A Mobile Application to avoid Online Deception

FYP21013
Lai Filbert Gibson (3035568013) (Author)
Lai Wai Yuet (3035568453)

Supervisor: Dr. Chim, Tat Wing

18th April 2022
Abstract

In the 21st century, information technology is widely used around the globe. Meanwhile, cyber security is becoming more prominent for all internet users since online deceptions might lead to pecuniary losses or leakage of personal information. Therefore, in order to avoid online deception and raise public awareness of online fraud or cyber pitfalls, this report introduces a project that aims to deliver a mobile application, ‘Online Guard’, to avoid online deception by providing a safe browsing environment and promoting cyber security. Unlike other similar mobile applications available in the market that require payment to unlock premium functions, Online Guard would be free to download with all functions provided. In the project, all proposed functions have been implemented. The application would provide three main functions. Firstly, the application would block reported phishing websites by leveraging technologies like Google Safe Browsing Service and application programming interfaces to large databases. Furthermore, unreported phishing websites would also be detected and blocked by real-time threat analysis. Lastly, the mobile application would introduce tips and latest news regarding cyber security, which serves to raise public awareness of cyber pitfall, with a short quiz to examine the users.
Acknowledgement

I would like to express my great appreciation to the project supervisor, Dr. Chim, Tat Wing, for his valuable suggestions and guidance on this project. I would also like to extend our thanks to the Department of Computer Science, The University of Hong Kong, for providing guidance and the required resources for the project.
# Table of Contents

1 Introduction .................................................................................................................. 11
   1.1 Background ............................................................................................................. 11
      1.1.1 Online Deceptions in the World ................................................................. 11
      1.1.2 Online Deceptions in Hong Kong ............................................................... 12
   1.2 Objectives ............................................................................................................... 12
   1.3 Motivations .......................................................................................................... 13
   1.4 Literature Review ................................................................................................. 14
   1.5 Contribution of Members .................................................................................... 14
   1.6 Scope .................................................................................................................... 14
   1.7 Outline of Report ................................................................................................. 15
1 Methodology ................................................................................................................. 16
   1.1 System Design ...................................................................................................... 16
   2.2 Main Function 1: Blocking Reported Phishing Websites ......................................... 18
      2.2.1 User-Defined Blocklist ............................................................................... 20
      2.2.2 Google Safe Browsing Service ..................................................................... 21
      2.2.3 APIs for Accessing Large Databases .......................................................... 22
   2.3 Main Function 2: Scanning for Suspicious Websites ............................................... 24
      2.3.1 APIs for Analysing URL ............................................................................ 27
      2.3.2 AI model ..................................................................................................... 28
   2.4 Main Function 3: Information Sharing regarding Cyber Security ............................ 30
      2.4.1 Latest Cyber Security News ........................................................................ 30
      2.4.2 Cyber Security Practices to avoid Online Deceptions ................................. 32
3 AI Model ....................................................................................................................... 33
   3.1 Dataset .................................................................................................................. 33
      3.1.1 Training and Validation Dataset .................................................................. 33
      3.1.2 Testing Dataset ............................................................................................ 34
   3.2 Model Selection ..................................................................................................... 34
   3.3 Data Pre-processing ............................................................................................... 35
   3.4 Model Evaluation ................................................................................................ 35
   3.5 Further Improvements .......................................................................................... 36
      3.5.1 Feature Selection ......................................................................................... 36
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.5.2</td>
<td>Hyperparameter Optimization</td>
<td>38</td>
</tr>
<tr>
<td>3.6</td>
<td>Final Performance</td>
<td>38</td>
</tr>
<tr>
<td>4</td>
<td>App Design</td>
<td>39</td>
</tr>
<tr>
<td>4.1</td>
<td>App UI Design and Logic Flow</td>
<td>39</td>
</tr>
<tr>
<td>4.1.1</td>
<td>Browser Page</td>
<td>40</td>
</tr>
<tr>
<td>4.1.2</td>
<td>Blocklist Page</td>
<td>45</td>
</tr>
<tr>
<td>4.1.3</td>
<td>News Page</td>
<td>49</td>
</tr>
<tr>
<td>4.1.4</td>
<td>Info Page</td>
<td>51</td>
</tr>
<tr>
<td>4.1.5</td>
<td>Quiz Page</td>
<td>51</td>
</tr>
<tr>
<td>4.1.6</td>
<td>Other Pages</td>
<td>54</td>
</tr>
<tr>
<td>4.2</td>
<td>App UX Design</td>
<td>56</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Responsiveness</td>
<td>57</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Self-Descriptive</td>
<td>58</td>
</tr>
<tr>
<td>4.2.3</td>
<td>Customization</td>
<td>58</td>
</tr>
<tr>
<td>4.2.4</td>
<td>Efficiency</td>
<td>59</td>
</tr>
<tr>
<td>4.2.5</td>
<td>Consistency</td>
<td>59</td>
</tr>
<tr>
<td>5</td>
<td>Product Evaluation</td>
<td>61</td>
</tr>
<tr>
<td>5.1</td>
<td>Methodology</td>
<td>61</td>
</tr>
<tr>
<td>5.2</td>
<td>Result</td>
<td>62</td>
</tr>
<tr>
<td>5.3</td>
<td>Analysis</td>
<td>64</td>
</tr>
<tr>
<td>6</td>
<td>Project Evaluation</td>
<td>65</td>
</tr>
<tr>
<td>6.1</td>
<td>Limitation</td>
<td>65</td>
</tr>
<tr>
<td>6.2</td>
<td>Future Development</td>
<td>65</td>
</tr>
<tr>
<td>7</td>
<td>Conclusion</td>
<td>67</td>
</tr>
<tr>
<td>References</td>
<td></td>
<td>68</td>
</tr>
<tr>
<td>Appendices</td>
<td></td>
<td>70</td>
</tr>
<tr>
<td>Appendix I: URLs of unsafe websites used in testing in section 5</td>
<td>70</td>
<td></td>
</tr>
<tr>
<td>Appendix II: Installation Guide of Online Guard</td>
<td>74</td>
<td></td>
</tr>
</tbody>
</table>
List of Figures

Figure 1.1: The distribution of deception cases in Hong Kong, 2021 .......................... 12
Figure 1.2: Pricing of Blocksite in Hong Kong .......................................................... 13
Figure 2.1: Overall System Design ............................................................................. 16
Figure 2.2: Screenshot of browser page when users go to a website with domain name exists in the user-defined blocklist .......................................................... 19
Figure 2.3: Settings that allow users to decide the behaviour of the app when a particular technology detects a website as a phishing or malicious website ........................................................................................................................................ 19
Figure 2.4: Screenshot of the blocklist page ................................................................. 20
Figure 2.5: Deceptive site detected by Google safe browsing service ......................... 22
Figure 2.6: Website detected by ‘URL Reputation API’ from URLVoid ........................ 23
Figure 2.7: Website detected by ‘Malicious URL Scanner API’ from IPQualityScore .... 24
Figure 2.8: System mechanism to fulfil main function 1 and main function 2 ............ 25
Figure 2.9: Examples of the security checks conducted by URLVoid ......................... 27
Figure 2.10: Screenshot of a website blocked by AI model ......................................... 28
Figure 2.11: Snack bar shown for users to report a website ....................................... 29
Figure 2.12: Pop-up window for users to report a website ......................................... 29
Figure 2.13: System mechanism of loading and fetching the latest cyber security news .......................................................... 31
Figure 3.1: Separation of the URL string ..................................................................... 34
Figure 3.2: Testing of XGBoost Model ....................................................................... 35
Figure 3.3: 15 most important features measured in MDI ......................................... 36
Figure 3.4: new analysis of the 15 most important features measured in MDI ........... 37
Figure 3.5: Performance of the final model ................................................................. 38
Figure 4.1: General user interface of the app ............................................................... 39
Figure 4.2: User interface of the browser page......................................................... 40
Figure 4.3: Choosing Online Guard to load the URL outside Online Guard.......... 41
Figure 4.4: Error handling in loading websites......................................................... 42
Figure 4.5: Screenshot sequences of loading a website blocked and reporting it..... 43
Figure 4.6: Blocking the website directly with a toast instead of showing a pop-up
window................................................................................................................. 43
Figure 4.7: Screenshot sequences of adding the current website to the blocklist.... 44
Figure 4.8: Reminder when users add invalid URL to the blocklist............................ 45
Figure 4.9: User interface of the blocklist page......................................................... 46
Figure 4.10: Adding item to the blocklist................................................................. 47
Figure 4.11: Deleting item in the blocklist............................................................... 47
Figure 4.12: Showing warning for invalid input in adding item to blocklist........... 48
Figure 4.13: User interface of News page............................................................... 49
Figure 4.14: reading a cyber security news in the news page............................... 50
Figure 4.15: reminding users in the News page when there is no internet connection.. 50
Figure 4.16: User interface of the Info page............................................................ 51
Figure 4.17: User interface of the Quiz page.......................................................... 52
Figure 4.18: Screenshot of the quiz page after submission..................................... 53
Figure 4.19: Reminding in the Quiz page when not all questions are answered....... 53
Figure 4.20: User interface of the Setting page....................................................... 54
Figure 4.21: Selecting Online Guard as the default browser app............................. 55
Figure 4.22: Toast generated for notifying users that the changes have been saved.... 55
Figure 4.23: User interface of the Help page.......................................................... 56
Figure 4.24: using toast to notify users when something has been changed by the users… 57
Figure 4.25: Bottom navigation bar when switching from Browser page to News page.... 57
Figure 4.26: Self-descriptive buttons in the top action bar……………………………………58
Figure 4.27: Hints on the input text boxes………………………………………………………….58
Figure 4.28: Possible Settings for the users to make decisions…………………………59
Figure 4.29: consistent UI design for the title of pages………………………………………..60
List of Tables

Table 2.1: Summary of the ability of each technology implemented in fulfilling main function 1 and main function 2................................................................. 26

Table 3.1: performances of different models during model selection.......................... 34

Table 5.1: Testing result in product evaluation................................................................. 62
# Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AI</td>
<td>Artificial Intelligence</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>APP</td>
<td>Application</td>
</tr>
<tr>
<td>DB</td>
<td>Database</td>
</tr>
<tr>
<td>HKD</td>
<td>Hong Kong Dollar</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol</td>
</tr>
<tr>
<td>IDE</td>
<td>Integrated Development Environment</td>
</tr>
<tr>
<td>IPQS</td>
<td>IPQualityScore</td>
</tr>
<tr>
<td>JSON</td>
<td>JavaScript Object Notation</td>
</tr>
<tr>
<td>MC</td>
<td>Multiple Choices</td>
</tr>
<tr>
<td>MDI</td>
<td>Mean Decrease in Impurity</td>
</tr>
<tr>
<td>UI</td>
<td>User Interface</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
<tr>
<td>USD</td>
<td>United States Dollar</td>
</tr>
<tr>
<td>UX</td>
<td>User Experience</td>
</tr>
<tr>
<td>WEB</td>
<td>Website</td>
</tr>
<tr>
<td>XGBoost</td>
<td>Extreme Gradient Boosting</td>
</tr>
</tbody>
</table>
1 Introduction

In the 21st century, information technology is widely used around the globe. Meanwhile, there is an increasing trend of online deceptions. This section would first introduce the situation of online deception in the world and in Hong Kong respectively, followed by the objectives, motivations and literature review of the project. The section ends by concluding the contribution of members in the project as well as listing the scope and outline of the report.

1.1 Background

Online deception is defined as fraudulent practices used on the internet to mislead individual victims [1]. As technology advances, the number of internet users has surged in the recent decade. According to the research conducted by Statista, there were around 4.66 billion active internet users worldwide in January 2021 [2]. The increase in the number of internet users acts as a catalyst for the development in online deceptions.

1.1.1 Online Deceptions in the World

In 2020, cybercrime is expected to cost the world $6 trillion USD, while the number is predicted to grow by 15 percent per year [3]. The cost of cybercrime in 2025 is predicted to rise to $10.5 trillion USD [3], which is partly contributed by online deceptions. According to the research conducted by Verizon, 36% of data breaches in 2021 involved phishing while the lack of awareness is considered as the major contributing factor [4]. If all cybercrimes are assumed to involve data breaches, the cost of online deception worldwide would roughly be $2 trillion USD in 2021.

For online deceptions worldwide, it is difficult to get the precise figure in the number of fraud cases. The situation in United States would be considered in this section instead. In 2020, there were 2.2 million reported cases regarding fraud complaints in United States [5]. Among the 2.2 million fraud cases, 34% of them involved pecuniary losses which costed $3.3 billion USD in total [5].
1.1.2 Online Deceptions in Hong Kong

In Hong Kong, online fraud has caused millions of losses. In 2021, there were 19,249 deception cases recorded which is increased by 24%, from 15,553 cases in 2020 [6]. Moreover, over 70% of deception cases reported in 2021 happened online [6]. Figure 1.1 shows the distribution of deception cases in 2021 where investment fraud and online employment fraud might involve phishing.

![Figure 1.1: The distribution of deception cases in Hong Kong, 2021](image)

1.2 Objectives

There are two main objectives behind the project. Firstly, the project aims to share the latest news and tips regarding cyber security with the public, in order to raise their awareness of online fraud and cyber pitfalls. Moreover, a short quiz is presented in the mobile application for knowledge consolidation.

Secondly, the project targets at providing a safe browsing experience for the public, by providing a customized browser in the application to block phishing and malicious websites. For reported phishing websites, APIs will be used for querying large databases in two platforms – URLVoid and IPQualityScore. Returning of a matching result would trigger a blockage of access to the corresponding malicious websites. Moreover, Google safe browsing
service is enabled in the browser to query the URL in the updated lists of unsafe web resources that maintained by Google. Furthermore, users could add any URL to a user-defined blocklist so as to block unfavourable web resources.

### 1.3 Motivations

The project is motivated by the arising cyber security issues in the world. Although there are similar applications in the market, like a mobile application ‘BlockSite’ in Google Play Sore [7], users are required to pay subscription fees in order to unlock premium functions [8]. Figure 1.2 shows the pricing of BlockSite in Hong Kong that require users to pay monthly subscription fees of $78 HKD or a one-time payment of $508 HKD for unlocking the premium functions like an unlimited blocking list. In order to promote cyber security without limitation, the project aims at delivering a mobile application which would be free to download with all functions provided.

![Figure 1.2: Pricing of BlockSite in Hong Kong](image_url)
1.4 Literature Review

A data article, ‘Datasets for phishing websites detection’, was reviewed before the construction of the AI model. The paper presents dataset for training the XGBoost model in the project which contains 58,645 legitimate websites and 88,647 phishing websites [9]. The paper mainly introduces the dataset, like the value and description of the data. Furthermore, the paper introduces an algorithm in pseudocode for the feature extraction process before feeding the features to the AI model for prediction.

1.5 Contribution of Members

Two members were responsible for the project, including Lai Filbert Gibson (UID: 3035568013) and Lai Wai Yuet (UID: 3035568453).

Lai Filbert Gibson was responsible for the development, testing and finalizing of the mobile application developed in Android Studio. Also, he was responsible for the development of the website crawling program in Python for fetching the latest cyber security news.

Lai Wai Yuet was responsible for the implementation, data engineering and improvement of the XGBoost model in the development server. Moreover, he was responsible for the development in the development server, such as the data extraction programs dedicated for the XGBoost model and the Flask server for receiving and sending HTTP requests.

Both members were responsible for the decisions of directions in development of mobile application, AI model and development server.

1.6 Scope

The project is divided into three parts based on the three main functions:
1. Blocking reported phishing or malicious websites using user-defined blocklist, Google Safe Browsing Service and APIs to large databases.
2. Scanning and blocking unreported phishing or malicious websites using artificial intelligence and APIs for real-time threat analysis.
3. Information sharing regarding cyber security, such as sharing of the latest news using website crawling program.

The project aims at delivering a cyber security solution in the form of a mobile application, developed using Android Studio as the IDE.

1.7 Outline of Report

The remaining of the report outlines the system design and the methodologies of implementing the three main functions in section 2, followed by discussion on the design of the AI model and mobile app in section 3 and 4 respectively. Before drawing the conclusion, evaluation on product and project would be conducted in section 5 and 6 respectively.
1 Methodology

This section introduces the overall system design, followed by the methodologies of implementing the three main functions.

1.1 System Design

Figure 2.1 shows the overall system design which is constructed by three parts – mobile application, development server and malicious URL detectors.

The mobile app contains different pages for different uses, including:

- **Browser page**: provides a customized browser with safe browsing experience, by using different technologies to block reported and unreported phishing or malicious websites.
- **Blocklist page**: allows users to view and edit a locally stored blocklist which contains the domain name and URL that the users want to block in the browser.
• **News page**: provides the latest cyber security news in a list while users can select any title in the list to load the corresponding URL for reading the news.

• **Info page**: provides useful cyber security practices to avoid online deceptions.

• **Quiz page**: examines the users about the knowledge provided in info page.

• **Other pages**:
  - **Setting page**: allow users to edit their preferences, such as enabling or disabling specific technology to block undesired websites.
  - **Help page**: introduces the application, such as the three main functions provided and some tips to use the app.

The development server consists of four parts, including:

• **HTTP Server**: communicates with the mobile app by sending or receiving HTTP requests, such as receiving URLs from mobile app and feeding it to the AI model for prediction.

• **AI model**: XGBoost model to detect potential phishing websites by feeding the URL to the feature extraction programs for extracting features from an URL and feed the features to the model for prediction.

• **Website Crawling Program**: fetch the latest cyber security news from different sources on the internet, including title, URL and source. After fetching, the data would be stored into a local database.

• **MongoDB database**: stores data like:
  - **Cyber security news**: including title, URL and source of the news.
  - **User Feedback**: feedbacks from users about reporting an URL as safe or phishing websites which helps to improve the AI model continuously.

The malicious URL Detector includes:

• **Google Safe Browsing Service**: constantly updated lists of unsafe web resources that maintained by Google and used in the Google Chrome browser.

• **URLVoid, IPQualityScore**: platforms that provide the service of real-time threat analysis and malicious URL checkers that check an URL with their maintained large-scaled databases.
The mobile application will provide three main functions, including (1) Blocking reported phishing websites, (2) Scanning for suspicious websites, (3) Information sharing regarding cyber security. For each of them, the methodologies and justifications would be discussed in the following sections.

### 2.2 Main Function 1: Blocking Reported Phishing Websites

The first main function provided by Online Guard would be blocking the access to reported phishing websites, which means the websites already known as phishing or malicious, and have been reported locally or online. This function is achieved by using three technologies – (1) a user-defined blocklist, (2) Google Safe Browsing Service, (3) Application programming interfaces to large databases.

If the website in the browser is detected as a phishing or malicious website, the access to the website will be blocked with a pop-up window generated for showing the details, like the screenshot shown in figure 2.2. Users can choose to cancel the browsing by pressing the ‘Cancel’ button on the pop-up window. Alternatively, users can choose to browse the website by pressing the ‘Still Go’ button. Moreover, users can choose between blocking the website directly or showing a pop-up window in the setting page. Figure 2.3 shows the related part in the setting page that allow users to decide the behaviour of the app when a particular technology detects a website as a phishing or malicious website.
Figure 2.2: Screenshot of browser page when users go to a website with domain name exists in the user-defined blocklist

Figure 2.3: Settings that allow users to decide the behaviour of the app when a particular technology detects a website as a phishing or malicious website
2.2.1 User-Defined Blocklist

In order to provide customizability to the users, Online Guard provides a user-defined blocklist, which allows users to maintain a list of URLs or domain name to block in the browser page of the mobile app.

Users can view and edit the blocklist in the blocklist page. For example, figure 2.4 shows a screenshot of the blocklist page that a domain name ‘youtube’ and an URL ‘https://www.cuhk.edu.hk/English/index.html’ exist in the blocklist.

Figure 2.4: Screenshot of the blocklist page

If a user is going to browse a website that exists in the blocklist, the default behaviour of the app would be blocking the website and display a pop-up window to show the reason of blocking. For example, if the user has added ‘youtube’ into the blocklist as a domain name to block, he will not be able to browse the youtube website directly but encounter a blockage, like the screenshot shown in figure 2.2.
There are two choices available for locating the blocklist. The first choice is to place the blocklist in the development server, that means that the blocklist is a shared blocklist and any users could update it directly so that every copy of the app would block the same set of websites according to the shared blocklist. However, it is concerned that some users may mistakenly or maliciously add safe websites to the blocklist so other users would not be able to browse the websites also. For example, if a user mistakenly added ‘youtube’ to the shared blocklist, all other users would not be able to browse the youtube website also due to the mistake. Hence, the second choice is adopted – placing the blocklist locally so that every blocklist will only take effect in the browser of the local app but not affecting other users.

More about the blocklist page would be discussed in section 4.1.2, such as the UI design and logic flow.

2.2.2 Google Safe Browsing Service

The user-defined blocklist only serves to block users’ unfavourable websites. Hence, it is not effective in blocking unsafe web resources, such as phishing and malicious websites. For the unsafe websites, Google safe browsing service and APIs would be leveraged for accessing large-scaled databases of unsafe websites.

Google safe browsing service is a service provided by Google that allow applications to check URLs against a constantly updated list of web resources maintained by Google [10]. The list contains a large number of unsafe web resources, such as phishing, deceptive and malicious websites. In 2022, the most popular browser is Google Chrome, with around 65% of market share [11], has used Google safe browsing service as one of the security measures [12].

In the browser page of Online Guard, the WebView component is configured to use Google safe browsing service for internet surfing. If the website is detected unsafe by Google safe browsing service, it will be blocked with a red background to warn the users, like the screenshot shown in figure 2.5.
2.2.3 APIs for Accessing Large Databases

Apart from Google safe browsing service, application programming interfaces to the large databases of two platforms are also leveraged to block reported phishing websites – URLVoid and IPQualityScore.

Firstly, Online Guard uses the ‘URL Reputation API’ from the platform ‘URLVoid’ to check the reputation of URL from more than 25 engines, such as OpenPhish, PhishTank and Phishstats [13]. If a website is detected as phishing website by the ‘URL Reputation API’, it will be blocked in the app with a pop-up window showing (1) the reason of blocking, (2) the risk score calculated by URLVoid, (3) the number of engines that returned the website as a phishing website, (4) title of the website which helps users to decide whether to proceed or not. Figure 2.6 shows the screenshot of the browser page when the website is detected as phishing by URLVoid.
Meanwhile, another API, ‘Malicious URL Scanner API’ from IPQualityScore, is used also to identify phishing, malware URL and parked domains over 20 data points [14]. If a website is detected as phishing website by the ‘Malicious URL Scanner API’, it will be blocked in the app with a pop-up window showing (1) the reason of blocking, (2) the risk score calculated by IPQualityScore, (3) reasons of blocking provided by the API, (4) website category provided by the API which would help users to determine whether to proceed or not. Figure 2.7 shows the screenshot of the browser page when the website is detected as phishing by IPQualityScore.
The two platforms are also able to detect unreported phishing websites which would be discussed in section 2.3.1. Also, the UI design and the logic flow of the browser page would be discussed in section 4.1.1.

### 2.3 Main Function 2: Scanning for Suspicious Websites

Apart from reported phishing websites, unreported phishing websites should also be detected as well to enhance the security level, which means that the websites are not known as phishing and have not been reported as unsafe.
For unreported phishing or malicious websites, the two APIs aforementioned are used to perform real-time threat analysis. In addition, a machine learning model was constructed in the development server to scan for suspicious websites.

Figure 2.8 shows the system mechanism in order to provide a safe internet environment. When a user browses a website in the browser page, the URL of the website will be checked against:

- A user-defined blocklist stored locally in the app.
- Malicious URL detectors, including Google safe browsing service, URLVoid API and IPQS API.
- AI model in the development server that the mobile app would send the URL to the AI model for prediction through the HTTP Server by GET request.

Table 2.1 shows a summary of the ability of each technology implemented in fulfilling main function 1 and main function 2.
<table>
<thead>
<tr>
<th>Technology</th>
<th>Block reported phishing/malicious websites? (Main Function 1)</th>
<th>Block unreported phishing/malicious websites? (Main Function 2)</th>
<th>Stored in / Provided by?</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>User-defined Blocklist</strong></td>
<td>✅ By checking the domain name and URL stored in the blocklist</td>
<td></td>
<td>Mobile Application</td>
</tr>
<tr>
<td><strong>Google Safe Browsing Service</strong></td>
<td>✅ By checking against a constantly updated list of unsafe web resources maintained by Google</td>
<td></td>
<td>Google</td>
</tr>
<tr>
<td><strong>URL Reputation API</strong></td>
<td>✅ Checking the reputation of the URL from more than 25 engines</td>
<td>✅ Uses thousands of smart internal rules to detect potential malicious URLs</td>
<td>URLVoid</td>
</tr>
<tr>
<td><strong>Malicious URL Scanner API</strong></td>
<td>✅ Identify phishing, malware URL and parked domains over 20 data points</td>
<td>✅ Uses a live threat intelligence to detect zero-day phishing links</td>
<td>IPQualityScore</td>
</tr>
<tr>
<td><strong>XGBoost AI model</strong></td>
<td></td>
<td>✅ Using the XGBoost model to do feature extraction and prediction on the URL</td>
<td>Development Server</td>
</tr>
</tbody>
</table>

Table 2.1: Summary of the ability of each technology implemented in fulfilling main function 1 and main function 2
2.3.1 APIs for Analysing URL

The two APIs mentioned in section 2.2.3 would be used for analysing zero-day phishing links also.

For ‘URL Reputation API’ from URLVoid, it uses thousands of smart internal rules to detect potential malicious URLs [13]. Deep analysis on the URL is performed, such as URL content, URL pattern, domain name, HTTP headers and top-level domain. Hence, smart rules would be used to detect suspicious or malicious websites, such as detecting suspended websites or suspicious domain name. After that, a risk score would be calculated according to different security checks, which ranges from 0 (safe) to 100 (dangerous). Figure 2.9 shows some examples of the security checks conducted by URLVoid.

```
"security_checks":{
  "is_host_an_ipv4":false,
  "is_suspicious_url_pattern":false,
  "is_suspended_page":true,
  "is_most_abused_tld":false,
  "is_uncommon_clickable_url":true,
  "is_phishing_heuristic":true,
  "is_suspicious_content":false,
  "is_empty_page_title":false,
  "is_domain_blacklisted":true,
  "is_suspicious_domain":false,
  "is_sinkholed_domain":false,
  "is_defaced_heuristic":false,
  "is_risky_geo_location":false,
```

Figure 2.9: Examples of the security checks conducted by URLVoid

For ‘Malicious URL Scanner API’ from IPQualityScore, it uses a live threat intelligence to detect zero-day phishing links in real time [14]. The live threat intelligence combines the results from different sources, such as Fraud Fusion Network which is a networked approach to verify fraudsters and cybercriminals [15]. Furthermore, A machine learning model is used to detect phishing attacks [16]. Risk score will be provided after real-time risk analysis. Also, the intelligence keeps improving by feeding threat data from their clients, such as Fortune 500s.
2.3.2 AI model

Apart from APIs, an AI model has also been developed in the development server to scan for unreported phishing or malicious websites. When the users browse a website, Online Guard would send the URL to the HTTP server in the development server by GET request. Then, the feature extraction program would extract the features like the number of domain characters and the domain name lookup response from the URL. Afterwards, the features would be fed to the XGBoost model for prediction. The prediction result would be sent back to the app in JSON format and the app would extract the prediction result from the JSON object. If the AI model has predicted the website as suspicious, the website would be blocked with a pop-up window generated, like the screenshot shown in figure 2.10.

![Figure 2.10: Screenshot of a website blocked by AI model](image)

Furthermore, if the users insist on going to a suspicious website, there would be a snack bar generated for the users to provide feedback. After the users have pressed the ‘Report’ button on the snack bar, there would be a pop-up window that allows users to report the website as a
safe or phishing website, like the screenshots shown on figure 2.11 and figure 2.12. The feedback would be stored into the database and the AI model would be updated in the early morning when the number of feedbacks has reached 100.

Figure 2.11: Snack bar shown for users to report a website

Figure 2.12: Pop-up window for users to report a website
For the database in the development server, there are two choices available – MySQL and MongoDB. MongoDB was selected since it is faster than MySQL in handling a large number of unstructured data [17]. Since the data to be stored in the database have a simple structure, there would be advantages in performance in using MongoDB.

Details of the AI model would be discussed deeply in section 3, such as dataset, feature engineering and performance.

**2.4 Main Function 3: Information Sharing regarding Cyber Security**

The last main function provided by Online Guard would be information sharing regarding cyber security. It is accomplished by sharing the latest cyber security news and some useful cyber security practices with a short MC quiz to examine the users for knowledge consolidation.

**2.4.1 Latest Cyber Security News**

Online Guard provides the latest cyber security news in the news page. Figure 2.13 shows the system mechanism of loading the news in the mobile app and fetching the news in the development server.

When the users load the news page in the mobile app, the application would fetch the title, source and URL of the news articles from the MongoDB database in the development server through the HTTP server by GET request. Afterward, the app would extract the JSON array from the JSON object returned by the server and display the title and the source of news in a list. When the users click a title in the list, the corresponding URL would be loaded for the users to read the article.
In order to collect and provide the latest cyber security news, there is a website crawling program written in Python in the development server which is executed once per day. During running, the program would fetch the title, source and URL of news from different sources and store the data into the MongoDB database. Before writing the data into the database, all records in the databases would be removed to remove older news and ensure that only the latest cyber security news are stored in the database.

![System mechanism of loading and fetching the latest cyber security news](image)

Figure 2.13: System mechanism of loading and fetching the latest cyber security news

For fetching the latest news from the internet, apart from the method above, it could also be done in the mobile device. For example, libraries like Jsoup could be used for the website crawling part instead of using the website crawling program in the development server. This idea was abandoned since it requires longer loading time as the time of reading data from the database in the development server is shorter than fetching and processing data from multiple websites.

UI design and logic flow of news page would be discussed in section 4.1.3.
2.4.2 Cyber Security Practices to avoid Online Deceptions

Apart from cyber security news, some useful cyber security practices are also shared in the mobile app, in order to promote cyber security to the public.

10 tips to avoid online deceptions are presented in the info page of the mobile app. After reading, users could attend a short multiple choices quiz in the quiz page for knowledge consolidation. The answer of users would be marked after pressing the ‘submit’ button. The correct answers would be displayed and the score would be calculated and shown as reference.

Screenshots, UI design and logic flow of info and quiz page would be shown in section 4.1.4 and 4.1.5 respectively.
3 AI Model

As mentioned in section 2.3.2, an AI model has been constructed to achieve main function 2 – scanning for suspicious websites. The AI model was located in the development server. When the users browse a website in Online Guard, the URL will be sent to the HTTP server in the development server. Then, the program will pass the URL to the AI model for feature extraction and prediction, and then return the prediction result to the mobile app.

This section discusses the detail of implementing the AI model, such as the model and dataset used, as well as the data pre-processing accomplished.

3.1 Dataset

Two separate datasets were prepared for training and testing respectively.

3.1.1 Training and Validation Dataset

As mentioned in section 1.4, a data article, ‘Datasets for phishing websites detection’, was reviewed before the construction of the AI model [9]. The paper introduces a dataset of URLs which was used in the project. The dataset consists of 58,645 legitimate websites and 88,467 phishing websites. However, in the dataset, the number of phishing websites is larger than the number of legitimate websites which is not the case in real life. Hence, only 30,467 phishing websites were used from the dataset in training so as to match the distribution in real life. The dataset was split into training and validation set in the ratio of 80:20.

For every website, 111 attributes are extracted as features to feed the AI model. Features could be divided into 6 main groups:

1. Features based on the whole URL, such as the number of characters.
2. Features based on domain, such as the URL domain in IP address format.
3. Features based on directory, such as the number of ‘/’ signs.
4. Features based on file, such as the number of ‘_’ signs.
5. Features based on parameters, such as the top-level domain in parameters.
6. Features based on resolving URL and external services, such as time response.

Figure 3.1 shows the mechanism of separation of the URL string into domain, directory, file and parameters.

![URL Separation Diagram](attachment:image.png)

Figure 3.1: Separation of the URL string

### 3.1.2 Testing Dataset

The testing dataset is a separate dataset, which has a different data source. The dataset consists of 200 phishing websites that collected from PhishTank and 290 legitimate websites collected from sources like Alexa. Testing dataset has the same format as the training dataset in order to apply the same extraction algorithm. For shortened and redirecting URLs, their final URLs would be used instead.

### 3.2 Model Selection

Multiple AI models has been built and tested in order to select the best AI model as the final model, including logistic regression, support vector machine as well as extreme gradient boosting. Logistic regression model was selected as the baseline model for performance comparison. Table 3.1 shows the performances of different models on the validation dataset after data pre-processing which would be discussed in the next section.

<table>
<thead>
<tr>
<th>AI Model</th>
<th>Accuracy on the phishing class</th>
<th>Recall on the phishing class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logistic Regression</td>
<td>0.9187</td>
<td>0.923</td>
</tr>
<tr>
<td>Support Vector Machine</td>
<td>0.9281</td>
<td>0.923</td>
</tr>
<tr>
<td>Extreme Gradient Boosting</td>
<td>0.9685</td>
<td>0.956</td>
</tr>
</tbody>
</table>

Table 3.1: performances of different models during model selection
The accuracy and recall of XGBoost model were both the best among the three models, with an accuracy of 96.85% and a recall of 95.6%. Therefore, XGBoost was chosen as the final model.

### 3.3 Data Pre-processing

Two data pre-processing were conducted before training the AI models.

Firstly, there are some columns that have the same value in all data. For instance, the number of slash character in domain is 0 in all URLs. These columns were removed since it provides no value to the AI model.

Secondly, there are some columns that contain continuous values instead of discrete values, such as response time of the website. The data in these columns were normalized into the same range.

### 3.4 Model Evaluation

XGBoost model was tested with the testing dataset. Figure 3.2 presents the testing result, which contains three matrixes, including confusion matrix, precision matrix and recall matrix. The overall accuracy was 67.29% while the recall was 56% which is much slower than the performance in model selection. One possible cause is the data mismatch problem between training dataset and testing dataset.

![Figure 3.2: Testing of XGBoost Model](image-url)
10-fold cross validation was conducted on the XGBoost model afterwards to check if there was an overfitting problem. The mean cross validation score is 96.96 with a standard deviation of 0.0028 which showed no overfitting problem. Therefore, it was confirmed to have a data mismatch problem instead of an overfitting problem.

In order to solve the data mismatch problem, the testing dataset was further split in the ratio of 80:20. The 80% group was moved to the training dataset for further fitting.

### 3.5 Further Improvements

Apart from splitting the testing dataset, two further improvements have also been carried out, including (1) feature selection, (2) hyperparameter optimization.

#### 3.5.1 Feature Selection

Before doing feature selection, an analysis of importance of features was conducted by measuring the mean decrease in impurity (MDI). Figure 3.3 shows the 15 most important features in the analysis. In the figure, the blue bar represents MDI of the feature while the black line presents deviation of the MDIs.

![Figure 3.3: 15 most important features measured in MDI](image)

Figure 3.3: 15 most important features measured in MDI
Two observations could be concluded in the analysis:

1. **Low MDI**: The MDI values of all features are lower than 0.1.
2. **High Deviation of substring attributes**: For the features related to substring, such as number of slash characters, the deviation (black line) is longer than the deviations in other features. A high deviation of MDI implies the problem of overfitting.

In order to solve the problem of low MDI and overfitting, the features with high deviation in MDI were removed, such as the substring attributes. After feature selection, the analysis was reconducted to check the changes of the importance of features. Figure 3.4 shows the new analysis of important of features measured in MDI.

![Feature importances using MDI](image)

Figure 3.4: new analysis of the 15 most important features measured in MDI

After feature selection, the MDI of features increased generally. For example, the highest MDI obtained increased from about 0.08 to 0.22. Furthermore, the deviation of the features decreased significantly and close to the mean value (right-end of the blue bar).
3.5.2 Hyperparameter Optimization

Hyperparameter optimization was conducted using a python library ‘Hyperopt’. The defined search space for tuning includes maximum depth, learning rate and other arguments. Moreover, tree of parzen estimators was used as the searching algorithm. 200 trails of tuning have been conducted in hyperparameter optimization.

3.6 Final Performance

Figure 3.5 shows the performance of the final model. After implementing the three improvements, the accuracy improved from 67.28% to 80.20% while the recall improved from 56% to 74.4%, which means that 74.4% of the phishing websites could be detected successfully by the AI model which is satisfactory.

![Figure 3.5: Performance of the final model](image-url)
4 App Design

This section discusses the design of the mobile application, including design in user interface, logic flow and user experience.

4.1 App UI Design and Logic Flow

The mobile application was developed using Android Studio as the IDE. Figure 4.1 shows the general user interface design of the app which consists of three parts:

1. **Top action bar**: shows the app name and menu buttons, including buttons to the blocklist page, setting page and help page.
2. **Main content**: content of the specific page. For example, it shows the browser when it is in the browser page.
3. **Bottom navigation bar**: allows users to switch between 4 pages, including browser page, news page, info page and quiz page.

![Figure 4.1: General user interface of the app](image-url)
In the following sections, the user interface and the logic flow in different pages would be introduced. Since the top action bar and the bottom navigation bar appear as the same way in different pages, only the user interface in the main content would be discussed.

### 4.1.1 Browser Page

Figure 4.2 shows the UI when the users load the Browser page. A loading panel would be presented before the page is fully loaded. After the page is fully loaded, it will replace the loading panel. The UI of the browser page consists of:

1. **Address Bar**: allow users to enter URL or search keywords using Google search engine. Also, the address bar will be updated when the users go to a new website.
2. **Go Button** (in green colour): load the URL or search keywords in the address bar. Alternatively, users can press the search button on the keyboard for the same function.
3. **Block Button** (in red colour): Add the current URL to the local blocklist.
4. **Browser**: a customized WebView with a customized WebViewClient implemented.

![Figure 4.2: User interface of the browser page](image)
Apart from searching or loading URL using the address bar, users can load an URL outside Online Guard, by choosing online Guard to load the URL. Figure 4.3 shows the screenshot of choosing Online Guard to load the URL outside Online Guard.

![Screenshot showing Online Guard to load URL](image)

**Figure 4.3: Choosing Online Guard to load the URL outside Online Guard**

Online Guard will check the internet connection before loading an URL. If there is no internet connection, there will be a toast generated to tell the users to check the internet connection. Similarly, the app will remind the user if the URL is invalid. Two screenshots are presented in figure 4.4 about these two cases.
Figure 4.4: Error handling in loading websites

If the user is loading a website that is detected as phishing by Google safe browsing service, the website will be blocked with a red background, as shown in figure 2.5. Meanwhile, if the user is loading a website that is blocked by blocklist, APIs or AI, the default behaviour of the app will be going back to the previous page and showing a pop-up window. For example, if the user is loading a website that is detected as phishing by AI from the home page, the app will block the website and return to the home page with a pop-up window generated. With the default setting, there will also be a phone vibration. Users can press the ‘Cancel’ button to dismiss the pop-up window, or press the ‘Still Go’ button to load the website. If the user chooses to load the website, there will be a snack bar showing for reporting the website. By pressing the ‘report’ button, users can report the URL as a safe or phishing website. Figure 4.4 shows the screenshot sequences of loading a website that blocked by the blocklist and reporting it. Alternatively, users can choose to block the website directly with a toast instead of showing a pop-up window, like the screenshot shown in figure 4.5.
Figure 4.5: Screenshot sequences of loading a website blocked and reporting it

Figure 4.6: Blocking the website directly with a toast instead of showing a pop-up window

To add a website to the blocklist, users can press the red button directly, there will be a pop-up window asking for confirmation. Similar to the previous example, users can choose to press the ‘Cancel’ button to dismiss the pop-up window. Alternatively, users can press the
‘Add’ button to add the currently URL to the blocklist. The browser will then return to the previous webpage and show a toast for notifying the users. Also, there will be a snack bar showing for the users to report the website if appropriate. Figure 4.6 shows the screenshot sequences of adding a website to the blocklist by pressing the red button.

Figure 4.7: Screenshot sequences of adding the current website to the blocklist

The home page of the browser is ‘https://www.google.com’. If users attempt to add the home page to the blocklist by pressing the red button, there will be a toast generated to remind the users that the home page cannot be blocked. Similarly, there will be a reminder if the users try to add a duplicated domain name or URL to the blocklist. Two screenshots are presented in figure 4.8 to show the related design.
4.1.2 Blocklist Page

Figure 4.9 shows the user interface of the Blocklist page. It consists of:

1. **Title**: showing the title of the page – CUSTOMIZED BLOCKLIST.
2. **Info panel**: showing tips and the current number of items in the blocklist which is updated instantly after adding or deleting item.
3. **Input text box**: allow users to enter URL or domain name to add to the blocklist.
4. **Add button**: adding the URL or domain name in the input text box to the blocklist. Changes are saved instantly in the local device.
5. **Blocklist**: a scrollable listview to show all the items in the blocklist. Blocklist is loaded when the blocklist page is opened.
For adding domain name or URL to the blocklist, users can type in the text box and then press the add button or the done button on the keyboard. For deleting item, users can tap the item on the list and a pop-up window will be generated to ask for confirmation. In the pop-up window, the item to delete would be displayed. Users can press the ‘Cancel’ button to cancel the operation or press the ‘Remove’ button to confirm the operation. For both adding and deleting, the changes will be saved into the local device immediately and a toast will be generated to notify the users that the operation is successful. Also, the listview will be changed immediately for the users to see the changes. Meanwhile, the number of URL in blocklist will be updated in the info panel. Figure 4.10 and 4.11 shows the screenshots of adding and deleting item in the blocklist respectively.
Figure 4.10: Adding item to the blocklist

Figure 4.11: Deleting item in the blocklist
Furthermore, if there are invalid input in adding item to the blocklist, there will be toast generated to warn the users. For example, the input should not contain space and the domain name ‘google’ could not be added to the blocklist since the home page of the browser is the Google search engine. Figure 4.12 shows the four possible cases of invalid input.

Figure 4.12: Showing warning for invalid input in adding item to blocklist
4.1.3 News Page

Figure 4.13 shows the user interface of News page. It consists of:

1. **Title**: title of the page – NEWS.
2. **News**: a customized scrollable listview using a customized View adapter, in order to show both news title and the source of news in every items. News are loaded from the development server through the HTTP server by GET request when the users go to the news page, as mentioned in section 2.4.1.

The users can tap any item to load the corresponding URL for reading the news, the website will be opened in a simplified version of the browser page that removed the address bar and the buttons. Before the page is fully loaded, there will be a loading panel in the centre. The fully loaded page will then replace the loading panel. Figure 4.14 shows the screenshot sequences of reading a cyber security news in the news page.
Similar to the browser page, Online Guard would check the internet connection before fetching the news from the development server. If there is no internet connection, there will be a toast generated to remind the users to check the internet connection, like the screenshot shown in figure 4.15.
4.1.4 Info Page

Figure 4.16 shows the user interface of the Info page. It consists of:

1. **Title**: title of the page – ’10 Tips to avoid Online Deception’
2. **Info**: a scrollview that contains 10 cyber security practices to avoid online deception, such as setting strong passwords and avoid opening suspicious links.

![Figure 4.16: User interface of the Info page](image)

After reading the page, users can attend a short MC quiz in the quiz page.

4.1.5 Quiz Page

Figure 4.17 shows the user interface of the Quiz page, which consists of:

1. **Title**: title of the page – QUIZ, with some additional information to tell the users that the first 5 questions are related to the 10 cyber security practices shown in the Info page while the last 5 questions are more challenging.
2. **Quiz**: A scrollview that contains 10 MC questions. Each of the question has 4 possible answers for selection.
3. **Submit Button**: allows users to submit their answer after finishing the quiz. Their answers will be marked and all the correct answers will be revealed.
After the user has answered all the questions, he can press the ‘Submit’ button to submit his answer. The score of the user will be calculated where each question worths 10 scores and the total score of the quiz is 100. The ‘submit’ button will then be replaced by the score of the user. Furthermore, the answers of the user will be marked and the radio buttons will be disabled. The correct answer will be highlighted in green colour if the question is answered correctly by the user. Alternatively, the correct answer will be highlighted in red colour if the question is not answered rightly by the user. Figure 4.18 shows the screenshot of the quiz page after submission.
Moreover, if the user tries to submit the quiz before all questions are answered, there will be a toast generated to remind the user to answer all questions before submission, like the one shown in figure 4.19.

Figure 4.18: Screenshot of the quiz page after submission

Figure 4.19: Reminding in the Quiz page when not all questions are answered
4.1.6 Other Pages

Figure 4.20 shows the user interface of the Setting page, which consists of:

1. **Title**: title of the page – SETTING AREA.
2. **Setting area**: switches and button for configuration, such as enabling or disabling a particular technology to block suspicious or malicious websites.

![User interface of the Setting page](image)

**Figure 4.20: User interface of the Setting page**

When the ‘Set as default browser’ button is pressed, the mobile phone will be redirected to the related part in the phone setting. The users can select Online Guard as the default browser app then. Figure 4.21 shows the corresponding screenshots for the setting. For the switches, enabling or disabling them will be saved to the local storage immediately with a toast generated to notify the users, like the one shown in figure 4.22.
Figure 4.21: Selecting Online Guard as the default browser app

Figure 4.22: Toast generated for notifying users that the changes have been saved
Figure 4.23 shows the user interface of the Help page, that consists of:

1. **Title**: title of the page – HELP.
2. **Information**: information about the app, such as the 3 main functions provided and some tips in using the app.

![Figure 4.23: User interface of the Help page](image)

### 4.2 App UX Design

In order to provide better user experience, some principles in UX design have been followed in the project, including:

1. **Responsiveness**: responds immediately after something is changed.
2. **Self-Descriptive**: provides guideline for the user interface.
3. **Customization**: allows users to decide their preferences on the behaviour of the app.
4. **Efficiency**: provides shorter way that can accomplish troublesome work.
5. **Consistency**: uses the same style in designing.

Details and examples would be given in the following sections.
4.2.1 Responsiveness

Online Guard responds immediately after something is changed by the users. For example, the number of items in the blocklist as well as the blocklist would be updated immediately after adding or deleting an item in the blocklist, as shown in figure 4.10 and 4.11. Moreover, as shown in figure 4.24, toast is used widely in the app to notify the users when something has been changed by the users, such as changing the settings of the app.

Figure 4.24: using toast to notify users when something has been changed by the users

Another example is the bottom navigation bar. When the users switch to a new page, the icon of the new page will become active immediately while the icons of other pages will become inactive. Figure 4.25 shows an example of switching from Browser page to News page.

Figure 4.25: Bottom navigation bar when switching from Browser page to News page
4.2.2 Self-Descriptive

Online Guard provides self-descriptive UI elements. Users can long press the button to check the function of it. Figure 4.26 shows the examples of long pressing the two buttons in the top action bar.

Moreover, there are hints on the input text boxes for showing what to input in the text box. Figure 4.27 shows the examples of the input text box in Browser page and Blocklist page respectively. For instance, in the input text box of Browser page, there is a hint to tell the users that he could either search something or enter a complete URL in the text box.

4.2.3 Customization

In Online Guard, users are allowed to decide their preferences on the behaviour of the app in Setting page. Figure 4.28 shows the possible settings for the users to make the decisions. Firstly, users can choose to enable or disable the phone vibration when a phishing or malicious website is detected. Secondly, users can choose to enable or disable specific technologies in detecting unsafe websites. Users can also decide between asking in a pop-up window before block the website or block the website directly and show a toast. Lastly, users can choose to set Online Guard as their default browser by pressing the ‘Set as default browser’ button.
4.2.4 Efficiency

Online Guard has provided some shortcuts for better efficiency. For example, as shown in figure 4.3, users can choose to use Online Guard to load an URL outside Online Guard, instead of copying the URL, and then opening Online Guard, pasting the copied URL in the address bar of the Browser page and loading the website.

Another example is the block button on the Browser page. If the users want to add the current URL to the blocklist, instead of copying the URL from the address bar, and then going to the Blocklist page, pasting the URL into the input text box and pressing the add button, users can just press the block button in red colour on the Browser page to add the current URL to the blocklist which will be saved to the local device instantly.

4.2.5 Consistency

The last UX design considered is consistency, which means that the style of the UI design should be similar throughout the app. Online Guard has used a consistent UI design. For example, the top action bar and the bottom navigation bar appear in the same look in different pages, so that users can switch to different pages easily. Moreover, in most of the pages, there
is a title of the page above a green dividing line with some information sometimes, like the screenshots shown in figure 4.29.

Figure 4.29: consistent UI design for the title of pages
5 Product Evaluation

The final product in the project would be the mobile application. Among the three main functions, only main function 1 – blocking reported phishing websites, could be tested and evaluated. This is because it is hard to find a large number of unreported phishing websites to test main function 2 – scanning for suspicious websites. Also, it is difficult to quantify and evaluate main function 3 – information sharing regarding cyber security.

This section would first discuss the methodology used in testing, followed by the result and the analysis.

5.1 Methodology

Since it is time-consuming to test a large amount of URLs in the mobile application manually, a python program has been developed to simulate the browsing with the same technologies implemented, including (1) Google Safe Browsing Service, (2) XGBoost model in the development server, (3) URL Reputation API from URLVoid, (4) Malicious URL Scanner API from IPQualityScore. Nonetheless, the program has simplified the process that the program only gets the prediction of a website as safe or unsafe from different technologies, instead of getting more details, like the risk score.

The program will first fetch the first 100 unsafe websites from a phishing database that is updated regularly and only shows the active links [18]. Then, for every URL, the program would ask for the prediction result from different APIs, including the API of model prediction in the development server, URL Reputation API and Malicious URL Scanner API. For Google Safe Browsing service, instead of using the API provided, a website crawling mechanism has been developed in order to query the Safe Browsing site status from Google for simplicity [19].

The result was calculated by the program and recorded into a excel spreadsheet which would be shown in the next section.
## 5.2 Result

Table 5.1 shows the testing results. 100 different unsafe websites were tested. For each website, the four technologies were used to test the URL to check if the technology could detect the website as a phishing or malicious website.

<table>
<thead>
<tr>
<th>Unsafe website</th>
<th>Can be detected by AI?</th>
<th>Can be detected by Google?</th>
<th>Can be detected by URLVoid?</th>
<th>Can be detected by IPQS?</th>
<th>Number of technologies that detected the website as unsafe</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>4</td>
</tr>
<tr>
<td>12</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>4</td>
</tr>
<tr>
<td>13</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>3</td>
</tr>
<tr>
<td>14</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>3</td>
</tr>
<tr>
<td>15</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>2</td>
</tr>
<tr>
<td>16</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>3</td>
</tr>
<tr>
<td>17</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>2</td>
</tr>
<tr>
<td>18</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>3</td>
</tr>
<tr>
<td>19</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>4</td>
</tr>
<tr>
<td>20</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>3</td>
</tr>
<tr>
<td>21</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>2</td>
</tr>
<tr>
<td>22</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>4</td>
</tr>
<tr>
<td>23</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>2</td>
</tr>
<tr>
<td>24</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>2</td>
</tr>
<tr>
<td>25</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>2</td>
</tr>
<tr>
<td>26</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>3</td>
</tr>
<tr>
<td>27</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>3</td>
</tr>
<tr>
<td>28</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>2</td>
</tr>
<tr>
<td>29</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>2</td>
</tr>
<tr>
<td>30</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>3</td>
</tr>
<tr>
<td>31</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>4</td>
</tr>
<tr>
<td>32</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>3</td>
</tr>
<tr>
<td>33</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>2</td>
</tr>
<tr>
<td>34</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>3</td>
</tr>
<tr>
<td>35</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>1</td>
</tr>
<tr>
<td>36</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>4</td>
</tr>
<tr>
<td>37</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>2</td>
</tr>
<tr>
<td>38</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>3</td>
</tr>
<tr>
<td>39</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>2</td>
</tr>
<tr>
<td>40</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>2</td>
</tr>
<tr>
<td>41</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>3</td>
</tr>
<tr>
<td>42</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>3</td>
</tr>
<tr>
<td>43</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>3</td>
</tr>
<tr>
<td>44</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>45</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>3</td>
</tr>
<tr>
<td>46</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>3</td>
</tr>
<tr>
<td>47</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>2</td>
</tr>
<tr>
<td>48</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>4</td>
</tr>
<tr>
<td>49</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>4</td>
</tr>
<tr>
<td>50</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>2</td>
</tr>
<tr>
<td>51</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>3</td>
</tr>
<tr>
<td>52</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>4</td>
</tr>
<tr>
<td>53</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>4</td>
</tr>
<tr>
<td>54</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>4</td>
</tr>
<tr>
<td>55</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>4</td>
</tr>
<tr>
<td>56</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>4</td>
</tr>
<tr>
<td>57</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>3</td>
</tr>
<tr>
<td>58</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>3</td>
</tr>
<tr>
<td>59</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>2</td>
</tr>
<tr>
<td>60</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>4</td>
</tr>
<tr>
<td>61</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>3</td>
</tr>
<tr>
<td>62</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>4</td>
</tr>
<tr>
<td>63</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>4</td>
</tr>
<tr>
<td>64</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>2</td>
</tr>
<tr>
<td>65</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>2</td>
</tr>
<tr>
<td>66</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>2</td>
</tr>
<tr>
<td>67</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>3</td>
</tr>
<tr>
<td>68</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>2</td>
</tr>
<tr>
<td>69</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>4</td>
</tr>
<tr>
<td>70</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>4</td>
</tr>
<tr>
<td>71</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>4</td>
</tr>
<tr>
<td>72</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>4</td>
</tr>
<tr>
<td>73</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>2</td>
</tr>
<tr>
<td>74</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>3</td>
</tr>
<tr>
<td>75</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>4</td>
</tr>
<tr>
<td>76</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>4</td>
</tr>
<tr>
<td>77</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>3</td>
</tr>
<tr>
<td>78</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>4</td>
</tr>
<tr>
<td>79</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>2</td>
</tr>
<tr>
<td>80</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>2</td>
</tr>
<tr>
<td>81</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>2</td>
</tr>
<tr>
<td>82</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>2</td>
</tr>
<tr>
<td>83</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>3</td>
</tr>
<tr>
<td>84</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>3</td>
</tr>
<tr>
<td>85</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>4</td>
</tr>
<tr>
<td>86</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>3</td>
</tr>
<tr>
<td>87</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>2</td>
</tr>
<tr>
<td>88</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>3</td>
</tr>
<tr>
<td>89</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>3</td>
</tr>
<tr>
<td>90</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>4</td>
</tr>
<tr>
<td>91</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>2</td>
</tr>
<tr>
<td>92</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>4</td>
</tr>
<tr>
<td>93</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>2</td>
</tr>
<tr>
<td>94</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>2</td>
</tr>
<tr>
<td>95</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>3</td>
</tr>
<tr>
<td>96</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>2</td>
</tr>
<tr>
<td>97</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>4</td>
</tr>
<tr>
<td>98</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>2</td>
</tr>
<tr>
<td>99</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>3</td>
</tr>
<tr>
<td>100</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>2</td>
</tr>
</tbody>
</table>

Number of websites: 53, 51, 100, 88
<table>
<thead>
<tr>
<th>detected by the technology</th>
<th>53%</th>
<th>51%</th>
<th>100%</th>
<th>88%</th>
</tr>
</thead>
</table>

Table 5.1: Testing result in product evaluation

URLs of the websites tested would be listed in Appendix I.

**5.3 Analysis**

The following conclusions could be drawn according to the testing results.

Firstly, a combination of the four technologies is effective in detecting reported phishing websites. In the testing conducted, all of the unsafe websites were detected by at least one technology which means that there is no unsafe website can be passed in the Online Guard.

Secondly, URL Reputation API from URLVoid and Malicious URL Scanner API from IPQualityScore are reliable in detecting reported phishing websites since they both have detected most of the unsafe websites. For URLVoid, it even detected all of the unsafe websites. This is because URLVoid is relatively stricter than IPQS in calculating the risk score.

Thirdly, Google safe browsing service has a similar performance as the XGBoost model developed in the development server. They both have achieved around 50% accuracy which is not as good as the two APIs aforementioned.

The overall performance is satisfactory since all unsafe websites have been detected successfully so that the users could be protected from the phishing or malicious websites.
6 Project Evaluation

The proposed three main functions have been implemented successfully. Nonetheless, there are still some limitations which would be discussed in this section.

6.1 Limitation

Two limitations of the project were encountered.

Firstly, there is a lack of dataset in AI development. It is estimated that the number of websites in the world is about 1.7 billion [20]. Hence, there are a lot of patterns to identify between safe and unsafe websites, due to the large number of websites in the world with different patterns in the features. However, there is a lack of resource in phishing dataset. The best dataset found in the exploration stage was the one used in model training which is not sufficient in learning the patterns. Hence, the XGBoost model has only achieved recall of 74.4% which still has room for improvement.

Secondly, there are quotas in using the two APIs for URL detection. Currently, a basic plan in URLVoid and a free plan in IPQS were used. Usage of APIs are limited to the corresponding quotas which are about 2000 to 5000. Hence, it might not be enough in the long term.

6.2 Future Development

Future development could be focused on solving the problems aforementioned.

For the first problem, the dataset could be enriched by merging datasets from different sources. Furthermore, some website crawling programs could be developed to fetch safe and unsafe websites for adding more data to the datasets.
For the second problem, adding advertisement in the mobile app for monetization could be considered, such as Google AdMob. The income in advertisement could be used in buying credits in the APIs. Furthermore, the process of adding credits could also be automated so that there would be an automation program in the development server which will buy credits for the APIs automatically when the number of quotas available is lower than a certain number.
7 Conclusion

This report has introduced the background, objectives, motivations and scope of the project in the introduction section. Moreover, the system design and the mechanism of implementing the three main functions are discussed in the methodology section. After that, the detail of implementing the AI model has been elaborated in the AI model section, such as the dataset, data pre-processing and improvements conducted. Next, the UI design, logic flow as well as UX design has been shown in the app design section. The report ends with the testing of the product in the product evaluation part, followed by limitation and future development in the project evaluation part.

The three main functions proposed at the beginning of the project has been implemented successfully, including detecting for reported phishing websites, scanning for suspicious websites and information sharing regarding cyber security. Multiple technologies are used for detecting reported and unreported websites in the browser of the app, including user-defined blocklist, Google safe browsing service, XGBoost Model in the development server, URL Reputation API from URLVoid as well as Malicious URL Scanner API from IPQualityScore. For information sharing, users could view the latest cyber security news and some practical techniques to avoid online deceptions with a quiz for knowledge consolidation.

The AI model and the mobile app have been tested and evaluated. Satisfactory performance could be found in the testing. However, there are still some limitations of the project, such as the lack of dataset in AI development and limited quotas in using the APIs for URL detections. The limitations could be tackled by merging the datasets from different sources and adding advertisement in the app respectively.
References


Appendices

Appendix I: URLs of unsafe websites used in testing in section 5

1: http://0wa477gswk848mbc7309gd.mattsenior1.repl.co
2: http://3dcloud.co.uk/wp-content/themes/twentyten/fonts/jw/rakne/Process/pc/7e7dbfdec8a5394fc27a6cde9a3d
3: http://3dcloud.co.uk/wp-content/themes/twentyten/fonts/jw/rakne/Process/pc/7e7dbfdec8a5394fc27a6cde9a3d/cardinfof4e.php
4: https://3lnbwozouaaxkmrqmtyxnhrosmfudwfyetiwmie.filesusr.com/html/46d2e9_6a7c2030c9ba4a404a357027e732be3f.html
6: http://188.128.202.35.bc.googleusercontent.com
7: https://687f7cce0d3669684.temporary.link/s/update.php
8: http://4666.co.kr/webpage
9: http://27976a9732.nxcli.net/account/fb.php
10: http://27976a9732.nxcli.net/account
11: http://21-123-2.vasdasdvvvv.repl.co
12: http://aapartments.com.ua/themes/zimbra/ZimbraWebSignIn.htm?email=oem@spal.it
14: https://accessboanlines.com/here/signin.php
15: http://account-device-register.com
17: http://activadigital-1.com
18: http://adsbusinessaccountcreditscoupon.com
19: http://adxpklseomcjptymzneldzastg-dot-polished-shore-301017.uk.r.appspot.com
20: https://agentwealthsuccess.com/wp-admin/users/office-2020
21: http://aggiornamentosicuro.com
22: https://agitated-swanson-2400de.netlify.app
23: http://albaitalshamy.com/3ed/suntrust/suntrust
24: https://alert-new-hali.com
26: https://alhazmico.com/assets/fonts/codropsicons/METRO/METRO
28: https://amazon-osaka.xyz
29: http://anbkstvpokqrdklfaobjectq-dot-polished-shore-301017.uk.r.appspot.com
31: http://anozlnd.com/personal-details.app.php?stage=step2&amp;webssl=2tV3ipFwdWm5elrCiJnYLILgT&amp;DMO
32: http://anvietlong.com/Login/Secure_Zone/Confirm/websc_signin
33: http://aphomes.in/img/clients
34: http://api.cargomanager.io
35: http://app11.easyfisendyapp.com/orangeposts
36: https://archiepba.com/cade/z0n51-paypal/60339ae3373bfb1/login.php
37: https://asfysfgqlassxrotawcmaqe-dot-polished-shore-301017.uk.r.appspot.com
38: https://assist-personal.com/hsbc
39: https://authorise-mydetails.com/lloyds/Login.php
40: https://authorise-mypayee.live/hsbc
41: https://auth-revoke.com/hsbc
42: http://autodex53.xyz/tttttaa/Login.php
43: https://averagesustain.com/Arica/Anderson
44: http://avisoperu.com/clientes
46: http://axomonline.com/ln/Exitkorea/upload
47: http://baithi1.xyz
48: http://bajatel.mx/wp-content/themes/twentynineteen/js/i2k/bizmail.php?email=&amp(rand=13vqcr8bp0gud&amp;lc=1033&amp;id=64855&amp;mkt=en-us&amp;bcxt=mai&amp;snc=1
49: https://bancapinternet-interbank.movilaplicativo.com/05948448/personas
50: https://bankgbzbhrvnlbtblweeipccts-dot-polished-shore-301017.uk.r.appspot.com
71: http://bramblebaybowlsclub.com.au/X08fg4he5e915180c/?xra=arx&amp;xav=anVsaWUuY XVjb2luQGludHJidG94LmNvbQ==
72: http://brave-poincare-c1bcee.netlify.app
73: https://brusselsarport.be/umglbk2b45c017jq
74: https://bshdnek.bdnnekwhjsdl.shop
75: http://btbrpfknucemdinwytnxcriu-dot-polished-shore-301017.uk.r.appspot.com
76: http://bulkydeafeningprofessional--five-nine.repl.co
78: https://bvhoncsrywanaofgtodzuzlwb-dot-polished-shore-301017.uk.r.appspot.com
79: https://bvkvlqjgkxuchtgjrvtswhpq-dot-polished-shore-301017.uk.r.appspot.com
80: https://bwduityovftlqqxdaqevbzxexfr-dot-gleow2021ja.te.r.appspot.com
81: http://cambalkoncum.net/secure.html
82: https://cambridge-edu.uk/ridge/image.png
83: http://camminoincodains.com/hxc6et672dy
84: https://cancelpayee-attempt.com/Lloyds/Login.php
85: https://capitalpower.com.pk/It/office365/login.php?cmd=login_submit&amp;id=2355c2415f 03d484ee6a9b6bb3431c2355c2415f03d484ee6a9b6bb3431c&amp;session=2355c2415f 03d484ee6a9b6bb3431c2355c2415f03d484ee6a9b6bb3431c
86: http://caseythomasrd.com/a01/chameleon/microsoft/login.microsoft.com/2Auth/location/sessi on-id/9588Tv9uyTrh489388399488A493004900349/2Fmail-Authentication
87: https://cateroo.id/assets/fonts/nn/1und1
88: http://cbsh.ca/wp- includes/SimplePie/Cache/include/swift/tsf/tsf/Wetransfer.com/9054/00875546/login
89: http://cbsndkewualde.shop
90: https://cclnjcyzkhpwtymxbsnolkpmpd-dot-polished-shore-301017.uk.r.appspot.com
91: http://cedubgxrueskeewugolnerutobmaljuixo-dot-owaonk3993993993.uk.r.appspot.com
93: http://ceula161.ru/ceula161.ru/bid/information.php?access.x36861750344&amp;&amp;data.x= en_81405df056e6e7d9c952dfecc
Appendix II: Installation Guide of Online Guard

There are two ways to install and run Online Guard.

Method 1: Running the app in a virtual device in Android Studio

1. There is a zip file submitted with the report called ‘Online_Guard.zip’. Unzip the file:

2. Open the folder in Android Studio:
3. Create a virtual device in Android Studio if there is no virtual device available:

![Virtual Device Manager](image)

4. Use the virtual device to run the App:

![App Running](image)

**Method 2: Install the app in an android device**

1. There is a zip file submitted with the report called ‘Online_Guard.zip’. Unzip the file:

![Unzip File](image)

2. Locate the file ‘app-debug.apk’ in the path ‘Online_Guard\app\build\outputs\apk\debug’:
3. Transfer the file to the Android phone by any mean. Click the apk file in the android phone for installation: