Identifying End-User Challenges and Mitigation Strategies in Software Ecosystems: A Large-Scale Empirical Study on User Feedback

by

Bachan Ghimire
B.Sc., London Metropolitan University, 2018
M.Sc., Collegium Humanum - Warsaw Management University, 2021

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

in the Department of Computer Science

Identifying End-User Challenges and Mitigation Strategies in Software Ecosystems:
A Large-Scale Empirical Study on User Feedback

by

Bachan Ghimire
B.Sc., London Metropolitan University, 2018
M.Sc., Collegium Humanum - Warsaw Management University, 2021

Supervisory Committee

Dr. Daniela Damian, Supervisor
(Department of Computer Science)

Dr. Neil Ernst, Departmental Member
(Department of Computer Science)

ABSTRACT

Objectives: The aim of this thesis is to analyze and articulate end-user challenges and pain points in Software Ecosystems (SECOs) based on user feedback. The problems faced by end-users in SECOs and the methods to address them have not been studied in existing research. The objectives include understanding developer responses to feedback, examining the growth of SECO reviews over time, and providing strategies to mitigate user challenges in SECOs.

Methods: Over 2.4 million user reviews were scraped from 283 ecosystem platforms in app stores and review websites. Among these, over 40,000 relevant reviews from 139 platforms were classified as "SECO reviews" through manual pair coding and automated techniques. Subsequently, a training dataset of 5,000 SECO reviews was labeled as problem categories identified. An XGBoost Machine Learning classifier was trained using this dataset to categorize SECO reviews into six distinct problem categories with an accuracy of 93%. Negative reviews were identified using sentiment analysis, and TF-IDF and Chi-Squared analysis were employed to extract features associated with each problem category. Finally, a thematic analysis was conducted on interviews with platform owners using a semi-structured interview approach to gather mitigation strategies.

Results and Recommendations: Six categories of challenges in SECOs were identified: Integration, Customer Support, Design & Complexity, Privacy & Security, Cost & Pricing, and Performance & Compatibility. Each category presented its own set of pain points, encompassing a wide range of issues experienced by users. Examples include dysfunctional API errors, user preference for live customer support instead of chat-based interactions, user switching to competitors due to interface complexity, concerns about scams and fake reviews in marketplaces, unexpected charges to bank accounts, and device-specific and operating system-specific compatibility issues. The findings show that developers respond to SECO reviews less frequently compared to general reviews and SECO-related reviews have exhibited significant growth in the past five years, with the integration demands heightened by the COVID-19 pandemic. To effectively address the challenges faced by end-users and developers, platforms should adopt an API-first mentality, foster a supportive community, implement stringent vetting processes in marketplaces, employ live feedback tracking tools, consider developing cross-platform software, establish comprehensive documentation and guidelines, and maintain transparent policies in user data management.

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

I would like to extend my heartfelt gratitude to my supervisor Dr. Daniela Damian for all the encouragement, funding, moral support, and guidance in writing this thesis and throughout my grad school.

Thanks to my dear friends Ami, Joao, Jordan, Gabi & Neha for tolerating me! Thanks to my colleague Zhe Shi Li (Zane) for his help with research design and my friend Padmanabh Chaturvedi (Paddy) for helping me out with pair coding of a dataset during the research. Extended thanks to all members of the SEGAL Lab.

Thanks also to my father Dr. Dhruba R. Ghimire and mother Rashmi Ghimire for the love and prayers.

Honorable mention of thanks to the professors Dr. Daniel German who taught me how to deal with massive datasets and Dr. Charles Perin who taught me how to create awesome visualizations.

DEDICATION

To my late elder brother Ruben Ghimire, and future students looking forward to writing a thesis/starting gradschool.

Chapter 1

Introduction

1.1 Background

Over the last decade, there has been a significant change in the way software companies function and use platforms as a type of open innovation to expand their markets, and have seen a significant increase in their revenue [37]. These platforms serve as the foundation for creating Software Ecosystems (SECO)s, where the platform provider, also known as the keystone organization, collaborates and innovates with other software vendors [38, 54]. Software ecosystems are complex and dynamic systems, consisting of various software components, platforms, and developers that interact with each other [38]. Software ecosystems have become an integral part of modern software development, with various stakeholders collaborating to develop, maintain, and evolve a complex software system. Companies such as HubSpot, Salesforce, Xero, Slack, Shopify, and Wix have thrived from their integration, marketplace, innovation, and other qualities that make a thriving ecosystem [73]. User feedback plays a crucial role in identifying issues and improving the quality of the system. Developers rely on this feedback to make informed decisions and prioritize their actions [57]. However, identifying common issues faced by end-users of software ecosystems and exploring developers' attempts in addressing them remains unexplored in software ecosystem research.

Various operating system-specific application stores, marketplaces, public review websites, and keystone platforms like Shopify, which often have their own stores provide user feedback in the form of reviews [39]. User feedback is a valuable source of information that can aid in enhancing software quality and user satisfaction. In

recent times, there has been a growing inclination towards examining user reviews to extract insightful knowledge about software products and recognize areas for improvement. Although there have been several previous studies to identify issues and user concerns through mining user reviews, my thesis explicitly aims to scrutinize software ecosystem-specific reviews as analysis of ecosystems and their data analytics remains a challenge in software ecosystems [66].

Analyzing software ecosystem reviews is difficult due to multiple feedback channels and the presence of third-party applications. It can be hard to distinguish if feedback is for a single partner application, multiple applications, or the core platform [25]. Platform providers must rely on partners to gather feedback and make it accessible. The distinction between the core product and partner apps might become unclear, making it challenging for end users to provide feedback and platforms to analyze feedback [43].

1.2 Research Objectives

The objective of this research is to identify, categorize and analyze the common types of Software Ecosystem (SECO) issues in user feedback and the corresponding responses of developers to these issues. It also aims to examine how SECO-related feedback changes over time and how platform/keystone businesses react to this feedback. Recognizing the most common SECO problems and how they are addressed can assist developers and platform/keystone organizations in improving the quality and security of their SECOs, resulting in better user experiences and higher trust in their products. Additionally, the thesis aims to examine the general sentiment of reviews and compare that with the sentiment of developer responses to determine whether there are any differences in the way developers and users perceive the same issues. Also, by understanding how Software Ecosystem-related feedback evolves, developers and platform/keystone organizations can stay ahead of potential issues and proactively address them. The thesis intends to enhance the overall quality and user experience of SECOs, which are becoming increasingly significant in the software industry due to the dependence of many software products on third-party components and services.

Furthermore, the thesis investigates how organizations in charge of software ecosystems respond to user feedback concerning SECO problems. It aims to provide several recommendations on what strategies can be opted to mitigate challenges and problems

faced by not only end-users but also the third-party developers of software ecosystems. The recommendations can be useful for emerging and established software platforms. The topic is motivated by the critical role platform/keystone organizations play in managing and sustaining software ecosystems, and their responsiveness to user input can significantly impact the overall quality and user satisfaction of the ecosystem [38, 73, 11]. The research aims to provide insights into the practices in place for managing and addressing SECO concerns by analyzing the tactics and techniques used by these organizations in handling user complaints. The thesis can assist in identifying areas where these businesses can improve their responsiveness to customer input, ultimately enhancing the overall quality and user experience of SECOs.

Overall, this research aims to improve software ecosystem quality and user experience by comprehensively analyzing user reviews on a large scale and by speaking to platform developers. To achieve these objectives, the thesis answers the following four research questions:

- **RQ1:** What are the different problems faced by end-users in software ecosystems?
- **RQ2:** How does the sentiment of developer responses and user feedback compare in software ecosystems?
- **RQ3:** How has the proportion of software ecosystem reviews changed over time?
- **RQ4:** What are the recommended strategies for mitigating challenges encountered in software ecosystems?

1.3 Research Contributions

Upon answering the research questions, the thesis provides several contributions. Firstly, it unveils a significant methodological contribution in the realm of review classification by proposing an approach that effectively distinguishes Software Ecosystem (SECO) reviews from non-SECO reviews. This methodology combines pair coding with Cohen Kappa's coefficient to achieve high inter-rater agreement, augmented by the manual validation of SECO-related keywords and contextual cues for precise categorization. The significance of this contribution is underscored by its potential

applications in recommender systems and future research endeavors. The thesis introduces six areas of software ecosystem feedback and provides an array of discussion topics and feedback for each category. Additionally, it also reveals how SECO-related feedback has grown over time which shows the increasing need for studies in this space. It then sheds light on the extent and general sentiment of developer response to user feedback. Furthermore, it investigates how developers or owners of the platform ensure that these problems do not go unnoticed, and examines how the issues are being addressed in the industry. Based on the findings from the interviews, the thesis provides seven strategies for mitigating challenges faced by actors in the software ecosystem. The thesis' two-fold design will significantly contribute to the existing knowledge of end-user concerns and the industrial perspective on software ecosystems and can guide platforms in designing and fostering better ecosystems.

Based on the research, several recommendations were proposed to enhance SECO. These recommendations focus on adopting an API-first mentality, prioritizing API development, and improving integration processes to address user and developer needs effectively. The research also highlighted the importance of actively engaging the user community, supporting developers, and fostering collaboration to mitigate end-user problems and improve the overall user experience. In terms of security, the research shed light on the challenges faced by SECO users, such as concerns about scams, fake reviews, data mining, and authentication issues. The recommendations emphasize implementing strict vetting processes, continuously monitoring third-party apps, and providing developer support to mitigate these security risks and ensure a safe and user-friendly environment. The thesis emphasizes the significance of design-related issues and their impact on user satisfaction. The recommendations include improving mobile responsiveness, creating intuitive interfaces, minimizing in-app ads, and addressing performance issues to retain users and prevent them from switching to competitor platforms. It highlights the importance of transparent policies, efficient incident response processes, and user privacy to foster trust and mitigate risks associated with data breaches. It emphasizes the need for platforms to establish clear guidelines, maintain transparent communication channels, and adhere to relevant regulations. Overall, the research contributions have provided valuable insights into the software ecosystem, enabling platform owners and developers to make informed decisions and improvements. The recommendations aim to enhance the user experience, streamline integration processes, ensure security and privacy, and ultimately contribute to the success and growth of the software ecosystem.

1.4 Challenges

The problem space related to identifying and analyzing problems faced by end-users in software ecosystems is riddled with numerous difficulties that need to be addressed. Firstly, the unavailability of a labeled dataset poses a significant challenge in accurately categorizing user reviews. Without labeled data, it becomes difficult to train machine learning algorithms to distinguish between software ecosystem-related reviews and those that do not pertain to software ecosystems. This challenge is addressed in the thesis by proposing a novel approach that combines pair coding with Cohen Kappa's coefficient to achieve high inter-rater agreement, augmented by the manual validation of SECO-related keywords and contextual cues for precise categorization.

Secondly, distinguishing between reviews that are relevant to the software ecosystem and those that are not is also problematic. For instance, a user may write a review regarding a feature-specific bug, but this issue may stem from an integration-induced problem. Although the complaint is relevant to the software ecosystem, it may not be identified as such since it remains primarily a feature-related problem from the user's perspective. Thus, it may not meet the criteria for inclusion in the subset of software ecosystem-related reviews. The thesis addresses this challenge by introducing six areas of software ecosystem feedback and providing an array of discussion topics and feedback for each category.

Next, to ensure comprehensive coverage of software ecosystem-related concerns, a wide range of topics following established software ecosystem taxonomies must be captured. This implies that there is a need for a robust methodology to ensure that all software ecosystem-related concerns are identified and included in the analysis. However, this poses a significant challenge as different software ecosystems have different characteristics and features, and there is no single taxonomy that fits all software ecosystems. The thesis addresses this challenge by providing seven strategies for mitigating challenges faced by actors in the software ecosystem.

Lastly, user reviews are known to vary significantly in terms of writing style, which can lead to errors in classification. Different users have different writing styles, and this can cause problems when attempting to categorize reviews into relevant subsets. However, all these challenges are addressed in this thesis through its two-fold design which significantly contributes to the existing knowledge of end-user concerns and the industrial perspective on software ecosystems.

1.5 Concepts and Definitions

Based on the reviewed literature of Software Ecosystems [54, 38, 39], I will be using the following definitions and concepts that are relevant to the research presented in the thesis.

Software Ecosystem: A software ecosystem is a collection of software components, tools, and services that interact with each other to create a complete software solution. It comprises different actors, such as software developers, third-party developers, end-users, and keystone organizations.

Keystone Organization: A keystone organization is an entity in a software ecosystem that plays a pivotal role in driving the development and growth of the ecosystem. It is a central organization that provides the necessary support and resources to facilitate the development and growth of the ecosystem. Examples of keystone organizations in the software ecosystem include Microsoft and Google.

Third-Party Developer: A third-party developer is an independent developer or a company that creates software applications or components that integrate with existing software products. They are not part of the keystone organization but play a crucial role in expanding the functionality of the software ecosystem.

End-Users: End-users are the individuals or organizations that use the software products and applications developed by software ecosystem actors. They are the ultimate consumers of the software and are the reason why the software ecosystem exists.

User Feedback: User feedback is the information provided by end-users about the software products they use. This feedback is essential for software developers and third-party developers to improve their products and enhance the user experience. User Feedback has also been referred to as 'Reviews' or 'User Reviews' which in this context means written feedback as a form of review.

App Stores: An app store is a digital marketplace where users can download and install software applications for their devices. App stores play a crucial role in the software ecosystem as they provide a platform for developers to distribute and monetize their software products.

1.6 Thesis Outline

Chapter 1: Introduction The opening section of the thesis provides a summary of the topic, which includes definitions and the research questions. It also presents the open problem that will be addressed, along with its context, impact, and overall motivation for the research. The section concludes with an outline of the thesis structure.

Chapter 2: Literature review This chapter provides an overview of related work in software ecosystems and user reviews mining. It discusses previous research studies, methods used to collect and analyze data, and highlights gaps in the existing literature that this thesis aims to address. The review serves as a foundation for the thesis's research questions and methodology.

Chapter 3: Methodology The third chapter describes the research methodologies used in this thesis, including quantitative and qualitative data collection and analysis techniques. It outlines the SECO data set, data collection process, and preprocessing steps to ensure data quality. The chapter also describes sentiment analysis techniques used to compare developer responses and user feedback, and methods used to analyze semi-structured interviews with platform owners.

Chapter 4: Findings and Analysis The fourth chapter will present the findings of the thesis. The chapter will provide a detailed analysis of the SECO user reviews. It will answer all of the research questions one by one. The analysis will include the most common problems faced by end-users in software ecosystems, the sentiment of developer responses and user feedback, the changes in the proportion of software ecosystem reviews over time, and strategies to mitigate problems in SECO based on how keystone organizations address user feedback.

Chapter 5: Discussion The fifth chapter reiterates the research findings and discusses their implications. It provides a critical evaluation of the methodology, limitations, and potential threats to validity. The discussion highlights the practical implications for software ecosystem developers and stakeholders, and suggests future research directions based on the limitations of the current thesis.

Chapter 6: Conclusion Finally, the conclusion will summarize the key findings and discussions of the thesis. The chapter will consist of a restatement of the claims and results of the thesis and will discuss potential future work.

Chapter 2

Literature Review

2.1 Characterization of SECOs

Software ecosystems are complex systems that involve multiple stakeholders, including platform owners, developers, and users. In a systematic mapping study [9], researchers present a list of primary characteristics of SECO where natural ecosystem characteristics, business ecosystem characteristics, architectural concepts, evolution, and open source models are discussed. One key characteristic of software ecosystems is the openness of the software platform, which refers to the degree of restrictions with respect to development, commercialization, and use of the platform [14]. Another important characteristic of software ecosystems is their architecture. A body of work in the literature has studied the architectural characteristics of software ecosystems, but there is still a lack of reference architecture that provides a unified picture of the business-related and technical architectural building blocks [31]. The architecture of a software ecosystem can have a significant impact on its success and sustainability. A software ecosystem is also characterized by the interaction of multiple actors on top of a common technological platform. These actors can include platform owners, developers, and users, who work together to create a variety of software solutions or services. This interaction is crucial for the success and sustainability of the ecosystem [54].

Another important characteristic of software ecosystems is the community of stakeholders that companies create to support their platforms. This community can be made up of a diverse range of actors, including developers, resellers, customers, and more. By fostering a strong community of stakeholders, companies can ensure

that their platforms are well-supported and can meet the needs of their users [5]. Software ecosystems are also characterized by their interconnection with various institutions, such as standardization organizations, open-source software communities, research communities, and related ecosystems. These connections help to ensure that the ecosystem can evolve and adapt to changing needs and technologies [26].

Software ecosystems can also be characterized by the co-evolution of software projects within the same environment. This environment can be organizational, such as a company, social, such as an open-source community, or technical, such as the Ruby ecosystem. The co-evolution of software projects allows for the development of a diverse range of solutions and services that meet the needs of the ecosystem's users. These involve a software platform, a set of internal and external developers, and a community of users that compose relevant solution elements to satisfy their needs [6].

Many successful software ecosystems are characterized by their ability to be integrated in real-time [35]. This enables the ecosystem to build more functionalities onto its core tools and expand its uses. By providing real-time integration, companies can ensure that their platforms can adapt quickly to changing needs and technologies. Finally, many successful software ecosystems are characterized by their cost-efficiency, reliability, robustness, safety, and security. These characteristics are crucial for ensuring that the ecosystem can meet the needs of its users and provide high-quality products and services [6].

2.2 User Feedback in SECOs

User feedback is an important means of validating requirements and discovering new requirements in continuous software evolution [20]. Many software users give feedback online about the applications they use. This feedback often contains valuable requirements information that can be used to guide the effective maintenance and evolution of a software product [70]. However, users have a low motivation to provide feedback and prefer applications that do not interrupt their work [20].

Companies have been using user feedback as a critical tool to improve their software products and services. User feedback can be provided in various forms, including online forums, social media, and support ticket systems. Despite the escalation of software ecosystem development, growing a sustainable and healthy SECO remains a significant challenge. One approach to mitigate this challenge is the utilization of a mechanism that collects feedback from distributors (distros) and end-users of

the SECO releases [25]. In a software ecosystem, user feedback is typically used to guide the development of new features, bug fixes, and overall improvements to the product. One common approach companies use to address user feedback is through the use of bug-tracking systems. This process involves the collection, categorization, and prioritization of user-reported issues. There is only a handful of user feedback analysis studies in software ecosystem, one of which is an assessment of the feedback mechanism by OpenStack [25]. Companies can then use this information to prioritize their development efforts and ensure that the most critical issues are addressed first. There are many tools available for collecting user feedback such as Customer Satisfaction (CSAT) or Net Promoter Score (NPS) surveys [32] but user reviews are the most prominent way of collecting user feedback.

In software ecosystems, user feedback plays a crucial role for several reasons. Though there is limited literature that discusses the significance of user feedback in software ecosystems, it can be observed that there are several platform organizations that discuss the importance of feedback on their respective blogs and online publications. End-user feedback provides developers with valuable insights into how to add value to a product by collecting and responding to user feedback [70]. By collecting user feedback, developers can better understand user needs and preferences, and use that information to improve the product. Additionally, Microsoft[1] discusses that user feedback helps developers see things in new ways, correct their course, and learn and improve their work [19]. Through feedback, developers can identify problems and bugs in the software, and make necessary changes to improve the overall quality of the product. Additionally, user feedback allows companies to make changes to their products based on customer suggestions [34]. This helps businesses to stay competitive in the market by addressing the needs and concerns of their customers. Furthermore, user feedback can help the development team focus on resolving priorities, which can ultimately improve software quality while reducing development costs. By prioritizing feedback and addressing the most pressing issues, developers can improve the overall quality of the software and reduce the time and resources required for development. Overall, user feedback is a critical component of software ecosystems, and its importance cannot be overstated.

[1] https://www.microsoft.com/

2.3 Challenges and Gaps

Software ecosystem research faces several challenges and gaps. In a more generic context, understanding the complex interactions and selection of various stakeholders is crucial but difficult to achieve in SECOs. Developing effective governance mechanisms and designing appropriate business models are some significant challenges in SECO research [7]. Requirement elicitation is also a critical aspect of SECO development, as it involves identifying the needs and expectations of various stakeholders and translating them into specific software requirements [18] which can be obtained from user reviews.

Analyzing reviews that are specific to software ecosystems is a difficult task for several reasons. First, the channels of collection of feedback are several. From platform communities, forums, app marketplaces, and mobile application reviews, it is a difficult task to consolidate feedback for a single platform [25], let alone the generalization of ecosystems. Second, platforms contain several third-party applications, and it is hard to distinguish if feedback is provided by a user based on their experience of a single partner application, multiple applications, or the core platform itself. Researchers describe how it is difficult to analyze user input in software ecosystems in a 2019 case study [43] using Xero because platform providers must rely on their partners to offer suitable mechanisms for gathering feedback and making that information accessible. Additionally, when partner applications are fully integrated with the core product, the distinction between the two might become unclear; making it challenging for end users to distinguish between the core product's and the partner app's capabilities. End users are thus frequently faced with a variety of feedback channels which can make it challenging for them to provide feedback.

One of the primary gaps is the need to understand the complex interactions and selection of various stakeholders involved in software ecosystems including end-users [33]. While user feedback is crucial for validating requirements and discovering new ones, users generally tend to have a low motivation to provide feedback [20], and there is limited literature discussing the significance of user feedback in software ecosystems. Researchers and companies can mitigate this challenge by developing a mechanism for SECOs that collects, monitors, and reports feedback from all actors, distributors, and end-users and using user reviews. This thesis aims to address some of the gaps in software ecosystem literature by analyzing user reviews specific to software ecosystems.

Previous studies [50, 59] have mainly focused on classifying software product and mobile application reviews into bug reports, user experience, feature requests, and other categories that describe the nature of the review, but no generalization of specific problems in ecosystems has been made, given the complex nature of SECOs. There is a need to specifically examine software ecosystem feedback to first classify and then identify the unique problems of end-users, which may not always be general feedback such as bug reports and feature requests, in order to address data and platform analytics research challenges in SECO. This is an opportunity and a gap in SECO literature that can help developers better understand their users' needs and improve their software products accordingly.

2.4 Related Work

Software ecosystems have become an important area of research in software engineering due to the increasing complexity of modern software systems. A software ecosystem is defined as the interaction of a set of actors on top of a common technological platform that results in several software solutions or services [54, 38]. The SECO concept has been widely used to describe various software systems, including mobile app ecosystems [61], cloud computing platforms [10], and open-source software communities [42]. SECOs are different than software products that is a collection of solutions produced by a single business and packaged together. In a mainstream product, the tools in that package are all intended to function together, but adding additional tools that weren't part of the original package can be difficult [75].

Several studies have identified various problems in SECOs, such as coordination problems [23], vendor lock-in [62], interoperability issues [4], and project management [29]. These problems can hinder the growth and sustainability of SECOs and limit their potential benefits to users and developers. The challenges of SECO research include understanding the complex interactions and selection of various stakeholders [45], developing effective governance mechanisms [7], and designing appropriate business models [38]. Furthermore, requirement elicitation is a critical aspect of SECO development, as it involves identifying the needs and expectations of various stakeholders and translating them into specific software requirements [18].

The use of Natural Language Processing (NLP) and user review mining has become a popular research topic in software engineering due to the increasing importance of user feedback in software development [51, 8, 67]. This approach involves

analyzing user reviews to extract useful information, such as feature requests, bug reports, and user opinions. User review mining has been applied to various software engineering areas, including mobile app development and social media analysis. Work similar to mine has been on identifying privacy themes from user feedback [56] and classifying advertisement-related reviews [27]. However, there currently is no study that examines user feedback specific to Software Ecosystems.

A meta-analysis of the research field of software ecosystems [66] presents a relevant list of literature and six themes in which challenges for software ecosystems can be grouped: Architecture and Design, Governance, Dynamics and Evolution, Data Analytics, Domain-Specific Ecosystems Solutions, and Ecosystems Analysis. Johnson et al. [43] discussed the challenges of managing the crowd and of analyzing the inputs when multiple, heterogeneous feedback channels exist in software ecosystems. Although previous studies have been made to identify problems and concerns through user reviews [28, 69], my thesis focuses on analyzing reviews that are specific to software ecosystems, which has been a difficult task due to the unique analytics challenges in SECO research such as variety, uncertainty, and inconsistency, speed of generated comments and data, and privacy issues, [66] in addition to the ones discussed, which is why a case of analysis of user feedback in SECOs can be made.

I see understanding end-user concerns and user review classification as an opportunity and a gap in SECO literature. This can help SECO developers better understand their users' needs and improve their software products accordingly. My work aims to address some of these gaps by demonstrating a range of problems in this space so that SECO development emphasizes the importance of user-centric design.

Chapter 3

Methodology

The thesis uses a mixed-method approach with both quantitative and qualitative data analysis. It can be summarized as illustrated in Figure 3.1

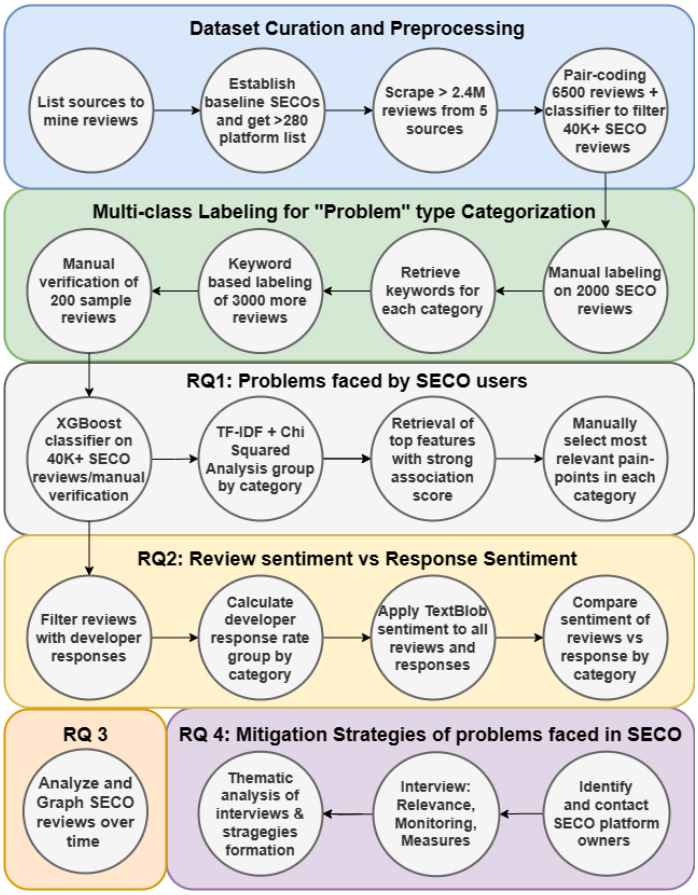

Figure 3.1: Research Design Summary

3.1 Baseline SECO platforms

Firstly, I listed 15 popular SECO platforms, based on their characteristics such as Integration, Innovation, Interoperability, Marketplace, Software as a Service (SaaS), and integration Platform as a Service (iPaaS) that define a SECO [38, 73, 42, 54] (also see SECO characterization in Chapter 2.2) in addition to the well-defined classification of software ecosystems by Jansen et al. [41]. According to this research, Software ecosystems are supported by different types of technology, including software platforms such as Andriod, iOS, Eclipse, etc., service platforms such as Wordpress, HubSpot, Salesforce, Spotify, etc., and software standards such as XBRL, Open Design Alliance, etc. I further expand on the discussed "service platform" by categorizing them according to service sectors. I selected one or two platforms for each sector that serve as a baseline to retrieve similar platforms. I picked e-commerce platforms (*Shopify* and *WooCommerce*), CRM tools (*HubSpot*, *ZenDesk*, and *MailChimp*), Software as a Service (SaaS) (*SalesForce* and *Xero*), Communications Platforms (*Slack* and *Teams*), Payment Integration software (*Square Up*), Integration Platform as a Service (iPaaS) solutions (*Zapier*), development platforms (*Wix* and *WordPress*), and Human Resources Integration Platforms (*Bamboo HR*).

3.2 Dataset Curation

I listed applications from Play Store and App Store search queries *(regex = "software" + "as a service/platform/ecosystems/integration")* and by retrieving platforms "similar" to the ones listed in baseline using the implemented libraries mentioned below. A total of 283 platforms were listed, but only **139** (see Appendix A for the complete list) of them were used for analysis based on relevant reviews which are discussed in the next. I used sources shown in Table 3.1 to collect user feedback from where I scraped **2,455,285** user reviews. The reviews were scraped using manual web scraping on TrustPilot, the *google-play-scraper*[1], and *app-store-scraper*[2] libraries in *Python*[3] for respective Google and Apple app stores, *Kaggle*[4] for Shopify store reviews, and directly from organizations. I combined all of it to form a single dataset with attributes *'source', 'platform', 'review content', 'review date', and 'developer response'.*

[1]https://github.com/JoMingyu/google-play-scraper
[2]https://github.com/cowboy-bebug/app-store-scraper
[3]https://www.python.org/
[4]https://www.kaggle.com/

Source	Reviews Collected	SECO Reviews
TrustPilot	100,666	4,146
Google Play	1,396,059	17,089
App Store	159,595	1,778
Shopify Store	797,967	16,250
Other	998	998
Total	**2,455,285**	**40,261**

Table 3.1: User Feedback Collection

I employed pair coding using Cohen Kappa's coefficient [47] with an agreement saturation exceeding 0.81. Before dividing the 6000 review sample, 500 randomized reviews (125 per rating) were chosen for manual assessment of software ecosystem review classification in pairs. An evaluator labeled 500 identical SECO-related reviews over 5 iterations of 100 reviews each, yielding an increasing agreement score until the 5th iteration. In total, 848 SECO-related reviews were identified through pair coding. SECO-related keywords such as "platforms, integration, ecosystems, plugins, sync" were used to identify SECO-reviews and were manually validated with an understanding of the context of the reviews. Reviews like "*Nothing but issues with this platform. You change a setting and it doesnt work on *third-party app name*, fix it on *plugin name* and the platform changes it back!! Terrible Customer service dont help much, just tell you to speak to *platform name*! Who say its an integration issue. Wasted two days trying to integrate this and would have been quicker doing it all manually!*" were marked as a SECO review whereas reviews like "*Its a very useless app. It cannot run in normal internet speed. It's a a lot of confusion to use this app. It buffers a lot while attending class*" were marked as not relevant.

Subsequently, an XGBoost classifier [15] was trained using the labeled 6000 reviews with a standard 80:20 proportion of train-test split for training the model. XGBoost is employed for binary classification due to its ability to handle imbalanced datasets effectively and provide accurate predictions. In the next phase of classification involving multi-class classification, XGBoost is chosen for its versatility in handling multiple classes and its capability to optimize for a variety of evaluation metrics [15]. The model was trained with 0.97 accuracy, 0.99 precision and 0.80 recall, and 0.89 F1-score, indicating high accuracy and reliability [55]. Having applied the 2.4 million reviews on this classifier, I was left with **40,261** reviews related to SECO from **139** platforms. Table 3.1 shows a breakdown of reviews retained from all the sources.

After listing and verifying 139 different platforms, I have presented a brief breakdown of these platforms in Table 3.2. The table provides an overview of the different categories of service platforms [41] that were identified and discussed in section 3.1 of the thesis. In order to better understand the nature of analyzed platforms, I manually classified them into the respective categories and then aggregated the count for each category, in order to provide a clear understanding of the distribution and prevalence of service platforms across different categories. For consistency of categorization, the third-party apps from the Shopify store are counted as 'Shopify' platform altogether. These platforms were grouped based on their nature by manual observation and by reading the app description (when needed) on their respective deployed sources.

Service Platform sub-category	**Platform Count**
E-commerce platform	57
CRM platform	35
Integration Platform	21
Payment platform	10
Website Development	10
Communications platform	6
Total	**139**

Table 3.2: Categories of Service Platforms

3.3 Manual Multi-class Labeling

On the 40,261 SECO-related reviews, I manually (2000 reviews) and with relevant keywords(3000 reviews) labeled over 12 percent of the SECO-review-only dataset. First, to select a statistically representative sample for the manual labeling, I assume a confidence level of 98 percent, a population proportion of 0.5, and a margin of error of 0.05. Using confidence to justify the sample [46], the sample size required to achieve this level of precision is calculated to be a minimum of 535. However, I selected 2000 reviews with 400 each from rating 1-5 to select a balanced dataset for manual labeling and further labeled 3000 more reviews as explained below.

Upon manual observation of the reviews, I listed 6 common SECO issue themes and performed single-label, multi-class, manual classification following a well-practiced card-sorting technique [68]. A considerable time was spent observing the reviews to develop the most relevant keywords, as shown in table 3.3, for each label. The rele-

vant keywords were created by observing term frequencies using TF-IDF Vectorizer [60] and manual observation on each subset of the label. All of the keywords in each category were mutually exclusive.

Label ID	**Label Name**	**Keywords**
0	Integration	integration, API, plugin, sync
1	Customer Support	customer, support, representative, speak
2	Design and Complexity	interface, confusing, hard, easy, design, customization
3	Privacy and Security	privacy, security, beware, fake, scam, login, authentication, password
4	Cost and Pricing	price, cost, refund, expensive, charge, buy, payment, credit, card, merchant, money
5	Performance and Compatibility	device, phone, slow, response, frequent, audio, video, crash, desktop, web, mobile, quality
6	Other	

Table 3.3: Relevant keywords for each SECO issue type

I use these keywords to filter the 3000 reviews from the dataset for further labeling. On this subset, I selected 600 reviews in each rating (1-5). I observed that a review belongs to a class with high confidence when at least 2 of the keywords were present in the review. For example, the review "*The integration does not seem to work. My social media doesn't sync anymore* included keywords such as "integration" and "sync", and was classified as an integration problem. Hence, to start, at least 2 keywords needed to be matched in either label to be classified. If none two matched, at least one keyword need to be matched. If none of the keywords matched, they were simply classified as 'Other'. I manually verified 200 randomized reviews and observed all of them accurately represented SECO-related concerns without any major overlapping of categories when filtered with at least 2 matching keywords.

3.4 Classifier and Analysis

I used XGBoost[5] as the primary classification model to classify reviews based on different categories. I used a dataset of 5000 train-test training reviews, which were preprocessed using well-used and known NLTK toolkit features[6] such as stop word removal, filtering out non-English words, tokenization, and lowercase conversion. I performed a training-test split with a frequently used ratio of 80:20, where 80 percent of the data was used for training, and 20 percent of the data was used for testing. I used precision, recall, and F1-score as evaluation metrics to measure the performance [55] of the model in different categories. The evaluation metrics were calculated for each category and then averaged using macro and weighted averages. As shown in table 3.4, the XGBoost model achieved an accuracy of 0.93, with a macro average precision of 0.92, recall of 0.89, and F1-score of 0.90, which indicates that the model was able to classify the reviews into different categories with very high accuracy. To validate the performance of the model, I manually verified a sample of 50 reviews from each category, which resulted in an accuracy of 91 percent.

Label ID	**Precision**	**Recall**	**F1-score**
0	1.00	0.97	0.99
1	0.99	0.97	0.97
2	0.97	0.93	0.95
3	0.94	0.79	0.86
4	0.82	0.82	0.82
5	0.88	0.74	0.80
6	0.85	1.00	0.92
Accuracy			0.93
Macro Average	0.92	0.89	0.90
Weighted Average	0.93	0.93	0.93

Table 3.4: Classification Report

I compared the XGBoost model's performance with the average (weighted) as shown in table 3.5, with other Scikit-learn[7] classification models, such as Linear SVC and Random Forest. The XGBoost model outperformed the other models with an accuracy of 0.93, while Linear SVC and Random Forest achieved an accuracy of

[5]https://github.com/dmlc/xgboost
[6]https://www.nltk.org/
[7]https://scikit-learn.org/

0.84 and 0.82, respectively. The methodology demonstrates the effectiveness of using XGBoost for classifying reviews into different categories. The evaluations suggest that the model can be used for the automated classification of SECO-related reviews, which can help in identifying customer sentiments and feedback about platforms.

Classifier	**Accuracy**	**Precision**	**Recall**	**F-1 Score**
XGBoost	**0.93**	**0.92**	**0.89**	**0.90**
Linear SVC	0.84	0.83	0.82	0.82
Random Forest	0.82	0.81	0.78	0.79

Table 3.5: Model Comparison (macro-average)

I confidently implemented the classifier on the 40,261 software ecosystem reviews. I identified the most frequent terms (also referred to as features) using a set of *negative* reviews for each category. The set of negative reviews belonging to each category is kept using Vader Sentiment [36] with a negativity score of over 0.4. Vader Sentiment generates polarity scores using the formula listed in Equation 3.1 where N is the number of words in the text, w_i is the sentiment intensity of word i (obtained from the Vader lexicon), C_i is the number of occurrences of word i in the text, and $\sum_{j=1}^{N} w_j$ is the sum of the absolute values of the sentiment intensities of all words in the text.

$$\text{Vader Sentiment Score} = \sum_{i=1}^{N} \frac{w_i C_i}{\sum_{j=1}^{N} w_j} \tag{3.1}$$

The frequent reviews terms are extracted using TF-IDF as "features". In equation 3.2, t is a term (word), d is a document, D is the corpus (collection of documents), 'tf' is the term frequency, and 'idf' is the inverse document frequency [60].

$$\text{tf-idf}(t, d, D) = \text{tf}(t, d) \cdot \text{idf}(t, D) \tag{3.2}$$

The reviews were preprocessed to remove non-English words, stop words, and tokenize them. I then performed Chi Squared analysis to measure the association between each feature and its' corresponding label. The chi-Squared analysis is a popular method not only for hypothesis validation but also useful for feature selection and computing association between features and their labels [78, 64, 65]. It can be implemented using the formula in 3.3 where χ^2 is the chi-squared statistic, n is the number of categories, O_i is the observed frequency in category i, and E_i is the expected frequency in category i.

$$\chi^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i} \tag{3.3}$$

Additionally, I compute three-tier sentiment polarity scores (positive, neutral, negative) of review content and their corresponding response content with TextBlob.[8] TextBlob is an effective open-source Python library used in the analysis of different textual contexts, especially understanding text-based sentiments [3]. It can be implemented using the formula listed in 3.4 where w_i is the weight of the i^{th} word and p_i is the polarity score of the i^{th} word. n is the total number of words in the text. The polarity ranges from -1 to 1. I group them by categories in order to understand developer response sentiment towards the reviews.

$$\text{TextBlob Polarity} = \frac{\sum_{i=1}^{n}(w_i \cdot p_i)}{\sum_{i=1}^{n} w_i} \tag{3.4}$$

3.5 Interviews

In the last part of the study, I employed qualitative research using semi-structured interviews [49]. The interviews were structured to elicit information about the measures they take to address issues that were identified in the user feedback analysis so that a set of mitigation strategies could be derived and articulated. The interviewees for this part of the study consisted of four platform executives who were selected based on their roles and positions within their respective companies as shown in table 3.6 in addition to the platform's profiling (platform's name, however, is coded as P1, P2, etc. to maintain anonymity). The selection of interviewees used for this study is purposive sampling [71], which involves selecting individuals based on their expertise and knowledge of the research topic as well as the companies' reviews being part of the analysis. The interviewees were asked questions such as how they monitor the user feedback in their platforms, how they ensure seamless integration, what strategies would they recommend in solving the found challenges, how they manage an evolving marketplace of vendors, and other questions relating to the findings present in Chapter 4.

The interviews were conducted virtually, and the questions were designed to elicit information about how the findings from user feedback analysis relate to these platforms. To ensure consistency and accuracy of the data collected, the questions were

[8] https://textblob.readthedocs.io/en/dev/

Title	Company	Established	Size (employees)	Country
Chief Technology Officer	P1	2017	100-200	Canada
VP Engineering	P2	2013	50-100	Canada
Platform Ecosystem Advocate	P3	2006	8000-10000	USA
Chief Technology Officer	P4	2017	300-500	Nepal

Table 3.6: Interviewee Profile

designed to relate to the findings from the user feedback analysis, which identified different categories of issues. The participants were presented with the results of the analysis in the category of issue from the classification and were asked specific questions about their awareness of these issues and the measures they take to address them.

The interviews were conducted following ethical principles, including informed consent, confidentiality, and privacy. The participants were informed about the study's purpose, their role in the study, and the confidentiality and anonymity of their responses. They were also given the option to withdraw from the study at any time.

The data collected from the interviews were transcribed, sorted, and analyzed using a thematic analysis approach. Thematic analysis is a widely used qualitative research method that involves identifying and analyzing patterns and themes in data [16]. This approach enabled me to identify and analyze the themes and patterns in the data related to how companies identify and address issues related to software ecosystems through user feedback. Table 3.7 provides an example of how key points from the interviews were grouped into themes and were transformed as a strategy.

Example Themes	**Platform**	**Example Quotes**
API vs One-off	P2	*like most about working with that small startup Is that they have an api first mentality? It's in the dna of the company that they're building an API. So they're not going to hit the same issues, right?*
	P4	*The entrepreneur is not reading. In their book says, Api first, you know, similar to a route that i have the companies like i work in right now don't have a data strategy.*
Marketplace Regulation	P3	*So we can't control how every single partner delivers that support obviously. You know, we have standards in our marketplace.*
	P4	*If somebody had essentially abandoned all supported their app and they would be removed from our marketplace*
Feedback management	P1	*We use a full story, which can monitor user interactions within the apps. And then, you know, we get notices of, like rage clicks, things like that, where they go.*
	P3	*But Yes, that would address the privacy and security. Issues from the platform side. Also, one of the things we noticed in user feedback, where the fake reviews in the marketplace, They were, so it's crucial to monitor feedback.*

Table 3.7: Thematic analysis example

Chapter 4

Findings and Analysis

In this chapter, I explore the distribution of SECO reviews and answer the research questions from Chapter 1 and discuss the findings.

4.1 Distribution

The software ecosystems reviews are classified into seven categories: Integration, Customer Support, Design and Complexity, Privacy and Security, Cost and Pricing, Performance and Compatibility, and Others. The distribution of ratings in each category can be summarized in Figure 4.1.

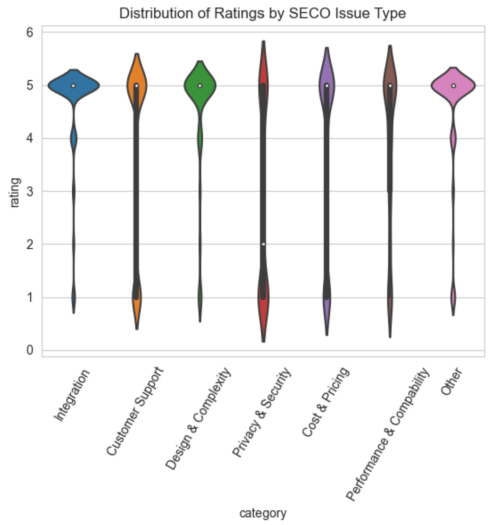

Figure 4.1: Distribution of Rating per Category

Out of a total of 40,261 reviews, 'Integration' has the highest proportion of software ecosystem reviews, at 28.85 percent. 17.67 percent of the reviews fall under 'Customer Support', making it the second highest category, followed by 'Design and Complexity' with 8.35 percent. The lower proportion of reviews can be observed in 'Privacy and Security' at 4 percent, 'Cost and Pricing' at 6.74 percent, and the lowest proportion of reviews belonging to 'Performance and Compatibility' at 2.80 percent. The irrelevant reviews as discussed in Chapter 3.3 were classified as 'Other' with 31.58 percent of the reviews. The count of final reviews per category per rating can be viewed in Table 4.1 and median ratings and review proportion can be viewed at a glance in Figure 4.2.

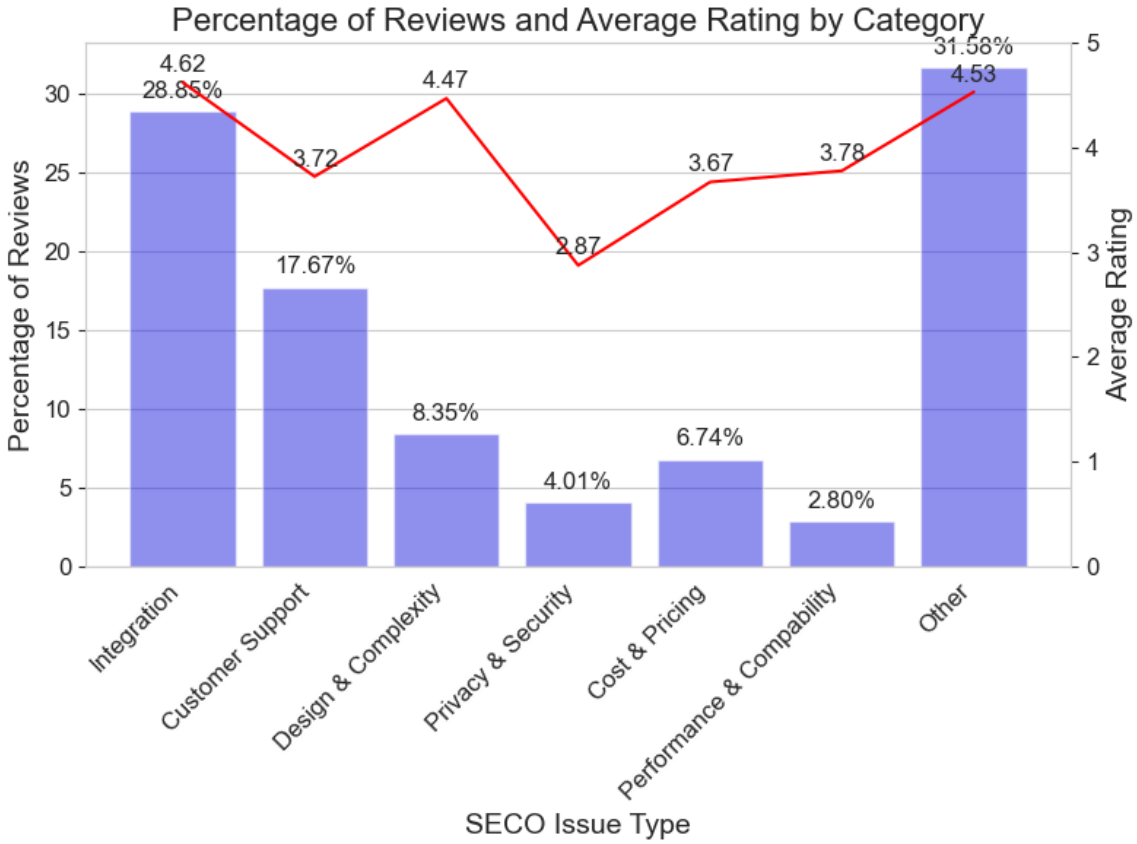

Figure 4.2: Proportion of Reviews and median Rating per Category

SECO reviews, on average have a rating of 3.85/5 (excluding the 'other' category). The distribution suggests that users place a high value on Integration when it comes to software ecosystems and that feedback related to customer support is a common occurrence. 'Integration', and 'Design and Complexity' are highly rated. The

Label ID	All	Rating 1	Rating 2	Rating 3	Rating 4	Rating 5
0	11615	502	215	319	1067	9512
1	7114	1884	260	208	339	4423
2	3360	245	90	89	345	2591
3	1616	711	98	86	125	596
4	2714	701	100	129	242	1542
5	1126	187	78	113	166	582
6	12716	910	167	315	1153	10171
Total	**40261**	**5140**	**1008**	**1259**	**3437**	**29417**

Table 4.1: Post-classification Reviews in Numbers

'Integration' category had the highest median rating, with a median rating of 4.62, indicating that users generally have a positive experience with integration-related issues. Similarly, the 'Design and Complexity' category received a median rating of 4.47, indicating that users have a generally positive experience with these types of issues.

'Privacy and Security' is of major concern. The 'Privacy and Security' category received the lowest median rating of 2.87, indicating that users are highly concerned about issues related to privacy and security. This is an important finding, as privacy and security are critical components of any software ecosystem. 'Cost and Pricing' receives mixed reviews but remains a significant concern for many as this category received a median rating of 3.67, indicating that users have mixed feelings about these types of issues. This suggests that pricing is a complex issue that depends on a variety of factors and that users have a range of opinions about what constitutes fair pricing. 'Performance and Compatibility' also received mixed reviews. The 'Performance and Compatibility' category received a median rating of 3.78, indicating that users have mixed feelings about these types of issues as well. This suggests that users have varying experiences when it comes to performance and compatibility-related issues and that these issues can be complex and difficult to resolve. The 'Other' category is quite diverse and covers a wide range of issues, making it difficult to draw specific conclusions about what users are most concerned about in this category.

It is interesting to note that 'Privacy and Security' with one of the lowest proportion of reviews received the lowest median rating of 2.87. It only accounts for 4 percent of all reviews, indicating that it may not seem a major concern for most users but the users that do see it as a concern, understand the severity of the issues. Similarly, 'Performance and Compatibility' has the lowest proportion of reviews at

2.8 percent but received a below-average rating of 3.78, again indicating the severity of issues that many users may not have noticed. It is worth noting that while the proportions of reviews in each category can provide insights, they do not necessarily indicate the quality of the software product. The median rating for each category is important to consider, as it can provide a more nuanced understanding of users' experiences. For example, while 'Customer Support' has a high proportion of reviews, its median rating of only 3.72 is lower than the median ratings of 'Integration' (4.62), 'Design and Complexity' (4.47), and all but 'Privacy and Security'. This suggests that while users may be more likely to leave reviews about customer support, their experiences have not always been positive.

The proportions of SECO reviews in each category can provide valuable insights into the areas of software ecosystems that are most important to users. While customer support and other general aspects of software products are of utmost importance, a holistic overview of all the aspects of software feedback needs to be captured. Additionally, the relatively low proportion but a very low median rating of reviews in the 'Privacy and Security' and 'Performance and Compatibility' categories may suggest that users have fewer but serious concerns in these areas. However, to gain a more complete understanding of users' experiences, it's important to consider both the proportions of reviews and the median rating for each category.

Finding 1: 'Privacy and Security' reviews, having one of the least proportion also have the least median rating, demonstrating fewer but extremely critical feedback. 'Integration' is the most discussed SECO-related review category, followed by 'Customer Support' and 'Design and Complexity', all of which have generally positive ratings. Despite 'Cost and Pricing, and 'Performance and Compatibility' topics having the least proportion of reviews, they have a lower-than-average rating.

4.2 RQ 1: SECO End-user pain points in User Feedback

In this section, I present the findings from reviews for all classified areas of SECO issues.

4.2.1 Information Retrieval

In order to find the discussion topics from user reviews, I performed the following set operations as an implementation of the analysis methodology discussed in Chapter 3.4:

Let C be a set of reviews with respective category IDs, where review r_i has a sentiment score $s_i \in positivescore, negativescore$.

Let $C = (l_i, R_i) \mid i = 1, 2, \ldots, n, s_i =$ negative score > 0.40 be the set of negative reviews.

Let $L = l_1, l_2, \ldots, l_n$ be the set of categories present in C.

Define $R_i = r_j \mid r_j \in R_i$ and $s_j =$ negative as the set of negative reviews belonging to category l_i.

Define TF-IDF$_c : R_c \rightarrow F$, where $F = (r, f) \mid r \in R, f \in W$ is the set of review features for all reviews in C.

Let $F'_l = f \mid (r, f) \in$ TF-IDF$c(R), r \in R_l$ be the set of features present in reviews of category l. The frequent reviews terms are extracted using TF-IDF as "features".

Let $\chi^2(f, l)$ be a statistical measure of association between feature f and category l. Then, the set of categories and their top 100 features with a $\chi^2(f, l)$ is given by $(Labels, (feature, score))[1, 100] = (l, F'_l, \chi^2(f, l)) \mid l \in C, f \in F'_l, \chi^2(f, l)$.

4.2.2 Integration

The first category of pain points in software ecosystems is related to integration which is summarized in Figure 4.3 which visualizes the top features extracted along with their respective Chi-square scores, with the most common issues being **"problems with integration"** and a **"lack of integration"** altogether. These are followed by **"cross-platform"** issues, **API errors**, and **"API key problems"**. An error code, **"API Error 411"** was one of the terms with higher associations. It is clear that users are frustrated with the difficulty of integrating different software components and systems, which leads to inefficiencies and lost productivity which is a key

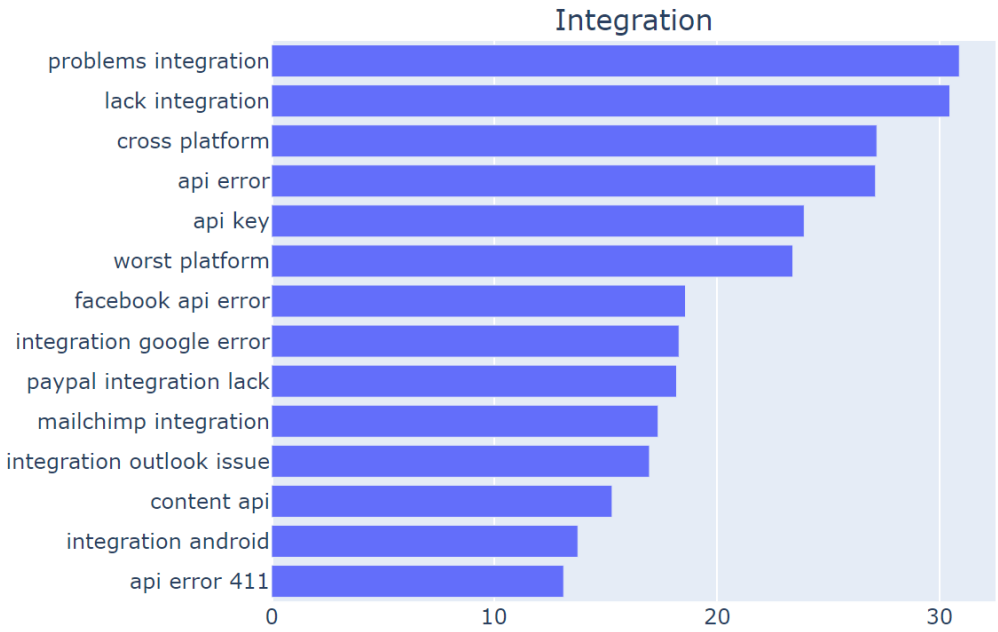

Figure 4.3: Top Features extracted from Integration Category

indicator of SECO health [17]. One of the most common integration complaints is regarding **"Facebook API errors"**. For instance, a user complained about trying to integrate their e-commerce store with Facebook for advertising purposes but cannot get the integration to work, resulting in wasted time and potential lost revenue. Similarly, integration errors with **"Google API"** caused issues with SEO and other critical aspects of online business. Another common integration issue mentioned in the data is the lack of **"PayPal integration"**, which can be a significant problem for businesses that rely on payment integrations. **"Mailchimp integration"** and **"Outlook Integration"** are other common issues that cause problems with email marketing campaigns, and not having **"Content API"** caused problems for content creators who want to integrate their content with various platforms. It is worth noting that several of the pain points in this category are related to specific platforms, such as **"Android integration"**. This suggests that users are encountering issues when trying to integrate these platforms with other software systems or components. The pain points related to integration in software ecosystems can have significant impacts on productivity and revenue [12].

"...worst customer service, api docs and implementation i have ever encountered during my career as a developer. i had to build everything from scratch, since they are not working with react native, and doesn't care about implementing it on the future. even then, the ios sdk simply doesn't work and there is no explanation for why the app crashes. i hope i never have to work with them again."

In the example review above, cited from the dataset, the user's negative experience with the software ecosystem provider has likely resulted in frustration and wasted time for the user, as well as potential negative impacts on their project or business. For the software ecosystem provider, this review signals a need to improve their customer service and integration support, as well as increase compatibility with popular development tools to better serve their customers and avoid negative feedback.

Finding 2: Dysfunctional integration such as API key errors, unavailable integrations, platform-specific, operating system-specific, and device type-specific errors are the leading integration problems faced by SECO users, directly impacting productivity.

4.2.3 Customer Support

The second category, summarized with features and their respective Chi-square scores in Figure 4.4, is regarding customer support, which is crucial for any software's success [30]. Although at first glance it may look generic customer support problems, they were extracted from the reviews that contained SECO-specific complaints. The top pain point in this category is **"worst customer service"** with the highest chi-squared value, which indicates significant dissatisfaction among users with the customer service provided by the software ecosystem. Other pain points related to reaching customer support include **"service impossible to reach"** and **"impossible to speak"**. These issues suggest that users are having difficulty contacting customer support representatives through available channels such as **"phone, email, or chat"**. Problems related to the quality of customer support include **"service joke"** and **"service rude"**. This suggests that users are dissatisfied with the quality of service provided by customer support representatives, either due to lack of knowledge, empathy, or professionalism as one of the frequent terms observed is **"doesn't answer question"**. Some reported the support staff to be **"rude"**. The language

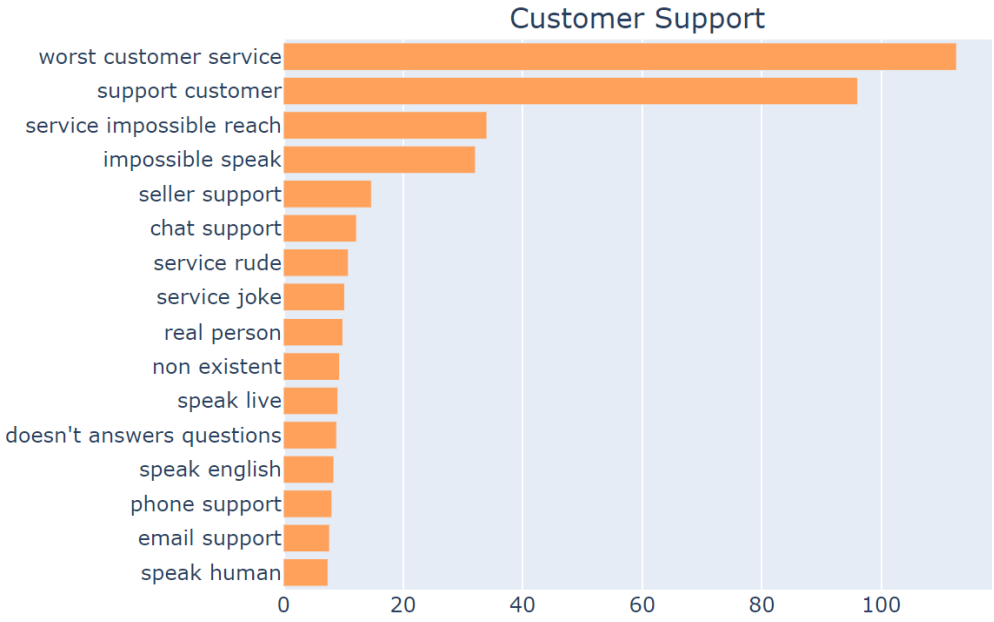

Figure 4.4: Top Features extracted from Customer Service Category

barrier is a problem as well, as the term **"speak English"** was frequently observed. Customers seem to prefer speaking to **"real humans"** over **"chat"**. This could be a major issue with software providers and companies relying on chat-bots [1, 24] over live support as poor customer service could result in lost customers and damage to brand reputation [63]. The fact that some users are finding it impossible to speak with a **"live person"** also suggests that there may be issues with the support channels available to users. Platforms may need to invest in better support channels to ensure that not only users, but the third-party developers have access to the help they need as **"no seller support"** was observed to be equally significant. Users have a variety of dissatisfaction with the customer support provided by the platforms, both in terms of reaching customer support and the quality of service provided. This can lead to frustration, poor user experience, and ultimately loss of customers especially in customer relationship management(CRM) and B2B platform models. [77].

> *"...makes it almost impossible to find their customer service number! the ** chat bot just leads you to the "help" page. what really makes me mad is that I do not take contacting telephone or email support lightly. I make every last attempt to solve a problem myself before contacting ***. the problem is that *** does not respect me at all. I hate that..."*

The review cited above[1], is a good example to consider the limitations in discussing how helpful chat-bots can be in customer support for platforms. Additionally, since ecosystems are prone to high uncertainty in a number of actors, internal and external, it is important for platforms to stay on top of making the end-users feel understood and valued.

Finding 3: SECO users prefer to speak to live customer support over chat and emails. Wait times, not having sufficient support for end-users and third-party developers, language barrier, and lacking empathy are the core customer service problems faced by SECO users.

4.2.4 Design and Complexity

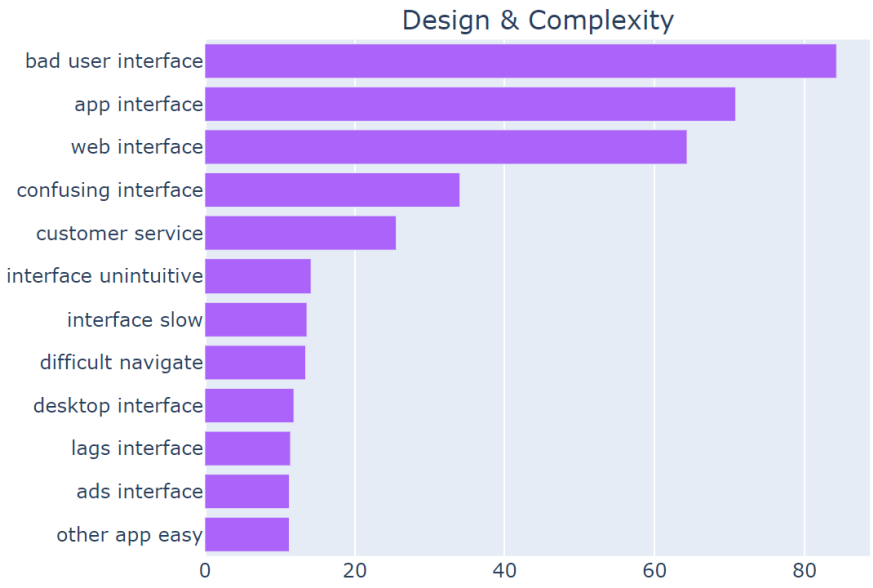

Figure 4.5: Top Features extracted from Integration Category

While the scope of user experience is very wide [48], in the thesis, the most frequent pain point in this category, summarized with features and their respective Chi-square scores in Figure 4.5, is around the topic **"bad user interface"**. Though this is a general finding, it can be evaluated in several ways from previously established theories [58] and my own findings such as problems in **"sorting"** and **"ads"** which

[1]Only the relevant part of a review is cited to trim the length.

range from a simple design issue to more complicated in-app advertisement problems. Some of the other topics provide more specific examples of what users find challenging about the software interface. For example, the **"mobile app interface"** topic showed that users have difficulty with software that is primarily mobile-based. The **"web interface"** related reviews mentioned that users find web-based software challenging to navigate. **"Confusing interface"** and **"interface unintuitive"** further emphasize the difficulty users have in understanding and using the software interface. Additionally, **"interface slow"** and **"lags"** indicate that users have problems with the performance of the software. Reviews suggested that **"slow"** and **"unresponsive"** platform interfaces are significant sources of frustration for users. Issues such as **"desktop interface"** and **"other app easy"** indicate that users have trouble with desktop-based software and that they may compare it unfavorably to other, more user-friendly applications, proving the need for platforms to be competitive. The topics in this category suggest that users find software with bad or confusing user interfaces frustrating and difficult to use. This can lead to decreased productivity, innovation, and satisfaction with the software, which ultimately influences the platform's life and death, as learned from the failures of early ecosystems [22].

> *"basic functions like changing your address or applying coupons are impossible! and support is hard to contact. i spent 45 minutes trying to apply the right coupons at check out and figure out how to change my address because i moved. eventually i just gave up and found similar products at other retail stores."*

The user review above highlights two major issues: firstly, the basic functions of the platform are not user-friendly and difficult to use, which can lead to frustration and potentially cause users to switch to alternative options. Secondly, the lack of accessible and responsive customer support makes it even more challenging for users to resolve their issues or get help, which can further exacerbate their frustration and lead them to abandon the platform altogether. I found several other feedback where users talked about switching to a competitor because of the complexity of ecosystems.

Finding 4: Mobile responsiveness, confusing and unintuitive interfaces, in-app ads, lags, and crashes among others are the primary design issues faced by SECO users which are causing them to switch to the platforms' competitors.

4.2.5 Privacy and Security

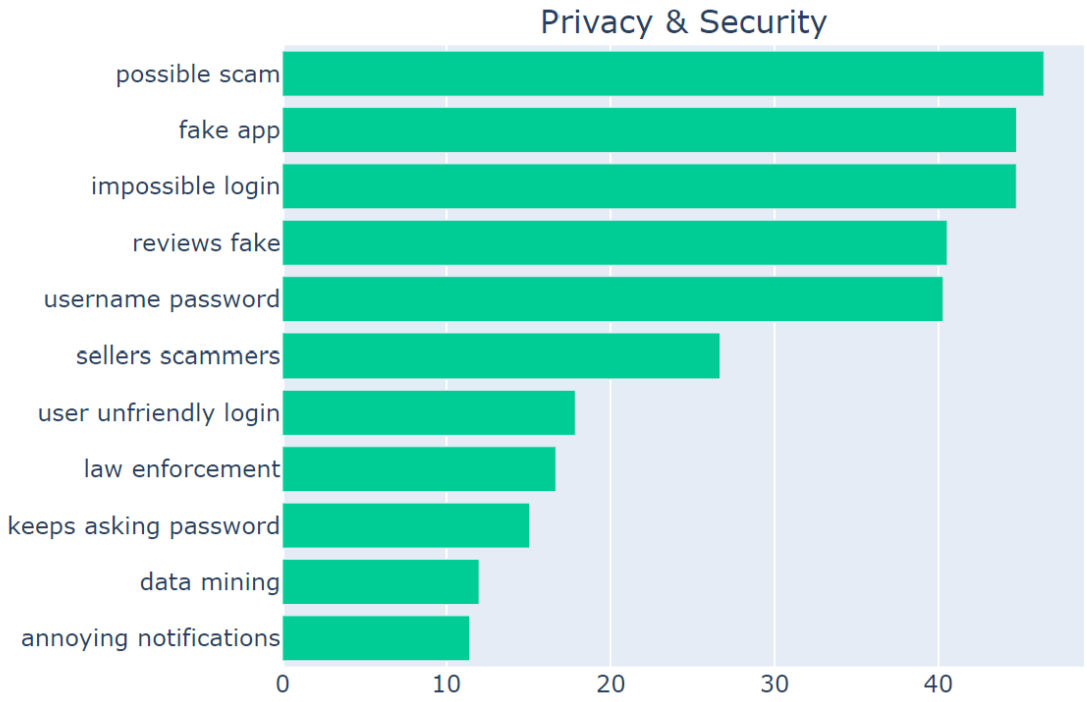

Figure 4.6: Top Features extracted from Integration Category

Privacy and security are critical concerns for most software users, especially in the e-commerce platform realm [72], and it is no surprise that this category, summarized with features and their respective Chi-square scores in Figure 4.6, has some of the most critical pain points. Users are often hesitant to trust a platform with their personal and sensitive information [2], and the reviews in this category reflect that. The features discussed in this category are **"possible scam"**, **"fake app,"** and **"fake reviews"** all of which suggested that users are worried about the legitimacy of the platform and the third-party apps they are using. These reviews stem from experiences with fraudulent websites or apps that have scammed users out of their money or personal information. Some important pain points in this category were **"impossible login"**, **"user interface login"**, and **"keeps asking password"** indicating that users are struggling to access their accounts. This was observed due to a variety of factors, such as technical issues or security measures that are too stringent. Regardless of the reason, it is clear that users find this issue frustrating and concerning. An interesting issue topic identified is **"data mining"** showing that users are concerned about how platforms are mining their personal data. Other pain points in this cate-

gory relate to user authentication and security measures. For example, **"username password"** and **"user unfriendly login"** suggest that users are struggling with the login process, whether it is too complex or too simplistic. The issue topic **"marketplace scammers"** suggests that users are worried about fraudulent third-party marketplace sellers on the platform, while **"law enforcement"** indicates that some users may have concerns about the platform's relationship with law enforcement and compliance. Platforms that can address these concerns and implement robust security measures by clearly stating policies, increased lucidity, and readability as readability are likely to have happier and more trusting users [21].

> *"platform doesnt do anything to make sure quality is a priority nothing to make sure the user is safe from being scammed. they have no policy in place of quality is low they just say its a matter of taste... save your money use a legitimate platform so at least you are guaranteed quality when i say bad i mean bad. you will not get a refund they prettty much doesnt do refunds unless mutual parties agree, but if somebody is scamming you from across seas they are not going to have the best moral compass and do the right thing and cancel. i have been scammed twice in this platform. remember review can be bought on this and they also can be bought on **** . unfortunately i had to learn the hard but you don't please if you are tight on money like i am don't waste it on ***."*

The user review highlights significant issues with the quality and safety of the platform, with no policies in place to address these concerns. They lack policies against marketplace scams and the potential for fake reviews raises serious doubts about the credibility of the platform. The user strongly advises against using the platform and suggests finding a legitimate alternative that guarantees quality and safety. Platforms should rigorously monitor the services offered by third-party developers to ensure quality and satisfaction. Additionally, they should have review monitoring and filtration systems to ensure that there is an absence of fake reviews in marketplace apps and services.

Finding 5: Possibility of being scammed and third-party applications with fake reviews in open marketplaces, causes fear in SECO users, alongside worries of possible user data mining, privacy regulations, and problems with authentication poses security concerns.

4.2.6 Cost and Pricing

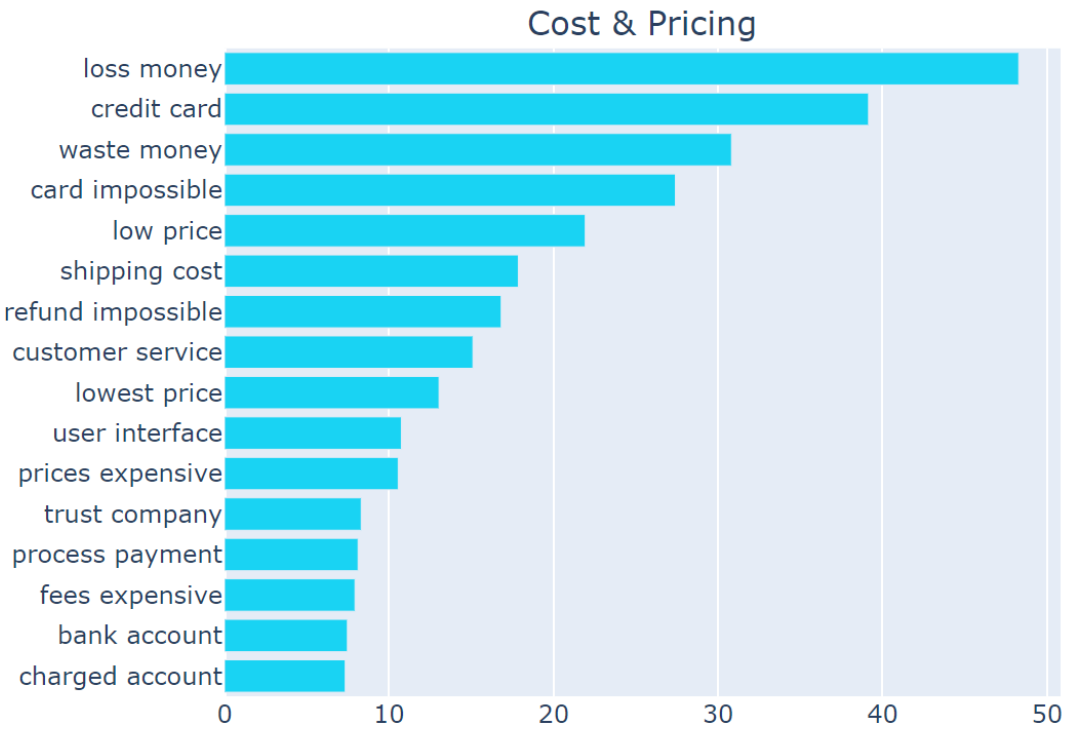

Figure 4.7: Top Features extracted from Integration Category

Pricing is an important characteristic of the ecosystem marketplace [13, 40]. This category, as illustrated with features and their respective Chi-square scores in Figure 4.7, focuses on the cost and pricing structures of SECOs. The main pain points raised by users were related to **"losing money"**, issues with credit card payments, and **"expensive fees"**. The term **"loss money"** had a high chi-squared score of 48.28, indicating that users were highly dissatisfied with the amount of money they were losing. This was further examined and it was found out that the reasons for this were **"unexpected charges"**, **"hidden fees"**, and ineffective **"refund policies"**. For example, a user signed up for a free trial but was automatically charged for a premium **"subscription"** without their knowledge. The pain point **"credit card"** had a significant association score, indicating that users had issues with their card payments. This was found be to a problem with the payment processing systems in the platforms. For example, a user attempted to make a payment but received an error message indicating that their card had been declined, even though their card was valid and had sufficient funds. The pain point **"waste money"** indicated that

users felt that they were spending money on a product that was not worth the cost. This was found to be caused due to a lack of features and poor performance compared to **"other platforms"** in the same price range. Other pain points related to cost and pricing include **"refund impossible"**, **"prices expensive"**, **"fees expensive"**, and **"charged account"**. These raised issues suggested that users lost the **"company trust"** and were dissatisfied with the pricing and fees associated with the platforms and their services and that they had difficulty obtaining refunds or finding affordable alternatives.

> *"we have used the free version of *** with great success but the added functionalities are a must. the starter plan - $25 month only gives you incremental features, then it goes to the pro plan $399, powerhouse plan $699 month, an extremely steep and unrealistic increase in prices. Could not find policy about pricing and refunds easily. Some people i know got their payment charged without notice. ideally we would of loved the pro plan functionalities but weighing the cost benefits, we've decided against it which is a shame."*

In the cited review above, the user review highlights the success of the free version of the platform and the necessity of additional features provided in the paid plans. However, the significant price increase between the starter plan and the pro plan is perceived as unrealistic and steep. This review underscores the importance of considering the cost-benefit analysis of upgrading to a higher plan and how it can impact the user's decision-making process. The user is expressing frustration not being able to find policies, account charges without notification, and the steep pricing structure of the software service they are using, which is preventing them from accessing desired features despite their success with the free version. This highlights the importance of balancing cost and benefit when considering software subscription plans. Platforms may need to consider offering more reasonable pricing plans to retain customers and avoid losing them to competitors.

Finding 6: Payment charged without the user's knowledge along with hidden and unexpected charges to their accounts makes users lose trust in the platform, which causes them to compare services value for money in alternative platforms, calling for changes in pricing and refund policies.

4.2.7 Performance and Compatibility

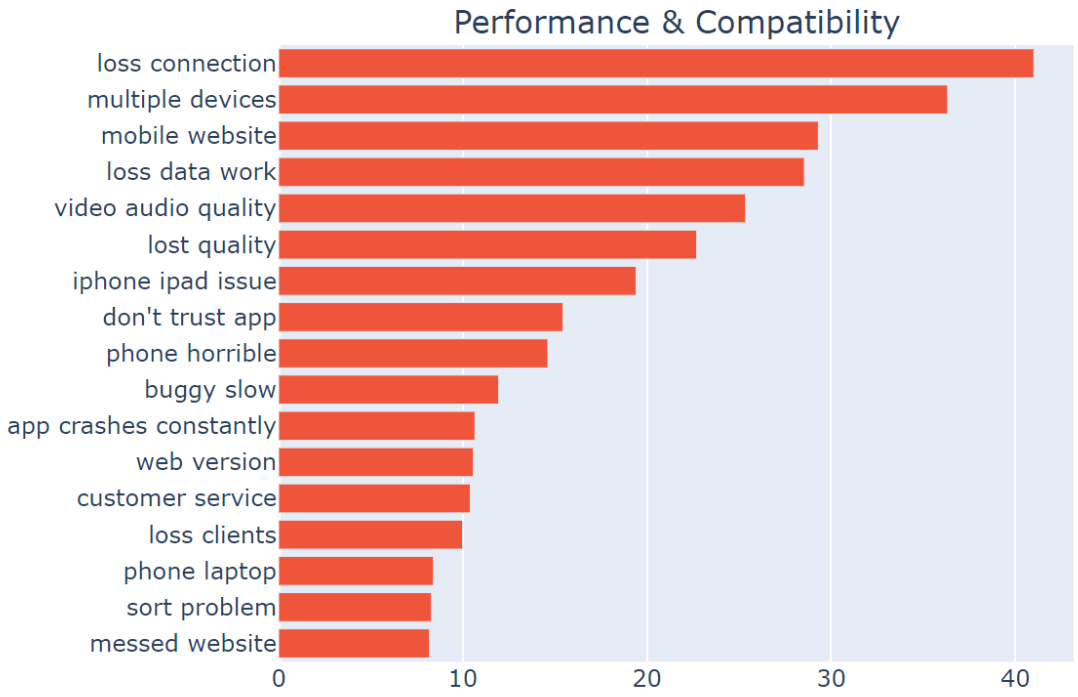

Figure 4.8: Top Features extracted from Integration Category

As companies are increasingly choosing cross-platform development over native development [76] and progressive web apps [52], the most significant pain points in this final category, as shown in Figure 4.8 with its features and respective Chi-square score, seem to be "web interface" and **"device version"**, followed closely by the topic **"multiple devices"** and **"loss connection"**. These pain points and corresponding reviews suggested that users are experiencing sync and connectivity issues across web, desktop, and mobile versions of the platform. For instance, a user was unable to log in to their account from their mobile after having already logged in from their computer. This was found to be frustrating for users who need access to their accounts from various locations and devices. Another common topic in this category is "**mobile website**" suggesting that users are having difficulty accessing and using the software ecosystem on their mobile devices as users complained about "**cross-platform**" usage. For example, a user had a platform working on the Andriod version, but not on the iOS. The pain point "**loss data work**" suggested that users are experiencing data loss or data corruption while using the software ecosystem which

is causing them to lose their work, a few reviews confirmed. This had happened due to network "**connectivity issues**" and alleged, "**software bugs**." For instance, a user lost important data while working on an accounting task for the software platform. Other pain points in this category included "**video audio quality**", "**lost quality**", "**iPhone iPad issue**", "**don't trust app**", "**phone horrible**", "**buggy slow**", "**app crashes constantly**", "**web version**", "**loss clients**", "**phone laptop**", "**sort problem**", and "**messed website**". These pain points suggest that users are experiencing issues with the overall functionality and reliability of the software ecosystem, causing them to lose trust in platforms, and even instances of businesses losing clients. For example, a review regarding online consultation reported losing their client due to poor video and audio quality of a communications platform. Issues with connectivity, data loss, and overall performance were found to be frustrating and resulted in users seeking out alternative platforms.

> *"Please make it a native app, current web based app horribly slow. not only in android, across all the platforms. Works fine on web from computer but crashes on the web browsers from phone. If it had a native mobile app it would be much easier. Video quality is horrible on mobile devices with web browser view. Not using this unless they have an actual app"*

The example user review above highlights the importance of having a native mobile app for a software ecosystem. A native app would provide better performance, particularly on mobile devices, and offer a more seamless and stable user experience. This review can serve as a reminder to software ecosystem developers that mobile optimization is crucial and that providing a native app for mobile users can enhance the overall user experience and improve customer satisfaction. Several reviews like these call for an evaluation of ecosystem development strategies that need to cater to different vendors and end-users who may not necessarily have the same platform needs.

Finding 7: Users lose work, data volume, and connection due to connectivity and feature-related bugs in platforms. Platforms are not always compatible with different devices and operating systems, resulting in crashes and ineffective all-around performance. I/O features such as audio and video are huge performance problems for SECO users.

4.2.8 Other Reviews

The "other" category in the SECO (Software Ecosystem) reviews classification encompasses a diverse range of issues that do not fall under the predefined six categories: Integration, Customer Support, Design & Complexity, Privacy & Security, Cost & Pricing, and Performance & Compatibility. These issues highlight unique and unexpected challenges faced by users within software ecosystem. An example of a review that fits into this "other" category could involve a user expressing frustration about a particular software's lack of integration with a newly emerging technology, such as augmented reality devices. In this case, the user might discuss how the software's failure to adapt to these new technological advancements is hindering their productivity and limiting the software's potential. For instance, reviews like *"i tried out the free version of the platform to see if this would be a good fit for my company. it is absolutely perfect, aside from 2 major flaws that you can not redeem more than one of the dollar amount rewards at a time and there is no way to redeem points if other discounts are activated"*. This scenario from a platform that allows users to redeem points from third-party vendors, showcases a challenge that doesn't neatly fit into the established categories but still underscores the evolving nature of software ecosystems and the dynamic array of user concerns.

4.3 Shopify Vs Xero: A Case Study

Shopify and Xero are two platforms that offer different functionalities but share a common nature: the platform ecosystem is at the heart of the business. Shopify is an e-commerce platform that helps businesses build and manage their online stores, while Xero is an accounting software that assists businesses in managing their financial operations. As a part of the thesis, I analyzed user feedback specific to these companies to compare and justify the two platforms based on six broad categories of issues.

4.3.1 Shopify

Shopify constituted over 14,800 reviews from the 4 sources mentioned in Chapter 3, out of which over 1500 reviews were about software ecosystems. It could be observed that the SECO-related review ratings in Shopify were lower than all-inclusive Shopify reviews. The median rating for Shopify reviews across all reviews was 3.5/5,

the median rating for it's SECO reviews was 3/5 while the benchmark rating for SECO reviews across all companies and platforms in the study is 3.85/5. It can be summarized in Figure 4.9.

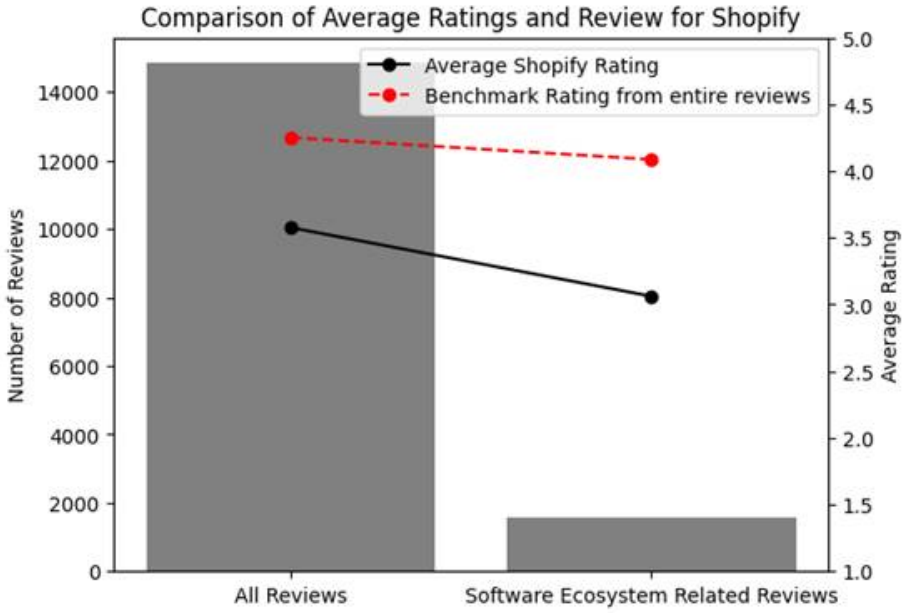

Figure 4.9: Shopify Review Comparison

The categories of platform review issues and the proportion of Shopify Reviews (before applying sentiment-based filtering) belonging to each category and the rating comparison with overall findings can be observed in Table 4.2:

Category	**Review Proportion**	**median Rating**	**Benchmark Rating**
0	30.70%	3.27	4.46
1	17.67%	2.37	4.14
2	15.61%	3.84	4.33
3	2.57%	2.20	2.46
4	6.17%	1.53	2.96
5	6.17%	3.00	3.16
6	21.10%	3.33	3.95

Table 4.2: Shopify Reviews Distribution

4.3.2 Xero

Xero constituted over 5432 reviews from the 4 sources, out of which over 636 reviews were about Software Ecosystems. It could be observed that the SECO-related review ratings in Xero were lower than all-inclusive Xero reviews. The median rating for Xero reviews across all reviews was 4.09/5, the median rating from Xero's SECO reviews was 3.45/5 and the benchmark rating for SECO reviews across all platforms in the study is 3.85/5. It can be summarized in Figure 4.10.

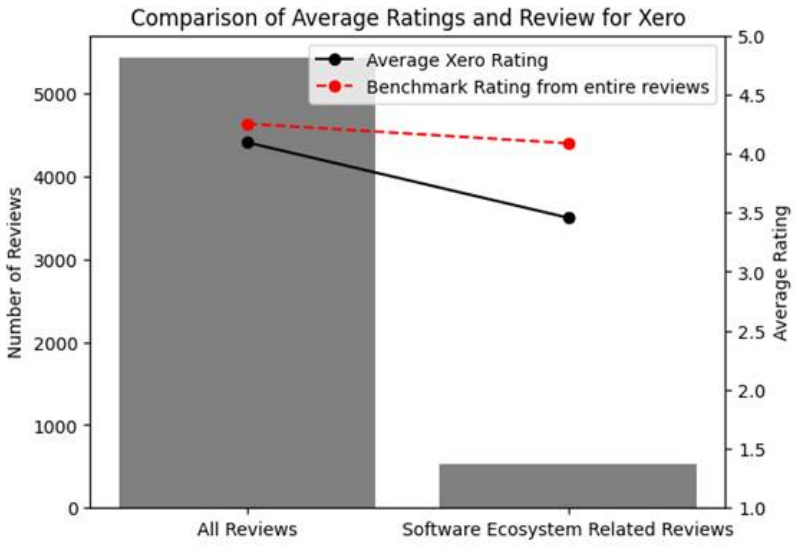

Figure 4.10: Xero Review Comparison

The categories of platform review issues and the proportion of Xero Reviews (before applying sentiment-based filtering) belonging to each category and the rating comparison with overall findings can be observed in Table 4.3

Category	Review Proportion	median Rating	Benchmark Rating
0	42.21%	3.64	4.46
1	29.42%	3.05	4.14
2	9.25%	3.78	4.33
3	2.64%	3.21	2.46
4	3.58%	2.74	2.96
5	1.70%	3.11	3.16
6	10.19%	3.94	3.95

Table 4.3: Xero Reviews Distribution

The comparison between Shopify and Xero reveals various aspects of their platform performance. Xero demonstrates a higher proportion of integration-related issues in its reviews compared to Shopify, with median ratings of 3.64/5 and 3.27/5, respectively. Both platforms, however, fall short of benchmark integration ratings, suggesting room for enhancement. Customer support ratings also indicate room for improvement, with Shopify's median at 2.37/5 and Xero's at 3.05/5. Design and complexity ratings show Shopify at 3.84/5 and Xero at 3.78/5, both below the benchmark. Privacy and security concerns yield median ratings of 2.20/5 for Shopify and 3.21/5 for Xero. Both platforms need to bolster these aspects. Similarly, in cost and pricing, both platforms face challenges, with Shopify's median rating significantly lower. In terms of performance and compatibility, Shopify scores 3.00/5, while Xero scores 3.11/5. Both platforms require enhancements in this domain as well.

4.3.3 Emerging Themes

In addition to analysis of reviews and rating distribution, I manually read through approximately 200 randomized reviews from both platforms. There were 4 common end-user complaints in addition to the categorical challenges discussed in the previous section.

Insufficient customization options: Many users have complained about the limited customization options provided by Shopify and Xero, which makes it difficult for businesses to tailor the platforms to their specific needs. For example, users wanted to customize their invoice template but found it difficult in doing so. This lack of customization options can limit the functionality of the platforms and frustrate users.

Feature Requests: Another common complaint among users is the lack of advanced features provided by Shopify and Xero. Some users commented that they require more advanced reporting and analytics features to manage their businesses effectively but find that the platforms do not provide these options to the extent that they want. This can limit the platforms' appeal to more sophisticated companies that require more robust features.

Glitches and technical issues: Users have also reported experiencing glitches and technical issues while using Shopify and Xero, which can disrupt their workflows and cause frustration. For example, a user might experience delays

while syncing data between Shopify and Xero, or encounter errors while trying to process payments. These technical issues can decrease the reliability and usability of the platforms.

Limited support for international businesses: Although few in numbers, some users have noted that Shopify and Xero provide limited support for international businesses. For example, the platforms might not support local tax laws or currencies, which can make it difficult for businesses operating in different countries to use the platforms effectively. This can limit the platforms' growth to businesses with a global presence.

Based on the analysis of user feedback, both Shopify and Xero have received mixed reviews across different categories. Both platforms have areas that need improvement, including integration, customer support, design and complexity, privacy and security, cost and pricing, and performance and compatibility. However, it is worth noting that the benchmark ratings for SECO reviews across all platforms in the study are directly proportional to these two platforms, indicating that these discussed issues are common concerns for all SECO platforms.

4.4 RQ2: Review vs Response Sentiment

In this section, I present findings from developer responses to SECO-related user feedback. This part of the study does not filter out any reviews, unlike the previous section where a different analysis method was opted for. Figure 4.11 summarizes the response rate and sentiment types (positive, neutral, negative) comparison between SECO-related user feedback and developer responses to it in each category of reviews.

I found that overall, only **20.87** percent of SECO-related user feedback is responded to by platform developers, while on average, the overall responses to all types of (general) reviews were 32.35 percent (excluding the review sources where developer response was not a part of the original dataset/was unavailable entirely). The category of reviews with the highest response rate was Customer Service where 24.75 percent of reviews were responded to despite not having the highest proportion of reviews, whereas the category with the least responses was Design and Complexity with only 16.30 percent despite having a relatively higher proportion of reviews.

Having applied TextBlob sentiment scores to all available SECO reviews and responses, as discussed in Chapter 3, it can be observed that the majority of the reviews

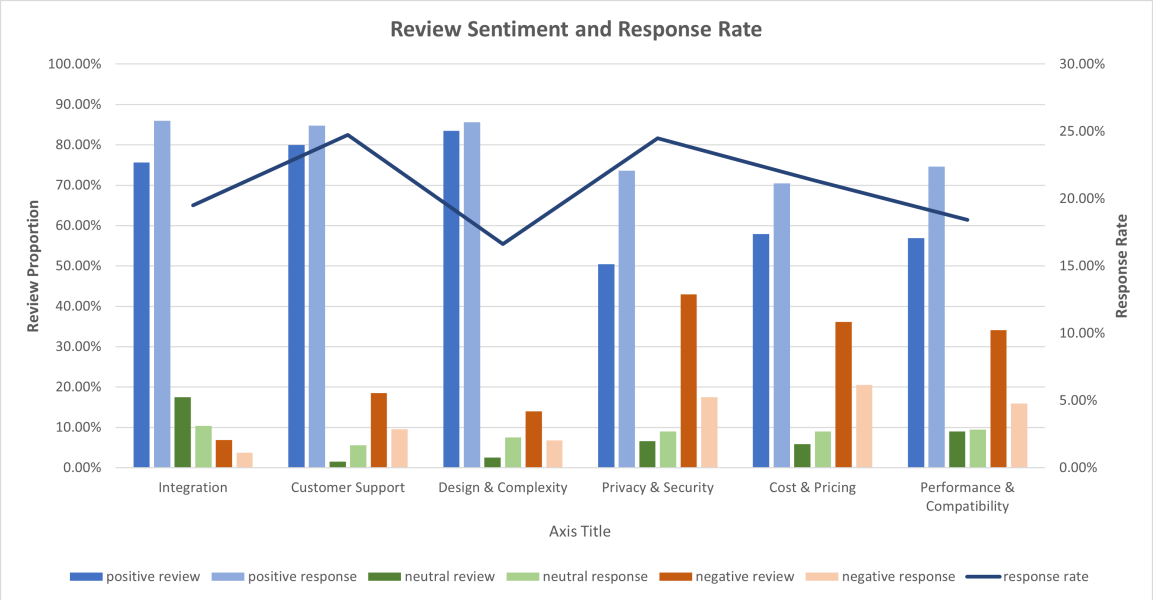

Figure 4.11: Response Rate and Sentiment Comparison

were marked positive across all categories, for which the average response was equally positive. Negative reviews remain very low in the categories with an overall high volume of reviews and average rating as discussed previously. But interestingly, the categories 'cost and pricing' and 'privacy and security', having relatively higher response rates, had the most negative reviews and relatively negative responses. This indicates that the reviews in these categories are critical, and the responses are more likely to be replied to despite not being warm and positive. Most reviews and responses seem to be polarized with either positive or negative comments, as the frequency of neutral reviews remains very low.

Finding 8: SECO-related reviews receive fewer replies compared to general reviews, resulting in a response rate of only 20.87%. While most of the SECO reviews and responses are not negative, an escalation in negative reviews corresponds to a proportional rise in negative responses.

4.5 RQ3: Change in SECO review numbers over time

In this section, I present findings by looking at the change in SECO-related review numbers over time. I mapped the reviews over time to analyze how the number of SECO reviews changed. I kept reviews only from 2013 January to map as the reviews scraped from before 2013 were insignificant in number and hence analyze the reviews from 2013 January to 2022 December. The reviews from 2023 were discarded as there were not enough reviews for the ongoing year. The reviews are then grouped by month and the number of reviews in each month is counted. The median count is calculated for all categories.

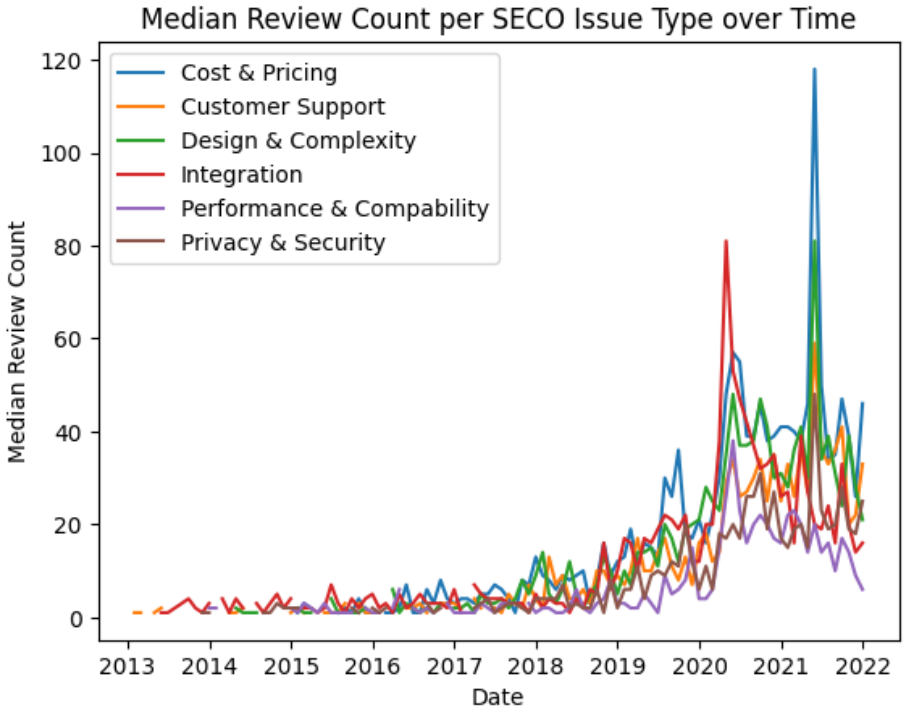

Figure 4.12: Change in SECO reviews over time

We can observe from Figure 4.5 that there is a significant rise in software ecosystem reviews in the last decade. It can be observed that the reviews regarding SECOs started growing significantly from 2016 onwards. From 2015, the reviews have not fallen lower than the monthly median value of 141 reviews (2013-2022 inclusive), indicating the rising interest of users in interacting with software platforms and hence giving increased user feedback.

The number of SECO reviews increased from 51 in 2013 to 4,610 in 2022, with the highest growth occurring between 2016 and 2020. In 2016, the number of reviews grew by 111.76 percent compared to the previous year, while in 2019, the number of reviews grew by 110.82 percent compared to the previous year. In 2020, the growth rate went to a 130.08 percent increase from 2019, but it declined in 2022 with a -26.75 percent growth rate compared to the previous year. The average growth rate from 2018 to 2022 was 258.11 percent. This calculation is based on the comparison of the total count of SECO reviews in the last 5 years with the review count in 2018. From our interviews, we confirmed that platform organizations, in 2020, faced an increasing demand for integration tools and customer support during the COVID-19 pandemic[2]. Overall, there has been a significant increase in the volume of SECO reviews over the years, with the SECO feedback volumes starting to skyrocket since 2016.

Finding 9: In the last 5 years, SECO-related user feedback on average has grown by over 228.57%. More recently, the demand for integration platforms has grown significantly due to COVID-19-induced software needs in businesses and users."

4.6 RQ4: Recommended Mitigation Strategies

In this section, I present findings from my interviews with platform owners. The themes discussed in this section is articulated in the form of strategies and recommendation to mitigate challenges faced by the user of software ecosystems having aggregated the analysis from all four interviews with senior members of keystone organizations.

4.6.1 API First Mentality

Application Programming Interface (API) first development is a strategy that focuses on building the API first before allowing third-party developers to make an integration request. The API is the interface that allows third-party developers to integrate their applications with the platform [42]. This prevents organizations from having to implement one-off integration specific to the developer request. For example, the VP of Engineering from P2 said *"...small startups have an API first mentality. It's*

[2]https://www.who.int/emergencies/diseases/novel-coronavirus-2019

in the DNA of the company that they're building an API so that they don't run into one-off issues.", which potentially addresses the most talked about API-related end-user concerns such as "lacks integration". The following recommendations were found to be useful and deemed essential in streamlining the integration process, improving customer experience, and ensuring the success of the software ecosystem:

- Authentication and Integration: The interviewees recommend using authentication methods that are consistent with customer preferences and outsourcing the entire authentication process to a third-party authentication provider to make it seamless for customers to integrate.
- Burdened Integration: The interviewees recommend separating work streams when integrating with various systems such as HR systems and avoiding trying to do all integration at once to avoid problems.
- API Integration: The interviewees highlight the challenge of deciding whether to invest in APIs or continue building one-offs and recommend advocating for APIs during the company's road map and quarterly planning.
- API-First Mentality: The interviewees recommend adopting an API-first mentality when building software ecosystems to ensure that the company builds an API, preventing issues that entrepreneurs face when they believe they are building apps.
- API vs One-Off Integration: The interviewees highlight the importance of separating design and implementation from operations, which forces them to think about APIs more and avoid taking shortcuts during the integration process.

Strategy 1: The importance of an API-first mentality in software ecosystems cannot be undermined. By prioritizing API development and adhering to recommended authentication methods and integration strategies, platform owners can enhance customer experience, streamline integration processes, and address user and developer needs, ultimately leading to the success of the software ecosystem.

4.6.2 User & Developer Communities

Mitigating customer support and other end-user problems in a software ecosystem requires actively engaging the user community, supporting developers, continuously improving the platform, and fostering collaboration and partnerships. These strategies help address issues, enhance the user experience, and align with evolving integration requirements, as quoted by P4's CTO, *"..an ecosystem doesn't thrive if there's no community for all the stakeholders.."*. These recommendations aim to enhance user retention, gather valuable feedback, and provide developers with the necessary resources for high-quality integration:

- Engaging User Community: Platform owners should actively engage with their user community to build a strong relationship and encourage continued platform usage, even in the presence of competitors. This engagement can help with user retention and provide valuable feedback for future platform development.
- Supporting Developers: Platform owners should prioritize providing developers with comprehensive documentation and resources to facilitate the development of high-quality integrations. By offering robust support, platform owners can help developers create seamless integrations that minimize issues for users.
- Continuous Improvement: The platform owners recognize the importance of continually improving their offerings based on user and developer feedback. This iterative approach allows them to address pain points, enhance features, and refine the overall platform experience to better serve their community.
- Collaboration and Partnerships: Platform owners should actively seek collaboration and partnerships with developers, seeking their input and involvement in the platform's evolution. By involving developers in decision-making processes, platform owners can ensure that the platform meets their needs and aligns with their integration requirements.

Strategy 2: Mitigating end-user problems in a software ecosystem requires actively engaging the user community, supporting developers, continuously improving the platform, and fostering collaboration and partnerships. These strategies help address issues, enhance the user experience, and align with evolving integration requirements.

4.6.3 Third-party App Control

Platform owners should mitigate security and financial risks and issues in their ecosystem by implementing a strict vetting process, continuously monitoring and auditing third-party apps, incentivizing safe and high-quality apps through pricing strategies, and providing developer support and resources. Platform P3's advocate says *"If somebody had essentially abandoned all supported their app and they would be removed from our marketplace"* which ensures compliance and monitoring in the marketplace. The following strategies can be opted to address challenges in software ecosystems:

- Strict Vetting Process: Platform owners enforce a comprehensive vetting process for all third-party apps to ensure compliance with guidelines and standards. This process involves reviewing the app's source code, features, and security measures to identify and remove any potentially malicious or unsafe apps.
- Continuous Monitoring and Auditing: Platform owners prioritize the ongoing monitoring and auditing of third-party apps to detect and eliminate any apps that may have bypassed the vetting process. This proactive approach helps maintain a secure app ecosystem and protects users from potential risks.
- Incentivizing Safe and High-Quality Apps: Platform owners employ pricing strategies that encourage developers to create safe and high-quality apps. By not charging a commission on transactions made through the apps, platform owners incentivize developers to prioritize app quality and user safety.
- Developer Support and Resources: Platform owners provide developers with comprehensive guidelines, documentation, and tools to help them build apps that comply with platform standards. By offering support and resources, platform owners empower developers to create robust and user-friendly apps that enhance the overall platform experience.

Strategy 3: Platform owners effectively mitigate security risks in the app ecosystem by implementing a strict vetting process, continuously monitoring and auditing third-party apps, incentivizing safe and high-quality apps through pricing strategies, and providing developer support and resources. These strategies ensure compliance, enhance user safety, and promote the development of secure and user-friendly apps within the platform.

4.6.4 Feedback-driven approach

In order to effectively mitigate design, complexity, and performance issues, adopting a feedback-driven approach is a valuable strategy for platform owners. As mentioned by the CTO of P4 *"We monitor user interactions within the apps. We get notices of, like rage clicks, things like that, where they go."*, implementing tracking tools, actively soliciting and carefully prioritizing feedback, incorporating user and developer input into the development process, and maintaining transparent communication channels are advisable. The following recommendations emerged from the discussions:

- Implementation of tracking tools: By utilizing tracking tools like "rage click tracking," platform owners can gain insights into user behavior and identify pain points or areas for improvement within the platform.
- Active solicitation of feedback: Platform owners should proactively seek feedback from both users and developers, providing accessible and user-friendly channels for communication. This can include feedback forms, surveys, or dedicated forums for discussions.
- Incorporation of user and developer feedback: Valuable insights gathered from users and developers should inform the development process, leading to the implementation of new features and functionalities that better align with their needs and expectations.
- Communication of changes and updates: Platform owners should maintain transparent communication channels to inform the community about the changes made based on user and developer feedback. Regular updates and release notes can help build trust and keep stakeholders engaged.

Strategy 4: By implementing tracking tools, actively soliciting feedback, carefully reviewing and prioritizing feedback, incorporating user and developer input into the development process, and maintaining transparent communication channels, platform owners can effectively enhance their platform based on user and developer needs. These strategies foster a user-centric approach, improve user satisfaction, and promote continuous platform improvement.

4.6.5 Cross Platform Development

Platform owners should prioritize cross-platform development and utilize progressive web apps (PWAs) to enhance the platform's accessibility and provide a consistent user experience across different devices. To quote P1's CTO, *"We would consider like a cross-platform Progressive Web App To make everything work with mobile devices across the board"*, extending the platform's reach and maintaining competitiveness through cross-platform development, platform owners can attract a wider audience and mitigate platform-specific user issues. The following recommendations emerged from the discussions:

- Consider cross-platform development: Platform owners should actively consider cross-platform development when building new applications and features. This approach enables the platform to be accessible on various devices, including mobile and desktop, catering to a broader user base.
- Utilize progressive web apps: Progressive web apps (PWAs) can serve as a valuable tool for achieving cross-platform compatibility. By developing PWAs, platform owners can provide a consistent and seamless user experience across different platforms and devices, enhancing user engagement and satisfaction.
- Extend the platform's reach: Cross-platform development can effectively extend the platform's reach to a wider audience. By accommodating multiple platforms, such as iOS, Android, and desktop operating systems, platform owners can attract new users and developers who prefer different devices or operating systems.
- Maintain competitiveness: Embracing cross-platform development is crucial for platform owners to remain relevant and competitive in a rapidly evolving technology landscape. By delivering a cross-platform solution, platform owners can meet evolving expectations and differentiate themselves from competitors.

Strategy 5: Platform owners should prioritize cross-platform development and utilize progressive web apps (PWAs) to enhance the platform's accessibility and provide a consistent user experience across different devices. By extending the platform's reach and maintaining competitiveness through cross-platform development, platform owners can attract a wider audience and meet the evolving expectations of users, ensuring long-term success in the technology landscape.

4.6.6 Documentations & Guidelines

Platform owners should prioritize comprehensive documentation, accessibility, quality and security guidelines, and developer support in optimizing the utilization of the platform's API. By providing clear instructions, easy access, and assistance to developers, platform owners can foster a collaborative and productive developer community, resulting in high-quality integration and improved platform success, as P3's advocate said, *"It starts with having really clear ATP documentation. I think having that publicly available, they start first ideating about the process."*. The following recommendations can be prioritized ensure the quality and security of applications:

- Comprehensive documentation: Platform owners must provide comprehensive documentation that thoroughly outlines the capabilities of their API and provides clear instructions on how to utilize it effectively. The documentation should cover key functionalities, use cases, and best practices. It should be easily accessible, well-structured, and regularly updated to reflect any changes or additions to the API.

- Accessibility and organization: Documentation should be easily accessible to developers, preferably through a centralized and user-friendly platform. Platform owners should invest in designing an intuitive and searchable documentation interface that allows developers to quickly find the information they need. Clear organization, including logical sections enhances the usability of the documentation.

- Guidelines for quality and security: Alongside documentation, platform owners should establish a set of guidelines that outline the quality and security standards for third-party applications. These guidelines should specify coding practices, data handling protocols, authentication requirements, and other essential aspects to ensure consistency and protect user data.

Strategy 6: Platform owners should prioritize comprehensive documentation, accessibility, quality and security guidelines, and developer support in optimizing the utilization of the platform's API. By providing clear instructions, easy access, and assistance to developers, platform owners can foster a collaborative and productive developer community, resulting in high-quality integration and improved platform success.

4.6.7 User Data Management

By providing transparent policies, establishing efficient incident response processes, prioritizing user privacy, and adhering to relevant regulations, platform owners can foster trust, protect user information, and mitigate potential risks associated with data breaches or non-compliance. For example, P1's CTO said, *"We don't hold the client information in our databases for any longer than, you know, The lifetime of an order which is the lifecycle of the data."*, and P2's VP of engineering mentioned *"Good user data management practice such as streamlined SSO authentication is a good practice to resolve integration as well as privacy issues"*, meaning platform owners must ensure that third-party applications delete user data when it is no longer needed, and secure authentication practices must be implemented. The following recommendations are proposed:

- Clear Guidelines of Data Life-cycle: Platform owners must provide clear and transparent guidelines outlining data retention and deletion policies. These guidelines should specify the duration for which user data can be held and emphasize the importance of deleting data when it is no longer necessary.
- Incident Response Protocols: Platform owners should have well-defined incident response protocols in place to handle data breaches or other incidents that may compromise user data.
- User Privacy Protection: Timely data deletion plays a crucial role in protecting user privacy. By deleting user data when it is no longer needed, platform owners demonstrate respect for user preferences and safeguard their personal information from unauthorized access or misuse.
- Compliance with Regulations: Platform owners must ensure compliance with relevant data protection regulations, such as General Data Protection Regulation[3] (GDPR) or California Consumer Privacy Act[4] (CCPA).

Strategy 7: By providing transparent policies, establishing efficient incident response processes, prioritizing user privacy, and adhering to relevant regulations, platform owners can foster trust, protect user information, and mitigate potential risks associated with data breaches or non-compliance.

[3]https://gdpr-info.eu/
[4]https://oag.ca.gov/privacy/ccpa

Chapter 5

Discussion

5.1 Synthesis

To the best of our knowledge, this thesis presents the first large-scale investigation of end-user perspectives on software ecosystems. Through the implementation of rigorous analytical techniques, a comprehensive examination of the various factors within software ecosystems that significantly affect the user experience was conducted. Integration issues, customer support, the complexity of design and user interface, issues with privacy and security, pricing issues, and platform compatibility were introduced as problem areas in software ecosystems, each having its own lists of nested challenges. The results discussed several challenges encountered by users, which offer valuable insights for researchers and practitioners alike. Moreover, the thesis also illuminated the strategies to be adopted and recommendations to be opted for by organizations in their attempts to tackle these issues. The thesis recommends having an API-first mentality, opting for a feedback-driven approach, developing a dynamic user and developer community, having control over third-party apps, opting for cross-platform development, providing comprehensive documentation and guidelines, and proper management of user data as strategies for software platforms to mitigate problems faced by their users. Additionally, the thesis also discussed how SECO-related reviews are replied to less compared to other feedback by users, demonstrating the need for caution in monitoring user feedback in SECO, especially as the thesis found that SECO reviews are dramatically increasing in number. As such, this study offers valuable implications for future research, as well as for practitioners and software ecosystem platform companies seeking to optimize the user experience.

5.2 Classifying SECO vs Non-SECO reviews

In order to classify Software Ecosystem (SECO) reviews from non-SECO reviews, a novel classification approach was developed, contributing methodologically to the field. The methodology involved employing pair coding with Cohen Kappa's coefficient as a measure of inter-rater agreement, which exceeded a threshold of 0.81, indicating robust agreement. Within a dataset of 6000 reviews, 500 reviews were randomly selected for manual assessment. These reviews were evaluated in pairs, with 500 SECO-related reviews being identified over five iterations, progressively increasing the agreement score. In total, 848 SECO-related reviews were successfully identified through this pair coding process.

The classification was further refined by utilizing SECO-related keywords such as "platforms," "integration,", "API", "ecosystems," "plugins," and "sync." These keywords were instrumental in distinguishing SECO reviews from non-SECO reviews and were manually validated based on contextual understanding. For instance, reviews containing contextual clues such as "integration issues," "third-party app name," and "plugin name" were classified as SECO-related. Conversely, reviews that lacked explicit SECO-related terminology, such as those discussing poor app performance or usability issues, were classified as non-SECO reviews. This nuanced approach effectively separated SECO reviews from non-SECO reviews, offering a valuable methodological contribution to review classification.

This classification methodology holds significant implications for recommender systems and future research endeavors. The ability to accurately distinguish SECO reviews from non-SECO reviews provides a robust foundation for developing tools and recommender systems in platforms, such that keystones get to prioritize and focus on important feedback from the crowd of user voices. By channeling SECO-related reviews to users interested in software ecosystem aspects and directing non-SECO reviews to those seeking general usability insights, the overall user experience can be greatly improved. Furthermore, this methodological innovation make the way for advanced research in sentiment analysis, user experience, and software ecosystem dynamics. Researchers can leverage and build on this approach to analyze trends and patterns specific to SECO-related discussions, thereby gaining deeper insights into user preferences and contributing to the evolution of software ecosystems.

5.3 Implications for SECO Researchers

More effective integration techniques and tools should be developed, and new approaches to API design, testing, and documentation should be explored by future research to help users avoid common errors and achieve successful integrations more quickly and easily. Customer support techniques and strategies should be focused on developing more effectively, and the use of reliable technologies for chatbots or AI-powered systems that can help customers resolve issues more quickly and easily should be explored. At the same time, the tradeoff between having a human conversation over a chatbot, and identifying the underlying causes of dissatisfaction with customer support should be examined to help software providers improve their support services. Factors that contribute to a "bad user interface" in a software ecosystem and how they can be improved should be identified by studies, and the relationship between visual design, usability, and user experience should be explored. How different types of interfaces affect user experience and satisfaction, and the impact of software interface design on productivity and user engagement in SECOs should also be investigated. Specific security and privacy concerns that users have when interacting with software ecosystems should be explored, ways that platforms can build trust with their users should be examined, and the best practices for user authentication and security measures that balance user convenience and security should be determined. The reasons behind the pricing structures of SECOs should be verified and validated, the costs associated with developing and maintaining a software ecosystem platform should be examined, the pricing strategies employed by successful SECOs should be analyzed, and the relationship between cost and value, and how users perceive the value they receive in exchange for the fees they pay should be explored by researchers. Finally, the role of different device types and operating systems that cause issues within software ecosystems should be understood by more studies. Solutions that address these pain points from the findings and improve the overall user experience can be developed by practitioners and platform companies. The specific factors that contribute to performance issues such as crashes, and loading times, as well as ways to optimize the user interface for desktop and mobile devices should be identified by these studies in the future. Standardized frameworks and tools for platforms in order to monitor, manage, and address end-user concerns in software ecosystems should be proposed by further research.

5.4 Implications for Platform Organizations

Platforms should consider implementing the proposed strategies from this research in their organizations. Serious consideration must be given to an API-first mentality, as it seems to be one of the few viable strategies for platform growth and sustainability. The overall integration experience for users should be improved by platform organizations, standardized integration frameworks or best practices for integration should be established in collaboration with software providers, and tools and resources that can help users troubleshoot integration issues and resolve them quickly should be invested in. The overall customer support experience for users should be improved, standardized support frameworks that can be used by software providers to ensure consistent and high-quality support services should be developed, and resources and training programs to help software providers improve their customer support capabilities should be provided. The usability and performance of their software interfaces should be improved by Keystone organizations, usability testing should be conducted, feedback should be gathered from users, iterative improvements based on this feedback should be made, and software developers should be encouraged to prioritize user interface design and consider user feedback when making improvements to their software. The findings of this study should be used by organizations to improve their overall platform security and privacy measures, robust security measures that protect user data from unauthorized access and use should be developed, policies and procedures that promote user trust should be implemented, and users should be provided with more control over their data by implementing tools such as privacy settings and data deletion options. Flexible pricing models that allow users to choose the features they need and pay only for what they use can be considered by companies, pricing and costs-related information policies should be improved to ensure that users can make payments with trust, obtain refunds without hassle, and user feedback should be analyzed to identify areas where costs can be reduced or pricing can be made more transparent and user-friendly to avoid losing clients to competitors. Finally, platform organizations should improve their software's reliability and functionality to address quality-related issues from the findings. Compatibility with different operating systems and devices should be ensured, a mobile-first approach [44] should be opted for where applicable, robust security and privacy measures should be implemented, regular updates and bug fixes should be provided, and reliable customer support should be offered, all of which boost platform health [53].

5.5 Threats to Validity

5.5.1 Internal Validity

Threats to internal validity in this study include potential biases in the data collection and data analysis processes. The data was collected from various sources, and the quality and accuracy of the data may vary depending on the source. Moreover, the sample of user feedback reviews analyzed may not be representative of the entire population of software ecosystem users, that is web-based and desktop-based platforms, potentially leading to sampling bias as the majority of the reviews considered in the study were mobile application reviews.

5.5.2 External Validity

The external validity of this study may be limited due to the use of data scraped from various sources such as the Play Store, App Store, Shopify Store, and Trustpilot. Although the data is extensive, it may not represent all areas of software ecosystem platforms and mobile applications such as open-source software (OSS) [74]. Therefore, the findings may not be generalized to all types of software ecosystems. Similarly, the semi-structured interviews with six keystone platform developers and owners may not be representative of all platform organizations. Additionally, the findings may not apply to users who do not leave feedback or those who have different experiences with software ecosystems.

5.5.3 Construct Validity

The identification of software ecosystem-related issues was crucial to the analysis which is a potential threat to the construct validity. However, I took several steps to mitigate it. The pair-coding approach with inter-rater agreement was the most ideal way of initially classifying what a SECO review is as it contained a blend of analysis of context and looking for specific keywords. Also, manually investigating the results of the automated classification to ensured accuracy alongside an optimal evaluation results of the classifier.

Chapter 6

Conclusions

The thesis presents a comprehensive analysis of user reviews across a range of software ecosystem platforms with the goal of enhancing the user experience. By answering four research questions, I identified six broad categories and problems in software ecosystems, analyzed the sentiment of developer responses to user feedback, observed the change in the number of SECO reviews over time, and investigated how keystone organizations address software ecosystem-related user feedback so that generalizable strategies in mitigating those challenges can be recommended. The thesis focused on analyzing user feedback to help platforms improve their overall user experience. The study examined a range of discussion topics and evaluated the response from developers to understand how they addressed the identified issues. By tracking the number of SECO reviews over time, the study revealed the growing importance of studying user feedback. It emphasized the need to address it for the sustainability of software ecosystems. Interviews with key platform developers and owners provided valuable insights into their strategies for handling user feedback, leading to the identification of best practices that other organizations can implement.

Overall, this thesis provides a contribution to the existing knowledge of end-user concerns and the industrial perspective on software ecosystems. By identifying key issues and providing recommendations in several aspects of a SECO platform, the findings can guide platforms in designing and fostering better ecosystems. The recommendations can inform industry standards and improve the overall quality of software ecosystems. Lastly, the methodology and techniques used in this study can serve as a blueprint for future research in this space. Future work should focus on empirically validating and adding to the identified challenges and implementing the proposed strategies to validate the recommendations and formalize the strategies.

Additionally, future work could include expanding the scope of the study to include more ecosystem platforms and user reviews. The two machine learning classifiers could be further refined to improve its accuracy in first identifying what kind of feedback is a SECO-related feedback, and secondly in categorizing SECO reviews according to the proposed problem areas. Additional problem categories could be identified and analyzed. The effectiveness of the mitigation strategies suggested could be evaluated through implementation and user feedback. Longitudinal studies could be conducted to track the changes in user challenges and developer responses over time. Finally, the impact of external factors such as the COVID-19 pandemic on SECOs could be further explored.

Bibliography

[1] Martin Adam, Michael Wessel, and Alexander Benlian. Ai-based chatbots in customer service and their effects on user compliance. *Electronic Markets*, 31(2):427–445, 2021.

[2] Esma Aïmeur, Oluwa Lawani, and Kimiz Dalkir. When changing the look of privacy policies affects user trust: An experimental study. *Computers in Human Behavior*, 58:368–379, 2016.

[3] Wajdi Aljedaani, Furqan Rustam, Mohamed Wiem Mkaouer, Abdullatif Ghallab, Vaibhav Rupapara, Patrick Bernard Washington, Ernesto Lee, and Imran Ashraf. Sentiment analysis on twitter data integrating textblob and deep learning models: The case of us airline industry. *Knowledge-Based Systems*, 255:109780, 2022.

[4] Sebastien Andreo and Jan Bosch. Api management challenges in ecosystems. In *Software Business: 10th International Conference, ICSOB 2019, Jyväskylä, Finland, November 18–20, 2019, Proceedings 10*, pages 86–93. Springer, 2019.

[5] Appdirect.com. What is a software ecosystem? https://www.appdirect.com/resources/glossary/software-ecosystem. Accessed: 2023-05-01.

[6] Jakob Axelsson, Efi Papatheocharous, and Jesper Andersson. Characteristics of software ecosystems for federated embedded systems: A case study. *Information and Software Technology*, 56(11):1457–1475, 2014.

[7] Alfred Baars and Slinger Jansen. A framework for software ecosystem governance. In *Software Business: Third International Conference, ICSOB 2012, Cambridge, MA, USA, June 18-20, 2012. Proceedings 3*, pages 168–180. Springer, 2012.

[8] Ayan Banerjee and Tapan Chowdhury. Reviewing system using exploratory data analysis and ensemble machine learning algorithms. In *2021 IEEE 2nd International Conference on Technology, Engineering, Management for Societal impact using Marketing, Entrepreneurship and Talent (TEMSMET)*, pages 1–6. IEEE, 2021.

[9] Olavo Barbosa12 and Carina Alves. A systematic mapping study on software ecosystems. 2011.

[10] Rahul C Basole and Hyunwoo Park. Interfirm collaboration and firm value in software ecosystems: Evidence from cloud computing. *IEEE Transactions on Engineering Management*, 66(3):368–380, 2018.

[11] Ítalo Belo and Carina Alves. How to create a software ecosystem? a partnership meta-model and strategic patterns. *Information*, 12(6), 2021.

[12] Jan Bosch. Architecture challenges for software ecosystems. In *Proceedings of the fourth European conference on software architecture: companion volume*, pages 93–95, 2010.

[13] Christoph Burkard, Tobias Draisbach, Thomas Widjaja, Peter Buxmann, et al. Software ecosystems: Vendor-sided characteristics of online marketplaces. In *GI-Jahrestagung*, page 421, 2011.

[14] Christoph Burkard, Thomas Widjaja, and Peter Buxmann. Software ecosystems. *Business & Information Systems Engineering*, 4:41–44, 2012.

[15] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[16] Victoria Clarke, Virginia Braun, and Nikki Hayfield. Thematic analysis. *Qualitative psychology: A practical guide to research methods*, 3:222–248, 2015.

[17] Simone da Silva Amorim, Félix Simas S Neto, John D McGregor, Eduardo Santana de Almeida, and Christina von Flach G. Chavez. How has the health of software ecosystems been evaluated? a systematic review. In *Proceedings of the XXXI Brazilian Symposium on Software Engineering*, pages 14–23, 2017.

[18] Daniela Damian, Johan Linåker, David Johnson, Tony Clear, and Kelly Blincoe. Challenges and strategies for managing requirements selection in software ecosystems. *IEEE Software*, 38(6):76–87, 2021.

[19] Devblobs.microsoft.com. The importance of feedback in software development. https://devblogs.microsoft.com/bharry/the-importance-of-feedback-in-software-development/. Accessed: 2023-05-01.

[20] Dora Dzvonyar, Stephan Krusche, Rana Alkadhi, and Bernd Bruegge. Context-aware user feedback in continuous software evolution. In *Proceedings of the International Workshop on Continuous Software Evolution and Delivery*, CSED '16, page 12–18, New York, NY, USA, 2016. Association for Computing Machinery.

[21] Tatiana Ermakova, Annika Baumann, Benjamin Fabian, and Hanna Krasnova. Privacy policies and users' trust: does readability matter? In *AMCIS*, 2014.

[22] Robert Evertse, Abel Lencz, Tea Šinik, Slinger Jansen, and Lamia Soussi. Is your software ecosystem in danger? preventing ecosystem death through lessons in ecosystem health. In *Agile Processes in Software Engineering and Extreme Programming–Workshops: XP 2021 Workshops, Virtual Event, June 14–18, 2021, Revised Selected Papers 22*, pages 96–105. Springer, 2021.

[23] Iris Figalist, Christoph Elsner, Jan Bosch, and Helena Holmström Olsson. Scaling agile beyond organizational boundaries: Coordination challenges in software ecosystems. In *Agile Processes in Software Engineering and Extreme Programming: 20th International Conference, XP 2019, Montréal, QC, Canada, May 21–25, 2019, Proceedings 20*, pages 189–206. Springer International Publishing, 2019.

[24] Asbjørn Følstad and Petter Bae Brandtzaeg. Users' experiences with chatbots: findings from a questionnaire study. *Quality and User Experience*, 5(1):3, 2020.

[25] Armstrong Foundjem. Cross-distribution feedback in software ecosystems. In *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, ICSEW'20, page 723–724, New York, NY, USA, 2020. Association for Computing Machinery.

[26] Oscar Franco-Bedoya, David Ameller, Dolors Costal, and Xavier Franch. Measuring the quality of open source software ecosystems using queso. In *Software*

Technologies: 9th International Joint Conference, ICSOFT 2014, Vienna, Austria, August 29-31, 2014, Revised Selected Papers 9, pages 39–62. Springer, 2015.

[27] Cuiyun Gao, Jichuan Zeng, David Lo, Xin Xia, Irwin King, and Michael R. Lyu. Understanding in-app advertising issues based on large scale app review analysis. *Information and Software Technology*, 142:106741, 2022.

[28] Necmiye Genc-Nayebi and Alain Abran. A systematic literature review: Opinion mining studies from mobile app store user reviews. *Journal of Systems and Software*, 125:207–219, 2017.

[29] Bachan Ghimire. Assessing the effectiveness of project management on software engineering. Master's thesis, Collegium Humanum, Warsaw Management University, 12 2021.

[30] Keith Goffin. Customer support: A cross-industry study of distribution channels and strategies. *International Journal of Physical Distribution & Logistics Management*, 1999.

[31] Sanket Kumar Gupta, Bahar Schwichtenberg, and Gregor Engels. A reference architecture for enhanced design of software ecosystems. In *Business Modeling and Software Design: 11th International Symposium, BMSD 2021, Sofia, Bulgaria, July 5–7, 2021, Proceedings 11*, pages 59–77. Springer, 2021.

[32] Hotjar.com. User feedback: 6 best ways to get useful feedback. https://www.hotjar.com/blog/user-feedback/. Accessed: 2023-05-01.

[33] Fang Hou and Slinger Jansen. A systematic literature review on trust in the software ecosystem. *Empirical Software Engineering*, 28(1):8, 2023.

[34] James Howison, Ewa Deelman, Michael J McLennan, Rafael Ferreira da Silva, and James D Herbsleb. Understanding the scientific software ecosystem and its impact: Current and future measures. *Research Evaluation*, 24(4):454–470, 2015.

[35] Hubspot.com. Building a technology ecosystem: What you need to know. https:blog.hubspot.com/website/technology-ecosystem/. Accessed: 2023-05-01.

[36] Clayton Hutto and Eric Gilbert. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Proceedings of the international AAAI conference on web and social media*, volume 8, pages 216–225, 2014.

[37] Pankaj Jalote and Pari Natarajan. The growth and evolution of india's software industry. *Commun. ACM*, 62(11):64–69, oct 2019.

[38] Slinger Jansen. A focus area maturity model for software ecosystem governance. *Information and Software Technology*, 118:106219, 2020.

[39] Slinger Jansen and Ewoud Bloemendal. Defining app stores: The role of curated marketplaces in software ecosystems. In Georg Herzwurm and Tiziana Margaria, editors, *Software Business. From Physical Products to Software Services and Solutions*, pages 195–206, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[40] Slinger Jansen, Sjaak Brinkkemper, and Anthony Finkelstein. Business network management as a survival strategy. In *Software Ecosystems*, pages 29–42. Edward Elgar Publishing, 2013.

[41] Slinger Jansen and Michael A Cusumano. Defining software ecosystems: a survey of software platforms and business network governance. In *Software ecosystems*, pages 13–28. Edward Elgar Publishing, 2013.

[42] Slinger Jansen, Michael A Cusumano, and Sjaak Brinkkemper. *Software ecosystems: analyzing and managing business networks in the software industry*. Edward Elgar Publishing, 2013.

[43] David Johnson, James Tizard, Daniela Damian, Kelly Blincoe, and Tony Clear. Open crowdre challenges in software ecosystems. In *2020 4th international workshop on crowd-based requirements engineering (CrowdRE)*, pages 1–4. IEEE, 2020.

[44] Bohyun Kim. Responsive web design, discoverability, and mobile challenge. *Library technology reports*, 49(6):29–39, 2013.

[45] Eric Knauss, Aminah Yussuf, Kelly Blincoe, Daniela Damian, and Alessia Knauss. Continuous clarification and emergent requirements flows in open-commercial software ecosystems. *Requirements Engineering*, 23:97–117, 2018.

[46] Daniel Lakens. Sample size justification. *Collabra: Psychology*, 8(1):33267, 2022.

[47] J Richard Landis and Gary G Koch. The measurement of observer agreement for categorical data. *biometrics*, pages 159–174, 1977.

[48] Effie Lai-Chong Law, Virpi Roto, Marc Hassenzahl, Arnold POS Vermeeren, and Joke Kort. Understanding, scoping and defining user experience: a survey approach. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 719–728, 2009.

[49] Robyn Longhurst. Semi-structured interviews and focus groups. *Key methods in geography*, 3(2):143–156, 2003.

[50] Walid Maalej and Hadeer Nabil. Bug report, feature request, or simply praise? on automatically classifying app reviews. In *2015 IEEE 23rd international requirements engineering conference (RE)*, pages 116–125. IEEE, 2015.

[51] B Madhurika and D Naga Malleswari. A systematic review on artificial intelligence-based opinion mining models. In *2022 International Conference on Edge Computing and Applications (ICECAA)*, pages 252–257. IEEE, 2022.

[52] Tim A Majchrzak, Andreas Biørn-Hansen, and Tor-Morten Grønli. Progressive web apps: the definite approach to cross-platform development? 2018.

[53] Konstantinos Manikas and Klaus Marius Hansen. Reviewing the health of software ecosystems–a conceptual framework proposal. In *Proceedings of the 5th international workshop on software ecosystems (IWSECO)*, pages 33–44. Citeseer, 2013.

[54] Konstantinos Manikas and Klaus Marius Hansen. Software ecosystems – a systematic literature review. *Journal of Systems and Software*, 86(5):1294–1306, 2013.

[55] Stuart McIlroy, Nasir Ali, Hammad Khalid, and Ahmed E. Hassan. Analyzing and automatically labelling the types of user issues that are raised in mobile app reviews. *Empirical Software Engineering*, 21:1067–1106, 2016.

[56] Preksha Nema, Pauline Anthonysamy, Nina Taft, and Sai Teja Peddinti. Analyzing user perspectives on mobile app privacy at scale. In *Proceedings of the 44th International Conference on Software Engineering*, ICSE '22, page 112–124, New York, NY, USA, 2022. Association for Computing Machinery.

[57] Duc Cuong Nguyen, Erik Derr, Michael Backes, and Sven Bugiel. Short text, large effect: Measuring the impact of user reviews on android app security &

privacy. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 555–569, 2019.

[58] Jakob Nielsen and Rolf Molich. Heuristic evaluation of user interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 249–256, 1990.

[59] Dennis Pagano and Walid Maalej. User feedback in the appstore: An empirical study. In *2013 21st IEEE international requirements engineering conference (RE)*, pages 125–134. IEEE, 2013.

[60] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[61] Thanasis Petsas, Antonis Papadogiannakis, Michalis Polychronakis, Evangelos P Markatos, and Thomas Karagiannis. Rise of the planet of the apps: A systematic study of the mobile app ecosystem. In *Proceedings of the 2013 conference on Internet measurement conference*, pages 277–290, 2013.

[62] Anshul Rani, Deepti Mishra, and Aida Omerovic. Multi-vendor software ecosystem: Challenges from company'perspective. In *Information Systems and Technologies: WorldCIST 2022, Volume 3*, pages 382–393. Springer, 2022.

[63] Seyed Mostafa Razavi, Hossein Safari, Hessam Shafie, and H Rezaei Vandchali. How customer satisfaction, corporate image and customer loyalty are related? *European Journal of Scientific Research*, 78(4):588–596, 2012.

[64] Furqan Rustam, Arif Mehmood, Muhammad Ahmad, Saleem Ullah, Dost Muhammad Khan, and Gyu Sang Choi. Classification of shopify app user reviews using novel multi text features. *IEEE Access*, 8:30234–30244, 2020.

[65] Hinrich Schütze, Christopher D Manning, and Prabhakar Raghavan. *Introduction to information retrieval*, volume 39. Cambridge University Press Cambridge, 2008.

[66] Alexander Serebrenik and Tom Mens. Challenges in software ecosystems research. In *Proceedings of the 2015 European Conference on Software Architecture*

Workshops, ECSAW '15, New York, NY, USA, 2015. Association for Computing Machinery.

[67] Yasir Ali Solangi, Zulfiqar Ali Solangi, Samreen Aarain, Amna Abro, Ghulam Ali Mallah, and Asadullah Shah. Review on natural language processing (nlp) and its toolkits for opinion mining and sentiment analysis. In *2018 IEEE 5th international conference on engineering technologies and applied sciences (ICETAS)*, pages 1–4. IEEE, 2018.

[68] Donna Spencer. *Card sorting: Designing usable categories*. Rosenfeld Media, 2009.

[69] Eduard Alexandru Stoica and Esra Kahya Özyirmidokuz. Mining customer feedback documents. *International Journal of Knowledge Engineering*, 1(1):68–71, 2015.

[70] James Tizard, Tim Rietz, Xuanhui Liu, and Kelly Blincoe. Voice of the users: an extended study of software feedback engagement. *Requirements Engineering*, 27(3):293–315, 2022.

[71] Maria Dolores C Tongco. Purposive sampling as a tool for informant selection. *Ethnobotany Research and Applications*, 5, 2007.

[72] Godwin J Udo. Privacy and security concerns as major barriers for e-commerce: a survey study. *Information management & computer security*, 2001.

[73] Ivo van den Berk, Slinger Jansen, and Lútzen Luinenburg. Software ecosystems: A software ecosystem strategy assessment model. In *Proceedings of the Fourth European Conference on Software Architecture: Companion Volume*, ECSA '10, page 127–134, New York, NY, USA, 2010. Association for Computing Machinery.

[74] Georg Von Krogh and Eric Von Hippel. Special issue on open source software development, 2003.

[75] Emilie Nøss Wangen. What is a software platform & how is it different from a product? HubSpot Blog, June 2023.

[76] Spyros Xanthopoulos and Stelios Xinogalos. A comparative analysis of cross-platform development approaches for mobile applications. In *Proceedings of the 6th Balkan Conference in Informatics*, pages 213–220, 2013.

[77] Yun E Zeng, H Joseph Wen, and David C Yen. Customer relationship management (crm) in business-to-business (b2b) e-commerce. *Information Management & Computer Security*, 11(1):39–44, 2003.

[78] Yujia Zhai, Wei Song, Xianjun Liu, Lizhen Liu, and Xinlei Zhao. A chi-square statistics based feature selection method in text classification. In *2018 IEEE 9th International conference on software engineering and service science (ICSESS)*, pages 160–163. IEEE, 2018.

Appendix A

Reviews Analyzed: Platform List

Table A.1: Platform Categories

Platform	**Category**
Shopify - Your Ecommerce Store	E-commerce
WordPress – Website Builder	Website Development
Slack	Communication Platform
Microsoft SharePoint	Integration Platform
HubSpot CRM: Grow better	CRM Platform
Zendesk Support	CRM Platform
Salesforce	CRM Platform
Wix Owner - Website Builder	Website Development
Ecwid Ecommerce	E-commerce
Shopify Point of Sale (POS)	Payment Platform
Etsy Seller: Manage Your Shop	E-commerce
DHgate-online wholesale stores	E-commerce
Etsy: Custom & Creative Goods	E-commerce
Share Products and Make Money	E-commerce
Shopify Inbox	Communication Platform
WooCommerce	Website Development
KWIKBOX SELLER: Create online	E-commerce
Amazon Seller	E-commerce
WED2C	E-commerce

Continued on the next page

Table A.1 (continued): Platform Categories

Platform	Category
windo - create ecommerce store	E-commerce
Yelp for Business	Integration Platform
eBay - Shop at the Marketplace	E-commerce
Temu	E-commerce
Redbubble	Integration Platform
Wish: Shop And Save	E-commerce
POS System and Stock by Kyte	Payment Platform
Banggood - Online Shopping	E-commerce
Premise - Earn Money for Tasks	E-commerce
Fiverr - Freelance Service	CRM Platform
Mercari: Your Marketplace	E-commerce
PayPal Business	Payment Platform
OurShopee - Online Shopping	E-commerce
Myne Sales & Inventory Manager	E-commerce
Depop - Buy & Sell Clothes App	E-commerce
CJdropshipping	E-commerce
Hive Work	CRM Platform
LetyShops cashback service	E-commerce
Wholee - Online Shopping App	E-commerce
Toluna Influencers	E-commerce
Flipp - Weekly Shopping	E-commerce
Learn Digital Marketing	CRM Platform
Alibaba.com - B2B marketplace	E-commerce
Vipon - Amazon Deals & Coupons	E-commerce
FARFETCH — Designer Shopping	E-commerce
MiniInTheBox Online Shopping	E-commerce
Truelancer: Freelance Work App	CRM Platform
ShopBack - Shop, Earn & Pay	E-commerce
Mailchimp: Marketing & CRM to	CRM Platform
Square Point of Sale: Payment	Payment Platform

Continued on the next page

Table A.1 (continued): Platform Categories

Platform	Category
COIN: Always Be Earning	E-commerce
SalesPlay POS - Point of Sale	Payment Platform
Squarespace: Run your business	Website Development
Nudge - Your Workplace App	Communication Platform
Zoho CRM - Sales & Marketing	CRM Platform
Zoho Social	CRM Platform
Microsoft 365 Admin	Integration Platform
BambooHR	CRM Platform
Chime – Mobile Banking	Payment Platform
Wise	Payment Platform
ServiceDesk Plus SaaS HelpDesk	Integration Platform
SAAS Digital Live	E-commerce
Made-in-China B2B Trade Online	E-commerce
B2B Trade	E-commerce
Dial4Trade: B2B Marketplace -	E-commerce
TradeIndia: B2B Marketplace	E-commerce
JdMart India's B2B Marketplace	E-commerce
ExportersIndia - B2B Directory	E-commerce
Global Sources - online market	E-commerce
Pakat - B2B Wholesale Market	E-commerce
Dukandar 24 : B2B Online Store	E-commerce
Wholesale Box - B2B Latest Fas	E-commerce
Onsight B2B Sales App	E-commerce
Just Business: B2B Network, Gr	E-commerce
Zyapaar - B2B Networking App	E-commerce
Konnectbox: B2B Marketplace	E-commerce
VyaparBharat : B2B Marketplace	E-commerce
Moglix - B2B Marketplace	E-commerce
Buy and sell - Marketplace	E-commerce
CRM by DealerSocket	CRM Platform

Continued on the next page

Table A.1 (continued): Platform Categories

Platform	Category
LEADer CRM Leads Sales Tracker	CRM Platform
Pocket CRM - Customers & Leads	CRM Platform
Personal CRM by Covve	CRM Platform
Bigin by Zoho CRM	CRM Platform
Pipedrive – Sales CRM	CRM Platform
Dex - Rolodex and Personal CRM	CRM Platform
HoneyBook - Small Business CRM	CRM Platform
Salesmate - Sales CRM	CRM Platform
3Sigma Mobile CRM App	CRM Platform
Bitrix24 CRM And Projects	CRM Platform
HelloLeads CRM - Sales Tracker	CRM Platform
Zendesk Sell - CRM, Leads and	CRM Platform
LeadSquared CRM	CRM Platform
CRM Analytics	CRM Platform
API Tester - REST HTTP Client	Integration Platform
Restler - REST API Client	Integration Platform
Android Device Policy	Integration Platform
GitHub	Integration Platform
Termux:API	Integration Platform
Google Admin	Integration Platform
APK Installer	Integration Platform
Termux	Integration Platform
AIDE- IDE for Android Java C++	Integration Platform
monday.com - Work Management	Integration Platform
Microsoft Teams	Communication Platform
alibaba	E-commerce
etsy	E-commerce
fiverr	E-commerce
hubspot	CRM Platform
mailchimp	CRM Platform

Continued on the next page

Table A.1 (continued): Platform Categories

Platform	Category
salesforce	CRM Platform
shopify	E-commerce
squareup	Payment Platform
wix	Website Development
wordpress	Website Development
xero	Integration Platform
zapier	Integration Platform
zendesk	CRM Platform
zoho	CRM Platform
slack	Communication Platform
microsoft-sharepoint	Integration Platform
shopify-your-ecommerce-store	E-commerce
wordpress-website-builder	Website Development
wix-owner-website-builder	Website Development
microsoft-teams	Communication Platform
square-point-of-sale-pos	Payment Platform
mailchimp-marketing-crm	CRM Platform
github	Integration Platform
zoho-crm-sales-marketing	CRM Platform
glassdoor-jobs-salary-talk	Integration Platform
hubspot-crm-grow-better	CRM Platform
alibaba-com-b2b-trade-app	E-commerce
woocommerce	Website Development
fiverr-freelance-services	E-commerce
redbubble-shop-original-art	E-commerce
bamboohr	Integration Platform
squarespace-run-your-business	Website Development
zendesk-support	CRM Platform
wise-ex-transferwise	Payment Platform
etsy-custom-creative-goods	E-commerce

Appendix B

Multilabel Classifier Pseudocode

Algorithm 1 Classifier Pseudocode

Import XGBClassifier from xgboost
Set *english_words* as a set of English words
Set *stop_words* as a set of English stopwords
Initialize a stemmer using PorterStemmer()
Function PREPROCESS_TEXT(*text*)
    Convert *text* to lowercase
    Split *text* into words and filter out words not in *english_words*
    Filter out words that are in *stop_words*
    Stem each word using the stemmer
    Join the stemmed words back into a single string
    Return the preprocessed *text*
End Function
Read the dataset from the CSV file located at '../dataset/seco_labeling_keyword.csv'
Split the data into training and testing sets: $X\_train, X\_test, y\_train, y\_test$ = train_test_split(data['review_description'], data['label_id'], test_size=0.2)
Create a pipeline with the following steps:
    Step 1: 'vect' - Initialize CountVectorizer with $ngram\_range = (1, 2)$
    Step 2: 'tfidf' - Initialize TfidfTransformer with $use\_idf = True$
    Step 3: 'clf' - Initialize XGBClassifier with $learning\_rate = 0.4$ and $max\_depth = 5$
Fit the pipeline on the training data: $pipeline.fit(X\_train, y\_train)$
Predict the labels for the test data: $y\_pred = pipeline.predict(X\_test)$
Print the classification report comparing the predicted labels (y_pred) with the true labels (y_test)

Appendix C

Feature Extraction Pseudocode

Algorithm 2 Pseudocode for Sentiment and Filtering

1: Set *seco_reviews* as a DataFrame read from the CSV file located at '../dataset/seco_reviews_labeled.csv'
2: Initialize a sentiment analyzer: $analyser = SentimentIntensityAnalyzer()$
3: Add a new column 'negative_prob' to *seco_reviews* using the 'review_description' column: STATE Apply a lambda function to calculate the negative polarity score using $analyser.polarity\_scores(body)["neg"]$
4: Create a subset *seco_reviews_neg* from *seco_reviews* containing rows where 'negative_prob' is greater than or equal to 0.4

Algorithm 3 Pseudocode for Feature Extraction using TF-IDF and Chi-squared

Import necessary libraries: *numpy*, *pandas*, *plotly*, and $sklearn.feature\_extraction.text$
Set *label_list* as a dictionary with 'label' and 'label_id' keys and their corresponding values
Convert *label_list* to a DataFrame named *label_id_df* and sort by the 'label' column
Merge *seco_reviews_neg* DataFrame with *label_id_df* using 'label_id' column and store the result in *seco_reviews_neg*
Create two dictionaries: *label_to_id* and *id_to_label* from *label_id_df* to map between labels and label ids
Create a $TfidfVectorizer$ object named $tfidf$ with sublinear term frequency, minimum document frequency of 3, L2 normalization, n-gram range of 2 to 5, and English stop words
Fit the $tfidf$ object to the 'review_description' column of *seco_reviews_neg* DataFrame and transform the data to an array named $features$
Assign the 'label_id' column of *seco_reviews_neg* DataFrame to *labels*
Set N as 10 or 100
Create an empty list named $label\_significant\_terms\_list$
for each row in $label\_id\_df$ **do**
    Get the label and label id from the current row
    Compute chi-squared scores for all features in $features$ that belong to the current label using $chi2$ function from $sklearn.feature\_selection$
    Sort the features by their chi-squared scores and get the top N significant terms for the current label
    Create a DataFrame named $label\_significant\_terms$ with 'term', 'score', and 'label' columns and the top N significant terms
    if $label\_significant\_terms$ is not empty **then**
        Append $label\_significant\_terms$ to $label\_significant\_terms\_list$
    end if
end for
Concatenate all DataFrames in $label\_significant\_terms\_list$ into a single DataFrame named $significant\_terms\_df$
Write $significant\_terms\_df$ to a CSV file named 'features_per_label2.csv' located in the '../analysis/' directory
Create a Plotly table object named *table* using $significant\_terms\_df$
Display the *table* object using Plotly's offline mode

Appendix D

Developer Response Pseudocode

Algorithm 4 Pseudocode for Calculating Developer Response Percentage by Category

1: Import the necessary library: *pandas*
2: Load the user reviews dataset from the CSV file located at '../dataset/seco_reviews_labeled.csv' and store it in a DataFrame named *reviews*
3: Create a dictionary named *label_dict* with label ids as keys and category names as values
4: Add a new column 'category' to *reviews* DataFrame by mapping the 'label_id' column with *label_dict*
5: Group the reviews in *reviews* DataFrame by 'category' and count the number of reviews with developer responses, storing the result in *dev_responses_by_category*
6: Group the reviews in *reviews* DataFrame by 'category' and count the total number of reviews per category, storing the result in *total_reviews_by_category*
7: Calculate the percentage of reviews with developer responses per category by dividing *dev_responses_by_category* by *total_reviews_by_category* and multiplying by 100, storing the result in *dev_response_percentage*
8: Calculate the overall developer response percentage in the entire dataset by summing up the values in *dev_responses_by_category*, dividing by the sum of values in *total_reviews_by_category*, and multiplying by 100. Store the result in *overall_dev_response_percentage*
9: Create a new DataFrame named *df_overall_dev_response_percentage* with a single row containing the overall developer response percentage value, using the index "Overall"
10: Concatenate *dev_response_percentage* DataFrame and *df_overall_dev_response_percentage* DataFrame along the rows, and store the result in *df_dev_response_percentage*
11: Print *df_dev_response_percentage*

Algorithm 5 Pseudocode for Sentiment Analysis and Calculation of Sentiment Percentages

1: Import the necessary libraries: *TextBlob* from *textblob*, *pandas*
2: Load the user reviews dataset from the CSV file located at '../dataset/seco_reviews_labeled.csv' and store it in a DataFrame named *df*
3: Remove rows with missing values in the 'developer_response' column from *df*
4: Define a function named *get_sentiment_type* that takes a text as input and calculates the sentiment using *TextBlob*. If the sentiment is greater than 0, return 'positive'. If the sentiment is less than 0, return 'negative'. Otherwise, return 'neutral'.
5: Apply the *get_sentiment_type* function to the 'review_description' column of *df* and store the result in a new column 'review_sentiment'
6: Apply the *get_sentiment_type* function to the 'developer_response' column of *df* and store the result in a new column 'response_sentiment'
7: Define a function named *calculate_sentiment_percentages* that takes a group as input. Within the function, count the number of each sentiment type in the group and calculate the percentage of each sentiment type. Convert the resulting Series to a dictionary and return the dictionary.
8: Group *df* by 'label_id' and apply the *calculate_sentiment_percentages* function to the 'review_sentiment' column, unstack the resulting Series, fill missing values with 0, and store the result in *review_sentiment_percentages*
9: Group *df* by 'label_id' and apply the *calculate_sentiment_percentages* function to the 'response_sentiment' column, unstack the resulting Series, fill missing values with 0, and store the result in *response_sentiment_percentages*
10: Print "Percentage of review_sentiment by category:" followed by *review_sentiment_percentages*
11: Print "Percentage of response_sentiment by category:" followed by *response_sentiment_percentages*

Appendix E

Review Trend Pseudocode

Algorithm 6 Pseudocode for Plotting the Number of Reviews per SECO Issue Type over Time

1: Import the required libraries: *pandas*, *matplotlib.pyplot* as *plt*
2: Load the reviews dataset from the CSV file located at '../dataset/seco_reviews_labeled.csv' and store it in a DataFrame named *reviews*
3: Create a dictionary named *label_dict* with label ids as keys and category names as values
4: Add a new column 'category' to *reviews* DataFrame by mapping the 'label_id' column with *label_dict*
5: Convert the 'review_date' column of *reviews* DataFrame to datetime format using the format "
6: Remove rows with missing values in the 'review_date' column from *reviews*
7: Filter *reviews* to keep only rows where the year of 'review_date' is less than 2022 and greater than or equal to 2013
8: Group the reviews in *reviews* DataFrame by 'review_date' and 'category', and count the number of reviews in each group. Reset the index and store the result in *reviews_by_label*
9: Pivot *reviews_by_label* DataFrame, using 'review_date' as the index, 'category' as the columns, and 'review_description' as the values. Store the result in *reviews_by_label_pivot*
10: Plot a line graph of the number of reviews per SECO issue type over time using *reviews_by_label_pivot*. Set the figure size to (12, 6)
11: Set the title of the plot as "Number of Reviews per SECO Issue Type over Time"
12: Set the x-axis label as "Date"
13: Set the y-axis label as "Number of Reviews"
14: Show the plot

Appendix F

Interview Questions

- **Category: Integration**

  1. How does your company ensure seamless integration with other platforms?
  2. What tools or methods do you use to identify issues related to integration?
  3. What are the common integration problems you've experienced in platforms?
  4. How does your company address the issues related to API errors and lack of integration?
  5. How do you handle cross-platform issues and ensure compatibility?

- **Category: Performance & Compatibility**

  1. How does your company ensure optimal performance across multiple devices?
  2. What steps do you take to ensure the compatibility of your software with mobile websites?
  3. How does your company resolve compatibility issues with different operating systems (iOS, Android)?
  4. Can you elaborate on the steps you take to resolve app crashes and slow performance issues?

- **Category: Privacy & Security**

    1. How does your company ensure the security of user data and prevent scams?
    2. What measures do you have in place to address fake apps and impossible login issues?
    3. How do you ensure the authenticity of reviews on your platform?
    4. Can you elaborate on the steps you take to address data mining and annoying notifications?

- **Category: Design & Complexity**

    1. How does your company ensure a user-friendly interface for your software?
    2. What measures do you take to improve your app and web interface?
    3. How do you handle issues related to confusing and unintuitive interface design?
    4. Can you share how your company addresses the difficulties in navigating your software?
    5. How do you improve your desktop interface and reduce lags?
    6. What steps do you take to avoid ads and make your interface less complex?

- **Category: Customer Support**

    1. Can you share your company's approach to customer support and how you ensure customer satisfaction?
    2. What measures do you have in place to address customer complaints and queries?
    3. How do you ensure that your customer support is easily reachable?
    4. Can you share how your company resolves seller support and chat support issues?
    5. How do you handle complaints related to rude customer service?
    6. What steps do you take to ensure that your customer support team can communicate effectively with customers?

- **Category: Cost & Pricing**

    1. How does your company ensure that your pricing is fair and transparent?
    2. What measures do you have in place to address issues related to loss of money and credit card problems?
    3. Can you share how your company handles refund and shipping cost issues?
    4. How do you address complaints related to expensive prices and fees?
    5. Can you share how your company ensures trust and builds trust with customers?
    6. How does your company ensure smooth payment processing and avoid bank account issues?