A Framework for reviewing security related requirements in requirements specification of Android Application Development

By

ARSALAN ALI

MSCSE 19

(Registration No.:319528)

Supervisor: Dr. Taimoor Zahid

DEPARTMENT OF COMPUTER AND SOFTWARE ENGINEERING,

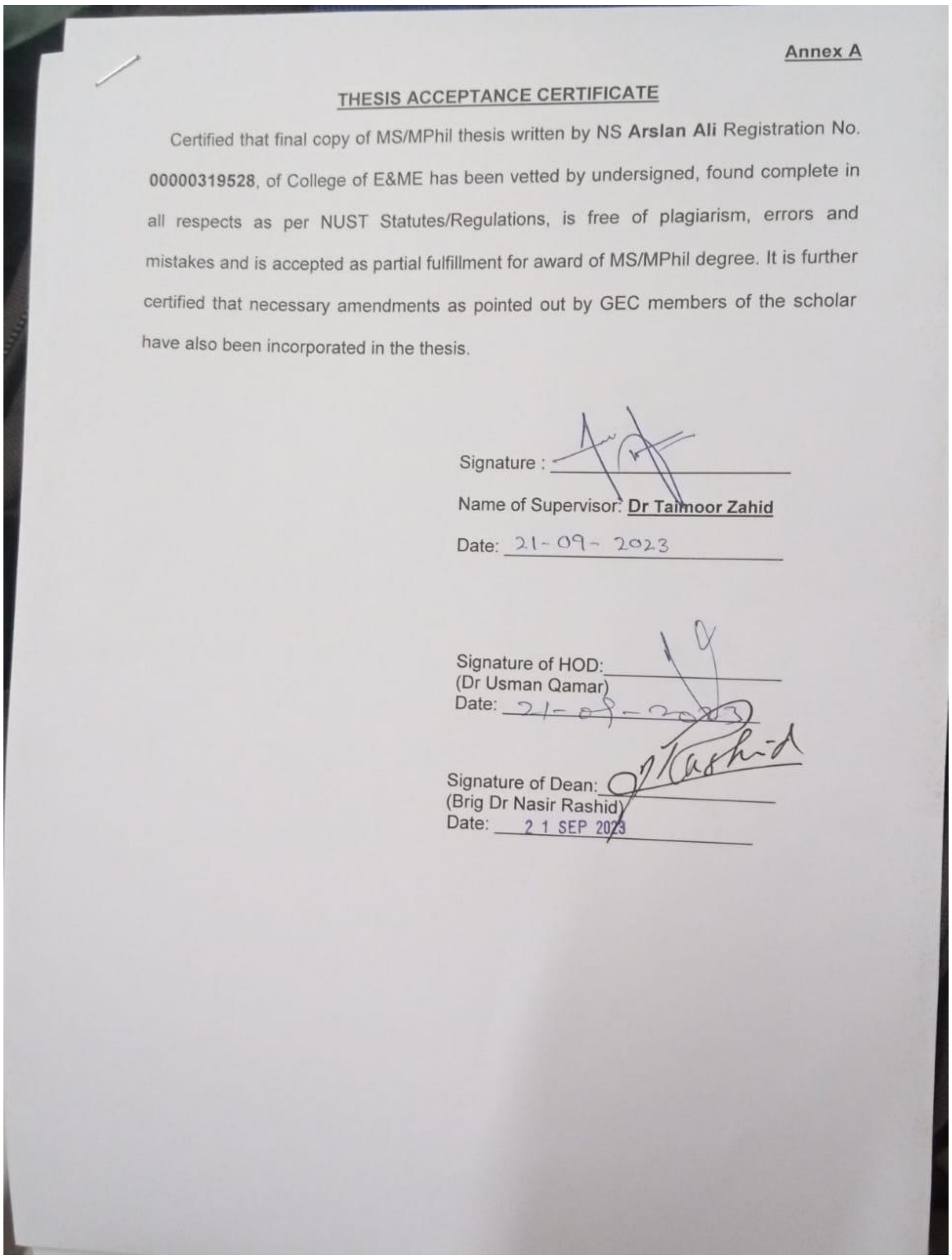
COLLEGE OF ELECTRICAL AND MECHANICAL ENGINEERING (E&ME),

NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY (NUST),

ISLAMABAD

SEP, 2023

Thesis Acceptance Certificate



Annex A

THESIS ACCEPTANCE CERTIFICATE

Certified that final copy of MS/MPhil thesis written by NS **Arslan Ali** Registration No. **00000319528**, of College of E&ME has been vetted by undersigned, found complete in all respects as per NUST Statutes/Regulations, is free of plagiarism, errors and mistakes and is accepted as partial fulfillment for award of MS/MPhil degree. It is further certified that necessary amendments as pointed out by GEC members of the scholar have also been incorporated in the thesis.

Signature : [illegible]

Name of Supervisor: **Dr Taimoor Zahid**

Date: 21-09-2023

Signature of HOD: [illegible]
(Dr Usman Qamar)
Date: 21-09-2023

Signature of Dean: [illegible]
(Brig Dr Nasir Rashid)
Date: 2 1 SEP 2023

Declaration

Dedicated to a hopeful couple, my mother and my father, and my Teachers whose motivations, and endless prayers led me to this achievement

Acknowledgments

I am thankful to ALLAH Almighty for his blessings throughout this research work. It was quite a challenging process that could not have been completed without the help of Allah Almighty and the strength that he given to me.

I would like to thank my sincere supervisor **‘Dr. Taimoor Zahid’** and co-supervisor **‘Dr. Wasi Haider Butt’** for their determined guidance and the entire faculty staff for their endless support. I cannot thank them enough for their role in the completion of my thesis report. I also thank my parents, siblings, and friends who encouraged me and kept me motivated during my master’s program.

I am also eternally grateful to the Department of Computer and Software Engineering and the management of College of Electrical and Mechanical Engineering, NUST, who helped me and supported me throughout this journey.

Abstract

In this technology-driven era, the demand for software application development, particularly on the Android platform, is soaring. However, the rapid and agile nature of development often leads to insufficient specification of security-related requirements, resulting in significant security risks. Neglecting these crucial elements can have severe consequences for software applications. This paper presents a systematic literature review of state-of-the-art requirements specification methods and frameworks from 97 research articles, with a specific focus on their treatment of security-related requirements. The aim is to gain insights into existing practices and identify potential gaps in addressing security concerns during the early stages of development. The study reveals that overlooking security-related elements in the early stages of development exposes organizations to major security risks. Unauthorized access becomes a critical concern, leaving sensitive data vulnerable to breaches. Inadequate data protection measures, such as weak encryption or improper data storage, increase the risk of data compromises, leading to reputational damage and potential legal repercussions. Moreover, when security requirements fail to address safeguards against privileged insiders abusing their access, insider threats become a significant concern. Additionally, lacking incident response planning hinders effective detection and mitigation of security incidents, resulting in extended downtime and increased damage. To address these risks and enhance the security of Android applications, this paper proposes a novel framework that leverages natural language processing (NLP) techniques in conjunction with the Naive Bayes model. The framework aims to extract and prioritize security-related requirements from raw requirement documents effectively. The Naive Bayes model is well-suited for this task due to its simplicity, efficiency, and ability to handle large volumes of textual data. The model leverages probabilistic principles to classify requirements as security-related or non-security-related based on the likelihood of occurrence of specific security-related terms and patterns in the text. By incorporating the Naive Bayes model within the proposed framework, security analysts can efficiently analyse and categorize requirements, ensuring that security-related elements are adequately addressed from the outset of the development process. Applying the proposed framework early in the development lifecycle empowers organizations to streamline the development process and mitigate potential security breaches and associated costs. By integrating security requirements seamlessly into the development process, teams can identify and address security concerns proactively, reducing the likelihood of vulnerabilities and ensuring robust protection of sensitive data. In conclusion, this research highlights the criticality of considering security-related requirements during Android application development. The proposed framework, powered by the Naive Bayes model, presents a promising solution to tackle the challenges of security specification in an agile development environment. By bridging the gap between security concerns and development activities, the framework enables organizations to develop secure and reliable Android applications, safeguarding both user data and the organization's reputation.

Keywords – *Requirements Specification, Requirement Elicitation, Security Requirements, Non-functional Requirements, Security Requirements identification, Tool Support for Security related requirements specification, Security Requirements in mobile App Development*

Table of Contents

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

In today's digital era, Android applications have become an integral part of our daily lives, revolutionizing the way we communicate, work, and access information. Android, being the most widely used mobile operating system worldwide, has a significant impact on the global app ecosystem. As of 2023, there were over 3.48 million Android applications available on the Google Play Store, catering to the diverse needs of users worldwide [1] The extensive range of Android apps reflects their popularity and the growing demand for convenient, on-the-go solutions. The exponential growth of Android app usage can be observed in the increasing number of app downloads. According to Statista, the number of mobile app downloads worldwide is projected to reach 300 billion in 2023, up from 255 billion in 2022 [2]. This upward trend is illustrated in the graph below, showcasing the rise in global app downloads over the years.

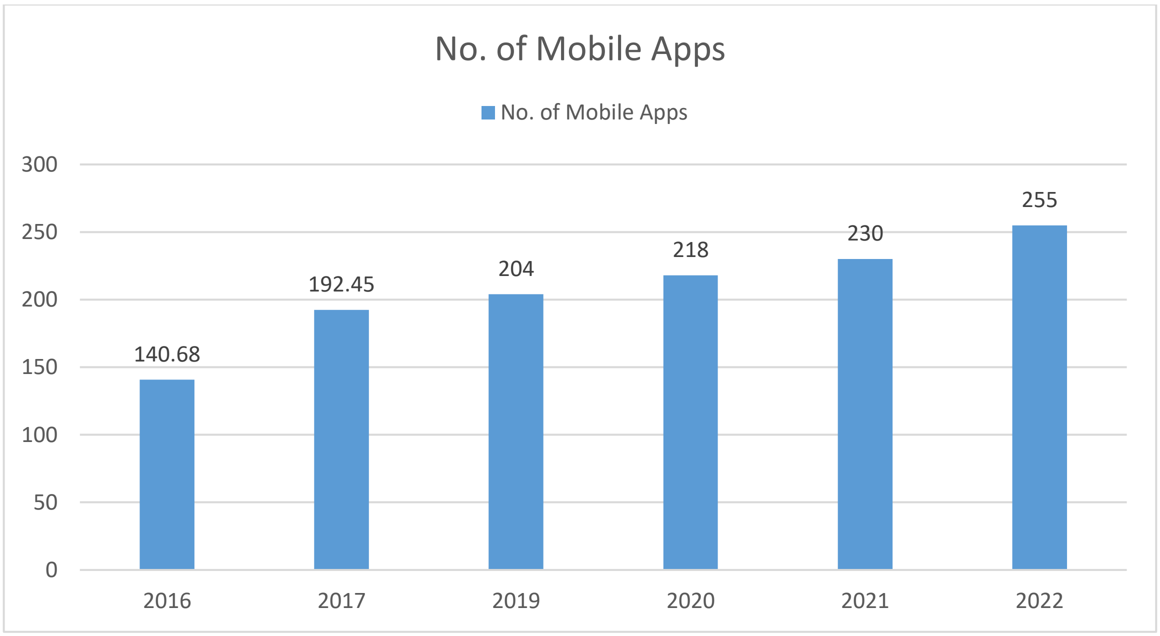


Figure 1: Number of mobile app downloads worldwide from 2016-2022

The significance of Android applications transcends personal use, with their impact extending to various sectors such as e-commerce, entertainment, education, healthcare, and more. These applications have transformed the way businesses operate and interact with customers, leading to enhanced productivity, streamlined processes, and improved user experiences. Furthermore, they have opened up new avenues for entrepreneurs and developers to innovate and bring their ideas to a global audience. The pervasive use of smartphones and Android applications is evident in a survey conducted by Pew Research Centre, which found that around 95% of the world's population owns a cell phone, and 80% own a smartphone [3]. Moreover, 53% of smartphone users reported using their devices to access online services and applications [3]. These statistics highlight the significant role Android applications play in facilitating seamless connectivity and enabling users to access a wide range of services and information conveniently. The immense popularity of Android applications can be attributed to their userfriendly interfaces, extensive app offerings, and constant advancements in technology. As smartphones become increasingly powerful and interconnected, Android applications continue to evolve, providing users with innovative features and personalized experiences. The continuous growth of the Android app ecosystem demonstrates the unwavering demand for convenient, reliable, and efficient mobile solutions. In short, Android applications have become a vital component of our modern lifestyle, offering countless benefits and transforming the way we interact with technology. The sheer number of available Android applications, coupled with the increasing app downloads and the widespread use of smartphones, signifies their indispensability in our daily lives. As technology continues to advance, the significance of Android applications is likely to grow, driving innovation, enhancing user experiences, and shaping the future of mobile technology.

However, the importance of security in Android applications cannot be overstated, as it plays a crucial role in protecting user data, maintaining privacy, and preventing potential cyber threats. Incidents involving security breaches in Android applications have highlighted the dire consequences of overlooking security requirements. One prominent example is the "Stagefright" vulnerability discovered in 2015, which affected millions of Android devices worldwide. This vulnerability allowed attackers to remotely execute malicious code through a multimedia message, potentially compromising sensitive user information and raising significant concerns about the overall security of the Android ecosystem [4]. Another notable example is the "Joker" malware that targeted Android applications. In 2020, researchers discovered multiple instances of malicious apps on the Google Play Store infected with the Joker malware. This malware had the capability to silently subscribe users to premium subscription services without their consent, resulting in financial losses and compromised privacy for unsuspecting users [5]. Such incidents serve as a stark reminder of the potential risks and damages that can arise from inadequate security measures in Android applications. The widespread distribution of malware-infected apps through the Google Play Store is another alarming incident that underscores the criticality of robust security measures. Such incidents have led to substantial financial losses and reputational damage for both users and developers, as unsuspecting users unknowingly downloaded and installed compromised applications [6]. Furthermore, the increasing use of Android applications for various activities, such as online banking, e-commerce transactions, and accessing personal information, amplifies the need for stringent security measures. A lack of proper security protocols exposes users to risks such as data breaches, identity theft, and unauthorized access to sensitive information. These incidents can have severe consequences, including financial losses, damage to personal and professional reputations, and even potential legal ramifications. The impact of security breaches is not limited to individual users; it also affects businesses and organizations that rely on secure communication and data handling.

To address these security concerns, developers must adhere to industry best practices and employ robust security measures throughout the development lifecycle of Android applications. This includes implementing secure coding practices, utilizing encryption for data transmission and storage, implementing user authentication mechanisms, and conducting regular security assessments and audits. Additionally, staying updated with the latest security patches and fixes provided by Android's security updates is crucial to address known vulnerabilities and protect against emerging threats. The significance of security in Android applications is also acknowledged by regulatory bodies and standards organizations. For instance, the Payment Card Industry Data Security Standard (PCI DSS) sets specific security requirements for applications handling credit card information, emphasizing the need for secure coding practices, encryption, and vulnerability management. Compliance with such standards not only ensures the security of user data but also helps establish trust and confidence among users and stakeholders. Having said that it should be noted that the importance of security in Android applications cannot be ignored in today's digital landscape. The incidents of security breaches and vulnerabilities in Android applications highlight the urgent need for robust security measures to safeguard user information, prevent unauthorized access, and protect against potential cyber threats. By prioritizing security throughout the development process and adhering to industry best practices and standards, developers can provide users with a secure and trustworthy experience while mitigating risks and safeguarding sensitive data. Mobile Security is a serious requirement and needs consideration as a vital part of software development. However, developing secure application is not a trivial task as it requires to address security from early development stages throughout the whole software lifecycle. Yet, in the majority of software projects, security is often dealt with in retrospect when the system has already been designed and put into operation [7]. Traditional, document-intensive requirement engineering includes practices to elicit, analyse, specify, and validate and verify requirements. In such practices, the communication of requirements is mostly based on formal and extensive documentation. Although the usage of such traditional approaches is widespread, especially in the software industries dealing with security and safety-critical systems, the necessity of delivering projects in shorter periods has triggered the desire and need to change towards gradually increasing the usage of other methods such as hybrid models and agile methodology. As the number of mobile applications continues to increase, so does the importance of ensuring their security. One critical step in the development process is the review of security-related requirements in the requirement specification [7]. Moreover, Android Application Development has different constraints as most often conventional methodologies cannot be applied to it because of its rapid development period. The widespread use of smartphones has led to an increase in the demand for mobile applications. As more and more apps are developed, it becomes essential to ensure that these apps are secure. Unfortunately, security is often overlooked in the app development process, which leaves users vulnerable to a range of security risks.

This research aims to identify state-of-the-art technologies and frameworks that researchers have developed over time to specify security-related requirements in application development. The Systematic Literature Review begins with the identification of such frameworks and methodologies that were being used in Web, Desktop, and even Mobile Application Development. By analysing the current landscape of security requirement specification methods, this research aims to contribute to the development of more secure Android applications. The ultimate goal is to ensure that developers prioritize security from the outset, effectively mitigating risks and safeguarding user data. With the continued growth and impact of Android applications, this research is timely and relevant for creating a safer digital environment for users worldwide.

1.1 MOTIVATION

The demand for Android application development has reached unprecedented heights. The vast array of Android apps available on the global stage has revolutionized the way we communicate, work, and access information. However, the rapid and agile nature of development often leads to the oversight of crucial security-related requirements, exposing software applications to significant security risks. Neglecting these elements can result in dire consequences, such as unauthorized access, data breaches, and potential harm to an organization's reputation. To address this critical issue, this research is driven by the need to thoroughly investigate the treatment of security-related requirements during the early stages of Android app development. Through a systematic literature review of cutting-edge requirements specification methods and frameworks, this study aims to identify existing practices and potential gaps. The ultimate objective is to propose a novel framework that harnesses the power of natural language processing (NLP) techniques and the Naive Bayes model to effectively extract and prioritize security-related requirements from raw documents. By integrating robust security measures early in the development process, this research endeavours to empower organizations to build secure and reliable Android applications, thereby safeguarding user data and fostering trust in the rapidly changing digital landscape.

1.2 PROBLEM STATEMENT

The absence of an efficient and systematic framework for identifying and prioritizing security-related requirements early in the development process hinders organizations from proactively addressing security concerns and mitigating potential security breaches and associated costs specially in mobile application development. The objective is to empower software development teams to enhance the security and reliability of Android applications while streamlining the development lifecycle.

1.3 AIMS AND OBJECTIVES

The primary aim of this research is to investigate and address the challenges posed by the insufficient specification of security-related requirements in Android application development. By conducting a comprehensive analysis of existing practices and proposing a novel framework, the study aims to enhance the security of Android applications, mitigating potential security risks and ensuring the protection of user data.

The objectives of this research activity are provided below:

1. To conduct a systematic literature review, analysing the state-of-the-art requirements specification methods and frameworks in Android application development, with a specific focus on their treatment of security-related requirements.
2. To identify the common security risks associated with the lack of proper security specification in the early stages of Android app development, including unauthorized access, data breaches, and potential regulatory non-compliance.
3. To propose a novel framework that leverages natural language processing (NLP) techniques and the Naive Bayes model for effectively extracting and prioritizing security-related requirements from raw requirement documents.
4. To validate the proposed framework's effectiveness in identifying security-related requirements through empirical analysis
5. To provide recommendations and best practices for developers and organizations to integrate security-related requirements proactively into their Android application development process, aiming for robust and reliable software solutions.

1.4 THESIS OUTLINE

The rest of the thesis outlines as follow:

Chapter 2 consists of the Systematic Literature Review in detail and various key state-of-the-art related work conducted by various researchers in previous years. Further in **Chapter 3** the proposed NLP based Approach with respect to Android has been discussed. Moreover, **Chapter 4** includes the Implementation and discussion on results. Whereas, Research Thesis concludes with **Chapter 5** that includes a summary of conducted work and some future directions.

CHAPTER 2

SYSTEMATIC LITERATURE REVIEW

In Chapter 2 Systematic Literature Review (SLR) has been conducted for the mentioned area of research. The guidelines of [8] has been followed in performing the literature review. The phase wise flow of the literature review has been shown in the **Figure 2** below.

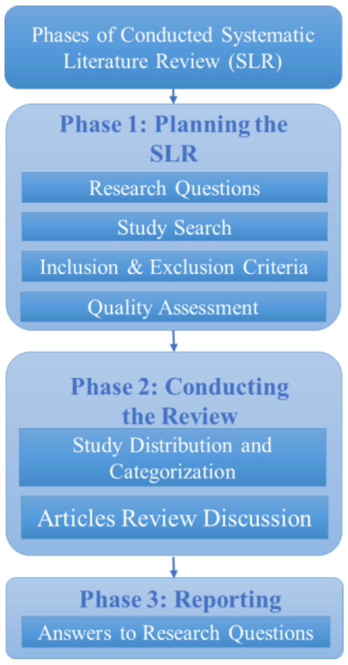

Phases of Conducted Systematic Literature Review (SLR)
Phase 1: Planning the SLR
Research Questions
Study Search
Inclusion & Exclusion Criteria
Quality Assessment
Phase 2: Conducting the Review
Study Distribution and Categorization
Articles Review Discussion
Phase 3: Reporting
Answers to Research Questions

Figure 2: Phase of Systematic Literature Review

The initial step of the Systematic Literature Review (SLR) involved the planning phase, which encompassed several key activities. Firstly, the process involved crafting research questions to guide the review. Additionally, during the study search, a collection of research databases was chosen. This selection process included creating search strings and focusing on studies published between 2010 and 2023. Furthermore, the first phase included establishing criteria for including or excluding studies. This step helped ensure the relevance of the selected studies. Additionally, a quality assessment was performed on the chosen studies to gauge their reliability. Moving on to the second phase, which involved the actual review process, the selected articles were sorted into different categories using various filters. These filters included organizing the articles based on their distribution across different corpora and years. Moreover, the chosen articles were classified according to the frameworks, methodologies, and approaches they employed. These approaches were subsequently discussed in-depth during the discussion phase. Finally, the third stage centred on drawing conclusions based on the conducted review. The research questions formulated during the initial phase of the SLR were addressed, and answers were provided based on the findings of the review process.

2.1 Research Questions

Based on the research area the formulated research questions are provide below:

RQ1: What are the existing Security Requirements Identification Methodologies and Tools Techniques?

RQ2: How effective are the different methodologies for reviewing security-related requirements in software requirement specifications?

RQ3: What are the available Tools and techniques or Framework for specification of security related Requirements in mobile application development?

RQ4: What are the key challenges and limitations associated with the current methodologies used for reviewing security-related requirements in software requirement specifications?

2.2 Study Selection

After formulating research questions, the next portion in this phase is the process to select the studies from various Research Directories that fulfil the needs of the area of research under consideration.

2.2.1 Study Search

To explore the current advancements and frameworks in technology, an extensive literature search was conducted across various databases *including IEEE, ACM, SPRINGER, TAYLOR and FRANCIS, ELSEVIER*, among others. The goal was to locate relevant articles. Different keywords and search phrases were employed to optimize the search results. Initially, the search began with a simple query such as "Security Related Requirements Specification." However, this generated a large number of results, which proved impractical to include entirely. To refine the outcomes, specific filters were applied. For instance, the publication years were confined to fall *between 2010 and 2023*. Additionally, operators like *"AND"* and *"OR"* were incorporated into the search strings, as illustrated in **Table 1**. Furthermore, techniques were utilized to enhance the search process, including the use of synonyms and phrase substitutions.

Table 1: Search keywords and combinations

No	Search Keywords	Alternatives
1	Security Requirements (SK1)	“security requirements” OR “security related requirement” OR “safety requirements”
2	Specification (SK2)	“specification” OR “elicitation” OR “gathering” OR “identification”
3	Methodology (SK3)	“methodology” OR “framework” OR “technique” OR “approach” OR “tool”
4	Software Development (SK4)	“software development” OR “web application development” OR “desktop application development” OR “mobile application development”

As a result of this iterative process, a final search phrase was formulated: ***(SK1) AND (SK2) AND (SK3) AND (SK4)***. This refined phrase aimed to capture the most relevant and pertinent articles for the review. Furthermore, an additional search method known as Snowballing [9] was employed to enhance the search results, aiming to gather even more valuable information for the intended Systematic Literature Review (SLR). After implementing all these refining filters in the search process, a total of 97 articles were carefully chosen. These selected articles are expected to provide valuable insights and contribute to addressing the research questions formulated earlier. Moreover, **Figure 3** illustrates the application of the Tollgate Approach [8], a method employed to refine the pool of selected studies. Initially, a total of 936 research articles were collected from various research databases, including prominent sources such as IEEE, SPRINGER, and ACM, as well as others like Elsevier, Taylor and Francis, among others. The initial phase involved evaluating the relevancy of these studies based on their titles, aligning with the predefined selection criteria.

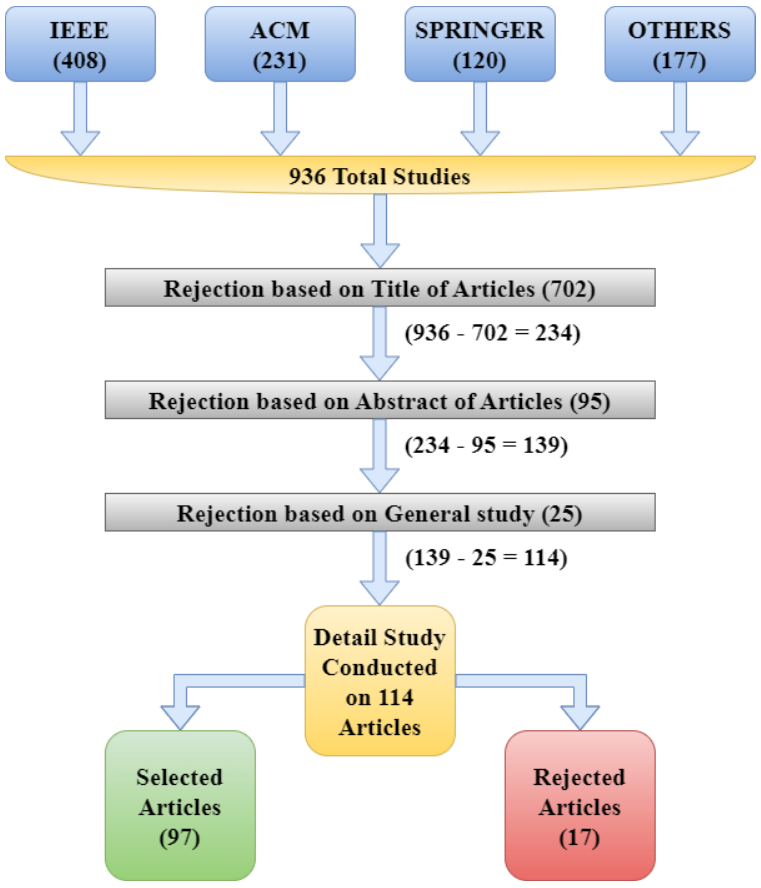

IEEE (408)
ACM (231)
SPRINGER (120)
OTHERS (177)
936 Total Studies
Rejection based on Title of Articles (702)
(936 - 702 = 234)
Rejection based on Abstract of Articles (95)
(234 - 95 = 139)
Rejection based on General study (25)
(139 - 25 = 114)
Detail Study Conducted on 114 Articles
Selected Articles (97)
Rejected Articles (17)

Figure 3: Tollgate Approach for Article Searching

Subsequently, the abstracts of the remaining articles were carefully examined. This process revealed that 95 research papers did not meet the inclusion criteria and were thus excluded. Furthermore, a more comprehensive review was conducted on the papers that remained after the initial filtering. These papers were subjected to a thorough assessment through skimming, with the aim of validating their suitability for inclusion. As the refinement process continued, a detailed examination was carried out on the remaining research articles. This meticulous analysis led to the identification of a final set of 97 research articles that demonstrated strong relevance and appeared promising in terms of offering insights to address the research questions formulated earlier.

2.2.2 Inclusion and Exclusion Criteria

Incorporation and disintegration are primarily composed of a defined set of criteria and rules that serve as the foundation for determining the inclusion or exclusion of specific research articles. These criteria encompass several phases through which research papers are evaluated to ascertain their eligibility for inclusion. Only those articles that align with these inclusion and exclusion criteria proceed for further investigation [9]. The selection criteria encompass various aspects, including the ***subject, publisher, publication year, and language.*** The process begins by identifying papers that are directly pertinent to the domain of security requirements specification. Specifically, research articles addressing security-related requirements are earmarked for more thorough examination. Studies that fall outside the scope of subject relevance are excluded from consideration. The second facet of the criteria focuses on sourcing studies from reputable scientific repositories like **IEEE, ACM, Springer,** and **other** respected sources such as **Elsevier, Taylor & Francis,** and **Semantic Scholar** etc. These repositories are recognized for their credibility, ensuring that the selected articles are of high quality and well-crafted. The third criterion considered in this Systematic Literature Review pertains to the publication year. Articles published within the time frame of **2010** to **2023** are included, while those published before 2010 are disregarded. Lastly, language is integrated into the inclusion and exclusion criteria. Research studies written exclusively in **English** are incorporated into this SLR. Conversely, articles composed in languages other than English, even if related to the subject, are excluded from consideration.

2.2.3 Quality Assessment

In this Systematic Literature Review (SLR), a deliberate effort was made to source research studies from reputable and influential sources. The chosen papers were carefully curated to ensure the credibility and robustness of the findings in the SLR. The search for research articles was conducted across esteemed repositories including IEEE, ACM, and Springer, etc. The ultimate quality assessment of the selected studies was conducted using a specifically designed checklist and criteria [8]. The quality assessment checklist, presented in **Table 2**, was devised to systematically evaluate the calibre of the research articles. This checklist comprises five distinct questions, as outlined in **Table 2**.

Table 2: Quality Assessment Checklist

Quality Assessment Question No.	Quality Assessment Questions Checklist
QA1	Does the picked study answer the research questions?
QA2	Does the selected study discuss security requirement specification?
QA3	Does the selected study propose any tool or framework for security requirements elicitation?
QA4	Does the study apply proper case study?

QA5	Can the proposed tool or methodology be applied in all domain of software development such as mobile, web and desktop?

Each question on the checklist has been associated with quality evaluation criteria, as outlined in **Table 3** The assessment of the selected articles' quality is carried out utilizing the prescribed checklist and criteria, adhering to the guidelines established in the referenced study [8].

Table 3: Quality Assessment Criteria

Quality Assessment Score No	Quality Assessment Criteria
QAS1	The studies were graded "1" that completely fulfilled Quality Assessment checklist.
QAS2	The studies were graded "0.5" that partially fulfilled Quality Assessment checklist.
QAS3	The studies were graded "0" that not fulfilled Quality Assessment checklist.

A comprehensive compilation of the assessed research studies is provided in **Table 4**. Where each research articles have been evaluated based on the above-mentioned checklist and criteria.

Table 4: Evaluation of Articles

Research No.	Paper Reference	QA1	QA2	QA3	QA4	QA5	TOTAL	PRECENTAGE (N=>5)
1	[7]	1	1	0.5	0.5	0.5	3.5	70
2	[10]	1	1	1	1	0	4	80
3	[11]	1	1	1	0.5	0	3.5	70
4	[12]	1	1	0.5	0.5	0	3	60
5	[13]	1	1	1	0.5	0.5	4	80
6	[14]	1	1	1	0	0	3	60
7	[15]	1	0.5	0.5	0.5	0	2.5	50
8	[16]	1	1	1	0.5	1	4.5	90
9	[17]	1	1	1	0	1	4	80
10	[18]	1	1	1	0	1	4	80
11	[19]	1	0.5	0.5	0.5	1	3.5	70
12	[20]	1	0.5	0.5	0.5	0.5	3	60
13	[21]	1	1	0.5	1	1	4.5	90
14	[22]	1	0.5	0.5	1	1	4	80
15	[23]	1	1	0.5	0.5	0	3	60
16	[24]	1	0.5	0	0	1	2.5	50
17	[25]	1	0.5	0.5	0.5	1	3.5	70
18	[26]	1	1	0.5	0	0	2.5	50
19	[27]	1	0.5	0.5	0.5	0.5	3	60
20	[28]	1	0.5	1	0.5	1	4	80
21	[29]	1	1	1	0.5	0	3	60
22	[30]	1	0.5	1	1	0	3.5	70
23	[31]	1	1	1	1	0	4	80
24	[32]	1	0.5	0.5	0	1	3	60
25	[33]	1	1	1	0.5	1	4.5	90
26	[34]	1	0.5	0.5	0.5	0	2.5	50

27	[35]	1	1	0.5	0.5	0	3	60
28	[36]	1	1	1	1	0	4	90
29	[37]	1	0.5	0.5	0.5	0	2.5	50
30	[38]	1	0.5	0.5	0.5	0	2.5	50
31	[39]	1	0.5	1	1	0	3.5	70
32	[40]	1	1	1	1	0	4	80
33	[41]	1	0.5	0.5	0.5	0	2.5	50
34	[42]	1	1	1	1	0.5	4.5	90
35	[43]	1	0.5	0.5	0.5	0	2.5	50
36	[44]	1	0.5	1	0.5	0	3	60
37	[45]	1	1	1	1	0	4	80
38	[46]	1	1	0.5	0	0	2.5	50
39	[47]	1	0.5	0.5	0.5	0	2.5	50
40	[48]	1	1	0.5	0.5	0	3	60
41	[49]	1	1	1	1	0	4	80
42	[50]	1	1	0.5	0.5	0	3	60
43	[51]	1	1	0.5	0	0	2.5	50
44	[52]	1	1	1	0.5	1	4.5	90
45	[53]	1	1	1	0.5	1	4.5	90
46	[54]	1	1	0.5	0.5	1	4	80
47	[55]	1	0.5	1	0.5	0	3	60
48	[56]	1	1	1	1	0	4	80
49	[57]	1	0.5	1	0.5	0	3	60
50	[58]	1	1	1	1	0	4	80
51	[59]	1	0.5	1	0.5	0	3	60
52	[60]	1	0.5	0.5	0.5	0	2.5	50
53	[61]	1	0.5	0.5	0	0.5	2.5	50
54	[62]	1	0.5	0.5	0.5	1	3.5	70
55	[63]	1	1	1	0	1	4	80
56	[64]	1	1	1	0.5	1	4.5	90
57	[65]	1	1	0.5	0.5	1	4	80
58	[66]	1	0.5	0.5	0.5	0.5	3	60
59	[67]	1	1	1	1	0	4	80
60	[68]	1	0.5	0.5	0.5	0	2.5	50
61	[69]	1	1	0.5	0	0	2.5	50
62	[70]	1	1	0.5	0	0	2.5	50
63	[71]	1	1	0.5	0	0	2.5	50
64	[72]	1	1	1	0	0.5	3.5	70
65	[73]	1	1	1	0.5	0	3.5	70
66	[74]	1	1	0.5	0	1	3.5	70
67	[75]	1	1	1	0	0	3	60
68	[76]	1	0.5	0.5	0	1	3	60
69	[77]	1	1	1	0	1	4	80
70	[78]	1	1	0.5	0.5	0	3	60
71	[79]	1	1	1	0	0.5	3.5	70
72	[80]	1	1	1	1	1	5	100
73	[81]	1	1	1	0	1	4	80
74	[82]	1	1	0.5	0	0.5	3	60
75	[83]	1	1	1	0.5	0	3.5	70
76	[84]	1	1	1	0	0	3	60
77	[85]	1	1	0.5	0.5	0	3	60
78	[86]	1	0.5	0.5	0.5	0	2.5	50
79	[87]	1	1	0.5	0	0	2.5	50
80	[88]	1	1	1	0.5	1	4.5	90
81	[89]	1	1	1	0.5	0	3.5	70

82	[90]	1	1	1	1	1	5	100
83	[91]	1	1	0.5	0.5	0	3	60
84	[92]	1	1	1	1	1	5	100
85	[93]	1	1	1	0.5	1	4.5	90
86	[94]	1	0.5	1	0.5	1	4	80
87	[95]	1	1	1	0.5	0	3.5	70
88	[96]	1	1	0.5	0.5	0	3	60
89	[97]	1	1	0.5	0	0.5	3	60
90	[98]	1	1	0	0	0.5	2.5	50
91	[99]	1	1	1	1	0	4	80
92	[100]	1	1	1	1	1	5	100
93	[101]	1	1	1	0.5	1	4.5	90
94	[102]	1	1	0	0.5	1	3.5	70
95	[103]	1	1	0	0.5	1	3.5	70
96	[104]	1	1	1	1	0	4	80
97	[105]	1	1	1	1	0	4	80

2.2.4 Study Distribution and Categorisation

Presented in **Table 5** is an overview of the distribution of pertinent research articles across various databases. The table additionally outlines the categorization of each research study according to its type and the respective research database it originates from.

Table 5: Research Articles Distribution

Research Databases	Type	Selected Research Articles	No. of Research
IEEE	C	[10] [11] [12] [13] [14] [15] [16] [17] [18] [19] [20] [21] [22] [23] [24] [25] [26] [27] [28] [29] [30] [31] [32] [33] [34] [35] [36] [37] [38] [39] [40] [41] [42] [7] [43] [44] [45] [46] [47] [48] [49] [50]	42
	J	[51] [52] [53] [54] [55] [56] [57] [58] [59]	9
ACM	C	[60] [61] [62] [63] [64] [65] [66]	7
	J	[67] [68] [69] [70] [71] [72] [73]	7
Springer	C	[74] [75] [76] [77]	4
	J	[78] [79] [80] [81] [82] [83] [84]	7
Others	C	[85] [86] [87]	3
	J	[88] [89] [90] [91] [92] [93] [94] [95] [96] [97] [98] [99] [100] [101] [102] [103] [104] [105]	18

The research articles were further categorized into two distinct types: Conference papers (C) and Journal papers (J), as illustrated in **Figure 4**. Among the total of 97 chosen research articles, 56 were identified as Conference papers, while 41 were categorized as Journal articles.

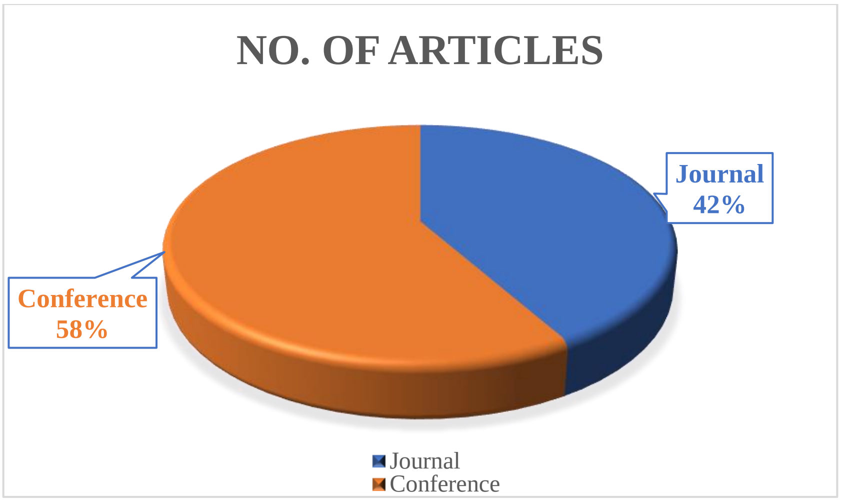


Figure 4: Research Studies Categorisation

Furthermore, the graphical representation in **Figure 5** illustrates the distribution of research articles from each individual database. Notably, a substantial portion of the selected articles originated from **IEEE**, comprising *42 Conference papers* and *9 Journal articles*. **ACM** the subsequent prominent contributor was the second database, yielding *7 Conference papers* and *7 Journal articles*. Similarly, the **Springer** database yielded *4 Conference papers* and *7 Journal articles* relevant to the subject matter. In the ***"Others"*** category, diverse repositories like Elsevier, Taylor and Francis, among others, were explored. These repositories collectively contributed *3 Conference papers* and *18 Journal articles* that aligned with the research topic.

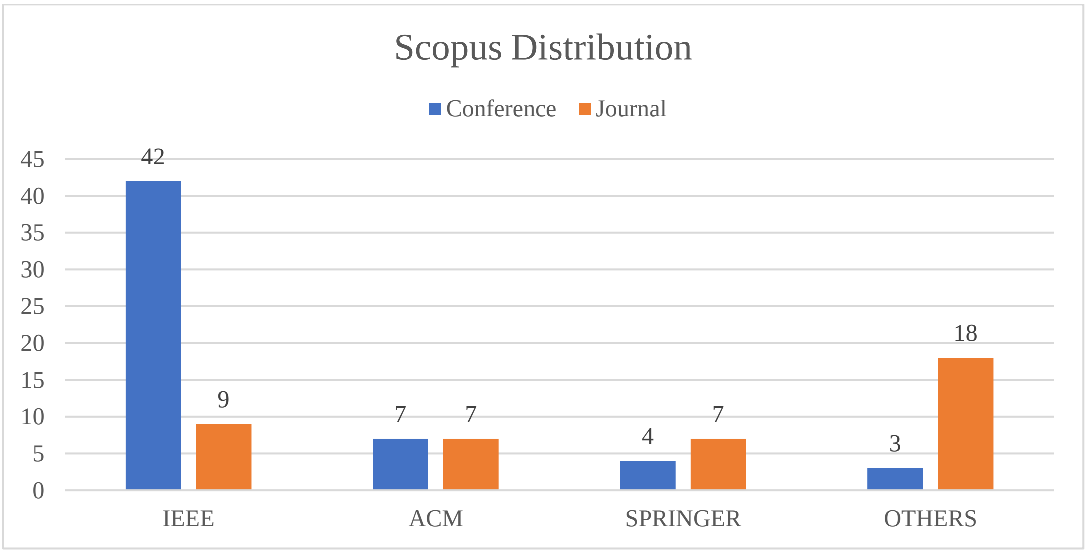


Figure 5: Scopus Distribution

Furthermore, **Table 6** furnishes data regarding the temporal distribution of selected research studies based on the year of publication. Notably, the highest number of selected papers, totalling 13, originated from the year 2019. Similarly, the years 2021, 2018, and 2014 each contributed 11 papers. In contrast, the years 2023, 2022, and 2013 accounted for 3 papers each, while the year 2011 had the lowest representation with only 2 papers.

Table 6: Temporal Distribution of Research Papers

Year	Number of Papers	Research Papers	Percentage %
2010	3	[29], [83], [85]	3.09
2011	2	[71], [74]	2.06
2012	8	[11], [12], [23], [50], [61], [76], [88], [101]	8.24
2013	3	[57], [84], [97]	3.09
2014	11	[13], [14], [24], [31], [34], [49], [51], [56], [62], [96], [100]	11.34
2015	7	[15], [19], [26], [29], [60], [86], [103]	7.21
2016	10	[22], [32], [46], [47], [48], [77], [81], [82] [87], [102]	10.30
2017	7	[16], [21], [45], [75], [79], [80], [98]	7.21
2018	12	[7], [17], [18], [20], [33], [36], [44], [52], [53], [66], [78], [91]	12.37
2019	12	[16], [27], [30], [35], [40], [41], [42], [43], [63], [64], [65], [69],	12.37
2020	7	[37], [38], [39], [55], [72], [90], [99]	7.21
2021	10	[25], [54], [58], [59], [68], [89], [92], [93], [94] [104]	10.30
2022	3	[73], [81], [95]	3.09
2023	3	[67], [70], [105]	3.09

On the other hand, **Table 7** presented below offers a detailed classification of research articles based on Security Requirements Elicitation techniques and methodologies. This categorization has been specifically developed within this research project following an extensive review of the literature. It aims to provide a comprehensive understanding of the current state of methodologies and frameworks. It's important to note that this list is not exhaustive and can be expanded based on specific research needs.

Table 7: Classification of Research Articles based on Existing Techniques and Methodologies

Security Requirements Identification Techniques/ Frameworks	No. of Articles	Articles
SR Backlogs	5	[10] [96] [86] [75] [87]
Modified User Stories (Abuser Story Like User Story)	12	[10] [17] [25] [31] [36] [37] [7] [43] [47] [59] [64] [86]
Framework / Methodology	43	[11] [15] [16] [18] [19] [20] [22] [23] [67] [88] [28] [33] [34] [39] [42] [44] [46] [48] [50] [55] [56] [57] [61] [62] [63] [66] [70] [71] [72] [73] [76] [78] [80] [83] [89] [90]

		[91] [95] [98] [100] [101] [102] [105]
Tool Support	21	[10] [12] [26] [27] [29] [30] [32] [40] [41] [45] [49] [54] [65] [68] [77] [79] [82] [93] [99] [103] [104]
Hybrid Software Process Life Cycle / Model	24	[13] [14] [21] [67] [24] [35] [36] [37] [38] [51] [52] [53] [58] [60] [69] [74] [81] [82] [84] [85] [86] [94] [96] [97]

In the paper, our classification of the literature is based on five distinct groups that emerged from our analysis. During the examination of the literature, a noticeable pattern emerged where discussions about the subject predominantly fell into these five categories:

- **Security Requirements Backlog (SR Backlog):** This term pertains to a systematically organized list of security-related requirements that demand attention within a software development or IT project. It holds particular significance in Agile and DevOps environments, functioning as a central repository to manage and monitor security requirements throughout the project's lifecycle. This backlog streamlines the handling of security concerns at various project stages [10].
- **Modified User Stories:** Often referred to as security-focused user stories or security-driven user stories, this category represents a specialized kind of user story used in software development. These stories, such as Abuser stories, inject a security dimension into requirements, ensuring that security matters are explicitly integrated into the development process [17].
- **Conceptual Frameworks and Methodologies:** This group encompasses systematic and structured approaches utilized for specifying security-related requirements. These frameworks and methodologies facilitate the identification, analysis, and resolution of security issues in software development and IT projects. They aid in understanding the security landscape, defining security objectives, and formulating precise security requirements [11].
- **Tools Supporting Security Requirements Specification:** In this category, tools play a crucial role in fortifying the security stance of software development endeavours. These tools serve a diverse range of functions and assist security experts and development teams in efficiently identifying, documenting, and managing security requirements [12].
- **Hybrid Software Process Lifecycle:** This term denotes an approach that amalgamates elements from various software development methodologies to craft a customized process suited to the specific demands of a project. Instead of strictly adhering to a single methodology, a hybrid approach integrates the strengths of different methodologies to optimize development efficiency and cater to project requirements [13].

These classifications serve as an insightful framework for understanding the landscape of security requirements specification within software development and IT contexts. **Figure 6** provides a visual representation of the distribution of articles across the aforementioned categories. This graphical illustration offers a clear overview of the number of articles that delve into each of the described categories.

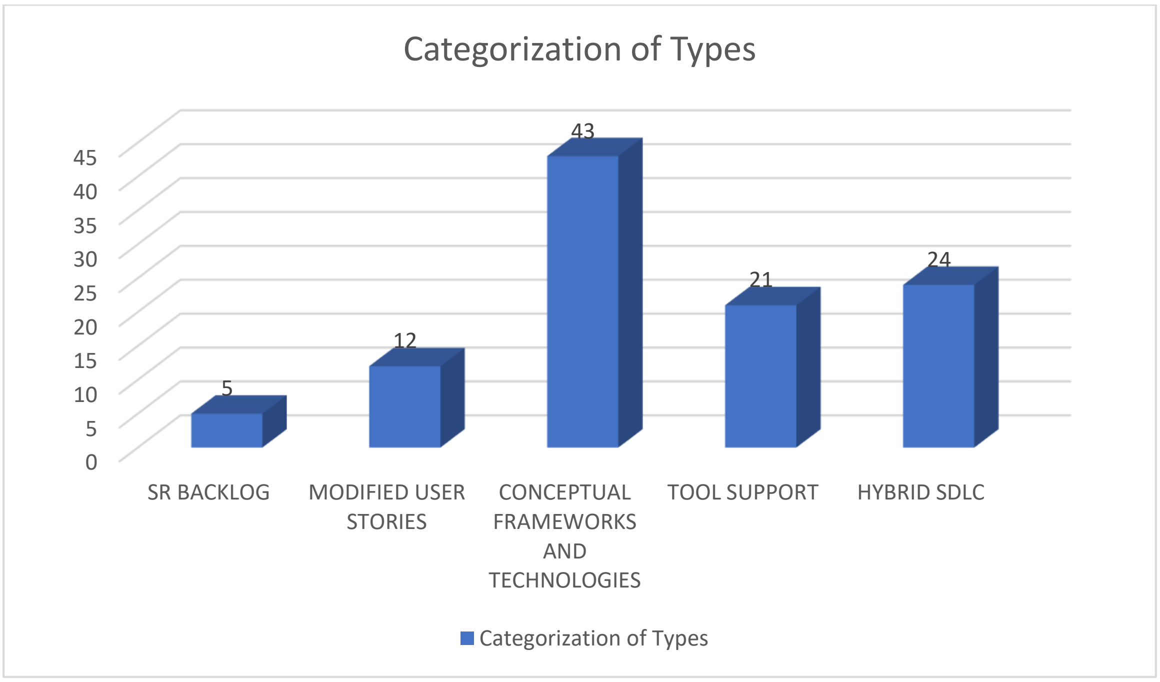


Figure 6: Derived Category of Types

In this chapter a Systematic Literature Review (SLR) was conducted to investigate security requirements specification methodologies. The study followed a well-defined process involving research question formulation, database selection, search string creation, and inclusion/exclusion criteria establishment. The review consisted of two phases: study selection and data extraction, with resulting articles categorized by distribution, year, and identified frameworks/methodologies. Quality assessment ensured reliable findings. The research identified five key categories: Security Requirements Backlog, Modified User Stories, Conceptual Frameworks and Methodologies, Tools Supporting Security Requirements Specification, and Hybrid Software Process Lifecycle. Graphical representation further visualized article distribution among these categories, contributing to a comprehensive understanding of the subject's state-of-the-art methodologies.

2.3 Results and Discussion

In this section, the identified techniques and frameworks have been explained in detail. As **Figure 6Error! Reference source not found.** illustrates through this SLR, 5 distinguish categories have been made that elaborates the outcome of conducted literature. From selected articles 5 papers discussed Security Backlog, 12 and 43 studies discussed Modified user stories and frameworks & Technologies that are being used to specify security related requirements respectively. Moreover, 21 articles proposed tools and lastly there were 24 studies that presented a hybrid software development process. Each one of them have their own strengths and limitations that are discussed lately in this section.

2.3.1 SR Backlog

The most basic way of handling security requirements in Software Application Development is by using Security Requirements (SR) Backlog. This traditional way of gathering security requirement could help developer in maintaining the security level by following the proper guidelines [7]. The security backlog act in accordance with the security principle to ensure that there are not risks and vulnerabilities exists in software product. With the help of Security requirement Backlog security related issues can be mitigated [75]. The SR backlogs were initially being used in Agile Development in Scrums provided the fact that it smooths the process of identification of security requirements and planning the methodology based on backlogs much easier [86]. In the article [87] the authors critically analyse the hybrid technique of security requirement backlogs for specifying security requirements. They combined three techniques known as Common Criteria, Misuse Case and Attack trees.

2.3.2 Modified User Stories

To address the security requirements in requirements specifications another method used by several practitioners is the extended form of user stories such as abuser stories and vulnerability analysis [10]. In the search article [86] the authors conducted their research to incorporate security-oriented development in scrum framework, they also discussed the standard maintained by France industry known as VAHTI. VAHTI provides the set of instructions to implement in Scrum framework by introducing modified user stories, sprint backlog and product backlog in development which are security centric. On the other hand, the article [17] proposes a storyboard-based design methodology to enable the specification and verification of security properties of an Android Application at design time. The research study proposes a new approach to measure confidence and uncertainty in assurance cases. Assurance cases are used to provide evidence that a system meets its intended requirements and functions as expected. However, assurance cases often contain assumptions and uncertainties that can affect the confidence in the system's performance and safety. The paper proposes a new approach to quantify the confidence and uncertainty in assurance cases by using a Bayesian network. The Bayesian network model is used to represent the dependencies and relationships between the different components of the assurance case and to calculate the probability of the system meeting its requirements. Similarly, in the paper [25] the author studies on the privacy requirements pattern for mobile operating system which consist of Android and IOS devices. The author of the article proposed 7 privacy patterns that includes, Authorized use of sensors or portal, avoidance of privacy leakage in user behaviour information collection, guard of personal mobile data, privacy protection over mobile cloud services, authentication of mobile users, financial information protection, and mobile communication secrecy. The authors in the article [64] put forward an extended secure designing methodology for enhancing User Experience (UX) at design phase. The limitation of the proposed solution was that it is only limited to UI Behaviour of Application and not able to capture Non-UI behaviour. Moreover, the author also suggests that most of the issues can be tackles with smart code technique like using HTTPS instead of HTTP when contact servers. On the other hand, the article [31] analysed the effectiveness of the security requirements templates in identification of security requirement in requirements specifications. While the paper [32] presents and an approach to test security requirements Misuse case Programming approach. The author of the paper suggested that such approach can be used for use case specification to acquire malicious behaviour. Moreover, the authors of the research paper [38] presented their approach for the elicitation of security requirements for web application using

Open Web Application Security Project (OWASP) that provides an industry standard security indicator. The authors have merged OWASP in their User Stories to relate them with security requirements. Similarly, the article [39, 43] also conducted research on User stories, wireframes that how they can be beneficial of extracting security requirements. Research study [55] conducted an experiment of user stories classifications and requirements extraction using NLP.

2.3.3 Frameworks and Methodologies

One of the main parts in this study is induction of such articles that presented or discussed framework or methodology for the identification of security requirements. **Figure 7** illustrates various prominent identified frameworks and techniques that are explained later in current section.

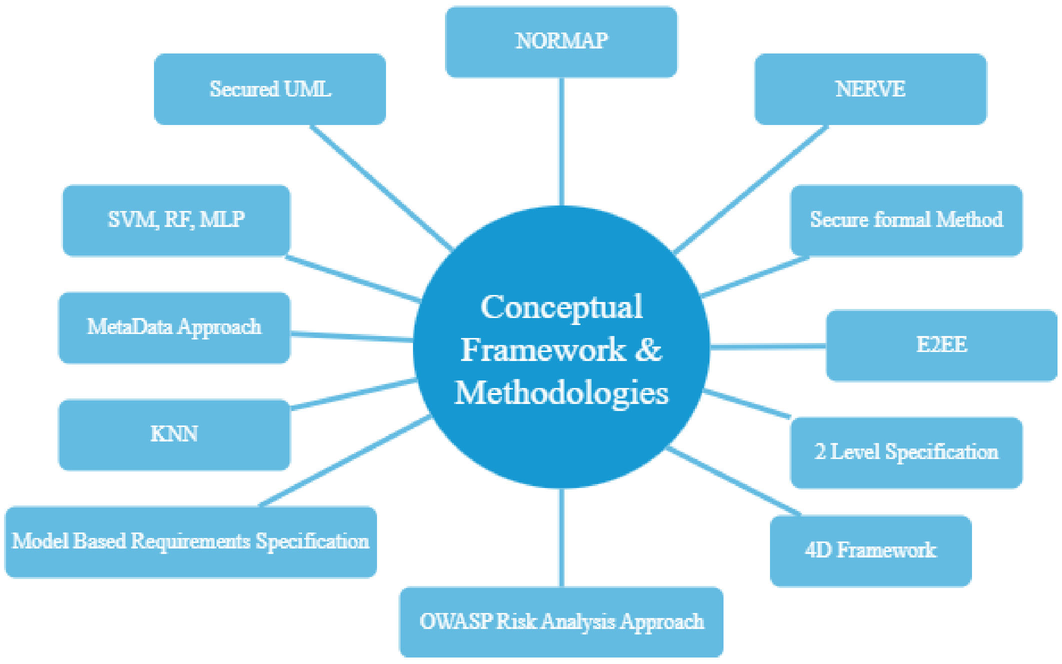


Figure 7: Identified framework & Methodologies

In this phase approximately 42 such studies were selected that proposed some methodology and framework for security requirement elicitation. However, each one has some sort of limitations and domain that will be discussed in this portion. In the article [4] the authors propose a methodology named as Non-functional Requirements Modelling for Agile Process (NORMAP) that uses Agile Use Cases, Agile Loose Cases and Agile Choose cases for requirements gathering, modelling and linking. It's a JAVA based tool having 87% accuracy. This article follows risk driven approach but focuses on all non-functional requirements instead of considering only security aspect. Non-Functional Requirements often get neglected in Agile Process, therefore the authors of the paper [8] developed NERV Methodology: Non-functional Requirements Elicitation, Reasoning, and Validation. In study [8, 92] the writers also investigated the mentioned methodology that can effectively help in extracting non-functional requirements. The methodology uses a combination of different artifacts of Quality models such as Boehm, McCall and ISO standards and uses certain criteria to opt from theses artifacts. The study [9] also conducted research on handling non-functional requirements for IOT and big data projects using scrum process. The proposed approach helps to deal with security and performance requirements individually as well as conflicts among them. The article [11] provides a detailed description of the scenario-based reading method and how it can be used for testing mobile applications. The authors also discuss the benefits of using this method FIT4Apps, such as increased test coverage and early detection of defects. This article presents a method for testing mobile applications using scenario-based reading with FIT4Apps, a tool for generating and executing test scenarios. Whereas, the research studies [13, 94] present a comprehensive review of the relevant literature and proposes a set of security requirements measures that can be used to predict software project and product measures. The authors conducted a preliminary study to validate their proposed security measures and evaluated their effectiveness in predicting project and product measures for a set of Android mobile applications.

In the research study [15] the authors propose a method for security requirement engineering using a structured object-oriented formal language for mobile banking applications. Overall, this article provides a valuable contribution to the field of security requirements engineering for mobile applications. The proposed method is well-designed and effective. The article also provides a comprehensive discussion of the limitations of existing methods and the need for new approaches in the context of mobile banking applications. In article [16] authors present a comparative study of the state-of-the-art End-to-End encryption (E2EE) techniques for mobile messaging applications. Authors of the study provides a detailed review of the existing E2EE techniques, including Symmetric and Asymmetric encryption, Hybrid encryption, and Homomorphic encryption. The authors then compare these techniques based on various criteria such as security, performance, and scalability. The paper [17] provides a novel approach known as a two-level specification approach for developing mobile agent applications. The authors argue that traditional software engineering approaches are not well-suited to mobile agent applications, which require a more flexible and dynamic approach to development. The two-level specification approach consists of a high-level specification and a low-level specification. The high-level specification defines the overall behaviour of the mobile agent application, while the low-level specification defines the specific actions and interactions of the mobile agents. The two levels are connected through a set of mappings that relate high-level specifications to low-level specifications.

The research report [18] discusses the challenges faced by the agency in selecting a biometric system that meets their needs and requirements. The agency decided to use mobile biometric testing and evaluation to assess the performance of different biometric systems in a real-world environment. It provides a detailed description of the mobile biometric testing and evaluation process, including the selection of test subjects, the data collection process, and the evaluation criteria. The articles [28, 44, 53, 57, 89], provides an overview of the current state of software engineering research for mobile apps and identifies several future trends in the field. The authors of the study begin by describing the unique challenges of software engineering for mobile apps, including issues related to platform diversity, limited resources, and user interface design. They then review the current state of software engineering research for mobile apps, discussing the most prominent research topics and approaches. The writers then identify several future trends in software engineering research for mobile apps, including the use of artificial intelligence and machine learning the development of new testing methodologies, and the exploration of new design paradigms. They also discuss the importance of addressing emerging issues, such as privacy and security concerns, and the need for interdisciplinary collaboration to tackle complex problems.

The research paper [12, 29] bring forward a risk-based approach to developing secure Android mobile software. The authors begin by describing the unique security challenges of Android mobile software, including issues related to app permissions, data storage, and network communication. They then introduce the OWASP Risk Analysis Framework, which provides a systematic method for identifying and assessing security risks. The authors propose a methodology for using the OWASP Risk Analysis Framework to develop security requirements specifications for Android mobile software. This methodology involves identifying the assets to be protected, identifying the potential threats and vulnerabilities, and assessing the likelihood and impact of each threat. Based on this analysis, the authors develop a set of security requirements that address the identified risks. Moreover, they also provide a case study that demonstrates the application of their methodology in practice. The case study involves the development of a secure Android mobile application for a financial institution, and the authors show how their methodology can be used to identify and address security risks specific to this context. Overall, this article provides a valuable contribution to the field of mobile software security. The proposed methodology offers a practical and systematic approach to developing security requirements that are tailored to the specific risks faced by Android mobile software. The case study provides concrete examples of how this methodology can be applied in practice, making it a valuable resource for developers and security practitioners alike.

The papers [34, 42, 67, 68, 70] presented a computer-aided approach to analysing requirements for software systems, with a focus on assessing their consistency, completeness, and correctness. The authors begin by describing the challenges of managing requirements for complex software systems, including issues related to ambiguity, inconsistency, and incompleteness. They then introduce their proposed approach, which involves using automated tools to analyse requirements specifications and identify potential issues related to consistency, completeness, and correctness. The authors describe several techniques for analysing requirements, including natural language processing [23, 44, 52], formal methods, and model-based techniques [53, 73, 83, 105]. They also provide a case study that demonstrates the application of their approach in practice, using a commercial aircraft control system as an example. The academic paper [37, 95] presents a theoretical framework for understanding and assessing data quality. The authors begin by defining data quality and describing its importance in various domains, including business, science, and public policy. They then introduce their proposed framework, which is based on four dimensions of data quality: intrinsic, contextual, representational, and accessibility. The intrinsic dimension refers to the inherent characteristics of data, such as accuracy, completeness, and consistency. The contextual dimension considers the context in which data is collected and used, including factors such as data source, purpose, and relevance. The representational dimension concerns the format and structure of data, including issues related to data modelling, encoding, and storage. The accessibility dimension focuses on the ease of access and use of data, including issues related to data security, privacy, and usability. The authors then provide a detailed discussion of each dimension, including specific sub-dimensions and metrics for assessing data quality. They also provide a case study that demonstrates the application of their framework in practice, using data from a healthcare system as an example. The authors of the article [40] begin by describing the challenges of extracting metadata from legal documents, including issues related to inconsistent terminology, ambiguous language, and complex syntax. They then introduce their proposed approach, which involves using natural language processing techniques to automatically identify and extract metadata from legal documents. The approach involves several stages, including pre-processing of documents, identification of relevant sections, and extraction of metadata using a combination of rule-based and machine learning approaches. The authors provide a detailed discussion of each stage, including the specific techniques and tools used.

In a research paper [46, 84] the authors present a collection of security patterns to capture regulatory security requirements early in the software development lifecycle. The authors argue that existing security patterns focus on technical aspects of security, while regulatory requirements are often overlooked or addressed only later in the development process. The authors begin by describing the challenges of capturing regulatory security requirements early in the development process, including issues related to ambiguity, complexity, and dynamic nature of regulatory requirements. They then introduce their proposed approach, which involves identifying patterns of constraints that capture common regulatory security requirements. The approach involves several steps, including identification of relevant regulatory requirements, identification of common themes across these requirements, and formulation of constraints that capture these themes. The authors provide a detailed discussion of each step, including the specific techniques and tools used. The authors then evaluate their approach using a case study, comparing the results to existing security patterns. The results show that their approach can capture regulatory security requirements effectively and efficiently, and can help ensure that these requirements are addressed early in the development process. This study [51] deduced an approach to automatically extract access control policies from natural language documents. The authors argue that manually extracting access control policies from these documents is time-consuming and error-prone, and that automated approaches can help improve efficiency and accuracy. The proposed approach involves several steps, including identification of relevant sections of the document, identification of access control policy rules, and conversion of these rules into a formal representation. The authors describe the specific techniques and tools used for each step, including natural language processing, machine learning, and logic programming.

The research studies [58, 59, 65] presents an approach for identifying and distilling privacy requirements for mobile applications. The authors argue that privacy is a critical concern for mobile applications, and that many developers struggle to identify and address privacy risks in their software. The provided approach involves several steps, including identifying potential privacy risks, mapping risks to privacy requirements, and then distilling those requirements into a concise and actionable set of guidelines. The authors describe the specific techniques and tools used for each step, including privacy risk analysis, privacy goal modelling, and privacy guideline generation.

The paper [66] discusses the impact of different vectorization methods on the classification of non-functional requirements. The authors have explored various vectorization techniques such as Word2Vec, Doc2Vec, and GloVe and evaluated their performance on a dataset of non-functional requirements. The authors introduced the concept of non-functional requirements and their importance in software engineering. The authors then describe the dataset used for their experiments, which comprises of 3000 non-functional requirements labelled with seven different categories such as usability, reliability, and performance. Moreover, they have used various vectorization techniques to convert the textual data of non-functional requirements into numerical vectors that can be used as input to machine learning algorithms. They have then trained multiple classifiers such as Support Vector Machines (SVM), Random Forests, and

Multi-Layer Perceptron’s (MLP) on the vectorized data and evaluated their performance using metrics such as accuracy, precision, recall, and F1-score. The paper [71, 96] provides an overview of the quality factors that should be considered when developing mobile applications. The authors discuss several best practices that can be used to ensure the quality of mobile applications, including usability, performance, security, compatibility, and maintainability. This article emphasizes the importance of designing mobile applications with the end-user in mind, as usability is a key factor in the success of any mobile application. They also highlight the importance of performance optimization, given the limited resources available on mobile devices. In addition, the article provides a detailed discussion of security considerations for mobile applications, including data encryption, secure data storage, and user authentication. The authors also discuss the importance of compatibility with different devices and platforms, as well as the need for maintainable code to ensure the longevity of the application.

The article [75] proposes a framework for ensuring the security of mobile applications adopted by small and medium enterprises (SMEs). The authors argue that SMEs are at a greater risk of cyber-attacks due to their limited resources and lack of technical expertise. The framework consists of five main components: risk assessment, security policies and procedures, technical controls, training and awareness, and incident response. Each component is designed to address specific security risks associated with the adoption of mobile applications in SMEs. The risk assessment component involves identifying potential threats and vulnerabilities associated with the use of mobile applications and assessing their potential impact on the organization. The security policies and procedures component involve establishing policies and procedures to ensure the security of mobile applications and the data they handle. The technical controls component involves implementing technical measures such as encryption, access controls, and monitoring to protect mobile applications and data. The training and awareness component involve providing training to employees on how to use mobile applications securely and raising awareness about the risks associated with their use. Finally, the incident response component involves developing a plan to respond to security incidents involving mobile applications. On the other hand [78] proposes a framework for eliciting and tracing security requirements to the design of a system. The framework is based on three key components: The Common Criteria, heuristics, and UMLsec.

The Common Criteria is an internationally recognized standard for evaluating the security of information technology products. It provides a set of security requirements that can be used as a basis for developing security requirements for a specific system. The authors propose using the Common Criteria as a starting point for eliciting security requirements.

Heuristics are guidelines or rules of thumb that can be used to identify potential security vulnerabilities or weaknesses in a system. The authors propose using heuristics to supplement the security requirements identified through the Common Criteria and to identify any additional security requirements that may not be explicitly stated in the Common Criteria.

UMLsec is a security extension to the Unified Modelling Language (UML) that provides a set of modelling constructs for specifying security requirements and mechanisms. The authors propose using UMLsec to model the security requirements identified through the Common Criteria and heuristics and to trace those requirements to the design of the system.

2.3.4 Tool Support

Another derived category from the literature in tool support for specification of non-functional requirements. In Figure 8 can be seen some of the famous tools identified in the literature. They are being used in various organizations around the globe. The research paper [5], presents an A visual tool for modelling non-functional requirements in agile processes known as NORMATIC. That is a visual modelling tool designed to help agile teams capture and manage non-functional requirements. It provides a graphical representation of the requirements, which can be easily understood by both technical and non-technical stakeholders.

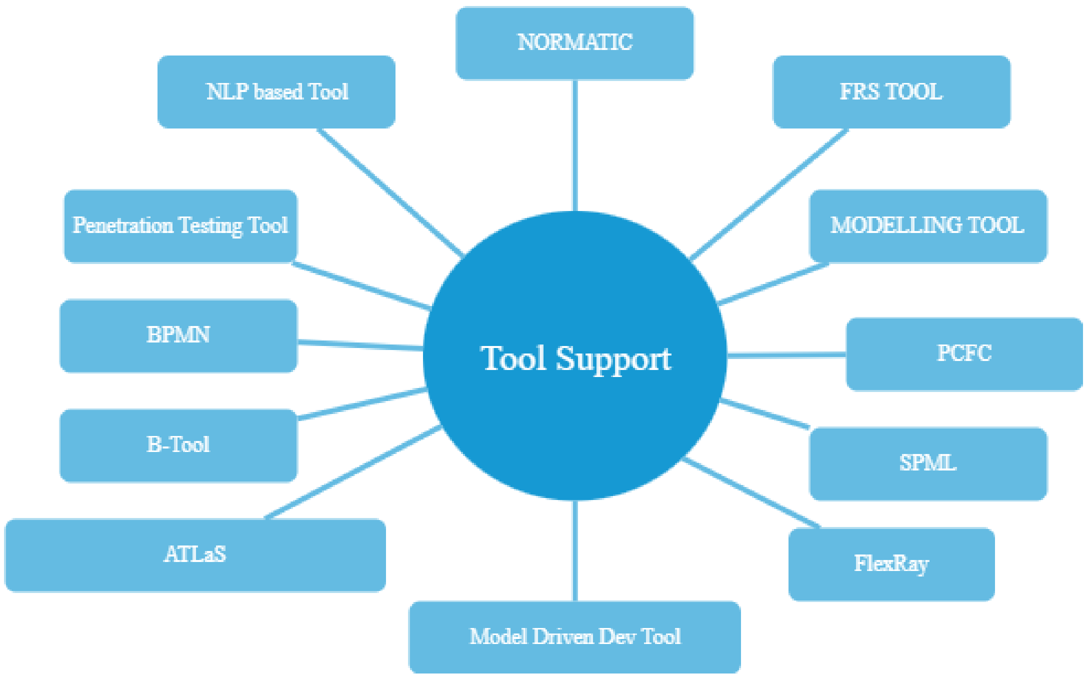


Figure 8: Tool Support

The research article [21] developed a model known as a forensic requirement specification (FRS) for developing forensic models for mobile device malware. The authors discuss that the increasing use of mobile devices has led to an increase in mobile malware, and there is a need for effective forensic models to investigate and analyse mobile malware. The FRS approach presented in the paper consists of a set of requirements that specify the functionality, performance, and security requirements for a mobile device malware forensic model. The FRS is based on the ISO/IEC 29110 standard for software engineering, and it provides a structured and systematic way to develop and evaluate forensic models. Moreover, the author in the paper [22, 35, 41] developed an NLP based tool to detect such privacy requirements that tends to repurpose or over-collect personal data and decisional user requirements. In [41, 45, 98] the author uses an NLP based machine learning algorithm to classify the identified requirements into different security categories, such as confidentiality, integrity, and availability. [61, 63] also conducted similar work and developed a tool that takes as input software requirements and generates security test cases based on those requirements. The tool first identifies the security-relevant requirements and then generates test cases that satisfy those requirements. [77] discusses the similar with addition of reusable security requirement concept.

The authors presented a formal model to analyse the permission authorization and enforcement mechanism in the Android operating system in the article [24]. According to paper Android's permission system is complex and can lead to security vulnerabilities if not properly implemented. The formal model presented in the paper is based on the Permission Control Flow Graph (PCFG), which captures the permission authorization and enforcement mechanism in the Android framework. The PCFG is used to model the permission flow in Android applications and identify potential security vulnerabilities. The study [25] developed a tool for visual specification and verification of secure process movements. The authors argue that secure process movements are critical for ensuring the security of sensitive data and systems, and that visual tools can help improve the accuracy and efficiency of the verification process. The tool presented in the paper is called Secure Process Movement Diagram (SPMD), which is a graphical tool that allows users to specify and visualize the movements of sensitive data and processes within a system. The tool is based on a formal language called Secure Process Movement Language (SPML), which allows users to specify the movement of data and processes in a precise and unambiguous way.

The article [27] presents a new approach to scheduling FlexRay communications that considers security considerations. FlexRay is a communication protocol used in safety-critical systems, such as automotive applications. However, the traditional approaches to scheduling FlexRay communications do not consider security, which can leave these systems vulnerable to attacks. The authors propose a new scheduling engine, called SAFE, that considers security considerations when scheduling FlexRay communications. The proposed approach involves analysing the security properties of each communication message and using this information to allocate time slots in the FlexRay schedule. The authors also conducted experiments to evaluate the effectiveness of the proposed approach. The results demonstrate that the SAFE scheduling engine is effective in reducing the vulnerability of FlexRay communications to security threats. The article [1, 87] provides a systematic review of literature on developing mobile applications using model-driven development (MDD). The review focuses on the benefits and challenges of using MDD for developing mobile applications and the various MDD approaches that have been proposed for developing mobile applications. The review also discusses the current state of research on MDD for mobile applications and identifies areas for future research.

The research paper [36] presented a tool for recovering traceability links between software artifacts. Traceability links are important for software maintenance and evolution, as they help developers understand the relationships between different artifacts and the impact of changes on the system. The tool developed in the paper is called ATLaS (Automatic Traceability Linking System), which combines information retrieval and semi-supervised learning techniques to recover traceability links. The framework uses natural language processing techniques to analyse the text of software artifacts and identify potential links based on the similarity of the text.

The paper [50, 97] discusses about penetration frameworks and development issues in secure mobile application development. The authors briefed that mobile applications are increasingly being used for sensitive and confidential tasks, such as financial transactions and healthcare data management, making security a critical issue for mobile app developers. The paper reviews a range of literature related to penetration testing frameworks and development issues in secure mobile application development. The authors identify several key themes, such as the importance of threat modelling and risk assessment, the need for secure coding practices and secure software development lifecycles, and the challenges of testing and assessing mobile app security.

The article [72] describes an automated approach to capture and validate security requirements for mobile apps. The authors address the challenge of capturing and validating security requirements for mobile apps, which are often complex and require expertise in security and mobile development. The approach involves a tool that automatically analyses the source code and generates security requirements based on the identified security risks. The tool uses a set of security rules to identify security risks in the code and generate corresponding security requirements. The generated requirements are then validated using a set of predefined criteria to ensure that they are complete and consistent. Whereas, in paper [74] the author presents a formal Android permission model that is based on the B Method. Paper address the challenge of designing a reliable and secure permission model for Android, which is a complex and dynamic operating system with a large number of apps and users. The developed permission model uses the B Method, which is a formal method for software development that is based on mathematical notation. The model defines the Android permission architecture in terms of the different types of permissions, their relationships, and the conditions under which permissions are granted or denied.

The research [93] article presented a framework for security requirements engineering (SRE) that uses the Business Process Model and Notation (BPMN) 2.0.2 extension model. The authors aim to provide a structured approach for eliciting, analysing, specifying, and validating security requirements in the development of information systems. The author put forward framework consists of four phases: 1) context definition, 2) security requirements elicitation, 3) security requirements analysis and specification, and 4) security requirements validation. The framework uses the BPMN 2.0.2 extension model to represent the security requirements and their relationships with the business processes.

2.3.5 Hybrid Software Process Cycle/Model

One of the most common categories used to alter requirement specification in software development is by combining one or more process together. **Figure 9** demonstrate some notable process models that were discussed in the literature.

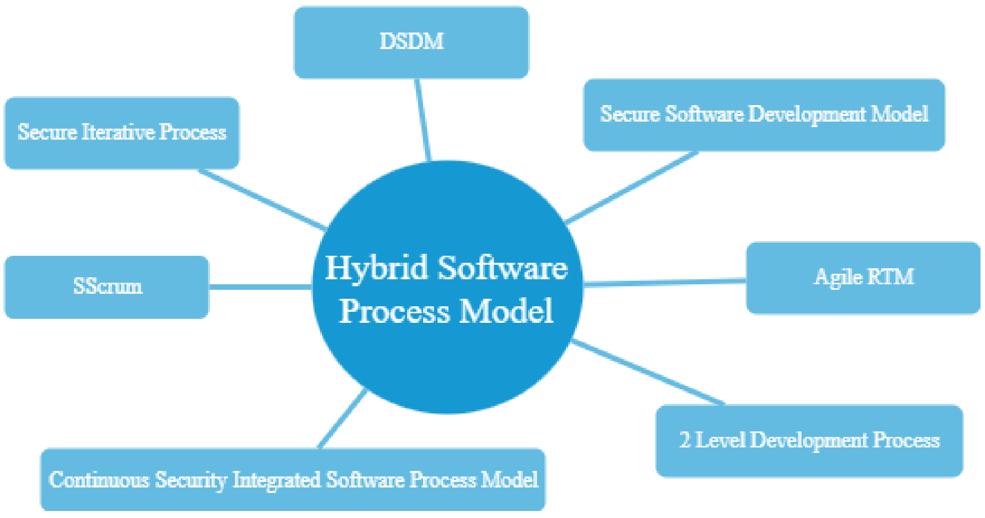

DSDM
Secure Iterative Process
Secure Software Development Model
Hybrid Software Process Model
SScrum
Agile RTM
2 Level Development Process
Continuous Security Integrated Software Process Model

Figure 9: Hybrid Process Model

The article [6,30] proposed a modified process for gathering of security requirements in software development. The author brings forward an iterative process for developing security features and security assurance techniques. Moreover, in [30] authors emphasized on malware analysis to identify vulnerabilities. In the research study [7, 19] the authors provided the information about DSDM (Dynamic Systems Development Method) and [80] propose a Secure Software Development Model. They are an Agile project delivery framework that is designed to be flexible and adaptable to changing requirements. On the other hand, the author in the article [14] proposed an Agile Requirement Traceability Matrix (RTM) that is lightweight, flexible, and integrated into the Agile development process. The Agile RTM is designed to be updated continuously throughout the development process, allowing teams to track the status of requirements in real-time and ensure that they are being addressed. The authors provide a step-by-step guide for implementing an Agile RTM, including how to define requirements, create the matrix, and update it throughout the development process. They also provide a case study demonstrating the effectiveness of the Agile RTM in improving traceability in an Agile development project.

The authors of the papers [17, 48, 49] argue that traditional specification methods are not well-suited to mobile agent applications, which involve autonomous agents that are capable of moving between different locations and interacting with different systems. [17] propose a two-level specification approach that separates the functional and mobility aspects of the system. The functional specification level defines the behaviour of the mobile agents, including their actions and interactions with other agents and systems. The mobility specification level defines the movement and interaction of the agents within the system, including their communication protocols, network connections, and security requirements. Whereas, [48,49] discussed interviews, surveys, and brainstorming sessions.

Other than that, in research papers [31,32, 54] the authors discussed the addition of Natural Language Programming Approach for Security based Requirements Specification and testing. In the research articles the authors implemented NLP model in hybrid Software Development Process.

The articles [56, 81, 90, 91] briefed about a model for safe agile development (ScrumS) that addresses the challenge of developing secure and reliable software using agile methodologies. The authors discuss that while agile development has many benefits, it often lacks sufficient guidance on how to address security concerns. ScrumS is designed to fill this gap by incorporating a set of security-specific practices into the standard Scrum framework. The authors outline a set of security practices that are incorporated into ScrumS. These practices are grouped into three categories: secure development practices, security testing practices, and security management practices. Examples of these practices include security requirements elicitation and analysis, threat modelling, security testing, and security code reviews.

The research paper [76, 77, 88] discusses the challenges in developing secure and accessible mobile applications. The authors argue that mobile research ecosystems, which involve various stakeholders such as developers, users, and regulators, need to collaborate to address these challenges effectively. The paper presents a review of related work on security and accessibility in mobile applications, and then describes a case study of a mobile research ecosystem that involves multiple stakeholders. The case study includes an analysis of the stakeholders' perspectives on security and accessibility, as well as their views on the importance of collaboration and communication in addressing these issues. The authors also propose a model for integrating security and accessibility considerations into the mobile development process. The model consists of three phases: requirements gathering, design and development, and testing and evaluation.

The authors of the research studies [79, 33, 47, 64, 69] brings a comprehensive hybrid framework for developing secure software. The authors argue that security should be integrated into every phase of the software development lifecycle and that secure software development should be viewed as a continuous process rather than a one-time event. The framework consists of four main components: planning and requirements, design and implementation, testing and verification, and deployment and maintenance. Each component is designed to address specific security risks associated with the software development process. 1) The planning and requirements component involve identifying security requirements for the software and incorporating them into the development plan. This includes identifying potential threats and vulnerabilities and determining how to address them through security controls and measures. 2) The design and implementation component involve designing and implementing security controls and measures to address the security requirements identified in the planning and requirements phase. This includes developing secure coding practices, using secure development frameworks, and incorporating security testing into the development process. 3) The testing and verification component involve testing the software to ensure that it meets the security requirements identified in the planning and requirements phase. This includes testing for vulnerabilities, conducting penetration testing, and verifying that security controls and measures are functioning properly. 4) The deployment and maintenance component involve deploying the software in a secure manner and maintaining it over time. This includes configuring the software to be secure in its operating environment, monitoring the software for security incidents, and applying patches and updates as needed.

2.4 Answers to Research Questions

The primary objective of this extensive research endeavour was to conduct a thorough exploration of the existing body of literature, aiming to unveil insightful responses to the carefully formulated research inquiries. This pursuit led us to precisely scrutinize a diverse array of articles carefully curated from prestigious Journals and Conferences, yielding a wealth of pertinent and valuable data. To facilitate a deeper comprehension of the landscape, a categorization of the chosen articles was undertaken, resulting in the delineation of five distinctive and prominent Categories: 1) Security Requirement Backlog, 2) Modified User Stories, 3) Frameworks & Methodologies, 4) Tool Support, and 5) Hybrid Software Development Process. The rationale behind such a systematic categorization stemmed from the prevalent adoption of these methodologies within industries, substantiating their relevance and importance.

The meticulous and systematic review and analysis of the meticulously selected Research Articles have yielded profound insights into the contours of the subject matter at hand. Throughout this comprehensive review, it became evident that researchers have made noteworthy strides in advancing the discourse within the domain under consideration. While considerable strides have been made, it remains equally clear that there exists a vast expanse of uncharted territory awaiting exploration. The cumulative efforts of these pioneering researchers have undoubtedly paved a well-illuminated path, one that beckons subsequent explorers to build upon these foundations and delve deeper into the nuances that continue to shape security-related requirements in software development. As these insights ripple through academia and industry, they not only enrich scholarly discourse but also provide tangible assistance to practitioners seeking to fortify their own research or operational endeavours. The convergence of research and practice, exemplified through these studies, stands as a testament to the perpetual evolution of the field and the relentless pursuit of knowledge in service of enhanced security and efficiency.

The following Research Questions were formulated. The scrutiny of the Literature provided us with the appropriate answers that we extracted during the SLR.

RQ1: *What are the existing Security Requirements Identification Methodologies and Tools Techniques?*

Based on the conducted Systematic Literature Review, it can infer that there are several existing Methodologies & frameworks and Tool techniques for identifying security related requirements in the context of software requirement engineering. Security Requirements Backlog and Modified User Stories are among basic approach used for elicitation of security requirements in software development. But there are other techniques and models also that could be handy with much more accuracy and fast paced as compared to SR Backlogs and Unified User Stories.

Table 8: List of Existing Methodologies and Framework Techniques

Methodologies & Framework Techniques	**Abbreviation**	**Accuracy**	**Research Study**
NORMAP	*Non-functional Requirements Modeling for Agile Process*	87%	[4]
NERV	*Nonfunctional Requirements Elicitation, Reasoning, and Validation*	85.6%	[8], [92]
FIT4APP	-	N/A	[12]
CBSR	*Case Based Security Reasoning*	N/A	[14], [94]
SRE - SOOFL	*Security Requirement Engineering using a Structured Object-Oriented Formal Language*	87%	[16]
E2EE	*End-to-End encryption techniques*	N/A	[16]
Two Level Security Specification	-	88%	[17]
OWASP	*Open Web Application Security Project*	94%	[13], [29]
CAASR	*Computer Aided Approach for Analyzing Security Requirement*	N/A	[34], [42], [67], [68]

Natural Language Process Based Requirement Specifications	-	80%	[23], [44], [52]
RE Using Formal Methods and Model Driven Technique	-	82%	[53], [73], [83]
4D Framework	*4-Dimensional Framework for Health-Related Application*	N/A	[37]
Extraction of Meta Data	-	N/A	[40]
Automatic Extraction of Access Control Policies	-	N/A	[51]
Distilling Privacy Related Requirements		82%	[62], [58], [59], [65]
Vectorization Methods		91%	[66]
Quality Factors for Mobile App	-	N/A	[71], [96]
Secure Mobile App Development for SME	*Secure Mobile App Development for Small and Medium Enterprises*	N/A	[75]
UMLSec	Secure Unified Modelling Language	85%	[78]

Above given **Table 8Error! Reference source not found.** provides an in site to the existing methodologies which researchers have identified. The provided information regarding frameworks and methodologies were initially grouped under title of Frameworks & Methodologies in **Section 3.3** in detail. Moreover, **Table 9** provides the list of Identified tools in **Section 3.4.**

Table 9: Existing Tools

Tools Support	**Description**	**Papers**
NORMATIC	Modelling Tool for non-functional Requirement	[5]
Forensic Requirements Specification Tool	-	[21]
NLP Based Tool	-	[22], [35], [41], [45], [61], [63], [98]
PCFC	Formal Model based Permission Control Flow Graph	[24]
SPML	Secure Process Movement Formal Language	[25]
FlexRay	Secure Communication Protocol Tool	[27]
Model Driven Development Tool	-	[87], [105]
ATLaS	Automatic Traceability Linking System tool	[36]

Penetration Testing Tool	-	[50], [97]
B-Tool	Automatic Capturing and Validation Requirements tool	[72], [79]
BPMN 2.0	Business Process Model and Notation	[99]

Apart from the state-of-the-art tools and techniques, this Paper also identified Hybrid Software Development Process Cycle that could also be used to develop secure Software Applications. The only drawback of such approach is its exhaustive nature that include a lot of manual work. it should be noted that a lot of tools are domain specific and can't be used as general. Mostly these are limited for one domain because of the fact they each type of niche has its own time frame and set of requirements. Especially Mobile Application Development that has the quickest creation phase. It was also observed that researchers have separately worked for the requirements specification methodologies and frameworks for Web, Desktop and Mobile Applications. Most of the Mobile App related work were also focused on Health, Banking and Defence related Applications. However, Development Models can be selected with little modifications. Overall, these are good framework and methodologies that can be used to specify security related requirements in Software Development.

The existing landscape of Security Requirements Identification Methodologies and Tools Techniques within the context of mobile application development has been a subject of profound exploration. This research endeavours to shed light on this crucial domain by systematically reviewing relevant literature. The primary objective is to discern the methodologies and techniques that have emerged from scholarly discourse and practical applications. The systematic review and analysis of these selected research articles reveal a substantial body of work by researchers. While much progress has been made in the field, it becomes evident that there is an ongoing journey towards comprehensive solutions. The methodologies and techniques put forth by these researchers serve as foundational building blocks, guiding the way for further advancements and practical implementations.

RQ2: *How effective are the different methodologies for reviewing security-related requirements in software requirement specifications?*

The evaluation of the effectiveness of various methodologies for reviewing security-related requirements in software requirement specifications (SRS) stands as a pivotal aspect of this research. By conducting a systematic literature review, we delve into this inquiry with the aim of providing insights into the strengths and limitations of these methodologies. Based on research conducted in this SLR study, several methodologies and tools have been identified for reviewing security-related requirements in software requirement specifications. These approaches vary in their effectiveness based on factors such as context-specific suitability, coverage of security aspects, formality and rigor, automation support, and expertise and skill of reviewers.

1 **Context-Specific Suitability:** The effectiveness of a methodology is closely tied to its suitability for the specific context of the software development project. For instance, [43] proposed an approach for reviewing security-related aspects in agile requirements specifications of web applications, considering the agile development context. Research

Study [76] focused on quality factors in mobile application development. By considering the specific context, these methodologies can align security requirements with the needs and constraints of the project, leading to more effective security reviews.

2 **Coverage of Security Aspects:** The effectiveness of a methodology is also determined by its ability to address various security aspects comprehensively. Some methodologies, such as those proposed in [50] and [65], consider multiple dimensions of security requirements, including confidentiality, integrity, and availability. By covering a wide range of security concerns, these approaches enhance the overall security posture of the software.

3 **Formality and Rigor:** Formal methods can bring a higher level of rigor and precision to the review process. For example, [100] presented an approach that uses formal modelling to systematically analyse and specify security requirements for converged web-mobile applications. Similarly, paper [80] proposed a rule-based multi-criteria framework for sustainable-security assessment of web applications. Formal methods can help in ensuring correctness and consistency in security requirements.

4 **Automation Support:** The effectiveness of a methodology can be augmented by leveraging automation tools for security analysis. For instance, [65] introduced a security testing tool, MCP, which is driven by requirements and automatically generates test cases for security-related requirements. Automation tools like MCP can efficiently identify security issues in requirements specifications, enabling early detection of vulnerabilities.

5 **Expertise and Skill of Reviewers:** Finally, the effectiveness of any review process depends on the expertise and skill of the reviewers involved. Studies like [95] provide a comprehensive survey of machine learning security attacks and defence approaches for emerging cyber-physical applications. Such surveys help reviewers stay updated with the latest security threats and mitigation techniques, thus improving the quality of security reviews.

The selected research articles encompass a spectrum of methodologies, ranging from Security Requirement Backlog to Modified User Stories, Frameworks & Methodologies, Tool Support, and Hybrid Software Development Process. This diversity of approaches mirrors the multifaceted nature of security concerns in software development. Through a comprehensive analysis of these methodologies, we strive to delineate their effectiveness based on several criteria, such as their ability to identify security vulnerabilities, streamline development processes, and minimize the risk of breaches. Furthermore, we assess their adaptability to different contexts and their scalability to various project sizes.

In conclusion, the effectiveness of different methodologies for reviewing security-related requirements in software requirement specifications varies based on factors like context-specific suitability, coverage of security aspects, formality and rigor, automation support, and the expertise of reviewers. By considering these factors and adopting appropriate methodologies and tools, organizations can enhance the effectiveness of their security reviews and ultimately deliver more secure software products.

RQ3: *What are the available Tools and techniques or Framework for specification of security related Requirements in mobile application development?*

During the investigation of the literature few tools, techniques, and frameworks have been identified for specifying security-related requirements in mobile application development.

These tools aim to enhance the security of mobile apps by identifying and addressing potential vulnerabilities and threats. Here are some of the notable ones:

1 **SeMA (Secure Mobile App):** SeMA is a design methodology for building secure Android apps. It focuses on integrating security requirements into the development process to ensure that security considerations are considered from the early stages of app development [64].
2 **OWASP Risk Analysis Driven Security Requirements Specification:** This framework is proposed for secure Android mobile software development. It is based on the Open Web Application Security Project (OWASP) guidelines and emphasizes analysing risks to drive the specification of security requirements in the development process [33] [57].
3 **NERV (Non-functional Requirements in agile software development):** NERV is a lightweight process for addressing non-functional requirements, including security, in agile software development. It aims to integrate the consideration of non-functional requirements into agile practices [14].
4 **SAFE (Security-Aware FlexRay Scheduling Engine):** SAFE is a framework that addresses security requirements for the FlexRay communication protocol used in automotive systems, including mobile applications. It focuses on securing communication protocols and ensuring safety in vehicular networks [31].
5 **ATLaS (Traceability Links Recovery Combining Information Retrieval and Semi-Supervised Techniques):** ATLaS is a framework that helps recover traceability links between security requirements and other artifacts. It uses information retrieval and semi-supervised techniques to improve traceability and ensure that security requirements are accurately linked to another project artifacts [40].
6 **SeMSy (Secure Mobile System):** SeMSy is a security modeling framework for mobile systems that includes mobile applications. It aims to model security requirements and analyze potential threats in mobile systems, including mobile apps [46].
7 **MCP (A Security Testing Tool Driven by Requirements):** MCP is a security testing tool that leverages security requirements to automatically generate test cases and validate the security of software applications, including mobile apps [65].
8 **Non-Functional Requirements Elicitation Guideline for Agile Methods:** This guideline provides insights into eliciting non-functional requirements, including security requirements, in agile development environments, helping to ensure that security considerations are not overlooked [98].

RQ4: *What are the key challenges and limitations associated with the current methodologies used for reviewing security-related requirements in software requirement specifications?*

Several key challenges and limitations associated with the current methodologies used for reviewing security-related requirements in software requirement specifications have been identified. These challenges can impact the effectiveness and reliability of the security review process and may lead to potential security vulnerabilities in the final software product. Here are the key challenges and limitations:

1 **Ambiguity in Security Requirements:** Security requirements can sometimes be vague or ambiguous, making it difficult for reviewers to precisely understand and address them.

This ambiguity may result from poorly defined terminologies or lack of clarity in expressing security needs [15].

2 **Incomplete Requirements:** Inadequate or incomplete security requirements can be a significant challenge. When security requirements are not fully specified, the reviewers may not have sufficient information to evaluate or implement the necessary security measures [15].

3 **Inconsistency and Conflicts:** Inconsistencies can arise when different security requirements or elements overlap or contradict each other. Resolving such conflicts is crucial to ensuring that the security specifications are coherent and effective [25].

4 **Lack of Integration with Development Process:** If the security review process is not well integrated into the overall software development process, security considerations may be overlooked or addressed too late in the development lifecycle [25].

5 **Human Error and Subjectivity:** Security requirement reviews are often conducted by human reviewers, and their effectiveness may be influenced by individual biases, expertise, and experiences. Human error and subjectivity could lead to missed security issues [25].

6 **Complexity of Security Analysis:** Analysing and validating security requirements can be complex, especially in large and intricate software systems. This complexity may hinder a comprehensive and accurate security review [32].

7 **Lack of Tool Support:** While there are some tools available to aid in security requirements review, the current toolset may not be comprehensive enough to cover all aspects of security analysis and verification [25].

8 **Dynamic Nature of Security Threats:** The threat landscape is constantly evolving, and new security threats emerge regularly. Static review methodologies may not be sufficient to address dynamic and emerging security concerns [34].

9 **Trade-offs with Functional Requirements:** In certain cases, security requirements may conflict with functional requirements, and finding an optimal balance between security and functionality can be challenging [34].

10 **Resource and Time Constraints:** Security reviews may require significant time and resources, which can be a constraint in fast-paced development environments [41].

11 **Lack of Domain-Specific Security Knowledge:** Reviewers may not always possess in-depth knowledge of the specific domain or technology, leading to potential oversights in domain-specific security requirements [58].

Addressing these challenges requires a comprehensive approach that combines standardized methodologies, automation through tools, integration with the development process, and continuous monitoring of emerging security threats. Additionally, employing experienced security professionals and domain experts in the review process can enhance the effectiveness of identifying and addressing security-related requirements.

2.5 Findings and Limitations

Our exhaustive examination encompassed an in-depth analysis of 97 research articles that presented diverse frameworks and methodologies aimed at the specification of security-related requirements within the context of software development, particularly focusing on Android applications. This comprehensive review process unveiled several noteworthy and thought-provoking findings:

- **Lack of Emphasis on Security Requirements:** A significant portion of the reviewed literature highlighted a prevailing trend of neglecting security-related requirements in the initial phases of Android application development. This observation underscores the need for greater awareness and incorporation of security considerations during requirement specification.
- **Diverse Approaches:** The identified research articles presented a range of methodologies, techniques, and tools aimed at integrating security requirements into the development process. These approaches exhibited variations in their scope, depth, and effectiveness, reflecting the evolving landscape of security in software engineering. However, majority were inclined towards Web and Desktop application development and very few included android aspect but were not effective because of the fact that Android Development process is very different as compared to desktop and website development.
- **Integration Challenges:** Many of the proposed methodologies faced challenges in effectively integrating security requirements without impeding the agile and rapid nature of Android application development. Striking a balance between security and development efficiency remains a key concern.
- **Regulatory Compliance:** While some frameworks acknowledged the significance of adhering to industry standards and regulatory requirements, there were instances of overlooking compliance-related security demands. This emphasizes the need for thorough consideration of legal and regulatory aspects during requirement specification.
- **Emergence of Android:** The integration of security into Android app development process was observed in some frameworks, signifying a growing acknowledgment of the need for security to align with their iterative and adaptive development processes.
- **Non-functional Considerations:** Security requirements were predominantly categorized as non-functional requirements, often intertwined with other non-functional aspects like performance and usability. This interdependency poses challenges in effectively addressing security concerns.
- **Variability in Terminology:** The lack of standardized terminology for security-related concepts led to inconsistencies in how security requirements were specified across different frameworks. A uniform vocabulary could facilitate clearer communication.

The validity of this study rests on the rigor of the systematic literature review methodology employed. The search process encompassed a wide array of reputable databases and a systematic inclusion/exclusion criterion, ensuring comprehensive coverage of relevant research articles. This comprehensive approach enhances the credibility of the findings and their relevance to the broader context of Android application security. However, it's important to discuss the limitations of the conducted systematic literature review:

- **Publication Bias:** The study is subject to potential publication bias, as it relies on the availability and publication of research articles related to security requirements in Android application development. Unpublished or inaccessible findings could impact the comprehensiveness of the analysis.
- **Framework Evaluation:** The review did not extensively evaluate the effectiveness of the proposed frameworks, methodologies in real-world scenarios due to limitations in article scope and availability of empirical evidence.

- **Scope Limitation:** The focus on security requirements specification leaves out other critical aspects of security such as implementation, testing, and deployment, which are integral to a comprehensive security posture.

2.6 Conclusion of SLR

The systematic literature review (SLR) embarked on a comprehensive expedition, delving into a diverse array of research papers and resources meticulously dedicated to the intricacies of security requirements identification methodologies and tools/techniques within the realm of software development. The far-reaching investigation unearthed a multitude of invaluable insights, spotlighting an assortment of preeminent approaches adroitly harnessed by the industry to masterfully elicit, scrutinize, and precisely stipulate security-oriented requirements. Among these, luminaries such as the Common Criteria (ISO/IEC 15408), SQUARE, Misuse Case Technique, Abuse Case Technique, and Security Patterns materialized as bedrock methodologies, celebrated for their efficacy in orchestrating a harmonious integration of security concerns throughout the software development lifecycle. These methods are like carefully designed plans that are put together very precisely. They provide step-by-step frameworks and helpful rules that are extremely important in dealing with security concerns as they change over time while the software keeps improving. But the valuable information doesn't end with just these methods. It also includes a wide range of tools and methods that are really useful in helping with the complex job of finding out what security needs the software has.

Standing tall among these technological allies were the formidable entities of OWASP, NERV, and NORMAP, each wielding the power to transcend mere tools and ascend to the echelons of creative catalysts. These tools, akin to the artisan's chisel, were adeptly employed for the purpose of sculpting ideas, shaping conceptual frameworks, and encouraging the emergence of security-related concepts that stood resilient against potential threats and incursions. In this expansive landscape, two particular treasures emerged - Security Requirement Patterns and Security Patterns for Mobile Applications. These beacons of innovation unfurled as reusable solutions, effectively illuminating pathways through the labyrinthine maze of common security challenges. Their significance is akin to that of guiding stars, offering both solace and direction to developers navigating the perilous seas of security concerns.

In short, the important findings from this detailed study strongly emphasize the deep significance of creating well-designed methods for identifying security needs. These methods are backed by a collection of helpful tools. They act as strong protectors in the constant battle to reduce security dangers while navigating the complex pathways of software creation. However, the spotlight shines brightest on the world of Android App Development. Here, there's a clear call for modern, flexible methods that can quickly identify security needs right from the start. This call isn't just about asking for something new, but it's a vital requirement in our ever-changing world of technology. Keeping software safe is an ongoing effort, always adapting to new innovations and the steady march of cyber threats. So, the determined quest to improve the methods we already have and exploring new ways forward becomes the guiding light for the software industry. The goal is to reach a point where security isn't an afterthought but a fundamental and essential part of the process. In this ongoing story of technological progress, the protection of software systems and user data takes top priority. As technology continues to move forward, the software industry, armed with the wisdom gained from studies like this, is ready to face new security challenges. This journey will leave a mark on the history of a safer digital world.

2.7 Research Gap

In this section the research gap has been discussed that was observed in literature. Examination was conducted on around 97 research studies various techniques, tools and methodologies were identified for security related requirements specifications. However, there were some drawbacks in the existing studies that proves to be that research gap on which further work can be done.

The gap found in the research studies was that no research was found that specifically provided framework or methodology to specify security related requirements in android app development automatically. While some of the articles such as [33], [46], [65] did include security requirements specification in android app development but were focused in one domain only either banking or defence related. Moreover, most of them required manual or semi-automated arrangement of requirements.

Hence, after SLR findings efforts are made to address the discussed research gap by implementing and Natural Language Processing Model to evaluate the Android Application Requirements and fetch security related requirements out of them. The Goal is to achieve maximum accuracy in lesser time so that it could be easily incorporated in rapid nature of android application development.

2.8 Summary

In this chapter, a Systematic Literature Review (SLR) was conducted following guidelines [8]. The review process involved three phases: planning, review, and conclusion. Research questions were formulated, databases selected, search strings created, and studies published between 2010 and 2023 were focused on. Inclusion/exclusion criteria were established, and quality assessment was performed on selected studies. The research questions included topics like security requirements identification methodologies, effectiveness of review methodologies, tools for security-related requirements in mobile app development, and challenges in current review methodologies.

The study selection process involved a comprehensive literature search across various databases. Initially, a broad search query was refined through filters, resulting in a final search phrase. Snowballing was also used to enhance results. This process yielded 97 relevant articles. A "Tollgate Approach" further refined these articles from an initial pool of 936. Articles were categorized into conference papers and journal articles, originating from various databases. Inclusion and exclusion criteria considered subject relevance, reputable sources, publication years (2010-2023), and English language. Quality assessment was conducted using a checklist with five questions, ensuring the credibility of selected studies. The distribution of articles across databases, years, and identified frameworks/methodologies was analysed. Five main categories emerged: Security Requirements Backlog, Modified User Stories, Frameworks and Methodologies, Tools Supporting Security Requirements Specification, and Hybrid Software Process Lifecycle. Among the findings, well-established methodologies like Common Criteria and SQUARE were highlighted, along with tools like OWASP and NERV. The importance of Security Requirement Patterns and Security Patterns for Mobile Applications was emphasized. The study underscores the significance of well-designed methods and tools for identifying security needs in software development. Android app development was identified as an area needing more focus on automated security requirement specification.

A research gap was observed, as no existing studies provided a framework or methodology for automatic security requirement specification in Android app development, especially across different domains. Most existing studies required manual or semi-automated processes. the aim is to address this gap by implementing a Natural Language Processing model to evaluate Android application requirements and extract security-related requirements more efficiently.

Overall, the SLR provided insights into the state of security requirement identification methodologies, tools, and challenges, paving the way for future research in the field.

CHAPTER 3

PROPOSED APPROACH

In this chapter, the proposed approach has been discussed for eliciting security-related requirement from requirements document of android application. The approach consists of several steps such as data collection, pre-processing of data, and setup of model.

The algorithm that is used in this approach is Naïve Bayes Model [106]. It is a powerful yet simple and fast model based on Bayes theorem. It is primarily used for classification tasks such as spam detection, sentiment analysis and text classification [106]. Given the nature of Android Application Development which is quite rapid and often proper requirements specification are ignored. Moreover, Android Operating System is based on Layered Architecture, consist of 5 layers known as ***Application layer***, ***Application Framework Layer***, ***Libraries & Runtime Layer***, ***Hardware Abstraction layer***, and ***Linux layer*** [107].

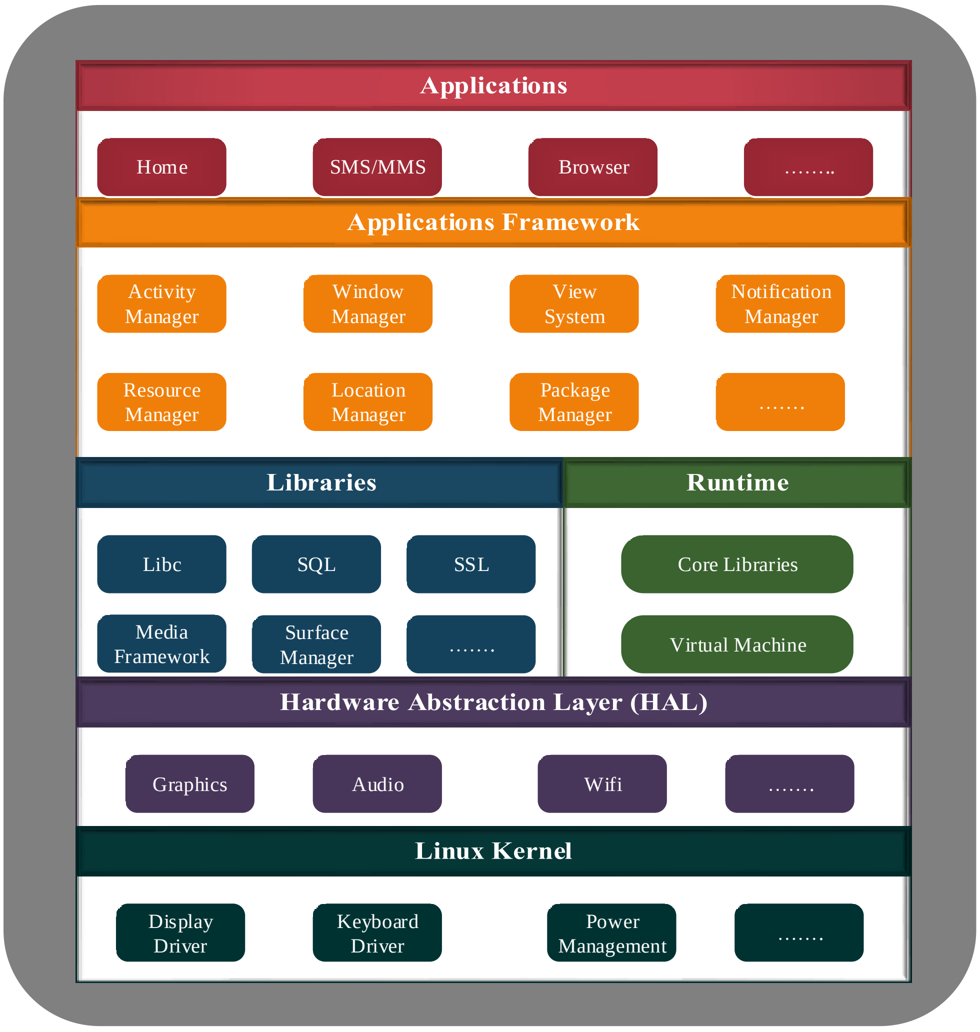


Figure 10: Android OS Architecture

Figure 10 represents the diagrammatic representation of Android Operating System Architecture. The development team must take care of this architecture and its security while creating the apps. As discussed earlier such openness and rapid development nature has severe consequences such as the major one security breaches. Having said that the security threats increase due to usage of multiple frameworks & Libraries, and APIs.

To mitigate these challenges Naïve Bayes Model has been Proposed which can be incorporated in Android Development because of its quickness and flexibility to match rapid nature of android app development.

3.1 Proposed Model

A Naive Bayes model is a probabilistic machine learning algorithm based on Bayes' theorem. It is commonly used for classification tasks, particularly in natural language processing (NLP), text analysis, and spam email detection. The "naive" aspect of Naive Bayes stems from its assumption of feature independence, which simplifies calculations and model training. It is a probabilistic classification algorithm that leverages Bayes' theorem to predict the probability of a specific class label for a given set of features. It assumes feature independence, allowing efficient calculation of conditional probabilities, and is widely applied in text classification, sentiment analysis, and spam detection [108].

The Naïve Bayes algorithm is comprised of two words Naïve and Bayes, which can be described as:

3.1.1 Naïve

It is termed "Naïve" due to its assumption that the presence of one specific feature is unrelated to the presence of other features. For example, when identifying a fruit based on attributes like colour, shape, and taste, a red, spherical, and sweet fruit is classified as an apple. Therefore, each feature independently contributes to the identification of the fruit as an apple, without relying on the presence of the others [108].

3.1.2 Bayes Theorem

Bayes theorem describes as the probability of an event given that another event has occurred. Mathematically it is written as

$$P(A|B) = (P(B|A) * P(A)) / P(B)$$

Where:

- P(A|B) represents the probability of hypothesis A being true given the evidence B.
- P(B|A) represents the probability of observing evidence B given that hypothesis A is true.
- P(A) is the prior probability of hypothesis A being true, before considering the evidence B.
- P(B) is the probability of observing evidence B.

In simple terms, Bayes' theorem allows us to calculate the probability of a hypothesis being true (the posterior probability) based on prior knowledge (prior probability) and the likelihood of observing new evidence if the hypothesis were true. It provides a formal framework for updating beliefs in light of new information.

Naïve Bayes Model consist of Multiple Algorithms, it is a group of 3 models.

Gaussian Model: The Gaussian model operates under the assumption that features within the data follow a normal distribution. In practical terms, this means that when the predictors are continuous values (as opposed to discrete), the model assumes that these values are drawn from a Gaussian distribution.

Multinomial Model: The Multinomial Naïve Bayes classifier is employed when the data exhibits a multinomial distribution. It finds extensive application in document classification tasks, where the goal is to categorize a particular document into predefined categories such as Sports, Politics, Education, and more. The classifier leverages the word frequencies as predictors to make these category assignments.

Bernoulli Model: The Bernoulli classifier operates in a manner similar to the Multinomial classifier, but it is specifically designed for predictor variables that are independent Boolean (binary) variables. In other words, it focuses on whether a particular word is present or absent in a document. The Bernoulli model is also well-known for its effectiveness in document classification tasks, particularly when considering the presence or absence of specific terms.

These three variants of the Naïve Bayes classifier - Gaussian, Multinomial, and Bernoulli, each have their unique strengths and are chosen based on the characteristics of the data and the requirements of the classification problem at hand.

3.2 Data Pre-Processing

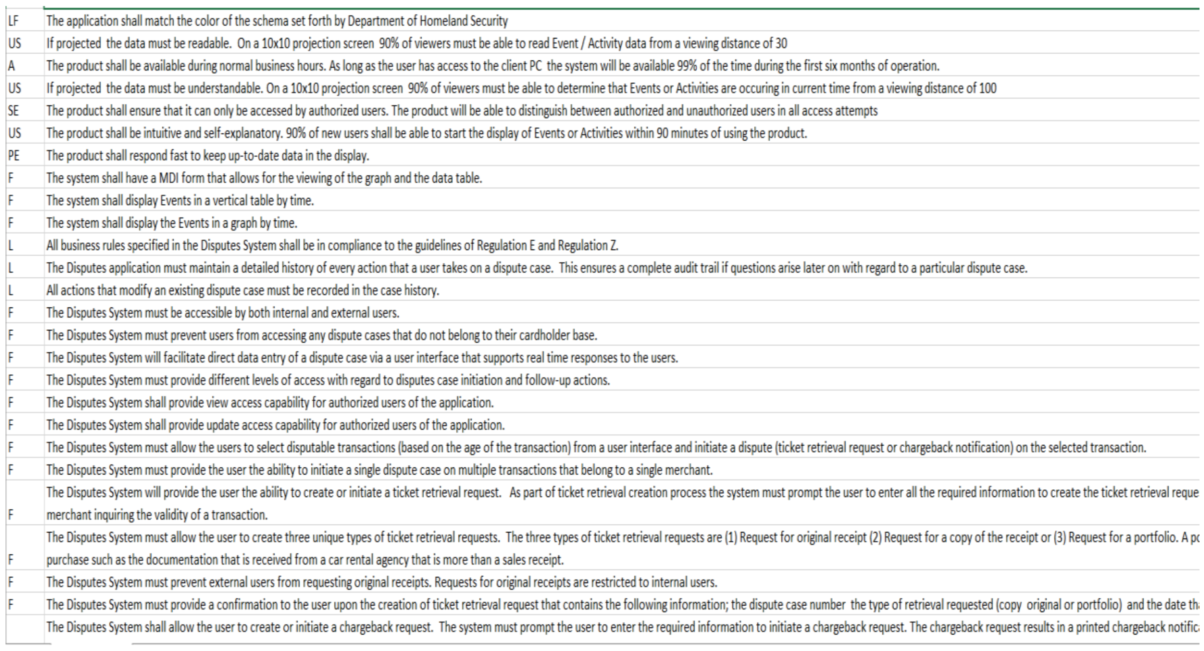
Data Pre-processing is the crucial part of preparing data for the model. In this section phases and their activities are discussed.

3.2.1 Dataset Creation

The dataset used for this model encompasses a diverse set of mobile application requirements. These requirements were compiled from multiple sources, with a portion of them being sourced from Kaggle [109], a well-known platform for data science and machine learning datasets. Additionally, the author of this project contributed a set of requirements based on their past work, enriching the dataset's diversity. Upon the completion of data collection, the dataset aggregated a substantial volume of requirements, totalling approximately 1600 entries. To facilitate the model's understanding and classification of these requirements, labels from the Kaggle dataset were assigned to the remaining entries as shown in **Figure 11**. This labelling process helps categorize the requirements into specific classes or categories, making it easier for the model to learn and make predictions.

Finally, for compatibility with the model's input requirements, the dataset was meticulously prepared and converted into the CSV (Comma-Separated Values) file format. This format is widely recognized and readable by machine learning models, enabling seamless integration with the model's training and evaluation processes.

In summary, the dataset compilation involved merging requirements from various sources, labelling to classify them, and ensuring data format compatibility to facilitate effective utilization by the model. This comprehensive dataset forms the foundation for training and testing the model's capabilities in understanding and categorizing mobile application requirements.



LF	The application shall match the color of the schema set forth by Department of Homeland Security
US	If projected the data must be readable. On a 10x10 projection screen 90% of viewers must be able to read Event / Activity data from a viewing distance of 30
A	The product shall be available during normal business hours. As long as the user has access to the client PC the system will be available 99% of the time during the first six months of operation.
US	If projected the data must be understandable. On a 10x10 projection screen 90% of viewers must be able to determine that Events or Activities are occuring in current time from a viewing distance of 100
SE	The product shall ensure that it can only be accessed by authorized users. The product will be able to distinguish between authorized and unauthorized users in all access attempts
US	The product shall be intuitive and self-explanatory. 90% of new users shall be able to start the display of Events or Activities within 90 minutes of using the product.
PE	The product shall respond fast to keep up-to-date data in the display.
F	The system shall have a MDI form that allows for the viewing of the graph and the data table.
F	The system shall display Events in a vertical table by time.
F	The system shall display the Events in a graph by time.
L	All business rules specified in the Disputes System shall be in compliance to the guidelines of Regulation E and Regulation Z.
L	The Disputes application must maintain a detailed history of every action that a user takes on a dispute case. This ensures a complete audit trail if questions arise later on with regard to a particular dispute case.
L	All actions that modify an existing dispute case must be recorded in the case history.
F	The Disputes System must be accessible by both internal and external users.
F	The Disputes System must prevent users from accessing any dispute cases that do not belong to their cardholder base.
F	The Disputes System will facilitate direct data entry of a dispute case via a user interface that supports real time responses to the users.
F	The Disputes System must provide different levels of access with regard to disputes case initiation and follow-up actions.
F	The Disputes System shall provide view access capability for authorized users of the application.
F	The Disputes System shall provide update access capability for authorized users of the application.
F	The Disputes System must allow the users to select disputable transactions (based on the age of the transaction) from a user interface and initiate a dispute (ticket retrieval request or chargeback notification) on the selected transaction.
F	The Disputes System must provide the user the ability to initiate a single dispute case on multiple transactions that belong to a single merchant.
F	The Disputes System will provide the user the ability to create or initiate a ticket retrieval request. As part of ticket retrieval creation process the system must prompt the user to enter all the required information to create the ticket retrieval reque merchant inquiring the validity of a transaction.
F	The Disputes System must allow the user to create three unique types of ticket retrieval requests. The three types of ticket retrieval requests are (1) Request for original receipt (2) Request for a copy of the receipt or (3) Request for a portfolio. A pc purchase such as the documentation that is received from a car rental agency that is more than a sales receipt.
F	The Disputes System must prevent external users from requesting original receipts. Requests for original receipts are restricted to internal users.
F	The Disputes System must provide a confirmation to the user upon the creation of ticket retrieval request that contains the following information; the dispute case number the type of retrieval requested (copy original or portfolio) and the date th
	The Disputes System shall allow the user to create or initiate a chargeback request. The system must prompt the user to enter the required information to initiate a chargeback request. The chargeback request results in a printed chargeback notific

Figure 11: Dataset Snap

3.2.2 Data Normalization

In this phase data have been cleansed from noise such as:

- Typing mistakes were Corrected
- Missing data were added
- Special characters, extra spaces were removed
- Duplication and repetitions of words were corrected.
- Outliers were removed
- Lemmatizing applied

3.2.3 Feature Selection

Another important portion of data pre-processing. For feature selection a library [110] has been used which is known as Count Vectorizer. Count Vectorizer is a feature extraction technique commonly used in natural language processing (NLP) and text analysis tasks. It's particularly useful when working with text data for tasks like text classification, sentiment analysis, and document categorization. Count Vectorizer converts a collection of text documents into a matrix of token (word) counts, which can then be used as input features for machine learning models

Here's how Count Vectorizer works and how it can be used in conjunction with a Naive Bayes classifier:

Count Vectorizer Process:

- **Tokenization:** Count Vectorizer starts by tokenizing the input text, which means breaking it down into individual words or terms. It also removes punctuation and converts text to lowercase for consistency.

- **Vocabulary Building:** It builds a vocabulary of unique words (tokens) present in the entire corpus (collection of text documents). Each word in the vocabulary is assigned a unique index.
- **Counting:** For each document in the corpus, Count Vectorizer counts the frequency of each word in the vocabulary. This results in a matrix where each row corresponds to a document, and each column corresponds to a word in the vocabulary. The matrix contains the word counts for each document.

3.3 Summary

In this chapter, an approach for extracting security-related requirements from Android application requirement documents is discussed. The approach involves multiple steps, including data collection, data pre-processing, and model setup. The chosen model is the Naïve Bayes Model, a probabilistic algorithm based on Bayes' theorem, commonly employed for tasks like spam detection and sentiment analysis. The Android Operating System's layered architecture is highlighted, emphasizing the importance of considering security in app development. To address the challenges posed by the rapid development nature and the use of multiple frameworks and APIs, the Naïve Bayes Model is proposed. Data pre-processing involves dataset creation, data normalization, and feature selection using Count Vectorizer, a technique for converting text into a matrix of token counts. These components collectively form the foundation for the model's capability to understand and categorize mobile application requirements.

CHAPTER 4

IMPLEMENTATION AND EVALUATION

This chapter focuses on the practical implementation of the proposed approach outlined in the previous chapter. The implementation process encompasses several critical components, including the setup of an Integrated Development Environment (IDE), the selection of libraries and tools, and an overview of the essential aspects of the applied approach. Additionally, this chapter delves into the preparation of results and the evaluation of the approach's performance.

4.1 Experimental Setup

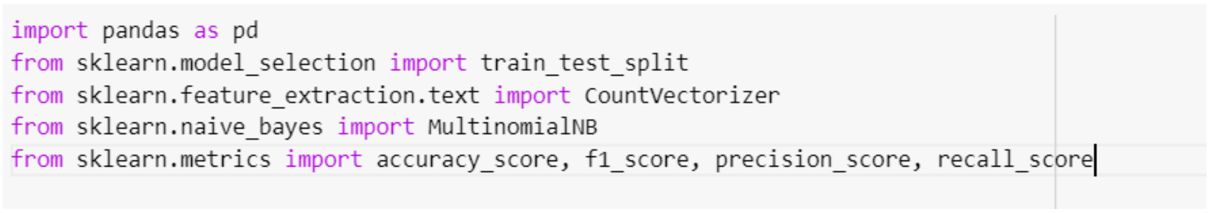
The composed methodology is conducted on online platform known as Google Colab, short of “Google Co-laboratory” [111]. It is a free cloud-based platform provided by Google that offers a hosted Jupyter notebook environment. It's primarily designed for machine learning and data analysis tasks but can be used for various other programming and research purposes as well. Here are some key features and aspects of Google Colab:

- **Jupyter Notebook Integration:** Google Colab allows users to create and run Jupyter notebooks directly in a web-based interface. Jupyter notebooks are popular in data science and research because they enable users to combine code, documentation, and visualizations in an interactive format.
- **Free Access to GPU and TPU:** One of the standout features of Google Colab is its provision of free access to Graphics Processing Units (GPUs) and Tensor Processing Units (TPUs). This is particularly beneficial for tasks that require significant computational power, such as training deep learning models.
- **Cloud-Based:** Google Colab is entirely cloud-based, meaning that you don't need to install any software or libraries on your local machine. All the computing resources are provided by Google's infrastructure.
- **Integration with Google Drive:** Colab integrates seamlessly with Google Drive, allowing you to store, access, and share your Jupyter notebooks and data files directly from your Google Drive account.
- **Pre-installed Libraries:** Google Colab comes with many popular data science libraries and tools pre-installed, including TensorFlow, PyTorch, scikit-learn, and more. This eliminates the need for manual installations.
- **Collaboration:** As the name suggests, Colab is designed for collaboration. Multiple users can work on the same notebook simultaneously, making it a useful tool for team projects and educational purposes.
- **Version Control:** Colab integrates with Git for version control, enabling users to track changes and collaborate with version control systems.
- **Sharing and Publishing:** You can easily share your Colab notebooks with others, either by providing them with a link to your notebook or by publishing them to the web. This is useful for sharing research findings, tutorials, and code with a broader audience.
- **Python Support:** Colab primarily supports Python, making it an excellent choice for Python-based data science and machine learning tasks. You can write and execute Python code within Colab notebooks.

Overall, Google Colab is a versatile and accessible platform that has gained popularity in the data science and machine learning communities due to its combination of free GPU/TPU access, Jupyter notebook integration, and collaborative features. It provides a convenient way for individuals and teams to work on data analysis, research, and machine learning projects without the need for extensive local computing resources.

4.2 Implementation

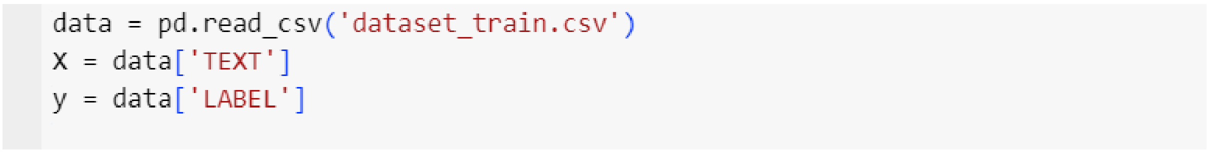
To execute the proposed approach, Python Language has been used in Google Colab. It uses the scikit-learn library for machine learning and pandas for data manipulation. This part consists of various major steps. Below has been provided a detailed explanation of each part of the code:



```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score
```

Figure 12: Importing Libraries

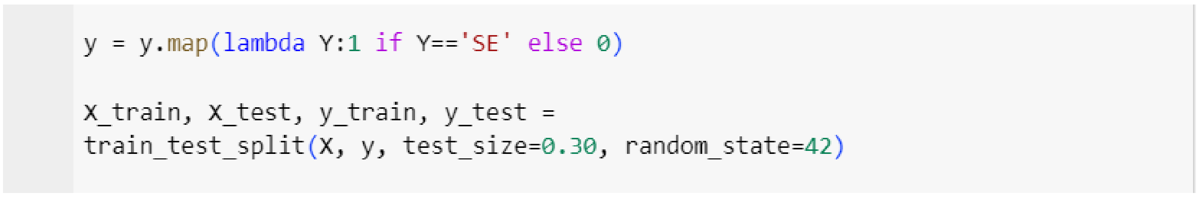
As **Figure 12** shows *Pandas* and *Sklearn* libraries has been used. Pandas is imported for data manipulation. Whereas, from Sklearn library multiple tasks has been accomplished. “Train_test_split” has been used for splitting dataset into training and testing sets. Moreover, “CountVectorizer” is for text classification and “MultinomiaNB” is the Multinomial Naïve Bayes Classifier that is used to classify the requirements into Security-Related and Non-Security-Related Requirements. Lastly, Various evaluations metrices are used to assess the calibre of the model.



```
data = pd.read_csv('dataset_train.csv')
X = data['TEXT']
y = data['LABEL']
```

Figure 13: Data Loading

Data loading step that followed by Libraries Import. As shown in **Figure 13**, The requirements dataset is composed in CSV file, that is loaded using pandas’ library. Further the dataset has been divided into two columns into X and Y. The text column of dataset that contains requirements statements are assigned to X and their respective Labels is assigned to Y variable.

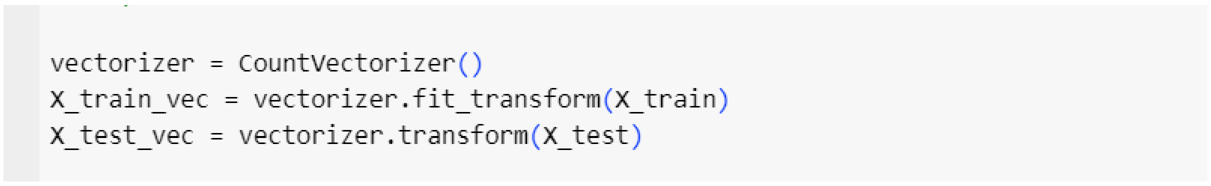


```
y = y.map(lambda Y:1 if Y=='SE' else 0)

X_train, X_test, y_train, y_test =
train_test_split(X, y, test_size=0.30, random_state=42)
```

Figure 14: Label Mapping and dataset Splitting

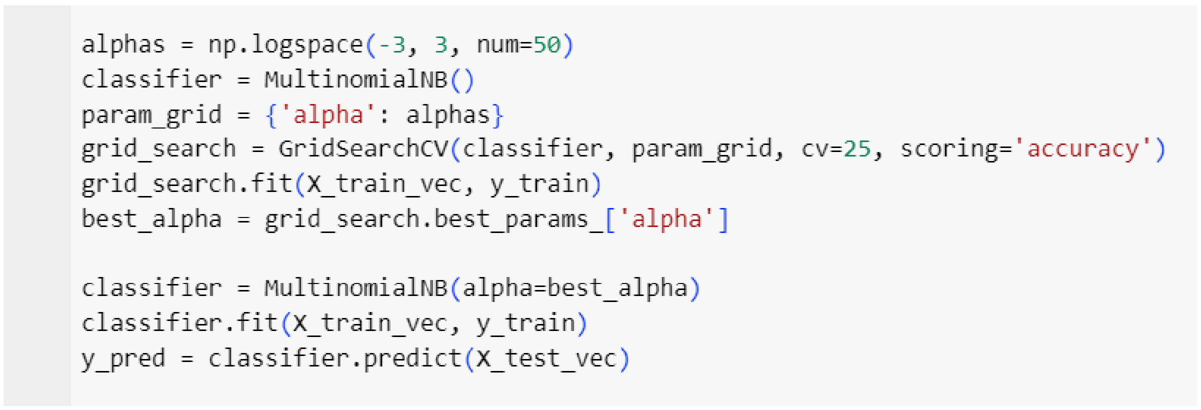
Moving further into the code, in **Figure 14** labels are mapped to binary values. As by default there are many labels such as F for Functional Requirement, SE for Security Requirement, A for Availability, U for Usability etc. to make classifier performance better and fast it is converted to binary only those requirements are assigned 1 that are Security related and for other 0 is assigned. Additionally, dataset is split into train and test group by using *train_test_split* library. Test size is taken 30% of the dataset the remaining 70% taken for training the model. Including random_state as 42 which ensure that the dataset is reproduceable.



```
vectorizer = CountVectorizer()
X_train_vec = vectorizer.fit_transform(X_train)
X_test_vec = vectorizer.transform(X_test)
```

Figure 15: Feature Setting

As **Figure 15** illustrates ***CountVectorizer*** has been used to fit and transform the dataset, as describe in earlier chapter that CounterVectorizer is used for feature selection based on number of words in a document. "X_train_vec" is the result of fitting and transforming the vectorizer on the training data, converting text documents into a matrix of token counts and "X_test_vec" is the result of transforming the test data using the same vectorizer.



```
alphas = np.logspace(-3, 3, num=50)
classifier = MultinomialNB()
param_grid = {'alpha': alphas}
grid_search = GridSearchCV(classifier, param_grid, cv=25, scoring='accuracy')
grid_search.fit(X_train_vec, y_train)
best_alpha = grid_search.best_params_['alpha']

classifier = MultinomialNB(alpha=best_alpha)
classifier.fit(X_train_vec, y_train)
y_pred = classifier.predict(X_test_vec)
```

Figure 16: Classifier Implementation

Finally, Multinomial Naïve Bayes Classifier has been implemented as seen in **Figure 16**. An Alpha parameter which is also know as smoothing technique has also been included, the best alpha was extracted by using GridSearchCV Library. The best value came out to be 0.001 has been kept so that underfitting could be mitigated. In Naïve Bayes model underfit occurs due to those event that has zero probability [108]. Moreover, this model is trained on vectorized training data. Lastly, after training the data set trained model is used to predict the test data.

4.3 Evaluation

After Predictions Model has been evaluated using various measurements, using Confusion Matrix. A confusion matrix, also known as an error matrix, is a fundamental tool for evaluating the performance of a classification algorithm, such as the Naive Bayes classifier. It provides a clear and detailed summary of how well a classification model is performing by comparing its predictions to the actual or ground truth labels. A confusion matrix is typically a square matrix with rows and columns as shown in **Table 10** corresponding to the classes in the classification problem [108].

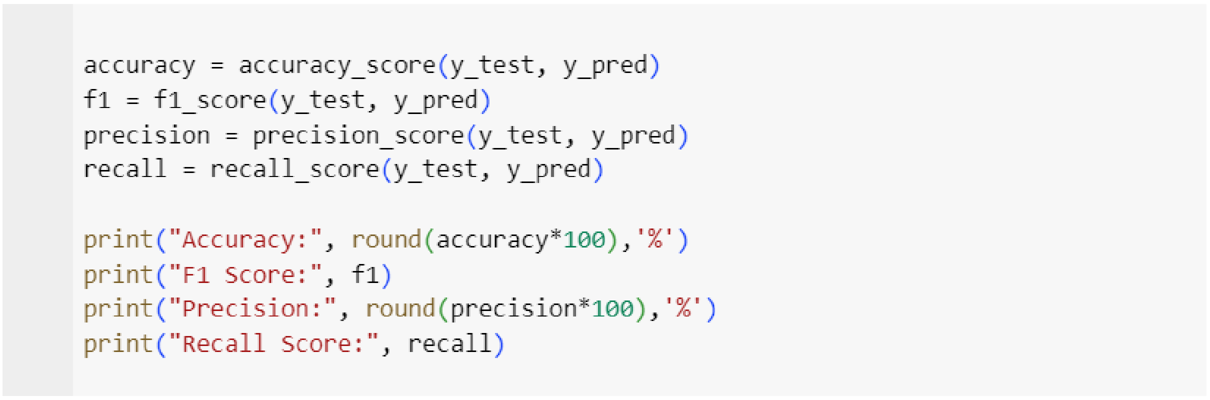
Table 10: Confusion Matrix

	Predicted Positive	Predicted Negative
Actual Positive	TP (438)	FN (1)
Actual Negative	FP (12)	TN (30)

Here's an explanation of the key components and terminology associated with a confusion matrix:

- **True Positives (TP):** The number of instances that belong to the positive class (e.g., "Security Related Requirements") and were correctly classified as positive by the model.
- **True Negatives (TN):** The number of instances that belong to the negative class (e.g., "Non-Security Related Requirements") and were correctly classified as negative by the model.
- **False Positives (FP):** The number of instances that belong to the negative class but were incorrectly classified as positive by the model. This is also known as a Type I error.
- **False Negatives (FN):** The number of instances that belong to the positive class but were incorrectly classified as negative by the model. This is also known as a Type II error.

With these parameters, other various performance indicators can be computed, such as, Accuracy, Precision, Recall and F1 score. At the end as **Figure 17** illustrates the computation of these performance measuring metrics were done in order to get results reliability.



```
accuracy = accuracy_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)

print("Accuracy:", round(accuracy*100),'%')
print("F1 Score:", f1)
print("Precision:", round(precision*100),'%')
print("Recall Score:", recall)
```

Figure 17: Model Performance Measuring

- **Accuracy:** The overall correctness of the model's predictions, calculated as (TP + TN) / (TP + FP + TN + FN). It measures the proportion of correctly classified instances out of all instances. In this model’s case it come out to be pretty well. It is **97.8%**
- **Precision:** The precision measures how many of the predicted positive instances were correctly predicted, calculated as TP / (TP + FP). It is a measure of the model's ability to avoid false positives. Our Model’s Precision calculated to be around **.97**
- **Recall (Sensitivity or True Positive Rate):** Recall measures how many of the actual positive instances were correctly predicted, calculated as TP / (TP + FN). It is a measure

of the model's ability to capture all positive instances. This proposed method scored **0.99**

- **F1-Score:** The F1-score is the harmonic mean of precision and recall and provides a balanced measure between precision and recall. It is calculated as 2 * (precision * recall) / (precision + recall). The proposed algorithm reached to about **0.97** of F1 Score.

4.4 Comparison

Lastly, comparison has been done between the proposed framework and the identified existing frameworks and methodologies in order to get better insight about the quality of newly developed novel framework. In **Table 11** below the comparison has been done between the proposed novel approach for specifying security related requirements in Android Application Development and existing techniques and frameworks that were identified in the Systematic Literature Review.

Table 11: Comparison Between Proposed Approach and State of the Art Frameworks and Techniques

Framework	Coverage	Approach	Customization & Adaptability	Cost and Resource Requirement	Scalability & Applicability	Accuracy
SR Backlog	All	Manual	Yes	Very High	Yes	-
Modified User Stories	All	Manual	Yes	Very High	Yes	-
NORMAP	Web	Automatic	Partially	High	No	87%
NERV	Web	Automatic	Yes	Very High	No	85.6%
Secure Formal Methods	Android (Banking)	Automatic	Partially	High	Yes	82%
OWASP	Web	Semi Auto	Yes	High	No	94%
Model Driven	Web & Android	Semi Auto	No	Moderate	Yes	85%
SVM, KNN	Web & Android	Automatic	No	Low	Yes	80%
NORMATIC	Web Based	Automatic	No	Moderate	No	82%
Naïve Bayes (Proposed Approach)	***Android***	***Automatic***	***Yes***	***Very Low***	***Yes***	***97%***

The **Table 11** provides a comprehensive illustration of the comparative analysis conducted between the proposed approach and the various state-of-the-art frameworks and methodologies previously identified in our meticulously conducted systematic literature review. This comparative analysis encompasses several crucial dimensions for evaluating the effectiveness and suitability of these frameworks.

First and foremost, it delves into the concept of "Coverage," shedding light on the domains within which each framework can be effectively employed. This dimension serves to elucidate the breadth and applicability of each methodology. Following this, we delve into the aspect of "Approach," discerning whether a given framework operates in a manual, semi-automatic, or fully automatic mode. This differentiation highlights the level of human intervention required, which can have significant implications for efficiency and ease of use. The third dimension, "Customization and Adaptability," provides insights into the degree of flexibility inherent in each methodology. This dimension evaluates whether these frameworks can be tailored to accommodate diverse and evolving requirements, catering to a wide array of needs. Furthermore, "Cost and Resource Requirements" offer valuable insights into the financial and resource demands of each framework. This assessment provides a practical perspective on the feasibility and sustainability of implementing these methodologies. Last but not least, we explore the dimensions of "Scalability" and "Accuracy." "Scalability" gauges a framework's ability to handle larger and more varied datasets, reflecting its potential for growth and adaptation. "Accuracy" provides a critical evaluation of the precision and reliability of predictions generated by these frameworks.

In summary, Table 11 serves as a comprehensive reference point for evaluating and comparing the proposed approach against existing state-of-the-art methodologies across multiple key dimensions, offering valuable insights for decision-making and further research in this domain.

4.5 Summary

In conclusion to this chapter, this section composed of implementation of the proposed model. Initially, the IDE setup has been discussed that online Google co lab has been opted to implement the proposed methodology. Further, moving forward the libraries that were used is briefed. Going to the next step in the chapter, the implementation of the logic has been discussed in detail, important steps of the code was shared. Finally, this chapter concludes by evaluating the implemented model by using performance evaluation metrics.

CHAPTER 5

CONCLUSION AND FUTURE DIRECTIONS

This chapter consists of the conclusion remarks about this thesis report. Here whole thesis has been summarized to provide the last insight. Moreover, few future path ways have been also shared that could pay path for Industrialists and Researchers for further research and discoveries.

5.1 Conclusion

Initially a Systematic Literature Review (SLR) was conducted, following a structured process involving planning, review, and conclusion phases. Research questions were formulated, databases were selected, and studies published between 2010 and 2023 were focused on. Inclusion and exclusion criteria were established, and quality assessment was performed on selected studies. The study categories included security requirements identification methodologies, review methodologies' effectiveness, tools for security-related requirements in mobile app development, and challenges in current review methodologies. The study selection process involved a comprehensive literature search, yielding 97 relevant articles from an initial pool of 936. These articles were categorized into conference papers and journal articles from various databases. Quality assessment was conducted to ensure study credibility. The distribution of articles across databases, years, and identified frameworks/methodologies was analysed, resulting in five main categories: Security Requirements Backlog, Modified User Stories, Frameworks and Methodologies, Tools Supporting Security Requirements Specification, and Hybrid Software Process Lifecycle. Among the findings, well-established methodologies like Common Criteria and SQUARE were highlighted, along with tools like OWASP and NERV. The importance of Security Requirement Patterns and Security Patterns for Mobile Applications was emphasized. However, a research gap was observed, as no existing studies provided a framework or methodology for automatic security requirement specification in Android app development. Most existing studies required manual or semi-automated processes.

To address this gap, an approach for extracting security-related requirements from Android application requirement documents has be proposed. This approach involves multiple steps, including data collection, data pre-processing, and model setup. The chosen model is the Naïve Bayes Model, a probabilistic algorithm based on Bayes' theorem, commonly employed for tasks like spam detection and sentiment analysis. The Android Operating System's layered architecture is highlighted, emphasizing the importance of considering security in app development. Data pre-processing involves dataset creation, data normalization, and feature selection using Count Vectorizer, a technique for converting text into a matrix of token counts. These components collectively form the foundation for the model's capability to understand and categorize mobile application requirements.

Finally, the implementation of the proposed model is discussed, including IDE setup using Google Colab and the libraries used. Important steps of the code implementation are shared, and the chapter concludes by evaluating the implemented model using performance evaluation metrics. This work provides valuable insights into security requirement identification in Android Application Development, paving the way for future research in the field.

5.2 Future Directions

As per Literature a lot of work has already been done for Web and Desktop Applications Development however, as far as Android Application Development is concerned there is a dire need for more research and development. Android Domain is yet to mature and has various security threats which was discussed in this research study. It is believed that this work will proof to be a stepping stone for further work and will help researchers and practitioners.

Moreover, this research does not stop here and intend to grow further by improving the model's capability to train and predict on larger set of data. We also have a plan to implement other feature selection tools and even evaluate non-supervised model on our dataset.

References

[1] Statista, "Statista," [Online]. Available: https://www.statista.com/statistics/271644/worldwide-free-and-paid-mobile-app-store-downloads/. [Accessed 06 2023].

[2] Statista, "Statista," [Online]. Available: https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/. [Accessed 06 2023].

[3] "Pewresearch," [Online]. Available: https://www.pewresearch.org/internet/fact-sheet/mobile/. [Accessed 06 2023].

[4] "BBC News," [Online]. Available: https://www.bbc.com/news/technology-33653472. [Accessed 06 2023].

[5] "ZDNet.," 2020. [Online]. Available: https://www.zdnet.com/article/google-removes-17-android-apps-caught-engaging-in-wap-billing-fraud/. [Accessed 06 2023].

[6] Forbes, "Google Deletes 1 Million Android Apps," https://www.forbes.com/sites/zakdoffman/2019/10/20/google-deletes-1-million-android-apps-why-you-should-be-worried/?sh=2915b4f36b59 , 2019.

[7] H. Villamizar, A. A. Neto, M. Kalinowski, A. Garcia and D. Méndez, "An Approach for Reviewing Security-Related Aspects in Agile Requirements Specifications of Web Applications," in *IEEE 27th International Requirements Engineering Conference (RE)*, 2019.

[8] B. Kitchenham, O. Brereton, D. Budgen, M. Turner, J. Bailey and S. Linkman, "Systematic literature reviews in software engineering – A systematic literature review," *Information and Software Technology,* vol. 51, no. 1, pp. 7-15, 2009.

[9] E. Mendes, K. Felizardo, C. Wohlin and M. Kalinowski, "Search Strategy to Update Systematic Literature Reviews in Software Engineering," in *45th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 2019.

[10] H. Villamizar, M. Kalinowski, M. Viana and D. M. Fernández, "A Systematic Mapping Study on Security in Agile Requirements Engineering," in *44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 2018.

[11] W. M. Farid, "The Normap methodology: Lightweight engineering of non-functional requirements for agile processes," in *2012 19th Asia-Pacific Software Engineering Conference - IEEE*, 2012.

[12] W. M. a. M. F. J. Farid, "NORMATIC: A visual tool for modeling non-functional requirements in agile processes," in *2012 Proceedings of IEEE Southeastcon - IEEE*, 2012.

[13] b. Othmane, L. a. Angin, P. a. Bhargava and Bharat, "Using assurance cases to develop iteratively security features using scrum," in *2014 Ninth International Conference on Availability, Reliability and Security,* 2014.

[14] J. Jabeen, Y. H. Motla, R. B. Mateen Ahmed Abbasi Dur-e-Benish Batool, S. Nazir and S. A. Anwer, "Incorporating artificial intelligence technique into DSDM," in *Asia-Pacific World Congress on Computer Science and Engineering*, 2014.

[15] Domah, Darshan, Mitropoulos and F. J, "The NERV methodology: A lightweight process for addressing non-functional requirements in agile software development," in *SoutheastCon 2015 - IEEE*, 2015.

[16] Sachdeva, Vaibhav, Chung and Lawrence, "Handling non-functional requirements for big data and IOT projects in scrum," in *2017 7th International Conference on Cloud Computing, Data Science \& Engineering-Confluence*, 2017.

[17] J. Mitra, Ranganath and P. Venkatesh, "SeMA: A Design Methodology for Building Secure Android Apps," *2019 34th IEEE/ACM International Conference on Automated Software Engineering Workshop (ASEW)*, pp. 19-22, 2019.

[18] Holl, Konstantin, S. A. Scherr, Elberzhager and Frank, "2018 11th International Conference on the Quality of Information and Communications Technology (QUATIC)," in *Using Scenario-Based Reading for Testing Mobile Applications with FIT4Apps*, 2018.

[19] K. Qian, R. M. Parizi and Dan Lo, "OWASP Risk Analysis Driven Security Requirements Specification for Secure Android Mobile Software Development," in *2018 IEEE Conference on Dependable and Secure Computing (DSC)*, 2018.

[20] R. Francese, C. Gravino, M. Risi, G. Scanniello and G. Tortora, "On the Use of Requirements Measures to Predict Software Project and Product Measures in the Context of Android Mobile Apps: A Preliminary Study," in *2015 41st Euromicro Conference on Software Engineering and Advanced Applications*, 2015.

[21] S. Jeong, H. Cho and S. Lee, "Agile requirement traceability matrix," in *2018 IEEE/ACM 40th International Conference on Software Engineering: Companion (ICSE-Companion)*, 2018, pp. 187-188.

[22] B. O. Emeka and S. Liu, "Security Requirement Engineering Using Structured Object-Oriented Formal Language for M-Banking Applications," in *2017 IEEE International Conference on Software Quality, Reliability and Security (QRS)*, 2017.

[23] M. Monshizadeh, V. Khatri and A. Gurtov, "NFV security considerations for cloud-based mobile virtual network operators," in *24th International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, 2016.

[24] V. Suma, B. Shubhamangala and L. M. Rao, "Impact analysis of volatility and security on requirement defects during software development process," in *International Conference on Software Engineering and Mobile Application Modelling and Development (ICSEMA 2012)*, 2012.

[25] X. Xuan, Y. Wang and S. Li, "Privacy requirements patterns for mobile operating systems," in *IEEE 4th International Workshop on Requirements Patterns (RePa)*, 2014.

[26] A. M. Alashjaee and M. Haney, "Forensic Requirements Specification for Mobile Device Malware Forensic Models," in *IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC)*, 2021.

[27] T. D. Breaux, D. Smullen and H. Hibshi, "Detecting repurposing and over-collection in multi-party privacy requirements specifications," in *IEEE 23rd International Requirements Engineering Conference (RE)*, 2015.

[28] N. A. Kilani, R. Tailakh and A. Hanani, "Automatic Classification of Apps Reviews for Requirement Engineering: Exploring the Customers Need from Healthcare Applications," in *Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS)*, 2019.

[29] W. Shin, S. Kiyomoto, K. Fukushima and T. Tanaka, "A Formal Model to Analyze the Permission Authorization and Enforcement in the Android Framework," in *IEEE Second International Conference on Social Computing*, 2010.

[30] Y. Choe, W. Choi, G. Jeon and M. Lee, "A tool for visual specification and verification for secure process movements," in *eChallenges e-2015 Conference*, 2015.

[31] A. B. Belle, T. C. Lethbridge, S. Kpodjedo, O. O. Adesina and M. A. Garzón, "A Novel Approach to Measure Confidence and Uncertainty in Assurance Cases," in *IEEE 27th International Requirements Engineering Conference Workshops (REW)*, 2019.

[32] G. Han, H. Zeng, Y. Li and W. Dou, "SAFE: Security-Aware FlexRay Scheduling Engine," in *Design, Automation Test in Europe Conference Exhibition (DATE)*, 2014.

[33] M. Nagappan and E. Shihab, "Future Trends in Software Engineering Research for Mobile Apps," in *IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, 2016.

[34] K. Qian, R. M. Parizi and D. Lo, "OWASP Risk Analysis Driven Security Requirements Specification for Secure Android Mobile Software Development," in *IEEE Conference on Dependable and Secure Computing (DSC)*, 2018.

[35] N. R. Mead and J. A. Morales, "Using malware analysis to improve security requirements on future systems," in *2014 IEEE 1st International Workshop on Evolving Security and Privacy Requirements Engineering (ESPRE)*, 2014.

[36] A. Martínez, M. Jenkins and C. Quesada-López, "Identifying implied security requirements from functional requirements," in *14th Iberian Conference on Information Systems and Technologies (CISTI)*, 2019.

[37] P. X. Mai, F. Pastore, A. Goknil and L. C. Briand, "A Natural Language Programming Approach for Requirements-Based Security Testing," in *IEEE 29th International Symposium on Software Reliability Engineering (ISSRE)*, 2018.

[38] A. Sleimi, M. Ceci, M. Sabetzadeh, L. C. Briand and J. Dann, "Automated Recommendation of Templates for Legal Requirements," in *2020 IEEE 28th International Requirements Engineering Conference (RE)*, 2020.

[39] W. Meincke, "Requirements in the loop : A computer-aided analysis of consistency, completeness, and correctness of requirements," in *2020 IEEE 28th International Requirements Engineering Conference (RE)*, 2020.

[40] A. Alzahrani and J. Feki, "Toward a Natural Language-Based Approach for the Specification of Decisional-Users Requirements," in *3rd International Conference on Computer Applications & Information Security (ICCAIS)*, 2020.

[41] E. E. Bella, S. Creff, M.-P. Gervais and R. Bendraou, "ATLaS: A Framework for Traceability Links Recovery Combining Information Retrieval and Semi-Supervised Techniques," in *IEEE 23rd International Enterprise Distributed Object Computing Conference (EDOC)*, 2019.

[42] J. Bicevskis, A. Nikiforova, Z. Bicevska, I. Oditis and G. Karnitis, "A Step Towards a Data Quality Theory," in *Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS)*, 2019.

[43] F. Gilson, M. Galster and F. Georis, "Extracting Quality Attributes from User Stories for Early Architecture Decision Making," in *IEEE International Conference on Software Architecture Companion (ICSA-C)*, 2019.

[44] A. Sleimi, N. Sannier, M. Sabetzadeh, L. Briand and J. Dann, "Automated Extraction of Semantic Legal Metadata using Natural Language Processing," in *IEEE 26th International Requirements Engineering Conference (RE)*, 2018.

[45] T. Li, "Identifying Security Requirements Based on Linguistic Analysis and Machine Learning," in *24th Asia-Pacific Software Engineering Conference (APSEC)*, 2017.

[46] R. Malhotra, A. Chug, A. Hayrapetian and R. Raje, "Analyzing and evaluating security features in software requirements," in *International Conference on Innovation and Challenges in Cyber Security (ICICCS-INBUSH)*, 2016.

[47] D. C. d. Leon and S. Shrestha, "Requirements are the New Code," in *IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*, 2016.

[48] M. Ruchika, C. Anuradha, H. Allenoush and R. Rajeev, "Analyzing and evaluating security features in software requirements," in *International Conference on Innovation and Challenges in Cyber Security (ICICCS-INBUSH)*, 2016.

[49] M. Riaz, J. King, J. Slankas and L. Williams, "Hidden in plain sight: Automatically identifying security requirements from natural language artifacts," in *IEEE 22nd International Requirements Engineering Conference (RE)*, 2014.

[50] Robin, A., Gandhi, Mariam and Rahmani, "Early security patterns: A collection of constraints to describe regulatory security requirements," in *Second IEEE International Workshop on Requirements Patterns (RePa)*, 2012.

[51] L. a. A. P. a. W. H. a. B. B. ben Othmane, "Extending the agile development process to develop acceptably secure software," *IEEE Transactions on dependable and secure computing,* vol. 11, no. 6, pp. 497-509, 2014.

[52] H. a. L. M. I. a. A. H. a. R. M. a. A. T. a. S. B. {Dar, "A Systematic Study on Software Requirements Elicitation Techniques and its Challenges in Mobile Application Development," *IEEE Access,* vol. 6, pp. 63859-63867, 2018.

[53] H. a. L. Dar, M. I. a. Ashraf, H. a. Ramzan, M. a. Amjad, T. a. Shahzad and Basit, "A Systematic Study on Software Requirements Elicitation Techniques and its Challenges in Mobile Application Development," *IEEE Access,* vol. 6, pp. 63859-63867, 2018.

[54] I. U. Haq and T. A. Khan, "Penetration Frameworks and Development Issues in Secure Mobile Application Development: A Systematic Literature Review," *IEEE Access,* vol. 9, pp. 87806-87825, 2021.

[55] M. Narouei, H. Takabi and R. Nielsen, "Automatic Extraction of Access Control Policies from Natural Language Documents," *Transactions on Dependable and Secure Computing,* vol. 17, no. 3, pp. 506-517, 2020.

[56] M. Riaz, J. King, J. Slankas and L. Williams, "Hidden in plain sight: Automatically identifying security requirements from natural language artifacts," *IEEE 22nd International Requirements Engineering (RE),* 2014.

[57] J. Slankas and L. Williams, "Automated extraction of non-functional requirements in available documentation," *1st International Workshop on Natural Language Analysis in Software Engineering (NaturaLiSE),* 2013.

[58] Hanan, Hibshi, Stephanie, Jones, Travis and Breaux, "A Systemic Approach for Natural Language Scenario Elicitation of Security Requirements," *IEEE Transactions on Dependable and Secure Computing,* 2021.

[59] I. K. Raharjana, D. Siahaan and C. Fatichah, "User Stories and Natural Language Processing: A Systematic Literature Review," *IEEE Access*, vol. 9, pp. 53811 - 53826, 2021.

[60] Maria, R. Esteves, R. Jr, L. Antonio, Pinto and N. Alves, "ScrumS: a model for safe agile development," in *Proceedings of the 7th International Conference on Management of computational and collective intElligence in Digital EcoSystems*, 2015.

[61] R. Lutz, S. Schäfer and S. Diehl, "2012 Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering," in *298-301,* 2012.

[62] K. Thomas, A. K. Bandra, B. A. Price and B. Nuseibah, "Distilling Privacy Requirements for Mobile Applications," in *Proceedings of the 36th International Conference on Software Engineering ICSE,* 2014.

[63] W. Guo, "Management system for secure mobile application development," in *Proceedings of the ACM Turing Celebration Conference*, China, 2019.

[64] J. Mitra and V.-P. Ranganath, "SeMA: A Design Methodology for Building Secure Android Apps," in *34th IEEE/ACM International Conference on Automated Software Engineering Workshop (ASEW),* 2019.

[65] P. X. Mai, F. Pastore, A. Goknil and L. C. Briand, "MCP: A Security Testing Tool Driven by Requirements," in *ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion),* 2019.

[66] S. Amasaki and P. Leelaprute, "The Effects of Vectorization Methods on Non-Functional Requirements Classification," in *44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA),* 2018.

[67] A. C. J. Fox, G. Stockwell, S. Xiong, H. Becker, D. P. Mulligan, G. Petri and N. Chong, "A Verification Methodology for the Arm® Confidential Computing Architecture: From a Secure

Specification to Safe Implementations," *Proceedings of ACM on Programming Languages,* vol. 7, no. 88, pp. 376-405, 2023.

[68] L. Zhao, W. Alhoshan, A. Ferrari, K. J. Letsholo, M. A. Ajagbe, E. V. Chioaşcă and R. T. Batista-Navarro, "Natural Language Processing for Requirements Engineering: A Systematic Mapping Study," *ACM Computing Surveys,* vol. 54, no. 3, pp. 1-41, 2021.

[69] Wagner, S. Fern\'{a}ndez, D. M. Felderer, M. Vetr\`{o}, A. Kalinowski, M. Wieringa, R. Pfahl, D. Conte, T. Christiansson, M.-T. Greer, D. Lassenius and etl, "Status Quo in Requirements Engineering: A Theory and a Global Family of Surveys," *ACM Transactions on Software Engineering and Methodology,* vol. 28, no. 2, pp. 1-48, 2019.

[70] T. Lopez, H. Sharp, A. Bandara, T. Tun, M. Levine and B. Nuseibeh, "Security Responses in Software Development," *ACM Transactions on Software Engineering and Methodology,* vol. 32, no. 3, pp. 1-29, 2023.

[71] R. A. Gandhi and S. W. Lee, "Discovering Multidimensional Correlations among Regulatory Requirements to Understand Risk," *ACM Transactions on Software Engineering and Methodology,* vol. 20, no. 4, pp. 1-37, 2011.

[72] I. A. Tøndel and M. G. Jaatun, "Towards a Conceptual Framework for Security Requirements Work in Agile Software Development," *International Journal of Systems and Software Security and Protection,* vol. 11, no. 1, pp. 33-62, 2020.

[73] D. Angermeier, H. Wester, K. Beilke, G. Hansch and J. Eichler, "Security Risk Assessments: Modeling and Risk Level Propagation," *ACM Transactions on Cyber-Physical Systems,* pp. 1-25, 2022.

[74] A. Singhal and Sonia, "Development of agile security framework using a hybrid technique for requirements elicitation," in *International Conference on Advances in Computing, Communication and Control,* 2011.

[75] Behutiye, W. a. Karhap{\"a}{\"a}, Pertti, D. Costal, Oivo, Markku, Franch and Xavier, "Non-functional requirements documentation in agile software development: challenges and solution proposal," in *International conference on product-focused software process improvement,* 2017.

[76] E. H. Marinho and R. F. Resende, "Quality Factors in Development Best Practices for Mobile Applications," in *International Conference on Computational Science and Its Applications,* 2012.

[77] N. Yusop, M. Kamalrudin, S. Sidek and J. Grundy, "Automated Support to Capture and Validate Security Requirements for Mobile Apps," in *Asia Pacific Requirements Engineering Conference,* 2016.

[78] V. H. Subburaj and J. E. Urban, "Specifying Security Requirements in Multi-agent Systems Using the Descartes-Agent Specification Language and AUML," *Information technology for management: Emerging research and applications,* pp. 93-111, 2018.

[79] L. Ren, R. Chang, Q. Yin and Y. Man, "A Formal Android Permission Model Based on the B Method," *Security, Privacy, and Anonymity in Compution, Communication and Storage,* vol. 10656, pp. 381-394, 2017.

[80] B. Hasan and J. M. Gómez, "Security Framework for Adopting Mobile Applications in Small and Medium Enterprises," *ICETE. Communications in Computer and Information Science,* vol. 764, pp. 75-98, 2017.

[81] R. Nacheva, S. Sulova and B. Penchev, "Where Security Meets Accessibility: Mobile Research Ecosystem," *Communications in Computer and Information Science,* vol. 15, no. 29, pp. 2016-231, 2022.

[82] A. Souag, R. Mazo, C. Salinesi and I. Comyn-Wattiau, "Reusable knowledge in security requirements engineering: a systematic mapping study," *Requirements Engineering,* vol. 21, pp. 251-283, 2016.

[83] S. H. Houmb, S. Islam, E. Knauss, J. Jürjens and K. Schneider, "Eliciting security requirements and tracing them to design: an integration of Common Criteria, heuristics, and UMLsec," *Security Requirements Engineering,* vol. 15, no. 1, pp. 63-93, 2010.

[84] K. Chatterjee, D. Gupta and A. De, "A framework for development of secure software," *CSI Transactions on ICT,* vol. 1, pp. 143-157, 2013.

[85] M. I. Daud, "Secure software development model: A guide for secure software life cycle," in *Proceedings of the international MultiConference of Engineers and Computer Scientists - Citeseer*, 2010.

[86] K. a. H. S. a. L. V. Rindell, "Securing Scrum for VAHTI.," in *SPLST*, 2015.

[87] M. H. Diallo, J. Romero-Mariona, S. E. Sim, T. A. Alspaugh and D. J. Richardson, "A Comparative Evaluation of Three Approaches to Specifying Security Requirements," in *12th Working Conference on Requirements Engineering: Foundation for Software Quality, Luxembourg*, 2016.

[88] E. P. Kukula, F. R. Shaw, R. Wallner, A. Breckenkamp, G. Kiebuzinski, L. Nadel and P. Wolfhope, "Use case mobile biometric testing amp; evaluation: A framework to cross-reference requirements and test methods," in *IEEE Conference on Technologies for Homeland Security (HST)*, 2012.

[89] R. Kumar, A. Baz, H. Alhakami, W. Alhakami, A. Agrawal and R. A. Khan, "A hybrid fuzzy rule-based multi-criteria framework for sustainable-security assessment of web application," *Ain Shams Engineering Journal,* vol. 12, no. 2, pp. 2227-2240, 2021.

[90] M. Katarahweire, E. Bainomugisha and K. A.Mughal, "Data Classification for Secure Mobile Health Data Collection Systems," *Development Engineering,* vol. 5, 2020.

[91] C. Cobb, S. Sudar, N. Reiter, R. Anderson, F. Roesner and T. Kohno, "Computer security for data collection technologies," *Development Engineering,* vol. 3, pp. 1-11, 2018.

[92] Y. Hanif and H. S. Lallie, "Security factors on the intention to use mobile banking applications in the UK older generation (55+). A mixed-method study using modified UTAUT and MTAM - with perceived cyber security, risk, and trust," *Technology in Society,* vol. 67, pp. 10-16, 2021.

[93] M. Shamsujjoha, JohnGrundy, L. li, HouriehKhalajzadeh and QinghuaLu, "Developing Mobile Applications Via Model Driven Development: A Systematic Literature Review," *Information and Software Technology,* vol. 140, pp. 1-25, 2021.

[94] F. Ebrahimi, M. Tushev and A. Mahmoud, "Mobile app privacy in software engineering research: A systematic mapping study," *Information and Software Technology,* vol. 133, pp. 1-17, 2021.

[95] J. Singh, M. Wazid, A. K. Das, VinayChamola and M. Guizani, "Machine learning security attacks and defense approaches for emerging cyber physical applications: A comprehensive survey," *Computer Communications,* vol. 192, no. 10, pp. 316-331, 2022.

[96] I. a. A. Z. a. J. S. R. Ghani, "Integrating software security into agile-Scrum method," *KSII Transactions on Internet and Information Systems (TIIS) - Korean Society for Internet Information,* vol. 8, no. 2, pp. 646-663, 2014.

[97] D. a. M. S. N. F. a. M. A. M. Mougouei, "S-scrum: a secure methodology for agile development of web services.," *{World of Computer Science & Information Technology Journal,* vol. 3, no. 1, 2013.

[98] Younas, M. a. Jawawi, D. a. Ghani, I. a. Kazmi and R, "Non-functional requirements elicitation guideline for agile methods," *Journal of Telecommunication, Electronic and Computer Engineering (JTEC),* 2017.

[99] A. A. Saima Zareen and S. A. Kham, "Security Requirements Engineering Framework with BPMN 2.0.2 Extension Model for Development of Information Systems," *Multidisciplinary Digital Publishing Institute (MDPI),* vol. 10, no. 14, 2020.

[100] D. Nyambo, Z. Yonah and C. Tarimo, "An Approach for Systematically Analyzing and Specifying Security Requirements for the Converged Web-Mobile Applications," *International Journal of Computing and Digital Systems,* vol. 3, no. 3, p. 13, 2014.

[101] M. Ongtang, S. McLaughlin, W. Enck and P. McDaniel, "Semantically rich application-centric security in Android," *Security and Communications Networks - Willey Library,* vol. 6, pp. 658-673, 2012.

[102] N. Yusop, M. Kamalrudin, M. M. Yusof and S. Sidek, "Meeting Real Challenges in Eliciting Security Attributes for Mobile Application Development," *Journal of Internet Computing and Services,* vol. 17, no. 5, pp. 25-32, 2016.

[103] N. Yusop, M. Kamalrudin and S. Sidek, "SECURITY REQUIREMENTS VALIDATION FOR MOBILE APPS: A SYSTEMATIC LITERATURE REVIEW," *Journal Teknologi (UMT) Science and Technogoly,* vol. 77, no. 33, 2015.

[104] H. Shariff and M. Y. Said, "Non-Functional Requirement Detection Using Machine Learning and Natural Language Processing," *Turkish Journal of Computer and Mathematics Education,* vol. 12, no. 3, pp. 2224-2229, 2021.

[105] H. Kahtan, M. Abdulhak, A. S. Al-Ahmad and Y. I. Alzoubi, "A model for developing dependable systems using a component-based software development approach (MDDS-CBSD)," *The Institution of Engineering and Technology, IET Software,* vol. 17, no. 1, pp. 76-92, 2023.

[106] G. f. Geeks, "Geeks for Geeks," [Online]. Available: https://www.geeksforgeeks.org/naive-bayes-classifiers/. [Accessed July 2023].

[107] "Geeks for Geeks," [Online]. Available: www.geeksforgeeks.org/android-system-architecture. [Accessed Jul 2023].

[108] [Online]. Available: https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/. [Accessed July 2023].

[109] "Kaggle," [Online]. Available: https://www.kaggle.com/datasets/iamsouvik/software-requirements-dataset. [Accessed June 2023].

[110] "Scikit Learn," [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html. [Accessed July 2023].

[111] Google, "Google Colab," [Online]. Available: https://colab.research.google.com/. [Accessed June 2023].