AI User Safety Application

SAN JOSÉ STATE
UNIVERSITY

The Project Workbook

By

Gulnara Timokhina

Mirsaeid Abolghasemi

Varun Bhaseen

November 4, 2020

Advisor: Professor Vijay Eranti

TABLE OF CONTENTS

*** PAGE INTENTIONALLY LEFT BLANK ***

Chapter 1. Literature Search and State-of-the-Art

1.1 Literature Search

Phishing is a security vulnerability that aims to trick unsuspecting users by mixing social engineering and website spoofing techniques into stealing their sensitive details (e.g. password, bank, or financial details). A typical phishing attack's lifecycle begins with the receipt of a fake e-mail, SMS, or instant message from scammers trying to make users think and believe that it comes from a legitimate source. The messages typically use persuasive claims and a link pointing to a fake web page that mimics the legitimate web page of the target brand. If the user enters their credentials, the life cycle of the attack will conclude by submitting confidential information to phishers which can be misused for online fraud or the misuse of private data.

Phishing is one of the oldest and well-known security vulnerability exploitation technique which relies more on human weaknesses rather than a technological weakness as stated by Bozkir et al. [1] and because of the human factor involved it becomes difficult to eliminate it. Although there are many ways in which a phishing website can be detected the traditional approach has been to rely on databases which maintain a list of these websites and can prevent them from opening at users browser but as mentioned by Abdelnabi et al. [5] any zero-day detections are not possible in these traditional approaches. The update of the database takes a considerable amount of time and usually, a certain number of users have already been affected.

Hence to compensate for human weakness Huang et al [2] in their paper have suggested a more artificial intelligence or deep learning-based technique in which the model can more readily be used to identify the malicious URL for the web page. Based on validation the output of the model will classify whether the web page is phishing or not. Whereas Le et al. [3] suggest a more focused and object detection-based approach where they have enhanced certain features and validate the URL based on a certain character and word frequencies.

Desai et al. [4] in their paper address the problem of the imbalanced dataset and how machine learning techniques can be used to still give good accuracy for the model chosen and features extracted. Bozkir et al. [1] has also compiled a very rich dataset that is publicly available for phishing website detection and classification. Redmon [6] presented a state-of-the-art model called YOLO to carry out a visual similarity and classify two objects in an image. This technique can be used for detecting the logo in an image and check its authenticity from a confidence score generated.

1.2 Research Paper Inputs

The first step for the project approach was to identify what existing research has been carried out for similar objectives. The first relevant paper that we came across was for “LogoSENSE” (Bozkir 95). In this paper, the author highlights what are the different underlying features that are present

in the dataset and how to carry out feature extraction. Also, it explains techniques like HOG (Histogram of Oriented Gradients) and what role it plays in detecting the phishing website by comparing logos and images. This paper uses machine learning techniques rather than deep learning techniques, but the approach to enhance the features in the dataset is useful and can be reused. The authors of this research paper have created their dataset (LogoSENSE) which we will be using as a baseline dataset for our project.

The second paper that we have identified is "Phishing URL Detection via CNN and Attention-Based Hierarchical RNN" (Huang 112). The authors have used CNN and RNN to extract URL texts or characters or words and use them as features to authenticate and verify a website. The authors worked on a dataset that takes a screenshot of a web page and then uses neural networks to extract features and classify the webpages. We will be using the approach for feature enhancement and feature extraction for URL validation as suggested by the authors.

The third paper which we identified is URLNet (Le 1802.03162). The authors of this paper focus on using lexical features in a URL to identify whether the website is phishing. This is achieved by using a CNN model that can focus on character level and word level embeddings. The authors have built their dataset and carried out feature extraction using position-sensitive bigrams and trigrams. The conclusion for this paper was to build a state-of-the-art model which can detect phishing website by analyzing the URL. We will be using the feature extraction technique and try to combine it with other URL validation techniques listed earlier for the detection of the phishing website.

The fourth paper we have is Malicious Web Content Detection Using Machine Learning (Desai 1432). This paper addresses the imbalanced dataset and what approach should be taken for sampling. The paper also addresses the different dependencies that are there for the creation of applications for data science (Machine learning and Deep learning models). We will be referring to this paper for application building approach for web browsers.

The final paper that we are referencing is VisualPhishNet. This paper highlights the zero-day phishing website detection approach. What it essentially highlights is that if a phishing website is created today and it is not a part of any phishing website database then how good the model is to detect this fresh new website as a Phishing website. We will be using the zero-day phishing detection from this paper (Abdelnabi 1909).

1.3 State-of-the-Art Summary

The primary objective of the project is to create an application that can detect phishing websites using deep learning algorithms and techniques. Deep learning techniques have a wide range for analyzing data in multiple formats and types, for example, we have CNN for analyzing data in image formats and then we have RNN that analyses data in textual format and so on. For each type of deep learning technique, there are many state-of-the-art models created and available for use.

For our project, we will be using the object detection technique YOLO for logo identification on any web page and give a confidence score on its authenticity. Although there are many state-of-the-art techniques, YOLO happens to be one of the fastest image classification and object detection techniques (Redmon 779).

For URL validation we will be using character CNN-RNN models. Currently, we are still exploring all the models to check which one performs the best with YOLO and can enable quick outputs. The third and final state-of-the-art model will be for malicious content detection on the web page. For this activity, we will be using attention-based NLP transformation models BERT, hierarchical RNN, LSTM, or something similar. For feature extraction, we will be using Tesseract which is the state-of-the-art OCR (Optical Character Recognition) for extracting texts, characters from images, or screenshots.

1.4 References

1. Huang, Yongjie, Yang, Qiping, Qin, Jinghui, & Wen, Wushao. (2019). **Phishing URL Detection via CNN and Attention-Based Hierarchical RNN**. *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*

2. Hung Le, Quang Pham, Doyen Sahoo, Steven C.H. Hoi. 2018. **URLNet: Learning a URL Representation with Deep Learning for Malicious URL Detection**. *In Proceedings of ACM Conference, Washington, DC, USA, July 2017 (Conference'17), 13 pages.*

3. Desai, A., Jatakia, J., Naik, R., & Raul, N. (2017, May**). Malicious web content detection using machine learning**. *In 2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT) (pp. 1432-1436). IEEE*.

4. Bozkir, Ahmet Selman, & Aydos, Murat. (2020). **LogoSENSE: A companion HOG based logo detection scheme for phishing web page and E-mail brand recognition**. *Computers & Security, 95, 101855.*

5. Abdelnabi, Sahar, Krombholz, Katharina, & Fritz, Mario. (2019). **VisualPhishNet: Zero-Day Phishing Website Detection by Visual Similarity**. https://arxiv.org/abs/1909.00300

6. Redmon, Joseph, Divvala, Santosh, Girshick, Ross, & Farhadi, Ali. (2016). **You Only Look Once: Unified, Real-Time Object Detection**. *2016, 779-788.*

Chapter 2. PROJECT JUSTIFICATION

A user's privacy protection is of prime importance nowadays. The majority of user privacy violations happen through phishing attacks. As per the reports published by the *Anti-Phishing Working Group for Q2 2020 (APWG, Q2-2020)*, it has been observed that the number of phishing sites detected was 146,994 of which 78% now use Secure Sockets Layer (SSL) protection. Also, there is a surge of 20% in phishing attacks on social media users.

The primary goal of a phishing attack is to exploit human weaknesses. The challenge with current phishing detection is the lack of availability of a reliable state-of-the-art tool which could compensate for these weaknesses. The majority of phishing detection technology is based on a classical approach where the agent (detector) relies on information on which it has been trained. The agent does not factor into a real-time artificial intelligence-based detection which can detect the most recently evolved techniques on which phishing attacks are based

We will be proposing an artificial intelligence-based real-time detector that can protect the user's privacy details by constantly scanning a web page for malicious scripts, phishing contents, domain authenticity, and logo identifiers. Our approach will be using not only natural language processing but also computer vision to detect the authenticity of the web page and the use of logos or images on that web page. The outcome will be to run a plugin on a browser that can consume minimal resources and can give a quick scan notification about the safety of the web page.

Chapter 3. PROJECT REQUIREMENTS AND SCOPE

3.1 PURPOSE AND OBJECTIVES

3.1.1 Purpose

The primary purpose of the project is to create a web browser plugin or extension that has Artificial Intelligence capability to detect a phishing website. The plugin should have a k-shot capability with an optional capability of zero-shot learning.

3.1.2 Objectives

The following are the key objectives that are expected out of the project:

a. The application must be a web browser plugin or extension

b. The application must generate a notification for the user in the following three categories:

   i. Alert notification: The webpage is malicious and is classified as phishing

   ii. Warning notification: The webpage and content seem to be malicious and are classified as suspicious

   iii. Safe logs: The webpage is safe and is classified as legitimate

c. The application must be able to take a screenshot of the entire web page for analysis

d. The application must be able to extract the URL (characters and texts) for analysis by the classification model

e. The application must identify the correct position of a brand logo.

f. The application should validate and verify the confidence score for a brand logo on the webpage

g. The application should not slow down the user experience in browsing

h. It is desired that not more than 7% CPU usage should be carried out while the user is browsing

i. The peak usage of 80% CPU should not be more than 1 minute and the model should give results within this duration.

The following is the optional features that can be considered:

a. The application can also detect the web page content using NLP transformation models

b. The application can also detect deep fakes along with phishing website

c. The application could contain an ad blocker and disable undesired pop-ups from opening

3.2 SCOPE OF WORK

Following is the scope for the project:

a. The application will detect the phishing impersonation for a minimum of 10 brand targets.

b. The application will be packaged as a browser plugin

c. The scope of the browser is restricted to the google chromium platform

d. The classification model will be based on Deep Learning Algorithms

e. The model will classify the web page as phishing or legitimate based on

   i. URL validation

   ii. Logo validation

3.3 NON-FUNCTIONAL REQUIREMENTS

Based on the purpose and objectives following are the list of non-functional requirements:

a. The application should be easy to install. Preferably one-click installation

b. The warning and alert notification should be user friendly, concise, and should not take a huge amount of screen space. Preferably 120 x 240 pixels or 5% of screen size

c. The dataset needs to be highly data augmented so that model prediction can be generalized across multiple scenarios for each brand target

3.4 FUNCTIONAL REQUIREMENTS

The requirements are divided into a Mu.S.Co.W. approach (**Mu**st Have, **S**hould have, **Co**uld have, and **W**ould have). Based on the purpose and objectives following are the list of functional requirements:

3.4.1 Essential

The below-mentioned requirements *must be* and *should be* part of the application:

a. Entire webpage Screenshot capability

b. The application must have the capability to hold screenshot in RAM for analysis

c. Capability to generate floating notification or message on user screen in a browser

d. OCR capability to extract text and characters for URL validation

e. Object detection capability to detect logo on a web page

f. Multi-label classification of a webpage (Phishing, Suspicious or Legitimate)

g. Best neural network weights or checkpoint to be used for runtime application usage

h. Application development on chromium platform

i. Multiple concurrent requests handling capability

j. Request Queue and time-out features for handling quick user browsing

k. Using state-of-the-art object detection technique or algorithm based on neural networks

l. Using state-of-the-art optical character and word recognition technique based on neural networks

m. Ten targets brand identification for phishing detection

n. Compiling dataset and carrying out exploratory data analysis for target brands

o. Manual annotation of the dataset

p. Data cleaning and pattern identification

q. Finetuning the model using the tensor board or similar techniques

r. Publication of evaluation metrics and performance metrics

s. Whitelist URL and domain names for target brands selected

t. Application runtime idle usage should not be more than 7% of CPU usage (Core-i5 6th generation)

u. Finetune performance and output generation for application with a latency of less than one minute during runtime

3.4.2 Desired

The below-mentioned requirements *could be* part of the application if time permits:

a. Build NLP Model to analyze the entire content of a webpage for malicious intent

b. Using state-of-the-art NLP (Natural Language Processing) technique based on a neural network to detect optical characters and texts for malicious intent

3.4.3 Optional

The below is the list of optional features that *would be* considered in the future to include in scope:

a. Model for deep fake detection for any video or image on a webpage

b. Adblocker capability

Chapter 4. PROJECT DELIVERABLES AND DEPENDENCIES

4.1 PROJECT DELIVERABLES

Following are the list of deliverables that will be provided as part of this project:

4.1.1 Technical Deliverables

a. Class diagrams and description

b. Packaged application

c. Deep Learning saved model weights

d. Source code for packaged application

e. Jupyter file for deep learning model training and tuning

f. Dataset required for deep learning model training

g. Data Science model architecture

4.1.2 Academic Deliverables:

a. Project report and/or research paper comprising the solution, metrics, approach, and methodology

b. Setup and installation guide

c. Workbooks, abstract, and other assignment deliverables

d. Project progress reports

e. Benchmarking reports

f. Minutes of meeting with the advisor

g. Trello board and Slack channel details and all accesses

4.2 DEPENDENCIES

4.2.1 Data Dependencies

a. Select most common vulnerable sites phishers attack (top ten out of top 50)

b. Compiling hand-annotated dataset for target brands

c. Generating enough data volume for the model to learn

d. Using state-of-the-art data augmentation techniques

e. Finding relevant patterns and features in exploratory data analysis

4.2.2 Technical Dependencies

a. Tensorflow or Pytorch backward compatibility for model building

b. Use of GPU while running the packaged application on a web browser

c. HPC machine for training and building the model

d. Finetuning the deep learning model to get the desired accuracy

4.2.3 Tool dependencies

a. Web browser or chromium platform limitations especially on URL validation access for packaged application

b. Using archive.org (way back machine) to compile webpages for various target brands and their evolution

c. Using PhishTank and/or PhishBank for building the library for phishing and malicious URL web pages

d. Using HyperLabel and/or a manual program to hand annotate and create a custom dataset for various target brands

e. A reliable database for validating domain addresses for URL validation model

4.2.4 Skill Dependencies

a. Finding relevant state-of-the-art models for the project objectives

b. Learning JavaScript for application building and packaging on chromium platform

c. Optimized application using relevant design patterns for scalability

d. Optimized performance by leveraging the right data structure and algorithms

e. Translating deep learning model into a packaged application

f. Combining NLP (Natural Language Processing) and CV (Computer Vision) as a Multi-task multi-label classification model

Chapter 5. Architecture Description

5.1 Data Preparation

Below is the data preparation architecture:

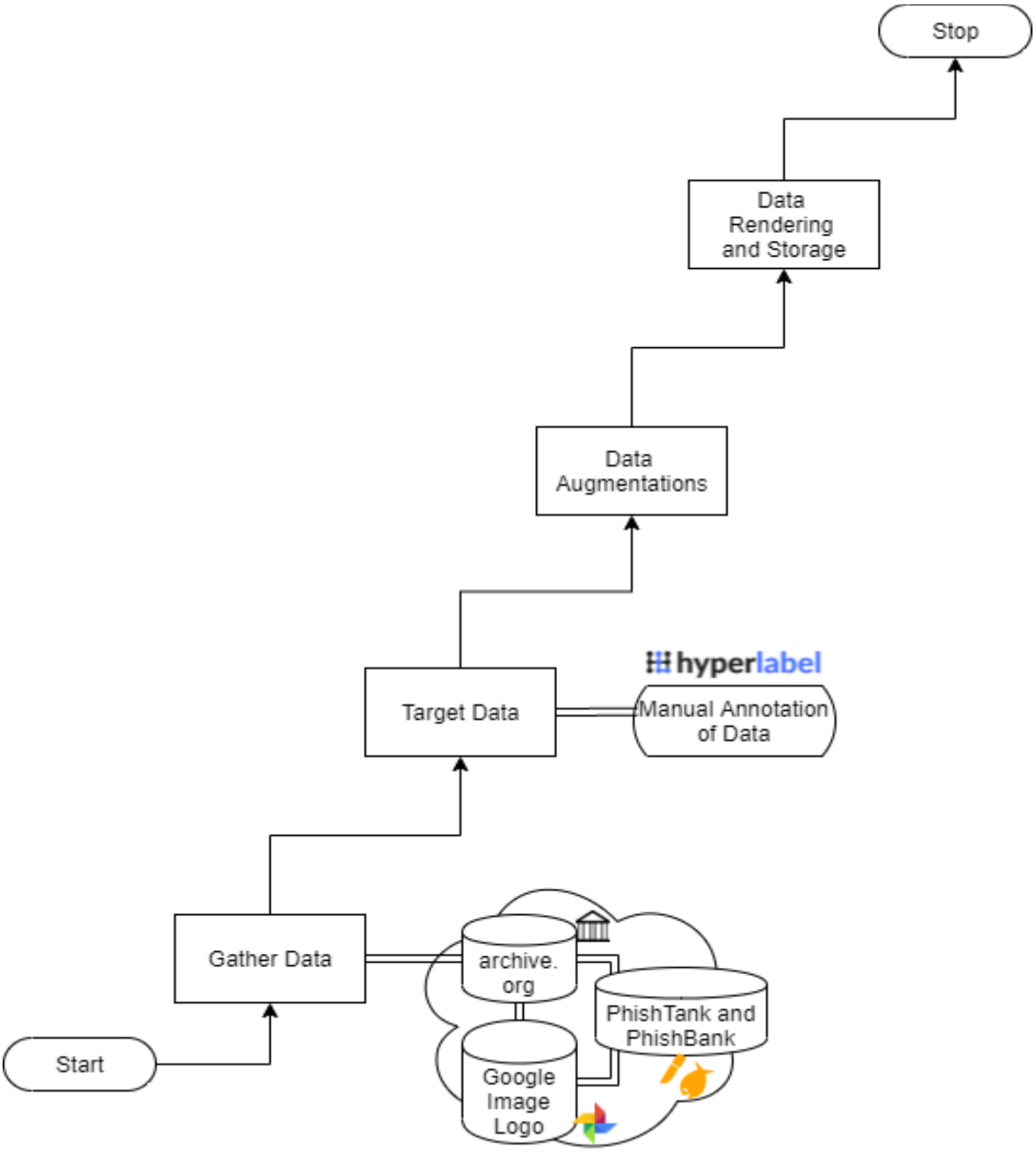

Figure 1: Data Preparation Architecture

5.2 Model Architecture

Below is the deep learning model architecture with input and output for each model:

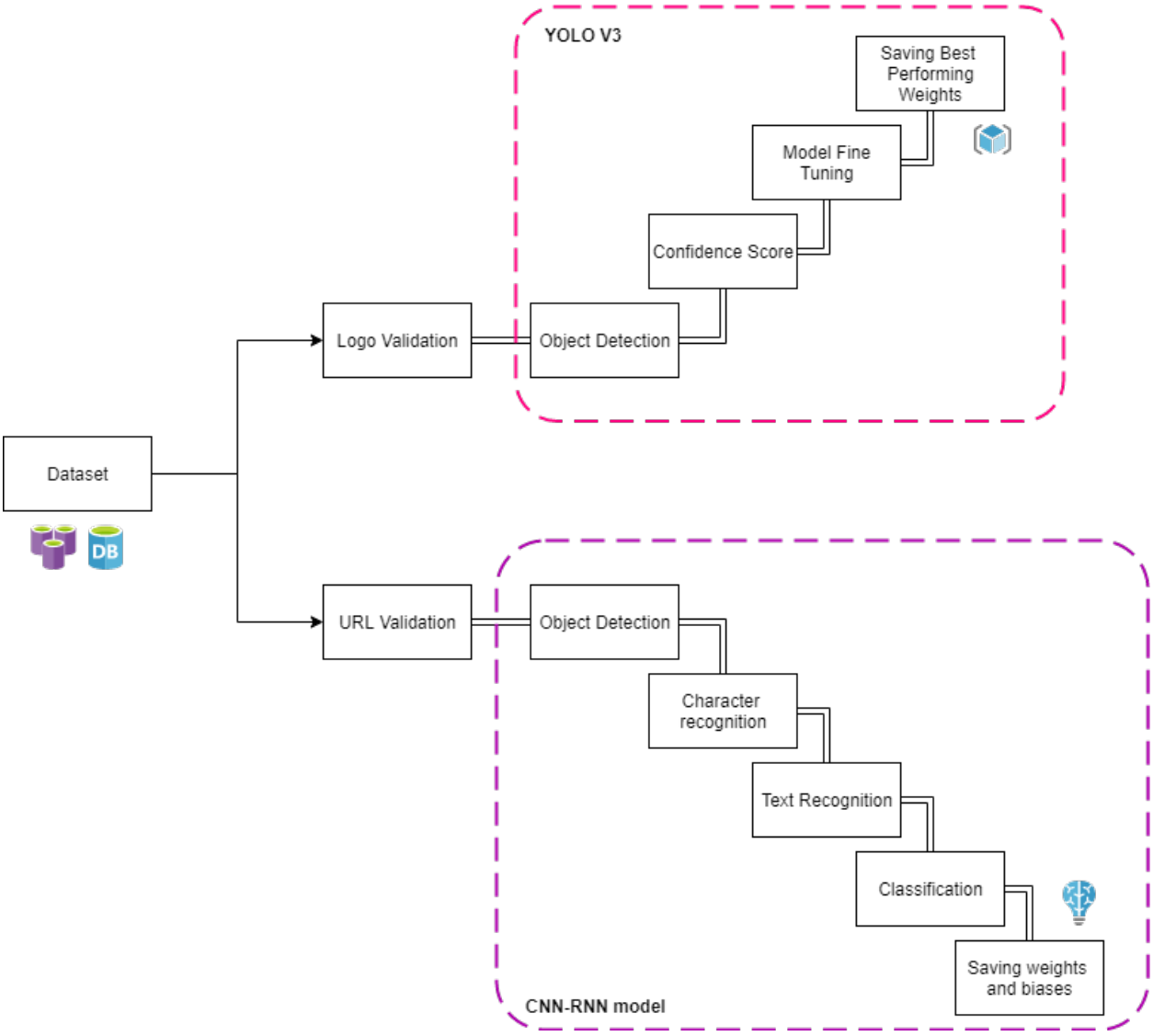

Figure 2: Deep Learning Model Architecture

5.3 High-Level Application Architecture

The figure below provides a high-level architecture for the application. In our application concept, we use two deep learning models based on YOLO and CNN-RNN multi-task multi-label classification. To get better phishing prediction, these algorithms will be applied to the features extracted from web pages like logo, texts, and address bar.

The deep learning models will be used in our application to get these extracted features and then it will classify the webpages to show that a webpage is phishing, suspicious, or legitimate. The application will be packaged as a web browser extension and/or a plugin.

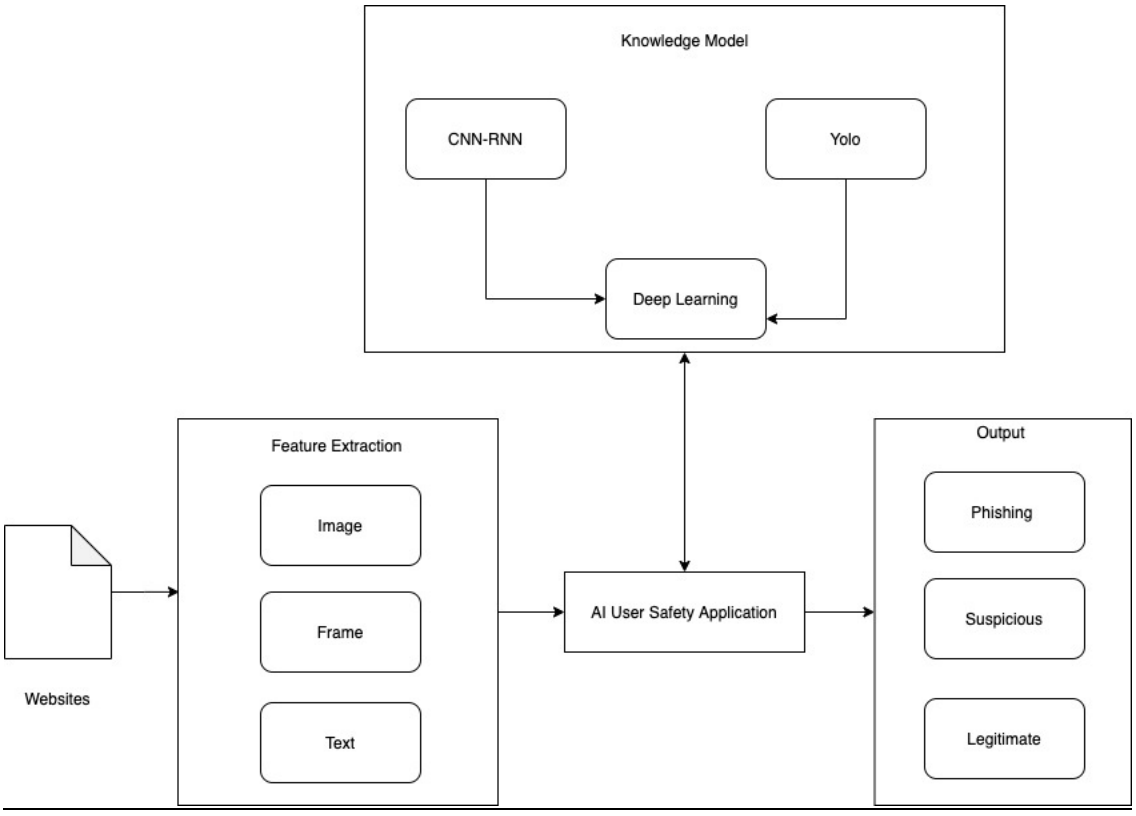

Figure 3: High-level application architecture

5.4 USE CASE DIAGRAM

Below is our initial use case diagram. The user will see the screen and our plug-in extension will take care of the status of the website. In our application, the first step is to take a screenshot of the webpage to extract the main features from it. One sub-step will be to detect the logo of the website and match it if it is fake or real. Meanwhile, URL validation will be executed. Also, it will be checked if the webpage consists of any form of submission or hyperlinks to other webpages. In the end, it will show a warning if the website is safe or not safe.

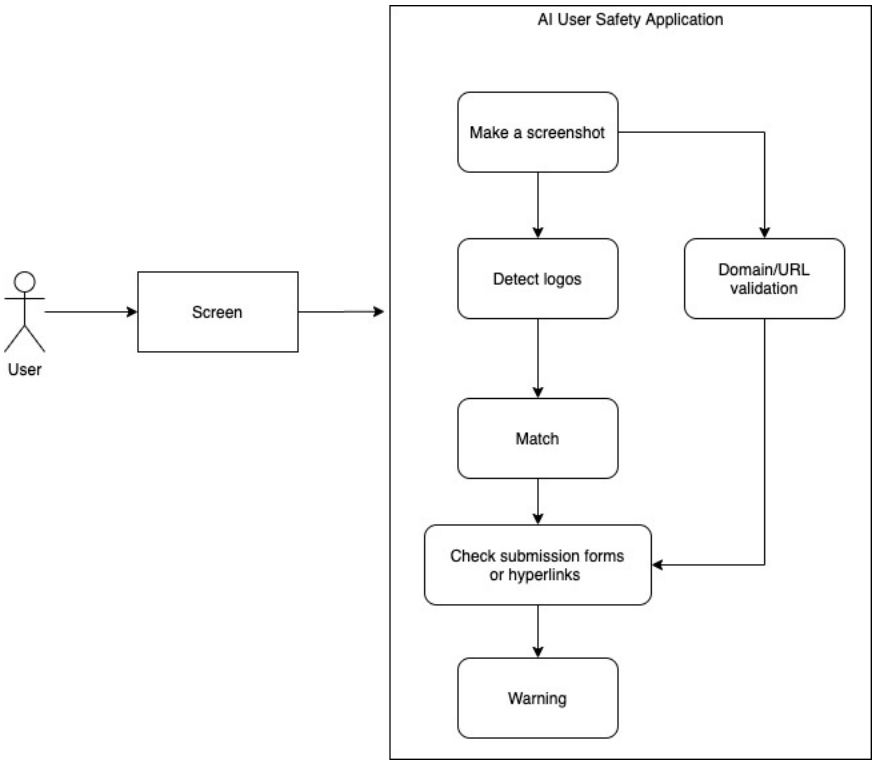

Figure 4: Initial use case diagram

Chapter 6. PROJECT DESIGN

6.1 UML CLASS DIAGRAM

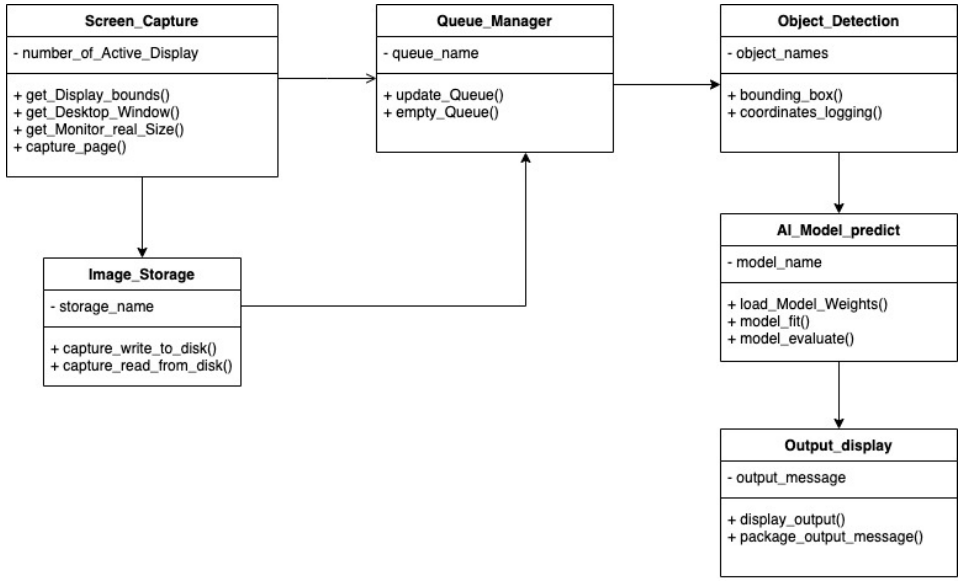

Figure 5: Initial class diagram

6.2 UML SEQUENCE DIAGRAM

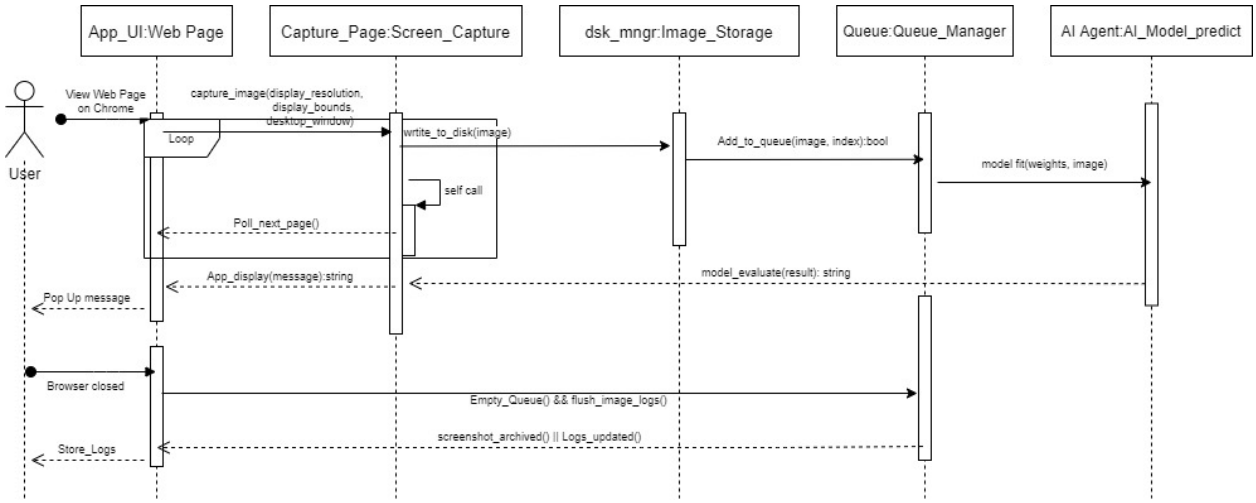

Figure 6: Initial sequence diagram

6.3 UI WIREFRAME FOR CHROME EXTENSION

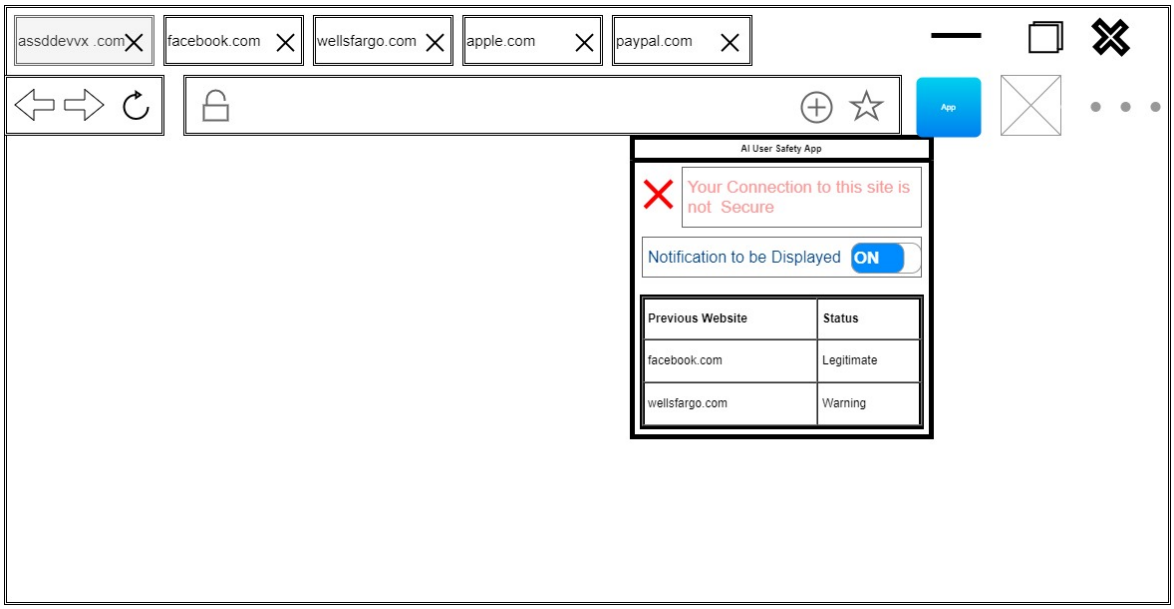

Figure 7: Initial UI Wireframe

Chapter 7. TEST PLAN

7.1 METRICS FOR MODEL EVALUATION

The primary objective or problem statement of our application is to classify whether a web page is a phishing webpage or not. The best metrics available would be the one that can let us know how many times we correctly identified the web page as a phishing web page and how many times we gave a false positive.

The best set of metrics for such a problem statement is the **Confusion Matrix.** The metrics that we will be using are defined as below:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F_1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

$$Accuracy = \frac{TP + TN}{TP + FN + TN + FP}$$

here TP *is True Positive;* FP *is False Positive,* FN *is False Negative and* TN *is True Negative*

Definitions:

- **Precision** measures the rate of samples correctly identified as phishing against all samples identified as phishing.
- **Recall** measures the rate of phishing samples correctly identified as phishing to all phishing samples existing in the set.
- **F1–score** measures the harmonic mean between precision and recall.
- **Accuracy** measures the total rate of correct prediction.

We also want to compare two models in terms of these metrics.

7.2 TESTING PLAN AND TEST CASES

NOTE: Apart from general unit tests for each class and function we will carry out the below tests

7.2.1 Logo Validation Model Testing approach

ID	MLDL001
Functionality	Logo Validation
Model	YOLO
Technique	Transfer Learning
Data Format Input	Images
Dataset	Self Annotated Logo Dataset
Sources	• LogoDet • Archive.org • Google Image Search Crawling
Volume	100 samples per target brand (Total 1000 Images)
Split Ratio	• Train: 80% • Validation: 20% of Train • Test: 20%
Overfitting Check	• K-Fold Cross Validation (Training Dataset (80%)) • Image Augmentations (Training Dataset (80%))
Underfitting Check	• Number of Epochs in training to be varied from 500 to 1000 • Best performing model to be selected
Metrics	• Confidence Score • Accuracy

7.2.2 URL Validation Model Testing approach

ID	MLDL002
Functionality	URL Validation
Model	CNN-LSTM
Stage-1 Input	Images
Stage-2 Input	Strings
Technique	Transfer Learning or Retraining all layers
Dataset	Prebuilt Dataset
Sources	• Archive.org • Phishtank
Volume	120 samples per target brand (1200 URLs)
Split Ratio	• Train: 80% • Test: 20%
Overfitting Check	• Early Stopping • Regularization • Masking
Metrics	• Recall • Precision • F1 Score • Accuracy

7.2.3 Chrome Extension PostMessaging Test Cases

ID	MLDL003
Type	Security Testing
Functionality	User privacy data exposure and leakage
Guideline	OWASP
Testing areas	Penetration testing
Risk area	PostMessaging

Script	{
	"name": “My Extension",
	"description": “My Super Chrome Extension",
	"version": “1.0",
	"background": {
	"scripts": [“js/background.js"]
	},
	"content_scripts": [
	{
	"matches": ["<all_urls>"],
	"js": ["js/jquery.js", "js/content.js"]
	}
	],
	"permissions": ["tabs", "http://*/*", "https://*/*"]
	}

API Checklist	• Cross-origin communication • Origin Validation • Message Listeners
Vulnerable areas	a. Iframes b. Opening a new window
Threat	postMessage data can be passed into background script context, and in some cases even reach OS via Native Messaging API
Best Practice	a. pay attention to origin validation in message listeners b. consider origin bypass tricks c. do not rely on magic strings

7.2.4 Chrome Extension Data Exposure Test Case

ID	MLDL004
Type	Security Testing
Functionality	User privacy data exposure and leakage
Guideline	OWASP
Testing areas	Penetration testing
Risk area	Data Exposure
API Checklist	• Caching details checklist • Encrypted logs
Vulnerable areas	a. Application logs b. Folder path exposure
Threat	The screenshot of web pages and application logs and checks can be exposed to other chrome extensions. This information can be exploited by other extensions or XML injections to parse malicious information directly to OS
Best Practice	a. Ensure that no sensitive information is cached b. Logs are encrypted and can be read and write by the application itself c. Ensure all input data is simplified and no complex XML data is in the input path d. Using JSON for string inputs

7.2.5 Chrome Extension Performance Testing

ID	MLDL005
Type	Performance Testing
Functionality	CPU, RAM, and Storage tests
Framework	Selenium or Jasmine
Testing areas	a. Load Testing b. Stress Testing c. Volume Testing
Mode	Automated testing using Selenium or Jasmine
System	Core i5 Generation 5^{th} (Latest is Generation 10^{th})
Memory	8 GB 1600 MHz DDR3
Storage	5600 RPM HDD
Test Category	Legacy System test
Expected Performance	• CPU consumption IDLE: <1% • CPU Consumption Peak: <= 40% (± 10%) ∀ (max) 30 seconds • RAM Consumption IDLE: ~ 140 MB (max) || ~ 60 MB (min) • RAM Consumption Peak: ~ 210 MB (max) (± 40 MB) • HDD 5600 RPM Storage writes (for logs): ~10 MB ∀ (max) 60 seconds
Actual Performance	TBD

7.2.6 Chrome Extension Smoke Test

ID	MLDL006
Type	Acceptance Test
Functionality	Application Testing
Framework	Manual Testing
Testing areas	a. Web page screenshot capture b. Model output generation c. Message Display on Screen d. Logs are written
Mode	Manual testing
Test Category	Final Acceptance test
Expected Performance	a. The application can take a screenshot b. The application can generate output c. The application can display output onscreen accurately d. Logs are written by the application at the backend
Actual Performance	TBD

Chapter 8. Implementation Plan and Progress

8.1 Implementation Plan

Table 1: Implementation plan

Milestone Description	Category	Start	No. Days	Status
1. Handshake and Collaboration Setup				Completed
1.1. Project Purpose freezing	Goal	8/9/2020	2	Completed
1.2. Team Formation	Milestone	7/10/2020	5	Completed
1.3. Slack Collaboration Channel setup	Low Risk	8/29/2020	1	Completed
1.4. Trello Board for Task management	Med Risk	8/6/2020	6	Completed
1.5. Integration of Slack and Trello Collaboration Platform	On Track	8/29/2020	7	Completed
1.6. MS One Drive and Google Drive setup	Low Risk	8/12/2020	2	Completed
1.7. Finalizing Advisor and Advisor Acceptance	Goal	7/13/2020	5	Completed
1.8. Idea Finalization	High Risk	7/18/2020	15	Completed
2. Project Kick-off				Completed
2.1. Literature Research	On Track	8/15/2020	37	Completed
2.2. Problem Identification	High Risk	8/11/2020	20	Completed
2.3. Project Scoping	Low Risk	8/30/2020	11	Completed
2.4. Abstract Draft	Goal	8/30/2020	4	Completed
2.5. Abstract Submission	Milestone	9/3/2020	1	Completed
2.6. Project Group Start-up Presentation	Milestone	9/7/2020	1	Completed
3. POC (Proof of concept) and Project prototyping				Completed
3.1. Data Preparation for Logo validation	Milestone	8/24/2020	4	Completed
3.1.1. Identify target brand label	On Track	8/27/2020	30	Completed
3.1.2. Manually annotate using HyperLabel	High Risk	9/11/2020	15	Completed
3.2. Creating a Demo validation Model for the dataset	On Track	9/7/2020	20	Completed
3.2.1. Create a model based on YOLO	Low Risk	9/9/2020	21	Completed
3.2.2. Train model based on sample dataset to validate the approach	Med Risk	9/9/2020	17	Completed
3.3. Validate and increase the scope of Data Validation	High Risk	9/9/2020	22	Completed
3.4. Select and Freeze the state-of-art-model for Logo validation	Low Risk	9/9/2020	22	Completed
3.5. identify and select the state-of-art-model for URL validation	On Track	9/21/2020	22	WIP

Milestone Description	Category	Start	No. Days	Status
3.6. Select and Freeze the state-of-the-art model for URL Validation	Low Risk	9/21/2020	22	WIP
3.7. Build and Test the ensemble model	High Risk	9/21/2020	22	WIP
3.8. Freeze the ensemble model approach and develop for scaling	Milestone	9/25/2020	22	
4. Train and Build the DL Model				
4.1. Freeze and Store the Dataset in a standard structured manner	High Risk	10/17/2020	15	
4.2. Create and develop the DL model for Logo Validation	On Track	10/27/2020	15	
4.3. Create and Develop the DL model for URL Validation	Med Risk	10/27/2020	15	
4.4. Create a Deep ensemble model	Low Risk	11/11/2020	10	
4.6. Finetune the ensemble model	Low Risk	11/12/2020	11	
4.5. Train and Save ensemble model weight	High Risk	11/25/2020	1	
5. Develop the packaged application	High Risk	12/22/2020	30	
6. Testing the packaged application	High Risk	1/4/2021	30	
7. Deploy the packaged application	Med Risk	1/31/2021	30	
8. Benchmark and publish performance review	Low Risk	2/13/2021	35	
9. Review of Optional features	Low Risk	2/28/2021	2	
10. Project Closure and Handover	Low Risk	3/12/2021	30	

Note: Milestone 5 to Milestone 9 subtasks will be shared as part of workbook 3

8.2 CURRENT PROGRESS

The project is on track and we are currently on milestone 3. A POC for a manually annotated dataset based on YOLO state-of-the-art model has been completed. We are currently in process of fine-tuning the model

The current ongoing POC is for URL validation. The dataset for phishing URL is being mined and fetched from Archive.org and Phishtank. In parallel, we are working on POC of character recognition CNN and string extraction for URL to be stored as a string. We are also working on an LSTM POC model and training it on mined phishing URL strings to learn the phishing website patterns.

Chapter 9. PROJECT SCHEDULE

9.1 PROJECT TIMELINE SNAPSHOT

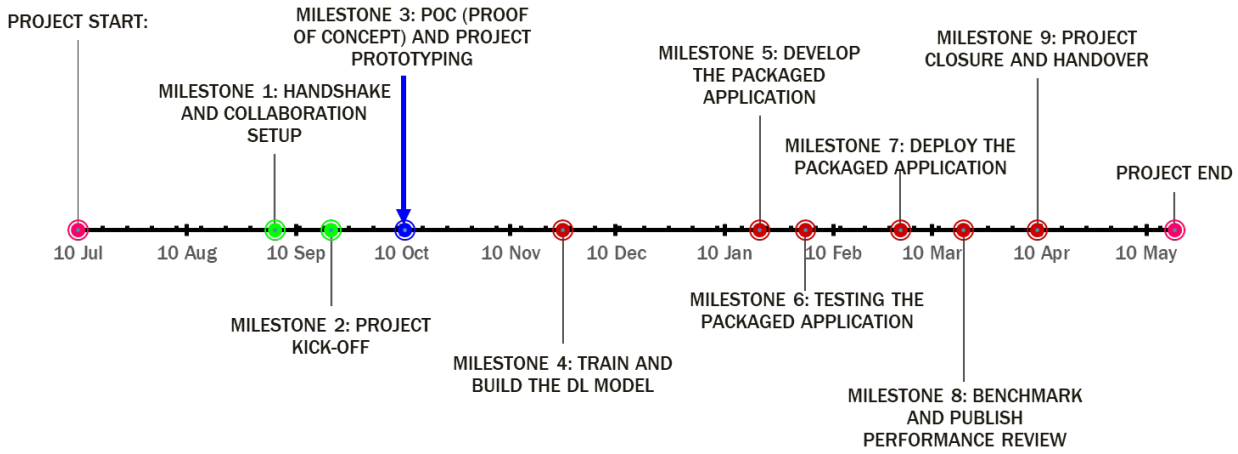

Figure 8: Project timeline snapshot

9.2 GANTT CHART

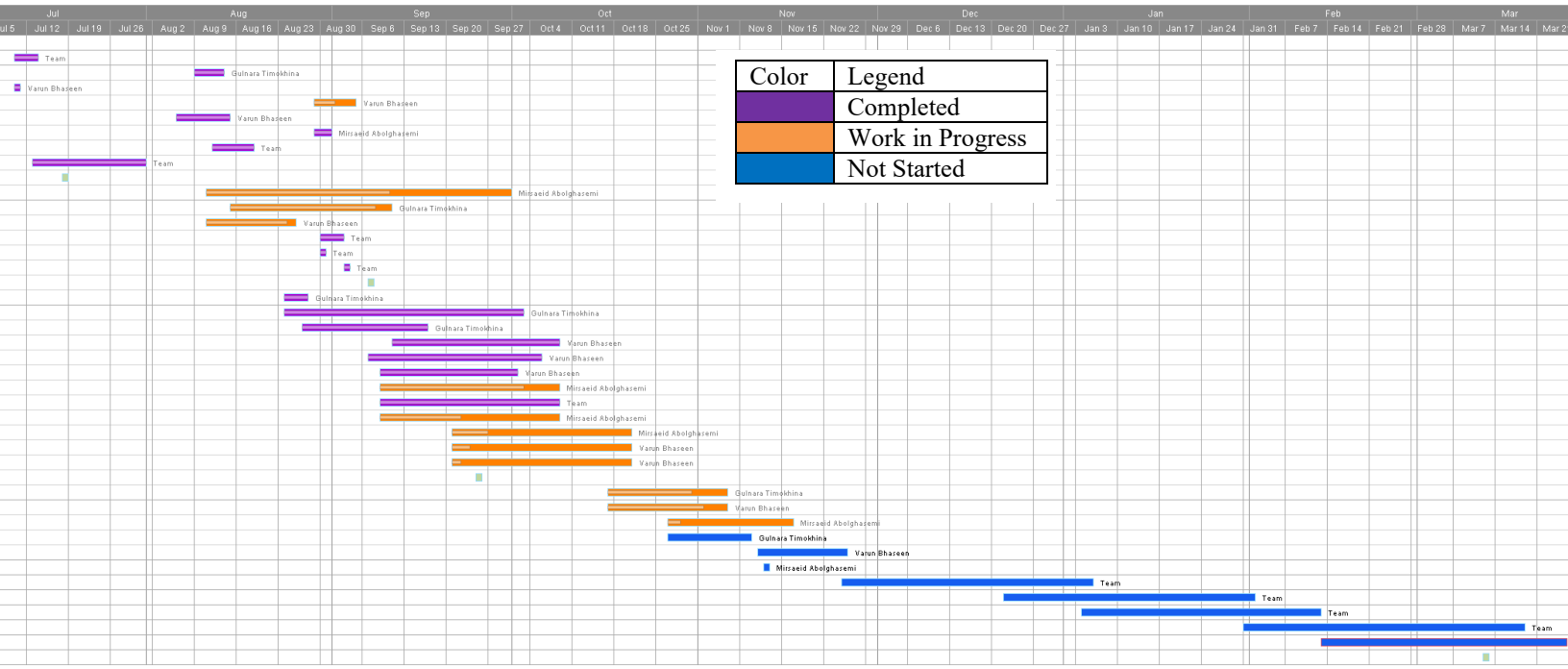

Figure 9: Gantt Chart

9.3 TEAM MEMBER TASKS AND EFFORTS DISTRIBUTION

Table 2: Tasks distribution for the team members

Resource and Task Ownership (from Pivot Table)	Days
Gulnara Timokhina	**129**
1.2. Team Formation	5
2.2. Problem Identification	20
3.1. Data Preparation for Logo validation	4
3.1.1. Identify target brand label	30
3.1.2. Manually annotate using HyperLabel	25
4.1. Freeze and Store the Dataset in a standard structured manner	30
4.4. Create a Deep ensemble model	15
Mirsaeid Abolghasemi	**120**
1.6. MS One Drive and Google Drive setup	2
2.1. Literature Research	37
3.3. Validate and increase the scope of Data Validation	22
3.5. identify and select the state-of-art-model for URL validation	22
3.6. Select and Freeze the state-of-the-art model for URL Validation	22
4.3. Create and Develop the DL model for URL Validation	15
4.5. Train and Save ensemble model weight	1
Team	**205**
1.1. Project Purpose freezing	2
1.7. Finalizing Advisor and Advisor Acceptance	5
1.8. Idea Finalization	15
2.4. Abstract Draft	4
2.5. Abstract Submission	1
2.6. Project Group Start-up Presentation	1
3.4. Select and Freeze the state-of-art-model for Logo validation	22
5. Develop the packaged application	30
6. Testing the packaged application	30
7. Deploy the packaged application	30
8. Benchmark and publish performance review	35
9. Project Closure and Handover	30
Varun Bhaseen	**120**
1.3. Slack Collaboration Channel setup	1
1.4. Trello Board for Task management	6
1.5. Integration of Slack and Trello Collaboration Platform	7
2.3. Project Scoping	11
3.2. Creating a Demo validation Model for a dataset	20

Resource and Task Ownership (from Pivot Table)	Days
3.2.1. Create a model based on YOLO	15
3.2.2. Train model based on sample dataset to validate the approach	12
3.7. Build and Test the ensemble model	15
3.8. Freeze the ensemble model approach and develop for scaling	18
4.2. Create and develop the DL model for Logo Validation	15
4.6. Finetune the ensemble model	6
Cumulative Total Resource Utilization	**578**

--- End of Document ---