**Chapter 5:**

In Chapter 5, I’ve learned about the widget tree, which is created by composing (or nesting) widgets to build both simple and complex layouts. As the widget tree becomes deeper with more nested widgets, the code can become harder to manage, so it’s best practice to keep the widget tree as shallow as possible. This chapter introduces me to the key widgets used in Flutter and explains the impact of having a deep widget tree on my code. I’ve also learned how to refactor a deep widget tree into a shallower one, making the code more manageable and easier to follow. The chapter covers three techniques for achieving a shallow widget tree: refactoring with a constant, with a method, and with a widget class. Each approach is explored, including the benefits and drawbacks of using each technique, helping you decide the best method for your app's needs.

