Journal of Software Engineering Research and Development, 2024, 12:7, doi: 10.5753/jserd.2024.3620

What Do Flutter Developers Ask About? An Empirical Study on Stack Overflow Posts

Anthony Wambua Wambua [Daystar University, Nairobi, Kenya | awambua@daystar.ac.ke]

Abstract

Since Google launched Flutter, an open-source framework, in 2017, many companies and software developers have turned to its use owing to its cross-platform feature. Other attractive features include hot reloading, a rich widget library, and improved performance compared to other cross-platform frameworks. Despite the rise in the use and adoption of the framework, little has been done to understand developers' challenges. This study aims to understand what Flutter developers post on Stack Overflow – a popular Q&A website for developers. Analyzing such posts would help us understand the challenges faced by Flutter developers. To meet this goal, the study used a topic modeling approach to analyze all "flutter" tagged posts between 2019 and 2023. This study revealed state management, widgets, navigation, packages, and persistence as some areas developers face challenges. Further, the study established that there is a growth in the number of Flutter-related posts and developers. While the Flutter framework is promising for companies and software developers, this study points out areas where Flutter trainers and developers should emphasize. Flutter Framework developers should provide more documentation and support as the language matures.

Keywords—*Flutter, Stack Overflow, LDA*

1. Introduction

Mobile application development has elicited profuse interest from users, trainers, developers, software engineers, and researchers. Mobile applications, commonly known as apps, have become prevalent among users, prompting massive interest in mobile application development among software developers. One serious challenge facing mobile application development is writing applications for both iOS and Android platforms. In many cases, developers must develop native applications for each dominant platform separately. Consequently, the concept of cross-platform software development has gained traction among developers and researchers. One of the dominant frameworks used for cross-platform development is Flutter, which was introduced in 2017 (Stanojević, Šošević, Minović, & Milovanović, 2022).

Many developers have resorted to using Flutter to reduce the development time and cost for different platforms. With Flutter being a newer framework compared to other mature mobile development environments such as Java, developers often face a myriad of challenges during development. Unfortunately, very little is known about these challenges owing to the paucity of research in this area. In their article on trends and challenges for software engineering in the mobile computing domain, Baresi et al. (2020) call on researchers to get out of their "Java-plus-Android" comfort zone and study new technologies, including Flutter. Against this background, this study seeks to heed the call 'to get out of Java-plus-Android comfort zone' in research by exploring challenges faced by Flutter mobile application developers.

Even though studies such as the study by Zohud and Zein (2021) and Amatya and Kurti (2014) have investigated challenges faced by cross-platform developers, they are not specific to the Flutter framework. Flutter makes major deviations from cross-platforms by focusing on widgets as the main components (Rao, Pavan, Srivastava, Amani, & Sharma, 2022). Further, since Flutter uses Dart as the programming language, a language that has not been extensively used and studied, the need to investigate unique challenges Flutter developers could face is paramount and urgent.

Understanding the challenges these developers face is vital for developers of the framework, researchers, practitioners, mobile application developers, and mobile application trainers. Such knowledge can result in proper documentation by the framework developers, development of handy tools by researchers as well focused isolation of topics of focus by Flutter trainers and thus improve the quality of mobile applications and further accelerate the adoption of the Flutter framework.

1.1 Flutter Framework

Flutter is Google's cross-platform open-source SDK for developing high-performance natively built applications for operating systems such as iOS and Android from a single codebase. Cross-platform development promotes the idea of Write Once Run Anywhere (WORA) (Shah, Sinha, & Mishra, 2019).

To cement its place as the go-to solution for cross-platform development, Google introduced Flutter support for other platforms such as the web, Windows, and MacOS. Besides cross-platform development capabilities and being open source, other attractive Flutter features include hot reloading, which allows developers to instantly see the changes they make to the UIs during development, a rich widget library that allows for the development of complex but appealing UIs, and improved performance by use of Ahead Of Time (AOT) compilation that speeds up app start-up. In a comparative study between Xamarin, Reactive Native, and Flutter cross-platforms, the study by Nawrocki, Wrona, Marczak, and Sniezynski (2021) found Flutter to be the best overall. The study focused on application size, start-up time, and RAM usage in both Android and iOS platforms. In the study, a native Android application was compared to applications developed with Flutter, Xamarin, and Reactive Native. Surprisingly, the Flutter-made application completed a task faster than all - even the native application, with less than 75% less CPU load than the native application.

This study attempts to understand the challenges faced by Flutter developers by analyzing what these developers ask Stack Overflow. Stack Overflow is a go-to Q&A website for developers and researchers for communication on challenges, issues, and discussion of software development across various technologies. Other studies have utilized data from Stack Overflow to answer research questions related to Software Engineering. For example, Uddin and Khomh (2021) developed a tool called "Opiner" that utilizes data from Stack Overflow to group all opinions about different APIs given the name. The opinions cover a range of issues, such as performance and usability. Abdellatif, Costa, Badran, Abdalkareem, and Shihab (2020) carried out a study that used data from Stack Overflow to understand the challenges faced by Chatbot developers. The study applied topic modeling and found that among the most challenging aspects of Chatbot development were Chatbot integration and training. Montandon et al. (2021) analyzed Stack Overflow job postings to understand the hard and soft skills IT companies were interested in when hiring developers. The study established that amongst the hard skills, developers were expected to have a great mastery of programming languages, while soft skills such as communication, collaboration, and problem-solving were identified as the top demanded skills. Lastly, Tahaei, Vaniea, and Saphra (2020) analyzed Stack Overflow posts to understand the confusion developers face when dealing with privacy-related questions. The study identified topics from the questions and qualitatively analyzed a random sample of the Stack Overflow dataset. Interestingly, that study found that only 28% of the accepted responses to the questions posted by developers and responses had links to the official documentation. Evidently, Stack Overflow has proven to be researchers' favorite for understanding developers' behavior and finding responses to key Software Engineering Questions.

1.2 Latent Dirichlet Allocation (LDA)

The enormous growth of information sources has led to the availability of a mammoth text corpus. When we need to make sense of such voluminous texts, certain information retrieval tools are needed. Amongst the many approaches, topic modelling has been used to reduce a huge collection of documents into topical subspaces (Chauhan & Shah, 2021). Topic modelling offers a way to present large data in low dimensionality by revealing hidden and cardinal features using probabilistic and non-probabilistic approaches (Chauhan & Shah, 2021). One of the common topical modelling techniques is Latent Dirichlet Allocation. LDA can help in understanding unstructured data written in a conversational manner, such as the posts made by users in Q&A forums.

First introduced by (Blei, Ng, & Jordan, 2003), LDA is an unsupervised generative probabilistic model for modeling a corpus. LDA aims to uncover latent topics in a collection of documents. The technique is premised on the fact that each document has a mixture of topics and that each word in a document is attributed to one of the topics. The algorithm, therefore, applies probabilistic computation in determining under which topic a certain word or term is attributed. The process, therefore, starts by determining the number of topics in which all the terms in the document should be placed. The number of topics can be determined manually or by an algorithm. An optimal number of topics is a key determinant in the success of LDA (Hasan, Rahman, Karim, Khan, & Islam, 2021).

1.3 Stack Overflow

Stack Overflow is one of the many question-answer websites for programmers previously owned by Stack Exchange Network. Stack Overflow has since been acquired by Prosus as of June 2021 (Contributors, 2024). Named after a common error that occurs in programming when call stack pointer exceeds the stack bound, the website has, since its launch in 2008, risen in popularity. The website has over 10 million contributors and, as such, has accumulated a large corpus of software engineering knowledge. By August 2023, the website had over 23 million questions and 35 million answers; this makes it the most popular Q&A forum.

In Stack Overflow, a user (questioner), mainly a programmer, can post a question. Other users (answerers) can respond to the question by attempting to provide the correct answers or by posting comments on the question or other users' posts. If the questioner is satisfied with the answers provided, they can mark that answer as accepted (Choetkiertikul, Avery, Dam, Tran, & Ghose, 2015). The website is designed so that other community members are involved in the discussion, even if they do not ask or offer an answer. Users can upvote or downvote a question or an answer. Users upvote questions or answers that they find very helpful and can downvote questions or answers they find unhelpful or wrong. As users create questions, they must use at least one tag and at most five tags based on the question's subject area for easy discovery by other users.

Further, to keep the community engaged in giving meaningful answers, Stack Overflow uses the reputation feature. A user's reputation can increase or decrease based on the upvotes or downvotes their questions or answers get from the community. Additionally, questions have views indicating the number of times users have visited the questions. Questions with a higher view score indicate a higher interest from the community.

We analyze Stack Overflow posts related to Flutter development and apply the Latent Dirichlet Allocation (LDA) topic modeling approach and other statistical procedures to understand the problems that Flutter developers face. Specifically, this study will focus on answering the following research questions:

RQ1: What topics do Flutter developers ask on SO?

From the thousands of questions, we seek to establish specific topics under which these questions belong. These areas represent areas of challenge in Flutter development. The identified topics can be prioritized in Flutter development, documentation, and training.

RQ2: What questions are top voted and top viewed?

In Stack Overflow, most voted and viewed questions represent the community's interest. As explained, users can upvote questions they deem important and downvote questions they consider irrelevant. Further, as developers visit a certain question, its "views" score increases. The question with the highest score thus represents an issue most developers face and thus the need to check out the suggested solutions. We seek to identify these questions as they represent the Flutter community's interest and areas where urgent attention is needed to solve challenges faced by developers.

RO3: How has the interest in Flutter been over the last five years?

We seek to establish the number of questions related to Flutter development over 2019-2023. Further, we want to establish the number of questioners in the same period. The period of five years is assumed to be sufficient to portray a trend in terms of the popularity of the Flutter framework.

This study leads to the following primary contributions:

a) We carry out empirical research to understand the challenges faced by Flutter developers. From our survey, we have not seen any prior study that focused on Flutter developers at a large scale.

b) We offer insights concerning challenges Flutter developers face that can guide Flutter developers, Flutter framework developers, and Flutter trainers.

c) We make recommendations regarding Flutter software development that are helpful to educators and developers.

The rest of the study is organized as follows: Section II presents related work. Section III describes the methodology adopted in this study. Section IV reports our findings, while Section V discusses the meaning and implications of our findings. Section VI discusses threats to the study, while Section VII reflects on the study, suggests areas of future work and offers recommendations to Flutter framework developers, practitioners, and trainers.

2. Related Work

Since its launch in 2008, Stack Overflow has become an indispensable Q&A website for researchers studying Software Engineering practices, developer behavior, and specific aspects of software development such as programming languages, error fixing, and APIs. Numerous studies have been conducted to characterize different aspects of the website. The first study involving the analysis of Stack Overflow data was carried out by Tahaei et al. (2020), who sought to categorize questions posted on the website and determine questions that had been well answered and those that had not been answered. Since then, many other studies have been carried out. We adopt a thematic approach in looking at the related work.

2.1 Developers' behavior studies

Zolduoarrati, Licorish, and Stanger (2022) researched whether collective and cultural behaviors impacted a developer's work, analyzed Stack Overflow posts, and revealed interesting conclusions. The study examined US, Chinese, and Russian developers and used content analysis to compare orientation, attitudes, and knowledge-sharing patterns. The study concluded that US developers, predominantly individualistic in their culture, had higher reputations, used the pronoun "I", and were more task-focused, while Chinese developers deemed to have a more collective culture used the pronouns “we” and “you.”

While studying developers’ reading behaviour, Saddler et al. (2020) used an eye-tracking approach to investigate how visual attractors and the quantity of a post’s content on Stack Overflow affected the duration a developer looked at a post. The study established that average-sized posts and code snippets held developers’ gaze the most, while posts with increased plain text were unattractive and held developers’ attention the least. These findings are critical in helping developers package their posts on Stack Overflow.

2.2 Programming languages & frameworks studies

Şahin and BAYAZIT (2020) carried out an empirical study to understand what Java developers talked about. The study employed LDA for topic modeling. The study established that topics such as JVM operations, GUI development, and web development dominated discussions of Java developers on Stack Overflow.

Chakraborty, Shahriyar, Iqbal, and Uddin (2021) conducted a study to understand developer discussions and support for newer programming languages. The study applied LDA to analyze Stack Overflow posts on Swift, Go, and Rust. The authors wanted to understand support for languages that came up after the establishment of Stack Overflow in 2008 as opposed to more mature languages such as C, C++, and Java that have been in use even before Stack Overflow launched. The study revealed that the difficult topics for newer languages included migration and data structures.

2.3 Mobile application development studies

Closer to the current study, in the domain of mobile application development, Stack Overflow has been used in numerous studies to investigate different aspects. Rahman and Roy (2022) analyzed Stack Overflow posts to study the impact of code fragment reuse among mobile application developers on software bugs and maintenance. The research established that the reuse of Stack Overflow code snippets in mobile application development was more rampant in industrial mobile apps than in open-source mobile application projects. Further, the study found that reusing Stack Overflow code snippets was responsible for bug occurrence in future project revisions.

Ahmad, Feng, Li, Asim, and Sun (2019) sought to understand the non-functional requirements (NFR) of iOS developers by analyzing posts from Stack Overflow. The study used the LDA topic modeling approach and established that the predominant topics in the discussion were reliability and functionality. The study focused on iOS developers and did not address Android developers.

2.4 The current study

Even though the related studies surveyed address specific areas of software engineering ranging from developer behavior to programming languages and even mobile applications, none addressed Flutter developers or Dart developers. The current study will analyze posts from Stack Overflow made by Flutter developers to understand what these developers talk about. This will help harness the unique challenges that these developers face.

3. Methodology

This section describes the methodology used in the study. The study’s main goal is to examine the questions Flutter developers post on Stack Overflow. To meet this goal, we analyze a dataset of questions from Stack Overflow.

3.1 Data source

Data utilized in this study was extracted from Stack Overflow through the Stack Exchange Data Explorer, which can be found here: https://data.stackexchange.com/. This tool allows researchers to extract data from any website owned by the Stack Overflow parent company. The Data Explorer tool allows one to issue queries and get responses. For example, one can query the most viewed question on the Stack Overflow website using the appropriate query language. Understanding the website’s database schema, including the different tables and the type of posts and their attributes, is required for proper querying. Stack Overflow was chosen due to its popularity amongst the developer community, its huge dataset, extensive usage in software engineering research, and the existence of the Stack Exchange Data Explorer tool that makes it easy to query for specific data.

3.2 Procedure

Figure 1 summarizes the procedure, while the following steps describe actions taken to meet the study’s research questions.

Step 1: *Data extraction.* On 6th July 2023, data was extracted from Stack Exchange consisting of all posts with the tag *“flutter”* that were posted between 2019 and 2023. The period of five years was deemed to be sufficient to generate data that would help in answering the research questions. Since queries have a processing time slot in the Stack Exchange, data for each year were extracted separately and then merged into a single .csv file. As such, queries did not timeout, and all the data for the period under study was extracted. This process yielded 157,735 records.

Step 2: *Identify Flutter topics*. To identify topics that dominate questions asked by Flutter developers between 2019 and 2023, we used the Latent Dirichlet Allocation (LDA) modeling technique. Related studies such as (Tahmooresi, Heydarnoori, & Aghamohammadi, 2020) have applied LDA in analyzing posts extracted from Stack Overflow. A subset of the data consisting of the post id and the title was extracted from the original dataset for topic modeling using R studio. Our review of the "body" column revealed that most developers posted code as part of their posts; as such, including the body column into the model to identify topics would lead to noise and, thus, inaccurate findings (Peruma et al., 2022). Similar studies, including (Khan, Farabi, & Iqbal, 2022), focused on the titles of posts while investigating different aspects utilizing data extracted from Stack Overflow.

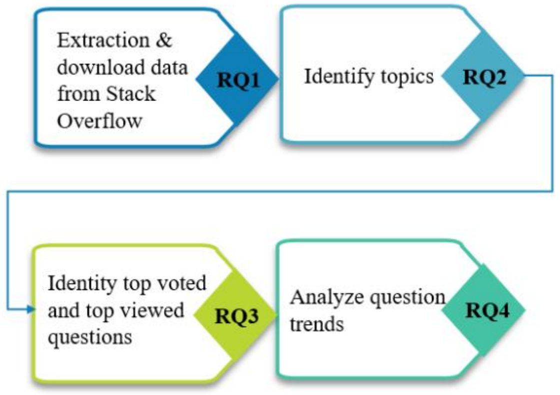

Extraction & download data from Stack Overflow
RQ1
Identify topics
RQ2
Identity top voted and top viewed questions
RQ3
Analyze question trends
RQ4

Figure 1. Research Methodology.

LDA does not identify the actual topics but scans through large datasets to identify words and phrases based on which to cluster the content in groups of words with similar expressions (Kalepalli, Tasneem, Teja, & Manne, 2020). The actual process of topic modeling is depicted in **Figure 2**.

After data for topic modeling was extracted from the data downloaded from Stack Overflow, the new subset was loaded into R studio. Before the topic identification is done, the text is pre-processed by converting all the text to lowercase, removing stop words, punctuations, numbers, and white spaces. Devoid of pre-processing, the identification of topics could have noise. Next is the identification of the number of topics present in the corpus. LDA requires a user to supply the number of topics against which terms will be mapped; this process can be very subjective, and thus, an optimal number of topics should be determined (Gan & Qi, 2021).

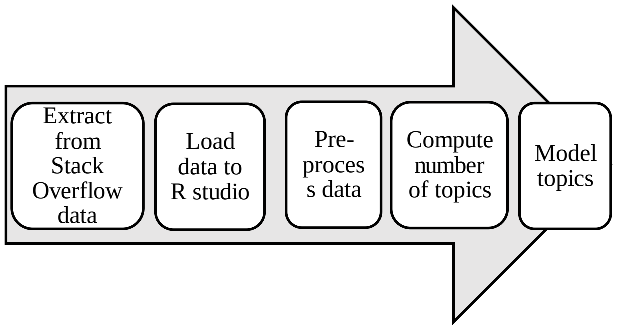


Figure 2. Topic Modelling Process.

An optimal number of topics ensures proper distribution of terms and, thus, non-overlapping of topics. We follow the approach described by (Nikita, 2016) to determine the optimal number of topics in our corpus. We ran through a model that accepts a starting range of a number of topics, and we specified two to ten at increment levels of one. The number of topics identification model uses several metrics; in this study, we used the metrics *CaoJuan2009* and *Griffith2004*. The model yields the optimal number of topics when the value of *Griffith2004* is at the highest and the value of *CaoJuan2009* at the lowest. As illustrated in **Figure 3**, this convergence happened at nine, indicating that the optimal number of topics from our corpus is nine. The metrics *CaoJuan2009* and *Griffith2004* have been used extensively in studies such as Baresi et al. (2020), Sinha et al. (2023), and So, Jang, Kim, and Choi (2023). Other commonly used metrics include *Deveaud2014* and *Arun2010*. So et al. (2023) found *Deveaud2014* to display an unusual decline in the process of computing the optimal number of topics. Lee (2022) found *Deveaud2014* to have low scores in a comparative analysis of the four metrics. In a study by Xu, Lakeh, and Ghaffarzadegan (2021), the authors found that *CaoJuan2009*, *Griffith2004*, and *Arun2010* arrived at a similar number of topics and therefore used these metrics in their study. Given this background, this study used *CaoJuan2009* and *Griffith2004*.

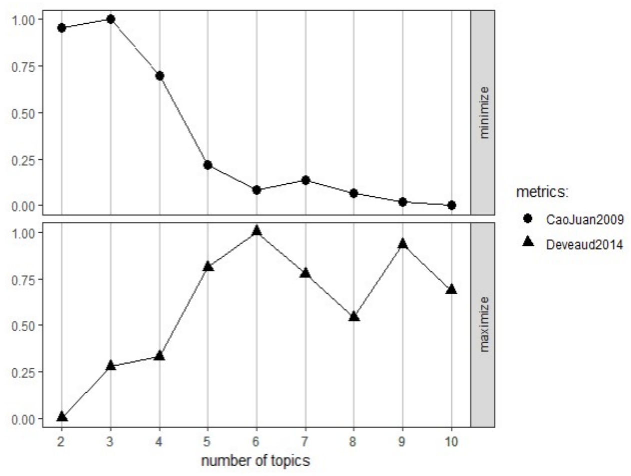


Figure 3. Identification of the optimal number of topics.

Lastly, with the number of topics identified, we applied this to the LDA model for clustering terms to generate the topics. The results are presented in the results section.

Step 3: *Identify top-voted and top-viewed questions*. To understand the interest of Flutter developers in the questions they ask, we identify the top-voted question. This is the question that most developers liked. We also identify the top-viewed question as it is the most likely to have solved most developers' code issues. The entire dataset was uploaded into a MySQL database to answer this research question, and the appropriate queries were used.

Step 4: *Analyze five-year trends*. To identify any increase or decrease among developers in interest for Flutter development, appropriate queries were used to group the posts by year for each year between 2019 and 2023. Further, we calculate the unique users who asked questions related to Flutter in the same period.

4. Results

This section presents the findings of our study by responding to the research questions.

4.1 Topics discussed by Flutter developers on Stack Overflow

To identify the topics discussed by Flutter developers on Stack Overflow, topic modeling was used. After processing 157,735 posts, the terms used in the posts were classified into nine topics. Based on the authors' experience in Flutter software development, the classified terms were used to identify nine topics. Figure 7 depicts how the nine topics were identified from the grouped terms.

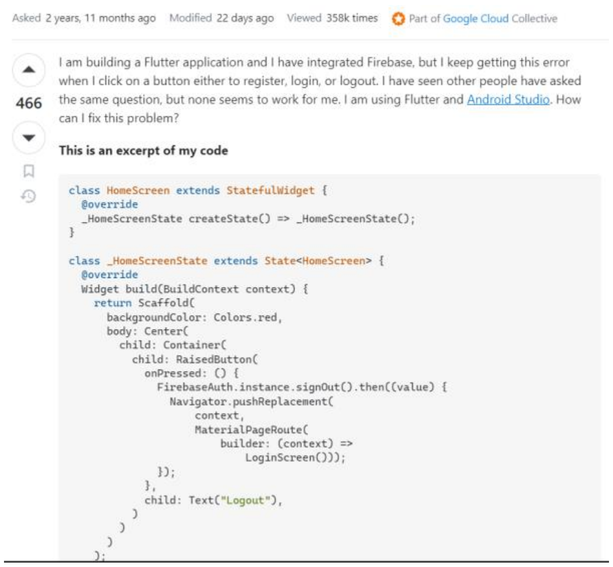
The identified topics are Dart (data) types, list views, functions, error/exception handling, state management, flutter widgets, navigation, packages, and data persistence. These topics are further discussed in the discussion section of this paper.

A similar study carried out by Blanco, Pérez-López, Fdez-Riverola, and Lourenço (2020) to understand the questions Java developers posted on Stack Overflow found revealed topics such as types of files, types of files, tools and Java setup and Java web. In both studies, the question of data types is common, but the other topics are significantly different. One would expect a lot more similarities given that Dart, the language used in the Flutter framework, is Object-Oriented and uses c-type styling just like Java. But there are also significant differences between the two languages; for example, Dart supports null safety, which Java does not. The concept of null safety is major in Dart, and from this study's findings, it is the most viewed question indicating the challenges that Flutter developers may be grappling with.

A study by Yunita (2023) seeking to understand the challenges faced by front-end Javascript revealed some topics similar to those identified in this study. These topics include state management, the use of external libraries (known as packages in Flutter), and the fetching of data. The commonalities can be explained by the fact that Flutter is also used for front-end development and that Dart has similarities with Javascript – it should be remembered that Dart was aimed initially at replacing Javascript.

4.2 Top-voted and top-viewed questions

This study identified the top-voted question, as captured in **Figure 4**, which deals with Firebase integration within a Flutter application. It had a score of 466 and had been viewed 358,000 times by the time the dataset used in this study was extracted.


Asked 2 years, 11 months ago Modified 22 days ago Viewed 358k times Part of Google Cloud Collective

466

I am building a Flutter application and I have integrated Firebase, but I keep getting this error when I click on a button either to register, login, or logout. I have seen other people have asked the same question, but none seems to work for me. I am using Flutter and Android Studio. How can I fix this problem?

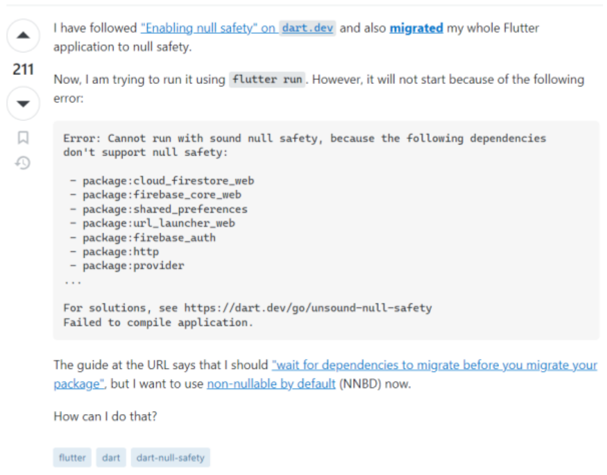
This is an excerpt of my code

```
class HomeScreen extends StatefulWidget {
  @override
  _HomeScreenState createState() => _HomeScreenState();
}

class _HomeScreenState extends State<HomeScreen> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.red,
      body: Center(
        child: Container(
          child: RaisedButton(
            onPressed: () {
              FirebaseAuth.instance.signOut().then((value) {
                Navigator.pushReplacement(
                    context,
                    MaterialPageRoute(
                        builder: (context) =>
                            LoginScreen()));
              });
            },
            child: Text("Logout"),
          )
        )
      )
    );
```

Figure 4. Top-voted Flutter question.

Further, the study established that the most viewed question, as captured in **Figure 5**, dealt with null safety, a major area in the Dart programming language. The question had been viewed 440,000 times by the time the dataset was extracted for this study.


211

I have followed "Enabling null safety" on dart.dev and also **migrated** my whole Flutter application to null safety.

Now, I am trying to run it using `flutter run`. However, it will not start because of the following error:

```
Error: Cannot run with sound null safety, because the following dependencies
don't support null safety:

 - package:cloud_firestore_web
 - package:firebase_core_web
 - package:shared_preferences
 - package:url_launcher_web
 - package:firebase_auth
 - package:http
 - package:provider
...

For solutions, see https://dart.dev/go/unsound-null-safety
Failed to compile application.
```

The guide at the URL says that I should "wait for dependencies to migrate before you migrate your package", but I want to use non-nullable by default (NNBD) now.

How can I do that?

flutter dart dart-null-safety

Figure 5. Top viewed Flutter question.

These two questions can be viewed as the most popular, as votes and views in Stack Overflow represent the community's interest.

4.3 Trends in Stack Overflow Flutter Posts

The sough to establish trends in the number of developers posting "Flutter" tagged questions and the questions themselves. It was assumed that the five-year period would be sufficient to establish a trend concerning the popularity of the framework. The analysis of the "Flutter" tagged posts between 2019 and 2023 shows a steady rise in the number of questions posted and the unique developers posting the questions. **Figure 6** presents the number of posts and developers posting the questions for the period under study. The decline in 2023 is because the dataset was extracted in July 2023. From **Figure 6**, a conclusion can be drawn that each developer posted around two to three questions in the period under study.

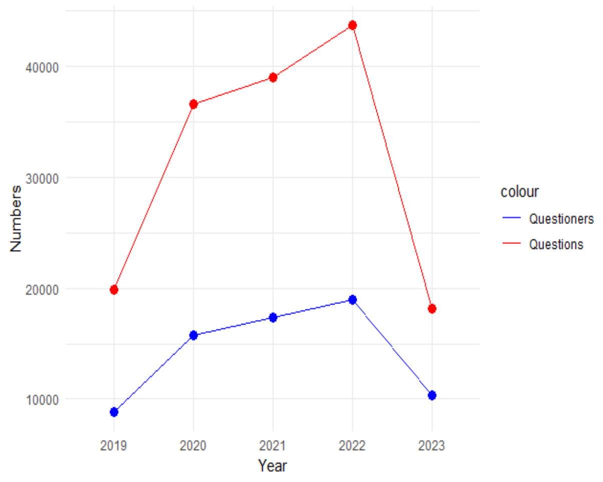


Figure 6. Trends in Stack Overflow Flutter Post.

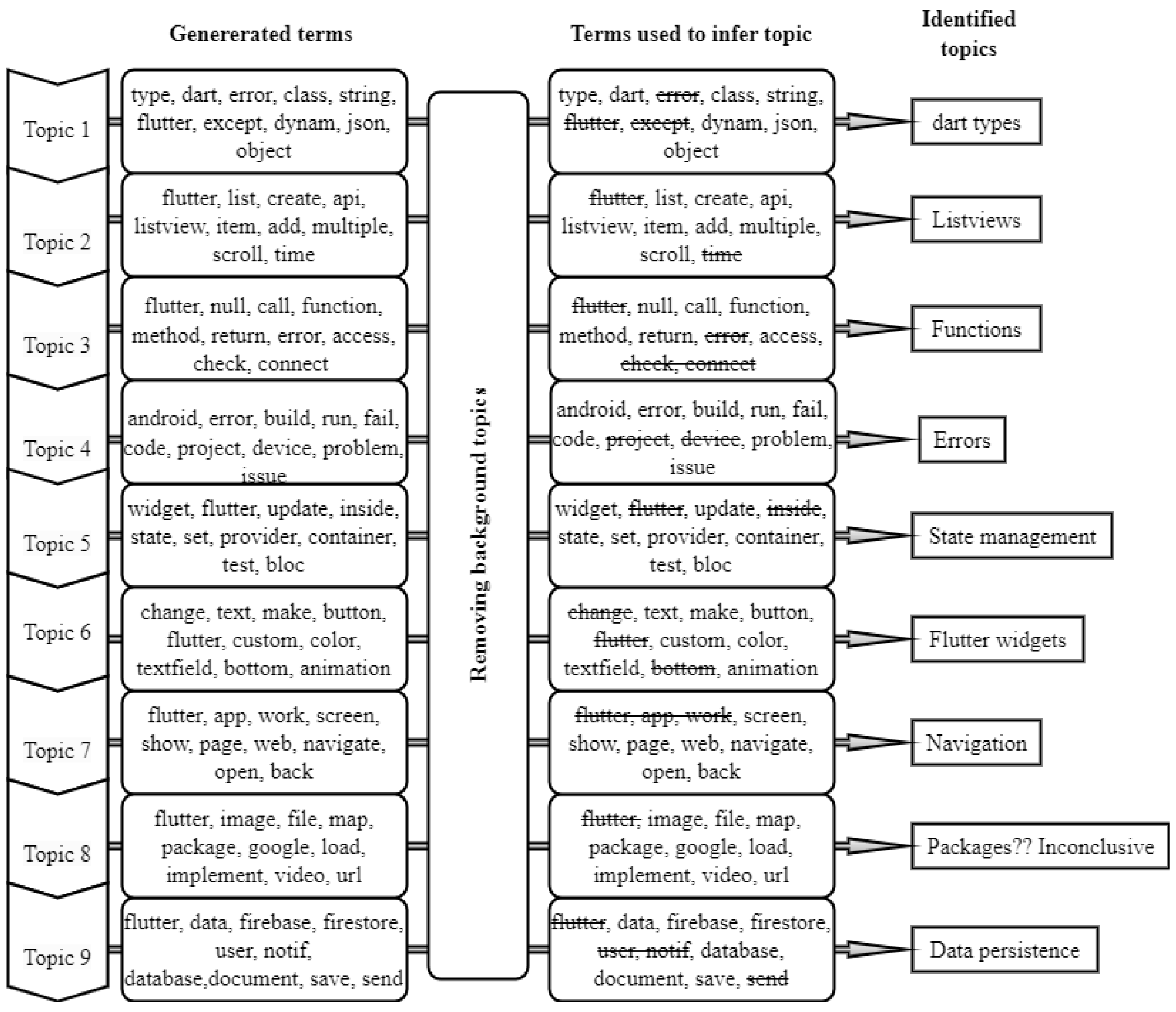

Genererated terms
Terms used to infer topic
Identified topics
Removing background topics
Topic 1
type, dart, error, class, string, flutter, except, dynam, json, object
type, dart, ~~error~~, class, string, ~~flutter~~, ~~except~~, dynam, json, object
dart types
Topic 2
flutter, list, create, api, listview, item, add, multiple, scroll, time
~~flutter~~, list, create, api, listview, item, add, multiple, scroll, ~~time~~
Listviews
Topic 3
flutter, null, call, function, method, return, error, access, check, connect
~~flutter~~, null, call, function, method, return, ~~error~~, access, ~~check, connect~~
Functions
Topic 4
android, error, build, run, fail, code, project, device, problem, issue
android, error, build, run, fail, code, ~~project~~, ~~device~~, problem, issue
Errors
Topic 5
widget, flutter, update, inside, state, set, provider, container, test, bloc
widget, ~~flutter~~, update, ~~inside~~, state, set, provider, container, test, bloc
State management
Topic 6
change, text, make, button, flutter, custom, color, textfield, bottom, animation
~~change~~, text, make, button, ~~flutter~~, custom, color, textfield, ~~bottom~~, animation
Flutter widgets
Topic 7
flutter, app, work, screen, show, page, web, navigate, open, back
~~flutter, app, work~~, screen, show, page, web, navigate, open, back
Navigation
Topic 8
flutter, image, file, map, package, google, load, implement, video, url
~~flutter,~~ image, file, map, package, google, load, implement, video, url
Packages?? Inconclusive
Topic 9
flutter, data, firebase, firestore, user, notif, database,document, save, send
~~flutter~~, data, firebase, firestore, ~~user, notif~~, database, document, save, ~~send~~
Data persistence

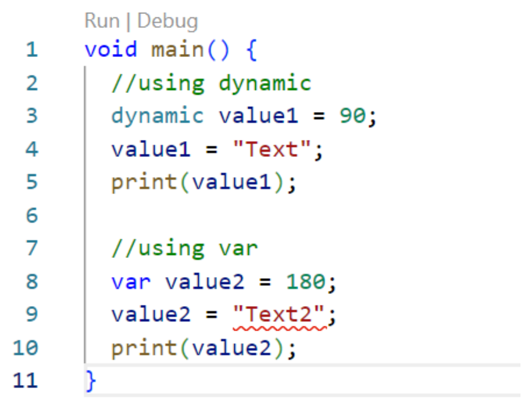
Figure 7. Identification of Flutter topics.

5. Discussion

5.1 Topics discussed by Flutter developers.

The current study sought to understand the discussions and questions asked by Flutter developers. The LDA topic modeling approach was used, and the authors, from their experience in Flutter development, manually inferred topics from the categorized terms. As shown in **Figure 7**, the topics discussed by Flutter developers include the following.

- Dart (data) types: This is a fundamental area in Flutter development. Since the Flutter framework uses Dart language, developers switching from languages such as Java, C, and C++ might find it challenging to use Dart data types. Dart introduces concepts around data types and variables inconsistent with Java, C, and C++. For example, the use of *dynamic* and *var* when declaring a variable, as shown in **Figure 8**. Variables declared with the keyword var can have their values changed but not their types. Variables declared with keyword dynamic can have both their values and types changed. This concept is new to strongly typed languages such as Java. This variation can explain why developers ask and post around these fundamental Dart language concepts.



```
void main() {
  //using dynamic
  dynamic value1 = 90;
  value1 = "Text";
  print(value1);

  //using var
  var value2 = 180;
  value2 = "Text2";
  print(value2);
}
```

Figure 8. Usage of dynamic and var in Dart.

Other additions in Dart that deviate from Java-like declarations that could be driving developers to Stack Overflow include the use of *late* keyword for variables whose value is not known at the time of declaration,

- Listviews: Listview is a widely used widget for large data presentation in Flutter as it allows scrolling. Most mobile applications will utilize listviews in one way or another. Posts around creating listviews, adding items, and populating listviews with data from APIs were found to be common. Flutter offers variations such as ListView.builder, ListView.seperated, and ListView.custom. Determining which variations to use could also create confusion and make developers post about this topic.
- Functions: Dart functions vary significantly from those used in matured languages such as Java, C, and C++. First, the return type of the functions and the data types of the parameters are optional. Further, even the curly braces around a function can be omitted, and the fat arrow operator is used.
  Dart functions can have optional parameters or required parameters. Optional parameters can further be classified as named or positional. Optional parameters can have default values specified so that if a function is called, the variables with default values do not need to have values passed to them unless a need to override such values arises. In Java, C, and C++, parameters are positional. This is a major deviation aimed at making functions in Dart flexible but one that can cause developers switching from Android and Java to flock to Stack Overflow.
- Errors: This topic was identified to have many posts, be it app crashing, exception handling, or troubleshooting. Developers need to be aware of the Flutter concepts around error handling.
- State management: This is an advanced topic in Flutter software development. State management allows the refreshing of widgets when data on the widgets changes during execution. Questions around the implementation of setState and state management packages such as provider, bloc, getx and riverpod underscore the importance of this subject to developers. Flutter trainers can do more to impart knowledge in these areas.
- Flutter widget: Given that everything that can be used to define structural elements in Flutter is a widget, the knowledge of methods and properties of common widgets is critical to Flutter developers.
- Navigation: Moving from one page or screen, commonly known as routes in Flutter, is an area of struggle for many Flutter developers. Developers ought to learn how to use Navigator or packages such as Getx that implement the same functionality.
- Packages: None of the terms placed in this group pointed to Packages as the exact grouping, hence the use of the word inconclusive next to the topic. However, an experienced Flutter developer can easily infer that posts in this category most likely reference the use of third-party packages to implement functionalities described by the terms. Flutter developers have tens of thousands of third-party libraries in the repository https://pub.dev/, which Google engineers and other Flutter community members have developed. Learning how to use these packages can speed up software development time and thus increase a developer's productivity (Ameen & Mohammed, 2022). Interestingly, research has confirmed that developers rarely update third-party libraries and that these libraries can be a source of bugs (Salza, Palomba, Di Nucci, De Lucia, & Ferrucci, 2020). It is paramount that Flutter trainers train upcoming developers on the proper use of packages to improve productivity and how to publish their code as libraries in a way to give back to the Flutter developer community.
- Data persistence: several approaches for data storage exist in Flutter applications. They include shared preferences, SQLite, Firebase, and Firestore. Developers seeking to develop Flutter applications must know each approach's strengths and shortcomings.

The nine topics form the areas where most developers posted or asked questions. This study recommends that Flutter framework developers make detailed and clear documentation on these areas. Further, we recommend that Flutter trainers emphasize these areas during training. Lastly, we recommend that beginner Flutter developers pay attention to the enumerated topics.

5.2 Top-voted and top-viewed posts

Voting in Stack Overflow indicates that other developers found the question helpful. **Figure 4** shows that the top-voted question has 466 votes. Interestingly, the question deals with Firebase, one of the areas that falls under persistence and has been identified as one of the topics talked about by Flutter developers. This underscores the need for developers to be conversant with concepts such as Firebase for storage or authentication.

The top-viewed post, as shown in **Figure 5**, had 440,000 views, depicting enormous interest from developers. This indicated that the issue addressed by the question is a common problem amongst many Flutter developers. The question deals with null safety, a concept under Dart types and fundamentals; unlike languages like Java, where code can crash because of null errors, null safety in Dart forces every variable to have a value or be declared nullable. This shift must be appreciated by developers switching from other languages to Flutter.

5.3 Five-year trends on Flutter posts

As shown in **Figure 6**, the number of Flutter posts and developers posting such questions has continued to increase steadily. This depicts growing interest in the framework given its benefits, such as the ability to write a single code base for multiple platforms such as Windows, Mac, Android, iOS, and the web. Our findings on this research question are consistent with that of authors Chakraborty et al. (2021) and Oliveira, Teixeira, and Ebert (2020), who observed that Stack Overflow posts about a language continued to increase as the language matured. Even though the decline in the number of Flutter-related posts in 2023 is explained by the fact that the data extraction was done in July 2023 before the year ended, one expects the number of posts to be more than half of the previous year. The rise of AI-powered platforms such as ChatGPT can explain this decline. These platforms offer fast, efficient, and personalized programming assistance and are likely to gain more traction than traditional Q&A forums. A study carried out by Liu, Tang, Li, Chen, and Liu (2023) that sought to compare Stack Overflow and ChatGPT in terms of their ability to enhance programmer productivity found that ChatGPT outperformed Stack Overflow in aspects such as task completion speed and the quality of response. Even though Kabir, Udo-Imeh, Kou, and Zhang (2023) in a study aimed at comparing responses from ChatGPT and Stack Overflow found that 52% of the sampled responses were inaccurate, the same study established that developers preferred obtaining programming assistance from ChatGPT due to its comprehensiveness and articulation. It is practical to surmise that a declining trend in the number of developers seeking help from Q&A forums is almost inevitable in the foreseeable future give the rising popularity of Generative AI (GenAI) tools such ChatGPT.

6. Threats to validity

Validity refers to the degree of trustworthiness of a study's findings and the extent to which such results are true and unbiased from the researcher's point of view (Wohlin et al., 2012). A study's validity thus affects its credibility, generalizability, reproducibility, and transferability.

Internal validity refers to the extent to which a researcher can conclude that changes in the dependent variable result from manipulations of the independent variable. As such, this inference guarantees a meaningful interpretation of results (Cahit, 2015). In this study, threats to internal validity could arise since only posts with the tag "flutter" and posted between 2019 and 2023 were retrieved during the dataset extraction. Since developers manually assign tags during posting on Stack Overflow, it is possible that some Flutter-related posts were not properly tagged or some non-flutter posts had the Flutter tag. Such an error can affect the posts being extracted. To mitigate this threat, the extracted posts were randomly checked to verify their relevance to Flutter development. A second area potentially leading to internal validity in this study is determining the number of topics as input to the LDA model. Using an optimal number of topics has an impact on the repeatability of LDA experiments, the isolation of topics, and the reduction of duplicate terms among topics (Gan & Qi, 2021). The number of topics can be manually specified; for example, in the study by Zou, Xu, Yang, Zhang, and Yang (2017) or use default parameters (Treude & Wagner, 2019) including the number of topics. Either way, a level of bias and instability is introduced. To alleviate this threat, the present study applied the approach Nikita (2016) described that automatically discovers the optimal number of topics by using metrics described earlier in the methodology section.

External validity is the degree to which a study's inferences can be generalized to a broader or different target population (Findley, Kikuta, & Denly, 2021). One potential threat is that the data used in the study was all from Stack Overflow. Although Stack Overflow is developers' most popular Q&A forum, other platforms such as Quora, Reddit, CodeProject, and Google Groups exist. It would be interesting to see if our findings hold in these forums. The study used Stack Overflow, given its popularity and extensive use in software engineering research.

7. Conclusion

This paper sought to understand the questions that Flutter developers ask on Stack Overflow. The motivation is that Flutter is a slightly newer cross-platform, and with its rising popularity, it is important to understand the challenges that the developer community faces. This study presented insights from analyzing 157,735 Stack Overflow posts on Flutter software development. LDA, a popular topic modeling approach, was used to analyze the downloaded posts. This was followed by the labeling of topics based on grouped terms. Our findings show that the specific areas within Flutter software development where developers face challenges are Dart (data) types, list views, functions, error/exception handling, state management, flutter widgets, navigation, packages, and data persistence. The study established that the most popular questions, the top-viewed and top-voted, deal with null safety and integration of Firebase.

Additionally, the study noted a steady rise in the number of questions and developers posting the question in the five years under study. In 2023, a slight decline was noted, which led to the discussion on whether the rise of GenAI tools could see traditional Q&A developer forums decline.

In line with the study's findings on the increasing popularity of the Flutter framework established by this study, we recommend that developers transitioning from other frameworks and beginner developers pay attention to the identified areas.

Further, we recommend that Flutter trainers, whether in institutions of higher learning or software training institutes, emphasize the areas identified in this study as they reflect issues that most Flutter developers could grapple with.

Similarly, we recommend that the developers of the Flutter framework offer more support by providing curated documentation in the areas identified by this study and even participate in Q&A websites such as Stack Overflow. Such data-driven decision-making in documentation can significantly help produce more useful documentation. To develop quality software, developers must be supported through their challenges, and proper documentation is critical in this undertaking.

The study opens the stage for academic researchers and practitioners to learn about the experiences of Flutter developers in making data-driven decisions on, for example, studies they would like to undertake, policies and guidelines they would like to put in place, and general awareness of challenges faced by the Flutter developer community.

Possible future work can explore the impact of GenAI tools, such as ChatGPT, on the future of traditional Q&A forums. Anecdotally, the slight decrease in the number of questions in 2023 could point to developers turning more to AI tools than Q&A forums. Given the enormous amount of data they already have, such a study would sensitize these Q&A forums to integrate AI into the platforms.

Stemming from Flutter developers' frustrations revealed in this study, further work can also go into assessing the quality and usage of the technical documentation provided by framework developers and companies that develop software. It would be interesting to see if standards for usability and specification of documentation can be developed and if they would have practical value for documentation quality.

References

Abdellatif, A., Costa, D., Badran, K., Abdalkareem, R., & Shihab, E. (2020). *Challenges in chatbot development: A study of stack overflow posts.* Paper presented at the Proceedings of the 17th international conference on mining software repositories.

Ahmad, A., Feng, C., Li, K., Asim, S. M., & Sun, T. (2019). Toward Empirically Investigating Non-Functional Requirements of iOS Developers on Stack Overflow. *IEEE Access, 7*, 61145-61169. doi:10.1109/ACCESS.2019.2914429

Amatya, S., & Kurti, A. (2014). Cross-platform mobile development: challenges and opportunities. *ICT Innovations 2013: ICT Innovations and Education*, 219-229.

Ameen, S. Y., & Mohammed, D. Y. (2022). Developing cross-platform library using flutter. *European Journal of Engineering and Technology Research, 7*(2), 18-21.

Baresi, L., Griswold, W. G., Lewis, G. A., Autili, M., Malavolta, I., & Julien, C. (2020). Trends and challenges for software engineering in the mobile domain. *IEEE Software, 38*(1), 88-96.

Blanco, G., Pérez-López, R., Fdez-Riverola, F., & Lourenço, A. M. G. (2020). Understanding the social evolution of the Java community in Stack Overflow: A 10-year study of developer interactions. *Future Generation Computer Systems, 105*, 446-454.

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of machine Learning research, 3*(Jan), 993-1022.

Cahit, K. (2015). Internal validity: A must in research designs. *Educational Research and Reviews, 10*(2), 111-118.

Chakraborty, P., Shahriyar, R., Iqbal, A., & Uddin, G. (2021). How do developers discuss and support new programming languages in technical q&a site? an empirical study of go, swift, and rust in stack overflow. *Information and Software Technology, 137*, 106603.

Chauhan, U., & Shah, A. (2021). Topic Modeling Using Latent Dirichlet allocation: A Survey. *ACM Comput. Surv., 54*(7), Article 145. doi:10.1145/3462478

Choetkiertikul, M., Avery, D., Dam, H. K., Tran, T., & Ghose, A. (2015). *Who will answer my question on stack overflow?* Paper presented at the 2015 24th Australasian software engineering conference.

Contributors, W. (2024, 4 February 2024 18:36 UTC). Stack Overflow. Retrieved from https://en.wikipedia.org/w/index.php?title=Stack_Overflow&oldid=1203360003

Findley, M. G., Kikuta, K., & Denly, M. (2021). External validity. *Annual Review of Political Science, 24*, 365-393.

Gan, J., & Qi, Y. (2021). Selection of the optimal number of topics for LDA topic model—taking patent policy analysis as an example. *Entropy, 23*(10), 1301.

Hasan, M., Rahman, A., Karim, M. R., Khan, M. S. I., & Islam, M. J. (2021). *Normalized Approach to Find Optimal Number of Topics in Latent Dirichlet Allocation (LDA)*, Singapore.

Kabir, S., Udo-Imeh, D. N., Kou, B., & Zhang, T. (2023). Who answers it better? an in-depth analysis of ChatGPT and stack overflow answers to software engineering questions. *arXiv preprint arXiv:2308.02312*.

Kalepalli, Y., Tasneem, S., Teja, P. D. P., & Manne, S. (2020, 13-15 May 2020). *Effective Comparison of LDA with LSA for Topic Modelling.* Paper presented at the 2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS).

Khan, M. S. A., Farabi, A. R., & Iqbal, A. (2022). *What Do Firebase Developers Discuss About? An Empirical Study on Stack Overflow Posts*. Paper presented at the Proceedings of the 9th International Conference on Networking, Systems and Security, Cox's Bazar, Bangladesh. https://doi.org/10.1145/3569551.3569558

Lee, S. C. (2022). Topic modeling of Korean newspaper articles on aging via latent Dirichlet allocation. *Asian Journal for Public Opinion Research, 10*(1), 4-22.

Liu, J., Tang, X., Li, L., Chen, P., & Liu, Y. (2023). Which is a better programming assistant? A comparative study between chatgpt and stack overflow. *arXiv preprint arXiv:2308.13851*.

Montandon, J. E., Politowski, C., Silva, L. L., Valente, M. T., Petrillo, F., & Guéhéneuc, Y.-G. (2021). What skills do IT companies look for in new developers? A study with Stack Overflow jobs. *Information and Software Technology, 129*, 106429.

Nawrocki, P., Wrona, K., Marczak, M., & Sniezynski, B. (2021). A comparison of native and cross-platform frameworks for mobile applications. *Computer, 54*(3), 18-27.

Nikita, M. (2016). Select number of topics for LDA model. *CRAN R Project*.

Oliveira, V., Teixeira, L., & Ebert, F. (2020). *On the adoption of kotlin on android development: A triangulation study.* Paper presented at the 2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER).

Peruma, A., Simmons, S., AlOmar, E. A., Newman, C. D., Mkaouer, M. W., & Ouni, A. (2022). How do i refactor this? An empirical study on refactoring trends and topics in Stack Overflow. *Empirical Software Engineering, 27*(1), 11.

Rahman, M. S., & Roy, C. K. (2022, 2-2 Oct. 2022). *An Insight into the Reusability of Stack Overflow Code Fragments in Mobile Applications.* Paper presented at the 2022 IEEE 16th International Workshop on Software Clones (IWSC).

Rao, P. S., Pavan, B., Srivastava, A., Amani, K. V., & Sharma, A. (2022). DISTINCTION OF MOBILE FRAMEWORKS: FLUTTER VS NATIVE APPS. *International Research Journal of Modernization in Engineering Technology and Science, 4*(6), 2582-5208.

Saddler, J. A., Peterson, C. S., Sama, S., Nagaraj, S., Baysal, O., Guerrouj, L., & Sharif, B. (2020, 18-21 Feb. 2020). *Studying Developer Reading Behavior on Stack Overflow during API Summarization Tasks.* Paper presented at the 2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER).

Şahin, A. S., & BAYAZIT, N. G. (2020). What Java Developers have talked about? An empirical study on Stack Overflow. *Avrupa Bilim ve Teknoloji Dergisi*(19), 354-365.

Salza, P., Palomba, F., Di Nucci, D., De Lucia, A., & Ferrucci, F. (2020). Third-party libraries in mobile apps: When, how, and why developers update them. *Empirical Software Engineering, 25*, 2341-2377.

Shah, K., Sinha, H., & Mishra, P. (2019, 29-31 March 2019). *Analysis of Cross-Platform Mobile App Development Tools.* Paper presented at the 2019 IEEE 5th International Conference for Convergence in Technology (I2CT).

Sinha, S., Arora, R., Chockalingam, K., van der Vyver, M., Ponich, B., Ambikkumar, A., . . . Biernaskie, J. (2023). Plastic Surgery Clinical Trials: A Systematic Review of Characteristics, Research Themes, and Predictors of Publication and Discontinuation. *Plastic and Reconstructive Surgery–Global Open*(12), e5478.

So, H.-J., Jang, H., Kim, M., & Choi, J. (2023). Exploring public perceptions of generative AI and education: topic modelling of YouTube comments in Korea. *Asia Pacific Journal of Education*, 1-20.

Stanojević, J., Šošević, U., Minović, M., & Milovanović, M. (2022). *An Overview of Modern Cross-platform Mobile Development Frameworks.* Paper presented at the Central European Conference on Information and Intelligent Systems.

Tahaei, M., Vaniea, K., & Saphra, N. (2020). *Understanding privacy-related questions on stack overflow.* Paper presented at the Proceedings of the 2020 CHI conference on human factors in computing systems.

Tahmooresi, H., Heydarnoori, A., & Aghamohammadi, A. (2020). An Analysis of Python's Topics, Trends, and Technologies Through Mining Stack Overflow Discussions. *arXiv preprint arXiv:2004.06280*.

Treude, C., & Wagner, M. (2019, 25-31 May 2019). *Predicting Good Configurations for GitHub and Stack Overflow Topic Models.* Paper presented at the 2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR).

Uddin, G., & Khomh, F. (2021). Automatic Mining of Opinions Expressed About APIs in Stack Overflow. *IEEE Transactions on Software Engineering, 47*(3), 522-559. doi:10.1109/TSE.2019.2900245

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., & Wesslén, A. (2012). *Experimentation in software engineering*: Springer Science & Business Media.

Xu, R., Lakeh, A. B., & Ghaffarzadegan, N. (2021). Examining the characteristics of impactful research topics: A case of three decades of HIV-AIDS research. *Journal of Informetrics, 15*(1), 101122.

Yunita, A. (2023). Challenges in front-end JavaScript development for web applications—Developers' perspective.

Zohud, T., & Zein, S. (2021). Cross-platform mobile app development in industry: A multiple case-study. *International Journal of Computing, 20*(1), 46-54.

Zolduoarrati, E., Licorish, S. A., & Stanger, N. (2022). Impact of individualism and collectivism cultural profiles on the behaviour of software developers: A study of stack overflow. *Journal of Systems and Software, 192*, 111427.