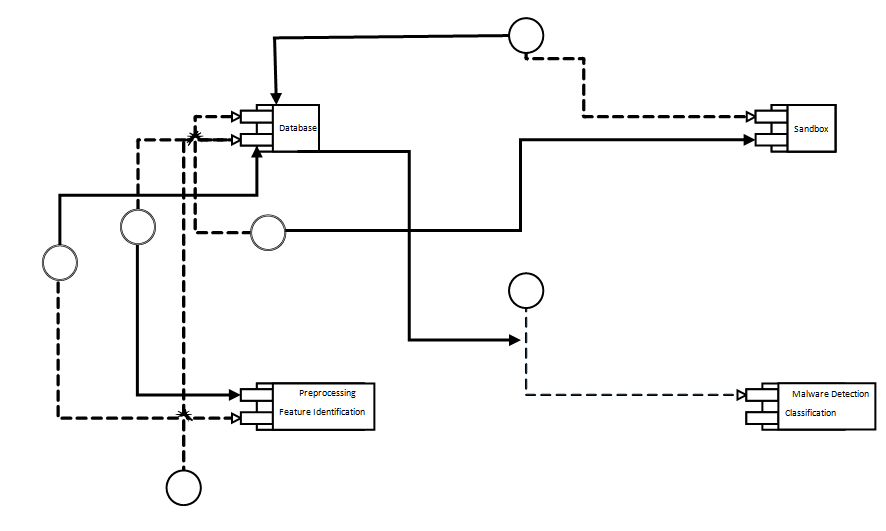


Figure 4. 3: Component Diagram

### **Deployment Diagram**

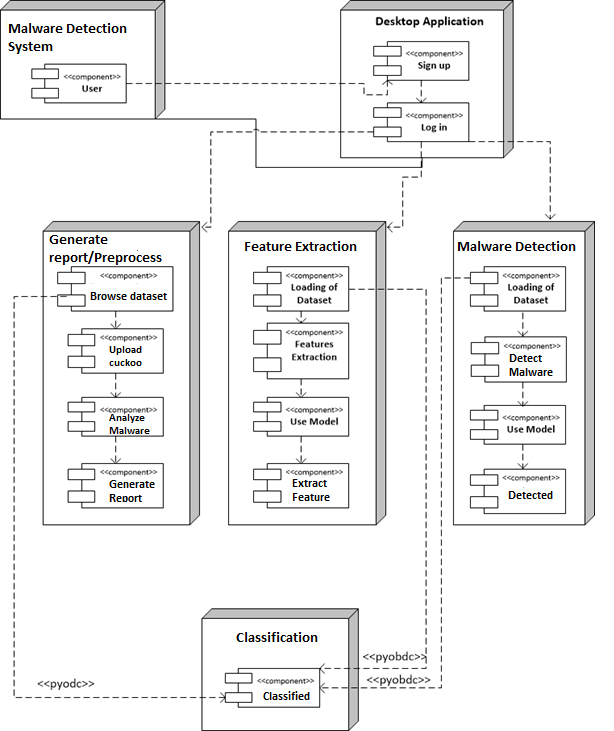


Figure 4. 4: Deployment Diagram

## **UML Behavioral Diagrams**

### **Activity Diagrams**

#### **Sign Up**

Diagram

Description automatically generated

Figure 4. 5: Activity Diagram (Sign Up)

#### **Login**

Diagram, schematic

Description automatically generated

Figure 4. 6: Activity Diagram (Login)

#### **Logout**

Diagram

Description automatically generated

Figure 4. 7: Activity Diagram (Logout)

#### **Feature Extraction**

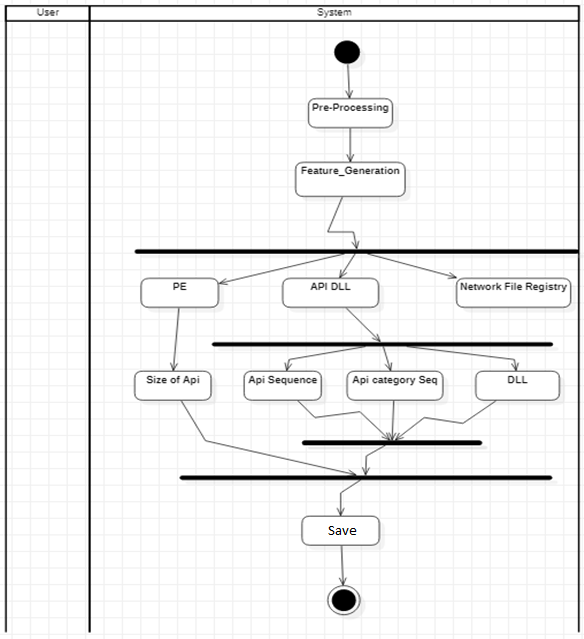
****

Figure 4. 7: Activity Diagram (Feature Extraction)

#### **Malware Detection**

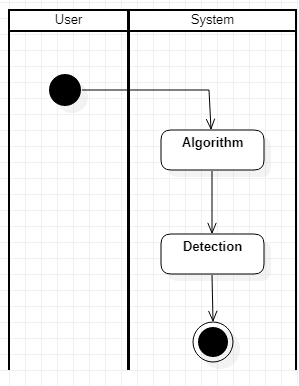


Figure 4. 8: Activity Diagram (Malware Detection)

#### **Classification**

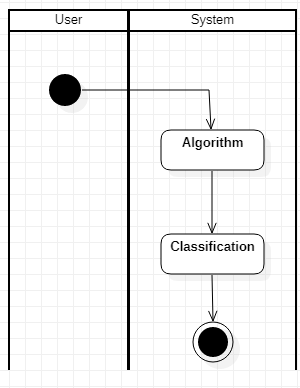
****

Figure 4.9: Activity Diagram (Classification)

## **UML Interaction Diagrams**

### **Sequence Diagrams**

#### **Sign Up**

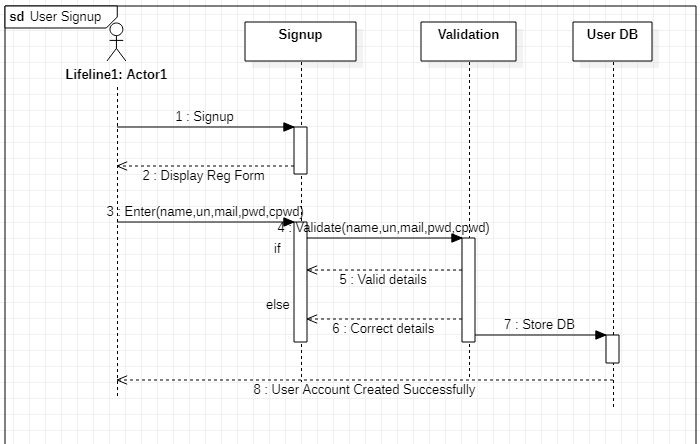


Figure 4.10: Sequence Diagram (Sign Up)

#### **Login**

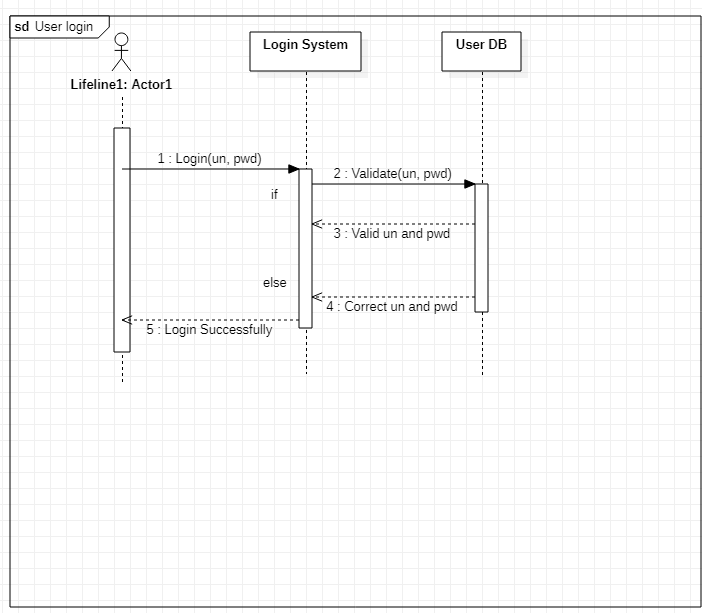


Figure 4.11: Sequence Diagram (Login)

#### **Log Out**

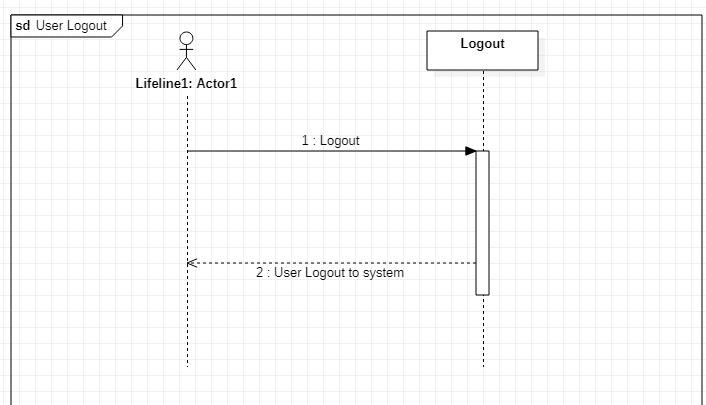


Figure 4.12: Sequence Diagram (Log Out)

#### **Malware Detection System**

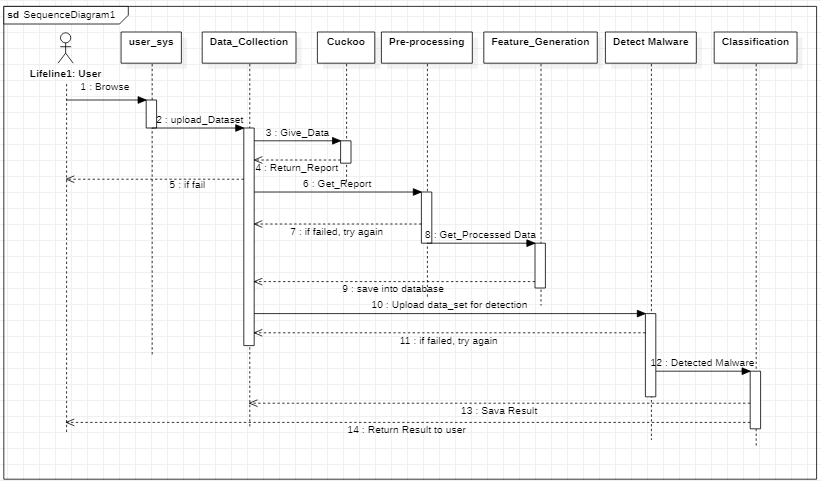
****

Figure 4.13: Sequence Diagram (Malware Detection System)

# Chapter 5: Implementation

## **Tools and Technologies**

### **PyCharm**

PyCharm is an integrated development environment used in computer programming, specifically for the Python language. It is developed by the Czech company JetBrains. PyCharm provides smart code completion, code inspections, on-the-fly error highlighting and quick-fixes, along with automated code refactoring’s and rich navigation capabilities.

### **Tkinter**

Tkinter commonly comes bundled with Python, using Tk and is Python's standard GUI framework. It is famous for its simplicity and graphical user interface. It is open-source and available under the Python License

### **Wamp Server**

It Stands for "Windows, Apache, MySQL, and PHP."

We used Wamp server to load datasets into database and also to store login information of Users and Admin.

It runs as a process in Apache and communicates with the MySQL database.

### **MS Word**

Microsoft Word is a graphical word processing program that we can type with. Its purpose is to allow us to type and save documents.

We used MS WORD to type the Documentation of our project.

It has advanced features which allow us to format and edit our files and documents in the best possible way.

### **MS Power Point**

PowerPoint is used to create presentations. The presentations are comprised of slides, which may contain text, images, and other media.

The purpose of PowerPoint is to act as a visual aid as we go along presenting their option, ideas, etc. We used PowerPoint to present our project idea and documentation in charts and visual form while the evaluation of project.

### **Python 3.9**

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.

### **Matplotlib**

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+.

### **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.

### **Pandas**

Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under The three-clause BSD license

### **MySQL Connector**

We used MySQL Connector to enable python program to access MySQL database. Because python needs a driver to access MySQL database. This is necessary because each make of database server has its own specific protocol for transporting requests to, and results from, the server to application programs.

### **NumPy**

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

### **Star UML**

Star UML is a UML tool by MKLab. The software was licensed under a modified version of GNU GPL until 2014, when a rewritten version 2.0.0 was released for beta testing under a proprietary license. After being abandoned for some time, the project had a revival to move from Delphi to Java/Eclipse and then stopped again.

### **Cuckoo Sandbox**

A Cuckoo Sandbox is a tool that is used to launch malware in a secure and isolated environment, the idea is the sandbox fools the malware into thinking it has infected a genuine host. The sandbox will then record the activity of the malware and then generate a report on what the malware has attempted to do while in this secure environment.

## **User Interface**

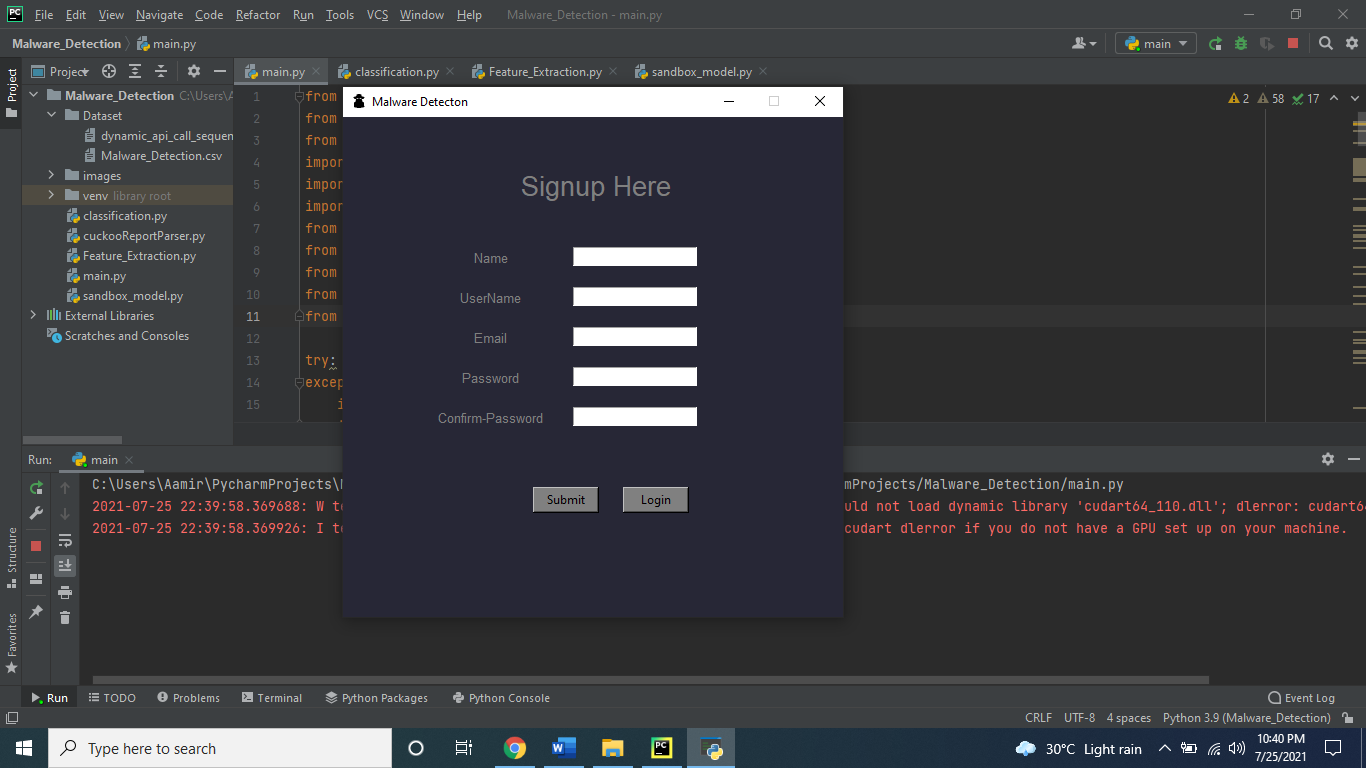
****

Figure 5. 1: User Signup

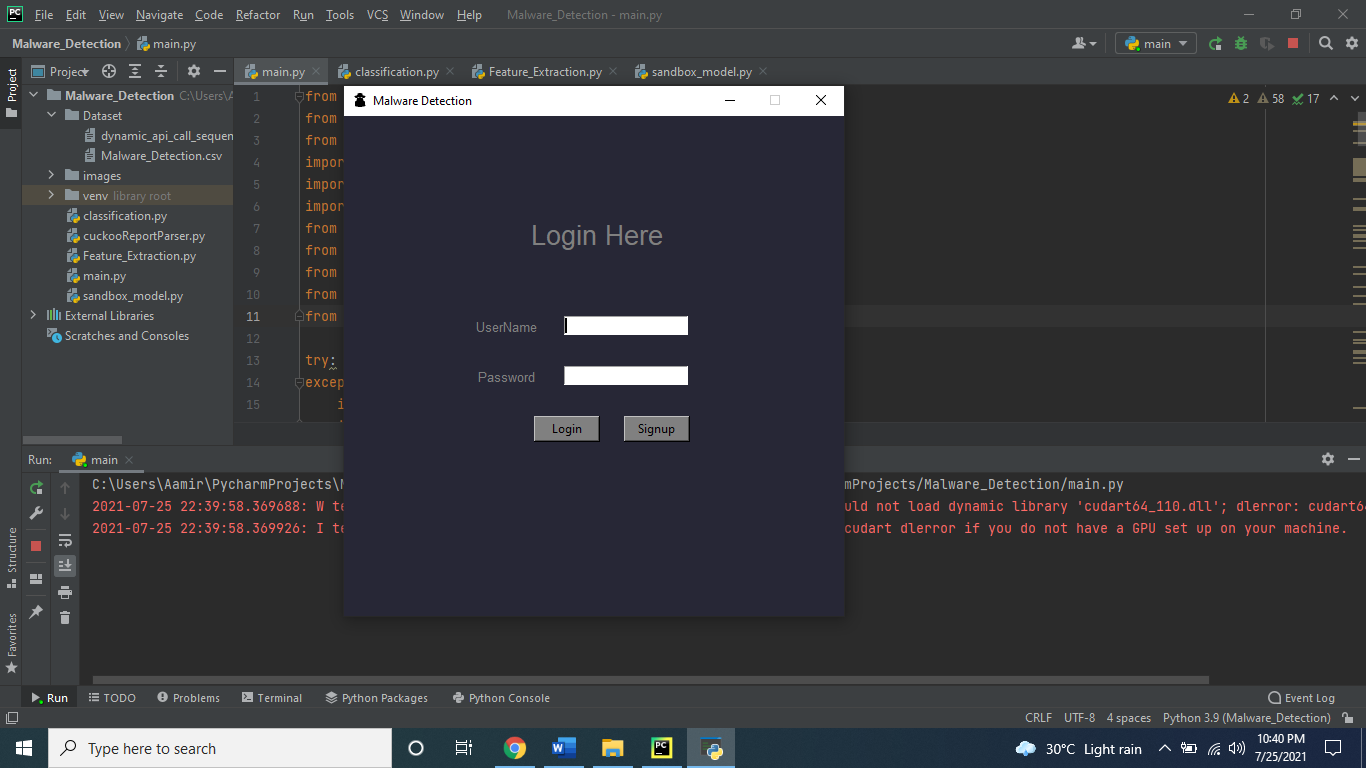
****

Figure 5. 2: User Login

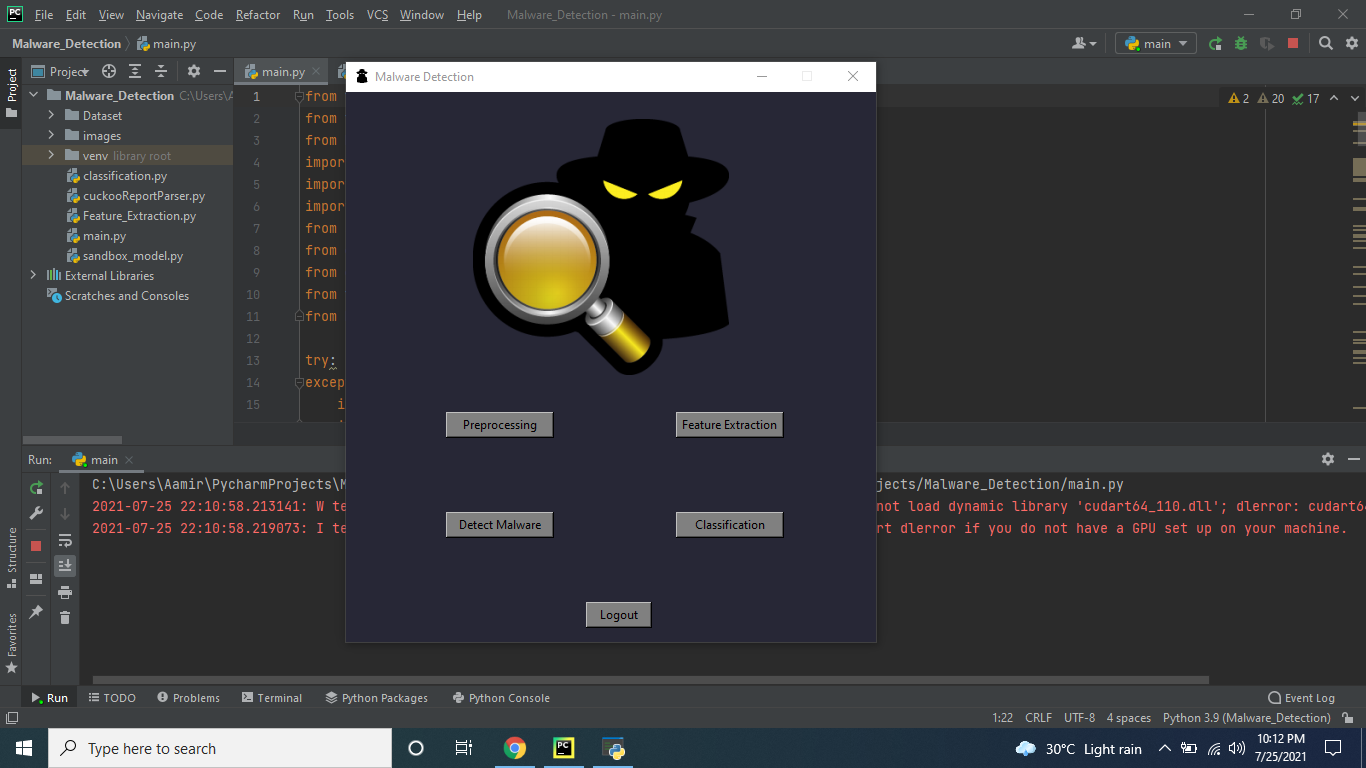
****

Figure 5. 3: User Interface

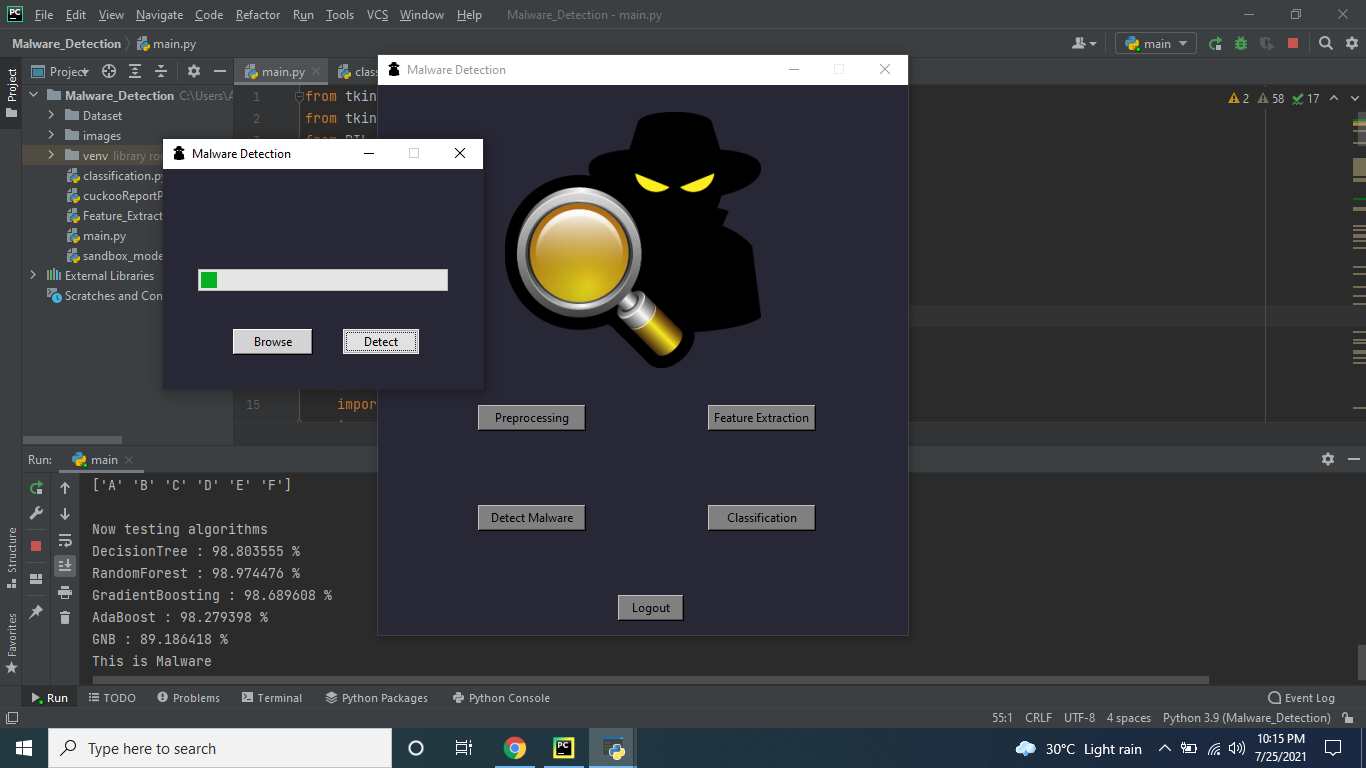
****

Figure 5. 4: Malware Detection

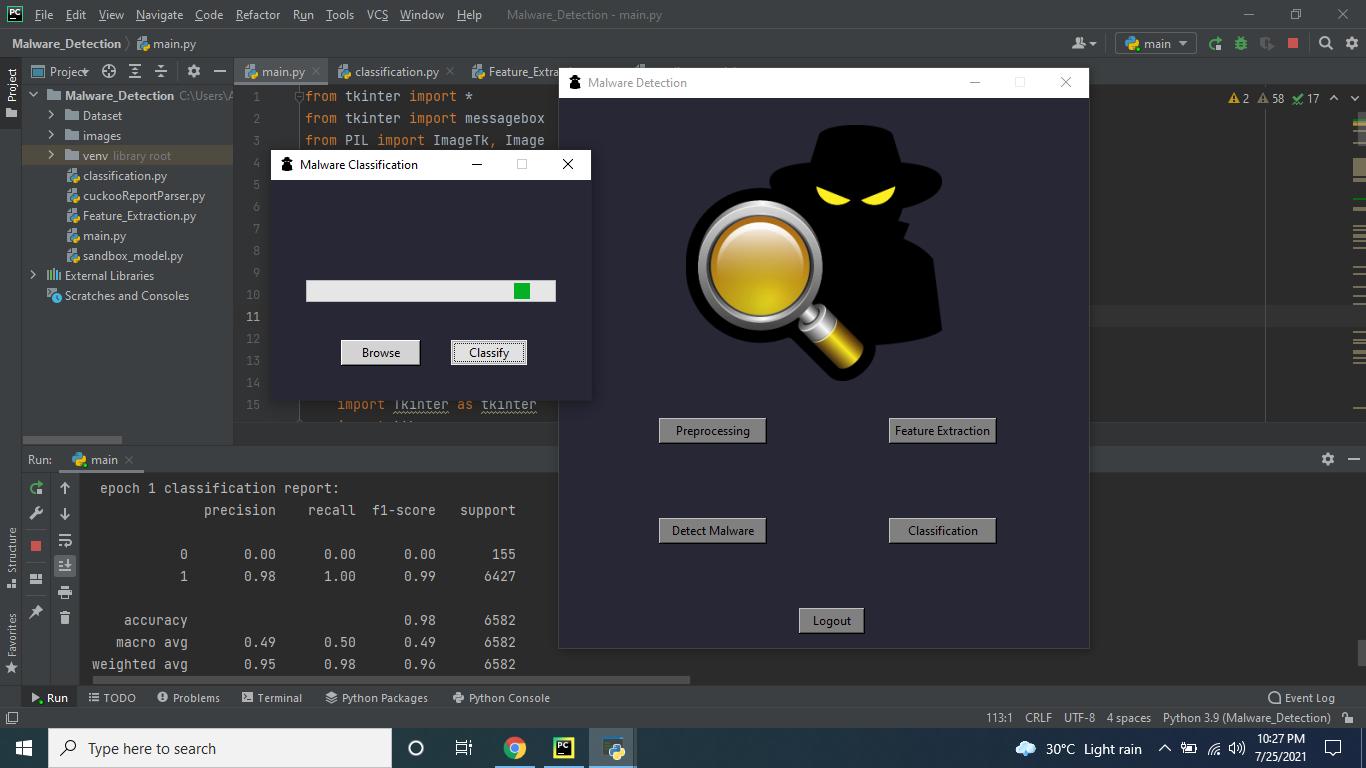
****

Figure 5. 5: Malware Classification

# Chapter 6: Testing and Evaluation

**Table 6.1: Test User Signup**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Project Name: Malware Detection** | | **Test Case: 1** | | | | | | | | |
| **Test Case ID:** T\_01 | | | |  | | | |
| **Test Priority (Low/Medium/High):** | | | | High | | | |
| **Module Name:** | | | | User Signup | | | |
| **Input:** | | | | Data for variable Name, username, password and conform password. | | | |
| **Test Title:** | | | | Entering all the Values that must be correct. | | | |
| **Description:** | | | | User press the Signup button on splash screen, a new window open with following requirement, Name, username, password and conform password, user enter values against them and press Signup button. | | | |
| **Pre-Condition:** | | | | User should enter all the required detail correctly. | | | |
| **Dependencies:** | | | | None | | | |
| **Steps** | **Test Steps** | **Test Data** | **Expected Result** | | **Actual Result** | **Status (Pass/Fail)** | **Notes** |
| 1 | Name | Amir |  | |  |  |  |
| 2 | User Name | Amir75 |  | |  |  |  |
| 3 | Password | 0022 |  | |  |  |  |
| 4 | Conform Password | 0022 |  | |  |  |  |
| 5 | Signup Button Clicked |  | All data entered by user will store in database named user dB and user can login through entered username and password. | | Show error message Password and conform password should be same | Fail |  |
|  |  |  |  | |  |  |  |

**Table 6.2: Test User Login**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Project Name: Malware Detection** | | **Test Case: 2** | | | | | | | | |
| **Test Case ID:** | | | | **T\_02** | | | |
| **Test Priority (Low/Medium/High):** | | | | High | | | |
| **Module Name:** | | | | User Login | | | |
| **Input:** | | | | Values against username and password | | | |
| **Test Title:** be correct. | | | | Entering all the Values that must | | | |
| **Description:** | | | | Test the login page. | | | |
| **Pre-Condition:** | | | | User should Have an account and user name and password should correct. | | | |
| **Dependencies:** | | | | User must be registered first. | | | |
| **Steps** | **Test Steps** | **Test Data** | **Expected Result** | | **Actual Result** | **Status (Pass/Fail)** | **Notes** |
| 1 | User Name | Armaan65 |  | |  |  |  |
| 2 | Password | 0011 |  | |  |  |  |
| 3 | Login Button Clicked |  | If username and password entered correctly, System granted access to user | | Navigate to main screen | Pass |  |

**Table 6.3: Test User Logout**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Project Name: Malware Detection** | | **Test Case: 3** | | | | | | | | |
| **Test Case ID:** | | | | T\_03 | | | |
| **Test Priority (Low/Medium/High):** | | | | High | | | |
| **Module Name:** | | | | User Logout | | | |
| **Input:** | | | | Press logout button | | | |
| **Test Title:** | | | | Logout user from system | | | |
| **Description:** | | | | User press logout button. | | | |
| **Pre-Condition:** | | | | User login to the system | | | |
| **Dependencies:** | | | | User must be login. | | | |
| **Steps** | **Test Steps** | **Test Data** | **Expected Result** | | **Actual Result** | **Status (Pass/Fail)** | **Notes** |
| 1 | Logout Button Clicked |  | System logout the user. | | System returns the user to login screen | Pass |  |

**Table 6.4: Test Preprocessing**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Project Name: Malware Detection** | | | | | |
| **Test Case: 4** | | | | | |
| Test Case ID | | | T\_04 | | |
| Test priority (low, medium, high) | | | High | | |
| Module Name | | | Preprocessing | | |
| Description | | | First upload raw dataset to cuckoo sandbox for malware analysis and generate a JSON report. Parse JSON report using report parser and generate a useful dataset. | | |
| Test Title | | | Preprocessing | | |
| Precondition | | | Unclassified | | |
| Despondency | | | Running cuckoo  Mongo dB should start  Virtual machine starts or restore mode  Should complete analysis | | |
| S. | Test steps | Expected result | Actual result | Status fail or pass | Note |
| 1 | * Open browser * Open web * In cuckoo module * Browse file * Upload file | Analyze malware and generate a JSON report | Generate a useful dataset | Pass |  |

**Table 6.5 Test Feature Extraction**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Project Name: Malware Detection** | | | | | |
| **Test Case: 5** | | | | | |
| Test Case ID | | | T\_05 | | |
| Test priority (low, medium, high) | | | High | | |
| Module Name | | | Feature Extraction | | |
| Description | | |  | | |
| Test Title | | | Feature Extraction | | |
| Precondition | | | Unclassified | | |
| Despondency | | | Running cuckoo  Mongo dB should start  Virtual machine starts or restore mode  Should complete analysis | | |
| S. | Test steps | Expected result | Actual result | Status fail or pass | Note |
| 1 | * Open browser * Open web * In cuckoo module * Browse file * Upload file | Predict Malware class | It shows message this a trojan horse | Pass |  |

**Table 6.6: Test Malware Detection**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Project Name: Malware Detection** | | | | | | |
| **Test Case: 6** | | | | | | |
| Test Case ID | | | | T\_06 | | |
| Test priority (low, medium, high) | | | | High | | |
| Module Name | | | | Malware Detection | | |
| Description | | | | Admin Run the malware detection | | |
| Test Title | | | | Malware Detection | | |
| Precondition | | | | Result on desktop interface | | |
| Despondency | | | | Internet | | |
| s. | Test steps | Expected result | Actual result | | Status fail or pass | Note |
| 1 | * Open desktop app * Browse file * Upload file | It should show the Report of file | It shows a short information of file. | | Pass |  |

**Table 6.7: Classification**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Project Name: Malware Detection** | | | | | |
| **Test Case: 7** | | | | | |
| Test Case ID | | | T\_07 | | |
| Test priority (low, medium, high) | | | High | | |
| Module Name | | | Classification | | |
| Description | | | After uploading file and completion of analysis it will show of malware classification. | | |
| Test Title | | | Classification | | |
| Precondition | | | Unclassified | | |
| Despondency | | | Running cuckoo  Mongo dB should start  Virtual machine starts or restore mode  Should complete analysis | | |
| S. | Test steps | Expected result | Actual result | Status fail or pass | Note |
| 1 | * Open browser * Open web * In cuckoo module * Browse file * Upload file | Predict Malware class | It shows message this a trojan horse | Pass |  |

# 

# Future Work

The system can be integrated by training the model that can remove the malware from the files or protect files against malware and can be upgraded to provide the security to datasets. In future the system can be integrated further according to our needs.

# Conclusion

In the developed model we can analysis the malware in given dataset by using cuckoo droid. We can perform static and dynamic analysis on the system. We can further detect the malware from the report generated and dataset using developed model. When detection process is completed we can classify the detected malware according their family.

# References

* 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)
* 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
* Ali Behforooz& Frederick J.Hudson, (1996), Software Engineering Fundamentals, Oxford University Press. Chapter 8, pp255-235. (book reference example)
* Page Author, Page Title, http://www.bt.com/bttj/archive.htm, Last date accessed. (web site)