Supporting Developers in Addressing Human-centric Issues in Mobile Apps

Hourieh Khalajzadeh, Mojtaba Shahin, Humphrey O. Obie, Pragya Agrawal, and John Grundy

Abstract—Failure to consider the characteristics, limitations, and abilities of diverse end-users during mobile app development may lead to problems for end-users, such as accessibility and usability issues. We refer to this class of problems as *human-centric issues*. Despite their importance, there is a limited understanding of the types of human-centric issues that are encountered by end-users and taken into account by the developers of mobile apps. In this paper, we examine what human-centric issues end-users report through Google App Store reviews, what human-centric issues are a topic of discussion for developers on GitHub, and whether end-users and developers discuss the same human-centric issues. We then investigate whether an automated tool might help detect such human-centric issues and whether developers would find such a tool useful. To do this, we conducted an empirical study by extracting and manually analysing a random sample of 1,200 app reviews and 1,200 issue comments from 12 diverse projects that exist on both Google App Store and GitHub. Our analysis led to a taxonomy of human-centric issues that characterises human-centric issues into three-high level categories: App Usage, Inclusiveness, and User Reaction. We then developed machine learning and deep learning models that are promising in automatically identifying and classifying human-centric issues from app reviews and developer discussions. A survey of mobile app developers shows that the automated detection of human-centric issues has practical applications. Guided by our findings, we highlight some implications and possible future work to further understand and better incorporate addressing human-centric issues into mobile app development.

Index Terms—Human-centric issues, GitHub repositories, Google Play Store, human aspects, machine learning, deep learning

1 Introduction

Even though software systems, including mobile applications (apps), are designed to fulfill the expectations of their diverse end-users, negligence of *human-centric issues* during the software development process can lead to hard-to-deploy, hard-to-maintain, and hard-to-use software [1], [2], [3], [4], [5]. *We define human-centric issues as the problems end-users might face when using mobile apps that stem from the lack of (proper) consideration of their specific/differing human characteristics, limitations, abilities, personalities, technical proficiency, emotional reactions to software systems, gender, age, and so on.* Mobile app developers need to be aware and carefully investigate the human-centric issues [6]. However, they are not necessarily aware of, have not experienced, or do not understand and effectively communicate the implications of such issues.

- *Hourieh Khalajzadeh is with the School of Information Technology, Faculty of Science Engineering and Built Environment, Deakin University, Melbourne, Australia 3125. E-mail: hkhalajzadeh@deakin.edu.au*
- *Mojtaba Shahin is with the School of Computing Technologies, RMIT University, Melbourne, Australia, 3000. E-mail: mojtaba.shahin@rmit.edu.au*
- *Humphrey O. Obie is with the Department of Software Systems and Cybersecurity, Faculty of IT, Monash University, Melbourne, Australia, 3800. E-mail: humphrey.obie@monash.edu*
- *Pragya Agrawal is with the Department of Software Systems and Cybersecurity, Faculty of IT, Monash University, Melbourne, Australia, 3800. E-mail: pagr0009@student.monash.edu*
- *John Grundy is with the Department of Software Systems and Cybersecurity, Faculty of IT, Monash University, Melbourne, Australia, 3800. E-mail: john.grundy@monash.edu*
- *Hourieh Khalajzadeh is the corresponding author.*

Manuscript accepted 23 Sept 2022.

There is no evidence-based knowledge about how different types of human-centric issues are reported by the end-users and whether they are sufficiently being discussed and addressed during mobile apps development. This work aims to develop a comprehensive taxonomy of human-centric issues, and also to investigate whether an automated tool can help effectively detect such human-centric issues. We are ultimately interested in understanding how software/app developers perceive the usefulness of such a tool.

App reviews are a rich resource of the issues that end-users face when using an app. These include the human-centric issues that we are interested to learn more about, from the end-users' perspective. Additionally, developer discussions can be a major factor in deciding how a mobile app evolves, and include information beyond how an app works [7], [8]. Online software repositories, e.g., GitHub, attract a lot of discussions between developers on a variety of different topics. These repositories provide developers with perspectives on the issues they face during the apps development process and how they react to them. They play a significant role in improving the capabilities of app developers/end-users and accelerating apps development [9]. Analysing the comments that developers leave in response to the issues might reveal human-centric issues from the viewpoint of developers.

To achieve the goals of this study, we first manually analysed 1,200 app reviews and another 1,200 issue comments collected from 12 mobile app projects that exist on both Google App Store and GitHub. Our analysis led to a taxonomy of human-centric issues. The taxonomy includes three high-level categories: *App Usage*, *Inclusiveness*, and *User Reaction*. The App Usage category consists of *Resource Usage*, *Buginess*, *Change & Update*, *UI & UX*, *Privacy &*

security, *Usage instruction*, *Access issues*, and *Others*. The Inclusiveness category covers issues related to *Compatibility*, *Location*, *Language*, *Accessibility*, and *Others*. Finally, the User Reaction category consists of *Fulfilling interests*, *Emotional aspects*, *Preference*, and *Others*. Our analysis found that there are almost twice as many human-centric issues reported in app reviews to issue comments. End-users report more App Usage related issues, followed by User Reaction and Inclusiveness. Developers discuss more App Usage issues, and equally Inclusiveness and User Reaction. We also developed several machine learning (ML) and Deep Learning (DL) models with promising results to automatically detect human-centric issues from app reviews and issue comments. Finally, we conducted an online survey with 16 software/app practitioners, showing the usefulness of automated detection of human-centric issues in app reviews and issue comments in practice.

Some preliminary results of this study were published in [10]. In this paper, we build on top of this previous study by extending the first Research Question (**RQ1**) and adding two new RQs (**RQ2** and **RQ3**). More specifically, we built a whole new dataset with new projects compared to the dataset and projects used in [10], leading to a more comprehensive taxonomy of human-centric issues in apps.

The main contributions of this work include:

- Manually analysing a relatively large number of app reviews and issue discussions from 12 GitHub projects;
- Developing a taxonomy of human-centric issues discussed in different GitHub repositories and Google App Store reviews;
- Developing ML and DL based models to detect and classify human-centric issues in app reviews and GitHub issue comments;
- Providing some implications and possible future research directions to better manage human-centric issues in the software development process;
- Collecting and analysing developers' viewpoint on the usefulness of using an automated way to detect human-centric issues; and
- Building and publicly releasing a replication package to enable researchers and practitioners to access all collected data and replicate and validate our study [11].

The rest of the paper is structured as follows. Section 2 provides a motivating example for this research. Section 3 presents our research methodology. Sections 4 - 6 present the approaches and results related to our three RQs. Section 7 discusses and reflects on the key findings. In Section 8, we list the possible threats to the validity of our study. Finally, Section 9 reviews key related work, and Section 10 draws conclusions and proposes avenues for future work.

2 Motivation

Human aspects – including *age*, *gender*, *culture*, *language and location*, *digital literacy*, *physical and mental impairments*, and *differing personalities and preferences* – play an essential role in the uptake of software [12]. Implications of their negligence are detailed in [10]. Consider as a motivating example a dyslexic person who wants to access a website to get some information on their diet. This user has specific requirements to be able to access the website content. As one of the most popular software repositories, GitHub issue tracker provides an option for end-users and developers to report issues and provide feedback on a software system (e.g., a diet website) hosted on GitHub. A discussion in the issue tracker initiates with a title (*issue title*), followed by subsequent posts (*issue comments*) from reporters and contributors, including project maintainers, developers, users, or the reporter itself.

Figure 1 shows such an issue in the GitHub issue tracking system made by a collaborator to discuss the dyslexic user's preferences. This is followed by a comment from another collaborator listing some other barriers and asking whether this is true: *"I'm not sure I agree with it"*. The reason for such disagreement is probably that the developer is not fully aware of the needs and preferences of the user. However, discussing such issues can help developers be aware of such challenges and consider such issues when designing software. This example shows the importance of paying attention to and discussing issues related to human aspects (i.e., we refer to such issues in this paper as *human-centric issues*) in the uptake of the software.

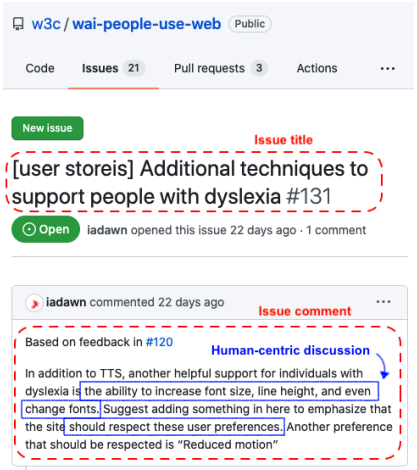

Fig. 1: An example human-centric issue from GitHub

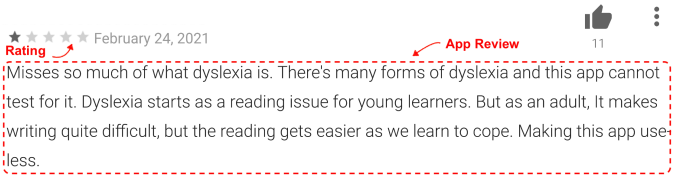

Fig. 2: An example human-centric issue on Google Play Store

App reviews, on the other hand, reflect on such aspects from the point of view of the end-users of the system. An app review includes a *comment* and a *rating* posted by the user, and the *number of likes* it has received from other users reading the comment. An example of an app review posted for Dyslexia Reading Test App by a dyslexic user is shown in Figure 2. Even though the app is designed for a dyslexic user, it is reported as useless by a user with a specific type of dyslexia. This example shows the importance of being aware of the needs of the users, which is not necessarily what a developer assumes. Awareness of and discussing

human-centric issues may lead to designing more inclusive software for diverse users. Such issues are not limited to the users with special needs. As another example, if an app has compatibility issues, it excludes users with a specific device or software from using it. If an app does not provide different languages, it excludes the users who do not understand the provided languages. Therefore, there is a need for better understanding, supporting, and promoting awareness of human-centric issues to be able to be able to design more inclusive software.

3 Research Method

Our study is motivated by the need to help practitioners and researchers be more aware of different types of end-user human-centric issues impacting software and to help identify possible areas for improvement and investment. We hope this would ultimately help us design software that better meets diverse end-users needs. To be able to achieve these, we formulated the following RQs:

> **RQ1**. What end-user human-centric issues typically manifest in mobile apps?

Motivation. Answers to **RQ1** will provide insights into different types of end-user human-centric issues in mobile apps. Previous research has extensively mined developer discussions in issue tracking systems and app reviews to extract different types of issues (e.g., socio-technical issues) in the software development process and software products. Hence, we use app reviews and developer discussions to identify and classify end-user human-centric issues.

> **RQ2.** Can we accurately and automatically classify end-user human-centric issues from developer discussions and app reviews?

Motivation. Although the answer to RQ1 informs app developers of different types of human-centric issues that may stem from an app, they still need to read the entire app reviews/issue comments of a mobile app to understand which of them include discussions on human-centric issues. If so, they also need to determine which types of human-centric issues are discussed in the app reviews and issue comments. Manually identifying and classifying human-centric issues might be tedious, time-consuming, and error-prone for app developers. An accurate and automated approach can help developers. To answer **RQ2**, we experiment with several ML/DL methods to automatically and accurately classify such issues in developer discussions and app reviews.

> **RQ3.** Do practitioners perceive that the automated classification of end-user human-centric issues in apps is useful?

Motivation. The automated classification of human-centric issues would be useful only if app developers perceive this process useful. We conduct an online survey to gather feedback from developers on the usefulness of the automated classification of human-centric issues. More specifically, as the survey respondents are not going to use the developed automated approaches in RQ2, our survey, similar to other studies [13], [14], [15], only seeks developers' opinions about possible benefits that the automated classification of human-centric issues might bring to mobile app development.

3.1 Data Collection

We needed a subset of open-source mobile apps to answer **RQ1** and **RQ2**. The issue tracking systems of such open-source mobile apps had to be available to access their developer discussions. At the same time, such mobile apps needed to be available on one of the popular mobile app stores to access their user reviews.

We leveraged a dataset of open-source mobile apps created by Mazuera-Rozo et al. [16] for this purpose. Mazuera-Rozo et al. built a dataset of 100 Android and 100 iOS mobile apps hosted on GitHub. These 200 Android and iOS mobile apps were randomly collected from an extensive collection of Android and iOS open-source apps identified through the open-source-android-apps project [17] and the open-source-ios-apps project [18]. Given the qualitative nature of **RQ1** and the lack of existing datasets of developer discussions and app reviews that include human-centric issues (**RQ2**), we applied a set of criteria on Mazuera-Rozo et al.'s dataset to sample a representative set of apps.

- **Android apps**. We only focused on Android apps among the top 100 Android [19] and 100 iOS mobile apps in Mazuera-Rozo et al.'s dataset [16].
- **Number of issue comments**. We further restricted our selection to Android apps with more than 100 issue comments, including closed and open issues in GitHub.
- **Number of stars.** We decided to opt for Android apps with more than 1,000 stars.
- **Available on the Google Play Store**. Android apps had to have an account on the Google Play Store and were available for Android users to post app reviews.
- **Number of downloads**. Finally, we only selected Android apps with more than 5 million downloads from the Google Play Store.

Applying these criteria led to reducing the number of Android apps in Mazuera-Rozo et al.'s dataset from 100 to 12 Android apps. Table 1 provides an overview of these 12 Android apps. We randomly collected 1,200 developer discussions (issue comments) from GitHub's issue tracking systems of these 12 apps. Each issue can include one or more issue comments. The 1,200 issue comments comprised 10 of 100 randomly selected issue comments from 12 Android apps. Similarly, we randomly selected 100 app reviews from each Android app to generate 1,200 app reviews.

4 Categories of End-user Human-centric Issues (RQ1)

4.1 Approach

As shown in Figure 3, the data analysis for RQ1 was conducted in two phases:

TABLE 1: List of projects studied in this paper. Number of Issues (❗); Number of Contributors (👥); Number of Stars (☆); Number of Forks (🍴); Number of Downloads in millions (📥)

Project Name	GitHub Repository	Google Play-Store ID	❗	👥	☆	🍴	📥
Signal	WhisperSystems/Signal-Android	org.thoughtcrime.securesms	8,871	230	20,700	4,900	50+
Bitcoin-wallet	bitcoin-wallet/bitcoin-wallet	de.schildbach.wallet	522	29	2,450	1,600	5+
Brave	brave/browser-android	com.brave.browser	819	11	1,028	186	10+
Duckduckgo	duckduckgo/android	com.duckduckgo.mobile.android	354	52	1,948	564	10+
Termux	termux/termux-app	com.termux	1,743	54	7,900	1,200	10+
Fbreader	geometer/FBReaderJ	org.geometerplus.zlibrary.ui.android	320	38	1,719	805	10+
K-9	k9mail/k-9	com.fsck.k9	3,118	226	5,800	2,200	5+
Pixel-dungeon	watabou/pixel-dungeon	com.watabou.pixeldungeon	66	1	2,788	1,000+	5+
Firefox	mozilla-mobile/fenix	org.mozilla.firefox	14,637	231	5,417	1,000+	100+
WordPress	wordpress-mobile/WordPress-Android	org.wordpress.android	6,468	149	2,478	1,200	10+
Cgeo	cgeo/cgeo	cgeo.geocaching	6,726	117	1,108	521	5+
Osmand	osmandapp/Osmand	net.osmand	7,568	746	22,674	817	5+

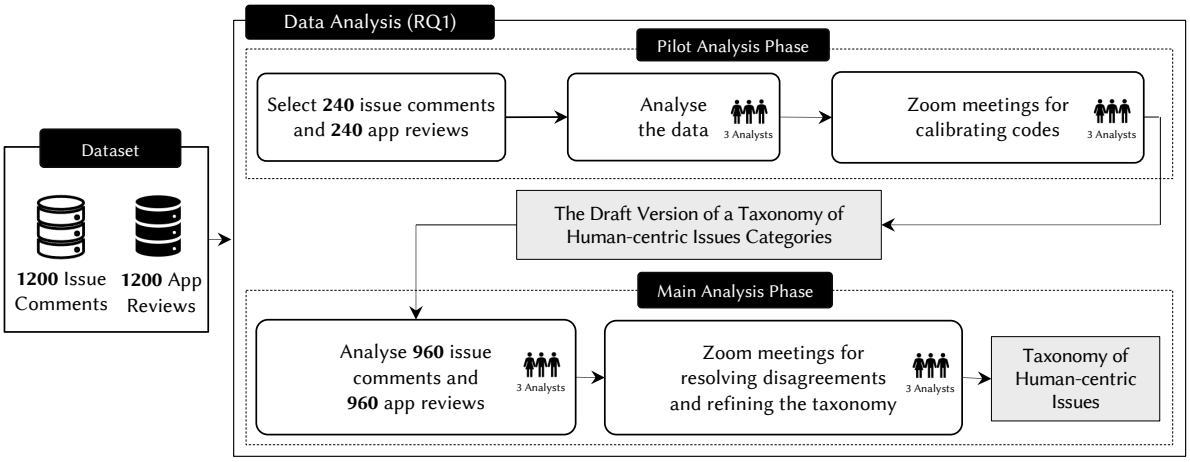

Fig. 3: An overview of data analysis for RQ1

4.1.1 Pilot Analysis Phase

We randomly chose 240 issue comments and 240 app reviews from 2,400 issue comments and app reviews, 20 app reviews and 20 issue comments from each project. The first three authors (analysts) independently conducted the open coding technique [20] to analyse the data. After finishing the coding process, they held several Zoom meetings to discuss the generated codes, identify duplicates, calibrate the codes, and resolve disagreements. This process resulted in the development of a draft version of a taxonomy of human-centric issues, which grouped human-centric issues into three categories: *App Usage*, *Inclusiveness*, and *User Reaction*. The App Usage category was further divided into *Resource Usage*, *Buginess*, *Change/Update*, *UI/UX*, *Privacy/Security*, *Usage Instruction*, and *Access Issues*. We also classified the Inclusiveness category into four subcategories: *Compatibility*, *Location*, *Language*, and *Accessibility*. Similarly, the User Reaction category was divided into *Fulfilling Interests*, *Emotional Aspects*, and *Preferences* subcategories. Our pilot analysis indicated that 185 (125 app reviews + 60 issue comments) out of the investigated 480 issue comments/app reviews included at least one human-centric issue. The pilot analysis phase led to three important observations:

Observation 1. The human-centric categories and subcategories are not mutually exclusive as an issue comment or app review could be coded with more than one human-centric issue category/subcategory. For example, the following app review was coded as "*App Usage*" and "*Inclusiveness*": "*The lack of universal night mode is the only thing preventing me from keep using this, the blinding white background on some sites are just unbearable.*"

Observation 2. The labelling process should not be driven by keywords. For example, we found many issue comments/app reviews that had bug-related words, but we did not label them in the "*Buginess*" category and vice versa. As an example, the following review was labelled as "*Buginess*", but it does not include bug-related keywords. "*It's nice to be able to see the ones the official geocaching apps make you pay for, but I can't see any way to log anything. After writing my log entry and selecting the date, I push the only button that looks like submit and it says it's downloading data, but I need to check my internet connection. I'm definitely online, so I don't know what's up.*"

Observation 3. Human-centric issues categories are not mutually exclusive to the technical categories in the literature, e.g., a review containing a technical bug report may also contain a human-centric issue. We found reviews/comments that purely discussed technical bugs and did not consider and label them as "*Buginess*", e.g. "*The bug is not resolved and very annoying*" is considered a non-

human-centric issue. Taking our definition of human-centric issues as a foundation for identifying human-centric issues in issue comments/app reviews, if a user discussed a technical issue in issue comments/app reviews that stems from the lack of (proper) consideration of their specific/differing characteristics, limitations, abilities, personalities, technical proficiency, emotional reactions to software systems, gender, age, and so on, we considered that comment/review as a human-centric issue. For example, the "*Buginess*" category should refer to any issues discussing the bugs an app has, which impacts the usage of the app by different users. The same process should be applied to the rest of the categories of human-centric issues in the taxonomy.

4.1.2 Main Analysis Phase

In this phase, the same analysts in the pilot phase analysed the rest of 1,920 issue comments/app reviews in several iterations. Each of the analysts manually analysed 640 individual issue comments/app reviews. In other words, each issue comment was analysed and labeled by two analysts. In each iteration, the analysts analysed approximately 200 issue comments or app reviews. We created a spreadsheet and shared it with three analysts based on the taxonomy created in the pilot analysis phase. The analysts were asked to indicate whether an issue comment or an app review included one or more human-centric issues. If so, they had to specify which of the categories/subcategories in the taxonomy of human-centric issues the given issue comment or app review belonged to and put "1" in the corresponding columns in the spreadsheet. The analysts had the freedom to capture and add any new human-centric issues category/subcategory to the taxonomy if they were not captured in the draft version of the taxonomy.

At the end of each iteration, the three analysts had a roughly 1-hour Zoom meeting to compare their labelling results and resolve possible disagreements. The majority of disagreements were resolved through discussions between the two assigned analysts by explicitly providing the reason behind their choices. The third analyst was asked to read and label the conflicting issue comment if the two analysts could not agree. Then, we voted to resolve the disagreement. We had a total of 49 issue comments among 1,200 issue comments and 56 app reviews among 1,200 app reviews that we further discussed. The reason for discussing them was not always that we did not come up with the same categories, sometimes because one of the analysers flagged the issue comment/app review for further clarification and discussion. We were able to reach an agreement for all of them after the initial discussion.

We found a few new human-centric issues compared to the draft version of the taxonomy and grouped them as "others" in each category. We found 3 issue comments related to the "app usage" category that did not belong to any of the sub-categories, 2 in the "inclusiveness" and 1 in the "user reaction". We categorised them in new categories as the "others" sub-categories under each category. In app reviews, we found 19 app reviews related to the "app usage" category, 3 in the "inclusiveness" and 4 in the "user reaction" that did not belong to any of the sub-categories, and similarly, categorised them in new "others" sub-categories under each category. The final taxonomy is shown in Figure 4. We acknowledge that our process in the main phase does not follow the idea of open coding. We decided this since it could avoid developing a potentially very large number of possible human-centric issue categories [21]. Also, it supported the analysts to reach and use consistent labelling without introducing substantial bias [21].

4.2 Findings

This section presents the taxonomy of human-centric issues.

4.2.1 App Usage

The App Usage category covers all the issues related to the app itself and how it impacts users experiences. This category consists of Resource Usage, Buginess, Change & Update, UI & UX, Privacy & Security, Usage Instruction, Access Issues, and Others.

Resource Usage: Resource usage refers to the issues end-user have with battery usage, internet connection, server, memory and resource management. An example of Resource Usage issue discussed in GitHub is:

💬 ***GitHub Issue*** - *"... continuously tries to detect the postal address... But in any case,* ***the battery drain sounds just too excessive****..."* - (Osmand)

A user reports a review of Bitcoin wallet app on Google App Store mentioning that "*there REALLY needs to be a way to* ***limit bandwidth*** *used when syncing with the blockchain.*" The user has bought a new phone, and transferred the wallet backup from the old phone and started the restore process, and is complaining that "*It's used* ***100GB of data so far in the last 24 hours****, and there are still three YEARS worth of blockchain to go.*" - (Bitcoin wallet)

Buginess: Buginess category refers to any issues discussing the bugs an app has, which impacts the usage of the app by different users. Examples of Buginess issue reported in GitHub and Google App Store are shown below:

💬 ***GitHub Issue*** - *"****I had one crash*** *after long time of using the live map, but was* ***unable to debug/reproduce****. Most probably a* ***problem*** *we have had already in the past."* - (Cgeo)

💬 ***App Review*** - *"... I find the support to be lacking and the* ***bugs to be plentiful****. Good luck getting rid of it though,* ***tried closing my account due to number of bugs*** *and no one could contact me anymore via SMS."* - (Signal)

Change & Update: This category refers to any issues that did not exist before, however, users face when updating the app or upgrading to a new version. Users sometimes feel unhappy with the new changes made and prefer or were more comfortable with the older options, or features. An issue discussed on GitHub relates to a user having always had some options turned off (unchecked), but with the new upgrade, "***these checkbox preferences*** *don't seem to be being respected when composing a message.*" - (K - 9). Another user complains about a "*Disastrous update*" through Brave App review, asking how they can get the **previous version** since "*Switch between tabs has become extremely difficult.*" - (Brave)

UI & UX: Any discussions related to the User Interface (UI) and User Experiences (UX) with the app are labeled in this category. Examples of UI & UX issue reported in GitHub and Google App Store are shown below:

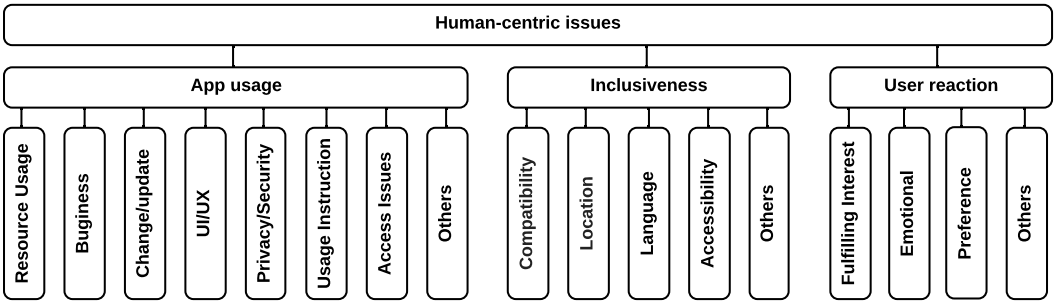

Fig. 4: A taxonomy of end-user human-centric issues

💬 ***GitHub Issue*** - *"We also need to consider the position of the toolbar for custom tabs. The browser is meant to* ***look more*** *like the native app itself to give the illusion of not leaving the context of the calling app, so we might want to think about re-positioning the toolbar to the top like most apps would have."* - (Firefox)

💬 ***App Review*** - *"...However please provide a navigation tool to understand the features. Tough for* ***not so technical people*** *to navigate.* ***Also please improve your notification icon. Make it catchy.*** *Most of new users don't even know that there is a message."* - (Signal)

Privacy & Security: This covers issues related to the users' privacy, security, data protection, reliability, and trust concerns. We classified discussions on accessing the location and private data of a user into this category. As an example, a developer has claimed on GitHub, that "*We frequently audit our browser to* ***detect any possible privacy implications****. If you have* ***detected any possible security issue****, reach us via hackerone, it will go through our privacy expert team...*" - (Duckduckgo). However, a user reported that "*Duckduckgo is* ***advertised as a security and privacy web browser****, yet there are almost no advanced options to customise those aspects. What about https, dns, and script options? or control over what information is shared back to you guys?*" - (Duckduckgo)

Usage Instruction: Some apps do not provide enough instructions for the users to be able to use it. Lack of usage instruction leads to some confusion for the users. We report all such issues in this category. For example, a developer discusses how to resolve such issue on GitHub:

💬 ***GitHub Issue*** - *"I am going to extrapolate here and guess that the average user will find the same problem, and may never find this context menu.* ***To solve that, we could either introduce a UI hint on first use****, or make it more hangouts-y..."* - (Signal)

And a user reports their experience using an app lacking enough documentation and instructions:

💬 ***App Review*** - *"There are several map rendering styles but unfortunately I was not able to find one that rendered the map in a way I liked for use when walking in GB. It is possible to create ones own rendering style which I have done* ***but documentation on how to do so is appalling and support from OsmAnd was lacklustre.***" - (Osmand)

Access Issues: Users sometimes have issues with downloading, registering and accessing and app. All these issues are categorised in Access issues category. There were a lot of app reviews reporting Access issues, such as a user reporting "*The app does not install in my phone please any advice to install in phone.*" - (Termux). However, we did not find any GitHub issues discussing how to deal with such an issue.

Others: Finally, we categorise all other issues related to the usage of the app, that are not directly related to the previous categories, in the Others category. Examples of Others issue reported in GitHub and Google App Store are shown below. The first example discusses how the developer thinks users expect/prefer and how to react accordingly, and the second one shows the lack of trust by the user to the app.

💬 ***GitHub Issue*** - *"I believe that we should sync the deletion of individual pages but that we shouldn't sync bulk deletion like "Clear All". Given mobile devices will have synced desktop History, a "Clear All" will wipe your desktop History on your local mobile device but we don't think* ***users would expect it to also wipe all of the History on their desktops too.***" - (Firefox - Android)

💬 ***App Review*** - *"...This is my 3rd month claim, brave is* ***fooling there users****. Or they* ***never added support programme for users****.Don't install."* - (Brave - Android)

4.2.2 *Inclusiveness*

This category covers the issues related to the inclusion, exclusion or discrimination toward a specific groups of users. It includes issues related to the age, gender, and socio-economic status of the users. We categorise Inclusiveness into five different subcategories, as: Compatibility, Location, Language, Accessibility, and Others.

Compatibility: Any discussions around the compatibility of an app with different devices, operating systems, and platforms are included in this category. Compatibility issues are normally thought of as technical, not human-centric issues. However, a common reason for them occurring can be because of the users' socio-economic status, i.e., not having access to the latest phones, or the developers' ignorance, i.e., not taking all different platform choices into account. Examples of Compatibility related issues reported on GitHub and Google App Store are shown below:

💬 ***GitHub Issue*** - *"...it appears that* ***FbReader 1.6.4 is marked as incompatible on my CM7.2*** *(Android 2.3.7) phone, even though the maxsdk is 10. FBReader 1.6.1 is compatible. Are we building a wrong branch perhaps?"* - (FbReader)

A user left a comment through Bitcoin wallet app review, complaining that they want to transfer to another app: "*Shame because it's been simple and reliable and easy to backup/restore...* ***Just isn't compatible with latest wallet standards.***" - (Bitcoin wallet)

Location: This covers issues related to the physical location from where the user is accessing the software. Based on our analysis, users' access may be limited if they are visiting a country and have no local phone number or App store account. A developer discusses on GitHub that *"need to be sure the field notes contain the correct z-time...It will only work if* ***user sets their time zone*** *on gc.com as well."* - (Cgeo). Through app reviews, a user has reported that *"The search engine is pretty bad, anything I search for* ***only shows results in the USA and beyond****.. Good but I have to put UK at the end of every search."* - (Duckduckgo)

Language: This category includes discussions about language or culture-related issues. For example, a user asks "*how about adding in a* ***Language Translation program/option*** *into DuckDuckgo, so you can translate other websites and mostly all* ***foreign languages****?*" - (Duckduckgo). On the other hand, on Duckduckgo GitHub, in response to another language-related issue, a developer discusses that "*...we don't currently support* ***Brazilian Portuguese****. We would love to* ***support as many languages as possible****, but at the same time, we need to keep a manageable size of translations...*" - (Duckduckgo)

Accessibility: This category covers issue comments and app reviews discussing accessibility issues. Discussions about the users with physical and mental impairments also fall into this category. Examples include:

💬 ***GitHub Issue*** - *"****If we disable accessibility services, how are disabled users*** *going to enter their passwords and passphrases?"* - (Bitcoin wallet)

💬 ***App Review*** - *"Not intuitive.* ***Difficult to change fonts, text size, etc.*** *Couldn't not figure out how to purchase a book."* - (FbReader)

Others: All other issues related to the inclusiveness of diverse users that are not directly related to the previous categories are considered in the Others category. Examples of Others issue reported in GitHub and Google App Store are shown below. These two examples are related to the technical knowledge and background of the users.

💬 ***GitHub Issue*** - *"The reason for this to ship in Firefox itself, instead of an add-on is* ***that many users do not know about add-on support. Most does not even know what an add-on is.*** *They just install the browser and get on with it. This should be in Firefox itself, if you ask me!"* - (Firefox)

💬 ***App Review*** - *" Maybe have the general search in the center of the app so people don't think they have to know the exact website address they are looking for in order to use the app.* ***That might be confusing to some who aren't so tech savvy.***" - (Duckduckgo)

4.2.3 User Reaction

This category covers the issues users have when accessing the application due to their specific preferences, interests, and emotional reactions. We further divided this into **Fulfilling interests**, **Emotional aspects**, **Preference**, and **Others**.

Fulfilling interests: Users use apps for different purposes. Some applications might be suitable for a user based on their goals and objectives, but not for another user with different needs. We categorise discussions related to user satisfaction due to the fulfilment of their interests in this category. For example, an issue comment discusses that due to the use of third-party libraries, a user is about to switch eBook viewers. "*FBReader is currently just not usable on my device, and eBook viewer is the main reason I use the device.*" - (FbReader). A user also reports that *"Termux isn't a good app to use its a app exec controlled app* ***they don't let you use sudo and the commands don't execute as you type them. Don't waste your time on this app."*** - (Termux). These two examples show how not fulfilling users' interest might disappoint them from using the app.

Emotional Aspects: This category includes the possible emotional impacts that the software can have on the users, including making the users confused, worried, scared and bored. Examples of Emotional Aspects issue reported in GitHub and Google App Store are shown below:

💬 ***GitHub Issue*** - *"The amount of off-topic content here is* ***shameful, and very stressful*** *for anyone that then has to clean up your mess. I really don't see how you expect a collaborative response when this is the attitude here."* - (Duckduckgo)

💬 ***App Review*** - *"After some update the app crashes on some specific access to the contacts* ***which is really really really annoying*** *when you have written a long mail and you loose all of it even in the draft. If the problem isn't fixed I am changing app forever"* - (K - 9)

Preferences: Any discussion related to the user's preferences fall into this category. This relates to the features or functionalities that users prefer based on their specific human characteristics. Preference-related discussions include different aspects: (1) requesting new features (2) issues or requests to change an existing feature, such as the position of user interface elements (3) and privacy-related issues due to personal reasons. Preferences are sometimes discussed according to users' feedback received through app reviews or by developers from the users' perspective. In some apps, developers often use the app themselves and discuss their usage experiences on GitHub. An examples of Preferences issue discussed in GitHub is that the developer has made some changes in dictionary code "*because not all users want to use English - German dictionaries. E.g.,* ***I prefer English - Russian & German - Russian****. ;) I think that is good idea to list only* ***'universal dictionaries'.***" - (FbReader)

A user on Google App Store complains about Firefox: "*no option to even download features that make the pc firefox so effective and easy to use. No clear recent search history and Bookmark browsing that takes painfully slow to scroll through, no book mark search feature.* ***I would rate this 5 stars but I would prefer Firefox not Firefox lite***" - (Firefox)

Others: All other issues related to the reaction of different users based on their diverse characteristics not directly related to the previous categories are considered in the Others category. Bellow shows two examples in the Others category reported in GitHub and Google App Store:

💬 ***GitHub Issue*** - *"...I will have a look at tomorrows NB and try again from the perspective of* ***a first time user***" - (Cgeo)

App Review - "I can't report an issue *In the app either because every time I press the option to report it just brings up the typical share menu for Android devices where it recommends apps or people to share with."* - (Bitcoin wallet)

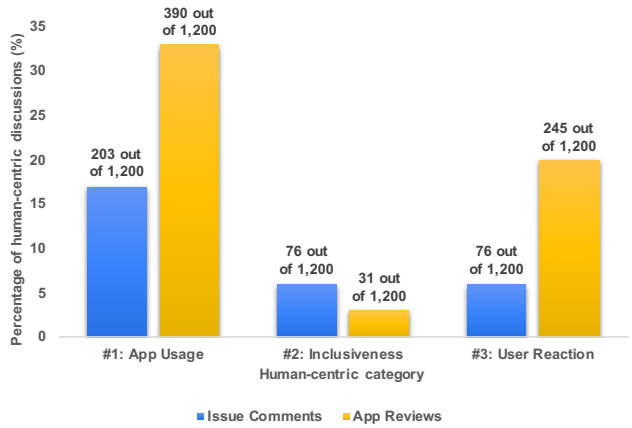

Fig. 5: Number and percentage of human-centric issues out of 1,200 issue comments and 1,200 app reviews in our 12 projects

4.3 Human-Centric Issues in Issue Comments Vs App Reviews and in Different Apps

Table 2 and 3 provide detailed information on the human-centric issues categories in the issue comments and app reviews of the 12 android projects. Overall, 25.5% of the issue comments (306 out of 1,200 issue comments) and 47.25% of the app reviews (567 out of 1,200 app reviews) studied in these 12 projects, discuss human-centric issues. These numbers suggest that human-centric issues are reported more frequently by the users through app reviews, than discussed by the developers through issue comments. How these human centric issues are spread among different categories, is shown in Figure 5.

In app reviews, out of the 567 human-centric issues, 390 reviews (32% of total reviews) are related to App Usage, 31 (0.2%) are related to Inclusiveness, and 245 (20%) User Reactions related. On the other hand, among 306 human-centric developers discussions in GitHub issue comments, 203 reviews (17% of total comments) are related to App Usage, 76 (6%) related to Inclusiveness, and 76 (6%) User Reactions related. Comments can belong to more than one category and therefore, the total percentage of different categories can be greater than the percentage of the total number of human-centric issues. Overall, App Usage followed by Inclusiveness and User Reaction are the most prevalence human-centric related discussion among developers, while App Usage, followed by User Reaction, and rarely Inclusiveness are more highly reported by the users.

The right columns in Table 2 and 3 show the total number of issues/reviews out of the 100 selected ones that are having at least one human-centric issue, and can discuss various issues. In issue comments, Signal (35 out of 100) has the maximum number of issue comments discussing human-centric issues followed by Cgeo (33 out of 100) and Firefox (30 out of 100), while Pixel-dungeon (16 out of 100), FBreader (18 out of 100) and Bitcoin (19 out of 100) have the minimum number of discussions. In app reviews, Firefox (79 out of 100), followed by K-9 (68 out of 100) and Signal (66 out of 100) have the maximum number of human-centric issues reported by the users, and Pixel-dungeon (17 out of 100), has limited number of reported human-centric issues.

RQ1 Summary. Our taxonomy of human-centric issues constructed based on manual analysis of app reviews and issue comments include three high-level categories: **App Usage**, **Inclusiveness**, and **User Reaction**. Human-centric issues are reported more frequently by the users through app reviews, than being discussed by the developers in issue comments. App Usage related issues are the most popular category to be discussed among both developers and end-users, followed by Inclusiveness among developers, and User Reaction by the users.

5 Automated Classification of End-user Human-centric Issues (RQ2)

5.1 Approach

In this section, we present our ML and DL models to automatically classify the app reviews and issue comments we collected and labeled, as described in Section 4. We only focused on identifying and classifying three high-level categories of human-centric issues in the taxonomy and adopted multi-label classification techniques for this purpose [22].

Dataset. We used the two datasets built in Section 4: 1) app reviews collected from Google Play Store, 2) issues comments from GitHub, and a combination of both app reviews and issues comments. Due to the data limitation, we aggregated the two datasets to see if merging the two varied sets of comments would help in model predictions. In total, we had a combination of 2,400 comments (1,200 app reviews, 1,200 issue comments) that we utilised to train our models for classification. We considered the classes as our three high-level categories identified in Section 4, and one extra category as a non-human-centric category. Non-human-centric refers to a comment/issue that does not belong to any of the human-centric categories.

Method. We developed and applied various ML and DL models to our datasets and evaluated their performance based on various metrics. Before employing the models, we applied data pre-processing to ensure that the data is clean and free from any noise. Here we discuss the pre-processing steps and models in detail.

5.1.1 Machine Learning

Pre-processing. The pre-processing phase for our ML models spans over five major steps.

Step 1. Convert Case: Conversion of all text to lower case helps in maintaining the consistency of the data, as it mitigates the same words of different cases being recounted in the vocabulary set.

Step 2. Remove Noise: Noise usually affects the classification accuracy, training time and size of the classifier, which leads to a faulty or erroneous representation of the

TABLE 2: Number (#) and percentage (%) of human-centric issues in issue comments in our 12 projects

	#1: App Usage								#2: Inclusiveness					#3: User Reaction				total (Out of 100)
	Resource Usage	Buginess	Change/Update	UI/UX	Privacy/security	Usage instruction	Access issues	Others	Compatibility	Location	Language	Accessibility	Others	Fulfilling interests	Emotional aspects	Preference	Others	
Signal	5	14	2	2	5	2	0	1	3	0	1	5	0	0	4	4	0	35
Cgeo	2	7	3	7	0	3	0	0	5	1	2	2	0	2	0	8	1	33
Firefox	3	1	0	9	7	0	0	1	1	0	1	9	1	0	1	10	0	30
Duckduckgo	2	8	1	6	8	1	0	1	0	0	3	0	0	1	1	6	0	29
K-9	2	8	3	6	4	0	0	0	4	0	1	0	0	2	0	6	0	29
WordPress	0	6	0	12	0	0	0	0	1	0	2	0	0	0	0	6	0	26
Brave	1	12	0	12	0	1	0	0	2	0	0	0	0	0	0	4	0	24
Osmand	3	2	0	6	1	6	0	0	2	0	3	0	0	1	0	10	0	23
Termux	2	5	1	6	5	1	0	0	1	0	0	1	0	0	0	1	0	22
Bitcoin	2	6	1	2	6	1	0	0	0	0	1	3	1	0	0	2	0	19
Fbreader	0	2	0	4	1	0	0	0	1	0	7	0	0	2	0	5	0	18
Pixel-dungeon	0	0	0	1	0	0	0	0	3	0	11	0	0	0	0	1	0	16
Total (#)	22	71	11	73	37	15	0	3	23	1	32	20	2	8	6	63	1	
Total (%)	1.83	5.92	0.92	6.08	3.08	1.25	0.00	0.25	1.92	0.08	2.67	1.67	0.17	0.67	0.50	5.25	0.08	

TABLE 3: Number (#) and percentage (%) of human-centric issues in app reviews in our 12 projects

	#1: App Usage								#2: Inclusiveness					#3: User Reaction				total (Out of 100)
	Resource Usage	Buginess	Change/Update	UI/UX	Privacy/security	Usage instruction	Access issues	Others	Compatibility	Location	Language	Accessibility	Others	Fulfilling interests	Emotional aspects	Preference	Others	
Firefox	6	19	23	35	9	2	0	0	1	0	0	0	0	16	3	23	0	79
K-9	2	22	14	13	0	1	1	0	5	0	0	0	0	10	1	23	0	68
Signal	8	17	4	3	5	0	0	0	0	0	0	1	0	4	0	34	0	66
Bitcoin	7	12	4	5	9	5	1	10	1	1	1	1	0	5	0	7	4	58
Osmand	1	10	4	21	0	4	0	2	0	4	1	1	0	11	0	9	0	57
Brave	1	10	12	14	2	1	0	1	0	0	0	0	1	5	2	9	0	46
Duckduckgo	4	6	3	11	3	0	2	1	1	2	1	3	1	6	1	21	0	44
Fbreader	2	9	9	11	0	1	0	3	1	0	0	1	1	3	1	12	0	39
Termux	1	15	2	3	2	2	4	1	1	0	0	0	0	2	0	4	0	35
Cgeo	1	8	2	5	0	0	5	0	0	0	0	0	0	6	0	9	0	29
WordPress	0	10	3	6	0	3	0	1	0	0	1	0	0	1	0	8	0	28
Pixel-dungeon	0	2	3	4	0	0	0	1	0	0	0	0	0	1	0	8	0	17
Total (#)	33	140	83	131	30	19	13	20	10	7	4	7	3	70	8	167	4	
Total (%)	2.75	11.67	6.92	10.92	2.50	1.58	1.08	1.67	0.83	0.58	0.33	0.58	0.25	5.83	0.67	13.92	0.33	

data. Noise handling technique involves [23]: 1) Removing numbers, 2) Removing punctuation and special characters, e.g.: "Hello!!!", "PS: please reply me back", and 3) Remove patterns, eg: "grreeettttt", "Soooon".

Step 3. Normalisation: Normalisation considers the abbreviations used and even though there is not any defined vocabulary for such words, some common occurrences of typically used words are converted to their original form. For example, "wasn't" is converted to "was not".

Step 4. Stemming: The process of converting the word to its word stem is called stemming [24]. We used the generally adopted Snowball stemmer, which is a better and aggressive version of porter stemmer [24].

Step 5. Stopwords: Words, such as "a", "the", "is", "are", which occur commonly and do not add valuable information to the modelling were removed in this step.

Feature Extraction. In this step, we employed various techniques to extract features from the corpus. These techniques are detailed as follows.

TF-IDF: In TF-IDF method, TF refers to term frequency in a document and IDF refers to inverse document frequency. TF-IDF gives the words occurring rarely more weight than the commonly occurring words. TF-IDF supports functionalities such as analyser and ngrams to try and improve fea-

ture extraction techniques. We tried multiple ngrams such as (1,1)(1,2)(1,3)(2,2)(3,3)(4,4). Moreover, we tried switching the analyser to 'char' to extract features based on character n-gram combinations instead of words [25].

Word2vec: Word2vec is a technique of processing text to numerical vectors which uses neural network model to create word embedding. We employed Google pre-trained word2vec model to convert words to their corresponding vector equivalent for a given sentence. This method helps in preserving the semantic of the sentence [25].

Word2vec & TF-IDF stack: Employing word2vec and TF-IDF simultaneously helps get better performance [26]. We investigated to know whether or not stacking the two models together will improve the feature extraction process.

Multi label classification strategies. Some ML models are not built to classify multi label data and therefore we needed to apply one of the following methods to convert multi label data into multiple binary class problems [27].

One-vs-Rest: It splits the data by accounting for one label at a time and grouping the rest. The process decomposes the labels into multiple binary classification problems where the model is equipped for each label. One-vs-Rest allows the model to focus on one attribute at a time, hence the model learns in a binary environment. We selected a sample of data from one class at a time and trained the binary classifier models to distinguish between the class and the others [28].

Classifier chains: It uses binary classification for multiple classes where each label is independently considered. This model works on interdependencies (correlation) of the label to boost performance and lower computational complexity. It is a feed forward model that takes input from the previous classifier and passes the output to the next one [29].

Classification Models. We used several ML models suitable for multi-label classification.

Logistic regression: The Linear Logistic (LR) function is a powerful discriminative method for independent binary variables [30]. LR is found to be very efficient in performing multi-label classification [27].

Support Vector Machine (SVM): Empirical study by Zhang et al. suggests that standalone SVM models are a viable option for multi-label classification [31]. We used a generic model, i.e. without Hyperboosting using ADTree or ADABoost techniques to check the baseline performance.

Random Forest (RF): RF is an ensemble method that uses multiple decision tree classifiers on chunks from a dataset. It averages the learning from each tree to improve its overall prediction. By using Gini index (joint score), RF can be used to simultaneously predict from multi label data [32].

XGBoost (XGB): XGB is a gradient boosting library that implements ML algorithms under its framework. Its ability to use parallel tree boosting makes it highly efficient and flexible [33].

5.1.2 Deep Learning

We used BERT [34], RoBERTa [35], and DistilBERT [36], which are the state of the art classifiers designed to bi-directionally train on the overall context of data in all layers. BERT is pre-trained on Wikipedia and Book corpus that is helpful in English text. However, since our dataset may have words that are technical and unknown to the model [37], we processed and trained the model on pre-trained data and by adding more output layers. Robustly optimised BERT approach (RoBERTa) is a replication study of BERT pretraining that measures the impact of key hyperparameters and training data size on top of BERT. RoBERTa is trained with dynamic masking, FULL-SENTENCES without Next Sentence Prediction (NSP) loss, large mini-batches and a larger byte-level Byte-Pair Encoding (BPE) [35]. DistilBERT is another extension of BERT that is pre-trained on a smaller general-purpose language representation model and can then be fine-tuned on a wide range of tasks resulting in good performances. DistilBERT is reported to reduce the size of a BERT model by 40%, while retaining 97% of its language understanding capabilities and being 60% faster, and also cheaper to pre-train [36].

As the first step, we needed to transform the data as required for these three models. Hence, we followed data pre-processing that allows them to interpret the data.

Pre-processing. For BERT, RoBERTa and DistilBERT classifiers, we followed the same steps as described in Section 5.1.1. Additionally, we needed to transform the text data into numerical values within the word limit specified by Transformer models (512 words at a time). Therefore, we tokenised and encoded the data in a structure as specified by HuggingFace (transformers) library [38]. The package provides API to perform NLP tasks to utilise its capacity over a vast variety of tasks [27].

Classification Models. The BERT classifier is ready to be utilised directly on the model, but it needs to be fine-tuned to produce better results. In a text classification task, the BERT base model outputs a vector of length 768 for each word (token) and pools the output. The pooled output contains overhead information while training, which helps in improving the predictions. Given we achieved sufficiently useful results for these three models, in order to keep the models comparable, we used the same settings as BERT for RoBERTa and DistilBERT models.

5.1.3 Performance Metrics

We considered 75%-25% ratio for our training and test sets, and used different metrics to measure the performance of our models.

Accuracy: Accuracy provides the ratio of correctly labelled data to complete data as either *right* or *wrong* [39]. This disregards the possibility of partial correctness of labels, and hence the disadvantage of this measure is that multi-label classification problems may report low score for models capability to identify some labels from the set [40].

Micro precision and recall: Precision and Recall can be calculated in either Micro or Macro averages. We rely on Micro for our evaluation as it considers the aggregate contribution of all classes for the metric.

Micro F1: Micro F1 is the harmonic mean of micro precision and micro recall which accounts for both false positives and false negatives. It takes frequency of label as a contributor into consideration while evaluating performance of a model. Hence, micro averaged F1 score works extremely well in highly imbalanced distribution of tags [25].

Hamming loss: In a multi label classification problem, partially correct predictions are not rewarded and hence the model performance cannot be determined completely. Hamming loss accounts for partially correct prediction to

TABLE 4: Results of different models on three different datasets

ML Strategy	Models	App Reviews					Issue Comments					App reviews + Issue comments				
		Precision	Recall	Accuracy	F1_Score	Hamming Loss	Precision	Recall	Accuracy	F1_Score	Hamming Loss	Precision	Recall	Accuracy	F1_Score	Hamming Loss
One vs Rest	tf–idf + LR	0.73	0.71	0.61	0.72	0.15	0.83	0.75	0.73	0.79	0.11	0.76	0.74	0.67	0.75	0.13
	tf–idf + SVM	0.75	0.69	0.60	0.72	0.15	0.82	0.70	0.72	0.75	0.12	0.81	0.60	0.62	0.69	0.14
	tf–idf + RF	0.77	0.58	0.56	0.66	0.16	0.83	0.70	0.71	0.76	0.11	0.78	0.58	0.60	0.67	0.15
	tf–idf + XGB	0.71	0.62	0.55	0.66	0.18	0.81	0.71	0.69	0.75	0.12	0.77	0.66	0.63	0.71	0.14
	word2vec + LR	0.58	0.73	0.47	0.65	0.22	0.58	0.73	0.47	0.65	0.22	0.50	0.71	0.39	0.58	0.27
	word2vec + SVM	0.57	0.73	0.45	0.64	0.23	0.79	0.69	0.71	0.73	0.13	0.82	0.58	0.60	0.68	0.15
	word2vec + RF	0.80	0.50	0.49	0.61	0.17	0.78	0.74	0.74	0.76	0.12	0.75	0.62	0.63	0.68	0.16
	word2vec + XGB	0.77	0.57	0.53	0.66	0.17	0.79	0.69	0.71	0.73	0.13	0.77	0.61	0.60	0.68	0.15
	word2vec + tf–idf + LR	0.68	0.73	0.57	0.70	0.17	0.73	0.74	0.67	0.73	0.14	0.69	0.74	0.60	0.71	0.16
	word2vec + tf–idf + SVM	0.69	0.69	0.54	0.69	0.17	0.81	0.72	0.74	0.76	0.12	0.84	0.62	0.63	0.71	0.13
	word2vec + tf–idf + RF	0.81	0.54	0.54	0.65	0.16	0.79	0.73	0.74	0.76	0.12	0.77	0.63	0.64	0.69	0.15
	word2vec + tf–idf + XGB	0.78	0.61	0.55	0.68	0.15	0.83	0.72	0.71	0.77	0.11	0.80	0.64	0.62	0.71	0.14
Classifier Chains	tf–idf + LR	0.73	0.72	0.66	0.73	0.15	0.77	0.81	0.76	0.79	0.11	0.74	0.76	0.70	0.75	0.13
	tf–idf + SVM	0.74	0.71	0.65	0.72	0.15	0.80	0.79	0.79	0.80	0.10	0.76	0.74	0.72	0.75	0.13
	tf–idf + RF	0.75	0.62	0.60	0.68	0.16	0.81	0.70	0.72	0.75	0.12	0.75	0.62	0.65	0.68	0.16
	tf–idf + XGB	0.68	0.65	0.61	0.66	0.18	0.78	0.76	0.76	0.77	0.12	0.72	0.70	0.69	0.71	0.15
	word2vec + LR	0.59	0.69	0.52	0.64	0.22	0.38	0.54	0.44	0.45	0.34	0.44	0.57	0.41	0.50	0.31
	word2vec + SVM	0.57	0.65	0.50	0.61	0.23	0.74	0.71	0.74	0.73	0.14	0.69	0.65	0.67	0.67	0.17
	word2vec + RF	0.70	0.57	0.57	0.63	0.19	0.76	0.75	0.75	0.76	0.13	0.70	0.67	0.68	0.68	0.17
	word2vec + XGB	0.69	0.65	0.63	0.67	0.18	0.78	0.76	0.77	0.77	0.12	0.71	0.67	0.68	0.69	0.16
	word2vec + tf–idf + LR	0.68	0.71	0.59	0.70	0.17	0.69	0.76	0.70	0.72	0.15	0.69	0.74	0.65	0.71	0.16
	word2vec + tf–idf + SVM	0.69	0.68	0.62	0.68	0.17	0.77	0.76	0.75	0.76	0.12	0.74	0.73	0.71	0.74	0.14
	word2vec + tf–idf + RF	0.80	0.52	0.53	0.63	0.17	0.79	0.74	0.75	0.77	0.12	0.76	0.62	0.64	0.68	0.15
	word2vec + tf–idf + XGB	0.68	0.63	0.61	0.65	0.18	0.79	0.77	0.78	0.78	0.11	0.75	0.71	0.71	0.73	0.14
BERT Classifier		**0.86**	**0.86**	**0.86**	**0.86**	**0.14**	0.88	0.88	0.88	0.88	0.13	0.83	0.83	0.83	0.83	0.17
RoBERTa Classifier		0.77	0.77	0.77	0.77	0.23	0.88	0.88	0.88	0.88	0.13	**0.88**	**0.88**	**0.88**	**0.88**	**0.13**
DistilBERT Classifier		0.79	0.79	0.79	0.79	0.21	**0.89**	**0.89**	**0.89**	**0.89**	**0.11**	**0.88**	**0.88**	**0.88**	**0.88**	**0.13**

the total label set and hence it measures each label individually [27]. It reports the average number of times that an example is incorrectly predicted to be related to a class label. It considers the prediction error, i.e. predicting an incorrect label, and missing error, i.e. not predicting a relevant label, normalised over total number of classes and examples [41]. Hamming Loss is interpreted from top bottom, with lower value as better score.

5.2 Findings

We present the highest scoring results from different ngram combinations of ML in addition to our DL methods in Table 4. We applied multiple models and feature extraction strategies for a multi-label approach, and hence, there is a collection of each classifier model that is paired to every method. The results are reported based on our two datasets and their combination.

Results from the App review dataset show that app reviews are notably more random and hard for models to learn. Among ML models, we observed that using [Classifier chains + TF-IDF (analyser = char, ngram = 4,4)] for feature extraction along with employing Logistic Regression, we were able to achieve a relatively high score in most of the performance metrics (Accuracy-0.66, F1 score-0.7278 and Hamming Loss-0.148). Nonetheless, among our DL models, the BERT classifier, performed significantly better than the highest scoring ML model by a marginal difference from its baseline performance (Accuracy 0.8583, F1 Score 0.8583 and Hamming Loss 0.141).

In contrast, models performed significantly better on the Issue Comments dataset as the language of comments was professional and involved more technical terminology. Thus, highest score of a ML model is with a combination of [Classifier chain + TF-IDF (analyser = char, ngram = 4,4)] with base Linear SVM performance (Accuracy-0.786, F1 score-0.797, and Hamming loss-0.104). Overall, DistilBERT has the highest overall performance among all ML/DL methods (Accuracy-0.885, F1 score-0.885, and Hamming loss-0.114).

By combining the datasets to see if increasing the comment count (training size) will improve the performance, model performances were drastically reduced by an average of 3-5% in ML models. Individual ML model performance is not high in all metrics and hence it becomes aimless to consider them. RoBERTa and DistilBert take the lead with higher scores in all metrics and equal Hamming Loss as some other methods. Overall, among the machine learning

methods, the linear SVM model and Logistic regression outperformed in different settings (different datasets) in our test run. We also noted higher performance with the Classifier chain as ML Strategy. Similarly, one best feature extraction technique is TF-IDF with 'char' combinations of ngram set as (4,4) that was observed to deliver the highest scores. We have considered multiple metrics but we prioritise the F1 score as our primary score due to the highly unbalanced nature of our datasets which makes it ideal. As our results indicate, the BERT, RoBERTa and DistilBERT classifiers significantly outperformed the other models for different datasets in multi label text classification for highly imbalanced data by a good margin.

RQ2 Summary. ML and DL algorithms can be used to automatically detect end-user human-centric issues from developer discussions and app reviews. The best results were achieved using BERT on app reviews, DistilBERT on issue comments, and both RoBERTa and DistilBERT classifiers on the combination of the two datasets.

6 Perceived Usefulness of Automatically Classifying End-user Human-centric Issues (RQ3)

6.1 Approach

We performed a survey to investigate the usefulness of the automated classification of human-centric issues from the perspective of software/app developers.

Protocol. We used the guidelines proposed by Kitchenham and Pfleeger [42] to design and execute our survey. The survey introduction included three parts: *Problem Statement*, *Approach*, and *Survey Goal*. In the *Problem Statement* part, we defined the human-centric issue concept and depicted our taxonomy of human-centric issues. As the goal of the survey was not to ask the practitioners to use the automated approaches (discussed in Section 5) in practice before providing the feedback, we introduced the functionality of these approaches to the survey respondents in the *Approach* part. To this end, we showed 8 user app reviews from the Signal Private app that BERT, as one of the best-performing automated approaches, could correctly classify them into Non-Human-centric Issues or Human-centric Issues. We also indicated that the approach classified those reviews labelled as Human-centric Issues into one or more of the following categories of human-centric issues: App Usage, Inclusiveness, and User Reaction. The next item (*Survey Goal*) was to describe the objectives of the survey. The survey was anonymous and hosted on the Qualtrics platform. We obtained ethics approval from the Human Ethics Committee at Monash University before initiating the research.

Our survey included 11 questions and took 5-7 minutes to complete. All questions except one were compulsory. Out of 11 questions, 5 questions sought the background information of the participants (e.g., "*what is your main role in software development?*"). The survey had 6 Likert scale questions and one optional open-ended question to allow the participants to share additional comments on the survey. The Likert scale questions asked the participants to rate the extent they agree or disagree with 6 statements (from "strongly agree = 5" to "strongly disagree = 1"). We also added the "I Don't Know" option to the Likert questions to not force the respondents to answer the statements that they were unsure about or were unclear to them. We leveraged the survey studies [13], [43] used to evaluate the usefulness of (the outputs of) ML/DL-based approaches and tools in the software engineering community to design the following statements.

- Statement 1. "*The tool is useful because issue comments or app reviews with human-centric issues identified by the tool convey meaningful and important information*".
- Statement 2. "*The tool is useful because issue comments or app reviews with human-centric issues identified by the tool can be used to make informed human-centric issues-related design decisions in the future or refine the existing sub-optimum decisions*".
- Statement 3. "*The tool is useful because I, as a practitioner, can find meaningful and important information in a reasonable timeframe from issue comments or app reviews with human-centric issues identified by the tool*".
- Statement 4. "*The tool is useful because issue comments or app reviews with human-centric issues identified by the tool can help us identify human-centric issues faster in mobile apps than if we did it manually*".
- Statement 5. "*The tool is useful because issue comments or app reviews with human-centric issues identified by the tool may contain information that can help us prioritize and resolve such human-centric issues in mobile apps more effectively*".
- Statement 6. "*The tool is useful because issue comments or app reviews with human-centric issues identified by the tool can provide hints and cues to trace forward and backward to codes, services, or features that lead to human-centric issues*".

Participants. We recruited the survey participants by broadly advertising our survey on social networks like Twitter and LinkedIn. In total, we got 16 valid responses. Table 5 outlines the demographic information of the survey respondents including the country they currently work in, the total number of years they have been involved in software development, their main role in software development, the size of their organisation, and the main domain of their organisation. Most of the respondents were working in India (10) followed by Australia (3), Canada (1), China (1) and Angola (1). They mostly had less than two years (10), followed by 3-5 years (4), 6-10 years (1) and 11-15 years (1) of experience in software development. We had four developers, three UI/UX Designers, three software engineers, two project managers, two consultants, one DevOps engineer, and one business analyst among the respondents. Five reported their organisation to have more than 1000, one between 500 and 1000, two to have between 100 and 500, three between 20 and 50, and five with less than 20 employees. Finally, most organisations (10) were in Consulting and IT services domain, two in E-commerce, one in healthcare, one in Finance and two other domains.

Data Analysis. We applied descriptive statistics to analyse the closed-ended questions, i.e., demographic and Likert scale questions.

TABLE 5: Demographics information of the survey respondents

Country	Experience	Main role	Organisation size	Organisation domain
China	0-2 years	DevOps Engineer	More than 1000 employees	Others
Angola	6-10 years	Developer	100 <employees <= 500	E-commerce
Australia	3-5 years	Software Engineer	Less than 20 employees	Consulting and IT services
India	0-2 years	Project Manager	Less than 20 employees	Consulting and IT services
India	0-2 years	UI/UX Designer	More than 1000 employees	Others
India	3-5 years	Project Manager	20 <= employees <= 50	Consulting and IT services
Australia	0-2 years	Business Analysis	20 <= employees <= 50	Consulting and IT services
India	0-2 years	UI/UX Designer	Less than 20 employees	Healthcare
Australia	3-5 years	UI/UX Designer	100 <employees <= 500	Consulting and IT services
India	3-5 years	Developer	20 <= employees <= 50	E-commerce
India	0-2 years	Consultant	Less than 20 employees	Consulting and IT services
India	0-2 years	Software Engineer	More than 1000 employees	Consulting and IT services
India	0-2 years	Consultant	More than 1000 employees	Consulting and IT services
India	0-2 years	Developer	More than 1000 employees	Consulting and IT services
India	0-2 years	Developer	Less than 20 employees	Consulting and IT services
Canada	11-15 years	Software Engineer	500 <employees <= 1000	Financial

TABLE 6: Participants' level of agreement with the statements (in %).

	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree	I Don't Know
Statement 1	18.75	50	31.25	0.0	0.0	0.0
Statement 2	25	43.75	31.25	0.0	0.0	0.0
Statement 3	18.75	68.75	12.5	0.0	0.0	0.0
Statement 4	37.5	37.5	25	0.0	0.0	0.0
Statement 5	25	62.5	12.5	0.0	0.0	0.0
Statement 6	25	31.25	43.75	0.0	0.0	0.0

RQ3 Summary. Practitioners indicate that the automated classification of end-user human-centric issues discussed in issue comments and mobile app reviews has several applications in practice, particularly in enabling practitioners to find meaningful and important information related to human-centric issues in a reasonable timeframe and helping practitioners prioritise and resolve human-centric issues more effectively.

6.2 Findings

Participants' level of agreement to Statements 1-6 is shown in Table 6. Among all the respondents, the majority strongly agreed (18.75%) or agreed (50%) that the tool is useful since the human-centric issues include noteworthy information (Statement 1). Regarding the usefulness of the tool to help make better human-centric issues-related design decisions or refine the existing ones, 25% strongly agreed and 43.75% agreed (Statement 2). 18.75% of the participants strongly agreed, and 68.75% agreed that the tool can help find meaningful and important information in a reasonable timeframe from the human-centric issues detected by the tool (Statement 3). 75% of the respondents strongly agreed/agreed with Statement 4, indicating the human-centric issues detected by the tool can help practitioners identify human-centric issues faster in mobile apps than if they did it manually (Statement 4).

Among the survey respondents, the vast majority of the respondents (87.5%) strongly agreed/agreed with the usefulness of the tool as the human-centric issues detected by the tool contain information that can help them prioritise and resolve such issues in mobile apps more effectively (Statement 5). Finally, 25% of the respondents strongly agreed and 31.5% agreed that the detected issues can provide suggestions to tace to codes, services, or features that lead to human-centric issues (Statement 6). None of the respondents disagreed/strongly disagreed with any of the statements. One developer mentioned that *"These applications are vital to identify issues beforehand, which can be used to avoid such issues any further."*

7 Discussion

There are discrepancies between human-centric issues reported by the users and discussed by the developers; In our data set we found there are almost twice as many human-centric issues reported in the sample 1,200 app reviews (47.25%) compared to the sample 1,200 issue comments (25.5%). Having a lower proportion of human-centric issues discussed in issue comments was foreseeable given the issue comments discuss the possible solution that might be reported through several app reviews. In the App Usage category, both users/developers most frequently mention UI/UX and Bugines issues, followed by users discussing changes/updates, while developers more frequently discuss Privacy/security. Users report many app access issues, but we did not find any discussions between developers to resolve such issues. We found several app reviews mentioning monetary aspects (we classified as Others), but no discussions of these in developers comments. In the Inclusiveness category, developers discuss many language, compatibility and accessibility related issues. Interestingly, such issues are rarely reported by the app users. This could be because developers have already discussed and resolved such issues and therefore, users did not experience many challenges. Finally, in the User Reaction category, users mainly report Preference and Fulfilling interests related issues. While developers also more often discuss Preference issues, they rarely discuss Fulfilling interests issues. These discrepancies can indicate that users face issues that are not being discussed by the developers, and at the same time, if developers discuss and resolve human-centric issues, users do not experience such issues. Therefore, developers need to be aware of the human-centric issues that the users

report, and carefully discuss and resolve them during app development.

Human-centric issues are different across projects; Our findings show that the prevalence of human-centric issues varies across different projects, as illustrated in Tables 2 and 3. In line with the previous discussion point, all the studied projects have less human-centric issues discussed via issue comments than reported by the users. There are projects with both frequent (e.g. Signal private messenger) and rare (e.g. Pixel-dungeon game) human-centric issues discussed by developers and reported by users. In some apps, Firefox browser as the most obvious example, users have very frequently reported human-centric issues (79 out of 100), noting this is just the number of comments containing at least one human-centric issue and some of them may include issues from several categories. However, there are very limited human-centric issues discussed by the developers (30 out of 100). Another examples are K-9 Mail and Bitcoin, the app for buying and using bitcoin and crypto. On the other hand, some apps have relatively frequent discussions of human-centric issues in issue comments but not in app reviews. Cgeo, an app for geocaching, has relatively high occurrence of human-centric issues in issue comments (33 out of 100), but not that of app reviews (29 out of 100). Similarly, Wordpress, the website building app, has 26 human-centric issue comments and only 28 app reviews. The rest of the apps follow more or less the same general trend, and have almost twice app reviews comparing to issue comments. This encourages future research to study how human-centric issues are impacted in different projects or devise guidelines for designing human-centric apps for general users apps (such as Firefox, K-9, Bitcoin).

Human-centric issues can be both technical and non-technical; Some of the categories, such as buginess, UI/UX, and compatibility might be perceived as technical issues, rather than human-centric issues. However, we have only considered an issue, including a technical issue, as a human-centric issue if it directly impacts the end-users of the app. We did not want to limit ourselves to any specific issues or ignore any kind of issues experienced by the end-users of the app if they are being explored as technical issues by other researchers if we believed they were related to the end users' human characteristics. The aim was to be able to build a comprehensive taxonomy that stems from the challenges that the end-users face. This is aligned with what is considered in the work by Ramos et al. [44]. Ramos et al. present a scoping review of 68 studies that employed one or more assessment tools to evaluate a mental health app. They aim to identify the extent to which the existing app evaluation frameworks capture diversity, equity, and inclusion factors. This paper has a different objective from ours, however, has various categories of diversity, equity, and inclusion to assess the app evaluation frameworks. Their diversity, equity, and inclusion criteria were adapted from the Culturally-Informed Design Framework [45]. Ramos et al. provide definitions of extracted diversity, equity, and inclusion variables in three domains: *access*, *content*, and *appearance*. Similar to our categories, their *access* domain covers variables such as Internet Connectivity, Data Usage, Cost, and System Requirements, which seem to be technical but are considered to measure Diversity, Equity, and Inclusion. Considering a comprehensive set of issues faced by the end-users led to the identification of the "Inclusiveness" category in our work that has not been explored in the past. This category can be the focus of other researchers' future work.

There is no structured way of reporting and addressing most human-centric issues; We found that there is currently no structured way for the users to report human-centric issues through app reviews, also found in other work on defect reporting [46]. Moreover, human-centric issues are mostly discussed from a technical perspective by the developers on GitHub issue comments. This suggests a need for a more human-centric issue reporting and follow-up process and tools. Issue reporting systems should include relevant details from not only a technical perspective but also a non-technical end-user understandable point of view. Our future work would allow users to report different human-centric issues, and incorporate such issues in a systematic way during the app development process. There exist some preliminary works on human-centric defect reporting. A recent work by Huynh et al. [47] captured a subset of specific disabilities, e.g., colour blindness, dyslexia, aphasia, hearing impairment, dexterity impairment, and vision impairment, using personas, and further developed mobile and web application prototypes to support defect reporting for this diverse user background. While the work of Huynh et al. [47] provides some preliminary defects reporting support for these end-users, it does not capture our broader objective of reporting a wider range of human-centric issues. Potential guidelines and features for such a tool would include a relatively simple in-app form(s), user tutorials on how to effectively report human-centric defects, form sections for how to reproduce human-centric defects (expected vs actual results), form sections to categorise the type of human-centric issues encountered, and user-reported level of criticality, amongst others. Such reporting tools should, of course, themselves be human-centric and support a diverse range of end-users of the reporting tools. There should also be better ways for the end-users and developers to communicate and become aware of human-centric issues. Our future work would pursue this line of research.

Awareness of human-centric issues can help developers and researchers to more effectively incorporate and report human-centric issues; Limited discussion about key human-centric issues among developers, in spite of the frequent reporting of such issues, reflects that there is still an ongoing challenge that lies in front of the software industry to design more human-centric software and mobile apps. Developers need to be more aware of the human-centric issues of their end-users in order to design more inclusive and human-centred apps and to avoid negative impacts on different end-user groups. Software engineers are typically very different from most end-users - a profession heavily dominated by men; relatively young; affluent; technical; most proficient in English; and while some have physical/mental challenges, these are generally different or of less severity than many users, especially for software targeted to challenged end-users [1], [12]. These influence the degree that developers appreciate and know how to address human-centric issues of their end-users. Training the developers, supporting them by providing required resources, and increasing their general awareness of the human-centric issues could

improve the consideration of these issues during the development process. Results from [48] indicate the importance of accessibility awareness to make app developers becoming ambassadors of accessibility in their organisations.

An automated tool may help detect human-centric issues from app reviews and GitHub issue comments; Our ML/DL models performance and feedback we collected through surveying practitioners suggest that an automated way of detecting human-centric issues can be useful for both users and app developers. Automatically categorising and prioritising app reviews can help developers in different ways. Some examples include suggesting the maintenance tasks developers have to accomplish by extracting the topics and classifying the intention of the reviewer [49], ranking informative reviews in order to support app developers in identifying and prioritising numerous informative reviews [50], and identifying common patterns in order to detect performance bugs for smartphone applications and further support follow-up research on avoiding performance bugs, testing, debugging and analysis for mobile phone applications [51]. Our work would help not only developers but also users to be able to project their issues and challenges to the developers through an automated tool that is able to detect such issues. Developers can more easily compile such issues among a huge pile of reviews they receive on a regular basis and make sure they are aware and can account for the issues reported by the users. Moreover, developers can easily search through the GitHub issue comments to understand what human-centric issues are already discussed in a project when contributing to a new project. Future research is needed to analyse other software (e.g. Jira, StackOverflow) and user-base repositories (e.g. user stories) and apply such smart tools to different sources of data. Furthermore, this can encourage other researchers and practitioners to set some actionable items and guidelines to incorporate human aspects in different software development stages to avoid the occurrence of these issues.

8 Threats to Validity

This section discusses possible threats or limitations of this study and the approaches adopted to mitigate threats [52].

8.1 Internal Validity

Data Collection. The selection of the 12 studied projects, 1200 issue comments, and 1200 app reviews from each project may have introduced threats to our study. First, we decided to use Mazuera-Rozo et al. [16]'s dataset, consisting of 100 Android projects randomly collected from an extensive dataset of GitHub Android projects. We then applied a set of criteria to reach 12 Android projects. This decision was motivated by the fact that it was not possible for us to manually analyse the issue comments and app reviews of all 100 Android projects. Although the 12 projects have different characteristics (e.g., they come from diverse domains), we accept there might be some important Android projects from the human-centric issue perspective that have been omitted. Second, manual analysis of all issue comments and app reviews from these 12 projects was not feasible. Hence, we randomly selected a subset of issue comments and app reviews from each project. We may have missed important developer/end-user discussions on human-centric issues from these projects.

Manual Classification (RQ1). The qualitative analysis used to build the taxonomy of human-centric issues might be subjective and error-prone. To this end, three analysts were involved in the qualitative analysis, and the taxonomy was built in two phases. In each phase, each issue comment/app review was independently analysed and labelled by two persons. Any disagreements between two analysts on labelling issue comments were resolved either by open discussions or involving the third analyst in the discussions. To avoid possible risks and mistakes, when it was not clear to identify the type of human-centric issue from a given issue comment, we labelled it as a non-human-centric issue. Hence, we are confident that our taxonomy of human-centric issues is credible with minimum mistakes.

Survey (RQ3). Completing the survey did not require particular skills from the practitioners. They only had to have some level of software/app development experience. Still, the respondents with poor knowledge of software/app development could threaten the validity of the survey's findings. To partially mitigate this threat, the "I Don't Know" option was added to each Likert scale question.

8.2 External Validity

Manual Classification (RQ1). Two factors can threaten the generalisability of the findings of **RQ1**. Firstly, the 12 selected projects are a small subset of all Android projects hosted on GitHub. Furthermore, our dataset does not include any iOS projects. We acknowledge that our taxonomy of human-centric issues may not be generalised to all different types of GitHub projects (e.g., iOS projects). Secondly, the identified categories of human-centric issues are exclusive to developer discussions on GitHub and end-user reviews on the Google Play Store and are not comprehensive. Hence, analysing other open-source software repositories (e.g., Bitbucket) and software artefacts (e.g., commits, requirement specifications) of proprietary and open-source projects may lead to identifying different and/or a more comprehensive set of human-centric issues categories.

Survey (RQ3). The number of responses (16 responses) to our survey is comparable to other similar surveys (e.g., [14], [15]) used to investigate the usefulness of ML/DL approaches in software engineering. Despite this, the survey findings cannot be generalised to all software/mobile app practitioners, software/mobile app development organisations, and types of apps. Further to this, the majority of the respondents came from India, which is another threat to external validity.

8.3 Construct Validity

Experiments (RQ2). We used several feature extractions techniques with four ML classifiers and developed one DL classifier (See Section 5). We also used four metrics, as precision, recall, accuracy, F1-score, and hamming loss to evaluate ML/DL classifiers. Many other feature selection techniques, ML/DL classifiers, and metrics could be used. However, as argued by Peters et al. [53], it was impossible for us to implement all ML/DL classifiers and use all metrics

in one study. The used feature selection techniques, DL/ML classifiers, and metrics are widely used in automating and classifying software engineering tasks. Despite this, we confirm that the use of other future selection techniques, ML/DL classifiers, and metrics can lead to different results. Finally, there are many data splitting methods [54]. We decided to use 75%-25% to split our dataset into the training and test set in our study. We acknowledge that using other splitting methods may lead to different performance results.

Survey (RQ3). The concept of *human-centric issues* may have different meanings for practitioners. In the survey introduction, we explicitly defined human-centric issues to avoid misunderstanding or multi interpretation of human-centric issues among the survey respondents. In this study, we used a survey to evaluate the usefulness of our automated learning approaches (e.g., BERT) in detecting and classifying human-centric issues from the perspective of software/app developers. While surveys are a common approach to show the usefulness of ML/DL approaches in software engineering, we should emphasise that our survey respondents did not use the developed automated approaches. They only assess the usefulness of the approaches based on their functionality introduced in the survey, along with some examples of app reviews that were correctly detected and classified by BERT as one of the approaches. We are aware that the real drawbacks and practical merits of an ML/DL approach can only be revealed when practitioners use it in practice. Hence, other types of research methodologies, for example, industrial case studies and user studies need to be conducted to explore all drawbacks and practical merits of our ML/DL approaches in practice.

8.4 Reliability

Another threat that might impact this study is that others attempt to replicate it but achieve different results. We constructed a replication package [11] and made it publicly available for those who want to replicate, validate, or extend this study. The replication package includes the dataset used to build the taxonomy, the results of ML and DL approaches, and raw responses to the survey.

9 Related Work

Online repositories, question and answer sites, and issue tracking platforms such as GitHub, StackOverflow, and Jira not only contain rich data discussing technical aspects of the software development process but also include information that provides insight into the **social and human aspects** of the software development process [55]. GitHub has been of considerable interest to software engineering researchers for years [56] due to many open source projects and rich technical and non-technical information to be mined. Many of the projects hosted on GitHub are public, and therefore anyone can view the activities, including actions around issues, pull requests, and commits within those projects.

9.1 Human Aspects in Software Development

Addressing the role of technical proficiency in the software development process, Rocetti et al. compared two approaches in participatory design of a large software artefact involving: 1) novice users, and 2) expert users. Their results show that most of the innovative proposals came from novice users [57]. This shows that designing human-centric software artefacts requires a more participation from novice users, in contrast to the traditional opinion that expert users provide more reliable contribution to the software design process. Rauf et al. analysed a dataset of app developers to examine the rationale behind developers' prioritisation of security in the software development process [58]. The study shows that social considerations, e.g., fear of users, influenced developers' reasoning in development activities, including security choices [58].

Moreover, a human aspect that has been discussed in recent years is the concept of human values (i.e., the guiding principles of what people consider important in life [59]), and its relationship with technology [60], [61]. Whittle et al. argued for the consideration and inclusion of human values at different stages in the software development life cycle [62]. Another study introduced a set of interventions for addressing human values in the SAFe Agile framework [63]. Other related works have proposed tools for supporting human values, e.g., a human values dashboard for software development [64], values Q-sort - an instrument for capturing the values of software engineers [65], and algorithms for detection the violation of values in Android APIs [19].

9.2 Human Aspects in Software Repositories

Pletea et al. focused on security-related discussions on GitHub, as mined from discussions around commits and pull requests [66]. Ko et al. analysed developer design discussions through Bugzilla bug reports to understand the design challenges and how the decisions are made to adapt to user needs [67]. Twidale et al. focused on usability bug reports in Bugzilla [68] while Andreasen et al. explored developers' opinions about usability through surveys, interviews, and mining software repositories [69]. Studies have also mined social aspects in repositories. Dabbish et al. mined GitHub for transparency and collaboration in GitHub projects [70], while Dam et al. mined open-source projects for social norms [71]. Barcellini et al. analysed and visualised social, thematic temporal, and design aspects of online software repositories to understand and model the dynamics of the open source software design process in mailing list exchanges [72].

Some works have focused on mining and classifying specific human aspects of developers in software repositories and issue tracking platforms. Mining more than 2 million issues in Jira from 4 open-source software projects [73], Ortu et al. found a positive correlation between developers' emotions and issue fixing time. Positive emotions resulted in shorter issue-fixing time while negative emotions related to longer issue-fixing time. Cabrera-Diego et al. developed classifiers for comments related to emotions on StackOverflow and Jira. Using features derived from different lexica, their results show significant improvements over the current state of the art in emotion classification [74]. Another study analysed software artefacts for the presence of emotional information in the software development process [75]. Results of an analysis of the Apache Software Foundation issue tracking show that developers do express emotion

while discussing technical issues. Although these studies focus on a specific human aspect (i.e., emotion) from a developer's perspective, they indicate that a rational view of the software development process is insufficient; human aspects such as emotions can negatively or positively affect the development process and be propagated into the resulting software artefact, e.g., happiness, a positive emotion, increases creativity [76], which is good for a successful software design [77].

Khalajzadeh et al. [10] conducted an empirical study of issue comments by extracting and manually analysing 1,691 issue comments from 12 diverse projects, ranging from small to large-scale projects. They categorised the human-centric issues into eight categories of: Inclusiveness, Privacy & Security, Compatibility, Location & Language, Preference, Satisfaction, Emotional Aspects, and Accessibility. However, this work only focuses on developers viewpoint on human-centric issues through analysing GitHub issue comments.

9.3 Human Aspects in App Reviews

Alshayban et al. conducted a large-scale study to understand the state of accessibility in android apps and found that accessibility issues are rife in the 1,000 apps they studied. In some cases, mobile app developers are not educated in accessibility principles and/or are not incentivised by their organisations to make their apps more accessible [48]. Furthermore, a recent study on the reflection of human values in mobile app reviews shows that a quarter of the 22,119 app reviews analysed contain perceived violation of human values in mobile apps, supporting the recommendation for the use of app reviews as a potential source for mining values requirements in software projects [81].

There has been some studies on categorising app reviews from the users' point of view. Khalid et al. [78] studied user reviews from 20 iOS apps and uncovered 12 types of user complaints: App Crashing, Compatibility, Feature Removal, Feature Request, Functional Error, Hidden Cost, Interface Design, Network Problem, Privacy and Ethics, Resource Heavy, Uninteresting Content, Unresponsive App, and Not Specific. McIlroy et al. [79] studied reviews from 20 mobile apps in the Google Play Store and Apple App Store and proposed an approach to automatically assign multiple labels to app reviews. They categorised the app reviews in the following groups: Additional Cost, Functional Complaint, Compatibility Issue, Crashing, Feature Removal, Feature Request, Network Problem, Other, Privacy and Ethical Issue, Resource Heavy, Response Time, Uninteresting Content, Update Issue, User Interface.

Chen et al. [80] identified four categories of user interface related issues: Appearance, Interaction, Experience, and Other generic UI related issues by manually analysing a random sample of 1,447 reviews out of the 3.3M UI-related app reviews. They further categorised these four issue categories into 17 UI-related issue types that users are concerned about in reviews. The Appearance category includes Layout, Legibility and Colour, Typography and Font, Iconography, and Image. The Interaction category includes Navigation, Notification, Motion, Gesture, and Accessibility. The Experience category consists of subcategories as Redundancy, Customisation, Advertisement, and Feedback. Finally, the Others category covers Generic Review, Comparative Review, and Design Specification. All these categories and sub-categories focus on user interface-related issues. Sorbo et al. [49] introduce SURF (Summariser of User Reviews Feedback), which automatically extracts the topics in app reviews, and classifies the intention of the reviewer to suggest the maintenance tasks developers have to accomplish. They categorise the intentions as Information Giving, Information Seeking, Feature Request, Problem Discovery, and Other. They also group together sentences covering the same topic, such as App, GUI, Contents, Pricing, Feature or Functionality, Improvement, Updates/Versions, Resources, Security, Download, Model, and Company.

Genc-Nayebi et al. [83] conducted a systematic literature review to identify the proposed solutions for mining app store user reviews, challenges and unsolved problems, contributions to software requirements evolution and future research directions in the domain. They provided a summary of the extracted app features in a list of mobile app feature extraction studies. Feature request, bug report, compatibility, customer support, updates, user experience, privacy, and resources are the features that are consistent with our findings. Finally, Fazzini et al. [82] conducted an empirical study focusing on app reviews of COVID-19 contact tracing apps. By manually analysing a dataset of 2,611 app reviews, they categorised them into nine categories of Age, Disability, Emotion, Gender, Language, Location, Privacy, Socioeconomic, and Miscellaneous. Even though this work has focused on human aspects, it is limited to the inclusiveness related aspects of the applications, specifically in COVID-19 contact tracing apps. Table 7 shows which of our categories in this paper are covered in the summarised studies.

9.4 Summary

All of the studies discussed in this section focus on different human and social aspects and provide insight into how these aspects are represented in the software development process and repositories. However, none of these works provide an analysis of how human-centric aspects of the end-users are discussed by both end-users and developers in the same projects. In addition, there currently does not exist a comprehensive taxonomy of human-centric issues from both developers and end-users point of view. Our work fills this important gap by providing a broader view perspective of these discussions, with a focus on **end-user human-centric issues**. This is the first work to look into these human-centric issues from both end-users and developers perspective, and also propose an automated way to detect such issues and validate it with real practitioners. In this paper, we developed categories for these human aspects based on a manual analysis of issue comments from different software projects on GitHub and app reviews of the same projects on Google Play Store.

10 Conclusion

Based on a manual analysis of 2,400 app reviews and issue comments from 12 different GitHub repositories, we investigated what human-centric issues are raised by the end-users on Google Play Store and discussed by developers of the same projects on GitHub. We categorised the human-centric issues reported by end-users on Google Play Store app reviews, and discussed by developers in GitHub

TABLE 7: Comparison with other works on categorising app reviews (Y = Yes, N = No, P = Partial)

	#1: App Usage							#1: Inclusiveness				#3: User Reaction		
Category	Resource Usage	Buginess	Change/Update	UI/UX	Privacy/security	Usage instruction	Access issues	Compatibility	Location	Language	Accessibility	Fulfilling interests	Emotional aspects	Preference
Khalid et al. [78]	Y	Y	N	P	Y	N	N	Y	N	N	N	N	N	P
McIlroy et al.[79]	Y	Y	Y	P	Y	N	N	Y	N	N	N	N	N	P
Chen et al. [80]	N	N	N	P	N	N	N	N	N	N	Y	N	N	N
Sorbo et al. [49]	Y	Y	Y	P	Y	N	P	P	N	N	N	N	N	P
Alshayban et al. [48]	N	N	N	N	N	N	N	N	N	N	Y	N	N	N
Obie et al. [81]	N	N	N	N	Y	N	N	N	N	N	N	Y	N	Y
Fazzini et al. [82]	N	N	N	P	Y	N	N	P	Y	Y	Y	N	Y	P

issue comments into three high-level categories: App Usage, Inclusiveness, and User Reaction. We reflected on the fact that there is no standard way of reporting and addressing such human-centric issues on both Google Play Store and GitHub repositories. We also developed ML/DL models to help developers with very different human aspects to many of their end-users to be able to automatically detect such human-centric issues. The results of our ML/DL models in addition to the feedback we received from 16 software/app practitioners supported that our approach can help developers to recognise and appreciate such diverse software end-user human-centric issues more easily. In our future work, we plan to investigate other repositories, question and answer sites, and issue tracking platforms, such as Jira and Stack Overflow. We believe there is a lot of space in further exploring the new "Inclusiveness" category that has emerged from our analysis. It can be the focus of our or other researchers' future work, and our automated tool can be used on larger scale datasets extracted from other repositories to be able to detect the issues related to the inclusiveness category and further explore its sub-categories. We also plan to formulate human-centric requirements to be able to model and incorporate them in different software development stages.

ACKNOWLEDGMENT

Support for this work from ARC Laureate Program FL190100035 and ARC Discovery DP200100020 is gratefully acknowledged.

REFERENCES

[1] J. Grundy, H. Khalajzadeh, and J. Mcintosh, "Towards human-centric model-driven software engineering." in *ENASE*, 2020, pp. 229–238.

[2] K. Hartzel, "How self-efficacy and gender issues affect software adoption and use," *Communications of the ACM*, vol. 46, no. 9, pp. 167–171, 2003.

[3] T. Miller, S. Pedell, A. A. Lopez-Lorca, A. Mendoza, L. Sterling, and A. Keirnan, "Emotion-led modelling for people-oriented requirements engineering: the case study of emergency systems," *Journal of Systems and Software*, vol. 105, pp. 54–71, 2015.

[4] S. E. Stock, D. K. Davies, M. L. Wehmeyer, and S. B. Palmer, "Evaluation of cognitively accessible software to increase independent access to cellphone technology for people with intellectual disability," *Journal of Intellectual Disability Research*, vol. 52, no. 12, pp. 1155–1164, 2008.

[5] S. Wirtz, E.-M. Jakobs, and M. Ziefle, "Age-specific usability issues of software interfaces," in *Proceedings of the IEA*, vol. 17, 2009.

[6] O. Kulyk, R. Kosara, J. Urquiza, and I. Wassink, "Human-centered aspects," in *Human-centered visualization environments*. Springer, 2007, pp. 13–75.

[7] J. Brunet, G. C. Murphy, R. Terra, J. Figueiredo, and D. Serey, "Do developers discuss design?" in *Proceedings of the 11th Working Conference on Mining Software Repositories*, 2014, pp. 340–343.

[8] J. Tsay, L. Dabbish, and J. Herbsleb, "Let's talk about it: evaluating contributions through discussion in github," in *Proceedings of the 22nd ACM SIGSOFT international symposium on foundations of software engineering*, 2014, pp. 144–154.

[9] W. Mo, B. Shen, Y. Chen, and J. Zhu, "Tbil: A tagging-based approach to identity linkage across software communities," in *2015 Asia-Pacific Software Engineering Conference (APSEC)*. IEEE, 2015, pp. 56–63.

[10] H. Khalajzadeh, M. Shahin, H. O. Obie, and J. Grundy, "How are diverse end-user human-centric issues discussed on github?" *arXiv preprint arXiv:2201.05927*, 2022.

[11] H. Khalajzadeh, M. Shahin, H. Obie, P. Agrawal, and J. Grundy, "Supporting Developers in Addressing Human-centric Issues in Mobile Apps [Data set]," 2022. [Online]. Available: https://doi.org/10.5281/zenodo.6982529

[12] J. Grundy, H. Khalajzadeh, T. Kanij, and I. Mueller, "Humanise: Approaches to achieve more human-centric software engineering," in *Evaluation of Novel Approaches to Software Engineering: 15th International Conference, ENASE 2020, Prague, Czech Republic, May 5–6, 2020, Revised Selected Papers*. Springer Nature, p. 444.

[13] A. R. Nasab, M. Shahin, P. Liang, M. E. Basiri, S. A. H. Raviz, H. Khalajzadeh, M. Waseem, and A. Naseri, "Automated identification of security discussions in microservices systems: Industrial surveys and experiments," *Journal of Systems and Software*, vol. 181, p. 111046, 2021.

[14] G. A. A. Prana, C. Treude, F. Thung, T. Atapattu, and D. Lo, "Categorizing the content of github readme files," *Empirical Software Engineering*, vol. 24, no. 3, pp. 1296–1327, 2019.

[15] S. Abualhaija, C. Arora, M. Sabetzadeh, L. C. Briand, and M. Traynor, "Automated demarcation of requirements in textual specifications: a machine learning-based approach," *Empirical Software Engineering*, vol. 25, no. 6, pp. 5454–5497, 2020.

[16] A. Mazuera-Rozo, C. Trubiani, M. Linares-Vásquez, and G. Bavota, "Investigating types and survivability of performance bugs in mobile apps," *Empirical Software Engineering*, vol. 25, no. 3, pp. 1644–1686, 2020.

[17] (2021, July) Open-source android apps project. [Online]. Available: https://github.com/pcqpcq/open-source-android-apps

[18] (2021, July) Open-source ios apps. [Online]. Available: https://github.com/dkhamsing/open-source-ios-apps

[19] C. Li, H. O. Obie, and H. Khalajzadeh, "A first step towards detecting values-violating defects in android apis," 2021.

[20] B. G. Glaser and A. L. Strauss, *The discovery of grounded theory: Strategies for qualitative research*. Routledge, 2017.

[21] N. Humbatova, G. Jahangirova, G. Bavota, V. Riccio, A. Stocco, and P. Tonella, "Taxonomy of real faults in deep learning systems," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, 2020, pp. 1110–1121.

[22] J.-Y. Mao, K. Vredenburg, P. W. Smith, and T. Carey, "The state of user-centered design practice," *Communications of the ACM*, vol. 48, no. 3, pp. 105–109, 2005.

[23] S. Gupta and A. Gupta, "Dealing with noise problem in machine learning data-sets: A systematic review," *Procedia Computer Science*, vol. 161, pp. 466–474, 2019.

[24] GeeksforGeeks, "Snowball stemmer – nlp," [2020]. [Online]. Available: https://www.geeksforgeeks.org/snowball-stemmer-nlp/

[25] S. Paul, "A detailed case study on multi-label classification with machine learning algorithms and predicting movie tags based on plot summaries!" 2019. [Online]. Available: https://medium.com/@saugata.paul1010/a-detailed-case-study-on-multi-label-classification-with-machine-learning-algorithms-and-72031742c9aa

[26] D. S. S. Exchange, "Word2vec embeddings with tf-idf," [2018]. [Online]. Available: https://datascience.stackexchange.com/questions/28598/word2vec-embeddings-with-tf-idf

[27] C. Prathibhamol, K. Jyothy, and B. Noora, "Multi label classification based on logistic regression (mlc-lr)," in *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE, 2016, pp. 2708–2712.

[28] K. P. Murphy, *Probabilistic machine learning: an introduction*. MIT press, 2022.

[29] J. Read, B. Pfahringer, G. Holmes, and E. Frank, "Classifier chains for multi-label classification," *Machine learning*, vol. 85, no. 3, pp. 333–359, 2011.

[30] H. Liu, S. Zhang, and X. Wu, "Mlslr: Multilabel learning via sparse logistic regression," *Information Sciences*, vol. 281, pp. 310–320, 2014.

[31] T. Li, C. Zhang, and S. Zhu, "Empirical studies on multi-label classification," in *2006 18th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'06)*. IEEE, 2006, pp. 86–92.

[32] S. Kouchaki, Y. Yang, A. Lachapelle, T. M. Walker, A. S. Walker, T. E. Peto, D. W. Crook, D. A. Clifton, C. Consortium *et al.*, "Multi-label random forest model for tuberculosis drug resistance classification and mutation ranking," *Frontiers in microbiology*, vol. 11, p. 667, 2020.

[33] M. Chen, Q. Liu, S. Chen, Y. Liu, C.-H. Zhang, and R. Liu, "Xgboost-based algorithm interpretation and application on post-fault transient stability status prediction of power system," *IEEE Access*, vol. 7, pp. 13 149–13 158, 2019.

[34] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[35] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[36] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," *arXiv preprint arXiv:1910.01108*, 2019.

[37] P. Bhargava, A. Drozd, and A. Rogers, "Generalization in nli: Ways (not) to go beyond simple heuristics," *arXiv preprint arXiv:2110.01518*, 2021.

[38] Huggingface, "Preprocessing data," [2021]. [Online]. Available: https://huggingface.co/docs/transformers/preprocessing

[39] C. Mathews, K. Ye, J. Grozdanovski, M. Marinelli, K. Zhong, H. Khalajzadeh, H. Obie, and J. Grundy, "Ah-cid: A tool to automatically detect human-centric issues in app," 2021.

[40] M. S. Sorower, "A literature survey on algorithms for multi-label learning," *Oregon State University, Corvallis*, vol. 18, pp. 1–25, 2010.

[41] P. Jadhav, "Valuation metrics for multi label classification," 2022. [Online]. Available: https://medium.datadriveninvestor.com/a-survey-of-evaluation-metrics-for-multilabel-classification-bb16e8cd41cd

[42] B. A. Kitchenham and S. L. Pfleeger, "Personal opinion surveys," in *Guide to Advanced Empirical Software Engineering*. Springer, 2008, pp. 63–92.

[43] D. N. Palacio, D. McCrystal, K. Moran, C. Bernal-Cárdenas, D. Poshyvanyk, and C. Shenefiel, "Learning to identify security-related issues using convolutional neural networks," in *Proceedings of the 35th IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2019, pp. 140–144.

[44] G. Ramos, C. Ponting, J. P. Labao, and K. Sobowale, "Considerations of diversity, equity, and inclusion in mental health apps: A scoping review of evaluation frameworks," *Behaviour research and therapy*, vol. 147, p. 103990, 2021.

[45] R. S. Valdez, M. C. Gibbons, E. R. Siegel, R. Kukafka, and P. F. Brennan, "Designing consumer health it to enhance usability among different racial and ethnic groups within the united states," *Health and Technology*, vol. 2, no. 4, pp. 225–233, 2012.

[46] N. S. M. Yusop, J. Grundy, and R. Vasa, "Reporting usability defects: A systematic literature review," *IEEE Transactions on Software Engineering*, vol. 43, no. 9, pp. 848–867, 2016.

[47] K. Huynh, J. Benarivo, C. Da Xuan, G. G. Sharma, J. Kang, A. Madugalla, and J. Grundy, "Improving human-centric software defect evaluation, reporting, and fixing," in *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*. IEEE, 2021, pp. 408–417.

[48] A. Alshayban, I. Ahmed, and S. Malek, "Accessibility issues in android apps: State of affairs, sentiments, and ways forward," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, ser. ICSE '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 1323–1334. [Online]. Available: https://doi.org/10.1145/3377811.3380392

[49] A. Di Sorbo, S. Panichella, C. V. Alexandru, J. Shimagaki, C. A. Visaggio, G. Canfora, and H. C. Gall, "What would users change in my app? summarizing app reviews for recommending software changes," in *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, 2016, pp. 499–510.

[50] S. Malgaonkar, S. A. Licorish, and B. T. R. Savarimuthu, "Prioritizing user concerns in app reviews–a study of requests for new features, enhancements and bug fixes," *Information and Software Technology*, vol. 144, p. 106798, 2022.

[51] Y. Liu, C. Xu, and S.-C. Cheung, "Characterizing and detecting performance bugs for smartphone applications," in *Proceedings of the 36th international conference on software engineering*, 2014, pp. 1013–1024.

[52] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in software engineering*. Springer Science & Business Media, 2012.

[53] F. Peters, T. T. Tun, Y. Yu, and B. Nuseibeh, "Text filtering and ranking for security bug report prediction," *IEEE Transactions on Software Engineering*, vol. 45, no. 6, pp. 615–631, 2017.

[54] Y. Xu and R. Goodacre, "On splitting training and validation set: a comparative study of cross-validation, bootstrap and systematic sampling for estimating the generalization performance of supervised learning," *Journal of analysis and testing*, vol. 2, no. 3, pp. 249–262, 2018.

[55] M. Ortu, A. Murgia, G. Destefanis, P. Tourani, R. Tonelli, M. Marchesi, and B. Adams, "The emotional side of software developers in jira," in *Proceedings of the 13th International Conference on Mining Software Repositories*, ser. MSR '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 480–483. [Online]. Available: https://doi.org/10.1145/2901739.2903505

[56] E. Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D. M. German, and D. Damian, "The promises and perils of mining github," in *Proceedings of the 11th working conference on mining software repositories*, 2014, pp. 92–101.

[57] M. Roccetti, C. Prandi, S. Mirri, and P. Salomoni, "Designing human-centric software artifacts with future users: a case study," *Human-centric Computing and Information Sciences*, vol. 10, pp. 1–17, 2020.

[58] I. Rauf, D. van der Linden, M. Levine, J. Towse, B. Nuseibeh, and A. Rashid, "Security but not for security's sake: The impact of social considerations on app developers' choices," in *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, ser. ICSEW'20. New York, NY, USA: Association for Computing Machinery, 2020, p. 141–144. [Online]. Available: https://doi.org/10.1145/3387940.3392230

[59] A.-S. Cheng and K. R. Fleischmann, "Developing a meta-inventory of human values," in *ASIS&T*, vol. 47, 2010.

[60] J. Grundy, I. Mueller, A. Madugalla, H. Khalajzadeh, H. O. Obie, J. McIntosh, and T. Kanij, "Addressing the influence of end user human aspects on software engineering," in *Evaluation of Novel Approaches to Software Engineering*, R. Ali, H. Kaindl, and L. A. Maciaszek, Eds. Cham: Springer International Publishing, 2022, pp. 241–264.

[61] W. Hussain, H. Perera, J. Whittle, A. Nurwidyantoro, R. Hoda, R. A. Shams, and G. Oliver, "Human values in software engi-

neering: Contrasting case studies of practice," *IEEE Transactions on Software Engineering*, pp. 1–15, 2020.
[62] J. Whittle, M. A. Ferrario, W. Simm, and W. Hussain, "A case for human values in software engineering," *IEEE Software*, vol. 38, no. 1, pp. 106–113, 2021.
[63] W. Hussain, M. Shahin, R. Hoda, J. Whittle, H. Perera, A. Nurwidyantoro, R. A. Shams, and G. Oliver, "How can human values be addressed in agilemethods a case study on safe," *IEEE Transactions on Software Engineering*, pp. 1–1, 2022.
[64] A. Nurwidyantoro, M. Shahin, M. Chaudron, W. Hussain, H. Perera, R. A. Shams, and J. Whittle, *Towards a Human Values Dashboard for Software Development: An Exploratory Study*. New York, NY, USA: Association for Computing Machinery, 2021. [Online]. Available: https://doi.org/10.1145/3475716.3475770
[65] E. Winter, S. Forshaw, and M. A. Ferrario, "Measuring human values in software engineering," in *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, 2018, pp. 1–4.
[66] D. Pletea, B. Vasilescu, and A. Serebrenik, "Security and emotion: sentiment analysis of security discussions on github," in *Proceedings of the 11th working conference on mining software repositories*, 2014, pp. 348–351.
[67] A. J. Ko and P. K. Chilana, "Design, discussion, and dissent in open bug reports," in *Proceedings of the 2011 iConference*, 2011, pp. 106–113.
[68] M. B. Twidale and D. M. Nichols, "Exploring usability discussions in open source development," in *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*. IEEE, 2005, pp. 198c–198c.
[69] M. S. Andreasen, H. V. Nielsen, S. O. Schrøder, and J. Stage, "Usability in open source software development: opinions and practice," *Information technology and control*, vol. 35, no. 3, 2006.
[70] L. Dabbish, C. Stuart, J. Tsay, and J. Herbsleb, "Social coding in github: Transparency and collaboration in an open software repository," in *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*, ser. CSCW '12. New York, NY, USA: Association for Computing Machinery, 2012, p. 1277–1286. [Online]. Available: https://doi.org/10.1145/2145204.2145396
[71] H. K. Dam, B. T. R. Savarimuthu, D. Avery, and A. Ghose, "Mining software repositories for social norms," in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, vol. 2, 2015, pp. 627–630.
[72] F. Barcellini, F. Détienne, J.-M. Burkhardt, and W. Sack, "A socio-cognitive analysis of online design discussions in an open source software community," *Interacting with computers*, vol. 20, no. 1, pp. 141–165, 2008.
[73] M. Ortu, G. Destefanis, B. Adams, A. Murgia, M. Marchesi, and R. Tonelli, "The jira repository dataset: Understanding social aspects of software development," in *Proceedings of the 11th International Conference on Predictive Models and Data Analytics in Software Engineering*, ser. PROMISE '15. New York, NY, USA: Association for Computing Machinery, 2015. [Online]. Available: https://doi.org/10.1145/2810146.2810147
[74] L. A. Cabrera-Diego, N. Bessis, and I. Korkontzelos, "Classifying emotions in stack overflow and jira using a multi-label approach," *Knowledge-Based Systems*, vol. 195, p. 105633, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0950705120300939
[75] A. Murgia, P. Tourani, B. Adams, and M. Ortu, "Do developers feel emotions? an exploratory analysis of emotions in software artifacts," in *Proceedings of the 11th Working Conference on Mining Software Repositories*, ser. MSR 2014. New York, NY, USA: Association for Computing Machinery, 2014, p. 262–271. [Online]. Available: https://doi.org/10.1145/2597073.2597086
[76] B. Fredrickson, "The role of positive emotions in positive psychology. the broaden-and-build theory of positive emotions." *The American psychologist*, vol. 56, no. 3, pp. 218–26, 2001.
[77] F. P. Brooks, "No silver bullet essence and accidents of software engineering," *Computer*, vol. 20, no. 4, p. 10–19, Apr. 1987. [Online]. Available: https://doi.org/10.1109/MC.1987.1663532
[78] H. Khalid, E. Shihab, M. Nagappan, and A. E. Hassan, "What do mobile app users complain about?" *IEEE software*, vol. 32, no. 3, pp. 70–77, 2014.
[79] S. McIlroy, N. Ali, H. Khalid, and A. E Hassan, "Analyzing and automatically labelling the types of user issues that are raised in mobile app reviews," *Empirical Software Engineering*, vol. 21, no. 3, pp. 1067–1106, 2016.
[80] Q. Chen, C. Chen, S. Hassan, Z. Xing, X. Xia, and A. E. Hassan, "How should i improve the ui of my app? a study of user reviews of popular apps in the google play," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 30, no. 3, pp. 1–38, 2021.
[81] H. O. Obie, W. Hussain, X. Xia, J. Grundy, L. Li, B. Turhan, J. Whittle, and M. Shahin, "A first look at human values-violation in app reviews," in *ICSE-SEIS*, 2021.
[82] M. Fazzini, H. Khalajzadeh, O. Haggag, Z. Li, H. Obie, C. Arora, W. Hussain, and J. Grundy, "Characterizing human aspects in reviews of covid-19 apps," in *9th IEEE/ACM International Conference on Mobile Software Engineering and Systems*, 2022.
[83] N. Genc-Nayebi and A. Abran, "A systematic literature review: Opinion mining studies from mobile app store user reviews," *Journal of Systems and Software*, vol. 125, pp. 207–219, 2017.

Hourieh Khalajzadeh is a Senior Lecturer in the School of Information Technology at Deakin University. Previously, she was a Research Fellow in the HumaniSE Lab at Monash University. Hourieh's research is situated at the intersection of software engineering and data science. She is currently looking at the human-centric issues in Software Engineering and is experienced in designing domain specific visual languages for different applications, including big data analytics development. She has received several awards such as ACM Distinguished Paper Award (MobileSoft 2022), Outstanding Reviewer Award (CHASE 2021), Best Paper Award (ENASE 2020), and Best Showpiece Award (VL/HCC 2020). She serves as CHASE 2023 co-chair and ICSE 2023 Schedule Chair.

Mojtaba Shahin is a Lecturer in the School of Computing Technologies at RMIT University, Melbourne. Previously, he was a Research Fellow at Monash University. His research interests reside in Empirical Software Engineering, Human and Social Aspects of Software Engineering, and Secure Software Engineering. He completed his PhD study at the University of Adelaide, Australia.

Humphrey O. Obie is an Adjunct Research Fellow with the HumaniSE Lab in the Faculty of Information Technology, Monash University. He has tackled problems in several domains as a data scientist, software developer, and researcher. His key interests include human-centric software engineering, human-centric IoT and smart cities, value-based software engineering, information visualisation and visual data storytelling. He completed his PhD at Swinburne University of Technology, Australia.

Pragya Agrawal is a second-year Master of Data Science student at Monash University, Australia. She has received a bachelor's degree in Computer and Communication Engineering from Manipal University Jaipur, India. Her current research interests include Data mining and Statistical Analysis.

John Grundy is an Australian Laureate Fellow and a Professor of Software Engineering at Monash University where he heads up the HumaniSE lab in the Faculty of Information Technology. His lab investigates "human-centric" issues in software engineering – these include, but are not limited to, impact of personality on software engineers and users; emotion-oriented requirements engineering and acceptance testing; impact of different languages, cultures and belief sets on using software and engineering software; usability and accessibility of software, particularly for ageing people and people with physical and mental challenges; issues of gender, age, socio-economic status and personal values on software, software requirements, and software engineering; and team and organisational impacts, including team climate. His lab's goal is to improve software engineering practices, tools, and thus the target systems to make better software for people.