Dynamic theming with Animated container

Workshop #2

Flutter Developer Bootcamp

# **Purpose**

This workshop demonstrates how to implement dynamic theming functionality in a Flutter application with Animated container.

# **Problem**

In the given workshop, Theme Switching allows users to toggle between a light and dark theme using a switch the theme is applied to the entire app using Theme Data. dark () and Theme Data. light().You need to create Resizable Box A blue box with the text "Resizable Box" is displayed in the center of the screen and also create two buttons: "Enlarge Box" and "Shrink Box".The box's size changes smoothly over one second using the Animated Container.

# **How to Solve**

1. Checkout the workshop from Git Repo:

git clone -b <user-branch> <repo-URL>

2. Open the root folder inside VS Code

3. To build the app click the run option in the main method{}

4. Create a resizable blue box labeled "Resizable Box" at the center of the screen. Include buttons labeled "Enlarge Box" and "Shrink Box" to adjust the box's size.

5. Click Enlarge box button, you can see that the box size enlarging.

* Identify the section where the resizable box is defined.
* Wrap this container widget with the AnimatedContainer widget.
* Set the duration property of the AnimatedContainer to specify the duration of the animation.

6. Go To File: <specific-file-with-theming-method> à <method-name>, implement your theming logic. Make the animation container works

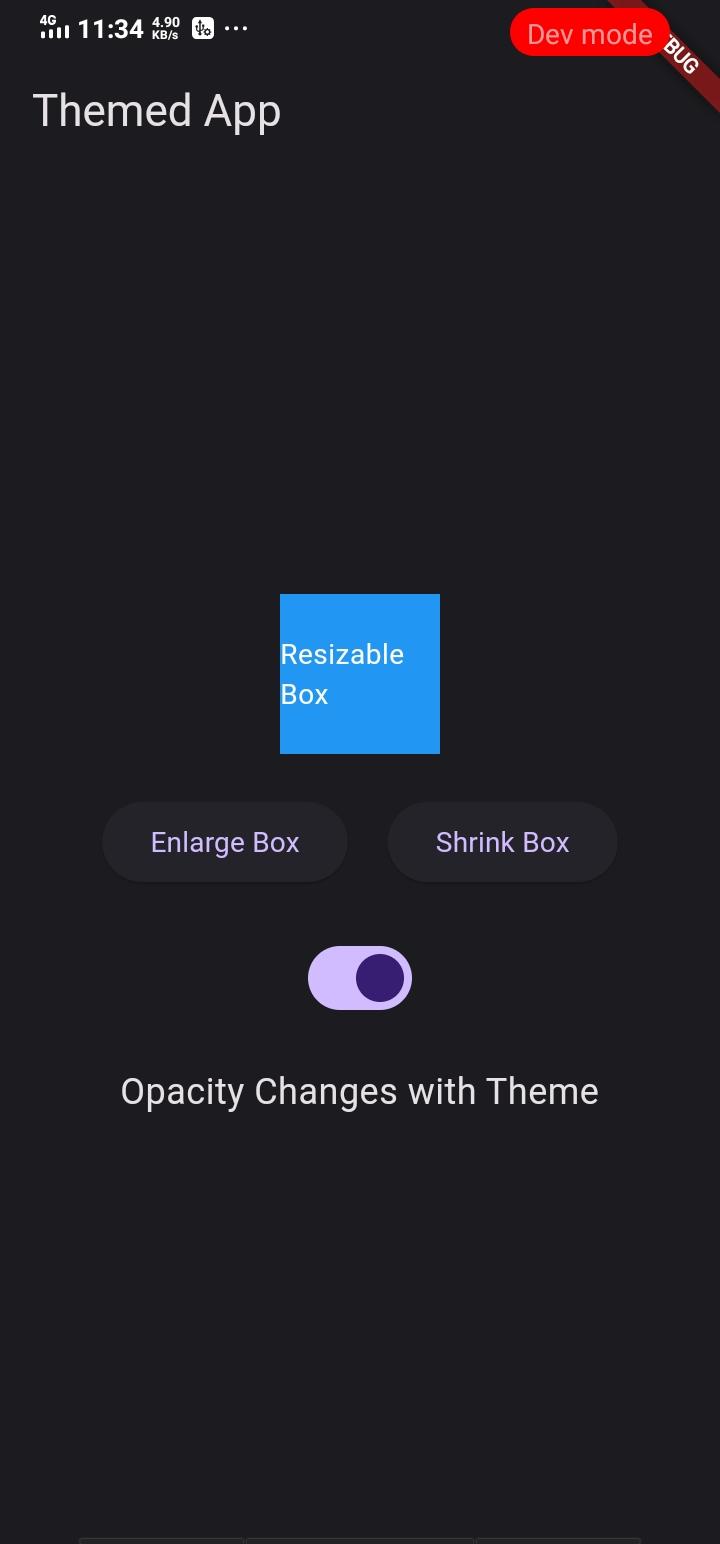
**You will Achieve**

When you complete this workshop you will learn the following:

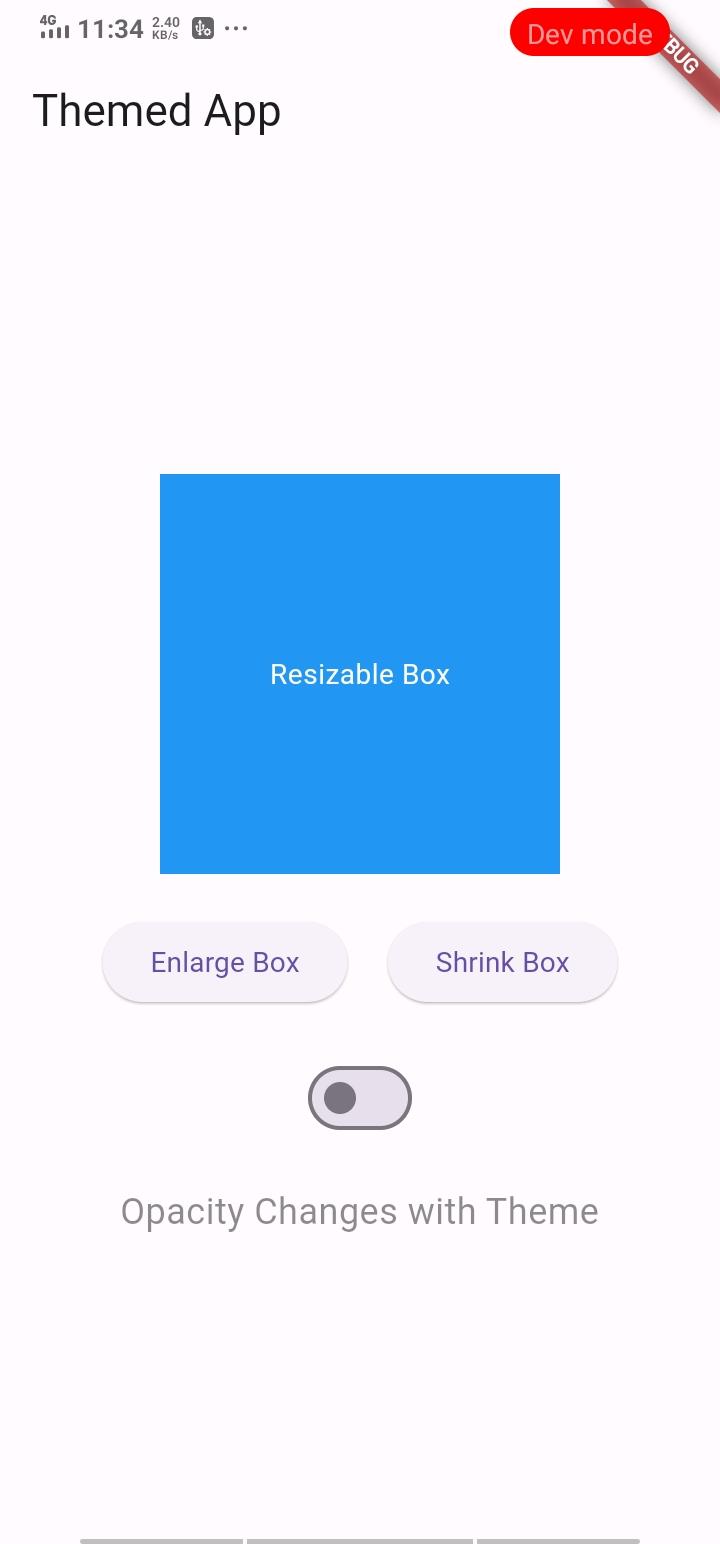
* **Appbar**: Utilize the AppBar widget to create a top app bar that can include navigation controls, branding elements, and action buttons, improving app navigation and usability.
* **Buttons**: Integrate various types of buttons into your app interface to trigger actions or navigate to different screens, enhancing interactivity and user engagement.
* **Icons**: Incorporate icons into your app design for visual representation of actions, categories, or features, improving visual clarity and user experience.
* **Padding**: Adjust spacing around widgets using the Padding widget to improve the layout and aesthetics of your app, ensuring a visually appealing design.
* **SizedBox**: Manage widget sizes and spacing within your app's layout using the SizedBox widget, providing flexibility and control over the layout's structure.
* **Text**: Display text content within your app interface, including headings, paragraphs, labels, and other textual information, ensuring effective communication with users.
* **Adjusting Text Alignment**: Control the alignment of text elements within your app interface to improve readability and visual consistency, enhancing the overall user experience.

# **Screenshots**

# **Before implementation (without Resizable box )**



## **After implementation (with Resizable box)**



# **How to submit your workshop**

Push your project back to the same git branch using command:

<command name>

# **Happy Coding!**