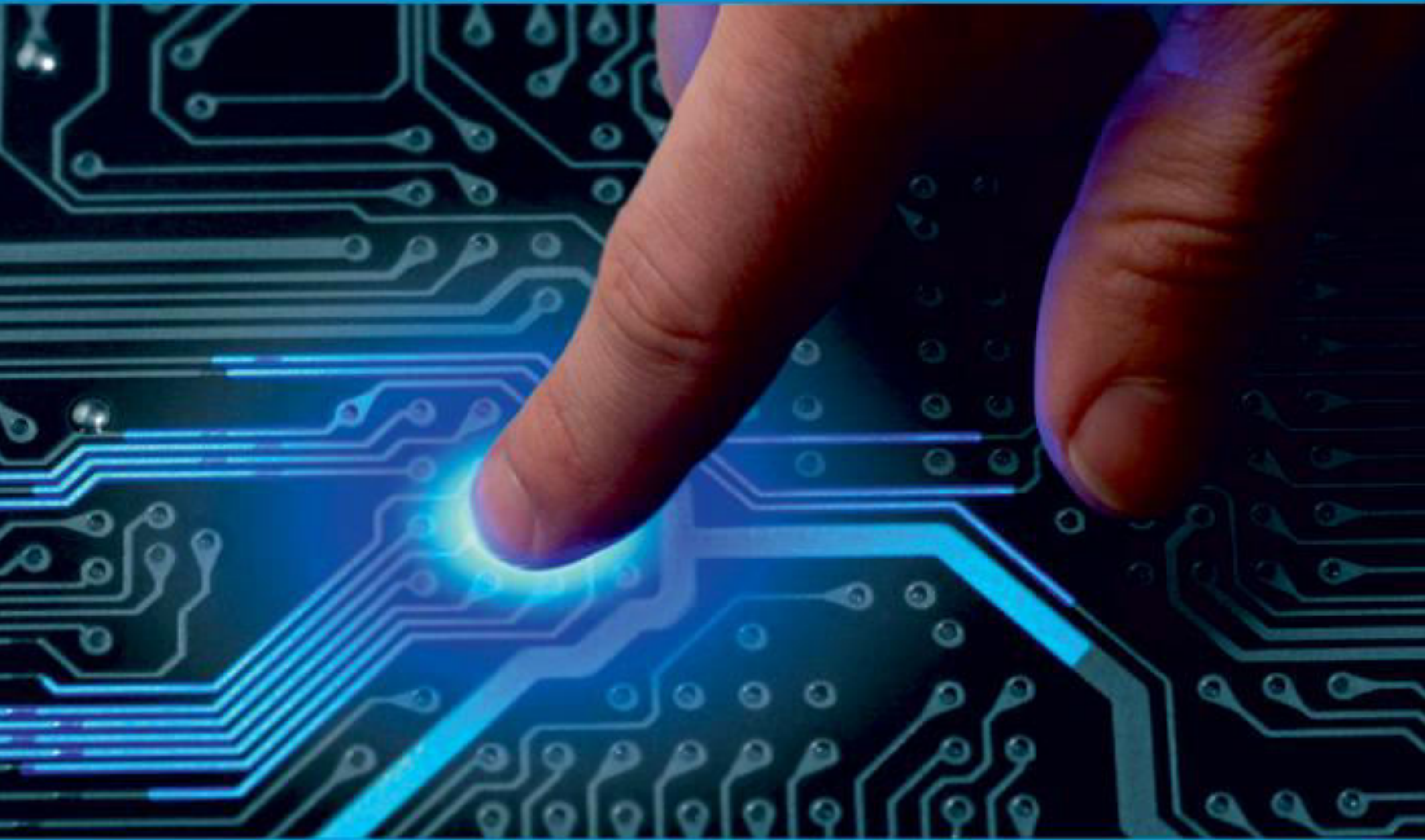
IJIRCCE

e-ISSN: 2320-9801 | p-ISSN: 2320-9798



INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

Volume 12, Issue 3, March 2024

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 8.379

International Journal of Innovative Research in Computer and Communication Engineering

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 3, March 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1203088 |

Exploring Flutter for Mobile App Development

Raj Joshi, Vaibhavi Parikh

Department of Computer Science and Engineering, Parul University, Vadodara, India

Asst. Professor, Department of Computer Science and Engineering, Parul University, Vadodara, India

ABSTRACT: This abstract presents a brief summary of Flutter, an open-source UI toolkit developed by Google for mobile app development. It discusses Flutter's strategy of utilizing a single codebase, its architecture based on widgets, and its seamless integration with popular development environments such as Android Studio and Visual Studio Code. Furthermore, it underscores Flutter's advantages in terms of speed, productivity, and code maintainability, while also acknowledging the challenges associated with platform-specific variations. In conclusion, it underscores the transformative impact of Flutter in facilitating the efficient creation of high-quality, cross-platform applications.

KEYWORDS: Application, Application store, Flutter, Dart, Android Studio, Android Operating System, Google, Material UI.

I. INTRODUCTION

In today's rapidly evolving technological landscape, mobile applications have become essential tools that streamline daily tasks, facilitate communication, and entertain users worldwide. With the widespread adoption of smartphones and tablets, businesses and individuals alike are increasingly leveraging mobile platforms to engage their audiences and deliver services efficiently. In this context, mobile application development has emerged as a crucial endeavor, fueling innovation and shaping user interactions.

Amidst the array of frameworks and platforms available for mobile app development, Flutter has garnered considerable attention for its flexibility, performance, and user-friendly nature. Flutter, an open-source UI software development kit developed by Google, empowers developers to create natively compiled applications for mobile, web, and desktop platforms from a unified codebase. With its extensive collection of widgets, robust tooling, and cross-platform support, Flutter presents an appealing solution for crafting high-quality, visually captivating applications that resonate with users across diverse devices and operating systems.

This introduction serves as an entry point into the realm of mobile app development using Flutter. Throughout this exploration, we will delve into the intricacies of Flutter development, uncover its distinctive features, and explore how it transforms the way developers craft immersive mobile experiences. From grasping the basics of Flutter to mastering advanced techniques, this journey will equip you with the knowledge and skills necessary to embark on your own mobile app development ventures confidently.

Whether you're an experienced developer looking to broaden your toolkit or a newcomer eager to venture into the dynamic realm of mobile app creation, this guide will act as your guide, steering you through the nuances of Flutter development and empowering you to breathe life into your app concepts with finesse and efficiency. Join us as we embark on this journey into the realm of mobile app development using Flutter, where innovation knows no bounds, and creativity flourishes.

Limitations of the Study

At present, this Flutter application is compatible only with Apple devices running iOS 10.0 and above. The application necessitates an internet connection as well as a memory capacity of more than 50MB. Additionally, users must possess an email address to register themselves as application users.

International Journal of Innovative Research in Computer and Communication Engineering

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 3, March 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1203088 |

II. RESEARCH GOAL

The primary objective of this thesis is to establish a comprehensive understanding of the application development process. This research investigates the efficacy of an application designed to address a longstanding issue. Additionally, it offers a documented application that can serve as a reference for individuals embarking on their own Flutter application development journey. The underlying hypothesis is that Fostlings will prove effective in mitigating the persistent issue of misinformation regarding car parts online, thereby saving time typically wasted on navigating through numerous unreliable sources and facilitating more informed decision-making during significant purchases. The aim of this thesis is to provide solutions to the aforementioned challenges and pave the way for a less stressful experience in the car-building process.

III. FLUTTER

This chapter serves as an introduction to the Flutter cross-platform framework, aiming to provide guidance for a better understanding of coding practices.

Flutter, developed by Google, is a high-performance cross-platform framework built on the Dart programming language. It offers easy-to-learn and highly customizable widgets that facilitate the creation of visually appealing applications. In Flutter, everything revolves around widgets, which are responsible for constructing the user interface. Flutter's composability feature enables developers to create sleek interfaces efficiently, a detailed exploration of Flutter's advantages will be presented later in this chapter.

3.1 Dart Coding Language

Dart, also developed by Google, is an object-oriented programming language that supports concepts such as classes and interfaces. The following code snippet illustrates a class in Dart, which resembles the structure of classes in many other object-oriented languages.

```dart
class Employee {
  String name;

  // Getter method
  String get emp_name {
    return name;
  }

  // Setter method
  void set emp_name(String name) {
    this.name = name;
  }

  // Function definition
  void result() {
    print(name);
  }
}

void main() {
  // Object creation
  Employee emp = new Employee();
  emp.name = "employee1";
  emp.result(); // Function call
}
```

3.2 Widgets

As previously mentioned, Flutter revolves around widgets. The key widgets responsible for building the visual representation of a Flutter application include state maintenance widgets, platform-specific widgets (for Android or Apple), layout widgets, and basic widgets.

State maintenance widgets are crucial components that manage the lifecycle of a Flutter application by tracking user interactions and data changes.

Platform-specific widgets cater to the requirements of specific platforms, allowing developers to tailor the application accordingly.

Layout widgets can contain either a single child or multiple children. Single child widgets, such as Container, Center, and Align, are used for arranging individual components, while multiple child widgets like Row, Column, ListView, GridView, and Expanded are employed for organizing multiple components.

3.3 State Management

State management plays a pivotal role in managing the lifecycle of an application. It encompasses ephemeral state, which pertains to the current state visible to the user and is managed by stateful widgets, and app state, which persists throughout the application's runtime and stores user data and session details.

3.4 Advantages of Using Flutter

Flutter offers several advantages, including high-performance applications, extensive customization options, and strong support from Google. Features such as hot reload facilitate rapid development by instantly reflecting code changes. Additionally, Dart's extensive library of software packages enables developers to implement creative ideas efficiently. Moreover, Flutter's single codebase minimizes development time, leading to faster time-to-market compared to native solutions for Android and iOS applications.

IV. DEPLOYMENT

This chapter is intended for individuals interested in publishing their applications on the Apple App Store. If this is not relevant to your goals, feel free to skip this section. It serves as a general guide to successfully navigate the process of uploading an application to the app store, providing high-level insights rather than a step-by-step tutorial. The primary aim is to help users avoid potential obstacles encountered during the application publishing process.

4.1Deployment

To initiate the application upload process, it is imperative to possess an Apple Developer account. This necessitates a payment of $100 annually. Once the account is set up, users can refer to https://docs.flutter.dev/deployment/ios for comprehensive instructions on preparing the application using Xcode. If encountering any issues with Xcode, utilizing Flutter doctor is recommended. This tool provides an overview of any missing components or malfunctioning features. In some cases, resolving issues may involve deleting and reinstalling CocoaPods.

4.2 Feedback Management

It is essential to remain attentive to feedback from users who leave reviews on the application. According to William Martin et al.'s article "A Survey of App Store Analysis for Software Engineering," they found that free apps are more likely to maintain their position in the top charts, with frequent feature updates being crucial for their sustained success, particularly in smaller categories. This underscores the importance of adhering to this principle, as Fostlings aims to remain relevant over time. Consistently monitoring feedback and regularly updating the application are key strategies for longevity.

Quick Note: Before finalizing the publication process, ensure that all selected devices and versions are properly configured in Xcode. Additionally, if the "Debug" red banner persists in the application, it cannot be published. To address this, simply incorporate the following code snippet into the main.dart page.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 3, March 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1203088 |

4.3Permissions

For those incorporating photo or camera functionality into their applications, managing permissions is crucial. Various solutions exist to address issues related to accessing photos or cameras within the application. However, it is essential to ensure that appropriate permissions are requested from the user, granting the application access to view photos or utilize the camera.

V. WIDGET ARCHITECTURE

This chapter provides an elucidation on how Flutter's widgets interact with each other. The figures presented below illustrate widget trees, depicting how each page invokes other widgets. Although the trees exclusively feature pages, in the actual application, these pages are interconnected, pointing to one another. This section serves as a brief guide to understanding widget construction in Flutter, highlighting the distinction between single and multiple widgets. Additionally, it underscores that all these widgets have the capability to modify or update the application's state.

5.1Login

The LoginPage Widget initiates the Scaffold, which contains two child calls. It invokes AppBar and SingleChildScrollView. SingleChildScrollView, in turn, invokes Column, which further calls Padding, FlatButton, and Container. The widget tree for the login page is depicted in Figure 1.

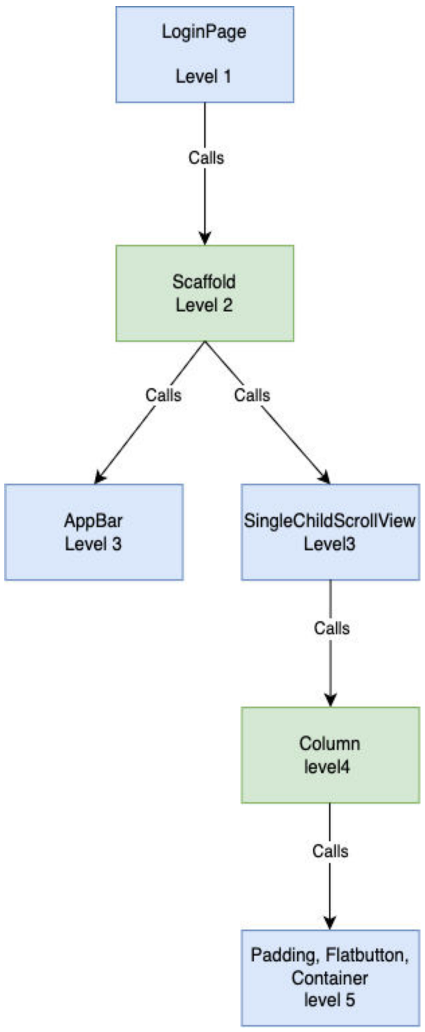


Figure 1: Login Widget Build

Sign Up

The SignUp widget invokes Scaffold, which comprises two children: AppBar and Column. Column then invokes Padding and Container. The widget tree for the sign-up page is illustrated in Figure 2 below.

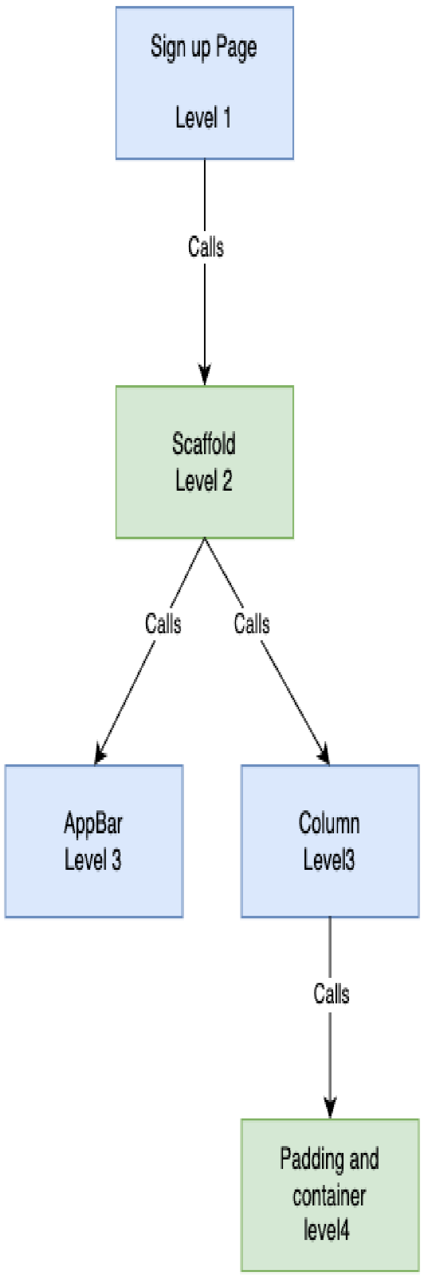


Figure 2: Sign Up Widget Build

Discussion

For Discussion Page it calls Scaffold, Scaffold has two children AppBar and Listview.Builder. AppBar calls IconButton which contains an action that calls a search method. ListView.Builder calls ListTile, ListTile calls container, Container calls row and row calls Container and Column. Figure 3 shown below contains the widget tree for the Discussion Page.

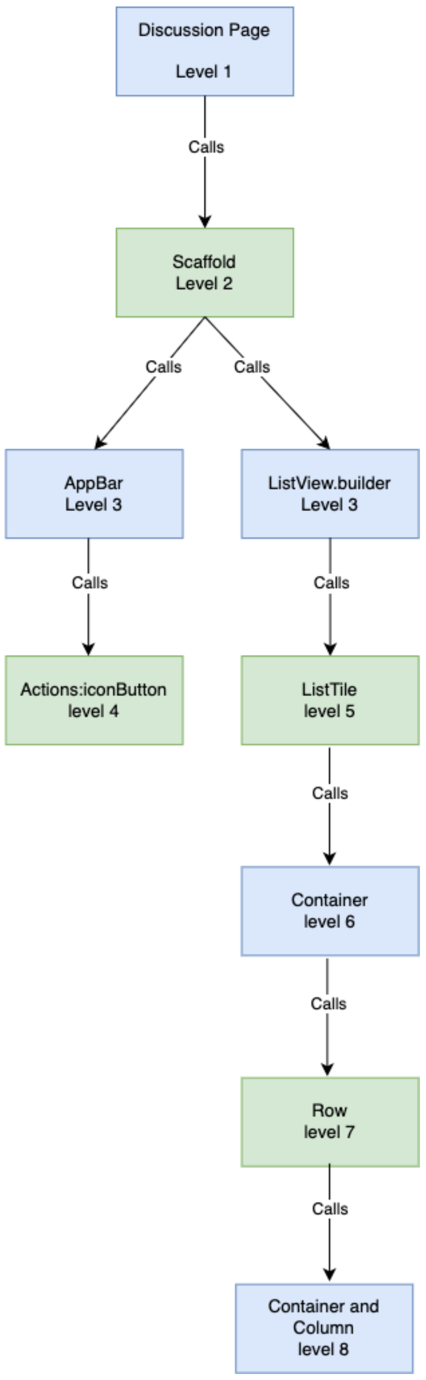


Figure 3: Discussion Widget Build

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 3, March 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1203088 |

Build Page

The Build page calls Scaffold, Scaffold contains two children AppBar and ListView.Builder. ListView.Builder calls ListTile and ListTile calls Image.Network. The widget tree for Build Page is shown in Figure 4.

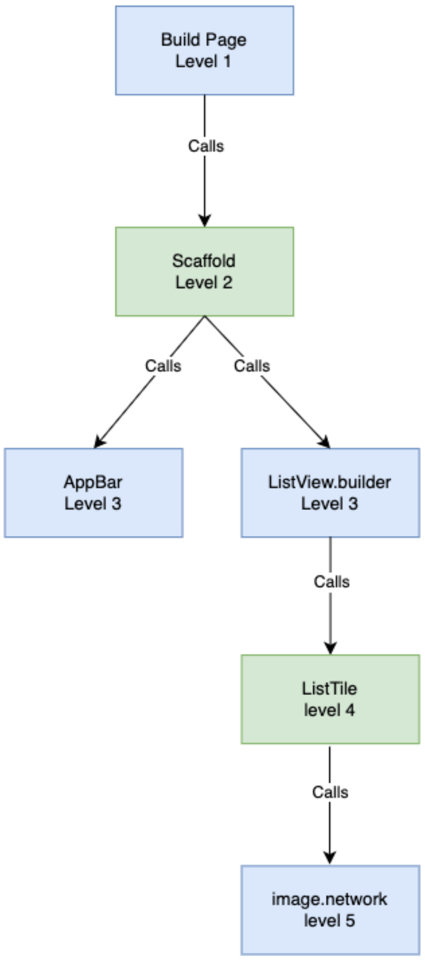


Figure 4: Build Widget Build

Specific Builds page

The specific Builds Page calls Scaffold, scaffold contains two children AppBar and SingelChildScrollView. SingleChildScrollView calls Column and Column calls Image.network and Padding. The widget tree for Specific build page is shown below in Figure 5

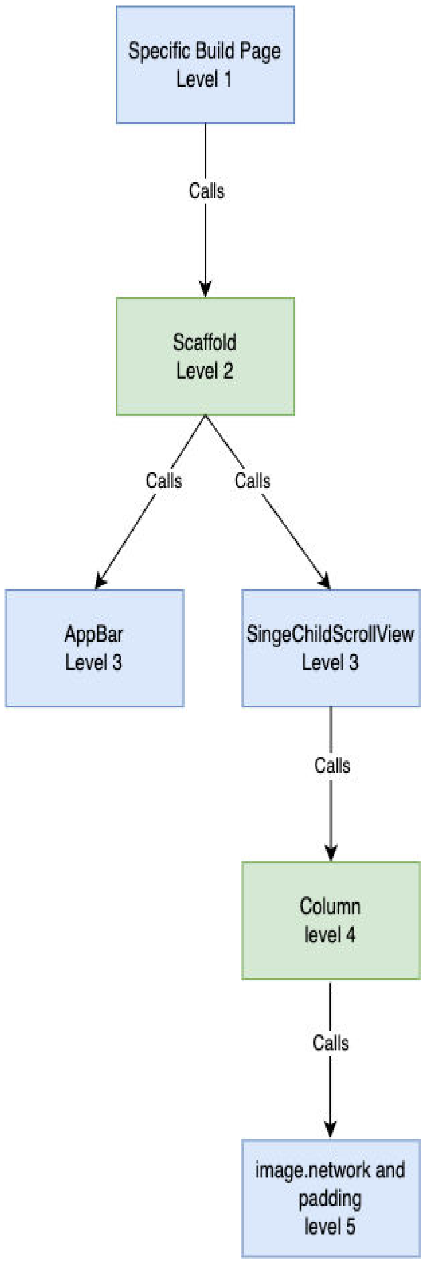


Figure 5: Specific Build Widget Build

Specific Discussion Page

For Discussion Page it calls Scaffold, Scaffold has two children AppBar and Column. Column calls Expanded calls Container, Container calls Row, Row Calls Expanded, Expanded calls Container and container calls BoxDecoration and RichText. Figure 6below contains the widget tree for the specific Discussion Page.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 3, March 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1203088 |

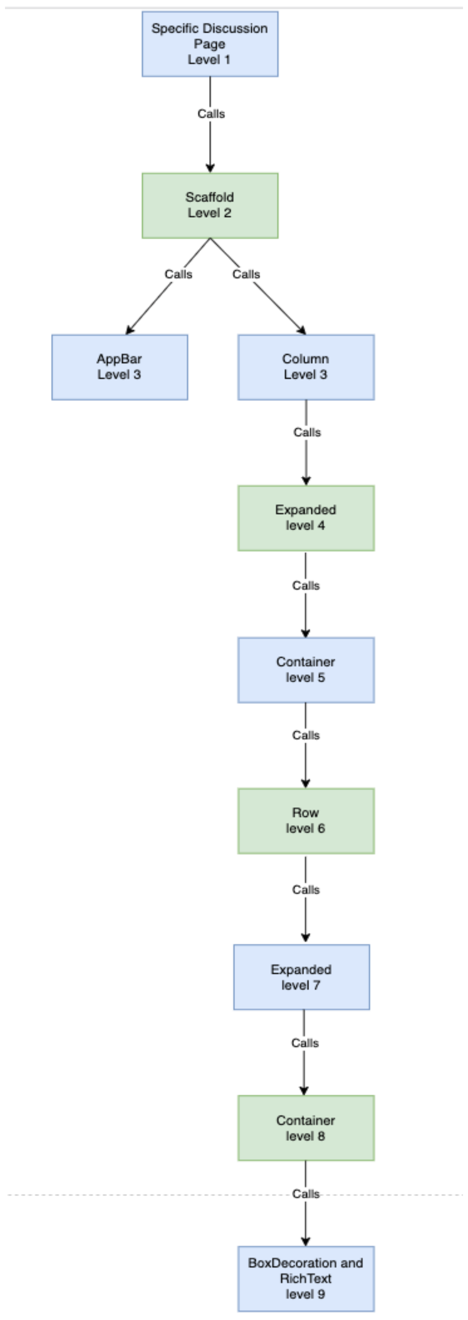


Figure 6: Specific Discussion Widget Build

VI. FUTURE WORKS AND CONCLUSION

This study has provided a comprehensive understanding of the software development life cycle. It is clear that when a program enters production, it necessitates vigilant monitoring due to the increased traffic on the application. This surge in traffic leads to numerous service calls, which can potentially cause issues for some users. Therefore, it is imperative to comprehend how to effectively manage these problems, as users expect solutions rather than excuses to ensure a seamless experience on the application.

Flutter has proven to be highly advantageous in this regard, as it eliminates the need to develop a separate program for deployment on different platforms. The simplicity of Dart has enabled the creation of the envisioned application from

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 3, March 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1203088 |

the inception of this thesis. However, obtaining conclusive results for this thesis within a short timeframe is not feasible, as the publication requires a fully functional application first.

REFERENCES

1. "Flutter Documentation": The official documentation provided by Google offers comprehensive guides, tutorials, and API references for both Flutter and Dart. It's an invaluable resource for developers of all levels. [Flutter Documentation](https://flutter.dev/docs)
2. **"Flutter Succinctly" by Ed Freitas**: This ebook provides a concise yet thorough overview of Flutter, covering topics ranging from installation and setup to building complex UIs and integrating with backend services. [Flutter Succinctly](https://www.syncfusion.com/ebooks/flutter-succinctly)
3. "Dart Programming Language Specification": Understanding the Dart language is essential for Flutter development. This specification outlines the syntax, semantics, and features of the Dart programming language in detail. [Dart Language Specification](https://dart.dev/guides/language/spec)
4. "Flutter in Action" by Eric Windmill: This book offers a hands-on approach to learning Flutter, guiding readers through the process of building real-world mobile applications using Flutter and Dart. [Flutter in Action](https://www.manning.com/books/flutter-in-action)
5. "Beginning App Development with Flutter" by Rap Payne, Seth Ladd, and Eric Windmill: Suitable for beginners, this book provides a step-by-step guide to building mobile applications with Flutter. It covers topics such as UI design, state management, navigation, and more. [Beginning App Development with Flutter](https://www.apress.com/gp/book/9781484251808)
6. "Flutter & Dart - The Complete Guide [2021 Edition]" by Maximilian Schwarzmüller: This Udemy course offers a comprehensive introduction to Flutter and Dart, covering everything from basic concepts to advanced techniques. It includes hands-on projects to reinforce learning. [Flutter & Dart - The Complete Guide](https://www.udemy.com/course/learn-flutter-dart-to-build-ios-android-apps/)
7. "Flutter Explained: How to Build iOS and Android Apps with Google's Flutter and Dart" by Andrea Bizzotto: This book provides a detailed explanation of Flutter's architecture, widgets, and best practices for building cross-platform mobile applications. [Flutter Explained](https://www.amazon.com/Flutter-Explained-Android-Apps-Googles/dp/1838986303)
8. "Flutter Tutorials" by The Net Ninja: This YouTube playlist offers a series of video tutorials covering various aspects of Flutter development, including UI design, state management, Firebase integration, and more. [Flutter Tutorials Playlist](https://www.youtube.com/playlist?list=PL4cUxeGkcC9jLYyp2Aoh6hcWuxFDX6PBJ)

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING