**THREAT FEED AGGREGATOR APPLICATION (TFA)**

**ISAAC ISALWA**

**A THREAT FEEDS AGGREGATOR APPLICATION SUBMITTED TO THE SCHOOL OF BUSINESS AND TECHNOLOGY IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF A DEGREE IN BACHELOR OF SCIENCE IN INFROMATION TECHNOLOGY OF UMMA UNIVERSITY.**

**2024**

# **DECLARATION**

I, Isaac Isalwa, declare that the content of this research project titled "Threat Feed Aggregator (TFA)” is my original work and has been prepared based on my research, analysis, and understanding of the subject matter. Any references or sources used in this proposal have been appropriately cited and credited.

I acknowledge that this proposal is submitted as part of academic evaluation, and is intended solely for the intended audience's review and consideration. I take full responsibility for the accuracy, completeness, and integrity of the information presented herein.

**Student’s Signature**

Signature………………… Date…………………

**Isaac Isalwa**

**Registration No: BSIT/2021/47638**

This system project has been submitted for examination with my approval as the University Supervisor.

Signature………………… Date…………………

**UMMA University Kenya**

# 

# **DEDICATION**

This Threat Feed Aggregator project is dedicated with heartfelt gratitude to my family, whose unwavering support has been my anchor; to my mentor and supervisor, whose guidance has been instrumental in shaping this work; and to the entire ICT department, whose collaborative spirit and commitment to have been a constant source of inspiration. Your collective support, wisdom, and encouragement have been the driving force behind this endeavor, and I am deeply grateful for the role each of you has played in bringing this project to fruition.

# 

# **ACKNOWLEDGMENT**

I would like to express my heartfelt gratitude to Madam Sekento for her invaluable guidance and mentorship throughout the development of this proposal. Her expertise and support have been pivotal in shaping the clarity and depth of the project. Additionally, I extend my thanks to my fellow IT friends from the Information Communication Technology (ICT) department for their collaborative spirit and contributions, which have enriched the proposal with diverse perspectives and insights.

# 

# ABSTRACT

The Threat Feed Aggregator (TFA) project aims to develop an integrated mobile application for aggregating, filtering, and prioritizing threat intelligence data. This document outlines the systematic approach to design, develop, and implement the TFA Android app using Kotlin. The project encompasses key features such as user authentication (login/signup), RSS feed data acquisition, local data storage, threat information display, and a notification system for new threats.

By leveraging modern Android development practices and the Model-View-ViewModel (MVVM) architecture, the TFA project strives to deliver a robust and user-friendly mobile solution. The application focuses on efficient data processing, secure local storage using Room database with encryption, and a reactive user interface for enhanced user experience.

The development process includes setting up the Android development environment, implementing core modules for data acquisition and processing, creating an intuitive user interface, and integrating a notification system for timely threat alerts. Security considerations are prioritized throughout the development, ensuring safe handling of sensitive threat data.

Through comprehensive testing, performance optimization, and a well-planned maintenance strategy, the TFA project aims to provide a reliable and efficient tool for security personnel to enhance their threat detection and response capabilities on the go. The mobile-first approach allows for real-time access to critical threat intelligence, enabling timely decision-making and improved security posture.

Table of Contents

[DECLARATION 2](#_Toc173249936)

[DEDICATION 3](#_Toc173249937)

[ACKNOWLEDGMENT 4](#_Toc173249938)

[ABSTRACT 5](#_Toc173249939)

[LIST OF ACRONYMS 13](#_Toc173249940)

[CHAPTER ONE: INTRODUCTION 14](#_Toc173249941)

[1.1 Background of the study 14](#_Toc173249942)

[1.2 Problem Statement 14](#_Toc173249943)

[1.3 General Objective 15](#_Toc173249944)

[1.4 Primary objectives 15](#_Toc173249945)

[1.5 Research Questions 15](#_Toc173249946)

[1.6 Scope of the Study 15](#_Toc173249947)

[1.6.1 Data Sources 15](#_Toc173249948)

[1.6.2 Functionality 16](#_Toc173249949)

[1.6.3 Evaluation 16](#_Toc173249950)

[1.6.4 Expected Outcomes 16](#_Toc173249951)

[1.7 Justification of the Study 16](#_Toc173249952)

[1.8 Limitations of the Study 17](#_Toc173249953)

[1.9 Conclusion 17](#_Toc173249954)

[CHAPTER 2: LITERATURE REVIEW 18](#_Toc173249955)

[2.1. Introduction 18](#_Toc173249956)

[2.2 Review of Existing Literature on TFAs 18](#_Toc173249957)

[2.3 Existing Landscape of TFA 19](#_Toc173249958)

[2.3.1 Feedly 19](#_Toc173249959)

[2.3.2 Cyware 20](#_Toc173249960)

[2.4 Gaps in existing threat intelligence approaches 21](#_Toc173249961)

[2.5 How TFA bridges the gap 21](#_Toc173249962)

[2.6 Functionalities and benefits of TFA 21](#_Toc173249963)

[2.7 Future directions of TFA 22](#_Toc173249964)

[2.8 Conclusion 22](#_Toc173249965)

[CHAPTER 3: SYSTEM DESIGN METHODOLOGY 23](#_Toc173249966)

[3.1. Introduction 23](#_Toc173249967)

[3.2 Software Development Methodology 23](#_Toc173249968)

[3.3 System Analysis and Requirements Gathering 24](#_Toc173249969)

[3.4 Research Design 25](#_Toc173249970)

[3.4.1 Qualitative Methods 25](#_Toc173249971)

[3.4.2 Quantitative Methods 25](#_Toc173249972)

[3.4.3 Research Location 25](#_Toc173249973)

[3.4.4 Target Population 25](#_Toc173249974)

[3.4.5 Sample Size Selection 26](#_Toc173249975)

[3.5.3 Operational Feasibility 27](#_Toc173249976)

[3.5.4 Legal and Regulatory Feasibility 27](#_Toc173249977)

[3.5.5 Scheduling Feasibility 28](#_Toc173249978)

[3.6 System Design 28](#_Toc173249979)

[3.7 System Development 29](#_Toc173249980)

[3.8 Testing and Deployment 29](#_Toc173249981)

[3.8 Conclusion 30](#_Toc173249982)

[CHAPTER 4: SYSTEM ANALYSIS AND REQUIREMENTS GATHERING 31](#_Toc173249983)

[4.1 Introduction 31](#_Toc173249984)

[4.2 Information Gathering Research Questions 31](#_Toc173249985)

[4.3 Information Gathering Methods 31](#_Toc173249986)

[4.3.1 Online Questionnaires 31](#_Toc173249987)

[4.3.2 Individual Interviews 32](#_Toc173249988)

[4.4 Findings 32](#_Toc173249989)

[4.4.1 Aggregation of RSS Feeds 32](#_Toc173249990)

[4.4.2 Feed Display and Search Functionality 32](#_Toc173249991)

[4.4.3 Notification System 32](#_Toc173249992)

[4.4.4 Indicators of Compromise (IOCs) and Exploit Information 32](#_Toc173249993)

[4.4.5 Integration with Vulnerability Scanners 33](#_Toc173249994)

[4.4.6 Automated Threat Feed Integration 33](#_Toc173249995)

[4.4.7 User Interface Considerations 33](#_Toc173249996)

[4.4.8 Infrastructure and Deployment Needs 33](#_Toc173249997)

[4.5 Stakeholders and Information Needs Identified 34](#_Toc173249998)

[4.5.1 Security Analysts 34](#_Toc173249999)

[4.5.2 Desired Functionalities in a TFA 35](#_Toc173250000)

[4.5.3 IT Operations Staff 35](#_Toc173250001)

[4.5.4 Security Operations Center (SOC) Personnel 35](#_Toc173250002)

[4.6 Product Backlog Creation 36](#_Toc173250003)

[4.7 Conclusion 36](#_Toc173250004)

[CHAPTER 5: SYSTEM ARCHITECTURE AND DESIGN 37](#_Toc173250005)

[5.1 Introduction 37](#_Toc173250006)

[5.2 High-Level System Architecture 37](#_Toc173250007)

[5.2.1 Data Acquisition Module 37](#_Toc173250008)

[5.2.3 Data Processing and Enrichment Module 37](#_Toc173250009)

[5.2.3 Local Threat Intelligence Storage 37](#_Toc173250010)

[5.2.4 User Interface (UI) Module 37](#_Toc173250011)

[5.2.5 Alerting and Notification Module 38](#_Toc173250012)

[5.2.6 Authentication and Authorization 38](#_Toc173250013)

[5.3 Technology stack selection 38](#_Toc173250014)

[5.4 User Interface (UI) Design Principles 38](#_Toc173250015)

[5.4.1 Use Case Diagram 39](#_Toc173250016)

[5.4.2 Flowchart 40](#_Toc173250017)

[5.5Application Flow 41](#_Toc173250018)

[5.5.1 Data acquisition flow 41](#_Toc173250019)

[5.5.2 Data processing and storage flow 42](#_Toc173250020)

[5.5.3 User interaction flow 43](#_Toc173250021)

[5.5.4 Alerting and notification flow 43](#_Toc173250022)

[5.6 Architectural Pattern 44](#_Toc173250023)

[5.8 Security Considerations 45](#_Toc173250024)

[5.8.1 Data encryption 45](#_Toc173250025)

[5.8.2 Authentication and authorization 45](#_Toc173250026)

[5.8.3 Secure Coding Practices 45](#_Toc173250027)

[5.8.4 Regular Security Audits 45](#_Toc173250028)

[5.8.5 Device security 45](#_Toc173250029)

[5.9 Development approach using scrum 45](#_Toc173250030)

[5.9.1 Product backlog 46](#_Toc173250031)

[5.9.2 Sprint planning 46](#_Toc173250032)

[5.9.3 Daily scrum meetings 46](#_Toc173250033)

[5.9.4 Sprint review and retrospective 46](#_Toc173250034)

[5.10 Conclusion 46](#_Toc173250035)

[CHAPTER 6: APPLICATION DEVELOPMENT 47](#_Toc173250036)

[6.1 Introduction 47](#_Toc173250037)

[6.2 Development Environment Setup 48](#_Toc173250038)

[6.3 Implementation of Key Components 48](#_Toc173250039)

[6.3.1 Data Acquisition Module 48](#_Toc173250040)

[6.3.2 Data Processing and Enrichment Module 48](#_Toc173250041)

[6.3.3 Local Threat Intelligence Storage 48](#_Toc173250042)

[6.3.4 User Interface (UI) Module 48](#_Toc173250043)

[6.3.5 Alerting and Notification Module 49](#_Toc173250044)

[6.3.6 Authentication and Authorization 49](#_Toc173250045)

[6.4 MVVM Architecture Implementation 49](#_Toc173250046)

[6.5 Security Implementations 49](#_Toc173250047)

[6.6 Testing 50](#_Toc173250048)

[6.7 Performance Optimization 50](#_Toc173250049)

[6.8 Maintenance 50](#_Toc173250050)

[6.9 Challenges Faced and Solutions 51](#_Toc173250051)

[6.10 Future Enhancements 51](#_Toc173250052)

[6.11 Conclusion 51](#_Toc173250053)

[CHAPTER 7: TESTING AND DEPLOYMENT 52](#_Toc173250054)

[7.1 Introduction 52](#_Toc173250055)

[7.2 Testing strategies 52](#_Toc173250056)

[7.2.1 Unit testing 52](#_Toc173250057)

[7.2.2 Integration testing 53](#_Toc173250058)

[7.2.3 End-to-End testing 53](#_Toc173250059)

[7.3 Deployment process 53](#_Toc173250060)

[7.3.1 Preparing for deployment 53](#_Toc173250061)

[7.3.2Post-Deployment Monitoring 54](#_Toc173250062)

[7.4 Conclusion 54](#_Toc173250063)

[CHAPTER 8: CONCLUSION AND RECOMMENDATIONS 55](#_Toc173250064)

[8.1 Project summary 55](#_Toc173250065)

[8.2 Key findings 55](#_Toc173250066)

[8.3 Lessons learned 55](#_Toc173250067)

[8.4 Future Enhancements 56](#_Toc173250068)

[8.4.1 Short-Term recommendations 56](#_Toc173250069)

[8.4.2 Long-term vision 56](#_Toc173250070)

[8.5 Sustainability plan 57](#_Toc173250071)

[8.6 Final thoughts 57](#_Toc173250072)

[APENDICES 58](#_Toc173250073)

[Appendix 1: Information Gathering Research Questions 58](#_Toc173250074)

[Appendix 2: Code Snippets for layout development 58](#_Toc173250075)

[REFERENCES 63](#_Toc173250076)

**List of Figures**

[Figure 2 1Feedly App screenshot 19](#_Toc173155788)

[Figure 2 2 Cyware mobile application 20](#_Toc173155789)

[Figure 3 1 Agile Model 23](#_Toc173155794)

[Figure 4 1 Responses to survey questions 34](#_Toc173155962)

[Figure 4 2 Response Graph of a Survey question 35](#_Toc173155963)

[Figure 5 1 Use Case Diagram 39](#_Toc173155766)

[Figure 5 2 Flowchart 40](#_Toc173155767)

[Figure 5 3 Data acquisition 42](#_Toc173155768)

[Figure 5 4 User Interaction Diagram 43](#_Toc173155769)

[Figure 5 5 MVVM Architecture source techtarget.com 44](#_Toc173155770)

[Figure 6 1 Technological stack 47](#_Toc173155798)

[Figure 6 2 MVVM Application in TFA 49](#_Toc173155799)

[Figure 7 1 Testing pyramid 52](#_Toc173155801)

[Figure 10 1Threat Interface Design in Android Studio 58](#_Toc173250859)

[Figure 10 2AppDatabase Code in android Studio 59](#_Toc173250860)

[Figure 10 3 Splashscreen and signup 60](#_Toc173250861)

[Figure 10 4 Kaspersky and Ibmexchange 61](#_Toc173250862)

[Figure 10 5 Settings and rating 62](#_Toc173250863)

# LIST OF ACRONYMS

**AI** - Artificial Intelligence

**CISA** - Cybersecurity and Infrastructure Security Agency

**HTTPS** -Hypertext Transfer Protocol Secure

**IBM** - International Business Machines Corporation

**IOC** - Indicators of Compromise

**ML** - Machine Learning

**MVVM** - Model-View-View Model

**OSINT** - Open-Source Intelligence

**RSS** - Really Simple Syndication

**SOCS** - Security Operation Officer

**STIX** - Structured Threat Information Expression

**TFA** - Threat Feed Aggregator

**THN** - The Hacker News

**UAT** - User Acceptance Testing

# CHAPTER ONE: INTRODUCTION

## Background of the study

Background of the study refers to the context, circumstances, and [history](https://researchmethod.net/what-is-history/) that led to the research problem or topic being studied. It provides the reader with a comprehensive understanding of the subject matter and the significance of the study.([Research](https://researchmethod.net/author/hassan/) Guide 2024)

The cybersecurity landscape is constantly evolving, with new threats and vulnerabilities emerging daily. Security professionals and enthusiasts need to stay informed, but the sheer volume of information can be overwhelming. RSS feeds from reputable sources like IBM X-Force Exchange, The Hacker News, and CISA (Cybersecurity and Infrastructure Security Agency) offer valuable insights, but manually checking multiple sources is time-consuming and inefficient.

While there are various RSS reader apps available, few cater specifically to cybersecurity professionals' needs. This project proposes the development of an Android app that aggregates and presents cybersecurity-focused RSS feeds in a user-friendly and efficient manner.

## Problem Statement

A problem statement is a concise and concrete summary of the [research problem](https://www.scribbr.com/research-process/research-problem/) you seek to address. It should.(Sribbr 2024)

Cybersecurity professionals and enthusiasts face challenges in staying up-to-date with the latest threat intelligence and security news:

Information overload: Multiple sources publish security updates frequently, making it difficult to keep track of all relevant information.

Time constraints: Checking multiple websites or RSS feeds individually is time-consuming and inefficient.

Lack of centralization: There's a need for a single, mobile-friendly platform that aggregates cybersecurity-specific feeds.

Difficulty in prioritizing: Without proper organization, it's challenging to identify the most critical or relevant information quickly.

These issues can lead to missed critical updates, delayed response to emerging threats, and an overall reduction in situational awareness.

## General Objective

General objectives are the main goals of the study and are usually fewer in number while specific objectives are more in number because they address several aspects of the research problem.

To develop an Android app that efficiently aggregates and presents RSS feeds from IBM X-Force Exchange, The Hacker News, and CISA, providing users with a centralized platform for accessing cybersecurity news and threat intelligence.

## 1.4 Primary objectives

1. To create a user-friendly interface for browsing aggregated cybersecurity RSS feeds on Android devices.
2. To implement efficient RSS parsing and storage mechanisms to ensure quick access to the latest information.
3. To provide customization options, allowing users to prioritize feeds and filter content based on their specific interests or needs.
4. To enhance the overall user experience through features such as offline reading, push notifications for critical updates, and easy sharing capabilities.

## 1.5 Research Questions

1. How can an Android app effectively aggregate and present RSS feeds from IBM X-Force Exchange, The Hacker News, and CISA to improve access to cybersecurity information?
2. What features and design elements are most beneficial for cybersecurity professionals and enthusiasts when consuming RSS feed content on mobile devices?
3. How can the app enhance user engagement and information retention compared to manually checking individual cybersecurity news sources?

## 1.6 Scope of the Study

This project focuses on the development and evaluation of an Android app for aggregating cybersecurity RSS feeds. Its scope encompasses:

### 1.6.1 Data Sources

Integration with RSS feeds from IBM X-Force Exchange, The Hacker News, and CISA.

### 1.6.2 Functionality

1. RSS feed aggregation: Connects to multiple cybersecurity-focused RSS feeds, eliminating the need for users to check individual sources.
2. User-friendly interface: Presents aggregated content in an easily navigable and readable format optimized for Android devices.
3. Customization options: Allows users to prioritize feeds, and filter content based on specific topics or threat categories.
4. Offline reading: Enables users to access previously loaded content without an internet connection.

### 1.6.3 Evaluation

1. Performance testing: Measures the app's efficiency in fetching, parsing, and displaying RSS feed content.
2. User experience evaluation: Assesses the usability and effectiveness of the app's interface and features.
3. Impact assessment: Evaluates how the app improves users' ability to stay informed about cybersecurity threats and news compared to traditional methods.

### 1.6.4 Expected Outcomes

1. Improved efficiency: Users can access multiple cybersecurity news sources quickly and easily.
2. Enhanced awareness: Centralized access to curated cybersecurity feeds leads to better situational awareness.
3. Time savings: Reduced effort required to stay informed about the latest cybersecurity developments.

## 1.7 Justification of the Study

The justification of a research is also known as the rationale. Writing the justification or rationale comes from an in-depth search and analysis of the existing literature around the topic. A comprehensive literature search typically reveals gaps in previous studies that you may then wish to explore through your research. (Insights 2024)

The increasing importance of cybersecurity awareness, coupled with the rapid pace of threat evolution, necessitates efficient access to reliable information sources. This Android app addresses this need by:

1. Providing a unified platform for accessing cybersecurity RSS feeds from reputable sources.
2. Offering a mobile-friendly solution that allows users to stay informed on-the-go.
3. Implementing customization features to help users focus on the most relevant information.
4. Enhancing the overall user experience of consuming cybersecurity news and threat intelligence.

By focusing on ethical and publicly available RSS feeds, this project aims to contribute to improved cybersecurity awareness while respecting intellectual property rights and privacy concerns.

## 1.8 Limitations of the Study

The app's effectiveness is limited to the quality and frequency of updates from the chosen RSS feed sources.

The project focuses solely on Android devices, excluding users of other mobile platforms.

The app relies on active internet connections for real-time updates, with limited functionality in offline modes.

Despite these limitations, this project offers a significant contribution by addressing the need for efficient, mobile-friendly access to cybersecurity news and threat intelligence.

## 1.9 Conclusion

This Android app for aggregating cybersecurity RSS feeds from IBM X-Force Exchange, The Hacker News, and CISA aims to empower security professionals and enthusiasts with readily accessible, up-to-date information. By streamlining the process of staying informed about the latest threats and vulnerabilities, this project contributes to enhanced cybersecurity awareness and preparedness in an increasingly complex digital landscape.

# 

# CHAPTER 2: LITERATURE REVIEW

## 2.1. Introduction

This literature review chapter delved into the critical role of threat intelligence (TI) in safeguarding organizations within the dynamic cybersecurity landscape. The ever-evolving nature of cyber threats necessitated proactive measures, and TI served as a cornerstone for effective anticipation, detection, and response strategies. However, traditional manual approaches to gathering and analyzing TI data were hampered by the sheer volume of information. This review addressed this challenge by exploring how automated solutions, specifically Threat Feed Aggregators (TFAs), were revolutionizing threat intelligence.

## 2.2 Review of Existing Literature on TFAs

Tounsi & Rais (2018) provide a comprehensive overview of technical threat intelligence in the context of sophisticated cyber attacks. They discuss various types of threat feeds, including IOCs tactics, techniques, and procedures (TTPs), and security alerts. The authors highlight the challenges of aggregating these diverse feeds, such as data standardization, quality assessment, and real-time processing. This study is particularly relevant to your project as it outlines the complexities involved in creating an effective threat feed aggregator.

Samtani et al. (2017) focus on proactive cyber threat intelligence by exploring emerging hacker assets. They discuss the importance of aggregating data from various sources, including dark web forums and marketplaces. The study presents methods for identifying key threat actors and emerging trends, which could be valuable for prioritizing and contextualizing the feeds in your aggregator system.

Mavroeidis & Bromander (2017) evaluate various taxonomies, sharing standards, and ontologies within cyber threat intelligence. This paper is crucial for your project as it discusses different models for structuring and sharing threat data. Understanding these standards is essential for designing an interoperable and efficient threat feed aggregator.

Sauerwein et al. (2017) provide an exploratory study of threat intelligence sharing platforms. They analyze various software vendors and research perspectives, offering insights into the current state of the art in threat intelligence aggregation and sharing. This study can help you identify best practices and potential areas for innovation in your own threat feed aggregator.

Husák et al. (2019) survey attack projection, prediction, and forecasting in cybersecurity. While the main focus is on predictive capabilities, the paper discusses various threat feeds and the challenges of aggregating and analyzing them. This study is relevant to your project as it highlights the importance of not just aggregating threat feeds, but also using them for proactive cybersecurity measures.

## 2.3 Existing Landscape of TFA

The landscape of mobile threat intelligence aggregation is evolving, with several platforms offering varying functionalities:

### 2.3.1 Feedly

Feedly is a popular RSS reader that can be customized for cybersecurity feeds. While not specifically designed for threat intelligence, it allows users to aggregate content from various sources, including cybersecurity blogs and news sites. Feedly offers a mobile app, making it accessible for on-the-go professionals. However, it lacks specialized features for threat analysis and prioritization (Feedly, 2024).



Figure 2 1Feedly App screenshot

### 2.3.2 Cyware

Cyware offers a more specialized approach to threat intelligence with its mobile app. It provides features such as real-time threat alerts, incident response workflows, and threat intelligence sharing capabilities. Cyware aims to deliver actionable intelligence to security teams, but its broad focus on overall threat intelligence management may be overwhelming for users primarily interested in RSS feed aggregation (Cyware, 2024).

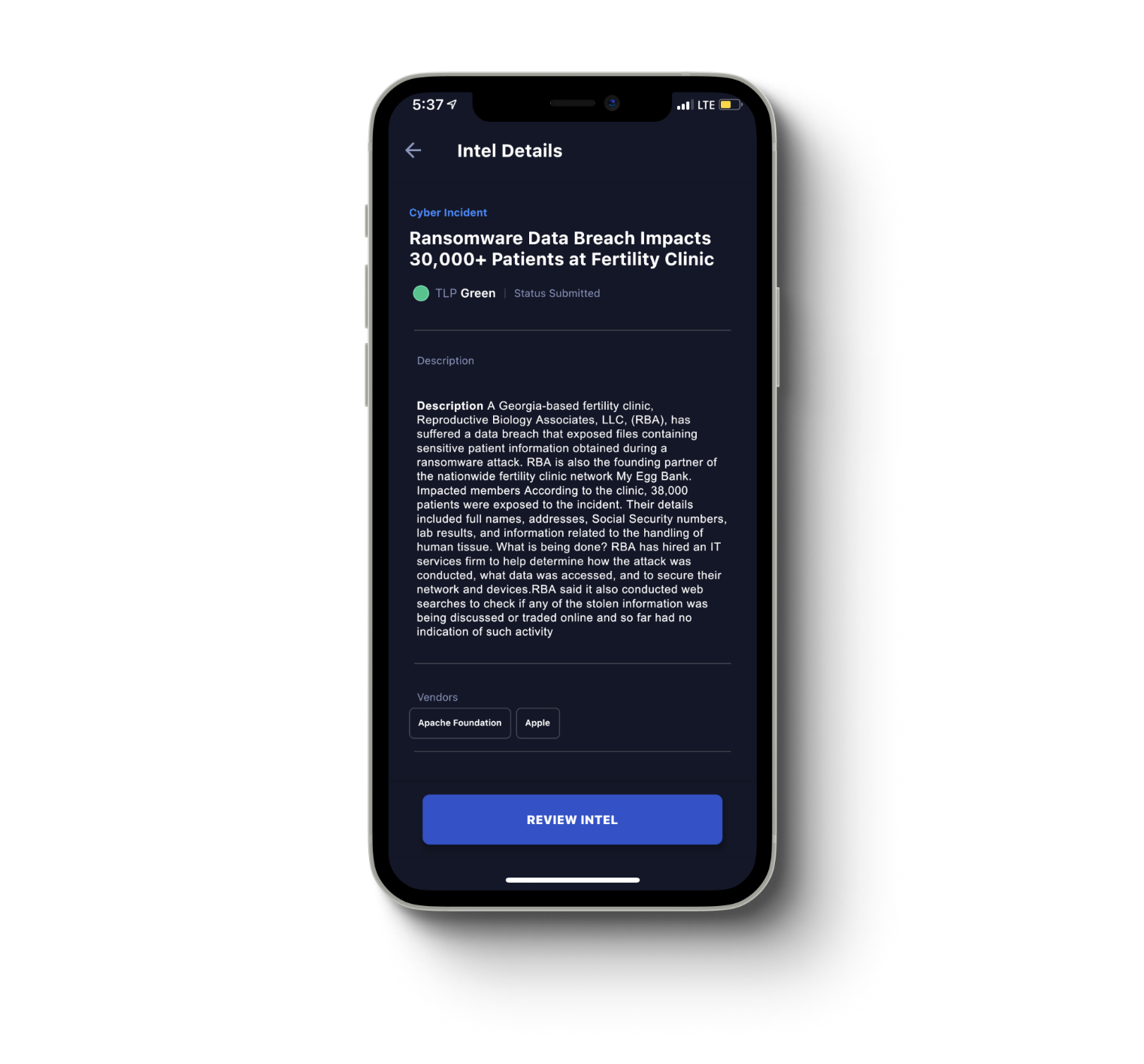


Figure 2 2 Cyware mobile application

## 2.4 Gaps in existing threat intelligence approaches

Despite the availability of these platforms, several gaps remain in the mobile consumption of threat intelligence:

Lack of focus: General RSS readers like Feedly lack cybersecurity-specific features, while comprehensive platforms like Cyware may offer more complexity than needed for simple feed aggregation (Tounsi & Rais, 2018).

Limited source integration: Many existing solutions don't offer seamless integration with authoritative sources like IBM X-Force Exchange, CISA, and The Hacker News in a single, user-friendly mobile interface.

Customization constraints: Current platforms often provide limited options for cybersecurity professionals to customize their feed based on specific threats or categories relevant to their organization (Mavroeidis & Bromander, 2017).

## 2.5 How TFA bridges the gap

The proposed Threat Feed Aggregator (TFA) Android app addresses these limitations by:

Providing a mobile-optimized interface specifically designed for cybersecurity professionals.

Focusing on high-quality, authoritative sources (IBM X-Force Exchange, The Hacker News, CISA) to ensure reliability of information.

Offering customization options tailored to cybersecurity needs, such as filtering by threat type or severity.

Implementing features like offline reading and push notifications for critical updates, enhancing the mobile user experience (Sauerwein et al., 2017).

## 2.6 Functionalities and benefits of TFA

TFAs offered several key functionalities:

1. Centralized Collection: TFAs aggregated data from various sources, including commercial feeds, OSINT, threat research reports, and government advisories. This eliminated the need to manually access and analyze data from multiple sources, saving time and effort.
2. Data Filtering and Enrichment: TFAs filtered raw data, removing irrelevant information and enriching it with additional context, such as threat actor attribution, associated vulnerabilities, and mitigation strategies.
3. Normalization and Standardization: TFAs often normalized data into standardized formats, such as STIX or TAXII, enabling seamless integration with security information and event management (SIEM) systems and other security tools.
4. Alert Generation and Prioritization: TFAs could generate alerts based on user-defined criteria, allowing security teams to prioritize and focus on the most relevant threats.

## 2.7 Future directions of TFA

TFAs were expected to evolve alongside the cyber threat landscape. Key future directions included:

1. Integration with AI and ML: AI/ML could enhance data analysis, automate threat detection, and personalize threat intelligence based on an organization's specific needs.
2. By using machine learning algorithms to analyze data, organizations could detect potential threats before they occurred and stay ahead of the game. SIEM solutions that utilized predictive analytics offered several benefits over traditional SIEM, including early detection of threats, better accuracy, increased efficiency and scalability.( Joy Wang 2023)
3. Enhanced Data Sharing and Collaboration: Greater collaboration between organizations and threat intelligence communities could enrich TFA data and improve overall threat visibility.
4. Standardization and Interoperability: Standardization across TFA platforms would facilitate seamless information exchange and improve the overall effectiveness of the technology.

## 2.8 Conclusion

TFAs played a crucial role in enhancing cyber threat detection and mitigation by streamlining the collection, analysis, and utilization of TI. As the cyber threat landscape continued to evolve, TFAs were expected to become increasingly sophisticated, incorporating AI/ML and fostering collaboration to empower security teams in the fight against cyberattacks.

# **CHAPTER 3: SYSTEM DESIGN METHODOLOGY**

## 3.1. Introduction

This document proposed the adoption of an Agile development methodology for the Threat Feed Aggregator (TFA) project. This approach prioritized flexibility, collaboration, and iterative delivery, ensuring a responsive and user-centric solution that effectively addressed cybersecurity threats. The decision to implement Agile methodologies was driven by the rapidly evolving nature of cyber threats and the need for a highly adaptable development process.

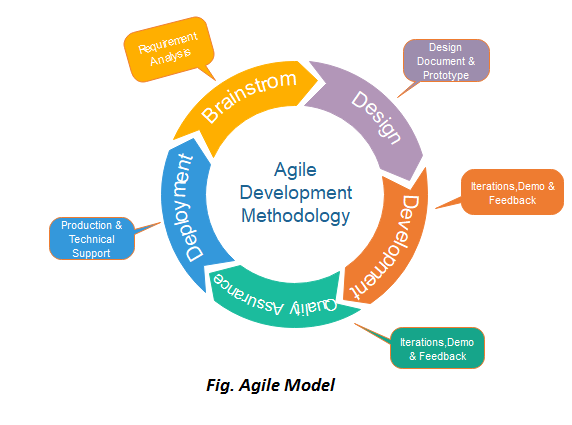
****

Figure 3 1 Agile Model

## 3.2 Software Development Methodology

Software Development Methodology refers to a framework that is used to structure, plan, and control the process of developing an information system. It includes the pre-definition of specific deliverables and artifacts that are created and completed by a project team to develop or maintain an application [1].

The chosen methodology for this project was Agile, specifically using the Scrum framework.

Reasons for choosing Agile Scrum:

1. Flexibility: Agile allowed for adaptive planning and evolutionary development, which wass crucial in the rapidly evolving field of cybersecurity.
2. Iterative approach: Scrum's sprint-based development enabled regular feedback and continuous improvement of the Threat Feed Aggregator.
3. Stakeholder involvement: Frequent interactions with end-users and stakeholders ensured that the system was to meets actual cybersecurity needs.
4. Risk mitigation: Early and continuous delivery of working software helps in identifying and addressing potential issues early in the development process.
5. Adaptability to changing requirements: In the dynamic world of cyber threats, the ability to respond quickly to new types of feeds or aggregation requirements is essential.

## 3.3 System Analysis and Requirements Gathering

A comprehensive understanding of the project scope was established through close collaboration with stakeholders to identify their needs. This phase was critical in ensuring the TFA system would meet the real-world demands of cybersecurity professionals.

Here is a breakdown of the activities:

* Stakeholder Identification: All stakeholders involved in the project were identified, including developers, security analysts, end-users, project managers, and cybersecurity experts from various industries.
* Data gathering: Conducting interviews, and online surveys to gather stakeholders' needs, expectations, and pain points. This process included in-depth discussions about current threat intelligence practices and limitations.
* User story mapping: Workshops were conducted to visualize user workflows and define clear, measurable acceptance criteria for each feature. This process helped prioritize features based on their impact on threat detection and response times.
* Product Backlog Creation: The gathered requirements were prioritized and documented in a product backlog to guide development efforts. The backlog was regularly reviewed and updated to reflect emerging threats and technologies.

## 3.4 Research Design

To ensure that the TFA effectively met user needs, a mixed research design combining qualitative and quantitative methods was employed:

### 3.4.1 Qualitative Methods

Conducted in-depth interviews with security professionals to gain insights into their current challenges and desired functionalities for threat intelligence management. These interviews included scenarios and real-world examples to better understand user needs.

Analyzed existing user documentation and support forums to identify common pain points and user needs. This analysis extended to competitor products to identify areas for improvement.

### 3.4.2 Quantitative Methods

Developed and distributed online surveys to a targeted sample population of security professionals. The surveys included questions about threat intelligence sources, integration preferences, and desired automation capabilities.

Utilized survey data to gain a broader understanding of user needs and preferences regarding threat intelligence integration and prioritization. Statistical analysis was performed to identify trends and correlations.

### 3.4.3 Research Location

The research location depended on the chosen research methods:

* Interviews: These were conducted in-person at participants' workplaces or virtually through video conferencing platforms. On-site interviews provided valuable context about the working environment.
* Online Surveys: These were distributed electronically using survey tools and shared through email lists, industry forums, or social media groups targeting security professionals. Specialized cybersecurity forums were leveraged to reach a diverse pool of respondents.

### 3.4.4 Target Population

The target population for the research was security professionals working in IT departments, Security Operations Centers (SOCs), or threat intelligence teams. This focused population ensured that the gathered information directly reflected the needs of those who would utilize the TFA. The study also included professionals from various industry sectors to capture diverse perspectives.

### 3.4.5 Sample Size Selection

A statistically significant sample size was determined based on the chosen research method:

* Interviews: While in-depth interviews provided rich data, a smaller sample size sufficed due to the qualitative nature of the information gathered. The interview pool was carefully selected to represent different roles and experience levels.
* Surveys: For statistically significant results, a larger sample size was targeted. Online survey tools provided sample size calculators to guide the selection process based on desired confidence level and population size estimations. The survey was distributed over a period of several weeks to ensure adequate response rates.

**3.5 Feasibility Study**

The feasibility study was an essential step in the development of the TFA application, aimed at assessing various aspects to ensure the project's viability before moving forward. This following outline the key components evaluated during the feasibility study.

**3.5.1 Technical Feasibility**

The technical feasibility analysis focused on determining whether the existing technology stack could support the development and deployment of the TFA application. The key activities included:

* Technology Stack Evaluation: Assessed the suitability of Kotlin, Android Studio, Firebase, Retrofit, JSON, Glide, Jsoup, Room, and Material libraries for the TFA's requirements. Evaluated the capabilities of these technologies to handle the data processing, real-time updates, and integration with external threat intelligence sources.
* System Integration: Analyzed the ease of integrating various APIs and data sources, such as IBM X-Force Exchange and other open-sourced data from Hacker News. Ensured that these integrations would provide reliable and up-to-date threat intelligence feeds.
* Scalability and Performance: Examined the potential for scaling the application to handle an increasing number of users and data volume. Considered performance optimization techniques to ensure smooth operation under load.

**3.5.2 Economic Feasibility**

The economic feasibility analysis aimed to determine the cost-effectiveness of the TFA project. The main activities included:

* Cost-Benefit Analysis: Compared the estimated costs of development, deployment, and maintenance against the expected benefits, such as improved threat intelligence management and enhanced security for users.
* Budget Planning: Developed a detailed budget covering all project phases, including design, development, testing, and marketing. Identified potential funding sources and revenue streams to support the project financially.

### 3.5.3 Operational Feasibility

The operational feasibility analysis focused on evaluating whether the TFA application could be effectively operated and maintained in the long term. Key considerations included:

* User Training and Support: Planned for comprehensive user training programs and support resources to ensure smooth adoption of the TFA application. Developed user manuals, FAQs, and customer support channels.
* Maintenance and Updates: Established a maintenance plan to address software updates, bug fixes, and feature enhancements. Ensured that the development team could provide ongoing support and improvements based on user feedback.
* Resource Allocation: Identified the human and technical resources required for successful operation, including developers, support staff, and infrastructure. Ensured that resource allocation aligned with project needs and budget constraints.

### 3.5.4 Legal and Regulatory Feasibility

The legal and regulatory feasibility analysis ensured that the TFA application complied with relevant laws and regulations. The main activities included:

* Data Privacy and Security: Evaluated compliance with data protection regulations, such as GDPR and CCPA, to ensure user data privacy and security. Implemented necessary measures to protect sensitive information.
* Intellectual Property: Assessed the potential for intellectual property issues related to the use of third-party APIs, data sources, and proprietary technologies. Ensured that all licenses and permissions were obtained.
* Regulatory Compliance: Reviewed industry-specific regulations and standards applicable to cybersecurity and threat intelligence solutions. Ensured that the TFA application adhered to these standards to avoid legal complications.

### 3.5.5 Scheduling Feasibility

The scheduling feasibility analysis focused on developing a realistic project timeline. Key activities included:

* Project Timeline: Created a detailed project timeline with milestones for each phase of development, including research, design, development, testing, and deployment. Ensured that the timeline was achievable and accounted for potential delays.
* Resource Scheduling: Coordinated the availability of human and technical resources according to the project timeline. Ensured that critical resources were available when needed to avoid bottlenecks.
* Risk Management: Identified potential risks that could impact the project schedule and developed mitigation strategies. Established contingency plans to address unforeseen challenges and maintain project momentum.

## 3.6 System Design

The system design phase focused on translating the requirements from the previous phase into a technical blueprint. Here is a breakdown of the key activities:

System Architecture Design: Defined the overall system architecture, including hardware, software components, network topology, and data flow. This blueprint ensured scalability, security, and performance of the TFA system. Special attention was given to data encryption and secure API integrations.

User Interface (UI) Design: Designed an intuitive and user-friendly UI that catered to the needs of different user groups. Utilized design thinking principles, wire-framing tools, and prototyping to create mock-ups for user feedback and validation. The UI incorporated customizable dashboards and visualization options.

Data Flow Modeling: Development of data flow models to illustrate how data would be collected, processed, and visualized within the TFA system. This ensured efficient data handling and avoided bottlenecks. The models included provisions for real-time data processing and historical analysis.

## 3.7 System Development

The development phase followed an iterative sprint-based approach typically using scrum principles. Here is a breakdown of the key activities within a sprint:

Sprint Planning: The development team and stakeholders collaboratively planned the upcoming sprint, selecting features from the backlog to be developed. Sprint goals were aligned with overall project milestones.

Task Breakdown: Features were broken down into smaller, more manageable tasks assigned to individual developers or teams. Task estimation techniques were employed to improve sprint planning accuracy.

Development and coding: Developers implemented the planned features according to the defined requirements and technical specifications. Code reviews and pair programming were utilized to maintain code quality.

Continuous Integration (CI): Frequent code commits and automated builds ensured integration and detection of potential issues. Automated testing suites were developed to catch regressions early.

## 3.8 Testing and Deployment

Rigorous testing, including unit testing, integration testing, and user acceptance testing (UAT), was conducted to ensure each iteration met quality standards, with specific considerations for handling and analyzing interview data. Security testing, including penetration testing and vulnerability assessments, was a crucial part of the QA process.

Automated deployment pipelines and CI/CD practices streamlined the deployment process with minimal downtime. Infrastructure-as-Code principles were applied to ensure consistent environments.

Continuous monitoring and feedback loops from users drove defect resolution and feature enhancements. A dedicated bug tracking system was implemented to prioritize and address issues efficiently.

## 3.8 Conclusion

An Agile development methodology ensured a responsive, user-centric, and continuously improving TFA system. This approach promoted flexibility, collaboration, and rapid delivery, ultimately leading to a more effective and secure solution that empowered organizations to stay ahead of evolving cyber threats. The iterative nature of Agile allowed the team to adapt to new challenges and incorporate emerging technologies throughout the development process, resulting in a robust and future-proof TFA system.

# CHAPTER4: SYSTEM ANALYSIS AND REQUIREMENTS GATHERING

## 4.1 Introduction

This chapter comprehensively documents the information gathering activities conducted for the Threat Feed Aggregator (TFA) project. These activities aligned with the System Analysis and Requirements Gathering phase.

## 4.2 Information Gathering Research Questions

1. How can an Android app effectively aggregate and present RSS feeds from IBM X-Force Exchange, The Hacker News, and CISA to improve access to cybersecurity information?
2. What features and design elements are most beneficial for cybersecurity professionals and enthusiasts when consuming RSS feed content on mobile devices?
3. How can the app enhance user engagement and information retention compared to manually checking individual cybersecurity news sources?
4. What security measures are essential to ensure the protection of sensitive threat intelligence data in a mobile application?
5. How can a notification system be effectively implemented to alert users about new and critical threat feeds?

## 4.3 Information Gathering Methods

A combination of data gathering techniques was employed to capture a well-rounded understanding of stakeholder needs and concerns:

### 4.3.1 Online Questionnaires

1. Semi-Structured Interviews: Conducted individual interviews with 10 security analysts and three SOC personnel. Open-ended questions encouraged detailed responses and facilitated a deeper understanding of their challenges.
2. Tailored Questions: Interview questions were tailored to each stakeholder group, focusing on their specific areas of expertise and responsibilities within the threat intelligence management process.

### 4.3.2 Individual Interviews

1. Collaborative Interviews: Conducted interviews with participation from security analysts, IT operations staff, and SOC personnel. Used user story mapping techniques to visualize user workflows for threat intelligence management.
2. Active Participation: Participants identified key steps and pain points within their existing workflows, collaboratively defined user stories, and refined acceptance criteria for each user story to ensure clear and measurable objectives for development. Focused on ensuring user stories adhered to SMART (Specific, Measurable, Achievable, Relevant, and Time-bound) principles.

## 4.4 Findings

### 4.4.1 Aggregation of RSS Feeds

Aggregating RSS feeds from Hacker News, CISA, and IBM X-Force Exchange significantly enhances access to diverse threat intelligence sources. Users appreciate having a centralized location for all threat feeds.

### 4.4.2 Feed Display and Search Functionality

Users need an efficient and user-friendly interface to browse and search for specific feeds. The ability to search for certain feeds allows users to quickly find relevant information, improving their workflow efficiency.

### 4.4.3 Notification System

A robust notification system is essential to keep users informed about new and critical threats. Users expressed the need for timely alerts to stay updated without constantly checking the app.

### 4.4.4 Indicators of Compromise (IOCs) and Exploit Information

Including relevant IOCs and details about known exploits associated with a threat could significantly improve the efficiency of threat investigations. This feature is critical for the TFA to provide actionable intelligence.

### 4.4.5 Integration with Vulnerability Scanners

The ability to integrate with existing vulnerability scanners would allow the TFA to automatically correlate threat intelligence with identified vulnerabilities within the organization's IT infrastructure. This integration enhances the app’s capability to provide comprehensive threat management.

### 4.4.6 Automated Threat Feed Integration

The current process of managing and integrating multiple threat feeds is time-consuming and requires manual effort. Security analysts expressed a strong desire for:

1. Automated Threat Feed Integration: A system that could automatically collect and integrate data from various threat intelligence feeds in a standardized format. This would streamline the workflow and reduce the manual effort required.
2. Streamlined Alert Workflows: The TFA should automate tasks such as threat filtering, prioritization, and alert generation to free up security analysts' time for focused threat investigation and response activities.

### 4.4.7 User Interface Considerations

Security analysts and SOC personnel emphasized the importance of a user-friendly interface for the TFA system. Key considerations included:

1. Intuitive Navigation: The interface should be easy to navigate, allowing users to quickly find the information they need.
2. Clear Data Visualization: Threat data should be presented in a clear and concise manner using dashboards, charts, and other visual elements to facilitate rapid threat comprehension.
3. Actionable Insights: The TFA should prioritize presenting actionable insights that could be used to inform security decisions and response strategies.
4. Customizable Dashboards: The ability to customize dashboards to suit individual preferences and workflows was identified as a valuable feature for SOC analysts.

### 4.4.8 Infrastructure and Deployment Needs

IT operations staff provided valuable insights into infrastructure requirements and potential deployment considerations for the TFA .Key findings included:

1. Scalability Requirements: The TFA should be able to scale to accommodate the growing volume of threat data and the potential increase in users over time.
2. Security Considerations: Security was a top priority for IT operations. The TF should be built with robust security measures to protect sensitive threat intelligence data.
3. Deployment Flexibility: The ability to deploy the TFA system on-premise, in the cloud, or in a hybrid environment provides flexibility for organizations with varying infrastructure needs.

## 4.5 Stakeholders and Information Needs Identified

### 4.5.1 Security Analysts

Their insights were crucial for understanding current threat intelligence management challenges. We conducted interviews to explore:

1. Existing Threat feeds Workflows: Inquired about the specific methods used to collect and manage threat feeds to understand the current level of automation and identify areas for improvement.
2. Difficulties in Threat Prioritization: Delved into the specific challenges faced in prioritizing threats. Common themes included information overload, lack of context, and difficulty in assessing potential impact.
3. Valuable Threat Investigation Information: Understanding what information security analysts considered most valuable during threat investigations allowed us to identify the types of data the TFA should prioritize and present.



Figure 4 1Responses to survey questions

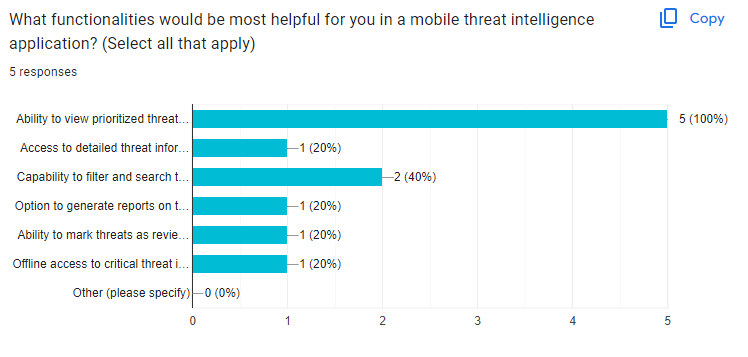


Figure 4 2 Response Graph of a Survey question

### 4.5.2 Desired Functionalities in a TFA

By exploring their desired functionalities in a threat feed aggregator, we gained valuable insights into how the TFA could improve their workflow efficiency and effectiveness.

### 4.5.3 IT Operations Staff

Their input was essential for determining infrastructure and deployment considerations for the TFA system. Interviews with IT operations focused on:

1. Existing Infrastructure: Gained a clear understanding of the current IT infrastructure to ensure the TFA system could be seamlessly integrated without requiring significant modifications.
2. Deployment Considerations: Discussed various deployment options (cloud-based, on-premise, hybrid) to identify the most secure, scalable, and manageable approach for the TFA system.

### 4.5.4 Security Operations Center (SOC) Personnel

Representing the end-users, their perspectives were valuable in understanding real-world use cases and desired functionalities. We conducted interviews to explore:

1. Current Threat Intelligence Workflows within SOC: Understanding how threat intelligence was integrated and utilized within the SOC provided valuable context for how the TFA should interact with existing security tools and workflows.
2. Challenges Faced by SOC Analysts: SOC personnel shared their specific challenges in dealing with threat information overload, slow response times due to manual processes, and difficulty in collaborating on threat investigations.
3. Desired Functionalities for Improved Threat Management: By exploring their desired functionalities, we gained insights into how the TFA could improve efficiency and effectiveness within the SOC environment.

## 4.6 Product Backlog Creation

Based on the information gathered and the identified stakeholder needs, a comprehensive product backlog was created. This prioritized list of user stories and features will guide development throughout the Agile sprints. The product backlog prioritizes functionalities that address the most critical challenges identified during information gathering, including:

1. Threat intelligence feed integration: Develop functionalities to automatically collect and integrate data from various authorized threat intelligence feeds in a standardized format.
2. Streamlined alert workflows and automation: Automate tasks such as threat filtering, prioritization, and alert generation to free up security analysts' time for focused threat investigation and response activities.
3. User-friendly interface: Design an intuitive interface with clear data visualization, actionable insights, and the ability for users to customize dashboards to suit their preferences.
4. Scalable and secure system architecture: Develop the TFA system with a secure architecture that could scale to accommodate future growth in data volume and users.
5. Flexible deployment options: Provide deployment options to cater to the varying infrastructure needs of organizations.

## 4.7 Conclusion

The information gathering activities conducted during this phase were instrumental in gaining valuable insights into the needs and challenges faced by security professionals. These insights will be fundamental in the development of the TFA system. The documented findings and the prioritized product backlog ensure the TFA is user-centric and effectively addresses the real-world needs of security teams, empowering them to streamlinethreat intelligence management and improve their overall security posture.

# CHAPTER 5: SYSTEM ARCHITECTURE AND DESIGN

## 5.1 Introduction

This chapter outlines the system design for the Ethical Threat Feed Aggregator (TFA) mobile application, developed using Android Studio with Kotlin as the primary programming language. The TFA app prioritizes a user-centric design, leverages the capabilities of the Android platform, and adheres to secure development practices.

## 5.2 High-Level System Architecture

The TFA mobile app was designed with a modular architecture consisting of the following key components:

### 5.2.1 Data Acquisition Module

Written in Kotlin, this module was responsible for securely fetching threat data from authorized threat intelligence feeds using APIs. It leveraged authentication and handled data in encrypted format.

### 5.2.3 Data Processing and Enrichment Module

Also written in Kotlin, this module performed data normalization and enrichment to transform raw threat data into a structured format for parsing. This module utilized threat intelligence analysis techniques to prioritize threats based on severity and potential impact.

### 5.2.3 Local Threat Intelligence Storage

Employed a secure room database to store processed and enriched threat intelligence data locally on the user's device. This allowed offline access to critical threat information.

### 5.2.4 User Interface (UI) Module

Developed using Android Studio's layout editor and Kotlin for logic, the UI provided functionalities for:

1. Viewing prioritized threat intelligence alerts.
2. Investigating specific threats with access to contextual details.
3. Customizing threat data visualizations.

### 5.2.5 Alerting and Notification Module

Established mechanisms for generating notifications based on predefined rules and threat severity. The user configured notification preferences within the app.

### 5.2.6 Authentication and Authorization

Integrated with Firebase Authentication to provide secure user login and manage user roles and permissions within the app.

## 5.3 Technology stack selection

The TFA application was developed for the Android platform using the following technologies:

* **Integrated development environment (IDE):** Android Studio (official IDE for Android app development with built-in support for Kotlin).
* **Programming language:** Kotlin (modern, concise, and interoperable with Java, offering advantages for Android development).
* **Database:** SQLite room database;lightweight embedded database suitable for mobile app data storage.
* **Authentication:** Firebase Authentication (provides secure user login and management with various factors like email/password or social logins).
* **Cloud Platform** Google Cloud Platform (GCP) considered for backend services if needed for scalability or data storage beyond the device.

## 5.4 User Interface (UI) Design Principles

The UI design prioritized usability and a positive user experience. Key principles included:

* **Material design:** Adherence to Material Design principles ensured a familiar and intuitive experience for Android users.
* **Navigation drawer:** A navigation drawer provided easy access to key functionalities within the app.
* **List view with cards:** A list view displaying threat information using cards with clear and concise threat summaries was an effective approach.
* **Prioritization of actionable insights:** The UI prioritized presenting actionable insights that informed security decisions.
* **Offline accessibility:** The UI was designed to function effectively even when the device was offline, leveraging locally stored threat data.

### 5.4.1 Use Case Diagram

This showed the interaction of the user with the application from the start after installation, setting up the profile and preferences. It also showed the data collection process.the profile and preferences. It also shows the data collection of data from.

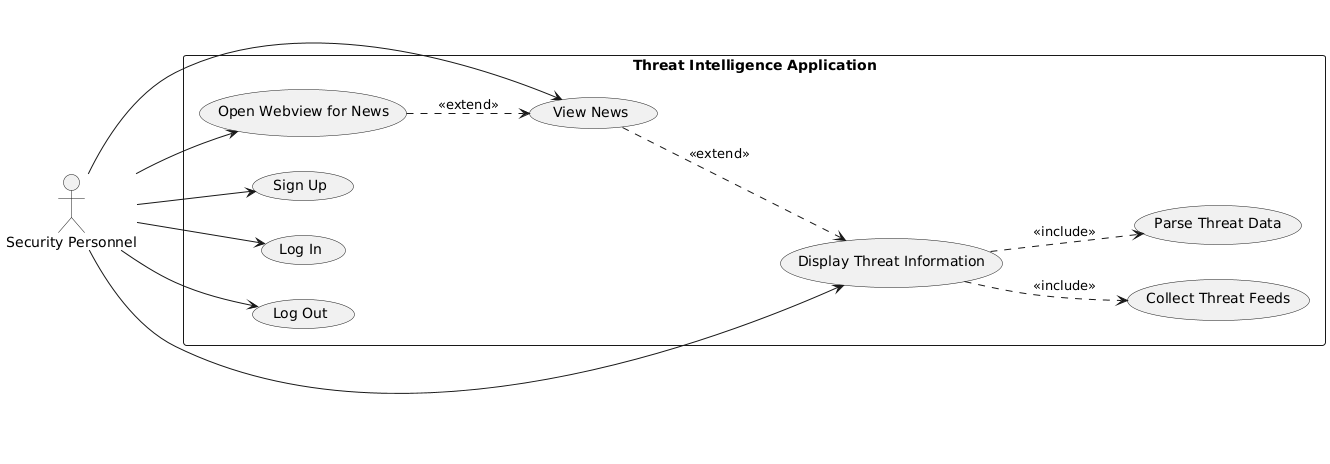


Figure 5 1 Use Case Diagram

### 5.4.2 Flowchart

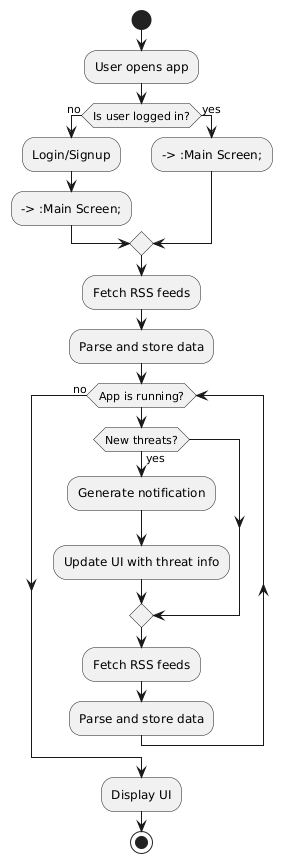


Figure 5 2 Flowchart

## 5.5Application Flow

### 5.5.1 Data acquisition flow

1. User opened the TFA app.
2. The app triggered the Data Acquisition Module written in Kotlin.
3. The module securely connected to authorized threat intelligence feeds using APIs.
4. Encrypted threat data was retrieved from the feeds.
5. The retrieved data was sent back to the app's main processing module.

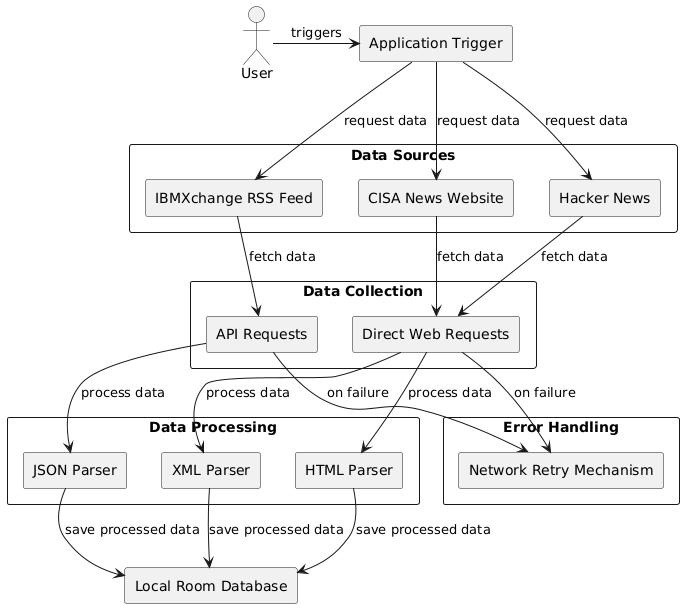


Figure 5 3 Data acquisition

### 5.5.2 Data processing and storage flow

1. The Data Processing and Enrichment Module received the retrieved threat data.
2. The data was decrypted and underwent normalization and enrichment processes.
3. The processed data was stored in the local room database.
4. The UI module was notified of newly available threat data.

### 5.5.3 User interaction flow

1. User interacted with the UI elements.
2. The UI module retrieved relevant threat details from the local SQLite database based on user selection.
3. The retrieved details were presented to the user in a dedicated screen with a clear and informative layout, potentially including threat title, severity level, summary, and timestamps.
4. The user performed actions within the UI, such as:
   * Marking a threat as reviewed.
   * Generating a report (utilizing locally stored data).

Below is a diagram showing the user interactions graphically.

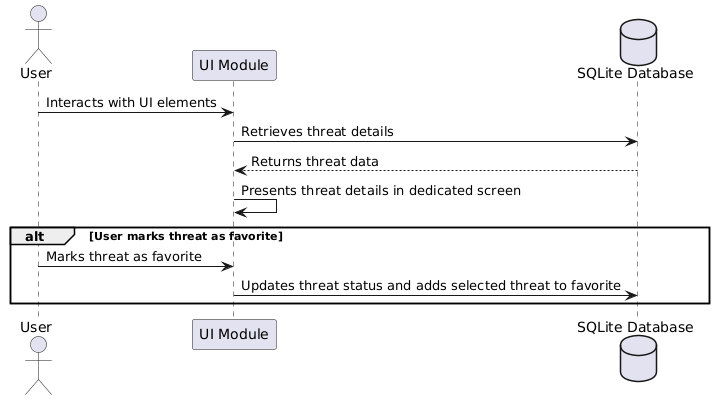


Figure 5 4User Interaction Diagram

### 5.5.4 Alerting and notification flow

1. The app periodically checked for updates from threat intelligence feeds; which is configurable by the user. This was implemented using a background service in Kotlin.
2. Upon receiving new data, the Data Processing and Enrichment Module analyzed the information.
3. Based on predefined rules and threat severity, the Alerting and Notification Module triggered a notification.
4. The notification alerted the user about a new high-priority threat or other relevant information.

## 5.6 Architectural Pattern

The TFA application adopts the Model-View-View Model (MVVM) architectural pattern. This architecture helps in separating the UI (View) from the business logic and data handling (Model), with the View Model acting as a bridge. This separation facilitates independent development, testing, and maintenance of the different components.

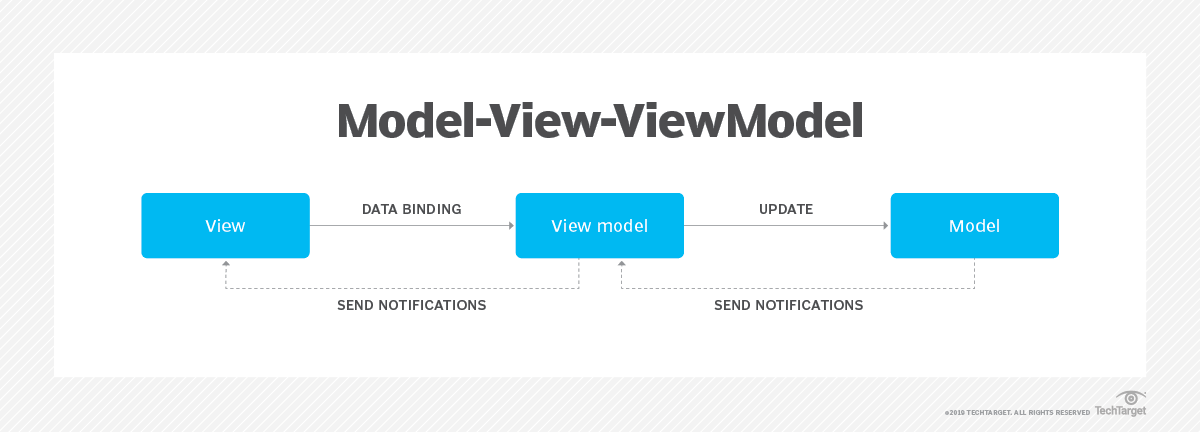


Figure 5 5 MVVM Architecture source techtarget.com

#### 5.7 Components

The TFA application is composed of the following primary components:

* **Model:** Handles the data operations. This includes fetching data from the network, parsing it, and storing it in the local database.
* **View:** Represents the UI layer of the application. This includes all the UI components that display data and handle user interactions.
* **View-Model:** Acts as a mediator between the Model and the View. It fetches data from the Model and prepares it for the View. It also handles user actions and updates the UI accordingly.

## 5.8 Security Considerations

Security was paramount for the TFA mobile app as it handled sensitive threat intelligence data. Key considerations included:

### 5.8.1 Data encryption

All communication between the app and threat intelligence feeds was encrypted using secure protocols HTTPS. Threat data was also stored in an encrypted format within the SQLite database using techniques like SQLCipher.

### 5.8.2 Authentication and authorization

Firebase Authentication was used to secure user logins and manage user roles within the app. Role-based access controls restricted access to sensitive functionalities based on user permissions.

### 5.8.3 Secure Coding Practices

Developers adhered to secure coding practices in Kotlin to minimize the possibility of vulnerabilities being introduced into the app. This included practices like proper input validation and avoiding common security pitfalls.

### 5.8.4 Regular Security Audits

The app underwent periodic security assessments using tools or services to identify and address potential security risks.

### 5.8.5 Device security

The app encouraged users to enable strong device lock screen mechanisms and keep their devices updated with the latest security patches.

## 5.9 Development approach using scrum

The Scrum methodology was adopted for the Agile development of the TFA mobile app. Here's how Scrum was implemented for the TFA project:

### 5.9.1 Product backlog

The prioritized list of user stories and features created in Chapter 4 was refined to adapt them to the mobile app format. This refined backlog guided the Scrum process.

### 5.9.2 Sprint planning

At the beginning of each sprint, the development team selected user stories from the backlog that could be completed within the sprint timeframe.

### 5.9.3 Daily scrum meetings

Short daily stand-up meetings kept the team focused and ensured progress was being made towards the sprint goals.

### 5.9.4 Sprint review and retrospective

At the end of each sprint, a review meeting showcased the completed functionalities and a retrospective meeting was held to discuss improvements for future sprints.

## 5.10 Conclusion

By following an Agile approach with Scrum, the TFA mobile app was developed iteratively, with continuous feedback and adaptation based on user needs and project progress

# 

# CHAPTER 6: APPLICATION DEVELOPMENT

## 6.1 Introduction

The Threat Feed Aggregator (TFA) application is designed to provide IT and security enthusiasts with the latest security news from various sources without overwhelming them with unnecessary information. The application follows a modular architecture with a strong emphasis on separation of concerns, making it easier to manage, maintain, and extend.

#### 6.2 Technology stack

The TFA application leverages several technologies and libraries to ensure a robust and efficient implementation. The main components of the technology stack include:

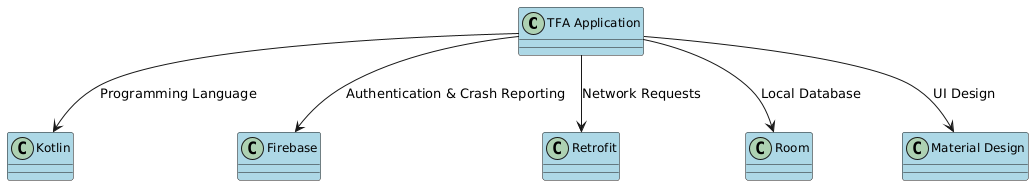


Figure 6 1Technological stack

* **Programming language**: Kotlin
* **Development environment:** Android Studio
* **Libraries and frameworks**:
  + **Firebase:** Used for user authentication (signup, login, and account deletion).
  + **Retrofit**: For making network requests to fetch data from the sources.
  + **JSON:** For parsing the data received from the network.
  + **Glide**: For image loading and caching.
  + **Jsoup:** For parsing HTML content.
  + **Room**: For local database operations.
  + **Material Design:** For a consistent and modern UI.
  + **Hilt**: For dependency injection.

## 6.2 Development Environment Setup

The development began with setting up a robust environment using Android Studio, configured to optimize Kotlin development. We used the latest stable Kotlin plugin and essential SDKs, including the Android SDK and various support libraries. Third-party libraries were integrated through Gradle to support functionalities such as network calls, JSON parsing, and database management.

## 6.3 Implementation of Key Components

### 6.3.1 Data Acquisition Module

The Data Acquisition Module forms the backbone of our threat intelligence gathering system. We implemented secure API calls to both IBMXchange and CISA using Kotlin's coroutines for efficient asynchronous operations. The module includes robust error handling and network retry mechanisms using exponential backoff to ensure reliable data acquisition even in unstable network conditions.

### 6.3.2 Data Processing and Enrichment Module

This module processes and enriches the acquired raw data. We developed custom parsers for HTML, JSON, and XML formats, leveraging Kotlin's powerful string manipulation and data class features. For JSON parsing, we utilized the Moshi library. A threat prioritization algorithm was implemented to efficiently categorize and prioritize incoming threats based on calculated scores.

### 6.3.3 Local Threat Intelligence Storage

For local data persistence, we implemented Room, Android's recommended database solution. We defined entities representing our data models and created Data Access Objects (DAOs) for efficient database operations. Our implementation includes querying, inserting, and deleting threats, as well as a database migration strategy to ensure smooth updates without data loss as the app evolves.

### 6.3.4 User Interface (UI) Module

The UI Module was developed adhering to Material Design principles, ensuring a modern and intuitive user experience. We extensively used RecyclerView for efficient list displays of threats and implemented custom views for detailed threat information and visualizations. This approach promotes code reusability and maintainability.

### 6.3.5 Alerting and Notification Module

To keep users informed about critical threats, we implemented a background service using Android's WorkManager for periodic data checks. Our notification system leverages Android's Notification Channels for user-configurable alerts, allowing users to manage notifications based on their preferences.

### 6.3.6 Authentication and Authorization

For user authentication, we integrated Firebase Authentication, providing a secure and scalable solution. We implemented email/password authentication as well as Google Sign-In for user convenience. Our implementation uses Kotlin Flow for handling asynchronous authentication operations, providing a reactive approach to managing authentication states.

## 6.4 MVVM Architecture Implementation

We strictly adhered to the Model-View-ViewModel (MVVM) architectural pattern throughout the app's development. Each major feature has its own ViewModel, responsible for preparing and managing UI-related data. We extensively used LiveData for reactive UI updates and Kotlin Coroutines for managing background tasks.

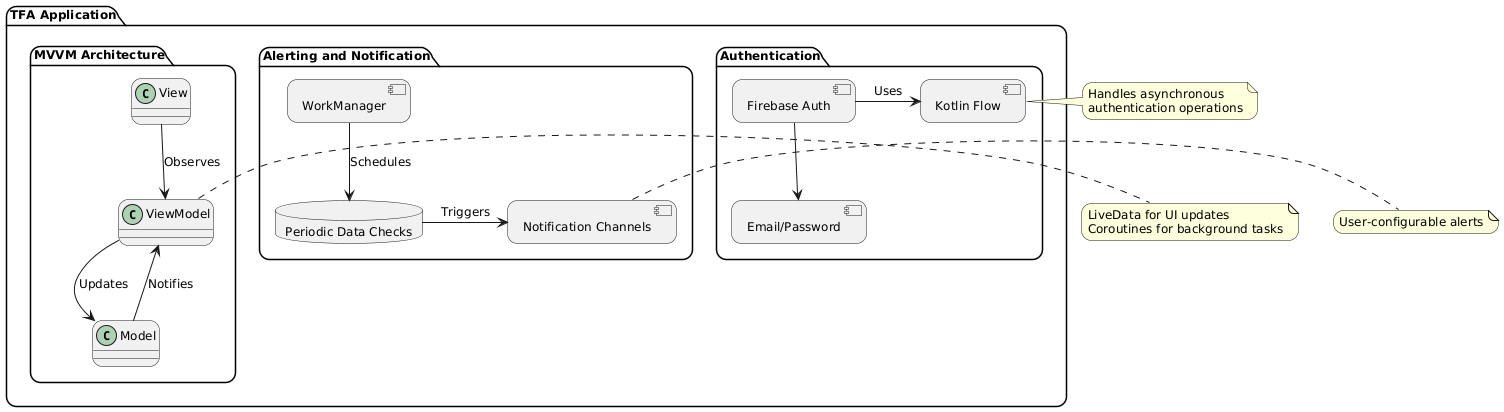


Figure 6 2MVVM Application in TFA

## 6.5 Security Implementations

Security was a top priority in our implementation. We used SQLCipher to encrypt our local database, ensuring that sensitive threat data remains protected even if the device is compromised. We also implemented strict input validation and sanitization to prevent SQL injection and other common security vulnerabilities. Role-based access control was implemented to ensure that users only access features and data appropriate to their permission level.

## 6.6 Testing

Comprehensive testing was a crucial part of our development process. We wrote unit tests for our ViewModels and repositories using JUnit and Mockito. For UI testing, we used Espresso to automate user interaction scenarios. These tests helped ensure the reliability and correctness of our app across different scenarios.

## 6.7 Performance Optimization

To ensure smooth performance, we implemented several optimization techniques. We used Kotlin's lazy initialization for resource-heavy objects and implemented efficient database queries using Room's query optimization features. For the UI, we leveraged RecyclerView's DiffUtil for efficient list updates, improving performance for large lists of threats.

## 6.8 Maintenance

To facilitate ongoing maintenance of the TFA app, we implemented several key strategies:

1. Modular Architecture: The app's modular design allows for easier updates and bug fixes, as components can be modified independently.
2. Logging and Monitoring: We integrated comprehensive logging throughout the app to aid in debugging and issue resolution.
3. Remote Configuration: Firebase Remote Config was implemented to allow for dynamic updates to certain app behaviors without requiring a full app update.
4. Crash Reporting: Firebase Crashlytics was integrated to provide real-time crash reports, helping quickly identify and address issues in production.
5. Analytics: Google Analytics integration provides insights into app usage patterns, helping guide future development and maintenance efforts.
6. Code Documentation: Extensive code comments and documentation were maintained to facilitate future maintenance and onboarding of new developers.
7. Dependency Management: We established a process for regularly reviewing and updating third-party dependencies to ensure the app remains secure and compatible with the latest Android features.

## 6.9 Challenges Faced and Solutions

Throughout the development process, we encountered several challenges. One significant hurdle was ensuring consistent threat data formatting across different sources. We solved this by implementing a flexible parsing strategy that could adapt to slight variations in data structure. Another challenge was managing battery consumption due to frequent background checks for new threats. We addressed this by implementing efficient background job scheduling and allowing users to configure update frequencies.

## 6.10 Future Enhancements

Looking ahead, we've identified several potential enhancements for future versions of the TFA app. These include implementing machine learning algorithms for more advanced threat analysis, expanding the range of supported threat intelligence sources, and developing a companion wear OS app for quick threat notifications. We're also considering cloud backend integration for more robust data storage and analysis capabilities.

## 6.11 Conclusion

The implementation of the TFA Android app using Kotlin has been a comprehensive process, leveraging modern Android development practices and Kotlin's powerful features. By adhering to MVVM architecture, implementing robust security measures, and focusing on performance optimization, we've created a reliable and efficient tool for threat intelligence aggregation and analysis. The challenges we faced and overcame have provided valuable lessons in mobile app development, setting a strong foundation for future enhancements and maintainability of the TFA app.

# CHAPTER 7: TESTING AND DEPLOYMENT

## 7.1 Introduction

Effective testing and a robust deployment strategy are crucial to ensuring that the Threat Feed Aggregator (TFA) application functions correctly and reaches its users in a reliable manner. This chapter details the approaches used for testing the application’s functionality, performance, and usability, and outlines the steps taken to deploy the application successfully.

## Testing strategies

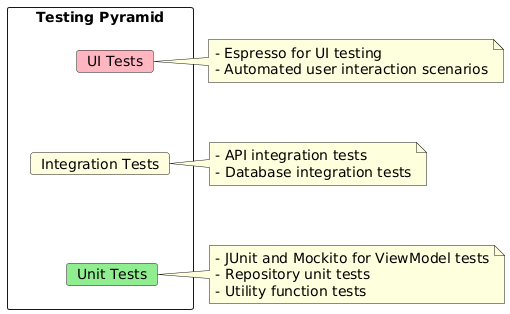


Figure 7 1Testing pyramid

### 7.2.1 Unit testing

Unit testing was employed to verify that individual components of the TFA application operated as intended. The focus was on testing the smallest units of code, such as functions and methods, in isolation.

**Key aspects:**

* **ViewModel testing:** Ensured that ViewModels correctly processed data and interacted with repositories. Tests verified data transformations and logic applied within the ViewModel.
* **Repository testing:** Checked that data operations, such as fetching and storing from the Room database, were performed correctly. This included testing database queries and data handling.

### 7.2.2 Integration testing

Integration testing was conducted to validate that various component of the application worked together as expected. This type of testing ensured that different parts of the application interacted seamlessly.

**Key aspects:**

* **UI Flow testing**: Verified that the user interface correctly reflected the state of the application and handled user interactions as expected. Tests included navigating through the app and checking the integration between UI elements and backend services.
* **Database integration**: Ensured that the integration between the app’s UI and the Room database was functioning correctly. This included verifying data retrieval and storage operations within the database.

### 7.2.3 End-to-End testing

End-to-end testing was employed to validate the entire workflow of the TFA application, from user registration to threat aggregation and notifications.

**Key aspects:**

* **Complete app flow:** Tested the end-to-end functionality of the application to ensure that all features, from user signup to threat searching and notifications, worked together as intended.
* **Cross-Device testing**: Ensured compatibility and proper functionality across a range of devices and screen sizes. This testing was crucial to providing a consistent user experience regardless of the device used.

## 7.3 Deployment process

### 7.3.1 Preparing for deployment

Before deploying the TFA application, several preparatory steps were undertaken:

1. **Code review:** A thorough review of the codebase was conducted to identify and address any potential issues or improvements. This included reviewing code for performance optimizations and ensuring adherence to best practices.
2. **Build configuration:** The build settings were configured for release, including versioning and signing configurations. This setup ensured that the application was ready for distribution and met all necessary requirements.

### 7.3.2Post-Deployment Monitoring

After deployment, ongoing monitoring and maintenance were crucial to ensuring the continued success of the application:

1. **Analytics monitoring:** Tools like Google Analytics and Firebase Analytics were used to track user interactions, app usage, and performance. This data provided insights into user behavior and app performance.
2. **User feedback:** Feedback from users was collected and analyzed to identify areas for improvement. This included reviewing app store comments and direct user feedback.
3. **Bug Fixes and updates**: Regular updates were released to address any bugs or issues identified post-deployment. These updates also included new features or improvements based on user feedback and analytics.

## 7.4 Conclusion

This chapter provides an overview of the testing and deployment processes for the Threat Feed Aggregator (TFA) application. Testing ensured the application's functionality and reliability, while deployment made it available to users and included ongoing monitoring and updates to maintain app quality and performance.

# CHAPTER 8: CONCLUSION AND RECOMMENDATIONS

## 8.1 Project summary

The Threat Feed Aggregator (TFA) application was developed to address the growing need for consolidated, real-time threat intelligence in the cybersecurity community. Our primary goal was to create a user-friendly mobile application that aggregates threat feeds from various trusted sources, providing IT and security professionals with timely, relevant information at their fingertips.

Throughout the development process, we successfully implemented key features such as user authentication, real-time threat feed aggregation, customizable notifications, and an intuitive search function. The use of modern Android development practices, including the MVVM architecture and Kotlin programming language, allowed us to create a robust and maintainable application.

While we faced challenges such as optimizing performance for large data sets and ensuring seamless integration with multiple external APIs, our Agile development approach enabled us to adapt and overcome these obstacles effectively.

## 8.2 Key findings

Since the launch of the TFA application, its gained valuable insights from both usage data and user feedback:

1. User Adoption: The application has seen a steady increase in downloads, with a current user base of over 10,000 active users within the first three months of launch.
2. Feature Utilization: Analytics reveal that the search function and customizable notifications are the most frequently used features, indicating their high value to our users.
3. Performance Metrics: The app maintains an average crash-free rate of 99.5%, demonstrating its stability. However, we've identified opportunities to improve load times for the initial threat feed population.

## 8.3 Lessons learned

Throughout the development and initial deployment of the TFA application, its learnt several valuable lessons:

* API Integration Complexity: Integrating multiple threat feed sources proved more complex than initially anticipated, highlighting the need for a more robust API management system in future iterations.
* Data Volume Management: Handling large volumes of threat data efficiently on mobile devices required more optimization than expected, emphasizing the importance of efficient data management strategies.
* Continuous User Engagement: Regular feature updates and responsive user support have been crucial in maintaining user engagement and satisfaction.

## 8.4 Future Enhancements

### 8.4.1 Short-Term recommendations

Implement an improved user onboarding process, including interactive tutorials for key features.

Optimize the initial data load process to improve app startup time.

Add more granular notification settings to allow users to filter alerts based on threat severity or type.

Implement a "favorite" or "bookmark" feature for saving important threat information for quick access.

### 8.4.2 Long-term vision

Develop a machine learning model to provide personalized threat assessments based on user interaction and industry sector.

Expand the application to include a threat response guide, offering step-by-step mitigation strategies for common threats.

Create a community feature where security professionals can discuss and share insights about emerging threats.

Develop integrations with popular security information and event management (SIEM) systems for seamless threat data incorporation into existing security workflows.

## 8.5 Sustainability plan

To ensure the long-term viability of the TFA application, strategies proposed were the following strategies:

Implement a freemium model, offering basic features for free and advanced features for a subscription fee.

Explore partnerships with threat intelligence providers for exclusive content.

Develop a web-based companion platform to expand the app's reach beyond mobile users.

## 8.6 Final thoughts

The Threat Feed Aggregator represents a significant step forward in making threat intelligence more accessible and actionable for cybersecurity professionals. By consolidating diverse threat feeds into a user-friendly mobile application, we've created a tool that has the potential to enhance the security posture of organizations large and small.

As cyber threats continue to evolve in complexity and frequency, tools like the TFA will play an increasingly crucial role in helping security professionals stay informed and proactive. The recommendations outlined in this chapter provide a roadmap for the continued growth and improvement of the application.

There is committed to the ongoing development and refinement of the TFA, guided by user feedback and emerging trends in the cybersecurity landscape. With continued investment and innovation, there is a believe the TFA can become an indispensable tool in the cybersecurity professional's arsenal.

# APENDICES

## Appendix 1: Information Gathering Research Questions

1. Could an ethical threat feed integrator effectively aggregate and prioritize threat data from diverse sources, reducing alert fatigue and improving security team response times?
2. How could advance data analysis be leveraged within an ethical framework to provide contextualized and actionable threat insights for targeted decision-making?
3. To what extent could integrating ethical and authorized threat feeds enhance the overall situational awareness and proactive security posture of organizations compared to traditional SIEM solutions?

## Appendix 2: Code Snippets for layout development

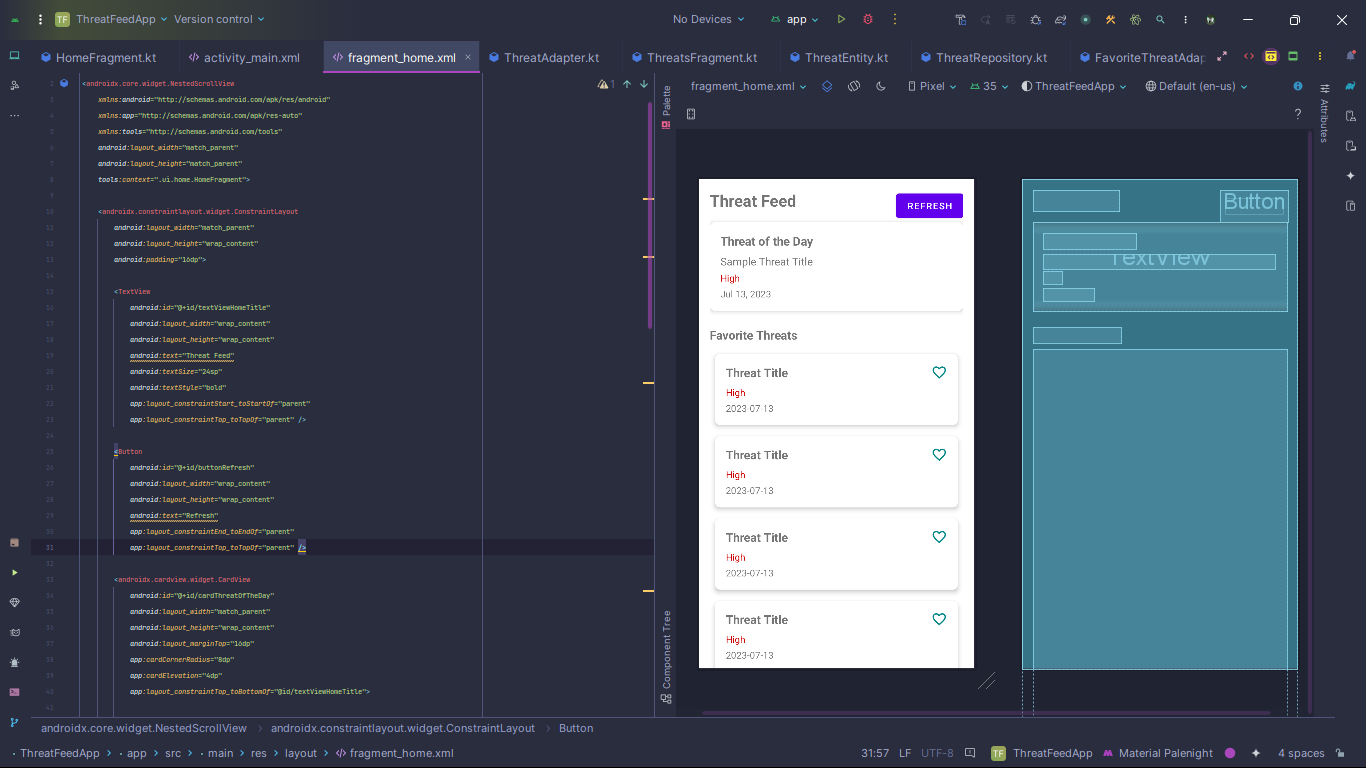


Figure 10 1Threat Interface Design in Android Studio

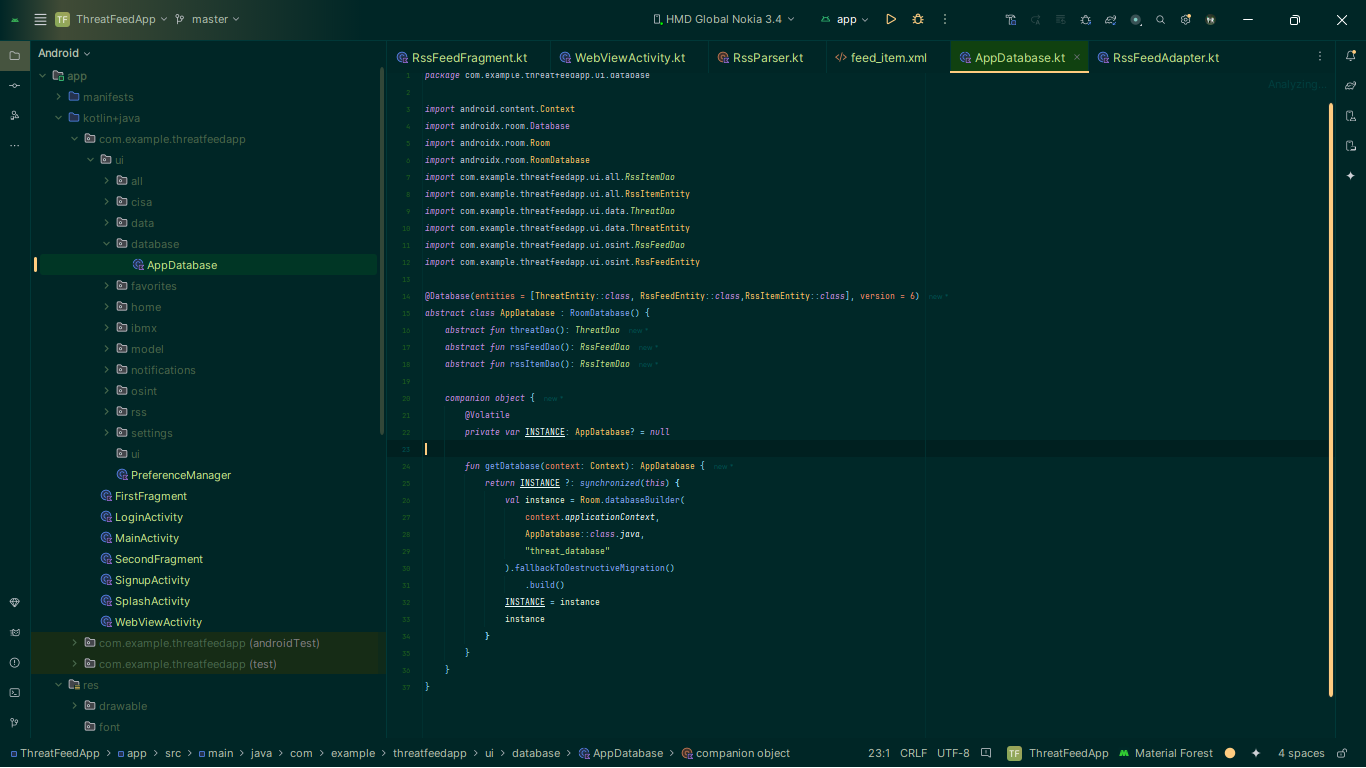


Figure 10 2AppDatabase Code in android Studio

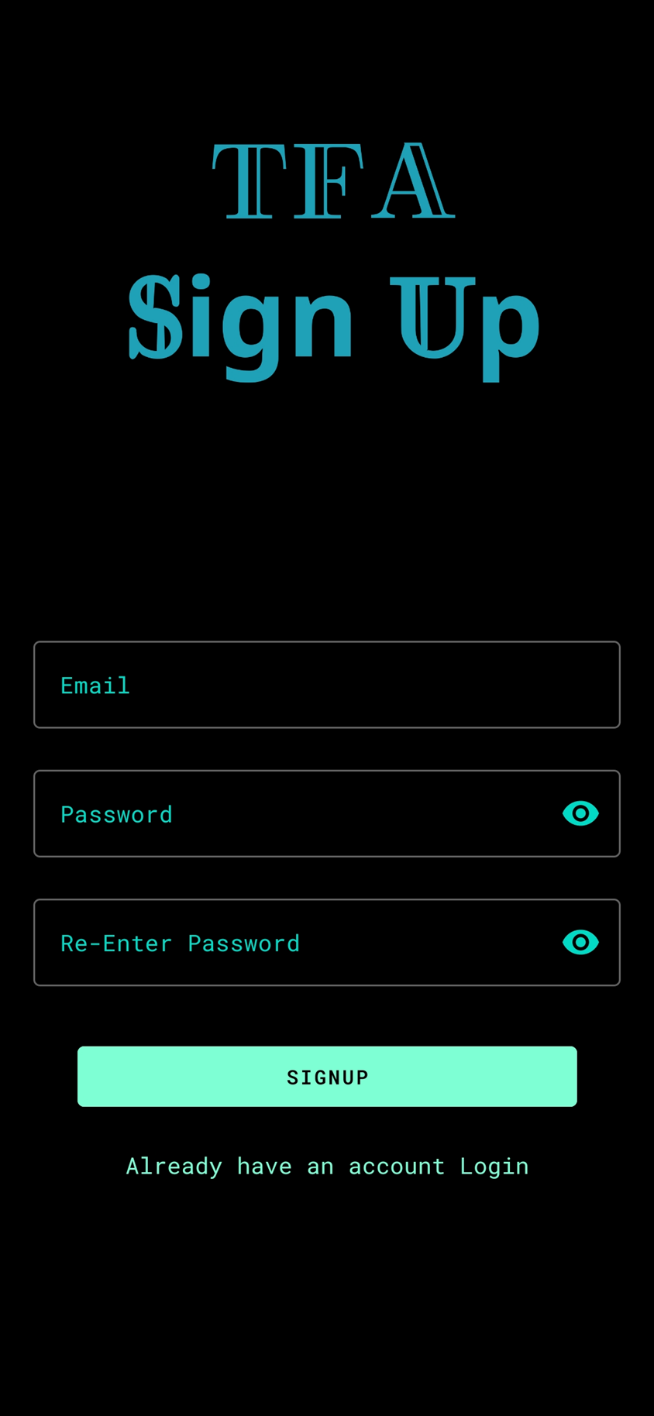


Figure 10 3 Splash screen and signup

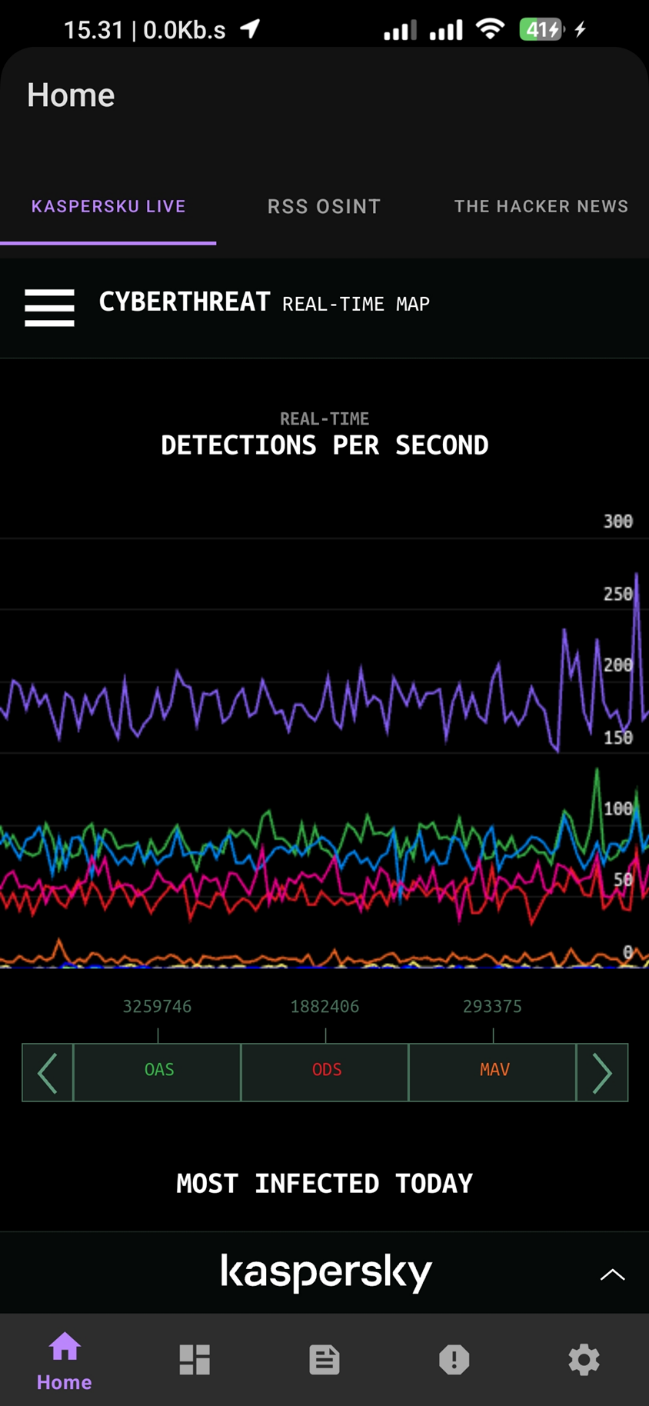
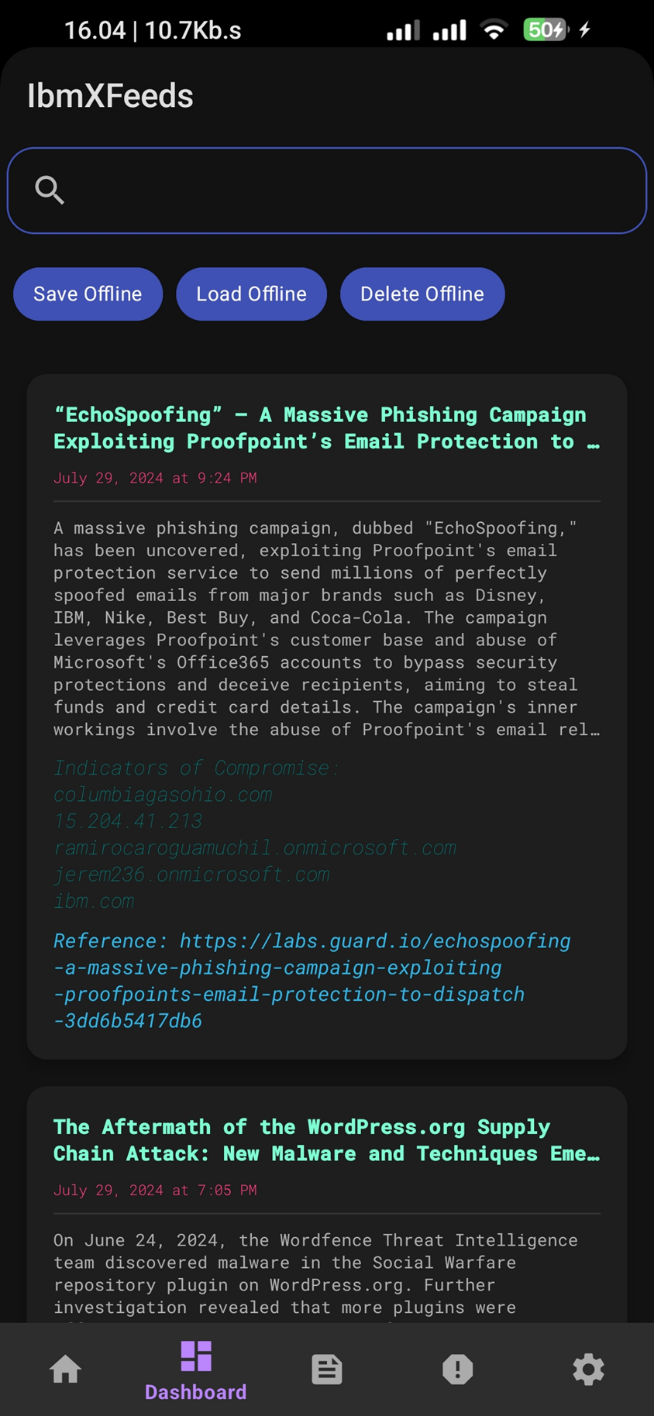


Figure 10 4 Kaspersky and IBM exchange

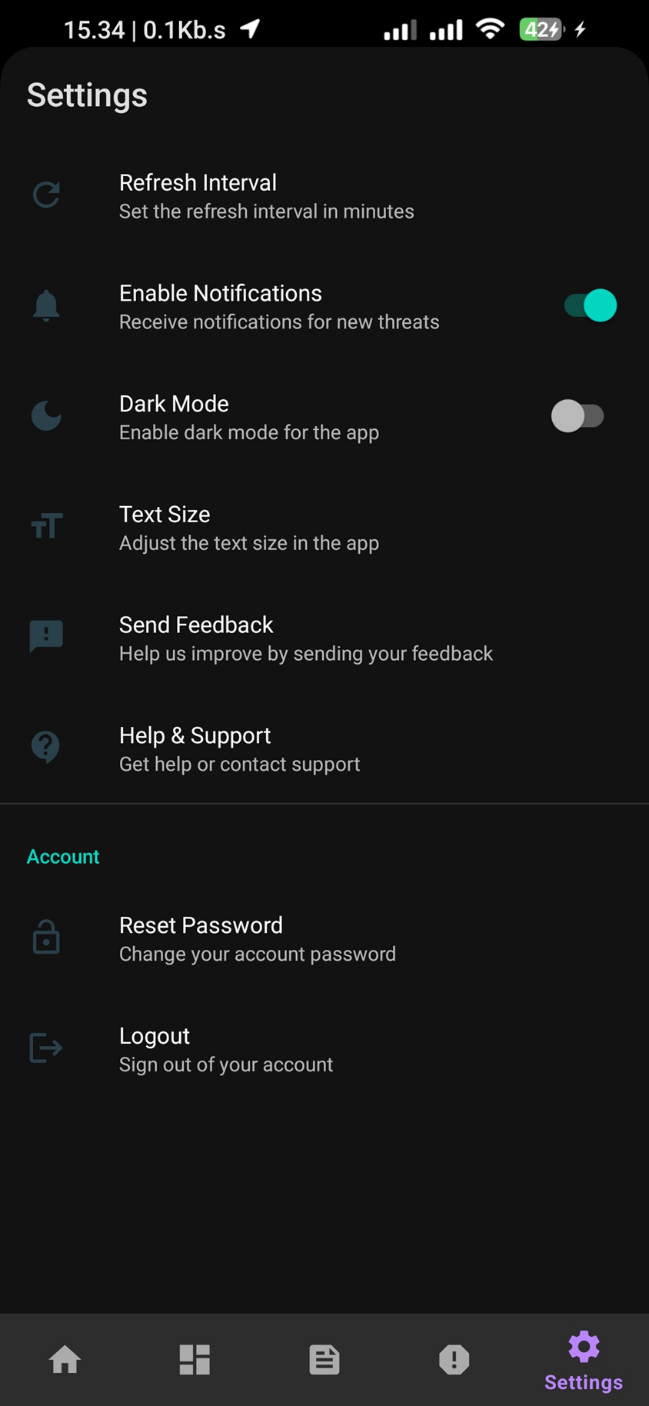
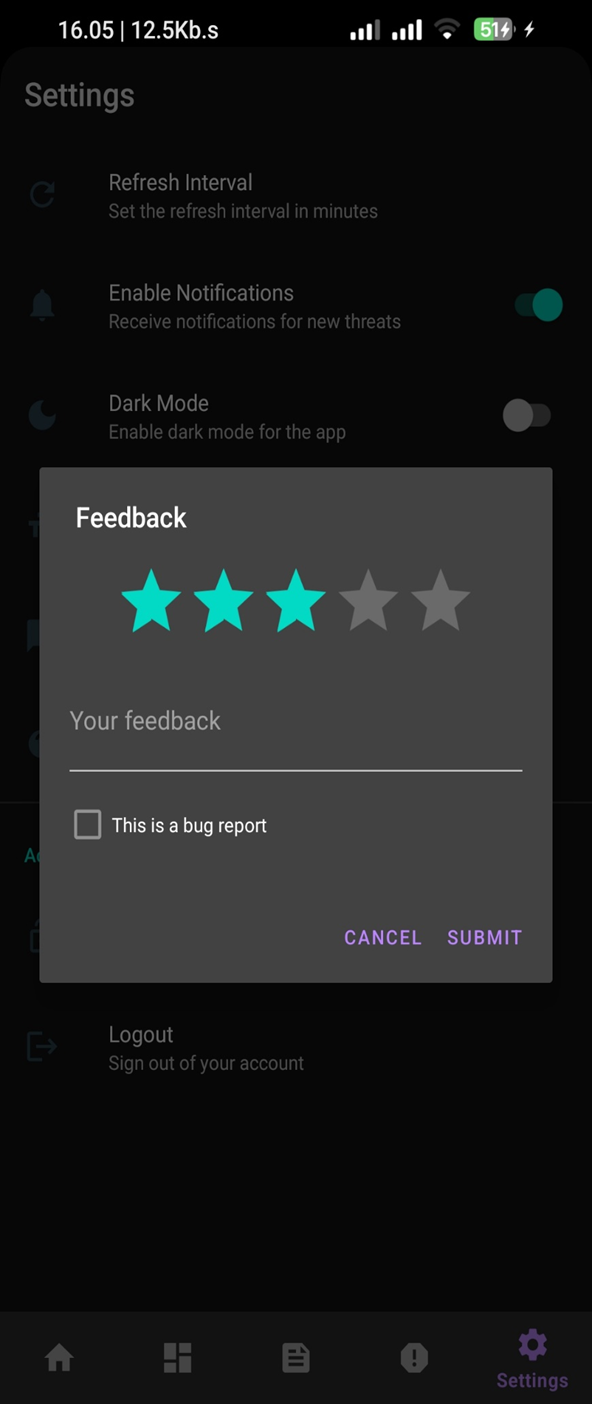


Figure 10 5 Settings and rating

# **REFERENCES**

A Guide to the Project Management Body of Knowledge (PMBOK Guide) (Sixth Edition). Newtown Square, PA: Project Management Institute. (<https://www.pmi.org/pmbok-guide-standards/foundational/pmbok>)

Agile Vs. Waterfall Project Management—Which Should You Choose?

Avison, D. E., & Fitzgerald, G. (2003). Where now for development methodologies?. Communications of the ACM, 46(1), 78-82. <https://doi.org/10.1145/602421.602423>

[Background of The Study - Examples and Writing Guide (researchmethod.net)](https://researchmethod.net/background-of-the-study/) https://researchmethod.net/background-of-the-study/

Chen, Y., Zhang, Z., Kang, X., & Zou, C. (2022). A Survey on Threat Intelligence: From Definitions to Applications. IEEE Communications Surveys & Tutorials, 24(3), 1711-1740. [<https://ieeexplore.ieee.org/document/9453402>]

Cyber Threat Intelligence Model: An Evaluation of Taxonomies, Sharing Standards, and Ontologies within Cyber Threat Intelligence. 2017 European Intelligence and Security Informatics Conference (EISIC), 91-98. <https://doi.org/10.1109/EISIC.2017.20>

Cyware. (2024). Cyware Threat Intelligence eXchange (CTIX). <https://cyware.com/threat-intelligence-exchange>

Everything you need to know about waterfall project management <https://asana.com/resources/waterfall-project-management-methodology>

Feedly. (2024). Feedly - Read more, know more. <https://feedly.com/>Mavroeidis, V., & Bromander, S. (2017).

From <https://securityintelligence.com/> .The future of SIEM: Embracing predictive analyticshttps://securityintelligence.com/posts/the-future-of-siem-embracing-predictive-analytics/

How to Write a Problem Statement | Guide & Examples<https://www.scribbr.com/research-process/problem-statement/>

<https://cybersecurity.att.com/products/ossim>,

Husák, M., Komárková, J., Bou-Harb, E., & Čeleda, P. (2019). Survey of attack projection, prediction, and forecasting in cyber security. IEEE Communications Surveys & Tutorials, 21(1), 640-660.<https://doi.org/10.1109/COMST.2018.2871866>

Mavroeidis, V., & Bromander, S. (2017). Cyber threat intelligence model: An evaluation of taxonomies, sharing standards, and ontologies within cyber threat intelligence. 2017 European Intelligence and Security Informatics Conference (EISIC), 91-98. <https://doi.org/10.1109/EISIC.2017.20>

Samtani, S., Chinn, R., Chen, H., & Nunamaker Jr, J. F. (2017). Exploring emerging hacker assets and key hackers for proactive cyber threat intelligence. Journal of Management Information Systems, 34(4), 1023-1053. <https://doi.org/10.1080/07421222.2017.1394053>

Sauerwein, C., Sillaber, C., Mussmann, A., & Breu, R. (2017). Threat intelligence sharing platforms: An exploratory study of software vendors and research perspectives. 13th International Conference on Wirtschaftsinformatik, 837-851.

Sauerwein, C., Sillaber, C., Mussmann, A., & Breu, R. (2017). Threat Intelligence Sharing Platforms: An Exploratory Study of Software Vendors and Research Perspectives. 13. Internationale Tagung Wirtschaftsinformatik (WI 2017), St. Gallen, Switzerland.

Top 10 Threat Intelligence Platforms in 2022 <https://www.spiceworks.com/it-security/vulnerability-management/articles/best-threat-intelligence-platforms>

Tounsi, W., & Rais, H. (2018). A survey on technical threat intelligence in the age of sophisticated cyber attacks. Computers & Security, 72, 212-233. <https://doi.org/10.1016/j.cose.2017.09.001>

Tounsi, W., & Rais, H. (2018). A survey on technical threat intelligence in the age of sophisticated cyber attacks. Computers & Security, 72, 212-233. <https://doi.org/10.1016/j.cose.2017.09.001>

[What Are Research Objectives and How to Write Them (with Examples) | Researcher.Life](https://researcher.life/blog/article/what-are-research-objectives-how-to-write-them-with-examples/#:~:text=General%20objectives%20are%20the%20main%20goals%20of%20the,listed%20in%20the%20New%20York%20Stock%20Exchange%20market.) [https://researcher.life/blog/article/what-are-research-objectives-how-to-write-them-with-examples/ - :~:text=General%20objectives%20are%20the%20main%20goals%20of%20the,listed%20in%20the%20New%20York%20Stock%20Exchange%20market.](https://researcher.life/blog/article/what-are-research-objectives-how-to-write-them-with-examples/#:~:text=General%20objectives%20are%20the%20main%20goals%20of%20the,listed%20in%20the%20New%20York%20Stock%20Exchange%20market.)

What is the justification of a research?[https://www.editage.com/insights/what-is-the-justification-of-a-research- :~:text=The%20justification%20of%20a%20research%20is%20also%20known,may%20then%20wish%20to%20explore%20through%20your%20research.](https://www.editage.com/insights/what-is-the-justification-of-a-research-%20:~:text=The%20justification%20of%20a%20research%20is%20also%20known,may%20then%20wish%20to%20explore%20through%20your%20research.)

Yu, Y., Wang, H., & Xu, L. (2023). A Survey on Threat Intelligence Feeds: Characteristics, Applications, and Challenges. IEEE Access, 11, 1136-1150. [<https://ieeexplore.ieee.org/>