VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

AI in Cyber Security

Malware Detection and Classification

Project Report

By Group 2.8:

Madiraju Chaitanya Raju

Akshay Kumar Pandey

Rohit Ritesh Maini

Kunawar Khurana

Under the guidance of: P. Manoj Sir.

The composition of our project team for the ‘Malware Detection and Classification project’ was carefully curated to ensure that we have a well-rounded group of individuals with the right skills and expertise to tackle the complexities of this AI and cybersecurity project.

- Rohit Ritesh Maini was appointed as the project manager due to his experience in leading complex software development projects. His strong organizational skills and ability to coordinate team efforts will be essential in keeping our project on track and within budget.

- Madiraju Chaitanya Raju is an expert in software engineering with a focus on real-time data processing and logging. He will oversee the design and implementation of the logging system, which is vital for capturing and storing data related to detected malware.

- Akshay Kumar Pandey was selected for his extensive background in cybersecurity and malware analysis. Their deep knowledge of malware behavior and classification methods makes them the ideal candidate to lead the development of the core detection and classification algorithms.

- Kunwar Khurana specializes in user interface and user experience (UI/UX) design. He will ensure that the application's interface is intuitive and user-friendly, allowing security professionals to interact seamlessly with the system.

In summary, the team members were chosen based on their specific qualifications and experience that align with the project's objectives. Each team member's role was assigned to leverage their strengths and expertise in different aspects of the 'Malware Detection and Classification Logger Application,' ensuring a comprehensive and effective approach to tackling the problem."

Project Details:

Introduction:

Our objective is to deliver a log-based detection system to firewall development companies and testers, enabling them to receive hourly reports on the status and classification of detected malware within their designated network.

Problem Statement:

Our problem statement was chosen to address the cybersecurity needs of organizations. With the escalating complexity of cyber threats, there's a growing demand for innovative solutions. We selected this problem statement to enhance firewall testing by offering real-time malware monitoring and classification during the testing process to check if the firewall was working as intended. We recognized the vital importance of instant threat identification, allowing companies to swiftly respond to potential breaches and vulnerabilities.

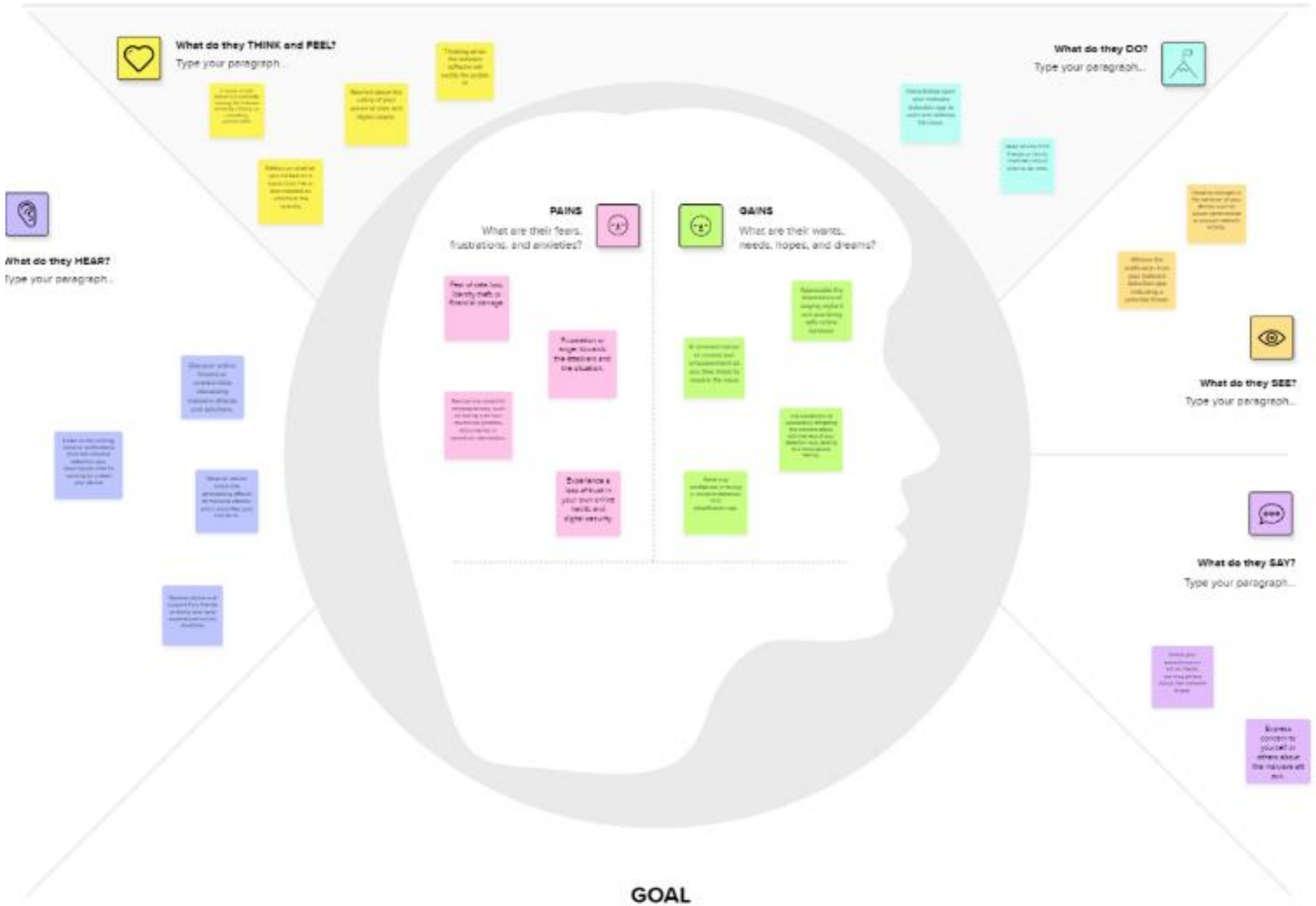
Our team's expertise in cybersecurity, data analysis, and software development, coupled with our desire for cybersecurity innovation, uniquely positions us for this endeavor.

Ideation:

Here’s some of the ideas that our group had before the development of the project:

- Pattern Creator
- Detailed Logs and Real-Time AI Communication
- Malware Graph Database
- Collaboration with Other Apps
- IoT Premium Feature
- Adblocker with Threat Notification
- User Education Integration
- Privacy Beacon with Data Usage Notification
- Incident Response Integration
- Feedback Mechanism
- Data Usage Limits and Analysis

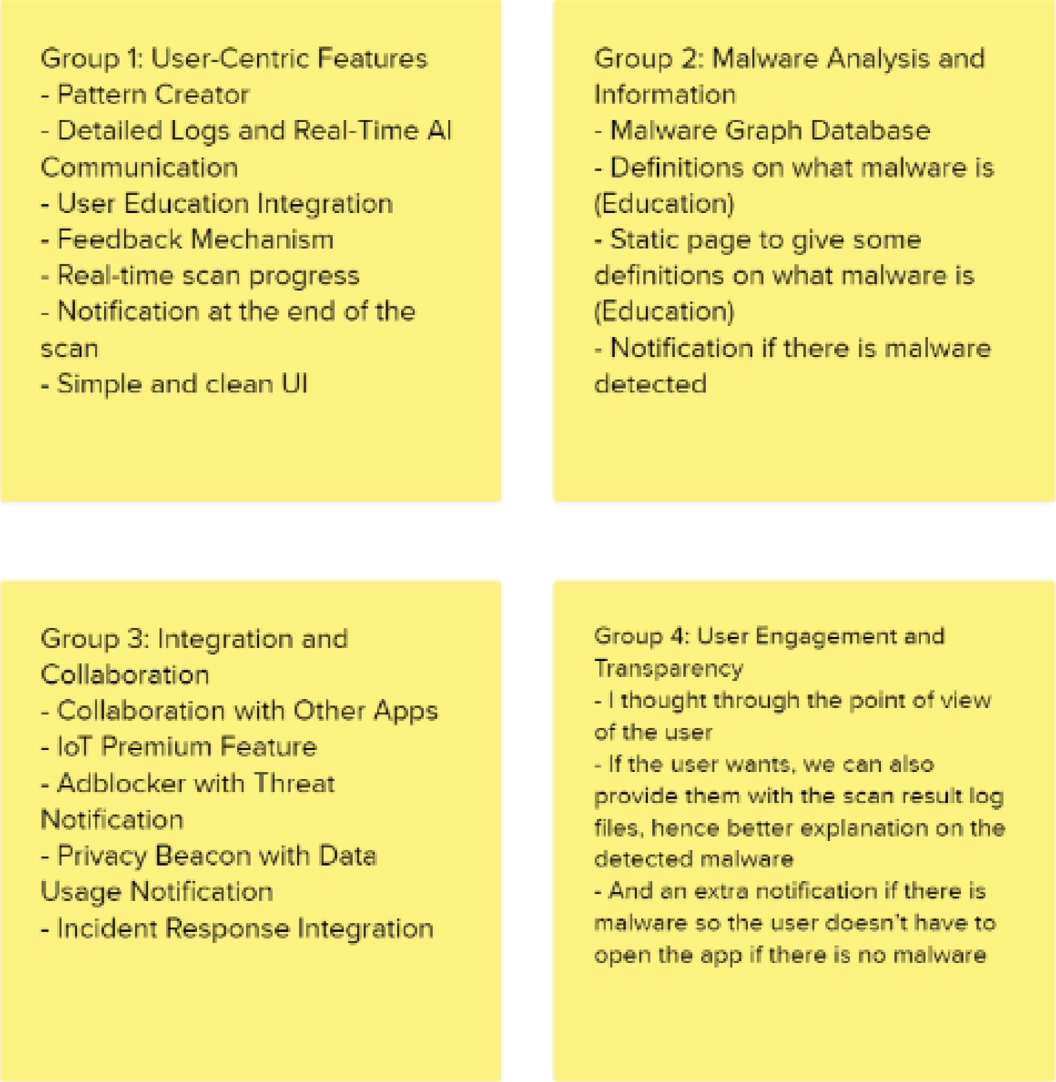
Our Project from the user’s point of view:


What do they THINK and FEEL?
Type your paragraph...
What do they DO?
Type your paragraph...
What do they HEAR?
Type your paragraph...
PAINS
What are their fears, frustrations, and anxieties?
Fear of data loss, identity theft or financial damage
Frustration or anger towards the attackers and the situation.
Experience a loss of trust in your own online habits and digital security.
GAINS
What are their wants, needs, hopes, and dreams?
What do they SEE?
Type your paragraph...
What do they SAY?
Type your paragraph...
Express concern to yourself or others about the malware att ack

GOAL

An AI automated Malware detection and classification

In the ideation phase, we prioritize our ideas to focus on what’s necessary.
We first grouped the ideas we had:



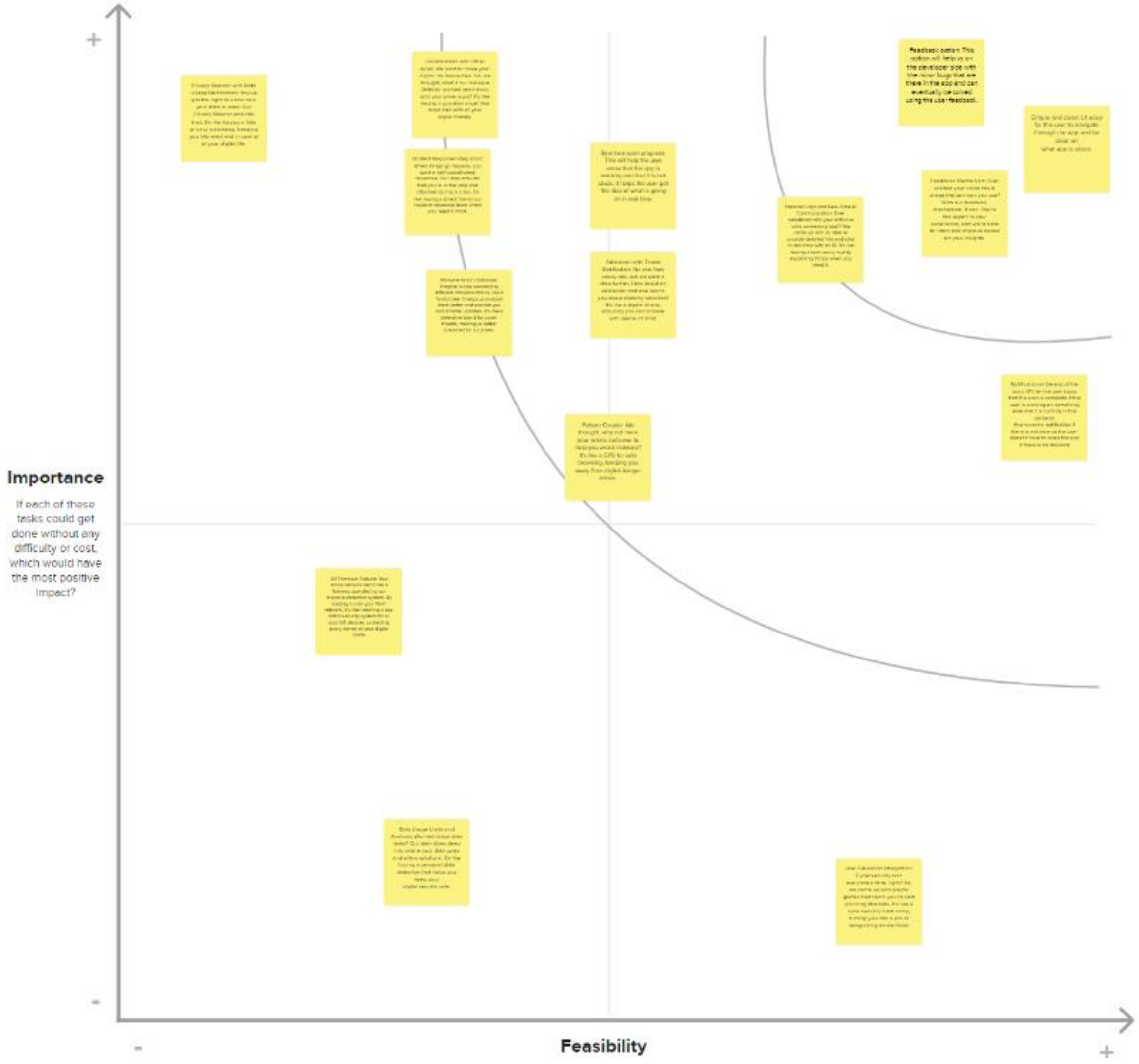
Group 1: User-Centric Features
- Pattern Creator
- Detailed Logs and Real-Time AI Communication
- User Education Integration
- Feedback Mechanism
- Real-time scan progress
- Notification at the end of the scan
- Simple and clean UI

Group 2: Malware Analysis and Information
- Malware Graph Database
- Definitions on what malware is (Education)
- Static page to give some definitions on what malware is (Education)
- Notification if there is malware detected

Group 3: Integration and Collaboration
- Collaboration with Other Apps
- IoT Premium Feature
- Adblocker with Threat Notification
- Privacy Beacon with Data Usage Notification
- Incident Response Integration

Group 4: User Engagement and Transparency
- I thought through the point of view of the user
- If the user wants, we can also provide them with the scan result log files, hence better explanation on the detected malware
- And an extra notification if there is malware so the user doesn’t have to open the app if there is no malware

We then mapped them according to their priority:



What is so unique about our project:

Our system's uniqueness lies in its comprehensive approach to malware detection and classification, real-time communication with an AI assistant, integration with other apps, IoT protection, and user education. The ability to provide detailed explanations in real-time sets us apart, as does our commitment to user feedback and continuous improvement.

What will be the social impact of our project?

Our system's impact extends to both individual users and organizations. It empowers users to navigate the digital landscape safely and educates them about cybersecurity. For businesses and developers, it enhances firewall testing by offering real-time malware monitoring and classification. Rapid threat identification allows for timely responses to potential breaches, ultimately increasing the security of digital assets. By prioritizing user satisfaction and offering a user-friendly interface, we aim to create a safer and more user-centric digital environment.

What is the Business Model of our Solution?

We propose a subscription-based revenue model with tiered pricing. Users can access a basic version of our system for free, while premium features, such as IoT protection and advanced analytics, are available through paid subscriptions. Additionally, we can offer enterprise-level licenses for firewall development companies and testers. Advertising partnerships and data analysis services can provide supplementary revenue streams.

What is the scalability of our solution?

Our solution is highly scalable. As the digital landscape continues to evolve, we can expand our malware database, enhance AI algorithms, and add new features to adapt to emerging threats. We can also scale up our infrastructure to accommodate a growing user base and ensure reliable service. Collaborations with other apps and integration with various devices and systems make our solution versatile and ready for integration into different environments. Our commitment to continuous improvement ensures long-term scalability and relevance.

Technologies Required

1. TensorFlow: TensorFlow is a freely available software library that empowers developers to create and train machine learning models. It was originally created by Google and has found widespread use in diverse fields, ranging from image and speech recognition to natural language processing and robotics. It is popular both in research and industry for its ability to handle complex computations and its ease of use.
   Usage in Malware Detection:
   - Deep Learning Models: TensorFlow is used to construct deep neural networks for processing complex patterns in .exe files.
   - Training: The framework facilitates the training of the neural network using the labeled dataset, enabling the model to learn to distinguish between benign and malicious files.
   - Inference: Trained TensorFlow models perform fast and accurate predictions, classifying .exe files as either malicious or benign.

2. Keras: Keras is a comprehensive library that offers an array of pre-designed layers to developers, including convolutional layers suitable for processing images, recurrent layers ideal for sequential data, and dense layers for general-purpose computation. In addition, Keras provides an array of training tools, including various optimization algorithms, loss functions, and metrics, that developers can specify while building their models to achieve the desired training outcomes.
   Usage in Malware Detection:
   - Simplicity: Keras simplifies the process of building neural networks, allowing quick prototyping and experimentation with different architectures.
   - Compatibility: As part of TensorFlow, Keras seamlessly integrates with TensorFlow's ecosystem, providing a high-level interface for creating complex neural networks.

3. ArgParse: ArgParse is a python standard library used for parsing command line arguments and options. It is very useful in creating

command line interfaces (CLI) that accept various options, arguments and sub-commands. In this project, ArgParse is used to give a CLI to exe- to-img, and specify startup mode for cnn-model.
Usage in Malware Detection:

- User Interaction: ArgParse is used to accept user inputs, such as paths to .exe files and trained models, directly from the command line interface.
- Flexibility: Command-line arguments make the system more versatile, allowing users to specify different files and models without modifying the source code.

Abstract:

In today's technology-driven world, the ever-present threat of malware poses a significant challenge, akin to a digital nemesis for your computer. Our initiative, "Malware Detection and Classification," aspires to fortify your digital existence by enhancing its security and user-friendliness. Leveraging the power of intelligent computing techniques, we endeavor to detect and categorize various forms of malware.

Our mission extends beyond just safeguarding your digital assets; we aim to educate and empower you in the art of preserving your data privacy, ensuring that your confidential information remains just that – confidential. Moreover, we pledge to be your digital ally in times of crisis, ready to assist when things take an unexpected turn. Our ultimate vision is to erect a robust shield against the elusive and ever-evolving threats that lurk in cyberspace, so you can navigate your digital realm with confidence, knowing you're shielded by a smart and vigilant team of digital guardians.

The significance of the "Malware Detection and Classification" project is increasingly pronounced in the current landscape. Malware is continuously evolving, becoming more intelligent and potent, capable of causing harm to your computer, pilfering your financial resources, and even tarnishing your reputation. While conventional antivirus programs are beneficial, they primarily react once a problem has manifested, and they struggle to keep pace with the rapid transformations in malware. What sets our project apart is our harnessing of cutting-edge technology and the power of machine learning to construct a proactive defense system, perpetually vigilant on your behalf.

Yet, our endeavor transcends mere security; it's about putting you in control of your digital universe. We envision a future where you're an integral part of the team, and we employ innovative technology to ensure your perpetual safety and contentment.

As the digital realm continues to expand, so does the spectrum of threats within it. The "Malware Detection and Classification" project represents a substantial stride forward in the battle to safeguard your digital assets. We firmly believe that by harnessing the latest technology, assisting you in your digital journey, and perpetually improving our defenses, we can revolutionize the way we perceive and implement computer safety. Our overarching objective is to transform the digital world into a better and safer place for you.

Model Design:

The escalating threat of malware in the digital landscape has given rise to an urgent need for more sophisticated and efficient detection methods. Conventional approaches like signature-based detection have proven inadequate in identifying novel or previously unknown malware strains. To combat this challenge, researchers have turned their attention to advanced technologies such as Convolutional Neural Networks (CNNs), which have demonstrated exceptional capabilities in image classification.\

CNNs are specifically engineered for tasks involving image recognition and classification. By employing convolution, clustering, and activation functions, CNNs can distill pertinent features from images. When applied to a malware dataset, the convolutional layer dissects the images to discern edges and textures, enabling the network to grasp underlying patterns in the data. The clustering layer's downsampling attributes reduce computational demands, while the activation function permits the model to grow increasingly intricate.

Malware detection utilizing CNN models presents a promising approach by amalgamating the strengths of neural networks and signature-based detection systems. The CNN model not only analyzes a program's behavior and functionality but also contrasts it against predefined signatures to pinpoint known malware.

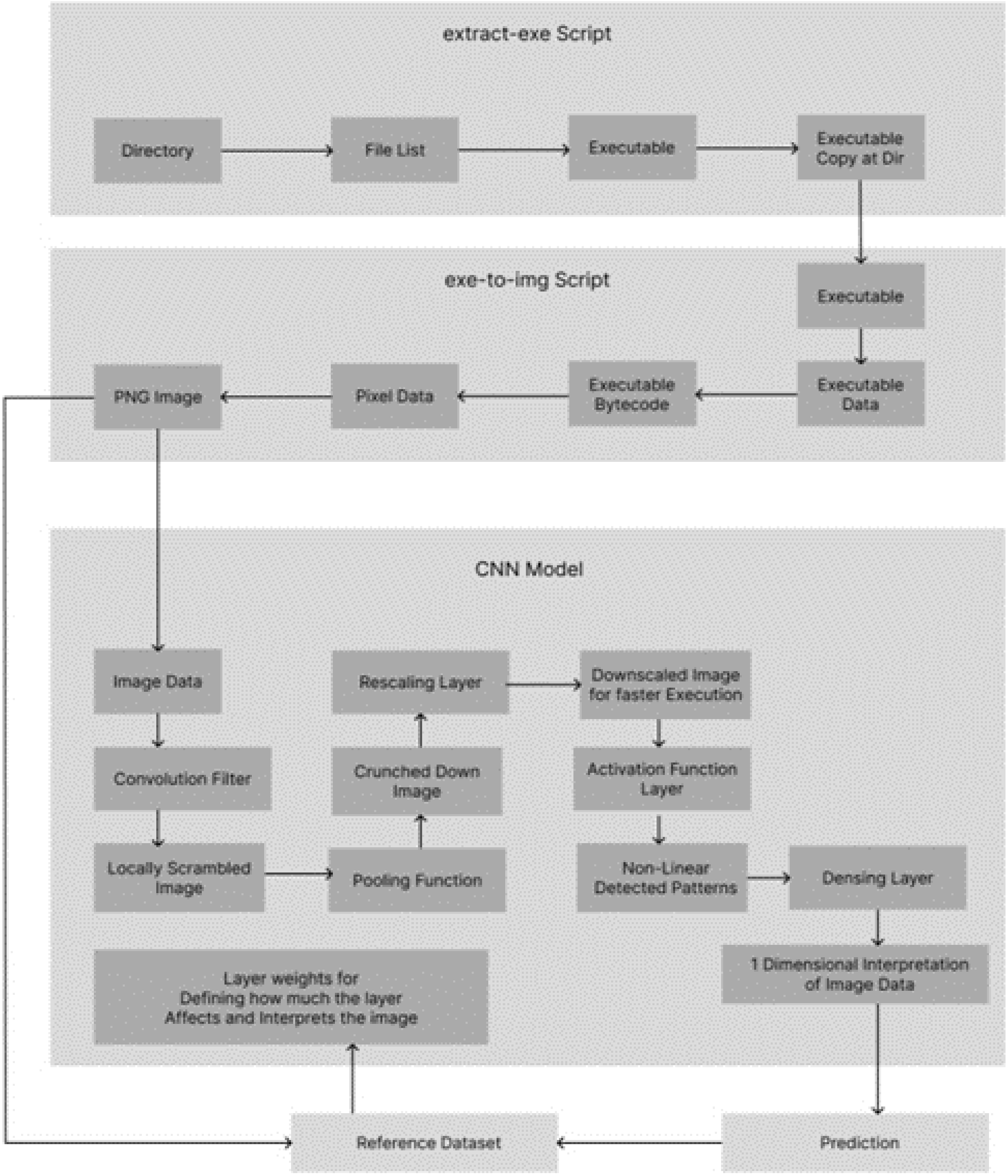
This strategy is pivotal in identifying emerging or unfamiliar malware iterations by identifying distinctive patterns in new entries. The innovation in our proposed system lies in visualizing malware as images and training classifiers using deep learning techniques, thereby ensuring enhanced performance, heightened accuracy, decreased computational expenses, prompt responses to threats, and resilience against cloaking techniques.

By translating malware into visual representations, this approach facilitates the application of image processing techniques like feature extraction and data enhancement. This innovative methodology, coupling CNN models with image processing, represents a significant advancement over conventional detection techniques. It not only offers a more precise and efficient means of detecting malware but also fortifies defenses against cyberattacks.

In the realm of information security, where the stakes are high, the prevalence of malware necessitates robust defenses. Hackers exploit malware to infiltrate computer systems, pilfer sensitive data, and disrupt networks. As organizations increasingly rely on computers and networks, robust malware detection becomes indispensable. Deep learning techniques, particularly CNNs, have emerged as a powerful ally in this battle against malicious software. CNNs, honed for image analysis, decipher complex patterns in malware datasets. By harnessing the synergy between neural networks and signature-based detection, our approach empowers systems to recognize both known threats and novel, evasive malware variants.

Traditionally, malware detection leaned on signature-based systems, relying on predefined patterns to identify known malware. However, the ever-evolving nature of malware renders these methods vulnerable. In contrast, neural networks scrutinize program behavior, bridging the gap between existing knowledge and emerging threats. This proactive approach, integral to our system, acts as an early warning system, safeguarding information systems against unauthorized access and data breaches.

Existing methods, including static and dynamic code analysis, possess their merits but struggle with challenges like code obfuscation and computational overhead. In response, our innovative system visualizes malware as images, capitalizing on the
robust capabilities of CNNs. By integrating TensorFlow and fastai, we've crafted a solution that not only maximizes accuracy but also ensures efficiency and adaptability in the ever-changing landscape of cybersecurity.



Architecture Diagram of the model

Planning:

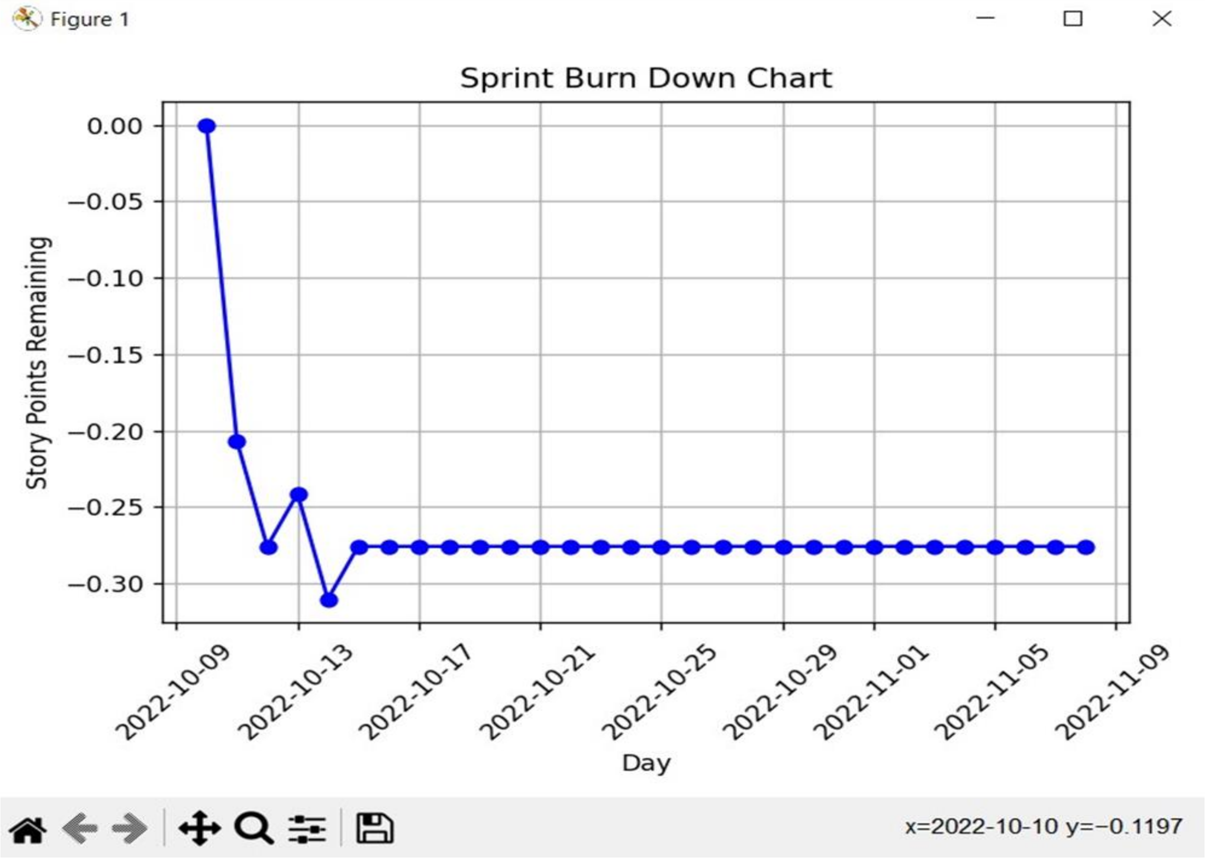
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint 1	Registration	USN-1	As a user, I can create an account by providing my email, setting a password, and confirming it. After registration, I can access my dashboard.	2	High	Chaitanya, Rohit
Sprint 1	Registration	USN-2	As a user, I should receive a confirmation email upon successful registration. I can receive the email and click the confirmation link to activate my account.	1	High	Akshay, Kunawar
Sprint 1	Login	USN-3	As a user, I can log into my account by entering my email and password. I can access my dashboard after a successful login.	2	High	Chaitanya, Rohit
Sprint 1	Registration	USN-5	As a web user, I can register for the application through the web interface. I can create an account by providing my email, setting a password, and confirming it.	1	High	Akshay, Kunawar
Sprint 1	Login	USN-6	As a web user, I can log into the application using the web interface. I can access my dashboard after a successful login.	2	High	Chaitanya, Rohit
Sprint 2	Dashboard	USN-4	As a user, I can view real-time scan progress and receive notifications at the end of the scan. I receive notifications and can monitor the scan's progress in real time.	1	Medium	Akshay, Kunawar
Sprint 2	Registration	USN-7	As a user, I can register for the application through Twitter.	2	Low	Chaitanya, Rohit
Sprint 3	Login	USN-8	As a user, I can log into the application through a fingerprint scanner.	1	Medium	Akshay, Kunawar
Sprint 4	Dashboard	USN-9	As a customer care executive, I can access the dashboard to monitor the status of malware detection for users. I can view the real-time progress of malware detection for all users.	2	High	Chaitanya, Rohit
Sprint 5	User Management	USN-10	As an administrator, I can manage user accounts, including creation, modification, and deactivation. I can create, update, and deactivate user accounts.	1	High	Akshay, Kunawar
Sprint 6	Data Usage Limits	USN-11	As an administrator, I can set data usage limits and analysis options for the system. I can configure data usage limits and analysis options.	2	Medium	Chaitanya, Rohit

Tracker:

Sprint	Total Story Points	Duration	Sprint Start Date	SprintEndDate (Planned)	Sprint Release Date (Actual)
Sprint 1	10	7 Days	October 4, 2022	October 10, 2022	October 11, 2022
Sprint 2	4	6 Days	October 12, 2022	October 17, 2022	October 18, 2022
Sprint 3	2	5 Days	October 19, 2022	October 24, 2022	October 25, 2022
Sprint 4	3	6 Days	October 26, 2022	October 31, 2022	November 1, 2022
Sprint 5	1	3 Days	November 2, 2022	November 4, 2022	November 5, 2022
Sprint 6	2	3 Days	November 6, 2022	November 8, 2022	November 9, 2022

AV=30/15=2
AV=sprint duration/velocity
=(7+6+5+6+3+3)/(10+4+3+2+1+2)
=30/15
=2

ROP(Rate of Progress):



The planning phase is a critical component of our endeavor to develop a robust malware detection and classification system using Machine Learning (ML) techniques. This phase lays the foundation for the entire project and ensures that we proceed systematically towards achieving our goals. Below, we outline the key aspects of the project planning phase:

1. Objective Definition:
- Clearly define the project's objectives, which in this case, are to develop a malware detection and classification system. Specify the desired outcomes and success criteria.

2. Scope Definition:
- Determine the scope of the project, which includes the types of malware to be detected and classified and the platforms or environments it will cover (e.g., Windows, Android).

3. Resource Allocation:
- Identify the necessary resources such as hardware, software, datasets, and personnel. Allocate a budget and establish a timeline for the project.

4. Data Collection and Preprocessing:
- Plan for data acquisition, considering factors like data sources, labeling, and data preprocessing. Collecting a diverse and representative dataset is vital for model training.

5. Model Selection:
- Decide on the ML algorithms and models to be used for malware detection and classification. Assess their suitability for the project's scope and objectives.

6. Feature Engineering:
- Develop strategies for extracting relevant features from the collected data. This is essential for enhancing the model's accuracy.

7. Model Training and Validation:
- Define the methodology for training the models and how they will be validated and fine-tuned. Consider techniques like cross-validation and hyperparameter tuning.

8. Evaluation Metrics:

- Determine the evaluation metrics that will be used to assess the model's performance, such as accuracy, precision, recall, and F1 score.

9. Integration and Deployment:
- Plan for the integration of the trained model into the final system and deployment in a real-world environment, if applicable.

10. Testing and Quality Assurance:
- Define a rigorous testing process to identify and rectify any issues or bugs. Quality assurance is vital to ensure the system's reliability.

11. Documentation and Reporting:
- Establish guidelines for documenting the project's progress and results, which will be crucial for creating a comprehensive final report.

12. Risk Assessment:
- Identify potential risks, such as data leakage or model vulnerabilities, and develop strategies to mitigate them.

13. Project Timeline:
- Create a detailed timeline with milestones and deadlines to track the project's progress and ensure it stays on schedule.

The project planning phase serves as a roadmap, guiding us through the development of a malware detection and classification system using ML. It provides clarity, structure, and a systematic approach to ensure the successful execution of the project while adhering to the defined objectives and constraints.

Approach:

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	With the escalating complexity of cyber threats, there's a growing demand for innovative solutions. We selected this problem statement to enhance firewall testing by offering real-time malware monitoring and classification during the testing process to check if the firewall was working as intended. We recognized the vital importance of instant threat identification, allowing companies to swiftly respond to potential breaches and vulnerabilities.
2.	Idea / Solution description	Our solution, the "Malware Detection and Classification" system, leverages advanced AI and cybersecurity techniques to monitor network traffic, detect malware, and classify it in real-time. Detailed malware detection and classification. User-friendly interface. Real-time communication with an AI assistant. IoT protection. Integration with other apps. User education on cybersecurity. Privacy control. Incident response integration. Feedback mechanism. Data analysis for data leak prevention.

3.	Novelty / Uniqueness	Our system's uniqueness lies in its comprehensive approach to malware detection and classification, real-time communication with an AI assistant, integration with other apps, IoT protection, and user education. The ability to provide detailed explanations in realtime sets us apart, as does our commitment to user feedback and continuous improvement.
4.	Social Impact / Customer Satisfaction	Our system's impact extends to both individual users and organizations. It empowers users to navigate the digital landscape safely and educates them about cybersecurity. For businesses and developers, it enhances firewall testing by offering real-time malware monitoring and classification. Rapid threat identification allows for timely responses to potential breaches, ultimately increasing the security of digital assets. By prioritizing user satisfaction and offering a user-friendly interface, we aim to create a safer and more user-centric digital environment.
5.	Business Model (Revenue Model)	We propose a subscription-based revenue model with tiered pricing. Users can access a basic version of our system for free, while premium features, such as IoT protection and advanced analytics, are available through paid subscriptions. Additionally, we can offer enterprise-level licenses for firewall development companies and testers. Advertising partnerships and data analysis services can provide supplementary revenue streams.
6.	Scalability of the Solution	Our solution is highly scalable. As the digital landscape continues to evolve, we can expand our malware database, enhance AI algorithms, and add new features to adapt to emerging threats. We can also scale up our infrastructure to accommodate a growing user base and ensure reliable service. Collaborations with other apps and integration with various devices and systems make our solution versatile and ready for integration into different

		environments. Our commitment to continuous improvement ensures long-term scalability and relevance.

Control Flow:
A Flow Diagram is a traditional visual representation of the information flows within a system. A neat and clear FD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is Stored.

In our proposed workflow, the user initiates the process by converting executables from a specified directory into images using our provided exe-to-img script. If needed, the segregation of these executables into training, validation, and testing datasets can be simplified using the auxiliary extract-exe script. This script allows the user to copy executable files from a user-defined directory to a
target directory.

It's important to note that if data segregation using the extract-exe script is required, it should be done before the conversion to images.
Once the dataset is successfully segregated, the cnn-model script comes into play. This script is responsible for training the CNN Model on the prepared training and validation datasets. To validate the model's performance, the user can utilize the "Make Prediction"
option within the cnn-model script.
Here's a step-by-step breakdown of the process:

1. Execute the extract-exe script to extract executable files:
Command: `python3 extract-exe.py`

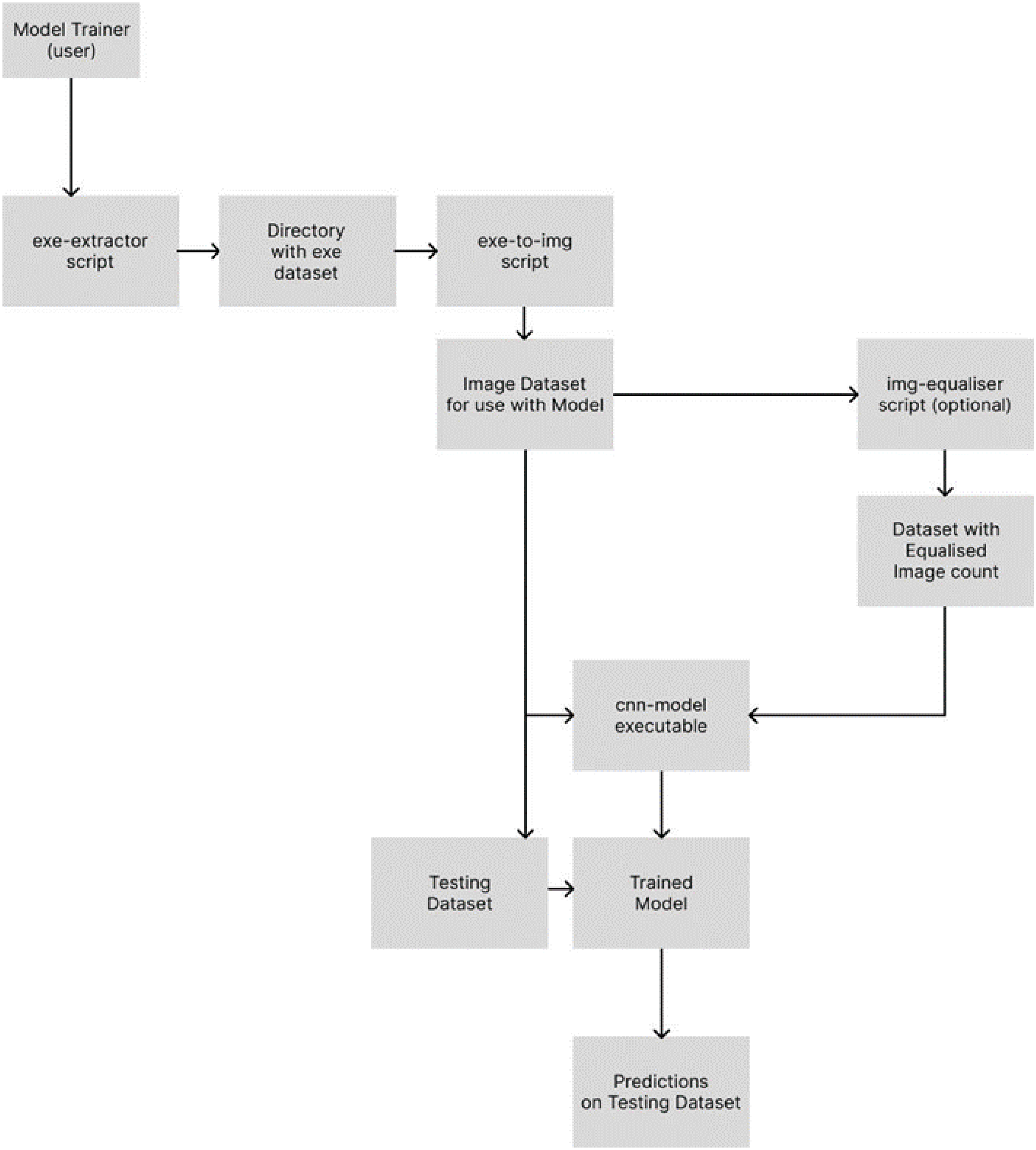
2. Utilize the exe-to-img script to convert executable files to images:
Command: `python3 exe-to-img --folder [Source Directory] --target [Target Directory] [--benign | --malware]`

3. Run the cnn-model script for model training and validation. Use the following options in the cnn-model main menu (order is not

critical):

- Train the model using the "Train Model" option (execute it first when in refresh mode).
- Test the model's performance with the "Make Prediction" option.
- Modify environment variables if image storage locations differ from the defaults:

i. Model location: `./.model/`
ii. Training and validation dataset location: `../Generated Images/`
iii. Testing dataset location: `../Equalised Images/


Model Trainer (user)
exe-extractor script
Directory with exe dataset
exe-to-img script
Image Dataset for use with Model
img-equaliser script (optional)
Dataset with Equalised Image count
cnn-model executable
Testing Dataset
Trained Model
Predictions on Testing Dataset

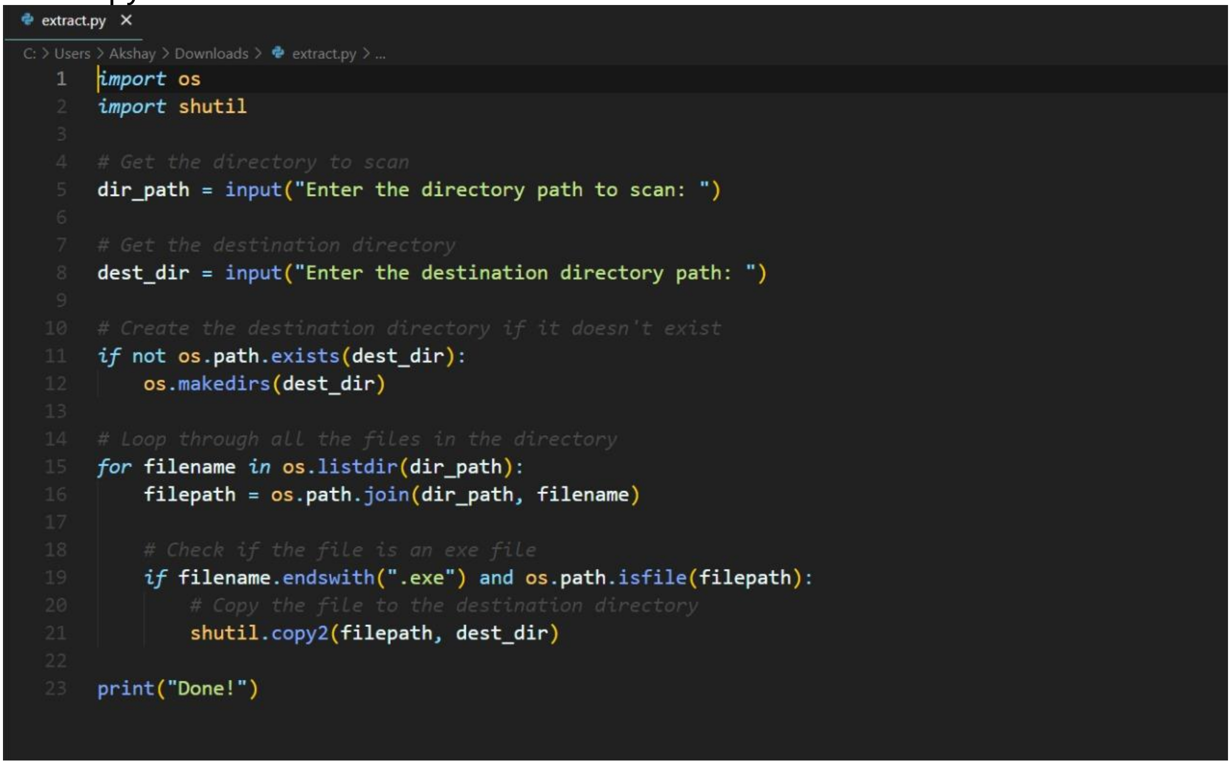
Projected User Stories:

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance Criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can create an account by providing my email, setting a password, and confirming it.	After registration, I can access my dashboard.	High	Sprint-1
Customer (Mobile user)	Registration	USN-2	As a user, I should receive a confirmation email upon successful registration.	I can receive the email and click the confirmation link to activate my account.	High	Sprint-1
Customer (Mobile user)	Login	USN-3	As a user, I can log into my account by entering my email and password.	I can access my dashboard after a successful login.	High	Sprint-1
Customer (Mobile user)	Dashboard	USN-4	As a user, I can view real-time scan progress and receive notifications at the end of the scan.	I receive notifications and can monitor the scan's progress in real time.	Medium	Sprint-2
Customer (Web user)	Registration	USN-5	As a web user, I can register for the application through the web interface.	I can create an account by providing my email, setting a password, and confirming it.	High	Sprint-1
Customer (Web user)	Login	USN-6	As a web user, I can log into the application using the web interface.	I can access my dashboard after a successful login.	High	Sprint-1
Customer (Web user)	Dashboard	USN-7	As a web user, I can access and view detailed logs and real-time AI communication to monitor malware detection progress.	I can access and view the logs, and chat in real-time with an AI for explanations.	Medium	Sprint-2
Customer Care Executive	Dashboard	USN-8	As a customer care executive, I can access the dashboard to monitor the status of malware detection for users.	I can view the real-time progress of malware detection for all users.	High	Sprint-3
Customer Care Executive	Incident Response	USN-9	As a customer care executive, I can access an	I can access incident response	Medium	Sprint-4

			incident response integration feature for crisis situations.	features and receive real-time notifications during crises.		
Administrator	User Management	USN-10	As an administrator, I can manage user accounts, including creation, modification, and deactivation.	I can create, update, and deactivate user accounts.	High	Sprint-5
Administrator	Data Usage Limits	USN-11	As an administrator, I can set data usage limits and analysis options for the system.	I can configure data usage limits and analysis options.	Medium	Sprint-6

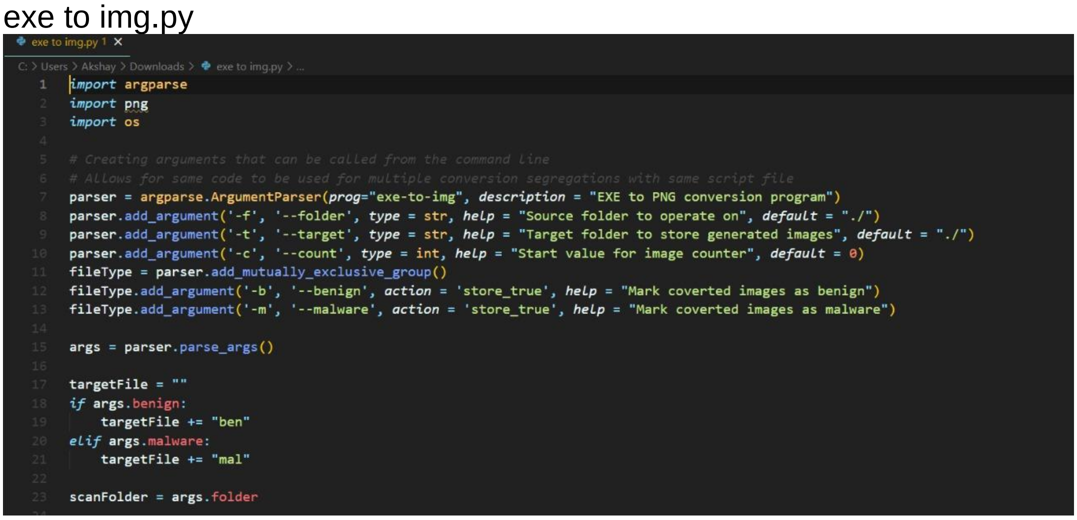
Implementation:

extract.py

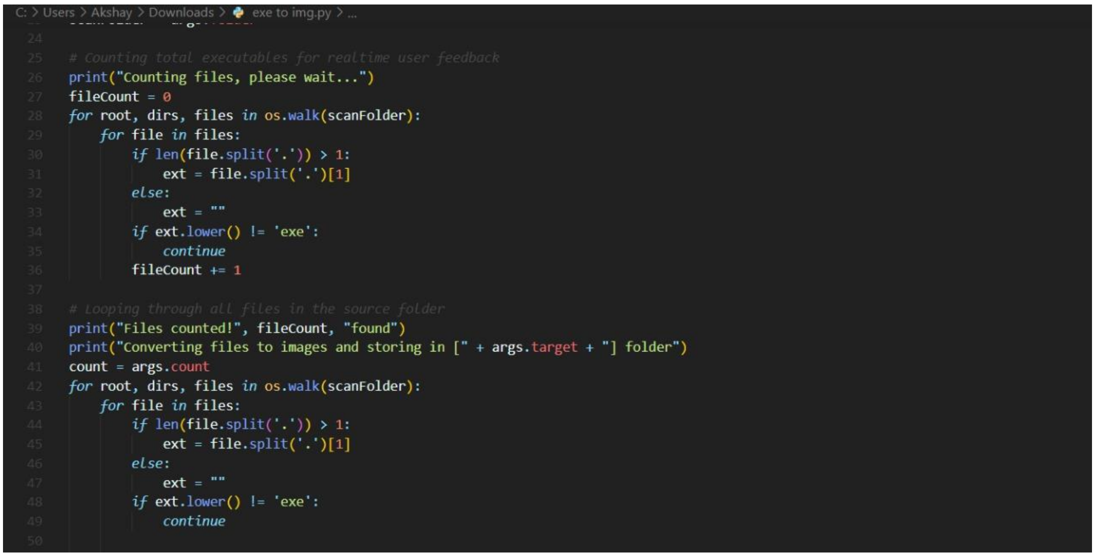


```python
import os
import shutil

# Get the directory to scan
dir_path = input("Enter the directory path to scan: ")

# Get the destination directory
dest_dir = input("Enter the destination directory path: ")

# Create the destination directory if it doesn't exist
if not os.path.exists(dest_dir):
    os.makedirs(dest_dir)

# Loop through all the files in the directory
for filename in os.listdir(dir_path):
    filepath = os.path.join(dir_path, filename)

    # Check if the file is an exe file
    if filename.endswith(".exe") and os.path.isfile(filepath):
        # Copy the file to the destination directory
        shutil.copy2(filepath, dest_dir)

print("Done!")
```

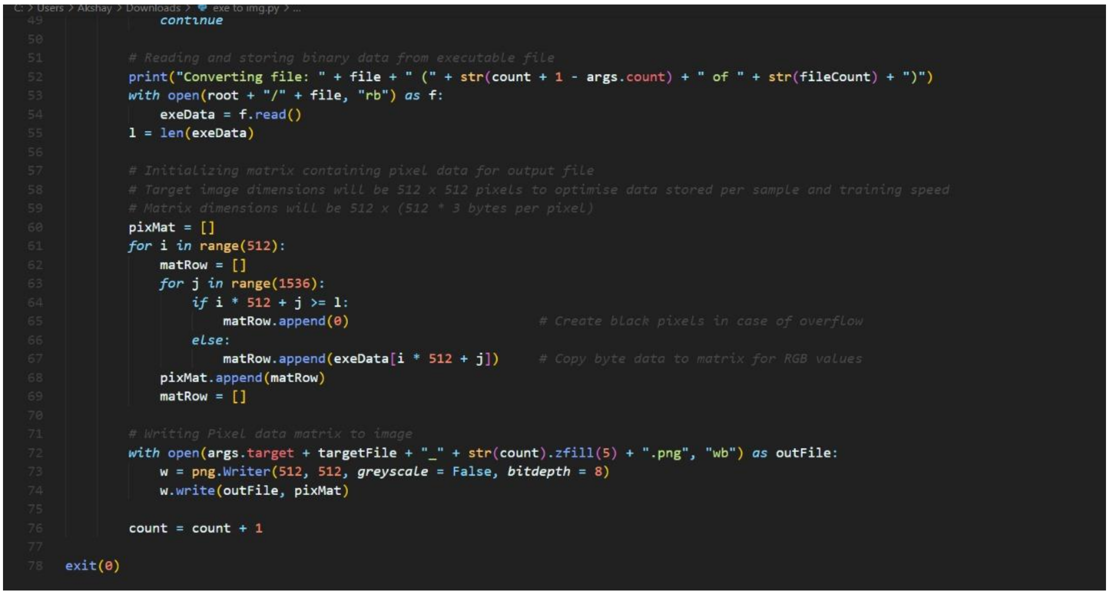
exe to img.py



```python
import argparse
import png
import os

# Creating arguments that can be called from the command line
# Allows for same code to be used for multiple conversion segregations with same script file
parser = argparse.ArgumentParser(prog="exe-to-img", description = "EXE to PNG conversion program")
parser.add_argument('-f', '--folder', type = str, help = "Source folder to operate on", default = "./")
parser.add_argument('-t', '--target', type = str, help = "Target folder to store generated images", default = "./")
parser.add_argument('-c', '--count', type = int, help = "Start value for image counter", default = 0)
fileType = parser.add_mutually_exclusive_group()
fileType.add_argument('-b', '--benign', action = 'store_true', help = "Mark coverted images as benign")
fileType.add_argument('-m', '--malware', action = 'store_true', help = "Mark coverted images as malware")

args = parser.parse_args()

targetFile = ""
if args.benign:
    targetFile += "ben"
elif args.malware:
    targetFile += "mal"

scanFolder = args.folder
```

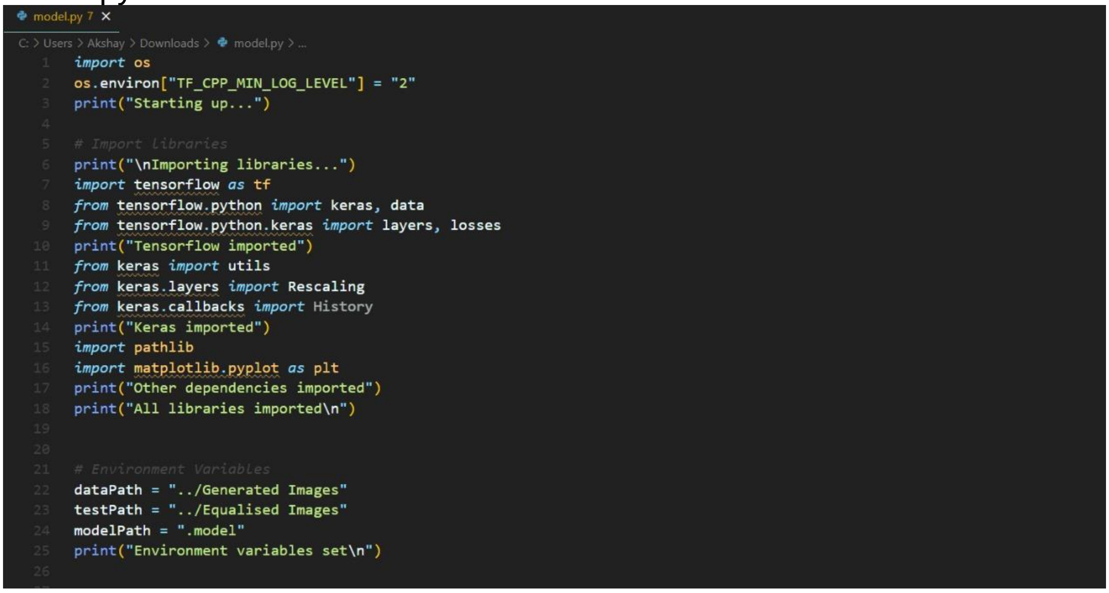


```python

# Counting total executables for realtime user feedback
print("Counting files, please wait...")
fileCount = 0
for root, dirs, files in os.walk(scanFolder):
    for file in files:
        if len(file.split('.')) > 1:
            ext = file.split('.')[1]
        else:
            ext = ""
        if ext.lower() != 'exe':
            continue
        fileCount += 1

# Looping through all files in the source folder
print("Files counted!", fileCount, "found")
print("Converting files to images and storing in [" + args.target + "] folder")
count = args.count
for root, dirs, files in os.walk(scanFolder):
    for file in files:
        if len(file.split('.')) > 1:
            ext = file.split('.')[1]
        else:
            ext = ""
        if ext.lower() != 'exe':
            continue

```

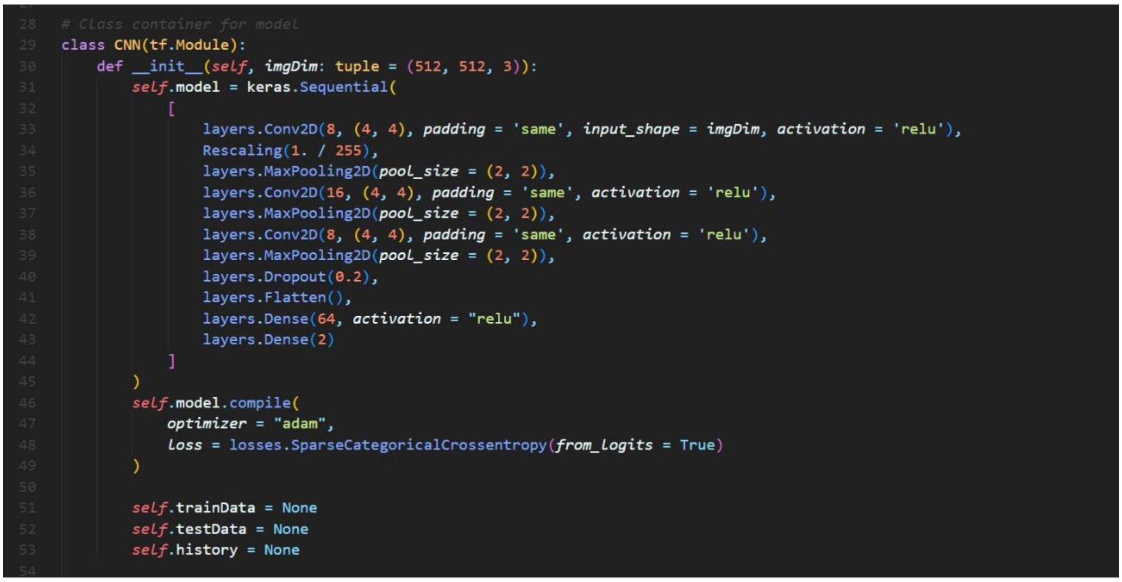


```python
            continue

        # Reading and storing binary data from executable file
        print("Converting file: " + file + " (" + str(count + 1 - args.count) + " of " + str(fileCount) + ")")
        with open(root + "/" + file, "rb") as f:
            exeData = f.read()
        l = len(exeData)

        # Initializing matrix containing pixel data for output file
        # Target image dimensions will be 512 x 512 pixels to optimise data stored per sample and training speed
        # Matrix dimensions will be 512 x (512 * 3 bytes per pixel)
        pixMat = []
        for i in range(512):
            matRow = []
            for j in range(1536):
                if i * 512 + j >= l:
                    matRow.append(0)                        # Create black pixels in case of overflow
                else:
                    matRow.append(exeData[i * 512 + j])     # Copy byte data to matrix for RGB values
            pixMat.append(matRow)
            matRow = []

        # Writing Pixel data matrix to image
        with open(args.target + targetFile + "_" + str(count).zfill(5) + ".png", "wb") as outFile:
            w = png.Writer(512, 512, greyscale = False, bitdepth = 8)
            w.write(outFile, pixMat)

        count = count + 1

exit(0)
```

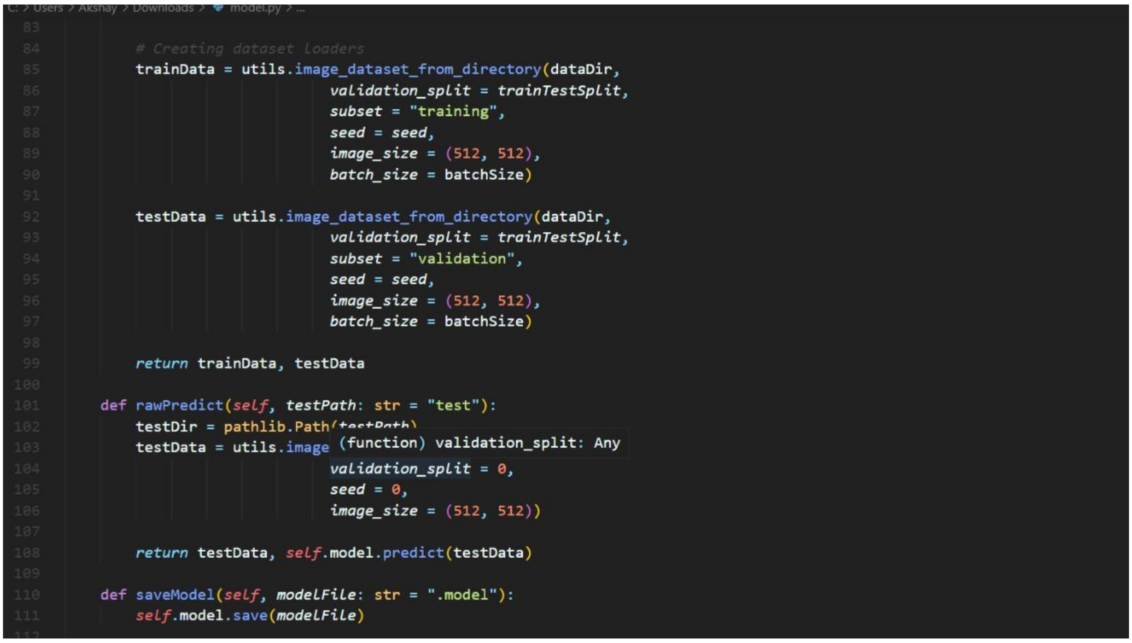
Model.py



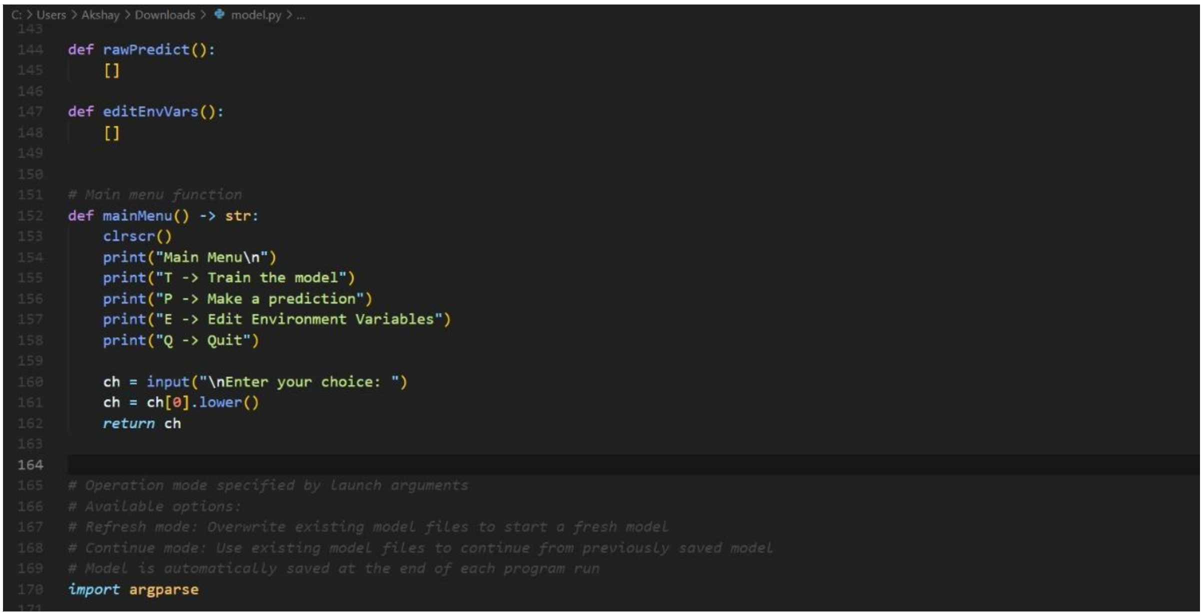
```python
import os
os.environ["TF_CPP_MIN_LOG_LEVEL"] = "2"
print("Starting up...")

# Import libraries
print("\nImporting libraries...")
import tensorflow as tf
from tensorflow.python import keras, data
from tensorflow.python.keras import layers, losses
print("Tensorflow imported")
from keras import utils
from keras.layers import Rescaling
from keras.callbacks import History
print("Keras imported")
import pathlib
import matplotlib.pyplot as plt
print("Other dependencies imported")
print("All libraries imported\n")


# Environment Variables
dataPath = "../Generated Images"
testPath = "../Equalised Images"
modelPath = ".model"
print("Environment variables set\n")

```



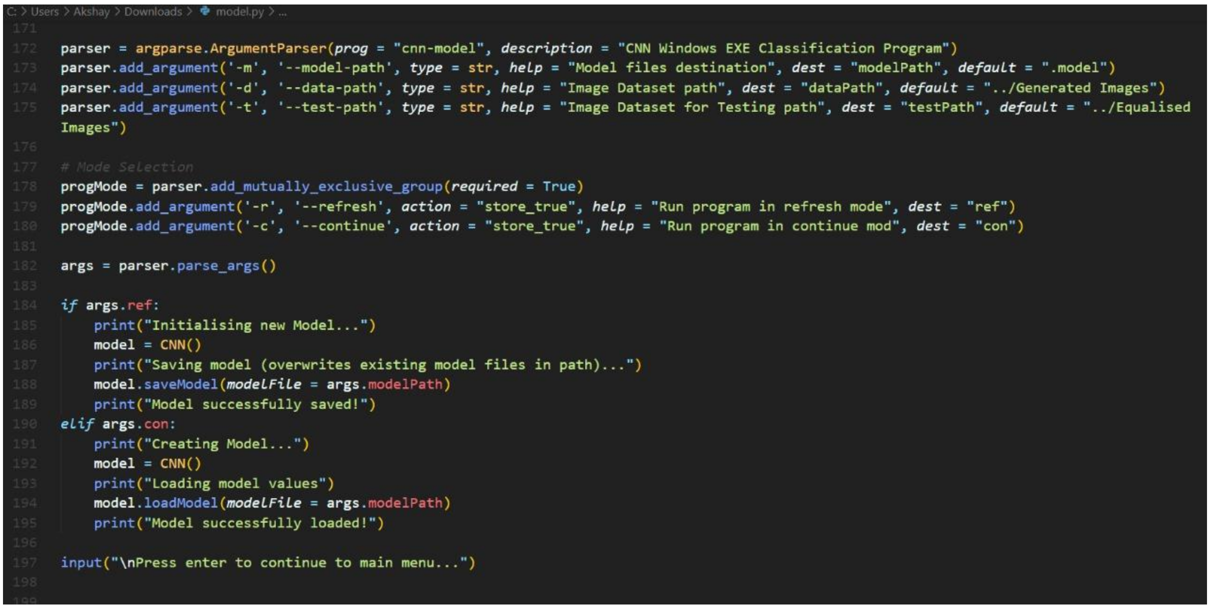
```python
# Class container for model
class CNN(tf.Module):
    def __init__(self, imgDim: tuple = (512, 512, 3)):
        self.model = keras.Sequential(
            [
                layers.Conv2D(8, (4, 4), padding = 'same', input_shape = imgDim, activation = 'relu'),
                Rescaling(1. / 255),
                layers.MaxPooling2D(pool_size = (2, 2)),
                layers.Conv2D(16, (4, 4), padding = 'same', activation = 'relu'),
                layers.MaxPooling2D(pool_size = (2, 2)),
                layers.Conv2D(8, (4, 4), padding = 'same', activation = 'relu'),
                layers.MaxPooling2D(pool_size = (2, 2)),
                layers.Dropout(0.2),
                layers.Flatten(),
                layers.Dense(64, activation = "relu"),
                layers.Dense(2)
            ]
        )
        self.model.compile(
            optimizer = "adam",
            loss = losses.SparseCategoricalCrossentropy(from_logits = True)
        )

        self.trainData = None
        self.testData = None
        self.history = None

```



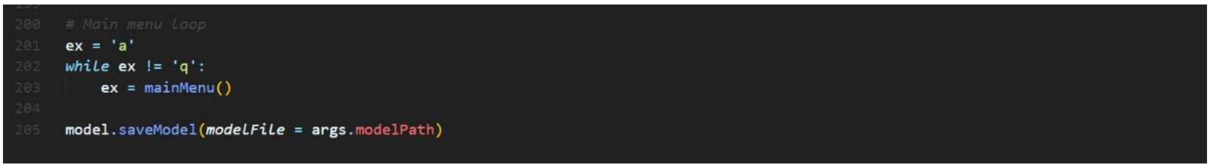
```python

        # Creating dataset loaders
        trainData = utils.image_dataset_from_directory(dataDir,
                              validation_split = trainTestSplit,
                              subset = "training",
                              seed = seed,
                              image_size = (512, 512),
                              batch_size = batchSize)

        testData = utils.image_dataset_from_directory(dataDir,
                              validation_split = trainTestSplit,
                              subset = "validation",
                              seed = seed,
                              image_size = (512, 512),
                              batch_size = batchSize)

        return trainData, testData

    def rawPredict(self, testPath: str = "test"):
        testDir = pathlib.Path(testPath)
        testData = utils.image (function) validation_split: Any
                              validation_split = 0,
                              seed = 0,
                              image_size = (512, 512))

        return testData, self.model.predict(testData)

    def saveModel(self, modelFile: str = ".model"):
        self.model.save(modelFile)
```



```python

def rawPredict():
    []

def editEnvVars():
    []


# Main menu function
def mainMenu() -> str:
    clrscr()
    print("Main Menu\n")
    print("T -> Train the model")
    print("P -> Make a prediction")
    print("E -> Edit Environment Variables")
    print("Q -> Quit")

    ch = input("\nEnter your choice: ")
    ch = ch[0].lower()
    return ch


# Operation mode specified by launch arguments
# Available options:
# Refresh mode: Overwrite existing model files to start a fresh model
# Continue mode: Use existing model files to continue from previously saved model
# Model is automatically saved at the end of each program run
import argparse
```



```python
parser = argparse.ArgumentParser(prog = "cnn-model", description = "CNN Windows EXE Classification Program")
parser.add_argument('-m', '--model-path', type = str, help = "Model files destination", dest = "modelPath", default = ".model")
parser.add_argument('-d', '--data-path', type = str, help = "Image Dataset path", dest = "dataPath", default = "../Generated Images")
parser.add_argument('-t', '--test-path', type = str, help = "Image Dataset for Testing path", dest = "testPath", default = "../Equalised Images")

# Mode Selection
progMode = parser.add_mutually_exclusive_group(required = True)
progMode.add_argument('-r', '--refresh', action = "store_true", help = "Run program in refresh mode", dest = "ref")
progMode.add_argument('-c', '--continue', action = "store_true", help = "Run program in continue mod", dest = "con")

args = parser.parse_args()

if args.ref:
    print("Initialising new Model...")
    model = CNN()
    print("Saving model (overwrites existing model files in path)...")
    model.saveModel(modelFile = args.modelPath)
    print("Model successfully saved!")
elif args.con:
    print("Creating Model...")
    model = CNN()
    print("Loading model values")
    model.loadModel(modelFile = args.modelPath)
    print("Model successfully loaded!")

input("\nPress enter to continue to main menu...")
```



```python
# Main menu loop
ex = 'a'
while ex != 'q':
    ex = mainMenu()

model.saveModel(modelFile = args.modelPath)
```

The model training phase is currently ongoing, and we are in the process of fine-tuning the model to meet our project's objectives. We are actively refining the model and its parameters to achieve the desired results.

Conclusion:

In conclusion, the "Malware Detection and Classification" project marks a significant advancement in the field of digital security. By harnessing state-of-the-art artificial intelligence techniques, we are committed to developing a sophisticated system with the capability to accurately identify and categorize a diverse array of malware types.

This approach not only promises to revolutionize digital asset protection but also equips organizations with proactive detection capabilities, fortifying their readiness to respond to security incidents. The integration of image processing with Convolutional Neural Network (CNN) models has yielded substantial improvements in terms of accuracy, speed, and resilience to cloaking techniques. This underscores the project's crucial role in addressing the challenges posed by malware in today's digital landscape.

In a world where cyber threats are becoming increasingly sophisticated, our commitment to advancing the state of the art in digital security remains unwavering. With the capability to detect emerging malware threats with unparalleled accuracy, our system ensures that organizations can stay one step ahead of cybercriminals.

The proposed enhanced speed of our system is particularly critical in an era where rapid response to security incidents is of paramount importance, and our solution guarantees that organizations can effectively combat potential threats in real-time.

In summary, the "Malware Detection and Classification" project is not just a technological milestone but a comprehensive solution that when refined and perfected, addresses the pressing security challenges faced by organizations today. With its cutting-edge features and unwavering dedication to innovation, it promises to usher in a new era of digital security where cyber threats are met with unyielding defenses and proactive detection capabilities.

Future Scope:

The future scope for research in this domain is both expansive and promising. One intriguing avenue for future exploration lies in a deeper investigation of transformation and aggregation learning methods to further enhance model accuracy. Techniques such as transfer learning, drawing from pre-trained CNN models on extensive datasets, and employing ensemble methods to combine multiple models offer opportunities to elevate overall performance. Moreover, the utilization of generative adversarial networks (GANs) in tandem with CNNs could open up new dimensions of research for refining the accuracy of malware detection models. These GANs could be used to generate adversarial samples, helping train models to be more resilient to sophisticated evasion techniques.

In addition to the technical aspects of model improvement, the integration of explainable AI (XAI) techniques holds the potential to enhance our understanding of the model's decision-making processes and potentially uncover new features to improve malware detection. Exploring techniques like saliency maps, gradient-based attribution methods, and feature visualization could provide valuable insights into how CNN models make decisions. This not only boosts transparency but also offers researchers an opportunity to fine-tune models based on the identified weaknesses and strengths.

As the influence of machine learning in the cybersecurity sector continues to expand, the integration of image processing with CNN models for malware detection remains a dynamic and ever-evolving landscape for future research and innovation. This field presents a wealth of opportunities to develop solutions that enhance digital security. Further avenues for research could involve exploring novel architectures and hybrid models that combine different deep learning techniques to leverage the strengths of each. Moreover, the application of reinforcement learning in optimizing model hyperparameters and training processes represents an exciting frontier in improving model performance.

Furthermore, the analysis of temporal data in addition to static images, such as the behavior of files or network traffic over time, offers a multifaceted dimension for future exploration. This time-series data analysis can provide insights into the evolving nature of cyber threats and help in developing proactive defense mechanisms.

In summary, the future of research in the domain of malware detection and classification is rife with possibilities for innovative advancements. The continuous evolution of machine learning and AI techniques, combined with a deepening understanding of model interpretability, has the potential to usher in a new era of digital security, ultimately contributing to a more robust and secure digital environment.