THE IMPACT OF USER INVOLVEMENT: ANALYSIS FROM APP REVIEWS

Erasmus University Rotterdam
Erasmus School of Economics
Economics and Business:
Marketing

Supervisor: Dr. Zhiying Jiang

Student: Miriam Erne
Student Number: 451235
Email address: miriam.erne@googlemail.com

Abstract

Application distribution platforms like Apple App Store or Google Play provide users of mobile applications with a space to provide feedback. Current literature assumes that user reviews provide important feedback for developers to improve their software, since they provide information about bugs, or ideas for new features. Several studies investigated these user reviews by mining and classifying the data in order to analyse user feedback in a fully automated way. Furthermore, for this research, several tools were developed that could eliminate "noisy" user reviews, locate comments on bugs, and group feature requests together. However, recent literature has never questioned whether user involvement, through reviews, is at all associated with app performance; this question is the focus of this research. The contribution of this study is twofold. First, a tool is presented that fully automates the classification of app store reviews, and then matches these reviews with version update logs on Amazon Mechanical Turk. Second, empirical evidence is presented on whether user involvement has an impact on app performance.

The findings imply that *Customer Led Innovation* has no impact on revenue, and that developers should rather listen to their intuition. Hence, developers must decide carefully whether to develop requested features or not. Nevertheless, when it comes to improving an application, it is much more economical for a company to rely on bugs reported by customers, rather than to test for bugs itself. However, these results might not be applicable to high-revenue apps that have already gained popularity.

Contents

List of Figures

List of Tables

List of Abbreviations

MT Amazon Mechanical Turk

App Mobile Application

1 Introduction

Nowadays, application distribution platforms such as Apple App Store and Google Play provide millions of different mobile applications (Apps) to users. In March 2017, there were around 2.8 millions Apps for Android users and 2.2 millions apps for App Store users available (Statista, 2017). App developers face great difficulty in surviving in the "hyper-competitive" mobile market (Comino et al., 2016). Among others, some recent literature states that the key to success for App developers is to listen to customers' voices in order to understand their needs and wants, and then to build Apps that customers love (Maalej et al., 2016; Maalej & Hadeer, 2015; Chen et al., 2014). Hence, customer reviews supposedly become essential for developers for several reasons. First, App developers might not always be able to spot non-working features or incompatibilities specific to certain combinations of smartphone and operating system amongst the plethora that exist. Moreover, such issues can even be complicated by users adopting different iterations of operating systems (e.g. iOS 10, iOS 10.3.2). Besides spotting bugs, user reviews also provide ideas for new features that can enhance user experience. App developers are constantly searching for new ideas to improve the App and to differentiate it from competitors. App reviews then become important sources for such valuable information, from which App developers do gain competitive advantages.

However, searching manually for useful feedback in user reviews is time and money consuming; Apps like Facebook receive more than 2000 reviews a day (Chen et al., 2014), which would be impossible for developers to manage manually. Therefore, several studies started began to mine customer reviews in a fully automated way, and categorised them into bugs, feature requests, user experiences, and ratings (Maalej et al., 2016; Maalej & Hadeer, 2015).

However, is it really beneficial to involve users in the app development process? Previous literature assumed without question that it is beneficial for Apps' performance, yet no research has ever investigated whether user involvement in the mobile application market has an effect on app performance at all. Further, there have been no studies on which specific forms of user involvement have what effects. As described earlier, users pay attention to broken features, which helps to improve the quality of an App. Meanwhile, users also help to bring innovation to an App with ideas for new features. Hence, there seems to be a difference between customer led improvement and customer led innovation.

What is missing is a study investigating whether Apps that take users feedback into account are more successful in terms of revenue, downloads and ranking. It is of great importance for App developers to have the knowledge and assurance that their

efforts to examine user feedback in reviews have an impact on their performance. This question will be answered with a comparison of user reviews and features released in updates. For this purpose, an analysing tool was built that classifies reviews into categories and at the same time search for matches with features from the updates. With an algorithm coded in Python, these matches are then assigned automatically to the independent variables *Customer Led Innovation* or *Customer Led Improvement*, indicating whether they improve or innovate an App.
Over 40,000 reviews were analysed, categorised, and matched in order to answer the question of whether user involvement in the App development process has a positive effect on App performance. The question of if user involvement is a key to success or whether App developers should follow their own intuition can now be answered more precisely. The result of this study reveals that there is a significant difference between addressed user innovation and improvement. While implemented user improvement can have a positive impact on revenue, especially if it is addressed quickly, user innovation does not. This means, if App developers transform user requests into new features, this does not impact on their revenue performance. Furthermore, this study reveals that the biggest impact on revenue is innovation initiated by developers, and the effect is regardless of how fast this innovation was published. However, when it comes to the improvement of an application from the developer's side, it is much more economical to rely on bugs reported by customers and fix them as soon as possible.

The subsequent chapters are structured as follows: Chapter 2 provides a review of the relevant literature. Chapter 3 introduces the conceptual model and hypothesis of the research. Chapter 4 explains the framework for the review categorisation process. Chapter 5 and 6 describe the data set and the methodological approach respectively. Chapter 7 discusses the results. Finally, Chapter 8 presents limitations of the work and ideas for further research.

2 Literature Review

"User involvement" is a well known concept in the literature and generally refers to the personal relevance and importance users can attach to information systems (Barki & Hartwick, 1989). Consumers have become more and more involved in the product development process behind the products they use and are creators too (Prahalad & Ramaswamy, 2013). User involvement is essential and indispensable for every company simply because it helps firms to gauge user requirements more accurately, and with that they are able to improve the quality of their products and

meet the needs and wants of their customers more precisely, which leads to greater user satisfaction (Kujala, 2008).

There are many ways for users to be part of the development of new products or improvement of existing products. In extant literature, abundant discussion has been devoted to the positive impact of user involvement on product development. Terms such as co-creation and co-design are used to describe the cooperation between developers and users whereby customers' needs and desires are translated into better products (Berger et al., 2005). Terms such as Quality Function Deployment (QFD), User-oriented product development, Concept testing, Beta testing, Consumer idealised design, Lead user method, and Participatory ergonomics, are frame-worked along two dimensions by Kaulio: the different phases of a product development project and the involvement of customers in the process (Kaulio, 1998). Similar to Kaulio, Kujala presents different user roles (providers of information, commentators, or objects for observations) – all of them result in more accurate user requirements, which leads to enhanced system quality, better-matched user needs, and thus, more satisfied customers (Kujala, 2008).

Though focusing on dimensions, these concepts support the same argument: that the involvement of users has a positive impact on product development (Kaulio, 1998).

In broad terms, user involvement is defined as "direct contact with users" (Kujala, 2003). App stores create a place where users and developers can come together. On such platforms, users can share their needs and experiences regarding an App, such as a missing features or functionality (Khalid et al., 2015; Panichella et al., 2015). Therefore, in this research, it is assumed that user reviews are a form of user involvement.

Barlow and Moller argue, for example, that complaining feedbacks are of true value for any developer. Complaints consist of hugely informative product feedback and help companies to focus on customers' needs and wants (Barlow & Møller, 1996). Many studies have shown that user feedback is inevitable when it comes to software quality. This was the motivation for many studies to mine user feedback automatically, since the amount of reviews became unmanageable, as well as because not all reviews are valuable (Carreño & Winbladh, 2013; Vu et al., 2015; Guzman et al., 2017; Chen et al., 2014; Pagano & Brügge, 2013; Pagano & Maalej, 2013; Panichella et al., 2015).

Early research on the subject started by studying the vocabulary used in App reviews and found out that the most frequently used words tended to express feelings (Hoon et al., 2012). This could be an indication that most reviews are about the user's overall attitude towards the App. With recent developments in computer and

data science, research is able to better utilise the immense amount of data available for review. Vu et al. (2015), for example, built the program MARK (Mining and Analyzing Reviews by Keywords), which allows you to search review content with predefined keywords. This might be helpful for developers to search for keywords often associated with bugs, like “fix”, “crash”, “freeze”. However, it becomes challenging when searching user feature requests (Vu et al., 2015). To solve this issue, Iacob and Harrsion designed MARA (Mobile App Review Analyzer), with which feature requests can be automatically detected (Iacob & Harrison, 2013). Carreño and Winbladh used automatic topic extraction to process App store reviews and found that automatically extracted information matched the manually extracted results (Carreño & Winbladh, 2013). This was one of the first proofs that the results of automatically extracted mining of reviews do not suffer in quality. The study by (Chandy & Gu, 2012) used a latent model to classify “bogus” reviews, which are reviews that prompt users to download spam applications. Another study introduced ALERTme – an approach to automatically classify Twitter tweets (Guzman et al., 2017).

The latest research on App store reviews has gone one step further, using natural language processing, topic modelling, clustering, and machine learning algorithms to extract features automatically from user reviews. Maalej and Nabil for example, test the performance of various classification techniques (String Matching, Bag of Words, natural language processing techniques (NLP)). They further investigated three popular algorithms for classification, namely Naive Bayes, Decision Tree, and multinomial logistic regression (MaxEnt). There was no one classification technique that fitted best; however, the algorithm Naive Bayes outperformed the others, because it is faster and a smaller training set is needed. The machine learning algorithms worked with vector properties, a training and testing phase to classify reviews into the four categories bugs, feature requests, user experience, and rating (Maalej & Hadeer, 2015).

Furthermore, in their work, (Chen et al., 2014) addressed two problems regarding App store reviews. First of all, not all feedback is valuable. Most of the reviews (64.9%) are noisy and irrelevant, and do not hold any useful information. Another problem is the amount of reviews some apps like Facebook receive; with more than 2,000 reviews a day, it is unmanageable to analyse them manually. They introduced a framework called AR-Miner (App Review Miner), which first eliminates all noisy reviews with text analysis and machine learning. With topic modelling, the remaining reviews are then grouped and ranked, and finally the most “informative” reviews are presented. The visualized ranking is of especially great value because it ranks the top-10 topics, for example, “support Chinese language” or “more themes”. This

way, developers are able to identify the important topics and decide the priority of Apps to develop. Similar to Chen et al., Guzman and Maalej present another framework with which features from reviews are extracted automatically by using natural language processing techniques and topic modelling. Moreover, they developed a coding tool for feature extraction, which served as an information base for the tool coded for this research (Guzman & Walid, 2014).
For further literature on automated feature extraction, (Martin et al., 2017) published a survey which summaries, among others, feature-related App Store analysis papers from 2010 to 2015.
Lastly, other studies have made the effort to build their own Apps to gather user feedback. (Seyff et al., 2014) built the AppEcho app, a feedback method that allows users to write individual feedback, because they believe that that feedback is vital for system improvement.

Current literature investigates the automation of user review classification, and several studies have analysed consumer reviews successfully. Reviews have been categorized into bugs, feature requests, user experiences, and ratings (Maalej et al., 2016; Maalej & Hadeer, 2015). Current research has the common view that with review extraction, developers can better understand user needs, thus enhancing system quality. However, the missing link is whether App developers take these reviews into account and update their apps accordingly, and whether doing so is beneficial. There is no evidence that heeding user opinions will result in better App performance. It might be true that user feedback information provides valuable information for App developers; however, the effect might be overrated in the App store environment. Ives and Olsen (1984), for example, present a "Descriptive Model of User Involvement", which points out that user involvement is only appropriate if certain involvement roles and development conditions are fulfilled. Hence, it is not only important to distinguish who should be involved, but also which type of system will be developed, and at which stage of the development process it is most beneficial. In the most extreme case, user involvement is not desired at all. Instead, technical expertise is needed (Ives & Olson, 1984).
This research intends to test this presumed positive effect of user feedback on App performance. All the aforementioned research has presumed that user feedback leads to improved systems. However, to our knowledge, such a presumption has yet to be tested. This paper fills the gap in the literature by examining if user involvement has an impact on App performance.

3 Conceptual Framework and Statistical Model

3.1 Conceptual Model

The conceptual model shown in Figure 1 presents the different relationships examined in this research.

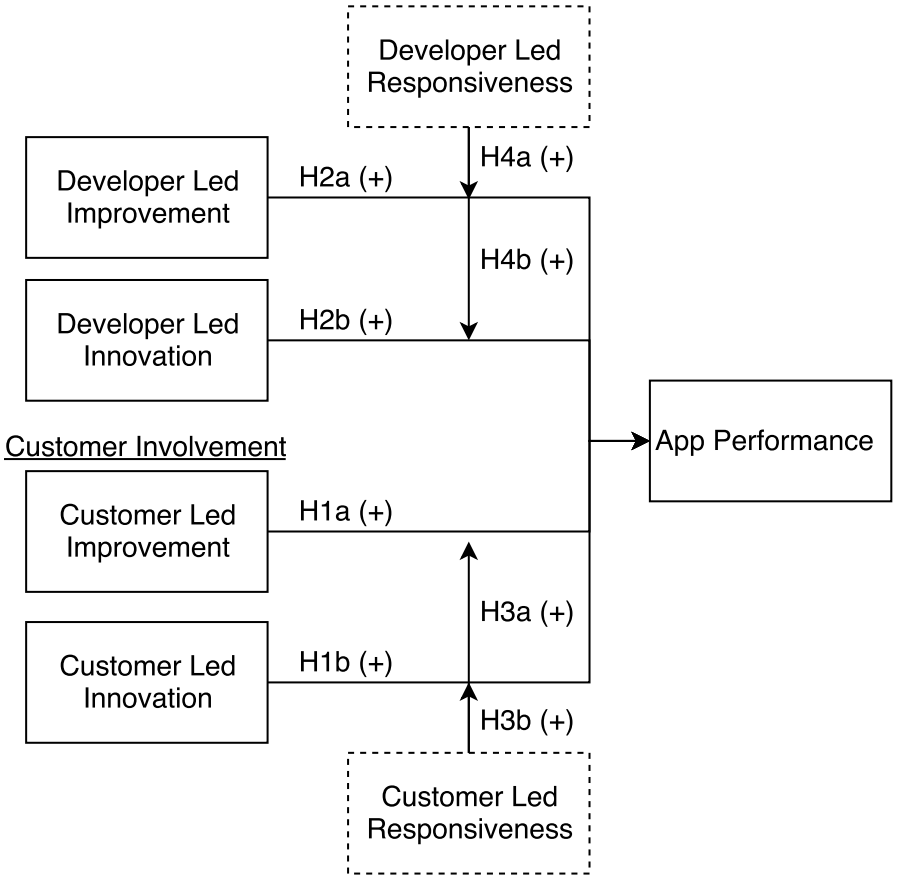


Figure 1: Conceptional Model

First, it is assumed that customer involvement in general has a positive effect on the App performance. With user involvement, developers do not only get insight into users needs and wants, they can build or alter features that users want, accept, and understand, which leads to more effective system use and increased user satisfaction (Kujala, 2008). In this research, customer involvement is further divided into Customer Led Improvement and Innovation. The purpose of the research is to study if user involvement has different effects on App performance.

Hypothesis 1a: Customer Led Improvement has a positive impact on App performance

Hypothesis (H1a) tests if improvement suggestions from customers have a positive

effect on App performance. Improvement suggestions are reviews that point to a specific problem (e.g. that their iPhone is slowed down by using an App). The suggestions are regarded as beneficial because App developers might not always be able to spot a non-working feature. Spotting bugs across the plethora of different types of smartphone, and different OS versions, can become quite challenging and time consuming. Moreover, such issues can even be exacerbated with different software updates (e.g. iOS 10, iOS 10.3.2). Fixing such non-working features increases the quality of the App and the satisfaction of its users, and hence, should result in better App performance. Furthermore, improvement suggestions can be related to usability problems. Non-tech- savvy users may have problems while using the App that developers never expected. This feedback can help developers to make the App more user-friendly.

Hypothesis 1b: Customer Led Innovation has a positive impact on App performance

As well as offering troubleshooting, some reviews express the wish for new functionality and features, which are regarded as Customer Led Innovation (H1b). They go beyond improving an App; these reviews can offer creative new solutions and suggestions for the App's development, and serve as an idea source for developers. Again, some developers might have missed these features because of their strong tech-based backgrounds.

Compared to customers' involvement, developers as App creators, must also have a positive effect on App performance.

Hypothesis 2a: Developer Led Improvement has a positive impact on App performance

Developer Led Improvement includes incremental system improvements, when developers fix non-working features. Naturally, it must have a positive effect on App performance because App quality is improved (H2a). However, the impact might be less pronounced, since finding and fixing bugs can cause problems and cost money.

Hypothesis 2b: Developer Led Innovation has a positive impact on App performance

Hypothesis H2b tests whether innovation initiated by developers themselves has a positive effect on App performance. What this means is that developers innovate the App, for example, by launching new features, without taking users opinions into

account. There is extensive research in the extant literature that suggests companies cannot build great products without involving customers and understanding their needs and wants. They might risk developing costly system features that the user did not want or cannot use (Kujala, 2008). However, others arguing that companies should lead their users and not the other way around. Successful innovations are shown to be built on hard work, extremely talented developers (for example, Steve Jobs), and radical new features. Consumers might not like this kind of innovation initially, as they dislike change, so they may not contribute to such innovations. Real competitive advantage must come from within a company; listening only to user ideas will lead to a market sameness of companies (Skibsted, Jens M. and Hansen, Rasmus B., 2014). The same logic may apply to Apps; users might only come up with ideas that make the App more convenient, whereas developers are the ones with proposals for radical innovation. Another argument supporting H2b is that developers come up with ideas that are producible. An empirical study in the mobile telephony market from Magnusson found out that users might have more creative and useful ideas for innovations than developers. However, these ideas were unfeasible and not producible (Magnusson, 2003).

Be the innovation or improvement initiated by customers or the developers, it still takes time to fix the issues or build the new features. Hence, whether the app enhancement is successful also hinges on the response time developers take to accomplish the task. The moderator variable, response time, first tests whether on average, it takes longer to implement customer led or developer led improvement and/or innovation. The effect is assumed to be that the longer the response time, the smaller the impact of app enhancement activities is on final app performance. However, one also needs to note that a fast response time results in more frequent updates and people do not like frequent updates. Updates interrupt user workflow and often cause problems with programs people rely on regularly (Vaniea & Rashidi, 2016; Schenck, Stephen, 2013; Armerding, Taylor, 2012). Hence, fast response times and the increased quality of the App will be enhanced with updates, but users may not value them.
Furthermore, there may be a difference between response times Customer and Developer led enhancements. Quick response times on the Customer side might have a stronger effect, as customers wait for a response to their request. Lastly, there may be a difference between quickly implemented improvements and innovations. An implemented improvement is assumed to be of great importance for exiting users, mostly because a feature is not working properly, which limits customers' usage of the App. Hence, developers should fix bugs as soon as possible, otherwise users

might stop using the application. On the other side, quickly implemented innovations might be important for attracting future users, who had not started using the App because of a missing functionality and feature.

H3a: With a shorter responses time to address a Customer Led Innovation, the impact of the Customer Led Innovation on app performance is larger.

H3b: With a shorter response time to address a Customer Led Improvement, the impact of the Customer Led Improvement on app performance is larger.

H4a: With a shorter response time to address a Developer Led Innovation, the impact of the Developer Led Innovation on app performance is larger.

H4b: With a shorter response time to address a Developer Led Improvement, the impact of the Developer Led Improvement on app performance is larger.

3.2 Statistical Model

The method chosen to test the above hypothesis is a Multiple Regression Analysis. The aim is to analyse the relationships and to explain the variation in the dependent variables (*revenue, downloads, ranking, and rating*) as much as possible on the basis of the variation in the independent variables (*Customer Led Improvement, Customer Led Innovation, Developer Led Improvement, Developer Led Innovation*). Furthermore, it is of interest whether there might be factors moderating the strength of such relationships. Therefore, a moderating variable (*Response Time*) is introduced. The advantages of linear regression are that it can indicate the significance and the magnitude of the impact from independent variables to the dependent variable. Assuming there are no structural changes, it is also possible to make predictions with the model (Mooi & Sarstedt, 2011).

Based on the seven hypotheses discussed earlier above, the following models are tested.

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_k x_k + \epsilon$$

Linear models (1) and (2) are tested for H1a, H1a, H2a, and H2b.

$$\begin{aligned} Revenue = \beta_0 + \beta_1 CustomerLedImprovement + \\ \beta_2 CustomerLedInnovation + \epsilon \end{aligned} \tag{1}$$

$$\begin{aligned} Revenue = \beta_0 + \beta_1 DeveloperLedImprovement + \\ \beta_2 DeveloperLedInnovation + \epsilon \end{aligned} \tag{2}$$

Model (3) and (4) test H3a and H3b.

$$\begin{aligned} Revenue = \beta_0 + \beta_1 CustomerLedImprovement + \\ \beta_2 CustomerLedImprovement \times \\ ResponsivenessCustomerLedImprovement + \epsilon \end{aligned} \tag{3}$$

$$\begin{aligned} Revenue = \beta_0 + \beta_1 CustomerLedInnovation + \\ \beta_2 CustomerLedInnovation \times \\ ResponsivenessCustomerLedInnovation + \epsilon \end{aligned} \tag{4}$$

Model (5) and (6) test H4a and H4b.

$$\begin{aligned} Revenue = \beta_0 + \beta_1 DeveloperLedImprovement + \\ \beta_2 DeveloperLedImprovement \times \\ ResponsivenessDeveloperLedImprovement + \epsilon \end{aligned} \tag{5}$$

$$\begin{aligned} Revenue = \beta_0 + \beta_1 DeveloperLedInnovation + \\ \beta_2 DeveloperLedInnovation \times \\ ResponsivenessDeveloperLedInnovation + \epsilon \end{aligned} \tag{6}$$

Lastly, model (7) checks the overall model in order to see which types of innovation or improvement have a larger effect on revenue.

$$\begin{aligned} Revenue = \beta_0 + \beta_1 CustomerLedImprovement + \\ \beta_2 CustomerLedInnovation + \\ \beta_3 DeveloperLedImprovement + \\ \beta_2 DeveloperLedInnovation + \epsilon \end{aligned} \tag{7}$$

4 Natural Language Processing: Review Analysis

The goal of this paper was to find a fully automated way to analyse reviews, so that app developers do not have to screen reviews manually for important user input. Sentiment analysis or opinion mining could be used, which is the automated computation of opinion, sentiment, and subjectivity in text, as stated by Pang and Lee (Pang et al., 2008). However, the topic is complex, with more than 7,000 written articles about sentiment analysis (Feldman, 2013), and the techniques nowadays are

far more advanced than simply looking at reviews as a polarity only (either positive or negative) (Pang et al., 2008). There are free sentiment analysis services available, for example, the Google Cloud Natural Language API (Google Cloud Platform, 2017) or the Text Analytics API from Microsoft (Microsoft Azure, 2017). Unfortunately, these services were not useful for the purpose of this study because they cannot tell if a match exists between a review and a feature update. They only note the polarity of the inserted review, as described by Pang and Lee, not the opinion of the user. Therefore, the question remains of to how to automatically analyse reviews and match them with version updates in a short period of time. There is a marketplace for human intelligence called Amazon Mechanical Turk (MT) from Amazon. Amazon Mechanical Turk provides businesses with "access to a diverse, on-demand, scalable workforce" (Turk, 2017a). This means Amazon provide a wide range of workers who work 24/7 on uploaded tasks from all over the world, where jobs can be completed within minutes (Turk, 2017b). When it comes to online outsourcing, quality control becomes an issue, as is discussed in more details the appendix A1. However, the benefits outweighed the drawbacks and MT was regarded as useful tool for this study.

Figure 2 shows the overview of the framework for App review and App update analysis:
(i) first step: separating *specific* user reviews by filtering *generic* reviews (noisy and irrelevant ones) (chapter 4.1),
(ii) second step: searching for matches between the reviews and feature updates, and simultaneously categorising the reviews into (1) bugs, (2) feature requests, (3) user experiences, (4) ratings (5) pricing, (6) too many advertisements, and then writing short descriptions about them (chapter 4.2).

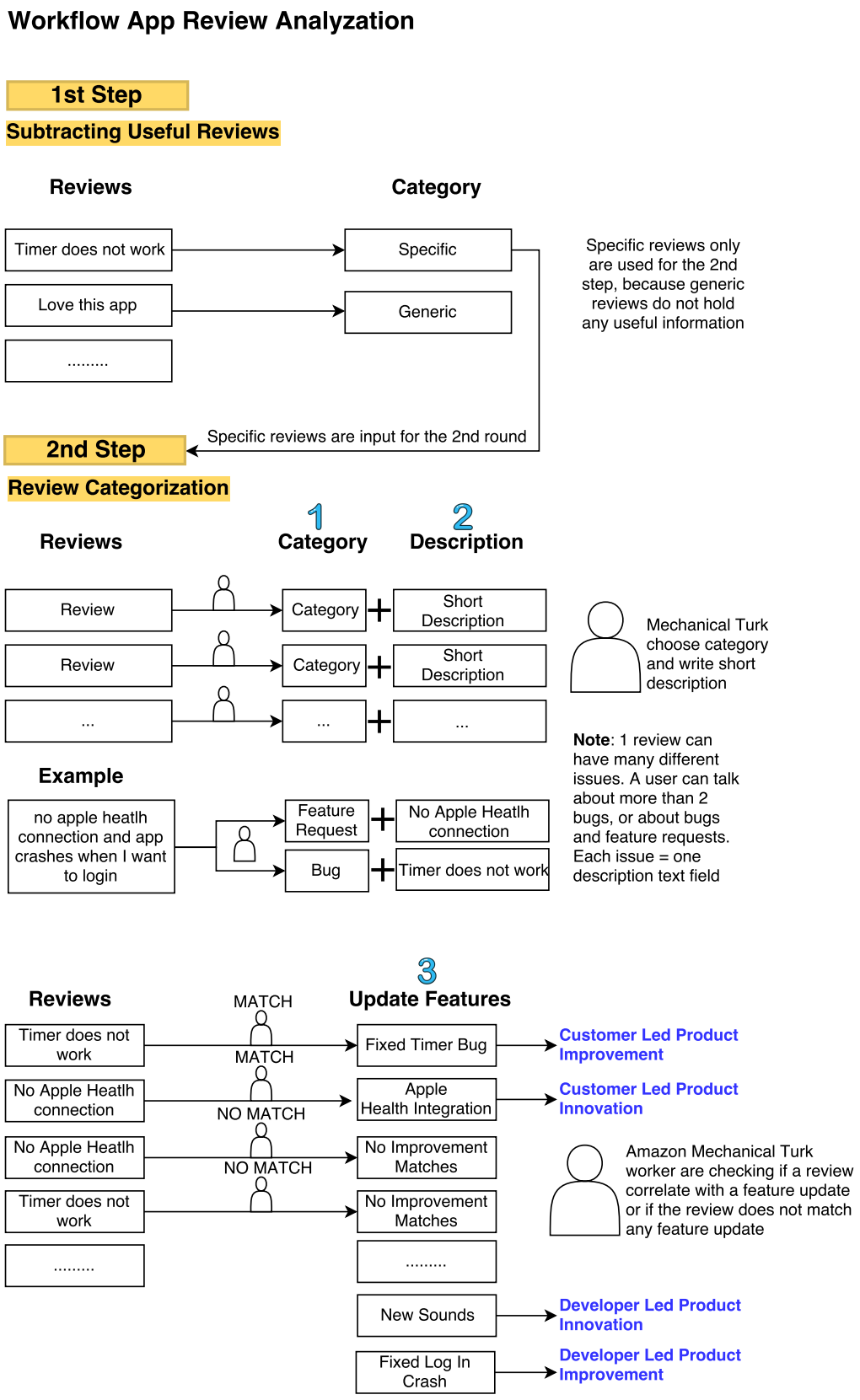


Figure 2: Framework for App review and App update analysis (Source: own)

4.1 Specific Versus Generic Reviews

Similar to Chen et al. (Chen et al., 2014), the first step shown in the above framework was to eliminate reviews that do not hold any useful information. Therefore, reviews were categorised in two broad categories, namely *generic* and *specific* user reviews. Only reviews that had a specific actionable function were of interest. A specific actionable function is something an app developer can fix, improve, or create. For example, a review with a purpose to fix an existing feature, or a review with a new feature request, or a review with a complain that a feature could be enhanced.

In this paper, *generic* reviews are noisy and irrelevant rating reviews or user experience reviews, as shown in Figure 3 and Figure 4.

★★★★★5.0	**Calm review** by **Gio Astro** I love this app so much it is really fun for me and my family 😊😊😘😍	Feb 28, 2017
★★★★★5.0	**Awesome app** by **byahyaa** Love it	Feb 28, 2017
★★★★★5.0	**Amazing!!** by **Amanda_v** I absolutely loooove this app! I can workout anywhere.	Feb 21, 2017
★★★★★5.0	**In a word awesome** by **Sarah-eve-angel** By far the best app on meditation!!	Feb 28, 2017
★★★★★5.0	**Mindful mom** by **Jewelzb74** Loving this app!	Feb 28, 2017

Figure 3: Source: App Annie (2017a,b,d)

★★★★★5.0	**EXCELLENT Meditation & Relaxation Tool!** by **Zylice** This app makes both meditation and relaxation accessible to ANYONE as they can do these exerci ses anytime anywhere. You can chose the length of a wide variety of mindfulness mediations for va rious things such as: cooking, anxiety, depression, commuting and even sleeping. Andy explains ho w to apply simple elements of mindfulness into everyday life and he does it so well that you gradua lly ease into the habit. All in all, if you want time no matter how small to dedicate yourself to some well earned relaxation in your day, download this app and never look back! 😊😌👍🙏	Feb 17, 2017
★★★★★5.0	**Amazing Meditation App** by **VickiHam911** I love this app and have done since the moment I started using it. The layout and design of it is real ly enjoyable, and Andy has a really nice voice and character! The fact that it keeps track of how ma ny days in a row you have meditated 100% makes me try as hard as possible to do it everyday. Als o affordable. Potentially helped me get through a period of anxiety/depression I was going through when I started using it.	Feb 17, 2017
★★★★★5.0	**Calm** by **Grlonfire2323023** I have suffered from insomnia for years. I take medication to help me sleep. I've had several back s urgeries and other issues and a lot of pain and it keeps me awake the first time I used Calm in t Th e sleep mode I was asleep within 15 minutes. I was amazed! So I did it again last night and I was so und asleep and didn't wake until four in the morning. Amazing. I'm going to try it on my granddaug hters who have a difficult time sleeping. I'm so very happy I found your app sincerely Judy Armstro ng	Feb 27, 2017

Figure 4: Source: App Annie (2017a,b)

User experience reviews are often quite helpful for other users; they express users

overall attitude toward the App experience. However, for this study, they were not suitable because the information is not helpful when it comes to App developments. It is rather a personal experience, including motivation for use, and their need to tell someone about it (Chen et al., 2014)[1].

4.2 Review Categorization

The second step from the above framework was the final review categorisation with the remaining *specific* reviews from the first round. This categorisation was divided into three parts:

1. Review categorisation
2. Enumeration of issues mentioned in a review
3. Check matches between issues from reviews and features from the version update

First of all, a review was categorised into a *bug, feature request, pricing, user experience, rating*, or *too many advertisements*. Note, Maalej and his colleagues classified reviews only into four categories, namely bug reports, feature requests, user experiences, and ratings (Maalej & Hadeer, 2015; Maalej et al., 2016). However, after screening hundreds of reviews manually and making notes about their categories, (5) pricing and (6) too many advertisements were added, because a lot of them had occurred in the initial screening process.

The goal of the first classification was to eliminate useless reviews, which were *user experience* and *rating* reviews; however, the results showed that MT workers were not always able to classify the reviews correctly, which means that a lot of *user experience* and *rating* reviews were not sorted out and were still in the dataset. For that reason, the two categories were added to the second classification round again, to detect and finally eliminate them. The option *other* was added as a category if an MT worker was unsure about the category. Below are the detailed descriptions of different categories.

[1]These reviews can tell app developers a lot about user motivation, and why the app is used. This knowledge can result in new and better ideas for App improvement. However, it was not focus of this study.

Review categories

In a recent study, Maalej and Nabil used keywords to indicate the category belonging (Maalej & Hadeer, 2015). Their keywords were used for the four categories *bug, feature request, rating*, and *user experience*. Keywords for the categories *pricing* and *too many advertisements* were added for this work, since they were not covered by Maalej and Nabil.

Bug

Keywords: bug, fix, problem, issue, defect, crash, solve
A bug report is in general an unwanted error in a program or system; they arise mainly because of programming failures by developers (The Jargon File, n.d.). A bug is any kind of problem with the app: a crash, an erroneous behaviour, or a performance issue (Maalej & Hadeer, 2015). Therefore, a bug is literally anything a user is complaining about that is not working right, but it is not a bug if it is a case of a user is wishing for a new feature.
Example bug reviews:

"It's not letting me sign up, and I deleted the app and re-downloaded it but it's not working"

"It isn't letting me make an account and says error try again later"

"If you open the app in the watch it tries to connect for a minute (literally a minute) then crashes"

Feature Request

Keywords: add, please, could, would, hope, improve, miss, need, prefer, request, should, suggest, want, wish
In general, a request for a new feature is when the user thinks something should have been developed that does not exist yet. Therefore, it requires new code (Cheung, 2013; Wiggins, 2015). A feature request is the wish for new and missing functionality, such as if users raise new ideas to make an application better in the future, or if they compare the App with missing features that similar Apps offer (Maalej & Hadeer, 2015). If somebody is asking for new functionality in the application, this is treated as a feature request too.
Example feature request reviews:

"Needs to have a value for calories burned for strength training too"

"Missing Apple Watch compatibility"

"The exercises are great workouts. I only wish they were a little longer, maybe 30mins and there were options for exercises that focus on core areas."

User Experience

Keywords: help, support, assist, when, situation

The ISO 9241-210 states, cited in (Hedegaard & Simonsen, 2013), as well as Law et al. (Law et al., 2009), that user experiences derives from the individual, not the social, and is this individual's response and perception, which emerges from their use of the product or service. In this research, user experience is described as the reflection of users experience with the App and App features (Maalej & Hadeer, 2015). User experience reviews are mostly positive with high star ratings, wherein users talk about why they like the app and how it changed their life.

Example user experience reviews:

"This app is fantastic in every way. If used correctly, it can be life changing. It integrates with MapMyWalk, transferring calorie expenditure to your daily calorie requirements. The community is warm and supportive and you'll have all the tools you need to lose weight and get healthy. Good luck!"

"The WW app keeps me focused on the choices of food I eat and how each selection effects my well-being. It's a great app!"

"I love this app! Tracking is so easy. I can find foods with the search bar quickly. The Connect feature gives support from members all over the world."

Rating

Keywords: Great, good, nice, very, cool, love, hate, bad, worst

Ratings are often text reflections of the numeric star rating (Maalej & Hadeer, 2015). They are useless for the app development process because they do not hold useful information; they are very generic.

Example rating reviews:

"Love it!"

"Great"

"It's a pretty good app, I like it."

Pricing

Keywords: expensive, price, pricing, rip off, $, cost, overpriced, fee, pay, payment, paid, cheaper

This indicates that a user is talking about the price of the app, e.g. that it is too expensive, unfair, a rip off etc.
Example pricing reviews:

"$10 to track macronutrients...used to be free. Will be using the $3 app that works just as well for a fraction of the cost!"

"Overpriced...over rated...only tracks calories...$10 a month to track macros...ridiculous"

"Your initial fee only gives you a handful of exercises. To get additional workouts it costs more $$."

Too Many Advertisements
Keywords: ads, advertisements, popping up, annoying, banner
Indicates that a user is complaining about too many advertisements, e.g. when ads are popping up all the time.
Example reviews for too many ads:

"The avalanche of ads makes it unusable unless you pay $3 each and every month."

"Paid for the ap. Still get ads pushed to me. Don't advertise to me if I paid the money for the non ad version."

"After paying around $12 for the app and add-ons, that stupid banner at the top is the most annoying thing! Isn't that why we pay for apps, to eliminate the need for ads?"

Since, one review can belong to more than one category, called mulitclass classification (Maalej & Hadeer, 2015), it was possible to select as many categories as needed. A user may have written a review that stated a login crash since the latest update and at the same time a wish for a new feature or a pricing topic. Therefore, a review was separated into different so called issues, where one issue presents one topic a user is referring to.
Example review (App Annie, 2017f):
"It's a great app. Just a couple of thing I wished it had like body measurements for non scale victories, and have a daily water intake goal. The daily points do not match what is in my booklet from my weekly meetings. It would also be great if the Fitbit fitness points were more consistent because I did more steps one day and it was less fit points than the day before when I did less steps. Other than these things the app is a great replacement of the paper tracker."
In this review, four different issues are described, namely:

1. Feature request: body measurements for non scale victories
2. Feature request: daily water intake goal
3. Bug: daily points do not match what is in my booklet from my weekly meetings
4. Bug: Fitbit fitness points need to be more consistent

Categorising these was the task of the MT worker; he chose a category and wrote each issue in a free text field. The last step for the MT worker was to check if an issue had a match with a feature from a version update. For that, a dropdown list was created, where all features from the update, plus the option "I checked, no matches here", were listed. Figure 5 shows the layout for the review classification process.

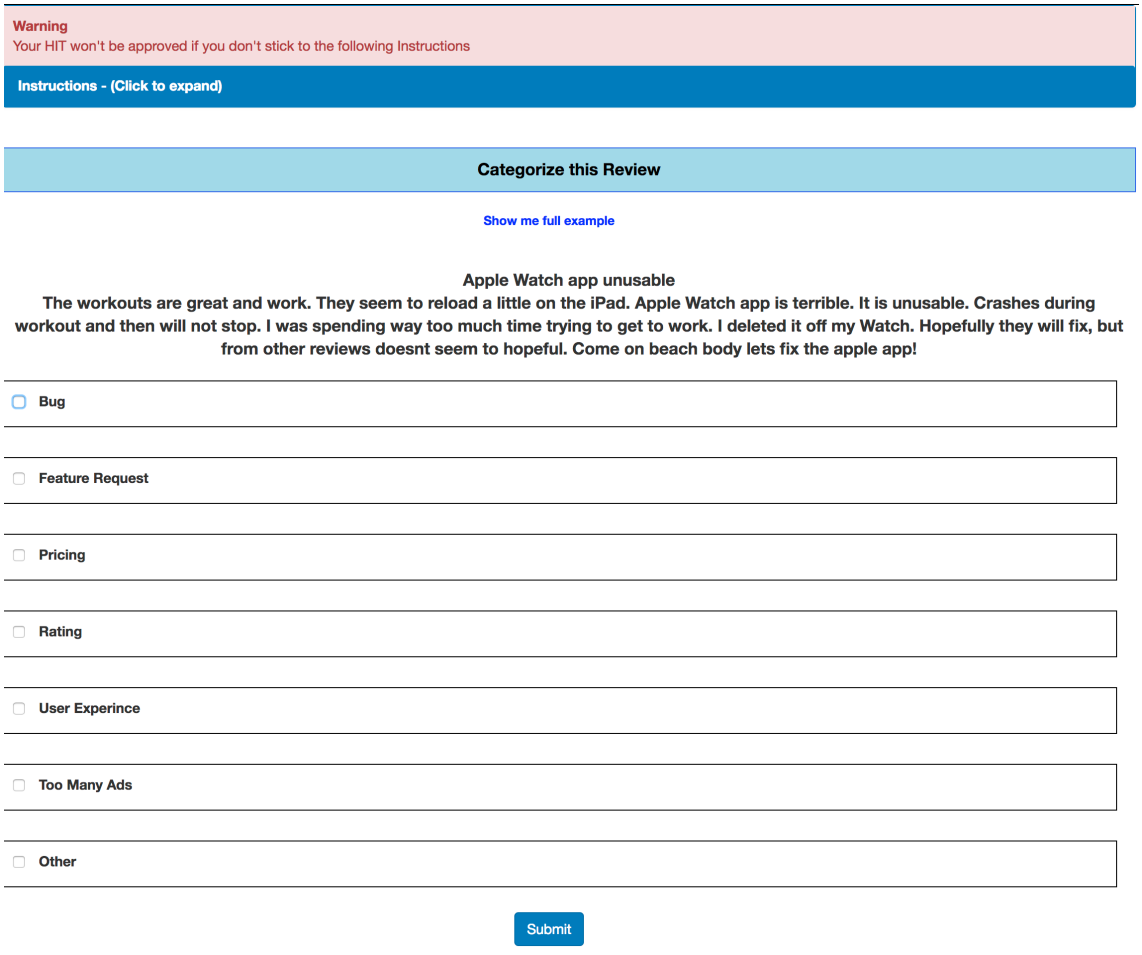


Figure 5: Layout Classification Tool

Figure 6 shows the layout where a category, here *bug*, is selected. The red "minus" icon is to remove an issue, and the green "plus" icon is to add another issue. It is

possible to add and remove as many issues as needed. Furthermore, two blue links are visible. If one clicks on "What is a Bug?", an overlay window pops up with the explanation of the category *bug*. "Show me example" shows a review where a *bug* is described along with how to fill out the free text plus dropdown list, if available. For each category, both links were created.

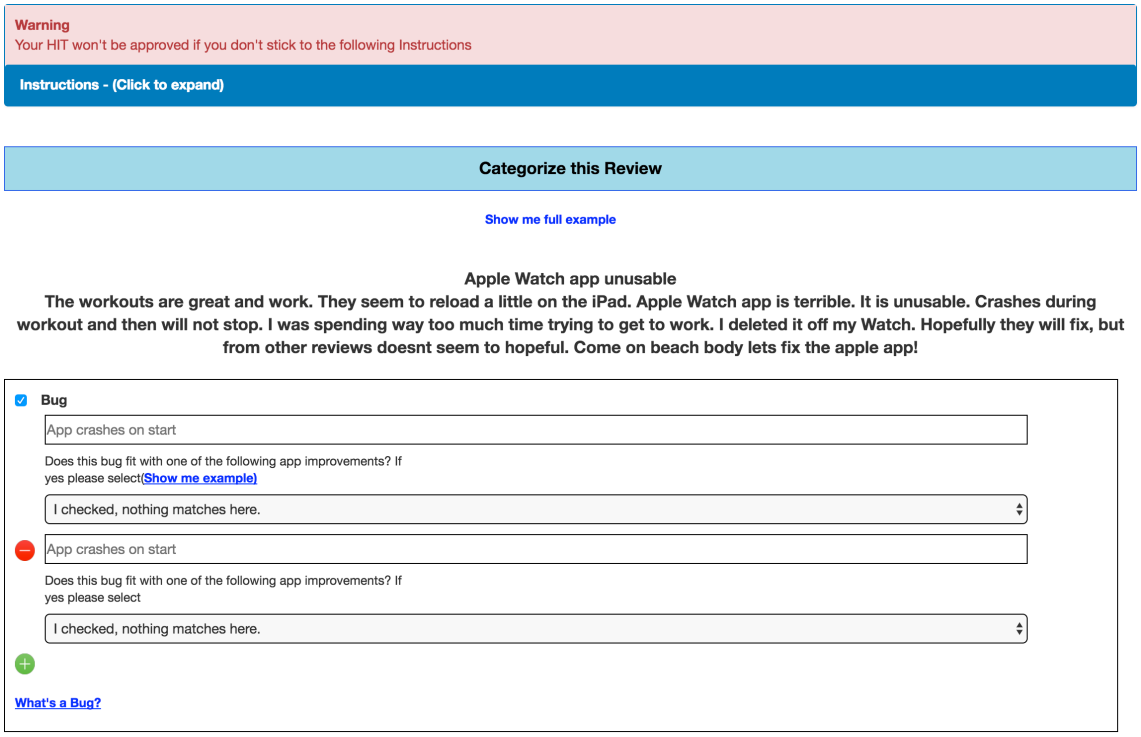


Figure 6: Layout Classification Tool, free text field

Figure 7 shows what the dropdown list looks like. Note, the dropdown list was developed in a way that shows exactly the amount of features an update had, e.g. the App "Sweat with Kayla" consists of eight feature updates, hence the dropdown lists shows these eight features plus the default for no matches. The app "Weight Watchers" had only two update features, hence only three entries in the dropdown list.

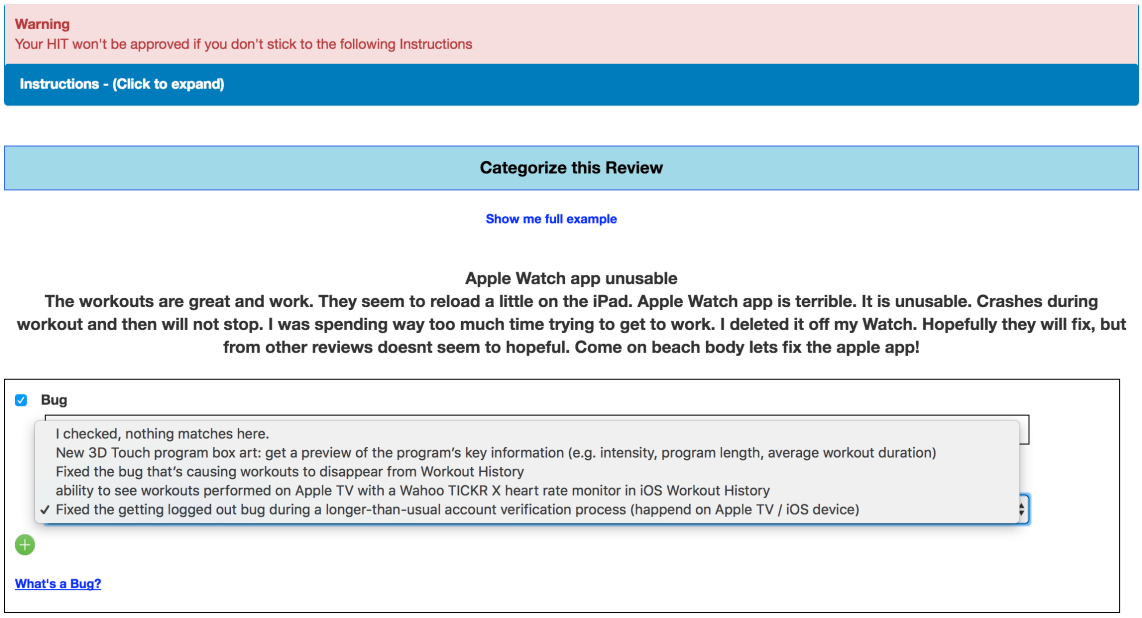


Figure 7: Layout Classification Tool, dropdown items

The form was coded in HTML, created by a developer from the German platform www.machdudas.de. For more details on how the forms were created for each App, please refer to Appendix A12).

Only the variables *Title* and *Review* were displayed, as well as the features from the version updates in the dropdown list. All other variables were coded as hidden fields, since they were not relevant for the MT worker, but essential for the analysis.

The final classification was done in several rounds. The first three rounds were only with a few reviews (around 50 - 100) to see if the HTML form was working correctly, as well as, most importantly, if the excel output file columns were named correctly, and if the columns were connected to the HTML form[2].

Furthermore, these testing rounds were important opportunities to receive feedback from the MT workers. Some wrote questions that were added into the briefing for better understanding. After fixing a couple of bugs, the HTML form was working correctly and the final classification was uploaded. The sample was divided in two sets, which were uploaded after one and another to have the opportunity to check the results in time. Approximately, 1000 reviews were rejected because of incorrect classification and automatically uploaded again for new analysis. A requester on MT can see the time in seconds a worker spent on one review. We use the time

[2]Several other checks had to be carried out, to make sure that the HTML form generated a correct output file (e.g. does every row reflects one review, to make sure: that all blue links with popping up windows were working fine; that the dropdown list was dynamic – this means that for every app the corresponding feature update items were displayed; that the plus and minus icons were working to add and remove lines, to check if the submit button worked, ...)

spent by the MT worker as a filter for quality control. All HIT's analysed in less than 20 seconds were quarantined first. During this control, some workers had been noticed who repeatedly classified reviews wrong and all the reviews submitted by these workers were screened afterwards. In this way, workers who tried to make fast money, while rushing through HIT's, were detected and blocked for future HIT's.

5 Variable Construction

5.1 Independent Variables

Customer Led Product Innovation I.e. if customers are part of the innovation process of the application, e.g. a user expresses the need for a new feature in a review and app developers implement that feature in a future update. This is a match between a review and a feature from the update. Example review from the data set: *"The app works great, but has become very out of date with all the new blends available. For the price of the app, I would expect regular updates as new oils and blends become available."*

In their next update, the corresponding app launched, amongst others things, the feature *Added new Oils and Blends*, hence there is a match.

The match between a user review and feature update counts only if the review date is prior to the update date, otherwise the feature is not innovated by customers. The level of measurement is interval scaled, where one unit represents one match between a review and feature update; if three different reviews all match with the same feature update, the variable has a value of three.

Customer Led Product Improvement The variable *Customer Led Product Improvement* is similar to *Customer Led Product Innovation*. When the user has helped to improve the App, it is likely to be a report about unwanted errors in the App (*bugs*), annoying advertisements, or usability problems. The level of measurement is interval scaled, and one unit represents a match between a review and update. The match counts only if the review date was prior the update date, otherwise the improvement was from App developer's side.

Developer Led Product Innovation The variable *Developer Led Product Innovation* assumes that the innovation always comes from the developers themselves when no user review match was found with a feature update. The level of measurement is interval scaled, and one unit represents a review with no feature update match.

Developer Led Product Improvement The variable *Developer Led Product Innovation* assumes that the improvement always comes from the developers themselves when no user review match was found. The level of measurement is interval scaled, and one unit represents a review with no feature update match.

5.2 Dependent Variables

Revenue One performance measure is the revenue that an application made during the time frame. The revenue does not only count app purchases, but also microtransactions within an app, as in, for example, in-app advertisement (IADV), which became very popular in recent years, and in- app purchases (IAP) (Ghose & Han, 2014). Revenue was calculated by downloading the daily revenue for the sampling period per app, and then summing them up. The variable is expressed in dollar values, and hence is interval scaled, where one unit represents one dollar.

Downloads App Annie counts unique downloads on a daily basis: unique because if the same user downloads an application on multiple devices it counts only as one download. Downloads for App updates are also not counted (Annie, 2017). Note: some of the apps have a paid and free version. In the dataset, both versions of an application were added separately, so that one can see the difference in free or premium versions. The variable is expressed in number of downloads, and hence is interval scaled, where one unit represents one unique download.

Rating Users can provide ratings on the Apple App Store on a scale from 1 to 5 stars. Ratings and reviews are consumers' attitudes towards an application. Each App on the App Store has a summary rating, which App developers can reset with a new version if they wish (Apple Developer, 2017b). For this research, rating is calculated as the mean of all analysed reviews, which includes also the *generic* reviews that were eliminated in the first MT round from the *specific* reviews. Rating can have any value between 1 and 5. To receive the most accurate value as possible, and especially to rank the apps based on their rating, decimal places were included.

Ranking There are 2 different measures for ranking: ranking by downloads, and ranking by revenue (grossing). The algorithm for how Apple calculates their search rankings is unknown; however, some ASO (app store optimization) specialists suspect several factors are of importance. The average App Store rating,

the rating / review volume, downloads and, uninstalls (retention and churn), App usage statistics, growth trends weighted toward recent, and the keyword density of the App's landing page (Walz, 2015) are all contributing factors. Therefore, ranking can be deemed as a composite index that accounts for various aspects of an App's performance. Moreover, only the ranking data for the Health and Fitness category was chosen, not the overall App Store ranking.

5.3 Moderating Variables

The response time variable was added to see whether an App's responsiveness has an effect on the impact of customer and developer's efforts regarding app performance, and especially if there is a difference in customer led and developer led response time. The level of measurement for all four variables is interval scaled, and one unit represents one day.

Responsiveness Developer Led Product Improvement refers to timespan from the old update date to a new update date in days. Based on the update descriptions, every app does improve / fix something in each update; it is unlikely that app developers only launch new features in an update. Therefore, it is assumed that the response to fix bugs is the difference in days of two consecutive updates.

Responsiveness Developer Led Product Innovation refers to the time needed to launch new features initiated by the developers. This is the time span from the current new feature to the next new feature update. Note: this variable is the time span per feature; this means, if in the future update four new features have been launched, the time span is divided by four. However, if the analysed application did not have an update with new features in the future, the last update from the past was chosen, where features have been released.

Responsiveness Customer Led Product Improvement refers to interval from the first user review until the bug was fixed / advertisement was removed in days. It is the time span from the update date minus the first mentioned review date.

Responsiveness Customer Led Product Innovation refers to time interval from the first review until the new feature was launched. It is the time span from the update date minus the first mentioned review date for new features.
The variable is the average of all features from the update where a match with

a user review was found. “Sweat with Kayla” released, for example, eight new features in the analysed update. For all user review matches with these features, the time span from the first review until the feature was launched has been computed. Next, the average of the computed time spans were calculated to receive the average customer led response time for one new feature.

5.4 Control Variables

As control variable, the experience of App publishers was chosen. It is assumed that the amount of apps published by a publisher has a positive impact on the App's performance (Lee & Raghu, 2014). The more applications a publisher has released, the more experience they have gained in the development and commercialisation of applications.

6 Data

6.1 Data Source

The data were obtained through App Annie, a business intelligence company (App Annie, 2017e). At App Annie, customer information such as customer reviews and ratings is available, as are important business data including rankings, downloads, revenues and prices. Furthermore, App Annie has three different ranking lists: free downloads, paid downloads and revenue. It offers further classifications based on the stores where the apps are sold. Lastly, it is possible to select a specific time frame to analyse (App Annie, 2017c).
For this paper, the iOS Health and Fitness data from the United States were analysed. Since the topic of this paper is based on the performance of app developers, apps were chosen based on their revenue performance. We sampled apps which were active between 1st March, 2016 and 28th Feb, 2017. In this way, the sampled apps have survived at least for one year. The reason assumed is that, in a long-tail industry like mobile apps, many apps lose before they actually take off. Including such apps in a study may lead to the contamination of confounding factors. The ‘Top App’ rank lists the best 1,000 apps in the category. From these 1000 apps, 43 apps are responsible for 75% of the total revenue in the Health and Fitness market[3]. The 43 top apps were analysed, wherein nine of these apps have both a free and a paid version. This results in 52 apps to analyse. Out of these 52 apps, two were

[3]The overall market revenue for the chosen time frame is $76,381,251; 75% of that is roughly $57,285,938. The first 43 apps are counted together for an overall revenue of $57,365,700

excluded. The first is the app '7 Minute Workout Challenge'. Even though this app is placed at No. 9, the last version update was in April 2015. The other excluded app is the paid version of 'Walking for Weight Loss: Training Plans, GPS, Tips', since this app has only two reviews. An overview of all chosen apps, their rankings, revenues and number of version updates within the given time frame can be found in Appendix A3.

Therefore, the population of this study consisted of 50 apps in total, which include 189,527 reviews. Within given budget constraints, it was not possible to analyse all the reviews. For example, each review needed to be analysed at least once via Amazon Mechanical Turk. The smallest possible cost per HIT (per review) would have been $0.02, resulting in $3,790. For this reason, a sampling procedure was chosen. The choice between a non-probability sampling and probability sampling procedure is discussed below.

The first possible data collection procedure was probability sampling, namely systematic sampling. Probability sampling procedures give every single review in the population the chance of being included in the sample (chance is not equal to zero). With the systematic sampling procedure, every 10th or 5th review would have been selected (Mooi & Sarstedt, 2011). The other option was a non-probability sampling procedure. Review data were collected based on a specific version update of the app. The version update needed to be in the chosen time frame (01.03.2016 - 28.02.2017). The first step was to choose the version update, and the second step was to collect the review data based on the version update. Most of the analysed apps had at least one version update in the given time frame. If there was more than one version update, the one with many changes / features was chosen. The purpose of the study is to investigate if consumers' feedback was taken into account in the app development process. The research question is only testable when developers enumerated their changes. Two different types of version updates are shown in Appendix A4. Hence, when more than one version update exists in the given time frame, the one with more enumerations has been selected for analysis. Therefore, the sampling procedure adopted was a non-probability procedure. The reviews were collected based on the version update.

Reviews were collected eight weeks prior to the update and four weeks after the update. In section 6.2.1, the review distribution is discussed. The sampling of all 50 apps finally consisted of 40,619 reviews, which represents 21% of the total population.

Next, the extracted review data were uploaded to Amazon Mechanical Turk for categorisation. However, the review data contain significant portions of ASCII charac-

ters, which are not supported on the Amazon Mechanical Turk platform. The Data Science Studio software was used to transform these data. The steps as well as the codes can be found in Appendix A5.1.

With matched review and updates data, the next step was to prepare the data for linear regression analysis, where the text data needed to be converted into numerical data. Unlike traditional data analysis, tables are the major form of the data. One feature of big data analysis is the large variety of data available for analysis, such as text data, image data, voice data etc. Hence, in the following chapters, the author will describe how the unformatted data such as text data were transformed into formatted numerical data for analysis.

6.2 Descriptive Statistics

6.2.1 Review Distribution over Time

As Pagano stated, most user reviews are written after a new version release. The review distribution is not exponential, but users write most feedback in the beginning and less over time. He concluded that user reviews are triggered by new version updates (Pagano & Maalej, 2013, p.127). Therefore, it is assumed that the most important time are the first weeks after a new version update. To check this assumption, apps were randomly chosen and review publication times analysed. Via the histograms, one can see the number of reviews at a specific date and the version updates of this app (see histograms in the Appendix A8). While most reviews are written immediately after a new version update, when manually screening hundreds of reviews in the first weeks after a new release, one can see that most reviews are written immediately when something is no longer working. Figure 8 shows typical reviews from users after a new version update (update date: 07.01.2017)

★☆☆☆☆ 1.0	**Need another update** by **Heather613** This app is great and worked just fine before the update today. Now whenever I try to add a recipe or go to my log, it crashes. I deleted and re-downloaded the app, it worked fine for a few minutes t hen started crashing again.	Jan 09, 2017
★☆☆☆☆ 1.0	**Oh no! An upgrade!** by **E from MA** I liked this program until the last upgrade! I wanted a simple food tracker that's resident on my pho ne. I don't want something I have to sign into that's web based. BOOOO! Now it's useless to me. Th ey should have kept it the way it was.	Jan 12, 2017
★☆☆☆☆ 1.0	**Freezes with new update** by **Whdgughj** Love the app and paid for premium, but can't even use it with this new update. It freezes at the logi n screen so I can't even get into my account and freezes on Apple Watch too. Please fix ASAP. I'm now paying for something I can't even use?!	Jan 14, 2017

Figure 8: Typical reviews after new version release

It is assumed that users had the latest update of the app on their smartphone, and they realised that the app was no longer working as it used to, so they wanted to get in touch with the developers to let them know. It is also assumed that, in the long run, after a new update, the number of reviews may become fewer, but the voices of the customers will change. In the long run, users talk about their experiences with the app, whether they like it or not, and state their wishes for new features or better implementation ideas for existing features. Consequently, the time frame was set to 12 weeks: eight weeks prior to get long-term feedback, and four weeks after to get the immediate reaction.

6.2.2 Category and Rating Distribution

The following Tables 9 and 10 show the frequency distribution of *generic* and *specific* reviews as well as the distribution of ratings.

Answer.sentiment

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	general	33761	81,5	81,5	81,5
	specific	7654	18,5	18,5	100,0
	Total	41415	100,0	100,0	

Figure 9: Frequency *generic* and *specific* reviews

Input.Rating

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	1	2117	5,1	5,1	5,1
	2	821	2,0	2,0	7,1
	3	1217	2,9	2,9	10,0
	4	5453	13,2	13,2	23,2
	5	31807	76,8	76,8	100,0
	Total	41415	100,0	100,0	

Figure 10: Frequency Rating

As expected, most reviews did not hold useful information and were classified as generic; however, with 81.5% generic and only 18.5% specific reviews, these results were higher than expected. In comparison, Chen et al. classified 35.1% of all app reviews as helpful for developers and 64.9% as not (Chen et al., 2014). One possible explanation is that reviews in Chen's research differ regarding the different app categories. For example, some categories such as Gaming might receive more useful feedback than others. Chen's dataset consisted of four different apps from four different categories.

For the rating distribution, 76.8% of all reviews had a 5-star rating, followed by 4 stars with 13.2%, and the average rating was 4.82. These findings imply that most of the users (90%) were, in general, satisfied with the analysed applications.
These results are bit higher than in Pagano and Maalej's study, where 61.96% gave a 5-star rating and 78.07% at least a 4-star rating (Pagano & Maalej, 2013).
Figure 11 shows the rating distribution for *generic* and *specific* reviews. As supposed (also see Figures 3 and 4), *generic* reviews had high star ratings (5 and 4 stars counting together 97.81%), *specific* reviews were more diversely distributed; most of them had a 5-star rating (33.77%), 1-star rating (24.11%), and 4-star rating (22.60%).

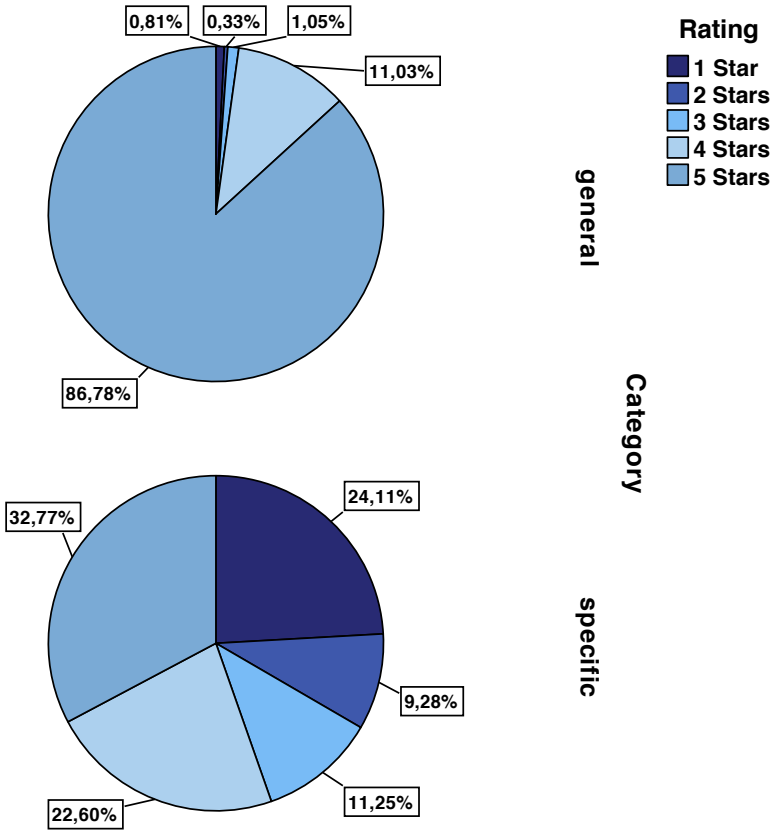


Figure 11: Distribution *generic* and *specific* reviews with rating

Furthermore, the distribution of *specific* and *generic* reviews differs per analysed app (an overview of all apps and their *generic* / *specific* distribution can be found in the Appendix A11). Therefore, it became clear that there is a correlation between *generic* and *specific* reviews and rating. Apps with a high amount of *generic* reviews tend to have a higher star rating than apps with a high amount of *specific* reviews. Table 1 shows some examples.

High Generic			High Specific		
App Name	**% Generic**	**⌀ Rating**	**App Name**	**% Specific**	**⌀ Rating**
Calm	85,6%	4.8	Sweat with Kayla	62.8%	2.7
Fitness Buddy	85.7%	4.6	Beachbody	71.1%	2.9
Instant Heart Rate+	88.5%	4.7	Fitplan	57.1%	3.3
Life Period Tracker	90.6%	4.9	Lifesum	49.6%	3.7
Relax Melodies	92.2%	4.8	Weight Watchers	50.4%	3.4

Table 1: Differences in average ratings of *generic* and *specific* reviews

Therefore, it seems that high-star ratings in general contain less important and less insightful information for developers in terms of improving software quality. Developers should rather focus more on the lower-rated reviews.

The 18.5% (7,654) remaining *specific* reviews were further classified, where nine reviews were lost in the second classification round with Amazon Mechanical Turk; hence, only 7,635 were further classified. Out of these 7,635 reviews, 3,604 (25.75%) were classified as *User Experience*, 3,377 as *Rating* (24.13%), 3,363 (24.03%) as *Bug*, 2,240 (16.01%) as *Feature Request*, 741 (5.29%) as *Pricing*, 390 (2.79%) as *Too Many Advertisements*, and 280 (2.00%) as *Other*, see Figure 12.

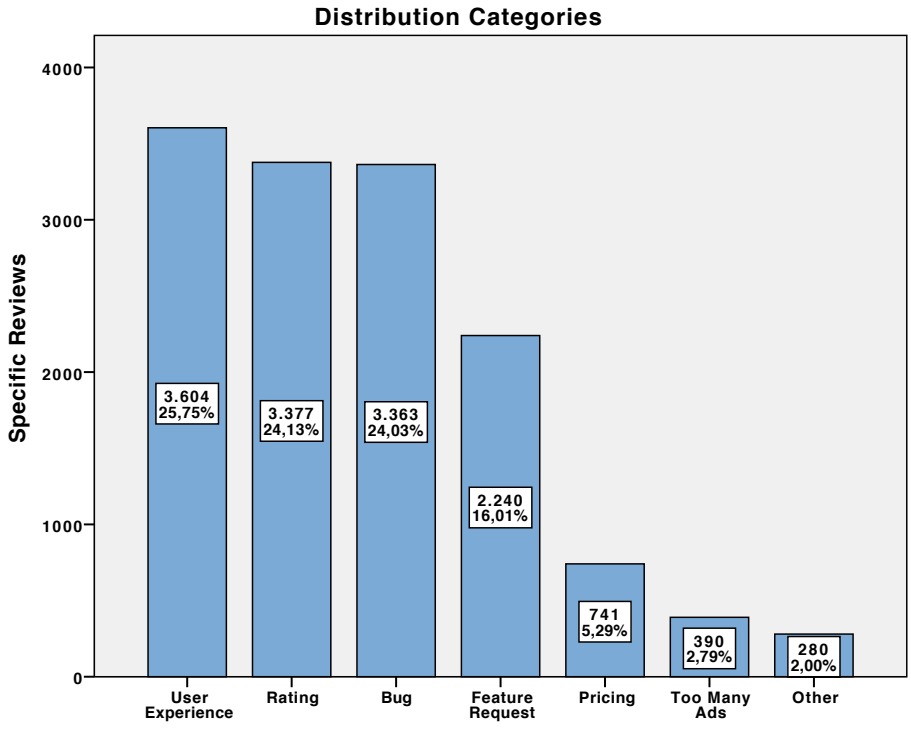


Figure 12: Distribution Categories

In comparison, in Pagano and Maalej's study, 50.3% were classified as *Rating*, 21.1% as *User Experience*, and 20% as *Requirements*, which are *Feature Requests* and *Bugs* (Pagano & Maalej, 2013). The results are quite different, except for *User Experience*; this may be due to the fact that *generic* reviews had been already sorted out in this research.
A closer look at the review distribution of the categories *User Experience* and *Rating* derived from the second classification round supports the assumption that they tend to have higher ratings. *Rating* had 52.06% of 5-star ratings and at least 77.61% of 4-star rating reviews, with an average rating of 4.03. Similarly, *User Experience* had 45.34% 5-star and at least 68.7% 4-star ratings, with an average rating of 3.75. Pagano and Maalej had an average rating of 4.95 stars for their topics *helpfulness, praise, recommendation*, which are similar to the two categories *Rating* and *User Experience*.
It is assumed that the lower star ratings in the second round are due to the fact that some of them should have been already sorted out as *generic* reviews, and others may belong to the category *Feature Request*. During the manual check of the MT results, it became clear that workers often mixed up the two categories *User Experience* and *Feature Request*, because of laziness and not misunderstanding. *User Experience* reviews tend to be long text descriptions, and, often, a *Feature Request* is hidden in the last sentence; therefore, it is assumed that they had not read the comment to the end.
Moreover, *Bug* had an average rating of 2.29, *Feature Request* of 3.49, *Too Many Ads* of 2.35 *Pricing* of 2.63, and *Other* of 2.74 (note that a pie chart with the rating distribution of all categories can be found in the Appendix A13). These results tend to differ a bit from Pagano and Maalej's results, where *Feature Request* had an average rating of 4.37 stars and *Bug* of 1.21 stars. This could be due to classification problems again; however, it was observed that *Feature Requests* had, on average, rather a 3- or 4-star rating than a 5, since users are in general quite satisfied with an application, but missed a specific function, which led to 1-star subtraction. Hence, Pagano and Maalej's 4.37 stars seems rather a bit too high. The low averages of *Bug*, *Too Many Ads*, and *Pricing* make sense, since all of them are associated with a negative sentiment and decreased user satisfaction. The rating distribution of the category *Other* seems rational too, since no pattern is recognisable in the distribution.

6.3 Data Preparation for Linear Regression Analysis

The independent and moderating variables were derived in several steps via detailed descriptions including the Python code. The variables computed can be found in

the Appendix A6 and Appendix A7

6.3.1 Extraction Independent Variables

The obtained data from MT is a huge Excel file with over 200 columns, with the original columns enumerated in the Appendix A5.2, plus informational columns added from the MT side (e.g. "AssigmentDurationInSeconds"), as well as the result columns from the classification. As shown in the Figures 6 and 7 in Chapter 4, MT workers needed to enumerate each found issue in a separate text field and select possible matches from the dropdown list. If, for instance, a bug text field was used, the column Bug_Text_1 was created in the background. If another bug text field was used, Bug_Text_2 was created, respectively. If there was a match between one of these text fields and a feature update, a new column called Bug_Text_1_Feature_Linked was created, which received the value of the linked feature update (e.g. Apple Watch Compatibility). Hence, matches could be detected if the Feature_Linked columns had a filled cell.
Next, the filled cells in the Feature_Linked columns were counted with a Python code. The independent variable *Customer Led Innovation* counts filled cells from the Feature_Request_Feature_Linked columns, and the *Customer Led Improvement* variable counts filled cells from the Bug_Feature_Linked and Too_Many_Ads_Feature_Linked columns. The variable *Developer Led Innovation* counts unfilled cells from the Feature_Request_Feature_Linked columns, and the *Developer Led Improvement* variable counts unfilled cells from the Bug_Feature_Linked and Too_Many_Ads_Feature_Linked columns, respectively.

6.3.2 Extraction Response Time Variables

Response time for Customer Led Innovation and Customer Led Improvement

The response time variables from the customer's perspective measures the time when the problem / wish was mentioned for the first time until it was fixed / launched. Therefore, the data needed to be prepared in a way that, for all matches, the first review was found and the time span computed.
To that end, the dataset was transformed. Until now, a row represented one review, and the feature updates were listed in columns. However, in this way, it was not possible to compute the time span for each feature update. Therefore, the dataset was transformed, so that one row represented a feature update. Now, for each matched feature update, the earliest review could be detected based on the difference in days from the feature update date and the review date.

Response time for Developer Led Innovation and Developer Led Improvement

For the response time on the developer side, the complete update history on App Annie was downloaded and modified. For the *Developer Led Improvement* response time, the difference in days from the analysed update and the previous update was computed. It is assumed that developers fixed at least one issue in every update. However, it is not assumed that they launched a new feature in every update. For this reason, for the *Developer Led Innovation* response time, the next future update was chosen where a new feature was launched. Sometimes, it was necessary to skip one future update, because no new features had been launched. While going manually through the application updates, it became clear that every publisher introduces new features with some explanatory words. Hence, it was easy to detect if new features were launched or not. Note that the response time was calculated on a one-feature basis. This means that, if in the next future update two new features were released, the difference in days was divided by two to receive the time span in days per feature.

7 Results and Discussion

In this section, the estimators for regressions will be first presented, which are used to test hypotheses 1 to 7 as in Chapter 3, followed by the model performance.

Coefficients[a]

Model		Unstand. Coefficients		Stand. Coefficients				Hypothesis	
		B	**Std. Error**	**Beta**	**t**	**Sig**	**F**	**Hyp**	**Status**
1	(Constant)	241998,174	63816,770		3,792	**,000**	5,517		
	Customer_Led_Improvement	7292,185	2456,607	,523	2,968	**,005**		H1a	supported
	Customer_Led_Innovation	-7761,506	8906,398	-,154	-,871	,388		H1b	rejected
2	(Constant)	193276,114	70499,949		2,742	**,009**	6,846		
	Developer_Led_Innovation	12733,269	4965,608	5,551	2,564	**,014**		H2a	supported
	Developer_Led_Improvement	-6084,074	2533,338	-5,199	-2,402	**,020**		H2b	supported
3	(Constant)	192585,641	60037,521		3,208	**,002**	7,341		
	Customer_Led_Improvement	29805,931	12470,637	2,139	2,390	**,021**			
	MV_Customer_Led_Improvement	-612,970	316,019	-1,736	-1,940	**,058**		H3b	supported[b]
4	(Constant)	237366,199	65082,121		3,647	**,001**	4,457		
	Customer_Led_Innovation	-14198,564	11458,130	-,281	-1,239	,221			
	MV_Customer_Led_Innovation	37,904	14,565	,590	2,602	**,012**		H3a	supported
5	(Constant)	145341,993	70829,287		2,052	**,046**	6,987		
	Developer_Led_Improvement	661,821	180,136	,566	3,674	**,001**			
	MV_Developer_Led_Improvement	-1,079	,413	-,402	-2,612	**,012**		H4b	supported

a Dependent Variable: Revenue_sum

b Only significant with alpha 0.1

Table 2: Multiple Regression Analysis Model 1 - 5

Coefficients[a]

Model		Unstand. Coefficients		Stand. Coefficients				Hypothesis	
		B	**Std. Error**	**Beta**	**t**	**Sig**	**F**	**Hyp**	**Status**
6	(Constant)	182052,338	73946,888		2,462	**,018**	4,217		
	Developer_Led_Innovation	910,197	316,931	,397	2,872	**,006**			
	MV_Developer_Led_Innovation	-2,310	2,112	-,151	-1,094	,280		H4a	rejected
7	(Constant)	154813,820	74208,289		2,086	**,043**	5,426		
	Developer_Led_Innovation	34292,001	15246,276	14,950	2,249	**,029**			
	Developer_Led_Improvement	-16345,722	7312,617	-13,968	-2,235	**,030**			
	Customer_Led_Improvement	-11845,611	7932,398	-,850	-1,493	,142			

a Dependent Variable: Revenue_sum

Table 3: Multiple Regression Analysis Model 6 - 7

The positive effect on app performance through user involvement in the app development can only be partially supported (Table 2, Model 1). *Customer Led Improvement* has a positive effect on revenue, but *Customer Led Innovation* cannot be supported. The overall model is significant, with a p-value of 0.007, which is smaller than alpha 0.05.
Hypothesis 1a, that *Customer Led Improvement* has a positive impact on app performance, can be supported ($0.005 < 0.05$). However, only the effect on revenue is significant, not on downloads, rating or ranking. Increasing *Customer Led Improvement* by one unit does increase revenue by $7,292.19. Note that the increase in revenue refers to a time frame of 12 weeks. This means that one transformed user improvement feature into the product development results in a revenue increase of $7,292.19 per quarter. The R^2 of 0.190 is satisfactory, since this is an exploratory research, using cross-sectional data, where values of 0.10 are typical (Mooi & Sarstedt, 2011).
Customer Led Improvement can have a positive impact on revenue; hence, it is of more interest for developers to find bugs mentioned in reviews and fix them. It makes sense that bugs reported by users can increase app performance. It is challenging for developers to find every bug in the jungle of different smartphones, of different versions, and software updates. Therefore, it can be valuable for users to notify developers about them. To do this, developers could use the MARK framework, built by Vu and his colleagues.
We saw that, in isolation, *Customer Led Improvement* has a positive impact on revenue, but not in combination with *Developer Led Improvement* and *Developer Led Innovation* (see table Model 7). One possible explanation is that apps somehow rely on input from somewhere, so *Customer Led Improvement* has a positive effect on revenue when it is the only input available.

Customer Led Innovation (H1b) is not significant ($0.388 > 0.05$); the direction, however, has a negative sign. This might be due to the fact that requested features are not be very representative. Users may wish features that would increase their own utility, but not the utility of all other users. Moreover, user feature requests may not be feasible for developers. Users may come up with creative and original ideas, but they are not implementable. This can be due to a lack of technical expertise as to what is feasible in app development (Magnusson, 2003). Hence, developing such features is simply not possible or is too expensive and not economical for developers. The proposed AR-Miner tool from Chen et al. could help developers as to which features to implement and which not.

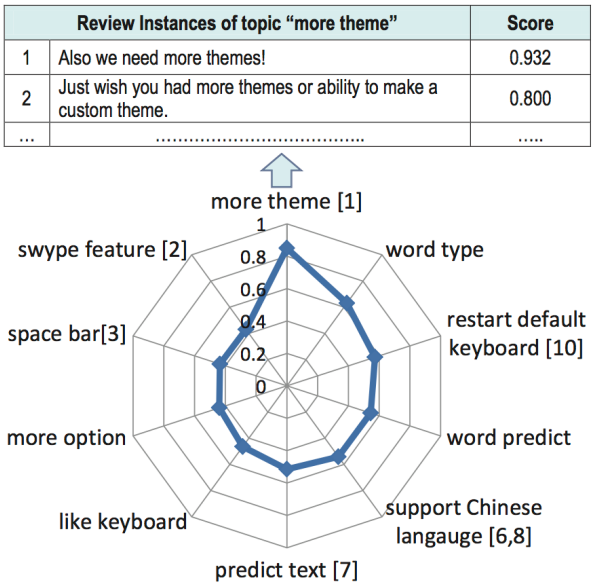


Figure 13: Visualisation of Top-10 ranked results achieved by AR-Miner (Source: Chen et al. (2014))

Figure 13 shows the AR-Miner tool of Chen et al., where the top-10 ranked topics are visible in the radar. The closer a topic is to the grids end, the higher the *"GroupScore"* and the more requested the feature is by user reviews (Chen et al., 2014). Hence, should developers decide to develop a user feature, it should be the most requested one, as indicated by AR-Miner.

Both Hypotheses, H2a and H2b can be supported (Table 2, Model 2), where *Developer Led Innovation* has a positive impact and *Developer Led Improvement* has a negative impact on app performance. However, only the effect on revenue is significant, not downloads, rating or ranking. The overall model is significant ($0.02 < 0.05$), and both variables, where the effect from *Developer Led Innovation* ($0.014 < 0.05$) is slightly stronger on revenue than *Developer Led Improvement* ($0.02 < 0.05$). With the increase by one innovation unit, revenue increases by $12,733.27.
Interestingly, *Developer Led Improvement* has not a positive impact (H2a), as expected, but a negative with minus $6,084.07 per quarter. This might be due to the fact that it is not economical to search for flaws when customers notify developers anyway. This does not mean that developers do not need to fix bugs; rather, they should not search for them actively themselves.
Moreover, it does support the theory that users may not like frequent updates, and developer led improvement comes with more frequent updates. Frequent updates mean two things: first, it means change, and users dislike any change that

does not make their life better. Change may force them to relearn everything they knew (Wodtke, Christina, 2013). Second, every update comes with the risk of new software bugs which lowers the quality of the app from the customers perspective. Hence, users may associate new updates not only with bugs, but also with the need to relearn everything, which decreases their utility.

In contrast, *Developer Led Innovation* is not only supported to be effective in isolation, but also in combination with the other predictors, where the strength is even increased (H2b). Moreover, from all independent variables, *Developer Led Innovation* has the strongest impact on revenues. This reveals not only that innovation is much more effective when it comes from the developers side, but also that innovation itself must be crucial for any apps success. For more ideas as to which features developers should implement, they could mine features from their category to find valuable input. Harman and colleagues found a strong correlation between downloads and competitors features (Harman et al., 2012). While innovation is inspired by competitors, these features might be missing in an application and are highly valued by customers.
However, these findings seem to be limited, because the scatterplots show something else: namely that high-revenue apps are not associated with any independent variable (see the other scatterplots in the Appendix A14). The scatterplot from Figure 14 shows that all occurrences on the upper left side are high-revenue apps, which are independent from *Developer Led Innovation*. Because of that result, the top-10 revenue strongest apps were further studied.

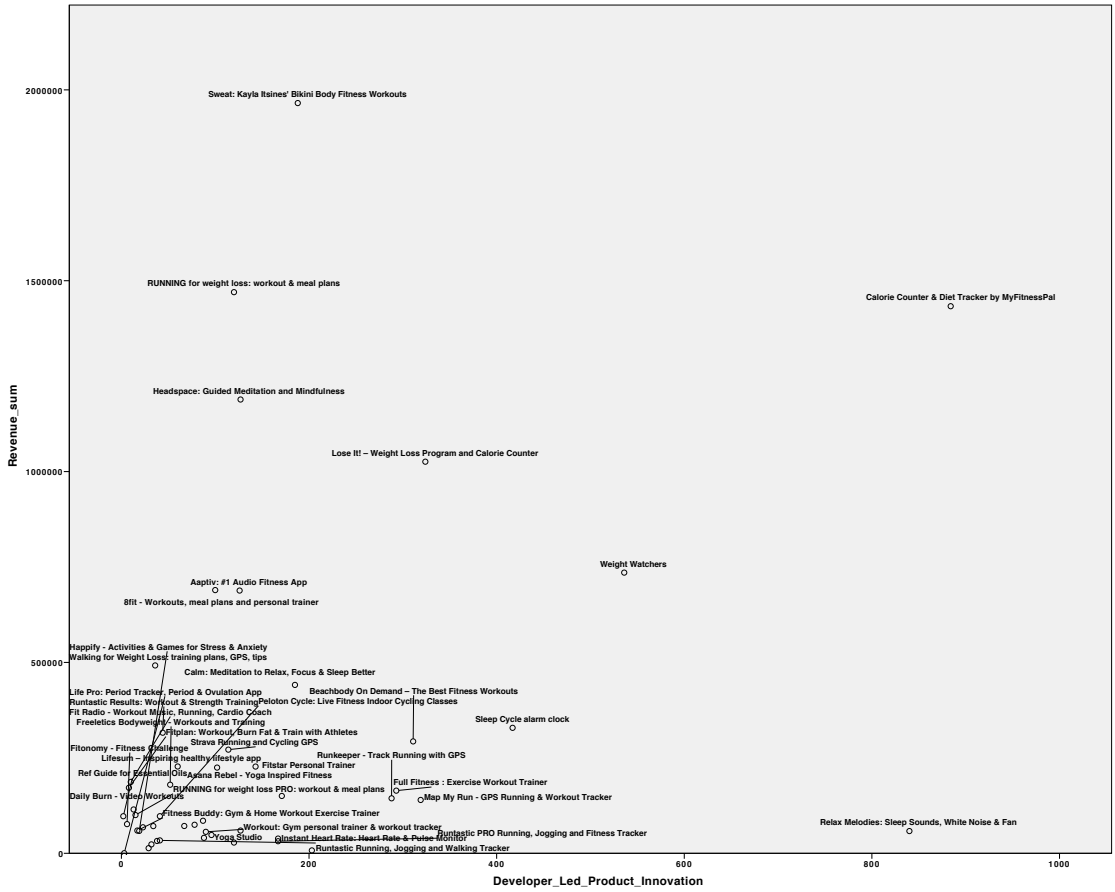


Figure 14: Scatterplot *Revenue* against *Developer Led Innovation*

Table 4 counts the occasions when the top-10 revenue apps seem to have no connection with the independent variables, via how often the apps occur on the upper left side of the scatterplot. 'Sweat with Kayla', for example, never shows a connection to any independent variable. Therefore, in total, it can be said that 67,50% the top-10 revenue apps have no connection to the independent variables *Developer- and Customer Led Innovation* and *Improvement*, which limits the results.

App Name	Revenue	CIN[a]	CIM	DIN	DIM	Frequency
Sweat with Kayla	$1,965,418	✓	✓	✓	✓	4/4
RUNNING for weight loss	$1,470,036	✓	✓	✓	✓	4/4
MyFitnessPal	$1,433,213	✗	✗	✗	✗	0/4
Headspace	$1,188,792	✓	✓	✓	✓	4/4
Lose It!	$1,025,913	✗	✓	✗	✗	1/4
Weight Watchers	$735,332	✗	✗	✗	✗	0/4
8fit	$689,535	✓	✓	✓	✓	4/4
Aaptiv	$688,296	✓	✓	✓	✓	4/4
Walking for Weight Loss	$492,135	✓	✓	✓	✓	4/4
Calm	$441,124	✓	✓	✗	✗	2/4
Sum						**67,50%**

a CIN is the abbreviation for *Customer Led Innovation*, CIM for *Customer Led Improvement*, DIN for *Developer Led Innovation*, and DIM for *Developer Led Improvement*

Table 4: Top Revenue Apps in the Scatterplots

The question remains as to why these apps are much more successful than others. The further assumption that app success correlates with the number of published apps by a publisher was also not supported. It may be that top-revenue apps need less improvement and innovation than others, as these apps have earned a strong reputation among the health and fitness apps and are already well established with less need for innovation. Moreover, it may be because they have already given themselves a name and gained popularity, rely on strong marketing activities, or have been featured on the app store. 'Sweat with Kayla', for example, became famous through Instagram. Kayla Itsines is an Instagram influencer with almost eight million followers, and her app was only developed after she became famous (Asseldonk, 2017). 'Headspace' has fewer followers on Instagram (276,000), but they additionally use paid advertisements on Instagram. Travel and Lifestyle influencer Amelia Liana has a cooperation with '8fit' and 'Weight Watchers', working together with ambassador Oprah Winfrey (Gesenhues, 2016; Schlossberg, 2015) (all Instagram examples can be found in the Appendix A15). Numerous online articles have stated that apps can only go viral under certain conditions. Social media is especially important in the Health and Fitness category, as it is all about sharing ones progress and activity. In addition, content marketing, web site presence, and community building (Giggs, 2016; Taş, 2017) are all important contributors. Moreover, whether an app is featured on the Apple App Store or not can have a huge impact on downloads and

revenue (Apple Developer, 2017a; Fiegerman, 2014; Hunter, 2012).
However, none of these possible influences were studied in this research. The scatterplots simply revealed that the high-revenue apps under study may not be a result of customer- and developer led innovation and improvement.

The question as to whether response time has a moderate impact on app performance is addressed with models 3 - 6.
Response time in addressing customer improvement and innovation (H3a and H3b) can be partially supported, where the moderator variable to address user innovation is significant ($0.012 < 0.05$) but, for user improvement, only with an alpha of 0.1 ($0.058 > 0.05$). It is somehow unclear why the moderator variable for customer led innovation is significant while the independent variable *Customer Led Innovation* is not. One conjecture could be that there is a direct effect of the response time of customer innovation on revenue, but such a variable is not tested here; hence, the moderator variable captured the effect.
H3b, the effect of the response time variable on customer improvement, can only be supported with an alpha of 0.1. The moderator variable for *Customer Led Improvement* does amplify the effect of *Customer Led Improvement* on revenue from \$7.292,19 to \$29.805,93; hence, the relationship strengthens by more than \$20.000 if developers address user improvement inquiries faster. This result is as assumed, because bugs decrease the quality and limits the use of the app.
Response time to address developer improvement (H4b) can be supported. Interestingly, the effect of *Developer Led Improvement* on revenue increased from \$-6.084,074 to \$661,821 by almost \$7.000 with faster addressed response time. Generally speaking, this result is as expected, since, similar to the response time variable on customer improvement side, quickly addressing bugs increases the quality of the app. The result is also in line with previous studies, which determined that constantly updating an app has a positive impact on sales (Lee & Raghu, 2014) and increases downloads (Comino et al., 2016). Thus, the concern that users dislike frequent updates in general is not valid, which also supports the findings of Comino et al. They found that developers not only update their apps because of new features and bugs, but they also need to stimulate 'buzz' to survive in the 'hyper-competitive' mobile market. Buzz is generated with a new updated version, because it can be talked about on blogs, online magazines and social media platforms, which then triggers downloads (Comino et al., 2016).

Lastly, H4a, the effect of the moderator variable on the independent variable *Developer Led Innovation*, cannot be supported ($0.280 > 0.05$). It was assumed to have

a positive effect on revenue, because a faster response time means that, in a shorter period of time, more innovative features are released. However, this result reveals that quality is more important than quantity. Radical change needs time and cannot be released within a few weeks. Radical change is a long-term, prospective goal and likely needs a longer horizon (Hobcraft, Paul, 2015).

Running the overall model with all four independent variables in the regression is not wise, because of multicollinearity issues. In running the regression with all four independent variables, SPSS automatically excludes the variable *Developer Led Innovation*. To control multicollinearity, the four independent variables were analysed again, each time with one of them as a dependent variable.

Coefficients[a]

Model		Collinearity Statistics	
		Tolerance	VIF
1	Customer_Led_Product_Improvement	,050	20,181
	Developer_Led_Product_Innovation	,000	2751,636
	Developer_Led_Product_Improvement	,000	2432,023

a. Dependent Variable: Customer_Led_Product_Innovation

Figure 15: Multicollinearity Diagnostic

Figure 15 clearly reveals that both developer side variables highly correlate with *Customer Led Innovation* with values over 2.000, but also *Customer Led Improvement* with a VIF over 20. Because of the collinearity issues, *Customer Led Innovation* was excluded from the regression; the variable in isolation (H1b) was already insignificant, anyway. Table 3, Model 7 shows the regression with the three predictor variables not excluded. Both developer side variables are significant with p-values of 0.029 and 0.030, which are smaller than α 0.05.
However, now *Customer Led Improvement* has a p-value of 0.142, which is not significant with an α value of 0.05 or 0.1. Despite the p-value, the direction of revenue changed too. An increase by one unit would result in a revenue decrease of \$16,345.72 per quarter. Hence, translating user involvement at all might be not wise. The beta coefficients $\beta_{\text{Developer Led Improvement}}$ and $\beta_{\text{Developer Led Innovation}}$ have both strengthened, whereas $\beta_{\text{Developer Led Innovation}}$ increased to \$34,292.00 and $\beta_{\text{Developer Led Improvement}}$ decreased further to \$16,345.72. Moreover, the predictive power of the model improved with an R^2 of 0.261 and Adjusted R^2 of 0.213. Therefore, *Customer Led Improvement* in the model explains some unique variance, which was not accounted for by *Developer Led Improvement* and *Developer Led Innovation* in the model alone.

Taking the above findings together reveals a bigger picture. Radical innovation is not produced by co-creation, wherein other persons such as customers are involved. Co-creation is only good for incremental changes, because customers create solutions based on their biased experiences (Gustafsson et al., 2012). Hence, it makes sense that *Customer Led Improvement* has a positive effect on revenue, and that the moderator variables strengthen that effect, because these can be seen as incremental improvements of the app. Incremental improvement is the kind of innovation that keeps companies going and that kind of innovation is released much faster (Gustafsson et al., 2012; Hobcraft, Paul, 2015). This explains the positive effect of the moderator variable for *Customer Led Improvement* and *Developer Led Improvement*. In contrast, *Customer Led Innovation* is not significant, because customers cannot produce innovation of radical change. This is why *Developer Led Innovation* has such a strong impact on revenue: it is radical change. Radical change is unthinkable in advance and needs time to be implemented. Hence, the moderator variable response time is not significant for *Developer Led Innovation*.
To sum up, these findings imply that *Customer Led Innovation* has no impact on revenue, and developers should rather listen to their own intuition and release features of radical change. Thus, developers must decide carefully whether to develop requested features or not. Nevertheless, when it comes to improving an application, it is much more economical to rely on customers reporting bugs and to fix them as soon as possible, rather than searching bugs themselves. However, these results might not be applicable for high-revenue apps which have already gained popularity by the time of this study.

8 Limitations and Future Research

Some limitations need to be taken into account. Besides the fact that the results seem not to be true for high-revenue apps, there are several other issues influencing the accuracy of the findings.
(1) The data sampling procedure may be questionable, whereby reviews were collected eight weeks prior and four weeks after a version update. While, for this research, another procedure was not an option, the risk is great that much information has been lost. It may be that many more matches are present, which were not captured by the data. Hence, a review containing a feature request prior to the eight-week period concerning product improvement and innovation would not be captured by the data. This problem can only be solved if all the data are analysed from the very beginning when the app was launched.

(2) Another data issue is the sample size, which needs to be sufficiently large; a rule of thumb is 104 + k where k is the number of independent variables (Mooi & Sarstedt, 2011). With four independent variables, the sufficient sample size should be 104 + 4 = 108. However, this study has only 50. Thus, there may be problems with low statistical power, raising the possibility that the findings do not reflect true effects (Button et al., 2013). Unfortunately, the sample could not be larger due to financial and time constraints.
(3) Furthermore, the version update data are problematic, because there are no guidelines or rules for developers to describe their version updates. It is left to the developers themselves to decide what will be published on their update descriptions. Hence, update descriptions differ in many respects and are not uniform among the apps. Some developers describe their changes in depth, while others only write 'we fixed some bugs'. For the latter, it is impossible to find matches, because it is unknown what they actually updated.
(4) Moreover, there should be no or little collinearity; however, as discussed earlier, some independent variables showed correlations (see the correlation matrix in the Appendix A16). The trouble with collinearity is that it tends to identify significant parameters as not significant (Mooi & Sarstedt, 2011).
(5) Furthermore, the assumption of homoscedasticity is not fulfilled. Figure 16 shows the scatterplot of the residuals against the predicted values of the dependent variable *Revenue*. If homoscedasticity is present, the scatter plot would have a rectangular

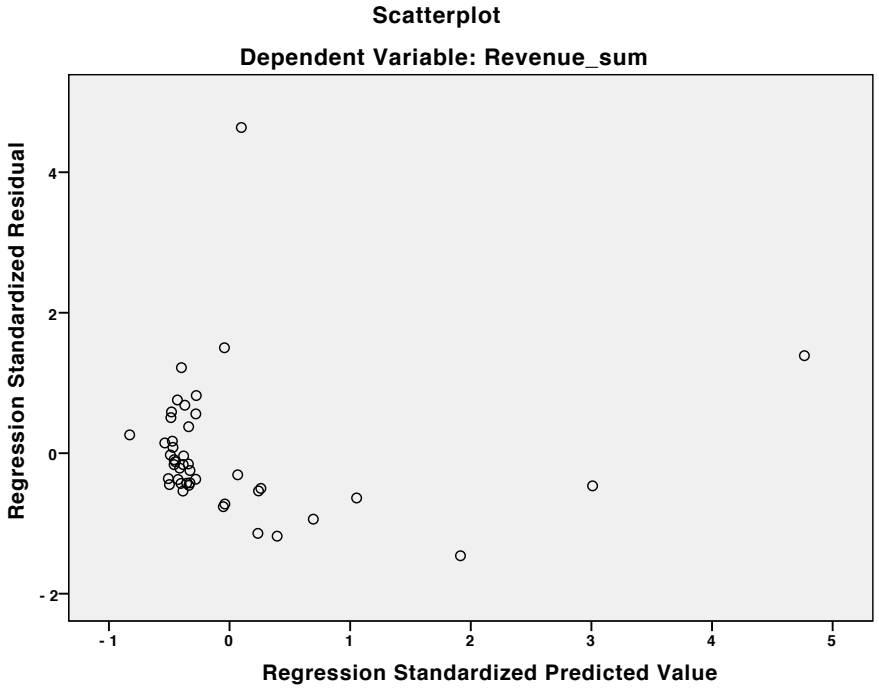


Figure 16: Scatterplot of the residuals against the predicted values of *Revenue*

shape, where the scores are concentrated in the centre, which is not present here. Hence, the standard errors are biased, which can lead to an incorrect significance of the regression (Statistics Solutions, 2013, 2017).
(6) The initial round to categorise reviews into *generic* and *specific* was not necessary per se; it would have been possible to start with the second round, as well. However, the reason was mainly because of cost efficiency. To keep the costs low, it was necessary to eliminate all noisy and irrelevant reviews in the beginning. This task was simple: and the payment was therefore low ($0.02/HIT). The second round was much more complex; therefore, the payment was higher ($0.05/HIT). Table 5 shows the differences in costs when working with one or two rounds, where the first number represents the number of analysed reviews.

One Round	**Two Rounds**
$41,415 * \$0,05 = \$2,070.75$	$41,415 * \$0,02 = \$828.30$
	$7,635 * \$0,05 = \$381.75$
$2,070.75	**$1,210.05**

Table 5: Calculation for the categorisation decision via Amazon Mechanical Turk

The decision to classify the reviews in two steps made it difficult to compare the results with other papers, such as, for example, with Maalej & Hadeer (2015) or Chen et al. (2014).
(7) Lastly, another major limitation was the accuracy of the classification done by the MT workers. This was conspicuous, because almost 50% were classified as *Rating* and *User Experience* reviews in the second round. Both categories should not have been chosen to that extent, since they do not hold specific user information. All kinds of these reviews should have been sorted out already during the first classification round (*specific* and *generic* classification). This means that, not only was the final dataset negatively impaired, but so was the amount of *specific* user reviews, which were the input for the final categorisation. It is unclear to what extent the reviews were categorised incorrectly. However, based on the 1.000 manually checked reviews, it is assumed that most mistakes happened in the first round, because of a wrong payment. The payment was set to $0.02 / review, the lowest possible payment, since the task itself was easy. Unfortunately, it seemed that workers rushed through the reviews to make as much money as possible. The payment was set much higher for the final classification, but it still appeared to be too low. It is assumed that, with a higher payment, the quality will increase; however, there might be always workers trying to make money quickly.

With the experience gained on the MT platform, it is highly recommended to find and keep good workers. As a recommendation for further cooperation with the platform, it is crucial to start with a small number of HITs, find good workers, and start the work with them.

However, even with good qualified workers, one can never ensure that the reviews will be classified correctly. Maalej and Khalid cite Herzig et al. as an example wherein manual, human classification leads to misclassification; Herzig and his colleagues analysed over 7,000 issue reports, and 33.8% of all bugs were incorrectly classified (Herzig et al., 2013; Maalej & Hadeer, 2015). However, these 33.8% bugs were not misclassified because of the manual work itself; rather, they were misclassified because users and developers do not share the same perspective, and bug-tracking systems are a tool whereby users report bugs which will be fixed by developers. Often, though, whenever something is not working for a user, they think it is a bug, when, in fact, it might be something else. A recommendation in reference to Herzig is that the initial training for a deep learning algorithm should always include human qualitative analysis as the input data (Herzig et al., 2013). This is why the tool presented here for review classification and matching, as well as the data cleaning process used in this research, can be re-used for further review analysis. Further research could use both tools in combination with highly qualified people to create an initial training set, which can be used as input data for machine-learning algorithms like Naive Bayes.

Future research should also distinguish between high-revenue apps from the rest, because it is assumed that they contain different mechanisms for development. High-revenue apps seem to be less dependent on development (innovation and improvement), because they have already gained their popularity. One of the biggest success factors for app popularity is their marketing activities, which is not covered by current literature.

References

Annie, A. (2017). *Market Data Intelligence.* https://support.appannie.com/hc/en-us/articles/219519908-Store-Intelligence-Product-Overview-Metrics-Methodology. (Last checked 25.09.2017)

App Annie. (2017a). *Calm - Reviews.* Retrieved from https://www.appannie.com/apps/ios/app/calm-meditate-sleep-relax/reviews/?start_date=2016-03-10&end_date=2017-02-28&country=US. (Lastchecked 18.06.2017)

App Annie. (2017b). *Headspace: Guided Meditation & Mindfulness - Reviews.* Retrieved from https://www.appannie.com/apps/ios/app/headspacecom-meditation-mindfulness/reviews/?start_date=2016-03-18&end_date=2017-02-17. (Lastchecked 18.06.2017)

App Annie. (2017c). *Store Intelligence iOS Top Apps.* Retrieved from https://www.appannie.com/intelligence/top/?feed=all&aggregation=unified&device=ios&country=US&category=health-and-fitness&granularity=daily&date=2017-06-10~2017-06-10&sort_by=value. (Lastchecked 12.05.2017)

App Annie. (2017d). *Sweat: Kayla Itsines' Bikini Body Fitness Workouts - Reviews.* Retrieved from https://www.appannie.com/apps/ios/app/sweat-with-kayla/reviews/?start_date=2016-03-01&end_date=2017-02-28&country=US. (Lastchecked 18.06.2017)

App Annie. (2017e). *We deliver data and insights to help build better app businesses.* Retrieved from https://www.appannie.com/en/about/?_ref=footer. (Lastchecked 17.05.2017)

App Annie. (2017f). *Weight Watchers - Reviews.* Retrieved from https://www.appannie.com/apps/ios/app/weight-watchers-mobile/reviews/?start_date=2016-05-28&end_date=2017-02-27&rating=3. (Lastchecked 27.06.2017)

Apple Developer. (2017a). *Discovery on the All-New App Store.* https://developer.apple.com/app-store/discoverability/. (Last checked 24.10.2017)

Apple Developer. (2017b). *Ratings, Reviews, and Responses.* https://developer.apple.com/app-store/ratings-and-reviews/. (Last checked 27.09.2017)

Armerding, Taylor. (2012). *Why users don't often upgrade software when they should.* https://www.csoonline.com/article/2132061/security-awareness/

why-users-don-t-often-upgrade-software-when-they-should.html. (Last checked 11.12.2017)

Asseldonk, E. v. (2017). *How a fitness entrepreneur and Instagram celeb went from being an 18-year old personal trainer to running a team of 20 and reaching millions.* Retrieved from http://www.businessinsider.de/kayla-itsines-interview-sweat-bikini-body-guide-2017-6?r=US&IR=T. (Lastchecked 23.10.2017)

Barki, H., & Hartwick, J. (1989). Rethinking the concept of user involvement. *MIS quarterly*, 53–63.

Barlow, J., & Møller, C. (1996). *A complaint is a gift: using customer feedback as a strategic tool.* Berrett-Koehler Publishers.

Berger, C., Möslein, K., Piller, F., & Reichwald, R. (2005). Co-designing modes of cooperation at the customer interface: learning from exploratory research. *European Management Review*, *2*(1), 70-87.

Button, K. S., Ioannidis, J. P., Mokrysz, C., Nosek, B. A., Flint, J., Robinson, E. S., & Munafò, M. R. (2013). Power failure: why small sample size undermines the reliability of neuroscience. *Nature Reviews Neuroscience*, *14*(5), 365–376.

Carreño, L. V. G., & Winbladh, K. (2013). Analysis of user comments: an approach for software requirements evolution. In *Software engineering (icse), 2013 35th international conference on* (pp. 582–591).

Chandy, R., & Gu, H. (2012). Identifying spam in the ios app store. In *Proceedings of the 2nd joint wicow/airweb workshop on web quality* (pp. 56–59).

Chen, N., Lin, J., Hoi, S., Xiao, X., & Zhang, B. (2014). AR-miner: mining informative reviews for developers from mobile app marketplace. *Proceedings of the 36th International Conference on Software Engineering*, 767–778.

Cheung, K. W. (2013). *A Feature Request Is A Bug... Is A Dumb Detail.* https://www.getdonedone.com/a-feature-request-is-a-bug-is-a-task-is-a-to-do-is-a-dumb-detail/. (Last checked 28.06.2017)

Comino, S., Manenti, F. M., & Mariuzzo, F. (2016). Updates management in mobile applications. itunes vs google play.

Feldman, R. (2013). Techniques and applications for sentiment analysis. *Communications of the ACM*, *56*(4), 82–89.

Fiegerman, S. (2014). *What Happens After Apple Features Your App in the App Store.* http://mashable.com/2014/01/03/apple-app-store-features/#WbD8cYfOCOqF. (Last checked 24.10.2017)

Gesenhues, A. (2016). *Most exciting development for Weight Watchers' top marketer? Oprah, of course!* Retrieved from https://marketingland.com/exciting-development-weight-watchers-top-marketer-oprah-course-190073. (Lastchecked 23.10.2017)

Ghose, A., & Han, S. P. (2014). Estimating demand for mobile applications in the new economy. *Management Science*, *60*(6), 1470–1488.

Giggs, J. (2016). *Health and Fitness Apps Promotional Methods: Catching the Wave of New Year Resolutions.* Retrieved from http://www.mobyaffiliates.com/blog/health-and-fitness-apps-promotional-methods-catching-the-wave-of-new-year-resolutions/. (Lastchecked 23.10.2017)

Google Cloud Platform. (2017). *Google Cloud Natural Language API.* Retrieved from https://cloud.google.com/natural-language/. (Lastchecked 13.09.2017)

Gustafsson, A., Kristensson, P., & Witell, L. (2012). Customer co-creation in service innovation: a matter of communication? *Journal of Service Management*, *23*(3), 311–327.

Guzman, E., Ibrahim, M., & Glinz, M. (2017). A little bird told me: Mining tweets for requirements and software evolution. In *Requirements engineering conference (re), 2017 ieee 25th international* (pp. 11–20).

Guzman, E., & Walid, M. (2014). How do users like this feature? a fine grained sentiment analysis of app reviews. *Requirements Engineering Conference (RE), 2014 IEEE 22nd International*.

Harman, M., Jia, Y., & Zhang, Y. (2012). App store mining and analysis: Msr for app stores. In *Mining software repositories (msr), 2012 9th ieee working conference on* (pp. 108–111).

Hedegaard, S., & Simonsen, J. G. (2013). Extracting usability and user experience information from online user reviews. In *Proceedings of the sigchi conference on human factors in computing systems* (pp. 2089–2098).

Herzig, K., Just, S., & Zeller, A. (2013). It's not a bug, it's a feature: how misclassification impacts bug prediction. In *Proceedings of the 2013 international conference on software engineering* (pp. 392–401).

Hobcraft, Paul. (2015). *Innovation needs different time and thinking horizons.* https://paul4innovating.com/2015/06/01/innovation-needs-different-time-and-thinking-horizons/. (Last checked 12.12.2017)

Hoon, L., Vasa, R., Schneider, J.-G., & Mouzakis, K. (2012). A preliminary analysis of vocabulary in mobile app user reviews. In *Proceedings of the 24th australian computer-human interaction conference* (pp. 245–248).

Hunter, J. (2012). *Being Featured on the App Store.* http://blog.anylistapp.com/2012/08/app-store/. (Last checked 24.10.2017)

Iacob, C., & Harrison, R. (2013). Retrieving and analyzing mobile apps feature requests from online reviews. In *Mining software repositories (msr), 2013 10th ieee working conference on* (pp. 41–44).

Ipeirotis, P. G. (2010). Analyzing the Amazon Mechanical Turk Marketplace. *XRDS: Crossroads, The ACM Magazine for Students*, *17*(2), 16–21.

Ives, B., & Olson, M. H. (1984). User involvement and mis success: A review of research. *Management science*, *30*(5), 586–603.

Kaulio, M. A. (1998). Customer, consumer and user involvement in product development: A framework and a review of selected methods. *Total Quality Management*, *9*(1), 141–149.

Khalid, M., Asif, M., & Shehzaib, U. (2015). Towards improving the quality of mobile app reviews. *International Journal of Information Technology and Computer Science (IJITCS)*, *7*(10), 35.

Kujala, S. (2003). User involvement: a review of the benefits and challenges. *Behaviour & information technology*, *22*(1), 1–16.

Kujala, S. (2008). Effective user involvement in product development by improving the analysis of user needs. *Behaviour & Information Technology*, *27*(6), 457–473.

Law, E. L.-C., Roto, V., Hassenzahl, M., Vermeeren, A. P., & Kort, J. (2009). Understanding, scoping and defining user experience: a survey approach. In *Proceedings of the sigchi conference on human factors in computing systems* (pp. 719–728).

Lee, G., & Raghu, T. S. (2014). Determinants of mobile apps' success: evidence from the app store market. *Journal of Management Information Systems*, *31*(2), 133–170.

Maalej, W., & Hadeer, N. (2015). Bug report, feature request, or simply praise? on automatically classifying app reviews. In *Requirements Engineering Conference (RE), 2015 IEEE 23rd International.*

Maalej, W., Kurtanović, Z., Nabil, H., & Stanik, C. (2016). On the automatic classification of app reviews. *Requirements Engineering*, *21*(3), 311–331.

Magnusson, P. R. (2003). Benefits of involving users in service innovation. *European Journal of Innovation Management*, *6*(4), 228–238.

Martin, W., Sarro, F., Jia, Y., Zhang, Y., & Harman, M. (2017). A survey of app store analysis for software engineering. *IEEE transactions on software engineering*, *43*(9), 817–847.

Microsoft Azure. (2017). *Text Analytics API.* Retrieved from https://azure.microsoft.com/en-us/services/cognitive-services/text-analytics/. (Lastchecked 13.09.2017)

Mooi, E., & Sarstedt, M. (2011). *A Concise Guide to Market Research.* Springer Berlin Heidelberg.

Pagano, D., & Brügge, B. (2013). User involvement in software evolution practice: a case study. In *Software engineering (icse), 2013 35th international conference on* (pp. 953–962).

Pagano, D., & Maalej, W. (2013). User Feedback in the AppStore: An Empirical Study. *Requirements Engineering Conference (RE), 2013 21st IEEE International.*

Pang, B., Lee, L., et al. (2008). Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval*, *2*(1–2), 1–135.

Panichella, S., Di Sorbo, A., Guzman, E., Visaggio, C. A., Canfora, G., & Gall, H. C. (2015). How can i improve my app? classifying user reviews for software maintenance and evolution. In *Software maintenance and evolution (icsme), 2015 ieee international conference on* (pp. 281–290).

Prahalad, C. K., & Ramaswamy, V. (2013). *The future of competition: Co-creating unique value with customers.* Harvard Business Press.

Rouse, M. (2017). *MD5.* http://searchsecurity.techtarget.com/definition/MD5. TechTarget. (Lastchecked 07.04.2017)

Schenck, Stephen. (2013). *Why I don't update apps.* http://pocketnow.com/2013/09/25/dont-update-apps. (Last checked 11.12.2017)

Schlossberg, M. (2015). *Sorry, Oprah — your magic touch can't save Weight Watchers.* Retrieved from http://www.businessinsider.com/oprah-cant-save-weight-watchers-2015-10?IR=T. (Lastchecked 23.10.2017)

Seyff, N., Ollmann, G., & Bortenschlager, M. (2014). Appecho: A user-driven, in situ feedback approach for mobile platforms and applications. In *Proceedings of the 1st international conference on mobile software engineering and systems* (pp. 99–108).

Skibsted, Jens M. and Hansen, Rasmus B. (2014). *User-Led Innovation Can't Create Breakthroughs; Just Ask Apple and Ikea.* https://www.fastcodesign.com/1663220/user-led-innovation-cant-create-breakthroughs-just-ask-apple-and-ikea. (Last checked 11.12.2017)

Statista. (2017). *Number of apps available in leading app stores as of March 2017.* https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/. (Lastchecked 17.05.2017)

Statistics Solutions. (2013). *Homoscedasticity.* http://www.statisticssolutions.com/homoscedasticity/. (Last checked 28.10.2017)

Statistics Solutions. (2017). *Scatter plot: An Assumption of Regression Analysis.* http://www.statisticssolutions.com/scatterplot-an-assumption-of-regression-analysis/. (Last checked 28.10.2017)

Taş, Y. (2017). *How To Promote Health And Fitness Apps.* Retrieved from https://appsamurai.com/how-to-promote-health-and-fitness-apps/. (Lastchecked 23.10.2017)

The Jargon File. (n.d.). *bug.* Retrieved from http://catb.org/jargon/html/B/bug.html. (Lastchecked 28.05.2017)

Turk, A. M. (2017a). *FAQ Overview.* https://www.mturk.com/mturk/help?helpPage=overview. (Last checked 13.09.2017)

Turk, A. M. (2017b). *Mechanical Turk is a marketplace for work.* https://www.mturk.com/mturk/welcome. (Last checked 13.09.2017)

Vaniea, K., & Rashidi, Y. (2016). Tales of software updates: The process of updating software. In *Proceedings of the 2016 chi conference on human factors in computing systems* (pp. 3215–3226).

Vu, P. M., Nguyen, T. T., Pham, H. V., & Nguyen, T. T. (2015). Mining user opinions in mobile app reviews: A keyword-based approach (t). In *Automated software engineering (ase), 2015 30th ieee/acm international conference on* (pp. 749–759).

Walz, A. (2015). *Deconstructing the App Store Rankings Formula with a Little Mad Science.* https://moz.com/blog/app-store-rankings-formula-deconstructed-in-5-mad-science-experiments. (Last checked 27.09.2017)

Wiggins, N. (2015). *The difference between a Bug, Error and Feature.* https://www.webigence.com/blog/the-difference-between-a-bug-error-and-feature. (Last checked 28.06.2017)

Wodtke, Christina. (2013). *Users don't hate change. They hate you.* https://medium.com/@cwodtke/users-dont-hate-change-they-hate-you-461772fbcac7. (Last checked 11.12.2017)

Appendix

A1 Discussion Usefulness MT

Most of the time, it is a trade-off between advantages and disadvantages when it comes to online outsourcing and the quality of workers. Most MT workers are from the United States (46.80%) and India (34%), and approximately 80% of Indian and 60% United States workers have a bachelor's degree or higher (Ipeirotis, 2010). Therefore, it is assumed, from the education level alone, that the quality of workers is satisfactory. Another concern was that workers are dependent on the income gained from MT, which could lead to the motive of rushing through the HITs to make as much money as possible, with quality loss as a result. However, Ipeirotis' study revealed that most workers (50 - 69%) view it as a 'fruitful way to spend time and earn some money', and 63% regard it as a source of secondary income only. Hence, it was assumed that workers are not heavily dependent on the MT income and do not rush through the tasks (Ipeirotis, 2010).
The payment on MT is per HIT, and one HIT is one review. The requester can choose the amount she or he wants to pay, and the appropriate payment is dependent on the difficulty and complexity of the work, as well as the time needed to complete one HIT. The lowest possible payment is \$0.02/HIT. Hence, it was assumed that, with an appropriate payment, the quality of the results should not suffer.

A2 MT First Round HTML Form

The following figures show the interface an Amazon Mechanical Turk worker saw during the first classification round, including the briefing, and *generic* and *specific* reviews.

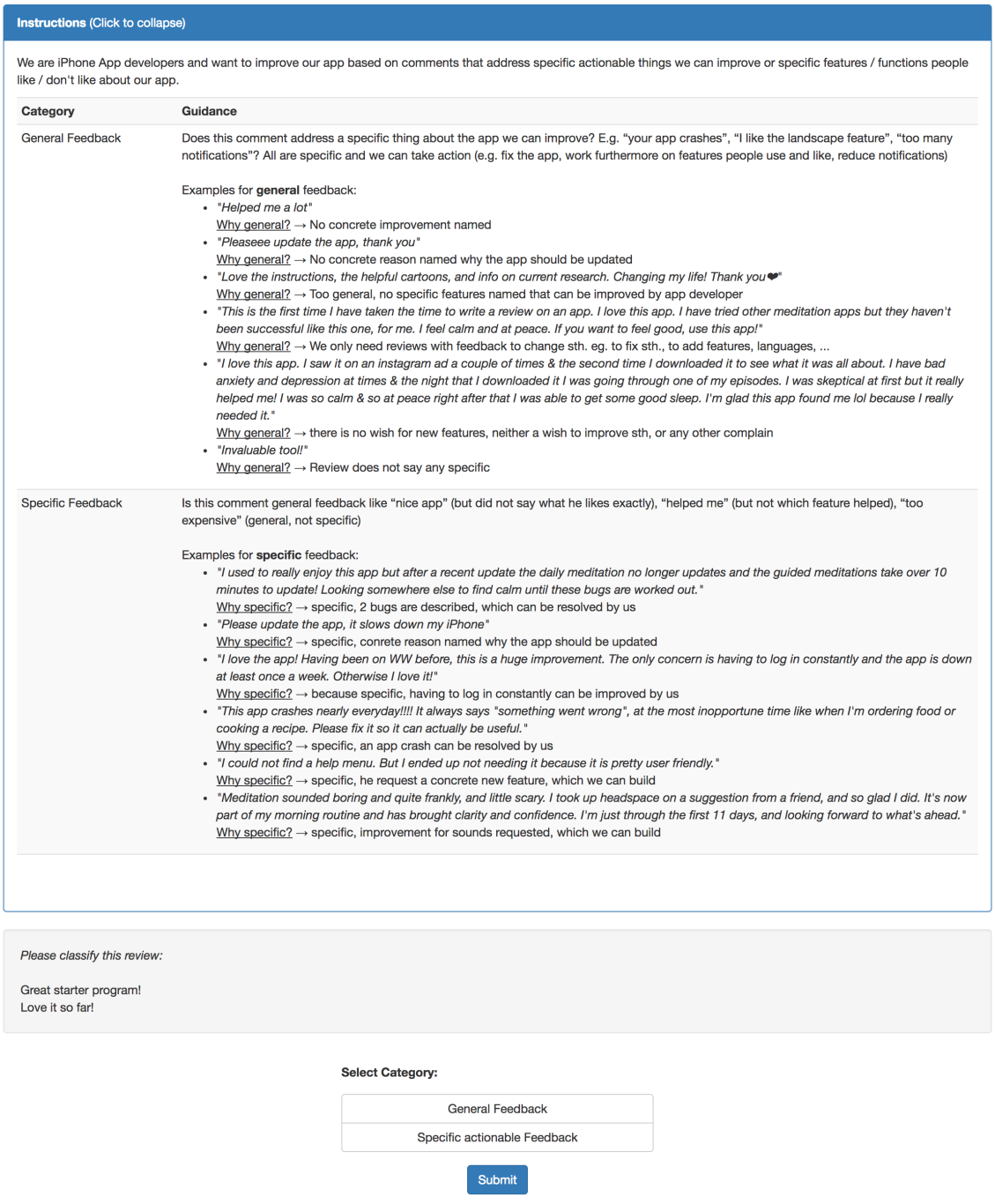


Figure 17: Interface with a *generic* review

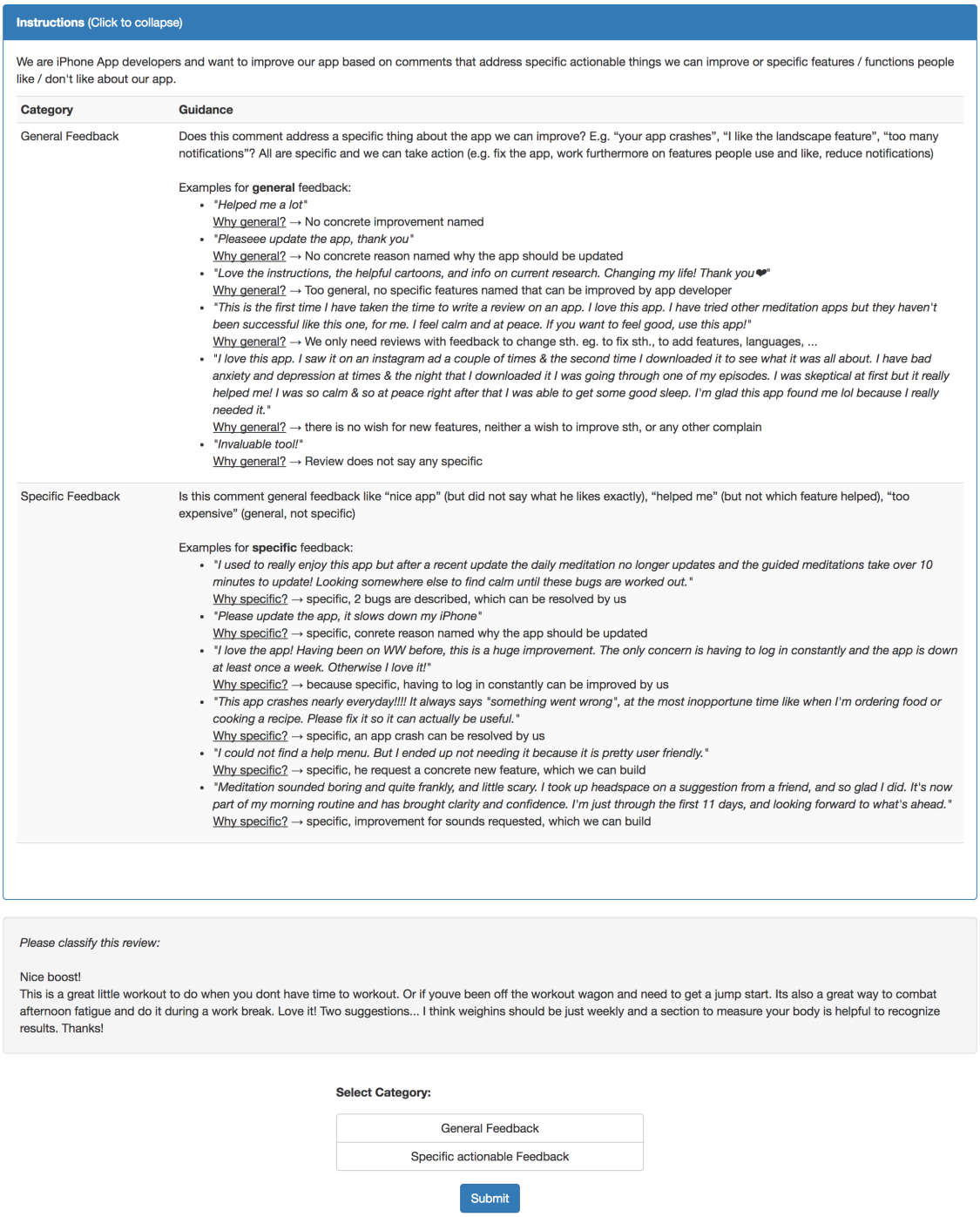


Figure 18: Interface with a *specific* review

A3 Overview Analysed Applications

Rank	App Name	Revenue	Versions
1	Sweat with Kayla	$8,817,452	17
2	MyFitnessPall	$6,812,815	33
3	Running for Weight Loss (free & paid version)	$5,584,888	25
4	Headspace	$5,315,932	17
5	Weight Watchers Mobile	$3,294,155	25
6	Lose It	$2,766,437	32
7	Calm	$2,443,813	19
8	Aaptiv	$1,967,906	24
9	Sleep Cycle alarm clock	$1,250,727	8
10	Strava Running & Cycling	$1,151,933	26
11	Runtastic Results Training App	$1,137,564	19
12	Lifesum	$1,099,371	30
13	8fit	$965,897	18
14	FIT Radio Workout Music	$818,777	9
15	Run with Map My Run (free & paid version)	$798,981	22
16	Walking for Weight Loss (free & paid version)	$763,167	4
17	Beachbody On Demand	$757,196	10
18	Full Fitness	$744,031	2
19	Runkeeper	$741,101	28
20	FitStar: Tony Gonzalez	$605,972	7
21	DailyBurn	$559,385	15
22	Freeletics	$557,363	29
23	Pacer Pedometer	$553,412	9
24	Fitness Buddy (free & paid version)	$529,101	7/4
25	Asana Rebel	$518,828	30
26	Workout: Gym personal trainer & workout tracker	$466,395	4
27	Ref Guide for Essential Oils	$380,545	5
28	Peloton Cycle: Live Fitness Indoor Cycling Classes	$345,144	14
29	Fitplan	$340,202	16
30	Happify	$330,928	16
31	iTrackBites	$330,876	5
32	Map My Ride (free & paid version)	$307,286	19/21
33	Yoga Studio	$297,895	4
34	Relax Melodies: Sleep & Yoga (free & paid version)	$290,266	1

Rank	App Name	Revenue	Versions
35	Sworkit	$289,057	14
36	Instant Heart Rate (free & paid version)	$286,366	6/4
37	LifeCycle (free & paid version)	$278,635	2/1
38	Runtastic Running & Fitness (free & paid version)	$272,453	16
39	Fitonomy - Fitness Challenge	$267,769	9
40	My Macros+	$264,657	5
41	Fooducate	$260,431	7
42	RockMyRun	$259353	14
		$57.365.700	230

Table 6: Analysed apps and their rankings, revenues, and amount of version updates

A4 Different Version Update Descriptions

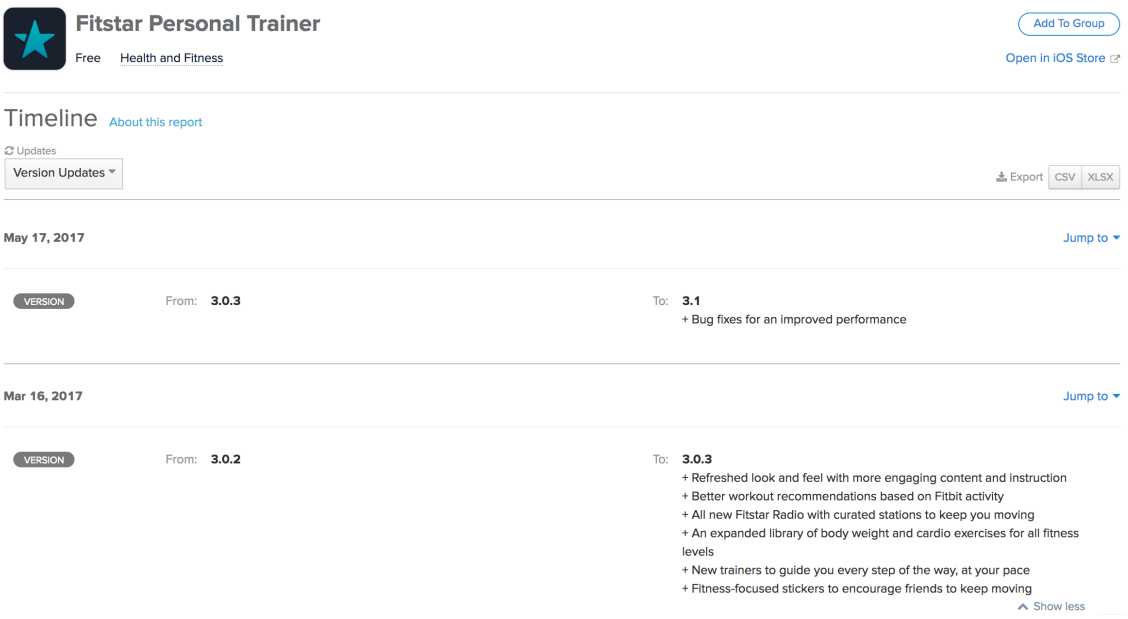


Figure 19: Different version update types
(Source: App Annie[4])

The figure shows two typical styles of updated descriptions. The first one (from 17.05.2017) only stated that some 'technical corrections' were undertaken. Another example of such a version update is when developers mention that they 'fixed bugs'. It is assumed that these are smaller updates, wherein they only change minor problems. The other version type (from 16.03.2017) is when a lot of things are fixed, and / or a lot of new features are added. It is not unusual that a 'big' update comes together with a new numerical version number, e.g. from version 2.3.1 to 3.0.

A5 Data Preparation

For step one and step two from the discussed framework in Figure 2, Chapter 4, the data needed to be transformed for the upload on MT. The several steps and code from both steps are explained below.

A5.1 Data Preparation for Useful Review Extraction

The input for Amazon Mechanical Turk were the downloaded .xlsx review files for each app from App Annie. However, the App Annie files were disordered, and it was impossible to upload the same file without adjustments. One big issue was the non-ASCII characters (e.g. smiles, Chinese characters) which are used in the columns User, Title, and Review, and which are not allowed on the Amazon Mechanical Turk upload. Nine steps were carried out with the DataScienceStudio software to clean the data for the MT upload.

1. Remove columns matching col_([0-9])+
2. Create column User if (isBlank(trim(User)==true),"no_name",User)
3. Create column md5_original_user with formula md5(User)
4. Replace in columns Review, Title, User values matching [^a-zA-Z0-9\s \!\,\; \. \?] with “no value”
5. Replace in columns Review, Title, User values matching [\r \n?| \n][5] with a blank space
6. Create column keep_entry if(and(Title==“",Review==“"), “", “keep")
7. Remove rows with empty values in column keep_entry
8. Remove column keep_entry
9. Create column User with formula if (isBlank(trim(User)),md5_original_user,User)

Each step is described in more detail below.
(1) For some reason, App Annie added additional empty columns, which needed to be removed.
(2) It is not possible to write a review in the Apple Store without a user name; therefore, the User field cannot be empty. However, the User column had, in fact,

[5][\r \n?|\n] are linebreaks

empty fields, because users used blank spaces. Therefore, the User column got the value no_value for supposedly empty fields (blank spaces) or his current value "User".
(3) Since it was only possible to upload CSV files with ASCII characters, all non-ASCII characters needed to be removed. Non-ASCII characters occurred in User, Title, and Review. But if a user name was made solely of non-ASCII characters and they were removed, the cell was empty and did not have a user name any longer. To fix this issue, a new column named md5_original_user was added. MD5 allowed one to generate the MD5 hash of any string. MD5 (Message-Digest algorithm 5) is an algorithm which computes a one-way cryptographic function with a 128-bit hash value. The input can be any string (e.g. User) and the output will be a fixed-length digest value (e.g. 8F9BFE9D1345237CB3B2B205864DA075) (Rouse, 2017). Chinese characters and emojis are strings; therefore, MD5 could be used for this problem.
(4),(5) In the next step, all non-ASCII characters and linebreaks were removed. Now several issues occurred:

1. empty Title or / and Review cell, if string was solely made off non-ASCII characters

2. empty User cell, respectively

(6) The first issue was fixed by adding a new column 'keep_entry', which means if in one row both the Title and Review were empty, the column 'keep_entry' became a value of zero. Otherwise, if only one cell was empty (either Title or Review), the row got the value 'keep'.
(7) Next, all rows with a value of zero in the 'keep_entry' column were deleted, since they did not have any information for analysis purposes.
(8) After the exclusion of rows with no value in the Review cell and Title cell, the column 'keep_entry' was deleted again, because it had served its purpose.
(9) The second issue (empty User cell) was fixed with the md5_original_user column. If a User cell was empty, this cell got the md5_original_user column value; otherwise, the cell kept the original User name. The md5_original_user had served its purpose too, but the column was kept, since it was not clear if it might have another purpose later on (e.g. for user comparison or matching purposes).
After conducting these steps, the file was ready for upload on Amazon Mechanical Turk without any disturbances.

A5.2 Data Preparation for Review Categorisation

Some rows and columns of the data obtained from the review extraction on Amazon Mechanical Turk needed to be cleaned. The downloaded CSV file contained 3,989 rows where the 'AssignmentStatus' was rejected. There were the HITs which were rejected because of a wrong classification.
Next, 33,761 rows where the 'Answer.sentiment' where *generic* were deleted, since only *specific* reviews were of interest. Lastly, all unnecessary columns added by Amazon Mechanical Turk were removed (e.g. 'AssigmentDurationInSeconds', 'AutpApprovalDelayinSeconds', 'RejectionTime').
Finally, this file and the updated version file were merged, where each row contained the following information (columns). Platform, Country, Review Date, App ID, App Name, Publisher ID, Publisher Name, User, md5_original_user, Version, Rating, Title, Review, Update Date, Previous Value, New Value, Feature_1, Feature_2, Feature_3, Feature_4, Feature_5, Feature_6, Feature_7, Feature_8, Feature_9, Feature_10, Feature_1_category, Feature_2_category, Feature_3_category, Feature_4_category, Feature_5_category, Feature_6_category, Feature_7_category, Feature_8_category, Feature_9_category, and Feature_10_category. The Feature_i and Feature_i_categories were only filled out if they had a value different from zero.
The merged file consisted of all reviews which were classified as *specific* and of all features from their updates and their category belonging.

A6 Detailed Steps Extraction Independent Variables

(1) First of all, the files from Amazon Mechanical Turk were merged. There were two files, because the classification was done in two rounds, whereby the first round had fewer records[6].
(2) As described earlier, Amazon Mechanical Turk adds additional columns with information about the batch as for example 'AssigmentDurationInSeconds'. These columns were deleted, and some other columns renamed.
(3) The next step was to enumerate the matches found between the user reviews and feature updates. This was done with the following Python codes.

[6]The first round was done with fewer records to be able to control the results and to check whether something was not working during the classification as it should. After the first round was working well, all remaining records were uploaded with the second round.

```
def process(row):
    count_matches = 0
    for key in row.keys(): # row.keys()
        if "Bug_" in key and "_Linked" in key:
            if isinstance(row[key], basestring) and len(row[key]) > 0:
    count_matches = count_matches + 1
    return count_matches
```

```
def process(row):
    count_matches = 0
    for key in row.keys():
        if "Feature_Request_" in key and "_Linked" in key:
            if isinstance(row[key], basestring) and len(row[key]) > 0:
                count_matches = count_matches + 1

    return count_matches
```

```
def process(row):
    count_matches = 0
    for key in row.keys():
        if "Too_Many_Ads_" in key and "_Linked" in key:
            if isinstance(row[key], basestring) and len(row[key]) > 0:
                count_matches = count_matches + 1

    return count_matches
```

Note, from the user review matches, two different independent variables were derived: *Customer-Led Product Innovation* and *Customer-Led Product Improvement*. *Customer-Led Product Innovation* counts the Feature Request matches, because newly launched features innovate the application. In contrast, *Customer-Led Product Improvement* emerges from fixed bugs and removed advertisements. There are matches with other categories, for example *User Experience* or *Pricing*, but these categories neither innovate nor improve the application.

(4) Next, the dataset was grouped by the columns App ID and App Name, so that one row represented one app with the enumerated matches.

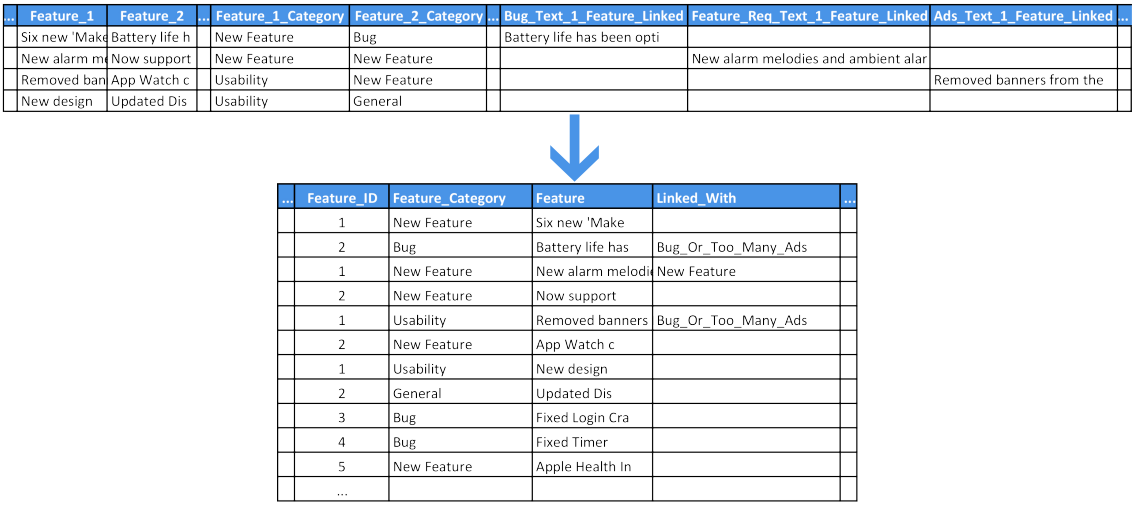
(5) To receive the Developer-Led variables, the differences of all reviews minus the match reviews were computed; hence, the *Bug, Feature Request* and *Too Many Advertisements* match. It is assumed that, whenever there is no match between a user review and a feature update, the improvement / innovation comes from the developer side.

(6) In the last step, the independent variables *Customer-Led Product Improvement* and *Developer-Led Product Improvement* were computed, by adding the *Bug* matches and *Too Many Advertisements* matches together, and the No *Bug* Matches and No *Too Many Advertisements* Matches, respectively.

A7 Detailed Steps Extraction Response Time Variables

Response time for Customer-Led Innovation and Customer-Led Improvement

Step (1) - (3) are the same as above (4) Transform *Feature_i* columns into rows to have one feature per row. Figure 20 shows the original and new dataset.



...	Feature_1	Feature_2	...	Feature_1_Category	Feature_2_Category	...	Bug_Text_1_Feature_Linked	Feature_Req_Text_1_Feature_Linked	Ads_Text_1_Feature_Linked	...
	Six new 'Make	Battery life h		New Feature	Bug		Battery life has been opti			
	New alarm m	Now support		New Feature	New Feature			New alarm melodies and ambient alar		
	Removed ban	App Watch c		Usability	New Feature				Removed banners from the	
	New design	Updated Dis		Usability	General					

...	Feature_ID	Feature_Category	Feature	Linked_With	...
	1	New Feature	Six new 'Make		
	2	Bug	Battery life has	Bug_Or_Too_Many_Ads	
	1	New Feature	New alarm melodi	New Feature	
	2	New Feature	Now support		
	1	Usability	Removed banners	Bug_Or_Too_Many_Ads	
	2	New Feature	App Watch c		
	1	Usability	New design		
	2	General	Updated Dis		
	3	Bug	Fixed Login Cra		
	4	Bug	Fixed Timer		
	5	New Feature	Apple Health In		
	...				

Figure 20: Transfrom columns to rows

In the original dataset, one row represented one review and the features from the updates were listed separately in columns. These columns were transferred into rows, so that each row represented one feature from the update, instead of one review. To do that, first the columns *Feature_1* - *Feature_10* were folded into rows, whereby the column for the fold name is *Feature_ID* and the column for the fold value is *Feature*. In the next step, the column *Feature_ID* was modified, so that values have ID as single number (e.g. 1,2,3), instead of 'Feature_1', 'Feature_2'. The column *Feature_Category* was created with the values from the old Feature category columns. Lastly, a new column Linked_With was created with the following Python code, to show if a feature update was linked with a review.

```
1 def process(row):
2     match = ""
3 
4     for key in row.keys(): # go through all column keys
5         if "_Linked" in key: # is the current column a linked one (
    decide by its name)?
6             cell_value = row[key] # get current column's value
7             feature = row['Feature'] # get the feature column's value
8 
9             if "Bug" in key and cell_value == feature: # is it a bug
```

```
    linked column and linked with the feature (so values are equal)?
10                  match = "Bug_or_Too_Many_Ads"
11              if "Too_Many_Ads" in key and cell_value == feature: # is it
     a too many ads linked column and linked with the feature (so
    values are equal)?
12                  match = "Bug_or_Too_Many_Ads"
13              if "Feature_Request" in key and cell_value == feature: # is
     it a feature request linked column and linked with the feature (so
     values are equal)?
14                  match = "Feature_Request"
15
16      #return ','.join(matches)
17      return match
```

The column *Linked_ With* can take three different values, namely "Feature Request" (*Customer Led Innovation*), "Bug Or Too Many Ads" (*Customer Led Improvement*), or an empty cell (no match). The new transformed dataset consists of 30,001 unique rows.

(5) In the next step, the dataset was aggregated by the minimum of the variable *Review_ Date*, and grouped by the variables *App_ Name, App_ ID, Linked_ With, Feature, Feature_ Category, and Update_ Date.*

(6) After that, with the variable *Timespan_ First_ Review_ To_ Update* the difference in days between *Review_ Date_ min* and *Update_ Date* were computed and all rows with an empty cell in *Linked_ With* deleted, so that only matched features were left. Figure 21 shows the dataset.

App_ID	App_Name	Update_Date	Feature	Linked_With	Feature_Categ	Review_Date_Min	Timespan_First_Review_To_Update	Count
1031660123	Beachbody On Demand – The	2017-02-22	Fixed the bug th	Bug_or_Too_Many_Ads	Bug	2017-01-04	-49	21
1049234587	Sweat: Kayla Itsines' Bikini Bod	2016-09-11	Apple Watch co	Bug_or_Too_Many_Ads	New Feature	2016-08-27	-15	7
1049234587	Sweat: Kayla Itsines' Bikini Bod	2016-09-11	Apple Watch fe	Bug_or_Too_Many_Ads	New Feature	2016-09-20	9	1
1067860796	Asana Rebel - Yoga Inspired Fi	2016-12-20	Workout Goal (	Bug_or_Too_Many_Ads	New Feature	2016-12-16	-4	1
291890420	Map My Run - GPS Running &	2017-02-01	New training da	Feature_Request	New Feature	2017-01-09	-23	1
292223170	Map My Ride - GPS Cycling & F	2017-01-31	Track your reco	Feature_Request	New Feature	2017-01-28	-3	2
297368629	Lose It! – Weight Loss Program	2017-01-07	Improved how	Bug_or_Too_Many_Ads	Bug	2017-01-28	21	1
297368629	Lose It! – Weight Loss Program	2017-01-07	We added the a	Bug_or_Too_Many_Ads	New Feature	2017-01-11	4	1
300235330	Runkeeper - Track Running wi	2017-01-24	We updated wo	Bug_or_Too_Many_Ads	Bug	2016-12-10	-45	2
320606217	Sleep Cycle alarm clock	2016-06-11	New alarm mel	Bug_or_Too_Many_Ads	New Feature	2016-04-26	-46	8
331308914	Weight Watchers	2016-07-15	Fixed some glit	Bug_or_Too_Many_Ads	General	2016-05-20	-56	93
331529422	Relax Melodies P: sleep sound	2016-12-08	New design	Bug_or_Too_Many_Ads	Usability	2016-12-12	4	7
341232718	Calorie Counter & Diet Tracke	2016-10-29	Create a custon	Feature_Request	New Feature	2016-09-06	-53	8
409625068	Instant Heart Rate: Heart Rate	2016-11-22	Introducing Hea	Feature_Request	New Feature	2016-10-18	-35	5
...	...	...	...	...	...	...	...	...

Figure 21: Difference in days from first review until it was fixed / launched

(7) Until now, for each application, the response time was computed for each feature; therefore, the average of days for all features was computed.
(8) In the last step, the dataset was split to compute the response time for *Customer-Led Innovation* on the one side and the response time for *Customer-Led Improvement* on the other. This was done with the variable *Linked_With*, where earlier was already distinguished between 'Feature_Request' and 'Bug_Or_Too_Many_Ads'.

A8 Reviews and Version Updates over time

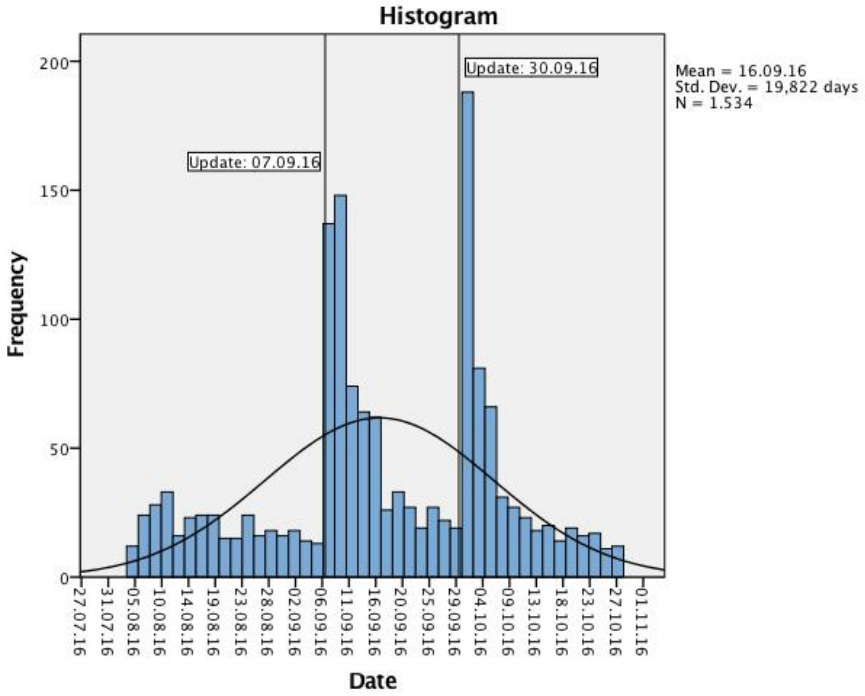


Figure 22: Review histogram Pacer with version updates

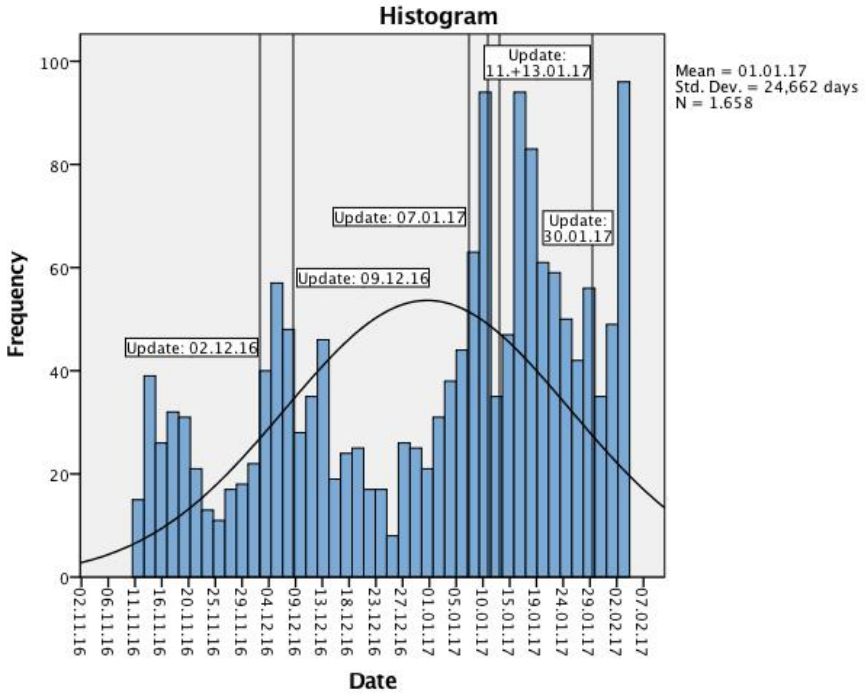


Figure 23: Review histogram Lose It with version updates

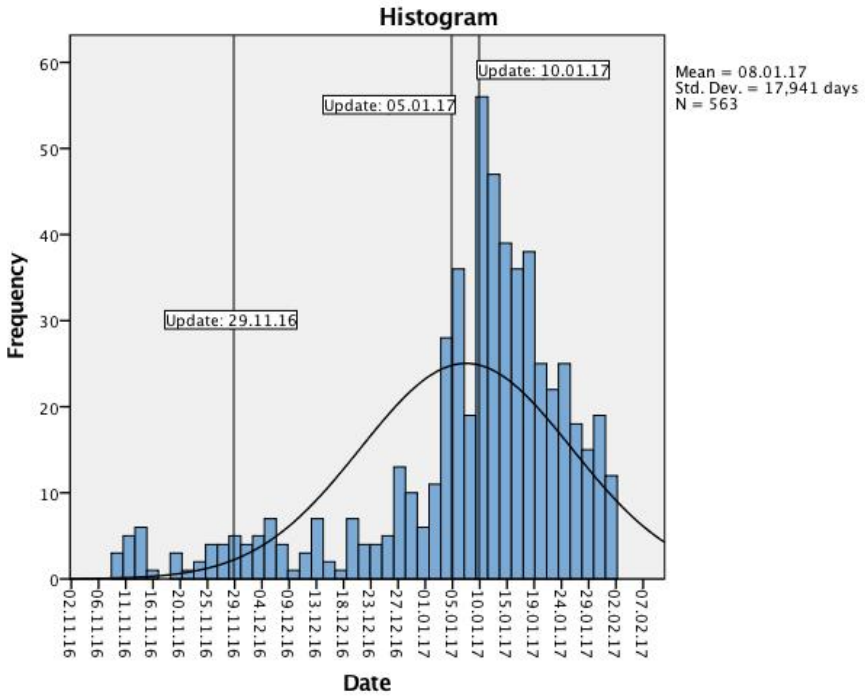


Figure 24: Review histogram FitStar with version updates

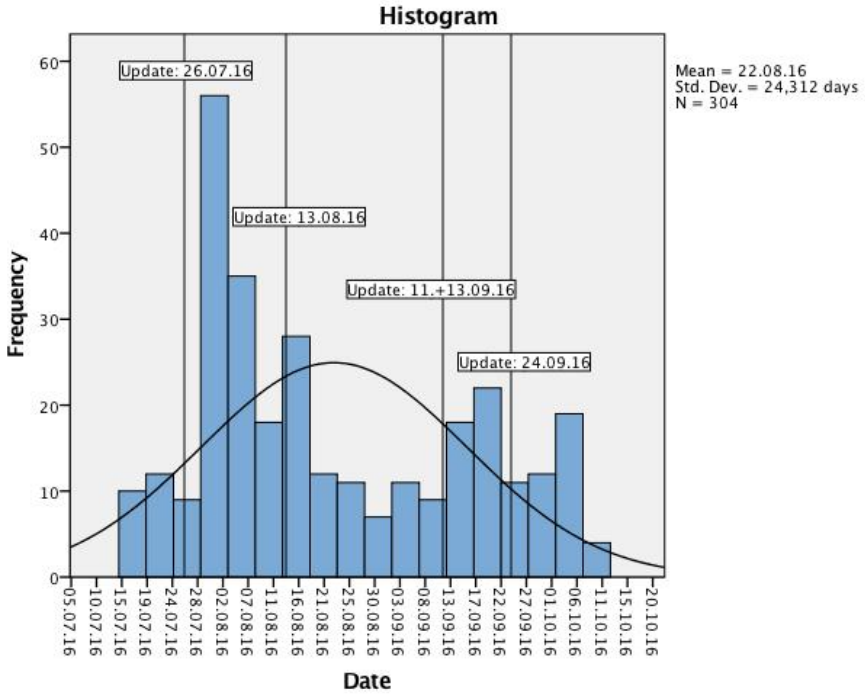


Figure 25: Review histogram Sweat with Kayla with version updates

A9 Amount of Version Updates Within One Year

Descriptive Statistics

	N	Minimum	Maximum	Mean	Std. Deviation
Amount Versions	42	1,0	33,0	14,286	9,2763
Valid N (listwise)	42				

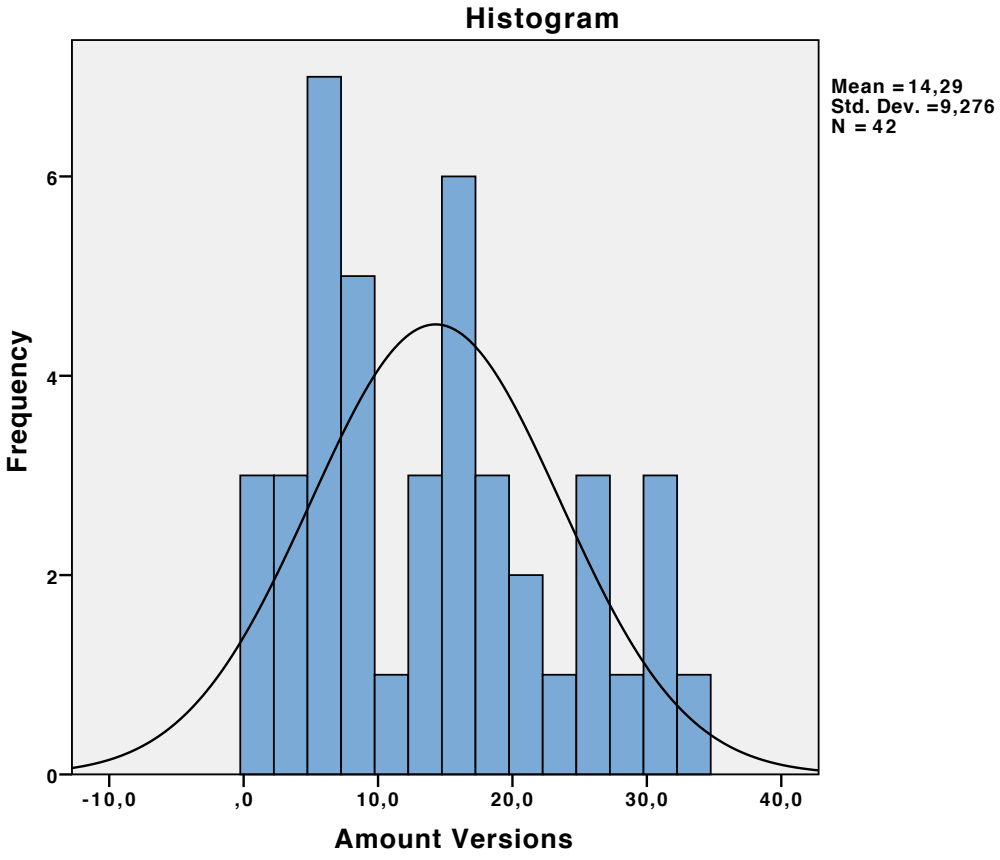
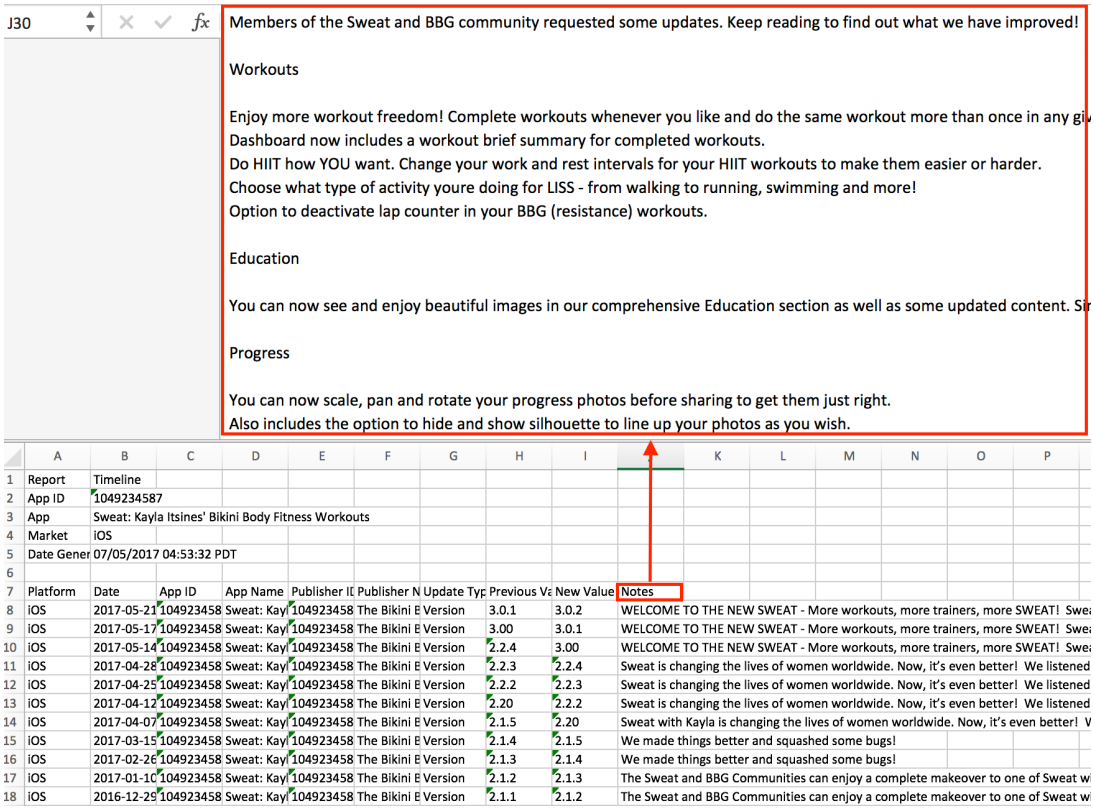


Figure 26: Amount of version updates of the 42 chosen apps within the time frame from 01.03.2016 to 28.02.2017

A10 Version Update File from App Annie



J30

Members of the Sweat and BBG community requested some updates. Keep reading to find out what we have improved!

Workouts

Enjoy more workout freedom! Complete workouts whenever you like and do the same workout more than once in any gi
Dashboard now includes a workout brief summary for completed workouts.
Do HIIT how YOU want. Change your work and rest intervals for your HIIT workouts to make them easier or harder.
Choose what type of activity youre doing for LISS - from walking to running, swimming and more!
Option to deactivate lap counter in your BBG (resistance) workouts.

Education

You can now see and enjoy beautiful images in our comprehensive Education section as well as some updated content. Si

Progress

You can now scale, pan and rotate your progress photos before sharing to get them just right.
Also includes the option to hide and show silhouette to line up your photos as you wish.

	A	B	C	D	E	F	G	H	I	K	L	M	N	O	P
1	Report	Timeline													
2	App ID	1049234587													
3	App	Sweat: Kayla Itsines' Bikini Body Fitness Workouts													
4	Market	iOS													
5	Date Gener	07/05/2017 04:53:32 PDT													
6															
7	Platform	Date	App ID	App Name	Publisher ID	Publisher N	Update Typ	Previous Va	New Value	Notes					
8	iOS	2017-05-21	104923458	Sweat: Kayl	104923458	The Bikini B	Version	3.0.1	3.0.2	WELCOME TO THE NEW SWEAT - More workouts, more trainers, more SWEAT! Swea					
9	iOS	2017-05-17	104923458	Sweat: Kayl	104923458	The Bikini B	Version	3.00	3.0.1	WELCOME TO THE NEW SWEAT - More workouts, more trainers, more SWEAT! Swea					
10	iOS	2017-05-14	104923458	Sweat: Kayl	104923458	The Bikini B	Version	2.2.4	3.00	WELCOME TO THE NEW SWEAT - More workouts, more trainers, more SWEAT! Swea					
11	iOS	2017-04-28	104923458	Sweat: Kayl	104923458	The Bikini B	Version	2.2.3	2.2.4	Sweat is changing the lives of women worldwide. Now, it's even better! We listened					
12	iOS	2017-04-25	104923458	Sweat: Kayl	104923458	The Bikini B	Version	2.2.2	2.2.3	Sweat is changing the lives of women worldwide. Now, it's even better! We listened					
13	iOS	2017-04-12	104923458	Sweat: Kayl	104923458	The Bikini B	Version	2.20	2.2.2	Sweat is changing the lives of women worldwide. Now, it's even better! We listened					
14	iOS	2017-04-07	104923458	Sweat: Kayl	104923458	The Bikini B	Version	2.1.5	2.20	Sweat with Kayla is changing the lives of women worldwide. Now, it's even better! V					
15	iOS	2017-03-15	104923458	Sweat: Kayl	104923458	The Bikini B	Version	2.1.4	2.1.5	We made things better and squashed some bugs!					
16	iOS	2017-02-26	104923458	Sweat: Kayl	104923458	The Bikini B	Version	2.1.3	2.1.4	We made things better and squashed some bugs!					
17	iOS	2017-01-10	104923458	Sweat: Kayl	104923458	The Bikini B	Version	2.1.2	2.1.3	The Sweat and BBG Communities can enjoy a complete makeover to one of Sweat w					
18	iOS	2016-12-29	104923458	Sweat: Kayl	104923458	The Bikini B	Version	2.1.1	2.1.2	The Sweat and BBG Communities can enjoy a complete makeover to one of Sweat w					

Figure 27: Downloaded Excel file from App Annie

A11 Crosstabs Specific and Generic Reviews by App

App Name * Answer.sentiment Crosstabulation

			Answer.sentiment		
			general	specific	Total
App Name	8fit - Workouts, meal plans and personal trainer	Count	945	101	1046
		% within App Name	90,3%	9,7%	100,0%
	Aaptiv: #1 Audio Fitness App	Count	321	131	452
		% within App Name	71,0%	29,0%	100,0%
	Asana Rebel - Yoga Inspired Fitness	Count	313	104	417
		% within App Name	75,1%	24,9%	100,0%
	Beachbody On Demand – The Best Fitness Workouts	Count	127	312	439
		% within App Name	28,9%	71,1%	100,0%
	Calm: Meditation to Relax, Focus & Sleep Better	Count	1110	186	1296
		% within App Name	85,6%	14,4%	100,0%
	Calorie Counter & Diet Tracker by MyFitnessPal	Count	1793	924	2717
		% within App Name	66,0%	34,0%	100,0%
	Daily Burn - Video Workouts	Count	16	13	29
		% within App Name	55,2%	44,8%	100,0%
	Fit Radio - Workout Music, Running, Cardio Coach	Count	36	10	46
		% within App Name	78,3%	21,7%	100,0%
	Fitness Buddy: Gym & Home Workout Exercise Trainer	Count	102	17	119
		% within App Name	85,7%	14,3%	100,0%
	Fitness Buddy+ Gym Workout Exercise, Home Trainer	Count	288	99	387
		% within App Name	74,4%	25,6%	100,0%
	Fitonomy - Fitness Challenge	Count	9	6	15
		% within App Name	60,0%	40,0%	100,0%
	Fitplan: Workout, Burn Fat & Train with Athletes	Count	6	8	14
		% within App Name	42,9%	57,1%	100,0%
	Fitstar Personal Trainer	Count	412	151	563
		% within App Name	73,2%	26,8%	100,0%
	Fooducate - Lose Weight, Eat Healthy,Get Motivated	Count	67	35	102
		% within App Name	65,7%	34,3%	100,0%
	Freeletics Bodyweight - Workouts and Training	Count	129	54	183
		% within App Name	70,5%	29,5%	100,0%

App Name * Answer.sentiment Crosstabulation

			Answer.sentiment		
			general	specific	Total
	Full Fitness : Exercise Workout Trainer	Count	440	308	748
		% within App Name	58,8%	41,2%	100,0%
	Happify - Activities & Games for Stress & Anxiety	Count	44	19	63
		% within App Name	69,8%	30,2%	100,0%
	Headspace: Guided Meditation and Mindfulness	Count	764	127	891
		% within App Name	85,7%	14,3%	100,0%
	Instant Heart Rate: Heart Rate & Pulse Monitor	Count	719	176	895
		% within App Name	80,3%	19,7%	100,0%
	Instant Heart Rate+: Heart Rate & Pulse Monitor	Count	292	38	330
		% within App Name	88,5%	11,5%	100,0%
	iTrackBites Plus - Smart Weight Loss Tracker & Point	Count	132	68	200
		% within App Name	66,0%	34,0%	100,0%
	Life Period Tracker, Health, Calendar, Ovulation	Count	855	89	944
		% within App Name	90,6%	9,4%	100,0%
	Life Pro: Period Tracker, Period & Ovulation App	Count	5	3	8
		% within App Name	62,5%	37,5%	100,0%
	Lifesum – Inspiring healthy lifestyle app	Count	63	62	125
		% within App Name	50,4%	49,6%	100,0%
	Lose It! – Weight Loss Program and Calorie Counter	Count	1324	334	1658
		% within App Name	79,9%	20,1%	100,0%
	Map My Ride - GPS Cycling & Route Tracker	Count	520	128	648
		% within App Name	80,2%	19,8%	100,0%
	Map My Ride+ - GPS Cycling & Route Tracker	Count	100	31	131
		% within App Name	76,3%	23,7%	100,0%
	Map My Run - GPS Running & Workout Tracker	Count	2614	324	2938
		% within App Name	89,0%	11,0%	100,0%
	Map My Run+ - GPS Running & Workout Tracker	Count	128	32	160
		% within App Name	80,0%	20,0%	100,0%

App Name * Answer.sentiment Crosstabulation

			Answer.sentiment		
			general	specific	Total
	My Macros+ Diet, Weight and Calorie Tracker	Count	152	82	234
		% within App Name	65,0%	35,0%	100,0%
	Pacer - Pedometer plus Weight Loss and BMI Tracker	Count	1358	174	1532
		% within App Name	88,6%	11,4%	100,0%
	Peloton Cycle: Live Fitness Indoor Cycling Classes	Count	74	41	115
		% within App Name	64,3%	35,7%	100,0%
	Ref Guide for Essential Oils	Count	7	3	10
		% within App Name	70,0%	30,0%	100,0%
	Relax Melodies P: sleep sounds, white noise & fan	Count	1108	219	1327
		% within App Name	83,5%	16,5%	100,0%
	Relax Melodies: Sleep Sounds, White Noise & Fan	Count	10383	882	11265
		% within App Name	92,2%	7,8%	100,0%
	RockMyRun - Workout Music & Running Tracker	Count	35	23	58
		% within App Name	60,3%	39,7%	100,0%
	Runkeeper - Track Running with GPS	Count	961	296	1257
		% within App Name	76,5%	23,5%	100,0%
	RUNNING for weight loss PRO: workout & meal plans	Count	44	15	59
		% within App Name	74,6%	25,4%	100,0%
	RUNNING for weight loss: workout & meal plans	Count	376	121	497
		% within App Name	75,7%	24,3%	100,0%
	Runtastic PRO Running, Jogging and Fitness Tracker	Count	244	169	413
		% within App Name	59,1%	40,9%	100,0%
	Runtastic Results: Workout & Strength Training	Count	152	44	196
		% within App Name	77,6%	22,4%	100,0%
	Runtastic Running, Jogging and Walking Tracker	Count	152	41	193
		% within App Name	78,8%	21,2%	100,0%
	Sleep Cycle alarm clock	Count	2366	426	2792
		% within App Name	84,7%	15,3%	100,0%

App Name * Answer.sentiment Crosstabulation

			Answer.sentiment		
			general	specific	Total
	Strava Running and Cycling GPS	Count	402	116	518
		% within App Name	77,6%	22,4%	100,0%
	Sweat: Kayla Itsines' Bikini Body Fitness Workouts	Count	113	191	304
		% within App Name	37,2%	62,8%	100,0%
	Sworkit - Custom Workouts for Exercise & Fitness	Count	359	89	448
		% within App Name	80,1%	19,9%	100,0%
	Walking for Weight Loss: training plans, GPS, tips	Count	58	36	94
		% within App Name	61,7%	38,3%	100,0%
	Weight Watchers	Count	552	560	1112
		% within App Name	49,6%	50,4%	100,0%
	Workout: Gym personal trainer & workout tracker	Count	650	106	756
		% within App Name	86,0%	14,0%	100,0%
	Yoga Studio	Count	541	130	671
		% within App Name	80,6%	19,4%	100,0%
Total		Count	33761	7654	41415
		% within App Name	81,5%	18,5%	100,0%

Figure 28: Distribution *generic* and *specific* reviews by analyzed app

A12 Requirements and Instructions HTML Form

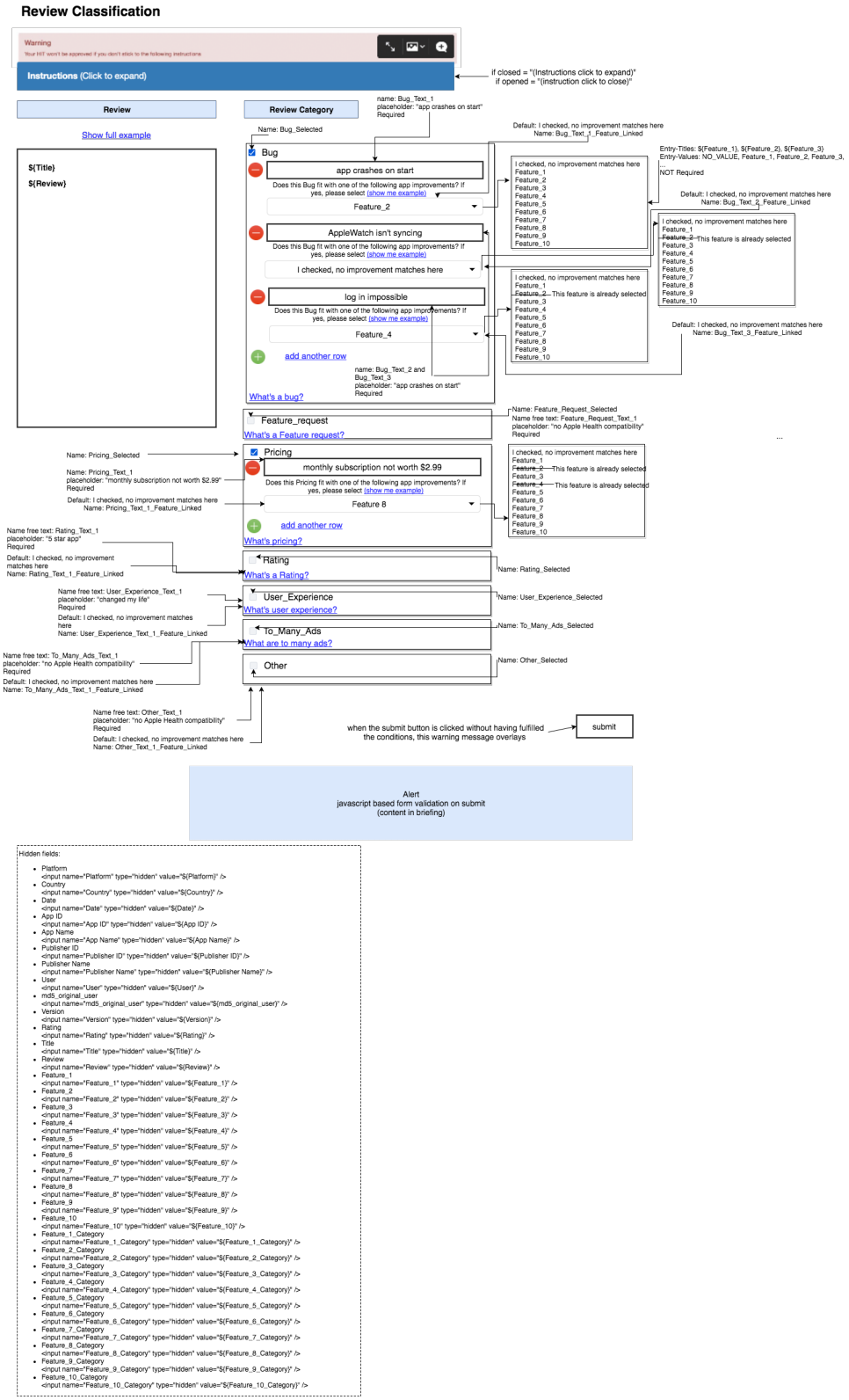


Figure 29: Briefing HTML form for the second classification round via MT (Source: own)

A13 Rating Distribution Review Categories

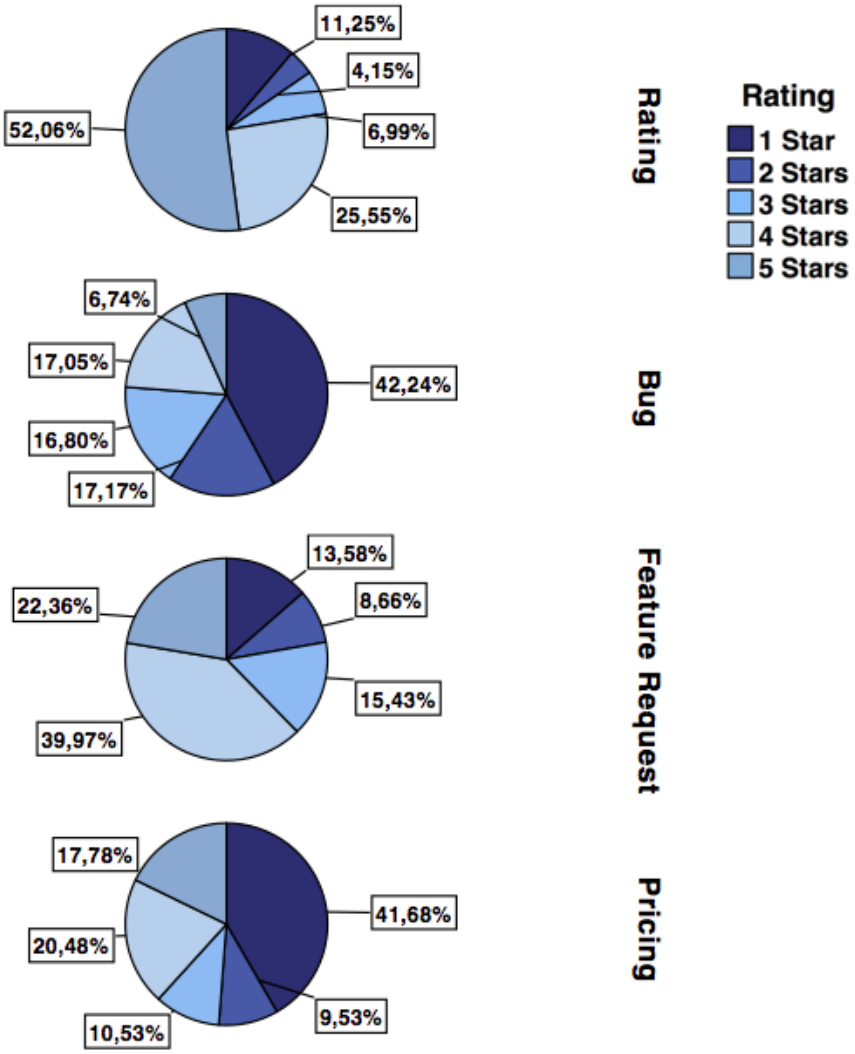


Figure 30: Rating Distribution Categories 1

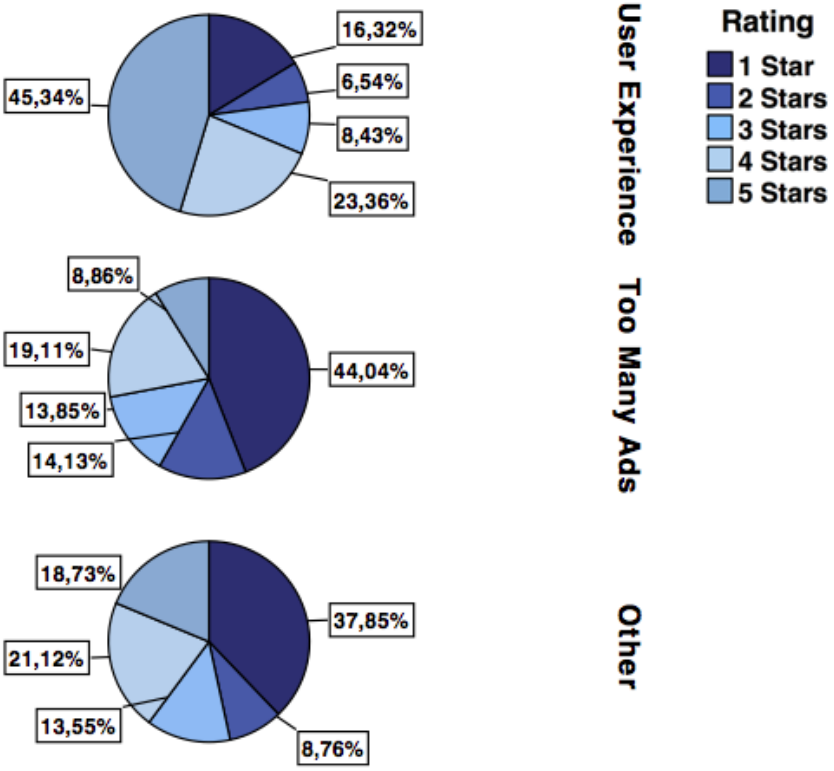


Figure 31: Rating Distribution Categories 2

A14 Scatterplots

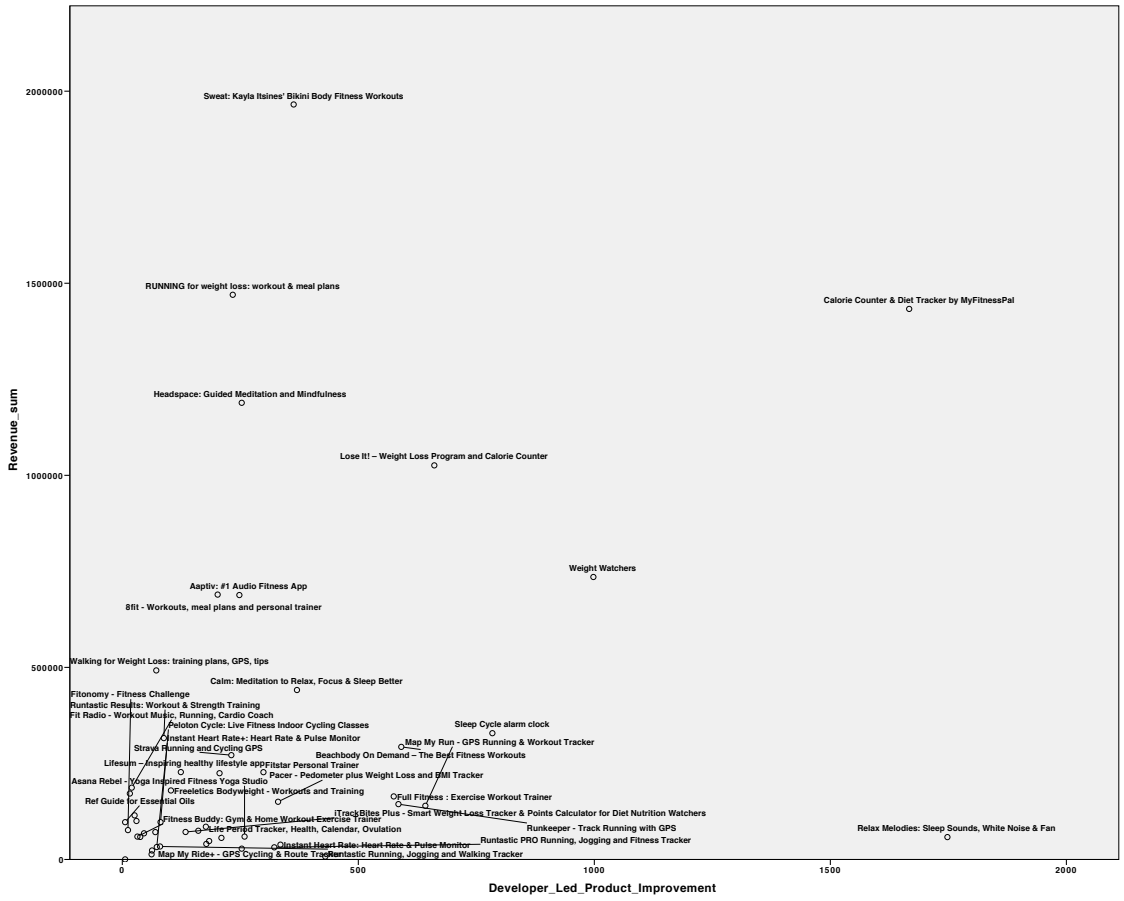


Figure 32: Scatterplot *Revenue* against *Developer Led Product Improvement*

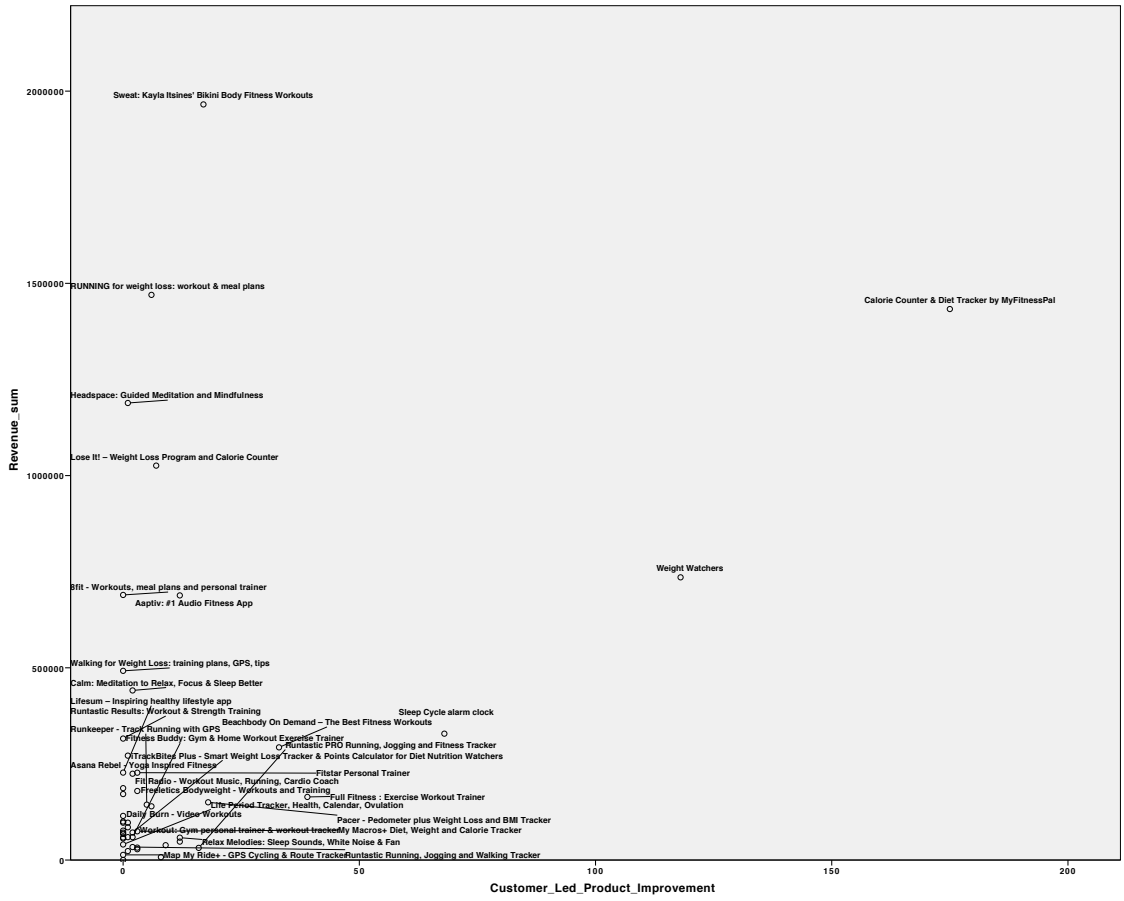


Figure 33: Scatterplot *Revenue* against *Customer Led Product Improvement*

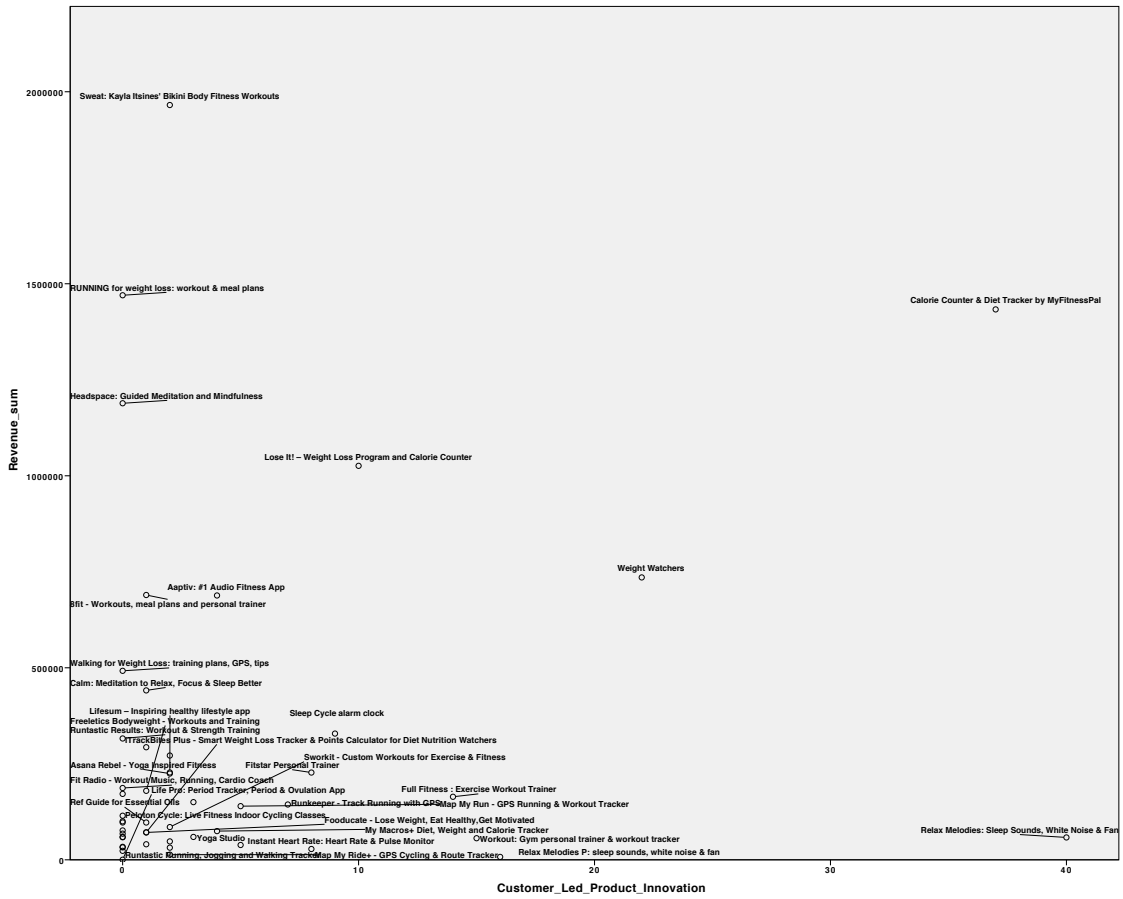


Figure 34: Scatterplot *Revenue* against *Customer Led Product Innovation*

A15 Apps on Instagram

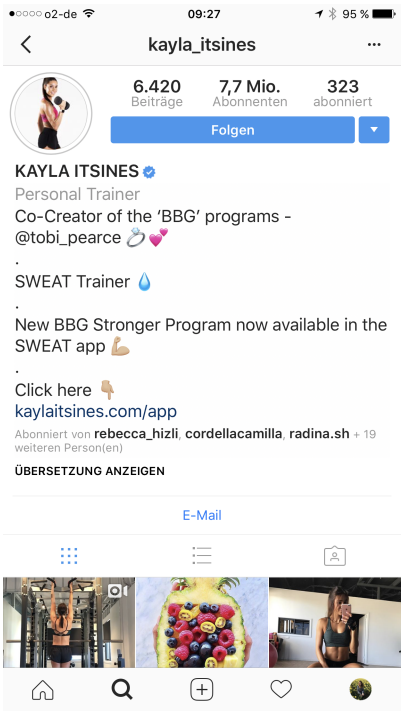


Figure 35: Instagram account of Kalya Itsines

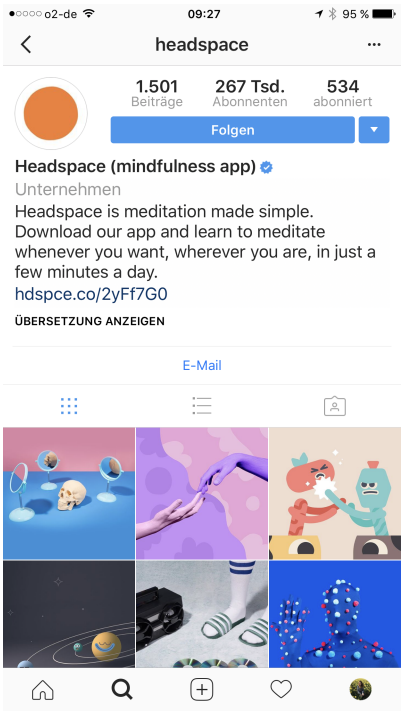


Figure 36: Instagram account of Headspace

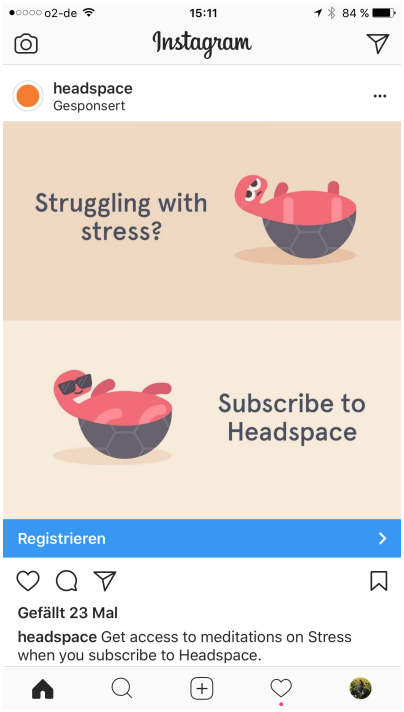


Figure 37: Instagram ad of Headspace

Figure 38: Instagram account of Weight Watchers

Figure 39: 8fit Instagram cooperation with Amelia Liana

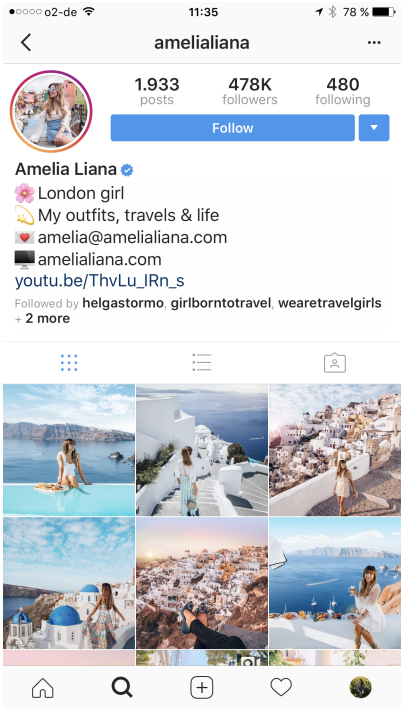


Figure 40: Instagram account of Amelia Liana

A16 Correlation Matrix

Correlations

		Customer_Led_Product_Innovation	Customer_Led_Product_Improvement	Developer_Led_Product_Innovation	Developer_Led_Product_Improvement	Downloads_sum	Rating_avg	Revenue_sum	Group_by_Ranking_Downloads	Group_by_Ranking_Revenue	Responsiveness_Developer_Led_Product_Improvement	Responsiveness_Developer_Led_Product_Innovation	Responsiveness_Customer_Led_Product_Improvement	Responsiveness_Customer_Led_Product_Innovation	Published_Apps_by_Publisher
Customer_Led_Product_Innovation	Pearson Correlation	1	,665**	,902**	,910**	,370**	,035	,193	-,194	-,077	,363**	,017	,399**	,442**	-,040
	Sig. (2-tailed)		,000	,000	,000	,008	,811	,180	,178	,595	,010	,906	,004	,001	,785
	N	50	50	50	50	50	50	50	50	50	50	50	50	50	50
Customer_Led_Product_Improvement	Pearson Correlation	,665**	1	,753**	,718**	,466**	-,244	,420**	-,190	-,128	,040	-,020	,540**	,424**	-,142
	Sig. (2-tailed)	,000		,000	,000	,001	,087	,002	,187	,374	,781	,888	,000	,002	,327
	N	50	50	50	50	50	50	50	50	50	50	50	50	50	50
Developer_Led_Product_Innovation	Pearson Correlation	,902**	,753**	1	,998**	,510**	-,038	,355*	-,303*	-,173	,247	-,056	,572**	,385**	-,118
	Sig. (2-tailed)	,000	,000		,000	,000	,795	,011	,033	,230	,083	,699	,000	,006	,416
	N	50	50	50	50	50	50	50	50	50	50	50	50	50	50
Developer_Led_Product_Improvement	Pearson Correlation	,910**	,718**	,998**	1	,499**	-,018	,337*	-,302*	-,169	,264	-,055	,557**	,378**	-,111
	Sig. (2-tailed)	,000	,000	,000		,000	,902	,017	,033	,239	,063	,705	,000	,007	,445
	N	50	50	50	50	50	50	50	50	50	50	50	50	50	50
Downloads_sum	Pearson Correlation	,370**	,466**	,510**	,499**	1	-,093	,743**	-,385**	-,296*	-,175	-,108	,148	,106	-,234
	Sig. (2-tailed)	,008	,001	,000	,000		,520	,000	,006	,037	,224	,457	,306	,463	,102
	N	50	50	50	50	50	50	50	50	50	50	50	50	50	50
Rating_avg	Pearson Correlation	,035	-,244	-,038	-,018	-,093	1	-,335*	,033	,248	-,003	-,102	-,240	,041	,166
	Sig. (2-tailed)	,811	,087	,795	,902	,520		,017	,820	,082	,984	,483	,094	,776	,251
	N	50	50	50	50	50	50	50	50	50	50	50	50	50	50
Revenue_sum	Pearson Correlation	,193	,420**	,355*	,337*	,743**	-,335*	1	-,293*	-,259	-,118	-,161	,140	,026	-,309*
	Sig. (2-tailed)	,180	,002	,011	,017	,000	,017		,039	,070	,413	,264	,334	,856	,029
	N	50	50	50	50	50	50	50	50	50	50	50	50	50	50
Group_by_Ranking_Downloads	Pearson Correlation	-,194	-,190	-,303*	-,302*	-,385**	,033	-,293*	1	,808**	,054	-,102	-,281*	-,255	-,163
	Sig. (2-tailed)	,178	,187	,033	,033	,006	,820	,039		,000	,712	,482	,048	,074	,257
	N	50	50	50	50	50	50	50	50	50	50	50	50	50	50
Group_by_Ranking_Revenue	Pearson Correlation	-,077	-,128	-,173	-,169	-,296*	,248	-,259	,808**	1	,177	,028	-,166	-,045	,045
	Sig. (2-tailed)	,595	,374	,230	,239	,037	,082	,070	,000		,218	,848	,250	,758	,757
	N	50	50	50	50	50	50	50	50	50	50	50	50	50	50
Responsiveness_Developer_Led_Product_Improvement	Pearson Correlation	,363**	,040	,247	,264	-,175	-,003	-,118	,054	,177	1	,596**	,043	,139	-,123
	Sig. (2-tailed)	,010	,781	,083	,063	,224	,984	,413	,712	,218		,000	,767	,337	,395
	N	50	50	50	50	50	50	50	50	50	50	50	50	50	50
Responsiveness_Developer_Led_Product_Innovation	Pearson Correlation	,017	-,020	-,056	-,055	-,108	-,102	-,161	-,102	,028	,596**	1	-,052	,017	,049
	Sig. (2-tailed)	,906	,888	,699	,705	,457	,483	,264	,482	,848	,000		,722	,904	,737
	N	50	50	50	50	50	50	50	50	50	50	50	50	50	50
Responsiveness_Customer_Led_Product_Improvement	Pearson Correlation	,399**	,540**	,572**	,557**	,148	-,240	,140	-,281*	-,166	,043	-,052	1	,350*	,054
	Sig. (2-tailed)	,004	,000	,000	,000	,306	,094	,334	,048	,250	,767	,722		,013	,710
	N	50	50	50	50	50	50	50	50	50	50	50	50	50	50
Responsiveness_Customer_Led_Product_Innovation	Pearson Correlation	,442**	,424**	,385**	,378**	,106	,041	,026	-,255	-,045	,139	,017	,350*	1	,108
	Sig. (2-tailed)	,001	,002	,006	,007	,463	,776	,856	,074	,758	,337	,904	,013		,454
	N	50	50	50	50	50	50	50	50	50	50	50	50	50	50
Published_Apps_by_Publisher	Pearson Correlation	-,040	-,142	-,118	-,111	-,234	,166	-,309*	-,163	,045	-,123	,049	,054	,108	1
	Sig. (2-tailed)	,785	,327	,416	,445	,102	,251	,029	,257	,757	,395	,737	,710	,454	
	N	50	50	50	50	50	50	50	50	50	50	50	50	50	50

**. Correlation is significant at the 0.01 level (2-tailed).

*. Correlation is significant at the 0.05 level (2-tailed).

Figure 41: Correlation Matrix