Bibliography

[1] Yifei Ma et al. Aladdin: Automating release of android deep links to in-app content. In *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*, pages 139–140, Buenos Aires, 2017.

[2] Yifei Ma, Xiaoxing Liu, Rui Du, Zhiqiang Hu, Yun Liu, Min Yu, and Guangtai Huang. Droidlink: Automated generation of deep links for android apps. *arXiv preprint arXiv:1605.0692*, 2016.

[3] Yizheng Liang and Xiaodong Yan. Using deep learning to detect malicious urls. In *2019 IEEE International Conference on Energy Internet (ICEI)*, pages 487–492, 2019.

[4] Zhiqiang Yang, Ming Yang, Yanchao Zhang, Guofei Gu, Peng Ning, and Xiaofeng Steve Wang. Appintent: Analyzing sensitive data transmission in android for privacy leakage detection. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, New York, NY, USA, 2013. Association for Computing Machinery.

[5] Min Zhang and Heng Yin. Appsealer: Automatic generation of vulnerability-specific patches for preventing component hijacking attacks in android applications. In *Network and Distributed System Security Symposium*, 2014.

[6] Elaine Chin and David Wagner. Bifocals: Analyzing webview vulnerabilities in android applications. In *Proceedings of the 7th ACM Conference on Security and*

Privacy in Wireless and Mobile Networks, University of California, Berkeley, USA, 2014.

[7] Liu F., Wang C., Pico A., Yao D., and Wang G. Measuring the insecurity of mobile deep links of android. In *Proceedings of the USENIX Conference on Security Symposium.*, 2017.

[8] De-Jun Wu, Chen-Hung Mao, Te-En Wei, Hao-Ming Lee, and Kuo-Pei Wu. Droidmat: Android malware detection through manifest and api calls tracing. In *2012 Seventh Asia Joint Conference on Information Security*, pages 62–69, Tokyo, Japan, 2012.

[9] R. Dhaya and M. Poongodi. Detecting software vulnerabilities in android using static analysis. In *2014 IEEE International Conference on Advanced Communications, Control and Computing Technologies*, pages 915–918, Ramanathapuram, India, 2014.

[10] Samuel Feldman, David Stadther, and Bin Wang. Manilyzer: Automated android malware detection through manifest analysis. In *2014 IEEE 11th International Conference on Mobile Ad Hoc and Sensor Systems*, pages 767–772, Philadelphia, PA, USA, 2014.

IMPLEMENTATION & TESTING OF DEEP LINK IN AN ANDROID APPLICATION

A Dissertation submitted in partial fulfilment of the requirements
for the award of the degree of

M.Sc. Computer Science (CyberSecurity)

by

DHRUTI RANJAN MOHANTY

(Enrollment No. MSCS21R004)

Under the Supervision of

Dr. P.THIYAGARAJAN

Department of Computer Science (CyberSecurity)

Rajiv Gandhi National Institute of Youth Development

Ministry of Youth Affairs and Sports, Government of India

Sriperumbudur, Tamil Nadu - 602105, India

June, 2023

Dedicated to my beloved parents.

Declaration

I, **Dhruti Ranjan Mohanty**, hereby solemnly declare that the written submission titled **"Implementation & Testing Of Deep Link In An Android Application"** presented herewith is a genuine representation of my own ideas, articulated in my own words. I affirm that this work has been prepared with utmost academic honesty and integrity, in adherence to the established principles and ethical guidelines. Throughout the development of this written submission, I have exercised due diligence to ensure the accuracy and integrity of the information presented. Whenever I have incorporated the thoughts, ideas, or words of others, proper attribution and referencing have been diligently provided to acknowledge the original sources. I acknowledge that any violation of the aforementioned principles and guidelines undermines the academic integrity of this submission.

By signing this declaration, I affirm that the work presented in this submission is a true reflection of my own ideas, and I bear full responsibility for its content and authenticity.

.....

Place: **Sriperumbudur, Tamil Nadu** **Dhruti Ranjan Mohanty**

Date: Enrollment No.: MSCS21R004

Certificate

This is to certify that the dissertation titled **"Implementation & Testing Of Deep Link In An Android Application"**, submitted by **Dhruti Ranjan Mohanty** to the **Rajiv Gandhi National Institute of Youth Development, Sriperumbudur, Tamil Nadu**, in partial fulfillment of the requirements for the degree of **M.Sc. Computer Science (CyberSecurity)**, is an authentic and original research work conducted by the candidate under our supervision and guidance.

The thesis has successfully met the standards and requirements prescribed by the regulations governing the award of the degree. The research findings and results presented in this thesis have not been previously submitted, either in part or in whole, to any other university or institute for the purpose of obtaining any other degree, to the best of our knowledge.

We attest to the originality, integrity, and academic rigor of the work carried out by **Dhruti Ranjan Mohanty**, and we recommend the acceptance of this dissertation for the award of the degree mentioned above.

Place: **Sriperumbudur, Tamil Nadu**
Date:

.....
Guide: **Dr. P.Thiyagarajan**
Associate Professor and Head,
Department of Computer Science (CyberSecurity)

.....
Controller of Examinations

Acknowledgements

I would like to take this opportunity to express my heartfelt gratitude towards all the individuals who have provided their unwavering support and guidance throughout my thesis work. I am immensely thankful to the Almighty for the abundance of grace and blessings that have enabled me to successfully complete this thesis.

I am deeply indebted to **Dr. P. Thiyagarajan**, Assistant Professor and Head of the Department of Computer Science (CyberSecurity) at Rajiv Gandhi National Institute of Youth Development, Sriperumbudur. Their valuable guidance, encouragement, and expertise have played a pivotal role in shaping this research work.

I would also like to extend my sincere appreciation to my Family and Friends for their enduring support, unwavering concern, and constant encouragement throughout this journey. Their presence has been instrumental in helping me overcome challenges and accomplish this significant task.

Dhruti Ranjan Mohanty

Abstract

Deep linking has emerged as a vital technology in the field of mobile app integration and user experience. It enables seamless transitions between different digital contexts by allowing users to access specific content or features within apps directly. This thesis explores the concept of deep linking, its implementation methods, challenges, and the impact of deep linking on user engagement and app discoverability. The paper begins by providing an overview of deep linking, outlining its purpose and potential benefits. It examines the technical aspects of deep linking, including the various approaches such as uniform resource identifiers (URIs), custom URL schemes, Android App Links, and iOS Universal Links. The implementation methods for deep linking are discussed, emphasizing the importance of proper integration into the apps codebase. One of the primary challenges associated with deep linking is device fragmentation, which encompasses differences in operating systems, screen sizes, resolutions, and hardware capabilities. Additionally, problems can occur if the user's smartphone doesn't have the software necessary for deep linking, which could cause friction along the user's journey. Deferred deep linking, which enables visitors to be redirected to specified information even if the app has to be installed first, is a technique used by marketers and developers to overcome this issue. This thesis examines the idea of deep linking, its methods of implementation, the difficulties brought on by device fragmentation, and the problems with app installation status. User journeys will be further streamlined and the promise of connected digital experiences will be fully realized with further innovation and breakthroughs in deep linking technologies.

Keywords: *Deep Link, Android Application, Android App Links, iOS Universal Links*

Contents

List of Figures

Abbreviations

APK	**A**ndroid application **P**ac**K**age
zip	**zip**ped file
URL	**U**niform **R**esource **L**ocator
URI	**U**niform **R**esource **I**dentifier
iOS	**i**Phone **O**perating **S**ystem
HTTP	**H**yper **T**ext **T**ransport **P**rotocol
HTTP	**H**yper **T**ext **T**ransport **P**rotocol **S**ecure
W3C	**W**orld **W**ide **W**eb **C**onsortium
DEX	**D**alvik **EX**ecutable
GPS	**G**lobal **P**ositioning **S**ystem
ADB	**A**ndroid **D**ebug **B**ridge
IDE	**I**ntegrated **D**evelopment **E**nvironment
API	**A**pplication **P**rogramming **I**nterface

Chapter 1

Introduction

1.1 Introduction

The current generation came into being in a time of quick technological development and easy access to knowledge. The experiences and behaviours of these users have been significantly shaped by Android applications. The ease, enjoyment, and access to a large digital universe offered by these programmes have become crucial parts of their everyday lives. People now do a variety of jobs, connect with others and consume material in whole new ways due to the growth of Android applications. Android applications serve a wide range of needs and interests, from social networking platforms to productivity tools to gaming apps to educational materials. They now serve as a primary platform for learning, amusement, communication, and even personal development. Android phones have evolved into crucial instruments for increasing productivity and controlling numerous facets of contemporary lifestyles. These programmes enable users to lead more effective, organised, and balanced lives. They range from productivity apps that simplify chores, organise calendars, and enable remote work to fitness and health apps that measure physical activity and encourage well being. A platform for customization and self-expression

is provided by Android apps. Through picture editing, video creation, and graphic design applications, users may personalise the look of their devices, their social networking profiles, and their creative expression. People may use these programmes to share with the world their distinctive personalities, skills, and opinions. The process that the current generation connects and communicates with one another has changed as a result of Android applications like messaging apps and social media platforms. These applications let users to communicate with friends, family, and coworkers anywhere in the world through instant messaging, phone and video chatting, and social networking capabilities. They have aided in the communication of ideas, supported online groups, and boosted interpersonal bonds. The way that today's generation obtains knowledge and receives information has been revolutionized by Android applications. Users may study a wide range of topics, keep up with current affairs, and improve their skills and understanding through interactive material, which is available on anything from news sites to educational applications. Users are now able to easily access knowledge through Android applications, broadening their horizons and intellectual interests. In the current age, Android apps have completely changed what it means to be entertained and have fun. These programmes offer a wide range of possibilities for entertainment, including streaming services for movies, TV shows, and music, as well as game apps with immersive experiences. They provide customers access to a customised and instant entertainment experience that enables them to unwind, relax, and partake in activities that are in line with their tastes and interests. In today's fast-paced and technologically driven world, mobile applications have become an indispensable part of our lives. To meet the demands and tastes of a generation that is becoming more and more dependent on smartphones, Android has emerged as the industry leader among mobile operating systems. The way we connect with information, services, and entertainment has changed as a result of Android applications, which have also had an impact on how

today's generation interacts with one another, lives, and uses technology. Since their origin, when Android applications were isolated and only capable of basic functions, they have advanced significantly. The need to link apps and build a more connected environment was recognized by developers as technology improved.

1.2 Basic Concepts of Deep Link

Deep linking, which enables users to immediately access particular material within an app, was consequently included in Android applications. Deep linking makes it unnecessary for consumers to go through many stages and gives them immediate access to the material they want. It may be started from a variety of places, including websites, emails, texts, and other apps. User experience is improved through deep links, which open the related app and direct users to the necessary material. Deep linking improvements have improved how seamlessly various programs integrate with one another. Due to the possibility for cross-app communication, deep links inside one app can launch certain activities or visit specific websites. Deep linking increases app discoverability by making deep connections shareable across several platforms. Deep linking also has the benefit of enabling personalized experiences within apps, which increases engagement and personalisation. Deep linking can be implemented uniformly across a variety of platforms and devices, although fragmentation makes this difficult. Deep connections must continue to work after app upgrades or structural changes, which adds complexity to maintenance. To solve security concerns connected to deep linking, appropriate procedures for authentication and permission are required. Deep linking may expose users to security threats including unauthorized access and URL spoofing. If deep connections are not deployed securely, user information may be compromised due to data leakage.

Users need to be made aware of the dangers and developers must encourage thorough link authenticity checks. Deep linking has transformed how consumers engage with Android apps, despite these restrictions and security worries. In addition to fostering seamless app integration, it has enhanced user experience and app discoverability. Developers now depend on deep linking as a key technique for increasing engagement and offering individualized experiences. Deep linking is anticipated to develop further as technology advances, overcoming constraints and security issues. In the contemporary digital world, mobile applications have a big influence on every part of our lives. Android has emerged as the top platform among the several mobile operating systems, grabbing a sizable portion of the market. As they have developed over time to improve user experience and allow seamless connection with other apps and services, Android applications now offer a wide variety of features. Deep linking is one such innovation that has transformed how people interact with apps and information on Android devices. Deep links connect a specific link to a specific piece of content or feature within the app by using a unique URL structure or URI (uniform resource identifier). The operating system detects a deep link when a user clicks on it and sends them to the appropriate app content. Since 2008, both iOS and Android have supported the most popular deep link. The mobile OS can register URI schemes with an app during installation if the programme wishes to be launched from the web. The format can vary depending on the platform or app.
For example: <**scheme**>://<**host**>/<**path**>?<**query-parameters**>. As a result, the app is able to register and manage particular URLs that are activated from either inside or outside the app. The custom URL scheme often begins with a distinctive identifier selected by the app developer, such as “myapp://” or “mycustomscheme://”. This type of URL launches or comes to the foreground when it is clicked or triggered, enabling the connected app to manage and process the particular URL. But the problem with this scheme URL is that any other app or

malicious app can register the same scheme and trick you into opening their app. As a result they can steal your information.

To address this problem, the Intent URL in 2013 and the App Link in 2015 came into picture. *(a) Intent URL:* It is limited to Android. Websites should call deep links in accordance with the intent URL, which is defined. In order to prevent misunderstanding, Intent URL includes the target app identification (i.e., package name) in the argument directly rather than invoking "myapp://path1". *(b) App Link:* Comparing App Link to conventional scheme URLs, you can see an improvement in security. App Link employs domain verification, which necessitates hosting particular JSON metadata files on the website connected to the app, to establish the connection between a domain and an app. By ensuring that the programme only handles URLs from recognised domains, this verification lowers the possibility of spoofing or harmful usage. In some circumstances, the system may ask the user to approve the action when an app attempts to handle an app link URL for the first time. This adds an additional layer of protection and prevents deep connections from being handled by unintentional or unauthorised apps. The "autoVerify" element found in the app's manifest gives App Link users a way to validate URLs. In order to avoid hijacking or manipulation, this guarantees that the URLs linked to the app are routinely checked.

1.3 Evolution of Deep Link

The concept of deep linking has evolved over time to enhance the user experience and enable seamless navigation between apps and content. Here is a brief overview of the evolution of deep linking:

a) **Scheme-Based Deep Links:** Initially, deep linking was primarily done using scheme-based URLs, which allowed apps to register a custom URI scheme (e.g., "myapp://") to open specific screens or perform actions within the app. Scheme-based deep links provided basic functionality for linking from one app to another or from the web to an app. However, they were limited to specific platforms and lacked universal support.

b) **App Links:** App Links were introduced by Google in 2015 as part of Android's App Links feature. App Links provided a standardized and more versatile way to associate web URLs with specific apps. It allowed websites to declare their association with corresponding apps, enabling a seamless transition between the web and app experiences. App Links offered enhanced verification and improved compatibility across devices and platforms.

c) **Universal Links:** It introduced by Apple in 2015 with iOS 9, were the iOS equivalent of Android's App Links. Universal Links enabled web URLs to directly open the corresponding app on iOS devices. Like App Links, Universal Links aimed to provide a seamless user experience between the web and app environments. They utilized app-associated domain verification and relied on standard HTTP/HTTPS URLs.

d) **Deferred Deep Links:** Deferred deep linking emerged as a solution to address the scenario where a user clicks on a deep link but doesn't have the corresponding app installed. Deferred deep links allow the user to be redirected to the app store to install the app, and upon installation, the deep link is handled to provide the intended content or functionality. This approach enhanced user acquisition and engagement by allowing users to seamlessly transition from a web page or advertisement to the app, even without the app initially installed.

e) **Standardized Deep Linking Standards:** Over time, efforts have been made to establish standardized deep linking standards that work across multiple platforms. Examples include the Deep Linking Metadata specification by W3C, which provides a common format for specifying deep link metadata, and the URI scheme and HTTP URL conventions that have become widely adopted practices for deep linking.

The evolution of deep linking has brought about improved compatibility, standardization, and enhanced user experiences, allowing users to seamlessly navigate between apps and content, regardless of the platform or device they are using.

1.4 Terminology

To grasp deep link functionality in mobile applications, you have to first understand the interoperability of the application, which is nothing but known as inter-app communication and the basic terminology related to android application.

a) **Activity:** The screen of an app or the webpage of a website is the same as an activity in an app.

b) **Services:** It is a continuous process that the user must intentionally interrupt in order to end it. For example: background tasks like GPS.

c) **Broadcast Receivers:** The term "broadcasts" refers to messages that are transmitted to mobile devices. Broadcast receivers are the components that keep an eye out for these signals. For example: Charging notification.

d) **Content Providers:** The term “content providers” refers to a connection between two applications that share data.

e) **Content Resolvers:** The term "content resolvers" refers to a component that an app uses to make data queries.

f) **Intent:** A key element of the operating system that facilitates communication between various activities is intent. It may be two types:

- *Explicit:* Intent that communicate between two or more activities of same application.
- *Implicit:* Intent that communicate between two or more activities of different application.

g) **Intent Resolution:** It's the process that checks which implicit intent needs to call which activity.

Android application files have an '.apk' extension, which is equivalent to a'.zip' file. Unzip command could easily extract a '.zip' file but it would be encoded. Some of the main file systems inside an apk file are listed below:

a) **AndroidManifest.xml:** The file that has all permissions, activities, content providers, services, and intents required for the app to run is listed.

b) **classes.dex:** It contains Java bytes in DEX, i.e., Dalvik Exchange format. This .dex file can be decompiled, and the application source code can be read.

c) **res:** A folder that has device configuration, bitmaps, and layouts.

d) **resources.arsc:** A file containing binaries of compiled components that might include images, strings, or other data used by an app.

e) **META-INF:** A folder containing the metadata and other manifest information about the Java package that the Java file is carrying. Files such as cert.sf, cert.rsa, and manifest.mf are included.

To enable communication between the web and mobile applications, mobile deep links cause a certain kind of intent to be activated. When users click on a link in the browser or in an app webview, the browser initiates an intent to activate the relevant component within the target app. In the case of mobile deep links, this process specifically triggers front-end activities within the app. It is important to note that mobile deep links are limited to launching activities within the app's user interface, unlike app-to-app communication which allows for broader interaction between different applications. Mobile deep links are used in two easy steps: *(a) Registration:* During the installation process, the "rgniyd" application is required to register its specific Uniform Resource Identifiers (URIs), such as "rgniyd://" or "https://rgniyd.com", with the mobile operating system. These URIs are then included within the "data" field of the intent filters, which defines the URIs associated with the application. *(b) Addressing:* When the "rgniyd://" button is pressed, the mobile OS checks all the intent filters for potential matches. The mobile OS will open the app "rgniyd" since the URL matches the URI of that app.

1.5 Tools & Software

a) **Android:** The Linux kernel serves as the foundation for the Android operating system, which provides a user-friendly interface that enables easy device navigation for consumers. Multi-tasking, alerts, widgets, and a huge selection of applications are just a few of the many features and functions that Android offers to meet a wide range of requirements and interests. The Google Play Store, which provides millions of programmes in a variety of categories, acts as the main app store for Android users. For enhancing the capabilities of their device, Android users have access to a broad range of options, including productivity tools, entertainment apps, social networking platforms, and gaming

experiences. The open-source Android operating system was created primarily for mobile devices. It was created by Google and has grown to be one of the most well-known and extensively utilised platforms worldwide. Android has completely changed how we use smartphones and tablets because to its adaptability, customization choices, and vast app ecosystem.

b) **Android Studio:** Android Studio serves as the endorsed Integrated Development Environment (IDE) for the purpose of Android application development. It provides a comprehensive set of tools and features specifically designed to streamline the process of building, testing, and deploying Android applications. The building of Android apps is the exclusive focus of Android Studio. It provides a comprehensive collection of tools and resources to help developers produce effective, high-quality Android applications. The Gradle build system, which is used by Android Studio and is in charge of managing dependencies, generating code, and producing APKs (Android application packages).

c) **Android SDK:** The resources and tools required for creating, testing, and debugging Android apps are provided by the Android SDK. It contains necessary elements including command-line tools, emulators, libraries, and APIs. One of the most important tools in the Android SDK is the SDK Manager. Different Android platform versions, system images, and other SDK components including support libraries, Google Play services, and tools are all accessible for download, installation, and management by developers. Android Studio includes an emulator that enables developers to run and test their apps on virtual Android devices. The emulator allows for testing app behavior on different screen sizes, device configurations, and Android versions.

d) **ApkTool:** APKTool is made to help with Android application reverse engineering. It offers a mechanism to decode APK files, extract resources, inspect

and alter the source code, and then recompile the altered APK into a useable format. APKTool has the ability to decompile APK files into the source code and resources that go with them. The compiled bytecode (DEX) of the APK is transformed into readable and editable Java source code files (SMALI) during this procedure, and additional resources, including as graphics, XML files, and assets, are also extracted. The primary method of operation for APKTool is through a command-line interface (CLI), which makes it ideal for automation, scripting, and integration with unique workflows. It gives users a collection of commands with a variety of choices to work with APK files in various ways.

e) **JADX:** With JADX, you may extract the original application's Java source code from decompiled APK files, which can then be edited and viewed. It is useful for comprehending the inner workings of Android apps, evaluating their operation, and making adjustments as necessary. Users may access the application's original source code and resources thanks to JADX's decompilation of APK files into Java source code. In order to make the compiled bytecode (DEX) simpler to comprehend and analyse, it is converted back into Java code. JADX provides a user-friendly interface for analyzing the decompiled Java code. It allows users to navigate through the classes, methods, and resources of the application, providing insights into the application's architecture, functionality, and implementation details.

1.6 Thesis Organization

This thesis consists of 6 chapters, including the present chapter (Chapter 1) of introduction to the thesis topic. This chapter describes the basic concept, evolution, terminology, software and tools required for the thesis work. Chapter 2 provides

a review of past literature related to Deep Link. Chapter 3 briefly describes the architecture and methodology adopted to fulfill the current objectives. Chapter 4 describes the results in details and provide a brief discussion. Challenges and best practices are discussed in Chapter 5. Chapter 6 summarizes the conclusions and the future work. In last all the source codes are mentioned in Appendix A page.

Chapter 2

Literature Review

2.1 Research Paper 1

Yifei Ma et al. [1] addressed Deep links are an essential tool for supporting programmed execution at specified app locations. By enabling users to go to specific areas or features inside the app, the release of deep links for applications is essential for enabling the programmable execution of apps. This feature improves the user experience and makes it possible to navigate the app ecosystem without any interruptions. Due to the tremendous complexity of contemporary apps, enabling programmable execution for every place inside an app manually poses substantial difficulties. Modern programmes now have complex user interfaces, numerous functionalities, and enormous volumes of data. The requirement to satisfy user needs for sophisticated functionality, integrated services, and individualised experiences gives rise to these difficulties. By providing a structured and effective method, authors minimise the difficulties involved while adding such functionality. To minimise these challenges, the authors gave an idea of a tool, which they named "Aladdin". This is an automation tool that combines source code analysis and behaviour analysis.

The authors studied deep links popularity, its coverage, and the proposed tool by taking 20 apps from the Google Play Store. One limitation of this paper is that the authors analysed the framework in the Android application only and not in the iOS application.

2.2 Research Paper 2

[2] noted that the primary method for using portable devices to access the Internet has evolved into a mobile application (app). Accessing a particular "content page" within an app necessitates navigating through various actions within the app from the landing page. In contrast, unlike the web, where each web page has a publicly accessible URL that allows direct access. The rigidity that frequently restricts the potential value added by "linked data" across apps limits interoperability among applications, which has in fact been a source of worry. Although the idea of linked data contains enormous potential for connecting and creating meaningful links across many applications, a number of obstacles prevent its full realisation. Achieving seamless interoperability is difficult due to the vast range and complexity of applications. Different programming languages, frameworks, technologies, and architectural styles are used to create apps. These variations may cause compatibility problems and obstruct the seamless integration of data and capabilities. It can also be difficult to maintain compatibility across connected data as applications change and expand. The navigation graph of Android apps can easily be built using a dynamic approach. The authors guaranteed that the basic principles of the methods could be customised for use on other platforms. User-specific cases that require user credentials to show user-related data are limitations of the navigation analyser.

2.3 Research Paper 3

[3] addressed malicious URLs that might accidentally or knowingly be included in the code or resources of Android applications. Researchers help preserve the security of the app ecosystem, stop fraudulent activity, and increase user confidence by discovering and reporting such URLs. The most recent attack methods and trends may be learned by examining and identifying malicious URLs in Android applications. With the use of this information, developers and researchers may enhance security procedures, create practical defences, and keep up with developing threats in the Android ecosystem. Authors use deep learning techniques as well as conventional machine learning approaches to classify malicious and benign URLs. Authors use conventional machine learning methods to learn patterns and make predictions from input. Later, they used the Deep Bidirectional Long Short-Term Memory (DBLSTM) classifier, a specific kind of recurrent neural network, for modelling. Feature engineering is not necessary for the Deep Bidirectional Long Short Term Memory algorithm, which reduces the time compared to traditional machine learning methods. This technique can be widely utilised to detect attacks that are initiated by different domain generation algorithms. This technique needs improvement to produce better results. DBLSTM models may be computationally costly, particularly when working with huge datasets or intricate topologies. A drawback in real-time or resource-constrained applications might be the training and inference periods, which can be much longer than for simpler models.

2.4 Research Paper 4

[4] addressed whether the transmission of sensitive data is secured or not from an Android app. The authors tried to find out whether the transmission was intended for the user or not. It is more likely to be user-initiated when a data-sharing feature is started or when a user grants authorization for a particular data transmission. The transfer of a file, for instance, might be seen as user-intended if the user actively chooses to share it via a messaging programme. Due to poor design, deceptive prompts, or hidden functionality in programmes, users may unwittingly start data transfers. In these circumstances, looking at user engagement and behaviour patterns might offer more information. For example, it may be a sign of privacy leakage if a user repeatedly avoids using a certain function or tries to stop data transfers. This indicates that the communication may not have been user-intended. The authors proposed the Event-Space Constraint Guided Symbolic Execution (ESCGSE) technique for Android apps. In the field of Android app examination, this strategy is used to investigate the execution routes and behaviour of an app. It combines event-driven analysis, which focuses on events like user interactions, system calls, and external inputs, with symbolic execution, which uses programme symbols rather than physical values. The proposed app validation framework, 'AppIntent' can extract app inputs that show user interactions in a reasonable amount of time. Additionally, with the help of dynamic analysis, 'AppIntent' has the ability to logically show the transmission's context information for sensitive data. Some limitations of this paper include: First, the proposed framework 'AppIntent' doesn't support native code, so it won't be able to capture privacy leakages. Second, Android *Instrumentation-TestRunner* does not directly support the instrumentation of network input, which limits the ability to simulate network inputs during symbolic execution within the

framework. Symbolic execution, by itself, may not provide a straightforward solution for simulating network inputs and analysing their impact on privacy.

2.5 Research Paper 5

[5] focused on the component hijacking problem, one type of vulnerability that frequently affects Android applications. Attackers might use these weaknesses to their advantage by exploiting insecure applications, which can then leak private data and risk the security of Android devices' data. There are some reasons why relying solely on developers to fix the vulnerabilities is difficult. It takes too much time for them to validate each weakness and develop a patch for it. They may not have the necessary expertise to fix the issue. Component hijacking occurs when a victim app receives tampered data from a non-legitimate app that extracts all the sensitive data or can tamper with the sensitive data and store it in a location. The authors proposed an automatic patch generation technique. First, the Dalvik Bytecode executable file (DEX) is converted into an intermediate representation (IR). Then they found out about component hijacking flows, which they named "taint slice. They applied the patch statement and generated a new IR. Later, they optimised the inserted patch statement, generated the final .apk file, and released it. Conversion from a DEX file to a JAR file may not always be successful. Automated patch mechanisms help users fix the problem without waiting for developers, while patch generation mechanisms are more costly when considering memory consumption and runtime costs.

2.6 Research Paper 6

[6] analysed Android developers lunching of a webpage inside an application with the help of Web View. Web view and browser are two different things. Although this sophisticated interaction makes it easier for developers to support several platforms, it leaves programmes vulnerable to attack. Authors build a framework called ‘Bifocals" to find these flaws and estimate how often insecure code is used. Based on the research, they proposed a change to the Web View security settings that shields susceptible applications while imposing little additional work on developers. The dataset the authors used for evaluation is two years old. The result may differ in the recent Android application. The potential for false negatives is one of the drawbacks of the crawling methodology. The static analysis approach they used here sometimes does not retrieve the full URI. The proposed framework can analyse both applications and web content to detect vulnerabilities. Bifocals can limit the chance of being exposed to malicious JavaScript with less developer effort.

2.7 Research Paper 7

[7] addressed the subject of component hijacking related to Android links, especially deep links, which is the main problem of this paper. The security issues associated with component hijacking, in which bad actors leverage flaws in deep link processing to lead users to unexpected or harmful app components, are highlighted by the authors. Component hijacking may occur as a result of incorrect deep link data validation, insufficient permission checks, or insecure deep link user input handling. Authors also put lights on the scheme URL, app link, and intent link. This paper

emphasises the requirement for safe deep link implementation practises while drawing attention to the component hijacking issue in Android connections. Keeping in mind the need to prevent link hijacking, the authors carried out the first extensive analysis of the existing mobile deep link ecosystem. Their objective is to identify and assess vulnerabilities related to link hijacking in both web and mobile applications, with the aim of quantifying the extent and impact of these vulnerabilities [7]. Authors use some applications to extract link registration entries (URIs) and then empirically track how often those URIs are used on websites. They create link collision groups for apps that register the same URIs in order to identify hijacking threats. For the sake of user privacy and app security, over-permission issues can be mitigated. When requesting permissions, just specify the minimum set necessary to carry out a certain action or access a particular resource on the device. Researchers mitigate unverified link issues, which also pose an equal threat to link hijacking. This paper is limited to Android links, and compared to the size of the web and the Android app market, the measuring scope is still small.

2.8 Research Paper 8

[8] addressed the possibility of malware cases rising due to the open environment of Android and the presence of third-party app stores. An essential part of an Android app that provides crucial details about the programme's services, components, and permissions is the manifest file. Researchers look for unusual or harmful behaviour in the manifest file, such as a high number of permissions or the inclusion of unidentified or hidden components. The manifest file for Android apps lists the permissions that are necessary. Authors look at the permissions requested to find instances where apps ask for too many or unnecessary permissions, which may be a sign of

malicious behaviour or privacy violations. To access user data and device resources, Android apps communicate with a variety of APIs. Researchers can spot suspicious or unauthorised access to sensitive data by following the API calls that apps make. Malware samples frequently leave characteristic marks in the manifest file or with certain API calls, which may be used to develop signatures for malware identification. The authors provide a feature-based technique to give static analysis a framework for identifying Android malware. The methods used the static data of Android apps to characterise the behaviour. DroidMat works well because it can tell the difference between different types of Android malware and their intended uses. It can also save the expense of setting up an environment and the manual work involved in inquiry by not having to dynamically examine the behaviour of an Android application in a sandbox or through emulation. Static analysis makes use of the ability to examine an app's source code without actually running it. Some malware uses obfuscation or anti-analysis tactics, which make it more difficult for static analysis to properly comprehend the app's real behaviour.

2.9 Research Paper 9

[9] addressed security flaws that have increased as a result of the growing use of Android-based applications, especially how these applications will handle links. Malicious links can be used to attack user devices, damage their security, and violate their privacy. Because of this, there is a demand for efficient static analysis approaches that can discover possible weaknesses in how links are handled in Android applications. To find potential weaknesses attached to link handling in Android applications, static analysis tools examine the source code of those applications. This requires taking a look at the code's parsing, validation, and processing of links. In

order to pinpoint the locations where connections are accessed, changed, or provided to external components, static analysis tools monitor the data flow across the programme. This aids in spotting possible weaknesses, such as poor link data validation or sanitization; the authors demonstrate malware protection methods for Android-based mobile phones based on static analysis and N-grammes. The classification of malicious and beneficial mobile applications is done using this approach. Static analysis enables the early detection of link-related vulnerabilities in Android applications, enabling developers to prevent security problems before they arise. Static analysis may go through the whole source, including dependencies and libraries from third parties, to make sure that all link-related vulnerabilities are fixed. Static analysis just examines the source code; therefore, it could miss some contextual factors like runtime behaviours or environment-specific configurations that have an impact on how links are handled. Links that only become vulnerable at runtime, such as those brought about by user inputs or server answers, are not taken into consideration by static analysis approaches.

2.10 Research Paper 10

[10] focused on Android manifest files to identify potential malicious behaviour and patterns. The manifest's defined intents and intent filters specify the message-sending and receiving capabilities of components. Determine the communication channels the programme uses by analysing the intents and filters. Malware can sign up to receive a certain kind of intent or utilise unique intent actions to interact with other harmful components or outside parties. It is possible to find probable virus communication patterns by looking at these intents and filters. The authors proposed 'Manilyzer," which is a static analysis and automatic disassembly tool. The

goal of the tool is to efficiently identify malware in the Android manifest file. Its design was inspired by the research of a large number of manifest files, including those for malicious and good apps. The Android manifest file enables the early detection of possible malware through static analysis. Suspicious patterns and signs of malicious behaviour can be found before installation or execution. The ability to detect new or undiscovered malware variants is strength of manifest analysis. It concentrates on the actions and patterns shown in the manifest, enabling the identification of unidentified dangers. The information contained in the manifest file is the only source of data for manifest analysis. It doesn't consider user interactions, runtime behaviours, or outside variables. Having a valid and accessible manifest file is essential for efficient manifest analysis. It is occasionally possible for third-party app stores or distribution platforms to remove or edit the manifest, which makes it challenging for analysis.

Chapter 3

Architecture & Methodology

3.1 Architecture

3.1.1 WebView App flowchart

In a mobile app, there are a number of architectural factors to take into account while constructing a WebView object. See Figure 3.1 . A WebView object is first created by the native mobile app framework as the first step in the architecture. Providing features including page loading, JavaScript execution, and navigation, the WebView object serves as a container for presenting online information. The required parameters, including JavaScript enablement, cache management, and the definition of additional WebView attributes, are defined when it is created. It is then necessary to load a URL into the WebView object that has just been constructed. This entails launching a network request to get the web data related to the given URL. In order to handle HTTP/HTTPS requests, control cookies, and cache resources, the WebView interacts with the network layer. The web content is then shown in the WebView when the server's response has been processed. Handling

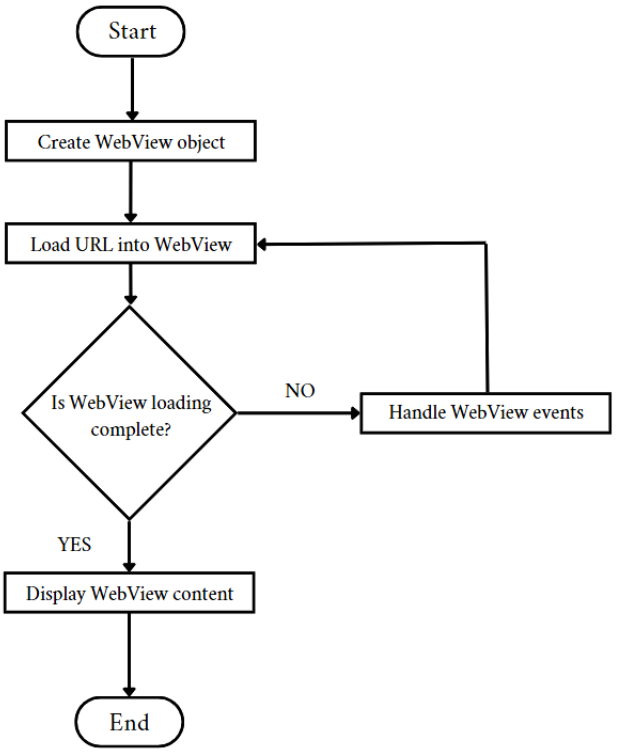


FIGURE 3.1: WebView App Creation Framework

WebView events is a part of the WebView architecture as well. The completion of a page load, navigation requests, form submissions, JavaScript callbacks, or failures are examples of events. Event delegates or listeners that catch these events are built into the WebView object. Developers may respond appropriately since the WebView automatically launches the callback or event handler in the app logic when an event happens. An event such as the conclusion of a page load may start additional processes or modify the user interface. The WebView component must be included in the app's user interface in order to display the WebView content. The WebView object offers ways to present web material in a specific layout or view. The UI is managed by the native mobile app framework, which also places the WebView component on the app's screen and supports user interactions with it. The rendered content of the WebView, including HTML, CSS, pictures, and media, is shown to the user, resulting in an intuitive surfing experience inside the app.

3.1.2 Instance Running Flowchart

An activity is a key component of the user interface (UI) of an application when developing for Android. Users can interact with the user interface on a single screen that it depicts. Activities often correlate to a particular feature or user process inside the programme, and they act as the entry points for users to access various portions of an app. A Java class that extends the android.app is how an activity is implemented.the more popular *androidx.appcompat.app* or the activity base class.Android Support Library class called *AppCompatActivity*. The methods and callbacks in the activity class determine the behaviour and lifespan of the activity.

When an activity becomes visible to the user but is not yet in the forefront, the *onStart()* function is invoked. It is generally employed to initiate or resume processes that ought to run only when an action is both visible and interactive. The activity loses attention and moves out of the foreground, which triggers the call to the *onPause()* function. Prior to the activity going into the background or maybe being deleted, there is a chance to preserve data, release resources, or carry out other cleanup tasks. See Figure 3.2 . The *onStop()* function is triggered when the user's view of the activity becomes invisible. It can be invoked when an activity is either ending or starting a new one. This function enables the execution of specific actions or cleanup tasks that are relevant when an activity is not visible. Upon termination of the activity, the *onDestroy()* function is called. This provides an opportunity to release resources or complete any necessary cleanup tasks before the activity is completely removed from memory. Because the system occasionally ends the activity without using this function, it is not always called. When an activity is relaunched after being stopped, the *onRestart()* function is invoked. It is a sign that after briefly stopping, the action is starting to move back into the forefront. Following this function are the *onStart()* and *onResume()* methods. If an activity is briefly deleted

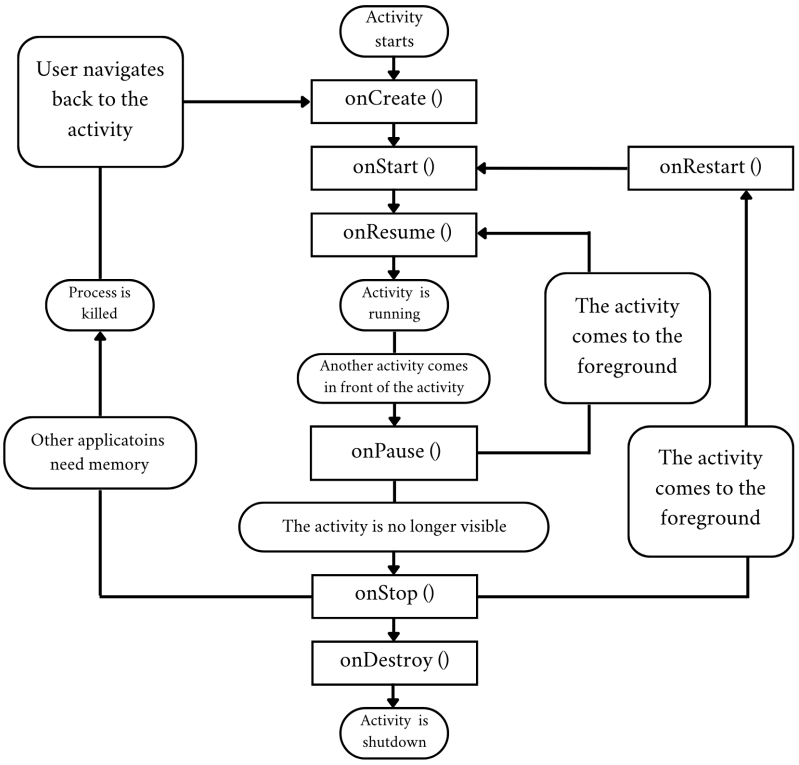


FIGURE 3.2: Android Activity Lifecycle

and then restored by the system, these techniques are used to save and restore its state. You can save any required data by using *onSaveInstanceState()* prior to the activity being paused or restarted. When the activity is being rebuilt, the function *onRestoreInstanceState()* is invoked, enabling you to restore the previously saved state. These lifecycle methods include hooks that let you carry out particular activities at various points in the lifespan of the activity. It's essential to comprehend and apply these techniques appropriately in order to manage resources, save data, and guarantee a smooth user experience while the activity switches between various stages.

3.2 Methodology

3.2.1 Lab Setup

Before going to create a web-view app make sure to install and configure all the required software and tools. Choose a development environment based on your preference, such as Windows, macOS, or Linux. In this paper, Windows 10, an i7-10th generation with 16 GB of RAM and a 256GB SSD, is used. But the minimum capacity of the operating system has to be Windows 10 or 11, Intel core 5 or 7, 10th generation or higher with 12 to16 GB RAM. Install an IDE for app development. In this paper, Android Studio is used, as is the Android SDK, which includes the necessary tools, libraries, and emulators for testing. Web technologies like HTML, CSS, and JavaScript are used to create the website. For testing and communicating with the Android emulator, PowerShell is used in this paper. Android Debug Bridge is another tool to connect the Android emulator for security testing.

3.2.2 WebView App Creation

There are several steps involved in creating a web-view application. The procedure is outlined in the following paragraphs. Building a web-view app requires a URL. In this paper a website is created and hosted on the free web server called 'netlify.com'. The URL used to create a web-view app is *'https://rgniyd.netlify.app'*.

To setup web-view in your app, open the layout file, i.e., *activity_ main.xml*, and add a web-view element. Customise the web-view's attributes as needed. Open the *MainActivity.java* file and set up the web-view configuration. Retrieve the web-view instance from the layout file. Enable JavaScript and other web-view settings. Then load the initial URL into the web-view. To handle different web-view events, such

as page loading, errors, and navigation, implement a web-view client. In order to handle certain URLs or deep links, override the *shouldOverrideUrlLoading* function. Set up your app to handle deep links or load only certain portions depending on the URL that was loaded. To go through the web-view history, you may optionally manage the Back button behaviour using the *onBackPressed* function.

Declare an intent filter for the activity that hosts the web-view. Make sure to add these three things in the intent filter i.e. *action*, *category* and *data* in *AndroidManifest.xml* file.

Connect the Android device or set up an emulator to run the app. For the execution of the experiments conducted in this paper, a Nexus 5 smartphone model with a 2.3GHz CPU and 2GB RAM, running Android OS 6, was utilized. Build and run the app from Android Studio then release it into the preferred location.

Test the web-view functionality by loading different web pages. Verify the deep links in order to open it in the application and ensure proper navigation.

3.2.3 Testing Deep Links

Perform a set of actions in order to test deep links. To start, choose the deep link URL you wish to test. It must follow the *'scheme://host/path?parameter'* format. As soon as you receive the URL, set up a test environment by making sure all the appropriate hardware, emulators, and target apps are loaded and ready for testing. The next step is to create the deep link by building the URL with the appropriate parameters or route segments. All necessary query parameters and route segments must be included. You have a few options for triggering the deep link once it is prepared. If a deep link is available through a website or another programme, one way to access it is to click on the deep link URL. The URL should open in the desired application when clicked. If you have access to the command-line interface, using

the ADB command is an additional choice. You may transmit an intent with the deep link URL by connecting the device or emulator and executing the necessary ADB command. As an alternative, you may initiate the deep link programmatically by utilising the appropriate APIs if you have access to the app's source code or are constructing an app. Once the deep connection has been activated, watch to see how the destination app reacts. Make sure the deep link opens the appropriate page or executes the desired action in accordance with the deep link URL. It's important to investigate diverse possibilities when testing. Test a variety of deep linkages, including legitimate, improper, and edge instances. Check to see if the app processes them properly and offers proper feedback or error management.

Testing the deep link undergoes two different methods. One is static analysis, and another one is dynamic analysis. To assess the potential risks related to exploiting deep links, it is essential to consider the Android version on which the app is operating. This can be determined by examining the Android Manifest file to verify if the *minSdkVersion* is 31 or higher. By using apktool to decode the app and analyse the Android Manifest file, you can quickly identify whether deep links (with or without custom URL schemes) are established. Look for intent-filter components in the Android Manifest file. Deep links must be confirmed in order to be regarded as App connections, even if they have the *android:autoVerify="true"* property. Check for any potential setup errors that can prevent thorough verification.

Chapter 4

Results & Discussions

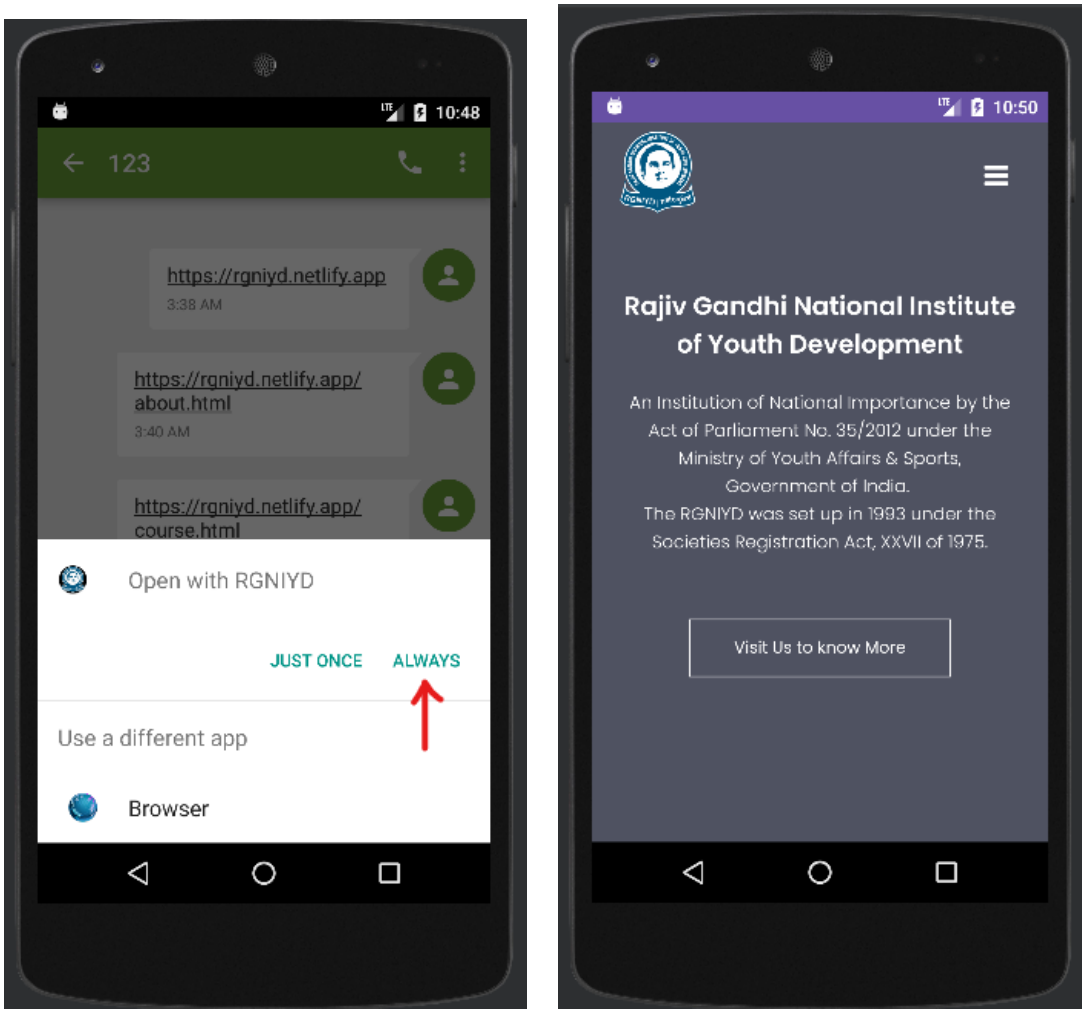
The goal of this research project was to develop an Android-based Web-View app and assess its usability, performance, and user interface. The Web-View app provided users with a native app interface through which they could browse and interact with online information.

The app's WebView component was successfully included, allowing the app's UI to render web information. In order to enable users to traverse through web pages, appropriate navigation controls, such as back and advance buttons, were developed. The programme let users to manually input URLs or click links inside the WebView to load online pages. To guarantee correct rendering and suitable presentation, the app was tested with a variety of web sites that contained various forms of material, including text, photos, videos, and interactive components. The software displayed online material properly, maintaining the original design and visual aspects.

Figure 4.1 shows the successful installation of the app in the Android emulator. I store different links in the Android message box to check the redirection. When

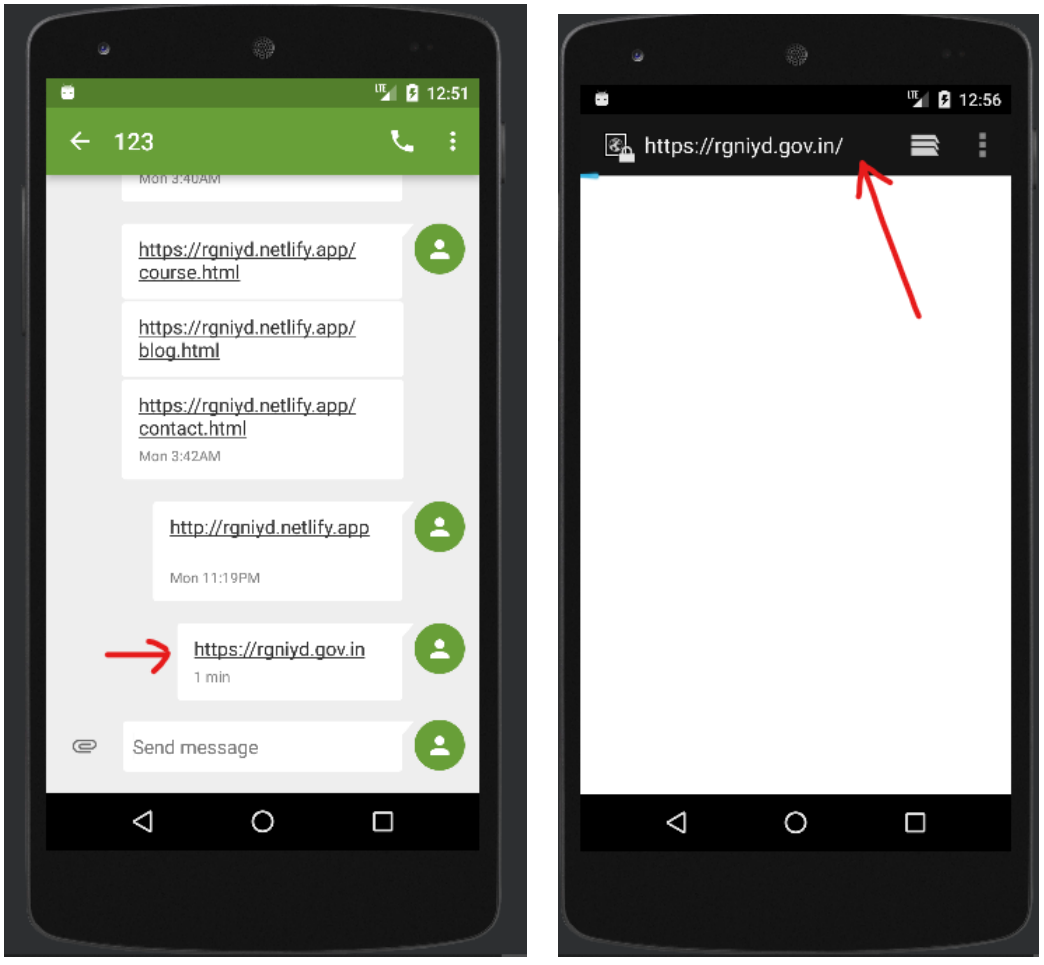
the user clicks any one of these links for the first time, the android will show an ambiguous message (see in Figure 4.2); once the user clicks on the always' button, they give permission for the links to open in the specific application. We can see the successful launch of the app from the specific link. And Figure 4.3, shows that if an app receives an attempt to open it with a scheme or host that is not specified in the intent filter of the manifest file, instead of launching the app, it redirects the request to the web browser of the emulator.

FIGURE 4.1: Install Application



(a) Ambiguous Message. (b) Lunch of the app.

FIGURE 4.2: Seeking User Permission



(a) Link with Different Host. (b) Browser.

FIGURE 4.3: Different Host Link Opened in Browser

Chapter 5

Challenges & Best Practices

5.1 Challenges in Deep Linking

In mobile marketing, deep linking is a crucial element that promotes involvement, persistence, and profits. The need to guarantee that deep links function properly in every medium is actually more important than ever. However, concerns with data transfer, security constraints, link routing, device compatibility, app installation status, and other factors may make its deployment more difficult. To navigate these issues successfully, this section offers detailed insights.

a) **Platform or Device Compatibility:** Device fragmentation is indeed a significant challenge when it comes to deep linking, particularly in the context of the Android and iOS platforms. This fragmentation encompasses various aspects, including operating systems, screen sizes, resolutions, and hardware capabilities, which can make it difficult to ensure consistent and seamless deep linking experiences across different devices. For various platforms as well as for

successive iterations of the same operating system, deep links must be set individually. Deep links may function flawlessly on one platform or OS version but malfunction on another as a result of this inconsistency. The two most popular mobile operating systems, iOS and Android, each offer a unique deep linking mechanism. Android uses Android App Links, but iOS utilises Universal Links. These weren't always the norm, though. Prior to Android 6.0 Marshmallow, deep linking in Android mostly utilised unique URL schemes. But this presented a number of problems, including the possibility of URL scheme issues if numerous apps utilised the same scheme. To solve this, Android added Android App URLs in version 6.0, allowing URLs that are particular to an app to open instantly in that app, provided that app is installed. If an app enables Android App Links and is installed on the user's device, for instance, clicking a link in an email that begins with "https://www.rgniyd.com/" will launch the app immediately. Maintaining compatibility across various Android versions presented a new problem for app developers with the advent of Android App Links. Although Android App Links are more dependable and secure, Android versions older than 6.0 do not support them. This indicates that in order to enable deep linking on these versions, programmes hoping for broad compatibility still need to use the earlier custom URL scheme technique.

b) **App installation status:** The status of user installed the app or not can have a big impact on the behaviour of deep links. Deep links that point to a web browser or app store instead of the programme itself, if the app isn't installed, may impede the user's progress. Deferred deep links are one way for marketers to get around this problem. Although the programme must initially be installed, deferred deep links enable users to be routed to specific information. Once the app is installed, they save the data sent through the

link and direct the user to the desired place. When a user interacts with a deep link, the term 'app installation status' refers to whether or not the user has your app loaded on their device. Depending on whether your app is already installed or not, numerous events might happen when a user clicks on a deep link. If installed, the deep link will open the appropriate in-app content immediately. If not, the link will often send the user to the app store where they may download your software, breaking the smooth user experience that deep linking is meant to offer. Link routing may be impacted by your app's status on a user's device, including whether it is installed, uninstalled, or operating in the background. It can be technically difficult to handle these cases effectively since each condition demands a separate routing path.

c) **Data Transmission Issue:** Data transmission across deep networks may be a challenging operation. There may be a number of problems that prevent the data from going where it should or from being properly processed by the app. The data sent over a deep link must be accurately interpreted by the app. However, problems like improper data formatting or unknown special characters might lead to processing failures and invalidate the data. It might happen that the information added to a deep link is unreliable or inaccurate. Technical problems or mistakes in the deep link generation process might be to blame for this. When there are problems with data transfer, the user experience may suffer greatly. The personalised experience you had in mind might not be realised, or users might not be sent to the desired area inside the app. This can make users less engaged and aggravated, which might affect conversion rates and general app performance.

d) **Security Restrictions:** As OS makers take methods to stop harmful programmes from abusing deep links, security constraints may have an influence

on deep linking. The user experience may be impacted by these restrictions by way of extra prompts. Following recommended deep linking procedures, such as making sure your website association files are properly configured and informing users as to why particular prompts may occur, will assist to prevent these problems. Before being routed from a deep link to an app on various platforms, users are prompted to confirm. This extra step might obstruct the user's trip and perhaps affect conversion rates. Certain data cannot be sent through deep networks to reduce the threat of data breaches. The degree of customised service you may offer may be impacted by this.

5.2 Best Practices

- Developers use a variety of techniques to overcome the obstacles, including integrating contemporary Android programme Links or iOS Universal Links as well as unique URL schemes into your programme. This will allow your app to fall back to the specific URL scheme on earlier versions of the device while still using the better technique on devices that support it. Make sure your deep connections function properly in various settings by thoroughly testing them on a range of hardware and OS versions. In your testing matrix, incorporate both recent and historical releases of Android and iOS.

- Deferred deep linking implementation is a two-step procedure that involves both your marketing team and the developers of your programme. Your marketing team must comprehend the customer path and make plans for how to employ postponed deep links successfully. Deferred deep connections must work as planned across all platforms and devices, thus your developers must simultaneously include the technology into your programme.

- Maintain open channels of communication, include them in the planning process, and solicit their opinions on link routing tactics. To discover and resolve any possible link routing issues, test your deep connections across a variety of devices, operating systems, and app states. The marketing and QA teams should work together to make sure that the user experience supports your marketing objectives.

- Contextual deep connections enable developers to provide consumers with a much more individualised and focused app experience immediately when they launch an app. Developers may use these connections to create features that send customers right to a unique welcome page after downloading or let them instantly upload a coupon. Additionally, they make it possible for marketers to compile data on the effectiveness of marketing initiatives and advertising campaigns.

Chapter 6

Conclusion & Future Work

6.1 Conclusion

Deep links are now a crucial component of creating modern mobile apps since they offer a smooth and practical approach for users to access particular app information. Deep links enable users to access the app's content directly from outside sources like websites, emails, or other applications by linking certain app screens or actions to specific URLs. Deeplinks' correctness and operation are ensured through thorough testing, providing a positive user experience. The establishment of deep links is an essential component of app development. Developers may make it simple for users to access particular material or carry out particular activities by specifying and tying deep links to specific app screens or actions. The deep link URLs must be registered in the app's manifest file, intent filters must be set up, and the app must handle incoming deep link intents.

Designing a coherent and relevant URL structure is crucial for deep link deployment. It should give a direct route to particular screens or activities and represent the hierarchy of the app's content. The user experience is enhanced by a well-designed URL structure that enables consumers to comprehend the deep link's purpose. To handle deep link intents, the app's manifest file's intent filters must be properly configured. The host names, routes, and schemes that the app may handle are specified by the intent filters. Developers may make sure that the app reacts correctly when a deep link is opened by specifying intent filters. Deep links should direct visitors to the appropriate app panels or activities. The incoming deep link intents must be handled by developers, who must then take the appropriate steps, such as launching a certain activity or fragment or carrying out a particular operation based on the deep link data. Testing is necessary to guarantee proper operation, check deep link data, manage edge situations, and deliver a consistent user experience across all devices and circumstances. Deep connections may be used to increase app usage, boost user engagement, and boost user happiness by developers when they build and test them properly.

6.2 Future Work

At the moment, deep links are usually defined statically in the manifest file of the app. In the future, apps could be able to create deep connections on the fly based on the context or preferences of the user thanks to improved dynamic deep linking capabilities. Because of this, deeper connection experiences might be more individualised and contextual. The security of deep link interactions is essential because they may provide access to sensitive information or open up app functionality. Future

work may concentrate on enhancing security controls for deep link handling, including safeguards for user privacy, procedures for secure deep link source validation, and prevention of harmful deep link assaults. Establishing universal standards for deep link structures, naming conventions, and metadata formats could simplify deep link implementation and improve interoperability between apps and platforms. This would make it easier for developers to adopt and integrate deep linking capabilities into their apps.

Appendix A

Source Code of WebSite & WebView App

.1 index.html Code

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>RGNIYD Clone</title>
    <link rel="stylesheet" href="style.css">
    <link rel="stylesheet" href=""><link rel="preconnect"
    href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com"
    crossorigin>
    <link href="https://fonts.googleapis.com/css2?
    family=Poppins:wght@100;200;300;400;600;700&display=swap" rel="stylesheet">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
    awesome/4.7.0/css/font-awesome.min.css">
</head>
```

```
<body>
    <section class="header">
        <nav>
            <a href="index.html"><img src="Img/logo1.png"></a>
            <div class="nav-links" id="navLinks">
                <i class="fa fa-times" onclick="hideMenu()"></i>
                <ul>
                    <li><a href="index.html">HOME</a></li>
                    <li><a href="about.html">ABOUT</a></li>
                    <li><a href="course.html">COURSE</a></li>
                    <li><a href="blog.html">BLOG</a></li>
                    <li><a href="contact.html">CONTACT</a></li>
                </ul>
            </div>
            <i class="fa fa-bars" onclick="showMenu()"></i>
        </nav>
        <div class="text-box">
            <h1>Rajiv Gandhi National Institute of Youth Development</h1>
            <p>An Institution of National Importance by the Act of
            Parliament No. 35/2012 under the Ministry of Youth Affairs & Sports,
            Government of India. <br>The RGNIYD was set up in 1993 under the
            Societies Registration Act, XXVII of 1975.</p>
            <a href="about.html" class="hero-btn">Visit Us to know More</a>
        </div>
    </section>

<!-----Courses ----->

    <section class="course">
        <h1>Courses We Offer</h1>
        <p>PG Programmes offered by us</p>
        <div class="row">
            <div class="course-col">
                <h3>Data Science</h3>
                <p>The masters degree concentrations and
                graduate certificate in Data Analytics at RGNIYD is one
                of the leading research<br>and teaching institutions and it
                can help you develop sought after expertise in data analytics.
                </p>
            </div>
            <div class="course-col">
```

```
                <h3>AI & ML</h3>
                <p>The masters degree concentrations and graduate certificate
                in AI & ML at RGNIYD is one of the leading research <br> and
                teaching institutions and it can help you develop sought after
                expertise in AI & ML.
                </p>
            </div>
            <div class="course-col">
                <h3>CyberSecurity</h3>
                <p>The masters degree concentrations and graduate certificate
                in CyberSecurity at RGNIYD is one of the leading research <br>
                and teaching institutions and it can help you develop sought after
                expertise in CyberSecurity.
                </p>
            </div>
        </div>
    </section>

<!--------- Call to action ------>

    <section class="cta">
        <h1>Enroll For Our Various Post Graduate Courses<br>
        Anywhere From The India.</h1>
        <a href="contact.html" class="hero-btn">CONTACT US</a>
    </section>

<!-------- Footer ----------->

    <section class="footer">
        <h4>About Us</h4>
        <p>RGNIYD, Sriperumbudur, Tamil Nadu is an Institution of National
        Importance by the Act of Parliament No. 35/2012 under the Ministry of
        Youth Affairs & Sports, Government of India. <br>The RGNIYD was set up
        in 1993 under the Societies Registration Act, XXVII of 1975.
        </p>
        <div class="icons">
            <a href="https://www.facebook.com/rgniyd/"
            target="_blank"><i class="fa fa-facebook"></i></a>
            <a href="https://twitter.com/rgniydt"
            target="_blank"><i class="fa fa-twitter"></i></a>
            <a href="https://instagram.com/students__of__rgniyd"
```

```
            target="_blank"><i class="fa fa-instagram"></i></a>
            <a href="https://www.linkedin.com/company/placement-cell-rgniyd/"
            target="_blank"><i class="fa fa-linkedin"></i></a>
        </div>
        <p>Made with<i class="fa fa-heart-o"></i>by Dhruti Ranjan</p>
    </section>

<!Javascript for Toggle Menu>

    <script>
        var navLinks = document.getElementById("navLinks");

        function showMenu()
        {
            navLinks.style.right = "0";
        }
        function hideMenu()
        {
            navLinks.style.right = "-200px";
        }
    </script>
    </body>
</html>
```

.2 about.html Code

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>RGNIYD Clone</title>
    <link rel="stylesheet" href="style.css">
    <link rel="stylesheet" href=""><link rel="preconnect" href=
    "https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
    <link href="https://fonts.googleapis.com/css2?family=Poppins:wght@100;
```

```
    200;300;400;600;700&display=swap" rel="stylesheet">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/
    font-awesome/4.7.0/css/font-awesome.min.css">
</head>

<body>
    <section class="sub-header">
        <nav>
            <a href="index.html"><img src="Img/logo1.png"></a>
            <div class="nav-links" id="navLinks">
                <i class="fa fa-times" onclick="hideMenu()"></i>
                <ul>
                    <li><a href="index.html">HOME</a></li>
                    <li><a href="about.html">ABOUT</a></li>
                    <li><a href="course.html">COURSE</a></li>
                    <li><a href="blog.html">BLOG</a></li>
                    <li><a href="contact.html">CONTACT</a></li>
                </ul>
            </div>
            <i class="fa fa-bars" onclick="showMenu()"></i>
        </nav>
        <h1>About Us</h1>
    </section>

<!-------- about us content -------->

    <section class="about-us">
        <div class="row">
            <div class="about-col">
                <h1>India's No. 1 Central University</h1>
                <p>The Rajiv Gandhi National Institute of Youth Development (rgniyd),
                Sriperumbudur, Tamil Nadu, is an Institution of National Importance
                by the Act of Parliament No.35/2012 under the Ministry of
                Youth Affairs & Sports, Government of India.
                The RGNIYD was set up in 1993 under the Societies Registration
                Act, XXVII of 1975. <br><br> The RGNIYD functions as a vital
                resource centre with its multi-faceted functions of offering academic
                programmes at Post Graduate level encompassing various dimensions of
                youth development, engaging in seminal research in the vital areas
                of youth development and coordinating Training Programmes for state
                agencies and the officials of youth organisation, besides
```

```
                the Extension and Outreach initiatives across the country.<br><br>
                The Institute functions as a think-tank of the Ministry and premier
                organization of youth-related activities in the country. As the
                apex institute at the national level, it works in close cooperation with
                the NSS, NYKS and other youth organizations in the implementation
                of training programmes. The Institute is a nodal agency for training youth
                as a facilitator of youth development activities in rural,
                urban as also tribal areas.<br><br> The RGNIYD serves as a youth
                observatory and depositary in the country thereby embarking on youth
                surveillance on youth-related issues. It has a wide network with various
                organizations working for the welfare and development of young
                people and serves as a mentor.
                </p>
                <a href="course.html" class="hero-btn red-btn">EXPLORE NOW</a>
            </div>
            <div class="about-col">
                <img src="https://pedestaltechnology.com/wp-
                content/uploads/2020/04/videoconference.jpg">
            </div>
        </div>
    </section>

<!--------- Footer --------->

<section class="footer">
    <h4>About Us</h4>
    <p>RGNIYD, Sriperumbudur, Tamil Nadu is an Institution of National Importance by
        the Act of Parliament No. 35/2012 under the Ministry of Youth Affairs & Sports,
        Government of India. <br>The RGNIYD was set up in 1993 under the Societies
        Registration Act, XXVII of 1975.
    </p>
    <div class="icons">
        <a href="https://www.facebook.com/rgniyd/" target="_blank"><i class=
        "fa fa-facebook"></i></a>
        <a href="https://twitter.com/rgniydt" target="_blank"><i class=
        "fa fa-twitter"></i></a>
        <a href="https://instagram.com/students__of__rgniyd" target="_blank"><i class=
        "fa fa-instagram"></i></a>
        <a href="https://www.linkedin.com/company/placement-cell-rgniyd/" target="_blank">
        <i class="fa fa-linkedin"></i></a>
    </div>
```

```
    <p>Made with<i class="fa fa-heart-o"></i>by Dhruti Ranjan</p>
</section>

<!------ Javascript for Toggle Menu ------>

<script>
    var navLinks = document.getElementById("navLinks");

    function showMenu()
    {
        navLinks.style.right = "0";
    }
    function hideMenu()
    {
        navLinks.style.right = "-200px";
    }
</script>
</body>
</html>
```

.3 Course.html Code

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>RGNIYD Clone</title>
    <link rel="stylesheet" href="style.css">
    <link rel="stylesheet" href=""><link rel="preconnect" href=
    "https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
    <link href="https://fonts.googleapis.com/css2?family=Poppins:wght@100;
    200;300;400;600;700&display=swap" rel="stylesheet">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/
    font-awesome/4.7.0/css/font-awesome.min.css">
</head>
```

```
<body>
    <section class="sub-header">
        <nav>
            <a href="index.html"><img src="Img/logo1.png"></a>
            <div class="nav-links" id="navLinks">
                <i class="fa fa-times" onclick="hideMenu()"></i>
                <ul>
                    <li><a href="index.html">HOME</a></li>
                    <li><a href="about.html">ABOUT</a></li>
                    <li><a href="course.html">COURSE</a></li>
                    <li><a href="blog.html">BLOG</a></li>
                    <li><a href="contact.html">CONTACT</a></li>
                </ul>
            </div>
            <i class="fa fa-bars" onclick="showMenu()"></i>
        </nav>
        <h1>Our Courses</h1>
    </section>

<!------- Course content ----------->

    <section class="course">
        <h1>Courses We Offer</h1>
        <p>PG Programmes offered by us</p>
        <div class="row">
            <div class="course-col">
                <h3>Data Science</h3>
                <p>The masters degree concentrations and graduate certificate in
                Data Analytics at RGNIYD is one of the leading research <br>
                and teaching institutions and it can help you develop sought after
                expertise in data analytics.
                </p>
            </div>
            <div class="course-col">
                <h3>AI & ML</h3>
                <p>The masters degree concentrations and graduate certificate
                in AI & ML at RGNIYD is one of the leading research <br>
                and teaching institutions and it can help you develop sought after
                expertise in AI & ML.
                </p>
```

```
            </div>
            <div class="course-col">
                <h3>CyberSecurity</h3>
                <p>The masters degree concentrations and graduate certificate
                in CyberSecurity at RGNIYD is
                    one of the leading research <br>and teaching institutions
                    and it can help you develop sought after expertise in CyberSecurity.
                </p>
            </div>
        </div>
    </section>

<!-------- Footer ----------->

    <section class="footer">
        <h4>About Us</h4>
        <p>RGNIYD, Sriperumbudur, Tamil Nadu is an Institution of National
        Importance by the Act of Parliament No. 35/2012 under the Ministry of
        Youth Affairs & Sports, Government of India. <br>The RGNIYD was set up
        in 1993 under the Societies Registration Act, XXVII of 1975.
        </p>
        <div class="icons">
            <a href="https://www.facebook.com/rgniyd/"
            target="_blank"><i class="fa fa-facebook"></i></a>
            <a href="https://twitter.com/rgniydt"
            target="_blank"><i class="fa fa-twitter"></i></a>
            <a href="https://instagram.com/students__of__rgniyd"
            target="_blank"><i class="fa fa-instagram"></i></a>
            <a href="https://www.linkedin.com/company/placement-cell-rgniyd/"
            target="_blank"><i class="fa fa-linkedin"></i></a>
        </div>
        <p>Made with<i class="fa fa-heart-o"></i>by Dhruti Ranjan</p>
    </section>

<!Javascript for Toggle Menu>

    <script>
        var navLinks = document.getElementById("navLinks");

        function showMenu()
        {
```

```
            navLinks.style.right = "0";
        }
        function hideMenu()
        {
            navLinks.style.right = "-200px";
        }
    </script>
    </body>
</html>
```

.4 blog.html code

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>RGNIYD Clone</title>
    <link rel="stylesheet" href="style.css">
    <link rel="stylesheet" href=""><link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href=
    "https://fonts.gstatic.com" crossorigin>
    <link href="https://fonts.googleapis.com/css2?family=
    Poppins:wght@100;200;300;400;600;700&display=swap" rel="stylesheet">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/
    ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
</head>

<body>
    <section class="sub-header">
        <nav>
            <a href="index.html"><img src="Img/logo1.png"></a>
            <div class="nav-links" id="navLinks">
                <i class="fa fa-times" onclick="hideMenu()"></i>
                <ul>
                    <li><a href="index.html">HOME</a></li>
                    <li><a href="about.html">ABOUT</a></li>
```

```
                    <li><a href="course.html">COURSE</a></li>
                    <li><a href="blog.html">BLOG</a></li>
                    <li><a href="contact.html">CONTACT</a></li>
                </ul>
            </div>
            <i class="fa fa-bars" onclick="showMenu()"></i>
        </nav>
        <h1>Our Certificate & Post Graduate Programs For 2023-24</h1>
    </section>

<!------ Blog page content ------>

    <section class="blog-content">
        <div class="row">
            <div class="blog-left">
                <img src="Img/blog.jpg">
                <h2>Our Certificate & Post Graduate Programs For 2023-24</h2>
                <p>The Rajiv Gandhi National Institute of Youth Development (rgniyd),
                    Sriperumbudur, Tamil Nadu, is an Institution of National Government of
                    India. The RGNIYD was set up in 1993 under the Societies
                    Registration Act, XXVII of 1975.
                    <br>
                    <br>
                    The RGNIYD functions as a vital resource centre with its
                    multi-faceted functions of offering academic programmes at Post Graduate
                    level encompassing various dimensions of youth development,
                    engaging in seminal research in the vital areas of youth development and
                    coordinating Training Programmes for state agencies
                    and the officials of youth organisation, besides
                    the Extension and Outreach initiatives across the country.
                    <br>
                    <br>
                    The Institute functions as a think-tank of the Ministry
                    and premier organization of youth-related activities in the country.
                    As the apex institute at the national level, it works
                    in close cooperation with the NSS, NYKS and other youth
                    organizations in the implementation of
                    training programmes. The Institute is a nodal agency for
                    training youth as a facilitator of youth development activities
                    in rural, urban as also tribal areas.
                    <br>
```

```
            <br>
            The RGNIYD serves as a youth observatory and depositary in the country
            thereby embarking on youth surveillance on youth-related issues.
            It has a wide network with various organizations working for the
            welfare and development of young people and serves as a mentor.
        </p>

        <div class="comment-box">
            <h3>Leave a Comment</h3>
            <form class="comment-form">
                <input type="text" placeholder="Enter Name :">
                <input type="email" placeholder="Enter Email :">
                <textarea rows="5" placeholder="Your Comment :"></textarea>
                <button type="submit" class="hero-btn red-btn">POST COMMENT</button>
            </form>
        </div>

    </div>
    <div class="blog-right">
        <h3>Course Categories</h3>
        <div>
            <span>M.Sc. Computer Science (Data Science)</span>
            <span>25</span>
        </div>
        <div>
            <span>M.Sc. Computer Science (AI & ML)</span>
            <span>25</span>
        </div>
        <div>
            <span>M.Sc. Computer Science (Cyber Security)</span>
            <span>25</span>
        </div>
        <div>
            <span>M.Sc. Mathematics</span>
            <span>20</span>
        </div>
        <div>
            <span>M.Sc. Applied Psychology</span>
            <span>35</span>
        </div>
        <div>
```

```
                    <span>M.A. English</span>
                    <span>22</span>
                </div>
                <div>
                    <span>M.A. Sociology</span>
                    <span>30</span>
                </div>
                <div>
                    <span>M.A. Development Studies</span>
                    <span>15</span>
                </div>
                <div>
                    <span>M.A. Public Administration</span>
                    <span>17</span>
                </div>
                <div>
                    <span>M.S.W. (Youth and Community Development)</span>
                    <span>40</span>
                </div>
            </div>
        </div>
    </section>

<!-------- Footer ----------->

    <section class="footer">
        <h4>About Us</h4>
        <p>RGNIYD, Sriperumbudur, Tamil Nadu is an Institution of National
        Importance by the Act of Parliament No. 35/2012 under the Ministry of
        Youth Affairs & Sports, Government of India. <br>The RGNIYD was set up
        in 1993 under the Societies Registration Act, XXVII of 1975.
        </p>
        <div class="icons">
            <a href="https://www.facebook.com/rgniyd/"
            target="_blank"><i class="fa fa-facebook"></i></a>
            <a href="https://twitter.com/rgniydt"
            target="_blank"><i class="fa fa-twitter"></i></a>
            <a href="https://instagram.com/students__of__rgniyd"
            target="_blank"><i class="fa fa-instagram"></i></a>
            <a href="https://www.linkedin.com/company/placement-cell-rgniyd/"
            target="_blank"><i class="fa fa-linkedin"></i></a>
```

```
        </div>
        <p>Made with<i class="fa fa-heart-o"></i>by Dhruti Ranjan</p>
    </section>

<!Javascript for Toggle Menu>

    <script>
        var navLinks = document.getElementById("navLinks");

        function showMenu()
        {
            navLinks.style.right = "0";
        }
        function hideMenu()
        {
            navLinks.style.right = "-200px";
        }
    </script>
    </body>
</html>
```

.5 contact.html Code

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>RGNIYD Clone</title>
    <link rel="stylesheet" href="style.css">
    <link rel="stylesheet" href=""><link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href=
    "https://fonts.gstatic.com" crossorigin>
    <link href="https://fonts.googleapis.com/css2?family=
    Poppins:wght@100;200;300;400;600;700&display=swap" rel="stylesheet">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/
    ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
```

```
</head>

<body>
    <section class="sub-header">
        <nav>
            <a href="index.html"><img src="Img/logo1.png"></a>
            <div class="nav-links" id="navLinks">
                <i class="fa fa-times" onclick="hideMenu()"></i>
                <ul>
                    <li><a href="index.html">HOME</a></li>
                    <li><a href="about.html">ABOUT</a></li>
                    <li><a href="course.html">COURSE</a></li>
                    <li><a href="blog.html">BLOG</a></li>
                    <li><a href="contact.html">CONTACT</a></li>
                </ul>
            </div>
            <i class="fa fa-bars" onclick="showMenu()"></i>
        </nav>
        <h1>Contact Us</h1>
    </section>

<!-------- contact us content ---------->

    <section class="location">
        <iframe src="https://www.google.com/maps/embed?pb=!1m18!1m12!1m3!
        1d3887.876011767785!2d79.956679314334253d12.979781218211755!2m3
        !1f0!2f0!3f0!3m2!1i1024!2i768!4f13.1!3m3!1m2!1s0x3a528d2b50bdd
        029%3A0x1be485e1fa9b290d!2sRGNIYD!5e0!3m2!1sen!2sin!4v16420
        86090282!5m2!1sen!2sin" width="600" height="450" style="border:0;"
        allowfullscreen="" loading="lazy"></iframe>
    </section>
    <section class="contact-us">
        <div class="row">
            <div class="contact-col">
                <div>
                    <i class="fa fa-home"></i>
                    <span>
                        <h5>Rajiv Gandhi National Institute of Youth Development</h5>
                        <p>Access through Secondary Gate, <br>Sri Ram Nagar, Sriperumbudur,
                        Tamil Nadu - 602105</p>
                    </span>
```

```
                </div>
                <div>
                    <i class="fa fa-phone"></i>
                    <span>
                        <h5>(091)044 - 27163127</h5>
                        <p>Monday to Friday, 10AM to 5PM</p>
                    </span>
                </div>
                <div>
                    <i class="fa fa-envelope-o"></i>
                    <span>
                        <h5>info@rgniyd.com</h5>
                        <p>Email Us Your Query</p>
                    </span>
                </div>
            </div>
            <div class="contact-col">
                <form action="">
                    <input type="text" placeholder="Enter your name :" required>
                    <input type="email" placeholder="Enter email address :" required>
                    <input type="text" placeholder="Enter your subject :" required>
                    <textarea rows="8" placeholder="Message :" required></textarea>
                    <button type="submit" class="hero-btn red-btn">Send Message</button>
                </form>
            </div>
        </div>
    </section>

<!-------- Footer ----------->

    <section class="footer">
        <h4>About Us</h4>
        <p>RGNIYD, Sriperumbudur, Tamil Nadu is an Institution of National
        Importance by the Act of Parliament No. 35/2012 under the Ministry of
        Youth Affairs & Sports, Government of India. <br>The RGNIYD was set up
        in 1993 under the Societies Registration Act, XXVII of 1975.
        </p>
        <div class="icons">
            <a href="https://www.facebook.com/rgniyd/"
            target="_blank"><i class="fa fa-facebook"></i></a>
            <a href="https://twitter.com/rgniydt"
```

```
            target="_blank"><i class="fa fa-twitter"></i></a>
            <a href="https://instagram.com/students__of__rgniyd"
            target="_blank"><i class="fa fa-instagram"></i></a>
            <a href="https://www.linkedin.com/company/placement-cell-rgniyd/"
            target="_blank"><i class="fa fa-linkedin"></i></a>
        </div>
        <p>Made with<i class="fa fa-heart-o"></i>by Dhruti Ranjan</p>
    </section>

<!Javascript for Toggle Menu>

    <script>
        var navLinks = document.getElementById("navLinks");

        function showMenu()
        {
            navLinks.style.right = "0";
        }
        function hideMenu()
        {
            navLinks.style.right = "-200px";
        }
    </script>
    </body>
</html>
```

.6 style.css Code

```
*
{
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    font-family: 'Poppins', sans-serif;
}
.header
{
    min-height: 100vh;
    width: 100%;
    background-image: linear-gradient(rgba(4,9,30,0.7),rgba(4,9,30,0.7)),
    url(https://www.rgniyd.gov.in/visit/wp-content/uploads/slider/
    cache/c2c9c6ecb417c289a88c1eb4066294fb/lib-YRC-scaled.jpg);
    background-position: center;
    background-size: cover;
    position: relative;
}
nav
{
    display: flex;
    padding: 2% 6%;
    justify-content: space-between;
    align-items: center;
}
nav img
{
    width: 60px;
}
.nav-links
{
    flex: 1;
    text-align: right;
}
.nav-links ul li
{
    list-style: none;
    display: inline-block;
    padding: 8px 12px;
```

```
    position: relative;
}
.nav-links ul li a
{
    color: #fff;
    text-decoration: none;
    font-size: 13px;
}
.nav-links ul li::after
{
    content: '';
    width: 0;
    height: 2px;
    background: #f44336;
    display: block;
    margin: auto;
    transition: 0.5s;
}
.nav-links ul li:hover::after
{
    width: 100%;
}
.text-box
{
    width: 90%;
    color: #fff;
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%,-50%);
    text-align: center;
}
.text-box h1
{
    font-size: 62px;
}
.text-box p
{
   margin: 10px 0 40px;
   font-size: 14px;
   color: #fff;
```

```
}
.hero-btn
{
    display: inline-block;
    text-decoration: none;
    color: #fff;
    border: 1px solid #fff;
    padding: 12px 34px;
    font-size: 13px;
    background: transparent;
    position: relative;
    cursor: pointer;
}
.hero-btn:hover
{
    border: 1px solid #f44336;
    background: #f44336;
    transition: 1s;
}

nav .fa
{
    display: none;
}

@media(max-width: 700px)
{
    .text-box h1
    {
        font-size: 20px;
    }
    .nav-links ul li
    {
        display: block;
    }
    .nav-links
    {
        position: fixed;
        background: #f44336;
        height: 100vh;
        width: 200px;
```

```
        top: 0;
        right: -200px;
        text-align: left;
        z-index: 2;
        transition: 1s;
    }
    nav .fa
    {
        display: block;
        color: #fff;
        margin: 10px;
        font-size: 22px;
        cursor: pointer;
    }
    .nav-links ul
    {
        padding: 30px;
    }
}

/*--------------- Course ------------------*/

.course
{
    width: 80%;
    margin: auto;
    text-align: center;
    padding-top: 100px;
}
h1
{
    font-size: 36px;
    font-weight: 600;
}
p
{
    color: #777;
    font-size: 14px;
    font-weight: 300;
    line-height: 22px;
    padding: 10px;
```

```
}
.row
{
    margin-top: 5%;
    display: flex;
    justify-content: space-between;
}
.course-col
{
    flex-basis: 31%;
    background: #fff3f3;
    border-radius: 10px;
    margin-bottom: 5%;
    padding: 20px 12px;
    box-sizing: border-box;
    transition: 0.5s;
}
h3
{
    text-align: center;
    font-weight: 600;
    margin: 10px 0;
}
.course-col:hover
{
    box-shadow: 0 0 20px 0px rgba(0,0,0,0.2);
}

@media(max-width: 700px)
{
    .row
    {
        flex-direction: column;
    }
}

/*------------- Call to action -----------------*/

.cta
{
    margin: 100px auto;
```

```
    width: 80%;
    background-image: linear-gradient(rgba(0,0,0,0.7),rgba(0,0,0,0.7)),
    url(https://static.businessworld.in/article/article_extra_large_image/
    1602580712_c3ecMk_Online_Education.jpg);
    background-position: center;
    background-size: cover;
    border-radius: 10px;
    text-align: center;
    padding: 100px 0;
}
.cta h1
{
    color: #fff;
    margin-bottom: 40px;
    padding: 0;
}

@media(max-width: 700px)
{
    .cta h1
    {
        font-size: 24px;
    }
}

/*--------------Footer-----------*/

.footer
{
    width: 100%;
    text-align: center;
    padding: 30px 0;
}
.footer h4
{
    margin-bottom: 25px;
    margin-top: 20px;
    font-weight: 600;
}
.icons .fa
{
```

```
    color: #f44336;
    margin: 0 13px;
    cursor: pointer;
    padding: 18px 0;
}
.fa-heart-o
{
    color: #f44336;
    margin: 0 13px;
}

/*-------- About Us page -------- */
.sub-header
{
  height: 50vh;
  width: 100%;
  background-image: linear-gradient(rgba(4,9,30,0.7),rgba(4,9,30,0.7)),
  url(https://media.tegna-media.com/assets/WKYC/images/95aaa24b-3b4b-49ac-9ed4-19c158049ac0/
  95aaa24b-3b4b-49ac-9ed4-19c158049ac0_1140x641.jpg);
  background-position: center;
  background-size: cover;
  text-align: center;
  color: #fff;
}
.sub-header h1
{
    margin-top: 100px;
}
.about-us
{
    width: 80%;
    margin: auto;
    padding-top: 75px;
    padding-bottom: 50px;
}
.row
{
    display: flex;
}
.about-col
{
```

```
    flex: 50%;
    padding: 30px 15px;
}
.about-col img
{
    width: 100%;
}
.about-col h1
{
    padding-top: 0;
}
.about-col p
{
    padding: 10px 0 25px;
    text-align: justify;
}
.red-btn
{
    border: 1px solid #f44336;
    background: transparent;
    color: #f44336;
}
.red-btn:hover
{
    color: #fff;
}

/*------ blog-content -------- */

.blog-content
{
    width: 80%;
    margin: auto;
    padding: 60px 0;
}
.blog-left
{
    flex-basis: 65%;
}
.blog-left img
{
```

```
    width: 100%;
}
.blog-left h2
{
    color: #222;
    font-weight: 600;
    margin: 30px 0;
}
.blog-left p
{
    color: #999;
    padding: 0;
    text-align: justify;
}
.blog-right
{
    flex-basis: 32%;
}
.blog-right h3
{
    background: #f44336;
    color: #fff;
    padding: 7px 0;
    font-size: 16px;
    margin-bottom: 20px;
}
.blog-right div
{
    display: flex;
    align-items: center;
    justify-content: space-between;
    color: #555;
    padding: 8px;
    box-sizing: border-box;
}
.comment-box
{
    border: 1px solid #ccc;
    margin: 50px 0;
    padding: 10px 20px;
}
```

```
.comment-box h3
{
    text-align: left;
}
.comment-form input, .comment-form textarea
{
    width: 100%;
    padding: 10px;
    margin: 15px 0;
    box-sizing: border-box;
    border: none;
    outline: none;
    background: #f0f0f0;
}
.comment-form button
{
    margin: 10px 0;
}

@media(max-width: 700px)
{
    .sub-header h1
    {
        font-size: 24px;
    }
}

/*--------- contact us ------*/
.location
{
    width: 80%;
    margin: auto;
    padding: 80px 0;
}
.location iframe
{
    width: 100%;
}
.contact-us
{
    width: 80%;
```

```
    margin: auto;
}
.contact-col
{
    flex-basis: 48%;
    margin-bottom: 30px;
}
.contact-col div
{
    display: flex;
    align-items: center;
    margin-bottom: 40px;
}
.contact-col div .fa
{
    font-size: 28px;
    color: #f44336;
    margin: 10px;
    margin-right: 30px;
}
.contact-col div p
{
    padding: 0;
}
.contact-col div h5
{
    font-size: 20px;
    margin-bottom: 5px;
    color: #555;
    font-weight: 400;
}
.contact-col input, .contact-col textarea
{
    width: 100%;
    padding: 15px;
    margin-bottom: 17px;
    outline: none;
    border: 1px solid #ccc;
    box-sizing: border-box;
}
```

.7 activity_main.xml Code

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <WebView
        android:id="@+id/webview"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</RelativeLayout>
```

.8 MainActivity.java Code

```
package com.rgniyd.app;

import androidx.appcompat.app.AppCompatActivity;
import android.graphics.Bitmap;
import android.net.Uri;
import android.os.Bundle;
import android.webkit.WebSettings;
import android.webkit.WebView;
import android.webkit.WebViewClient;
public class MainActivity extends AppCompatActivity {
    private WebView myWebView;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        myWebView = findViewById(R.id.webview);
        myWebView.setWebViewClient(new MyWebViewClient());
        WebSettings webSettings = myWebView.getSettings();
        webSettings.setJavaScriptEnabled(true);
        // Load the default URL in the WebView
```

```
        myWebView.loadUrl("https://rgniyd.netlify.app/");
    }
    private class MyWebViewClient extends WebViewClient {
        @Override
        public void onPageStarted(WebView view, String url, Bitmap favicon) {
            super.onPageStarted(view, url, favicon);
        }
        @Override
        public boolean shouldOverrideUrlLoading(WebView view, String url) {
            Uri uri = Uri.parse(url);
            String host = uri.getHost();
            String path = uri.getPath();

            if (host != null && host.equals("rgniyd.netlify.app")) {
                if (path != null && path.equals("/about.html")) {
                    // deep link for '/about.html'
                    view.loadUrl("https://rgniyd.netlify.app/about.html");
                    return true;
                } else if (path != null && path.equals("/course.html")) {
                    // deep link for '/course.html'
                    view.loadUrl("https://rgniyd.netlify.app/course.html");
                    return true;
                } else if (path != null && path.equals("/blog.html")) {
                    // deep link for '/blog.html'
                    view.loadUrl("https://rgniyd.netlify.app/blog.html");
                    return true;
                } else if (path != null && path.equals("/contact.html")) {
                    // deep link for '/contact.html'
                    view.loadUrl("https://rgniyd.netlify.app/contact.html");
                    return true;
                }
            }
            // For any other URLs, load them normally
            view.loadUrl(url);
            return true;
        }
    }
    @Override
    public void onBackPressed() {
        if (myWebView.canGoBack()) {
            myWebView.goBack();
```

```
        } else {
            super.onBackPressed();
        }
    }
}
```

.9 AndroidManifest.xml Code

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">
    <uses-permission android:name="android.permission.INTERNET" />
    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.RGNIYD"
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
            <intent-filter android:autoVerify="true">
                <action android:name="android.intent.action.VIEW" />
                <category android:name="android.intent.category.DEFAULT" />
                <category android:name="android.intent.category.BROWSABLE" />
                <data android:scheme="https" />
                <data android:host="rgniyd.netlify.app" />
                <data android:pathPrefix="/about.html" />
                <data android:pathPrefix="/course.html" />
                <data android:pathPrefix="/blog.html" />
```

```
                <data android:pathPrefix="/contact.html" />
            </intent-filter>
            <intent-filter android:autoVerify="true">
                <action android:name="android.intent.action.VIEW" />
                <category android:name="android.intent.category.DEFAULT" />
                <category android:name="android.intent.category.BROWSABLE" />
                <data android:scheme="http" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

References

[1] Yifei Ma et al. Aladdin: Automating release of android deep links to in-app content. In *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*, pages 139–140, Buenos Aires, 2017.

[2] Yifei Ma, Xiaoxing Liu, Rui Du, Zhiqiang Hu, Yun Liu, Min Yu, and Guangtai Huang. Droidlink: Automated generation of deep links for android apps. *arXiv preprint arXiv:1605.0692*, 2016.

[3] Yizheng Liang and Xiaodong Yan. Using deep learning to detect malicious urls. In *2019 IEEE International Conference on Energy Internet (ICEI)*, pages 487–492, 2019.

[4] Zhiqiang Yang, Ming Yang, Yanchao Zhang, Guofei Gu, Peng Ning, and Xiaofeng Steve Wang. Appintent: Analyzing sensitive data transmission in android for privacy leakage detection. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, New York, NY, USA, 2013. Association for Computing Machinery.

[5] Min Zhang and Heng Yin. Appsealer: Automatic generation of vulnerability-specific patches for preventing component hijacking attacks in android applications. In *Network and Distributed System Security Symposium*, 2014.

[6] Elaine Chin and David Wagner. Bifocals: Analyzing webview vulnerabilities in android applications. In *Proceedings of the 7th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, University of California, Berkeley, USA, 2014.

[7] Liu F., Wang C., Pico A., Yao D., and Wang G. Measuring the insecurity of mobile deep links of android. In *Proceedings of the USENIX Conference on Security Symposium.*, 2017.

[8] De-Jun Wu, Chen-Hung Mao, Te-En Wei, Hao-Ming Lee, and Kuo-Pei Wu. Droidmat: Android malware detection through manifest and api calls tracing. In *2012 Seventh Asia Joint Conference on Information Security*, pages 62–69, Tokyo, Japan, 2012.

[9] R. Dhaya and M. Poongodi. Detecting software vulnerabilities in android using static analysis. In *2014 IEEE International Conference on Advanced Communications, Control and Computing Technologies*, pages 915–918, Ramanathapuram, India, 2014.

[10] Samuel Feldman, David Stadther, and Bin Wang. Manilyzer: Automated android malware detection through manifest analysis. In *2014 IEEE 11th International Conference on Mobile Ad Hoc and Sensor Systems*, pages 767–772, Philadelphia, PA, USA, 2014.