Module 3: Introduction to Flutter Widgets and UI Components

1. Difference between Stateless and Stateful Widgets

- **Stateless Widget**:
  - A widget that does not change over time. Once built, it does not rebuild unless the parent widget changes.
  - It's used for static content that doesn't depend on any mutable state.
- **Stateful Widget**:
  - A widget that maintains a state that can change during the widget's lifetime.
  - It allows for dynamic behavior such as responding to user interactions, changing data, or other updates.
  - A StatefulWidget has an associated State object that holds the mutable state and manages its changes using the setState() method.

2. Widget Lifecycle and State Management in Stateful Widgets

- **Lifecycle**:
  1. **createState()**: The first step in creating a stateful widget, it returns a State object.
  2. **initState()**: Called once when the state object is created. It's useful for initializations.
  3. **build()**: Rebuilds the widget tree whenever the widget's state changes. This method returns the UI representation of the widget.
  4. **didUpdateWidget()**: Called when the parent widget is rebuilt, and the state of the widget may need updating.
  5. **setState()**: A method that triggers a rebuild when the widget's internal state is modified.
  6. **deactivate()**: Called when the widget is removed from the tree temporarily.
  7. **dispose()**: Used for cleaning up resources when the widget is permanently removed from the widget tree.

- **State Management**:
    - **State** is managed within the State object, and when setState() is called, it triggers a rebuild to reflect the updated state.
    - The state is mutable, which allows dynamic changes in the widget during the app's runtime.

3. Common Flutter Layout Widgets

1. **Container**: A versatile widget that can be used to decorate, position, and size its child. It allows properties like padding, margins, background color, and more.
2. **Column**: A widget that arranges its child widgets vertically. It allows alignment and spacing between children.
3. **Row**: A widget that arranges its child widgets horizontally. It is useful for creating horizontal layouts.
4. **Stack**: A widget that allows children to be layered on top of each other, useful for overlapping elements like text on an image.
5. **Expanded**: A widget that takes up remaining space in a Row, Column, or Flex layout. It ensures that a widget stretches to fill available space, distributing the space evenly among its children.

These layout widgets provide flexibility to design responsive and complex UIs by controlling the arrangement and distribution of child widgets.