Name: Vrushabh Ghuse Class: D15A Roll no. 22

MAD PWA Exp 3

Aim: To include icons, images and fonts in flutter app Theory:

Flutter, Google's open-source UI toolkit, empowers developers to craft visually appealing and highly performant applications. A fundamental aspect of creating a compelling user interface lies in the seamless integration of icons, images, and custom fonts.

Icons

Icons serve as intuitive visual cues, aiding users in navigation and comprehension. Flutter provides an extensive set of built-in icons, and developers can easily integrate custom icons. Icons contribute to a consistent design language and enhance the overall user experience.

Principles:

1. Selecting Icon Packages: Developers often leverage existing icon packages, such as "material_icons" or "font_awesome_flutter," to access a diverse range of symbols. This streamlines the integration process and ensures a cohesive visual language.
2. Installation and Implementation: The chosen icon package is added to the project's dependencies. Icons can then be effortlessly implemented using the Icon widget, specifying the desired icon type, size, and color.

Images

Images play a pivotal role in conveying information, setting the mood, and creating an immersive user experience. Flutter supports various image formats and provides mechanisms for both local and network-based image integration.

Principles:

1. Organizing Image Assets: Structuring image assets within the 'assets' folder and updating the pubspec.yaml file facilitates proper asset management. This step ensures clarity and ease of access for images within the Flutter project. 2. Loading Images: Utilizing the Image.asset widget for local images and Image.network for network images simplifies the integration process. This allows developers to incorporate visuals seamlessly into their application interfaces.

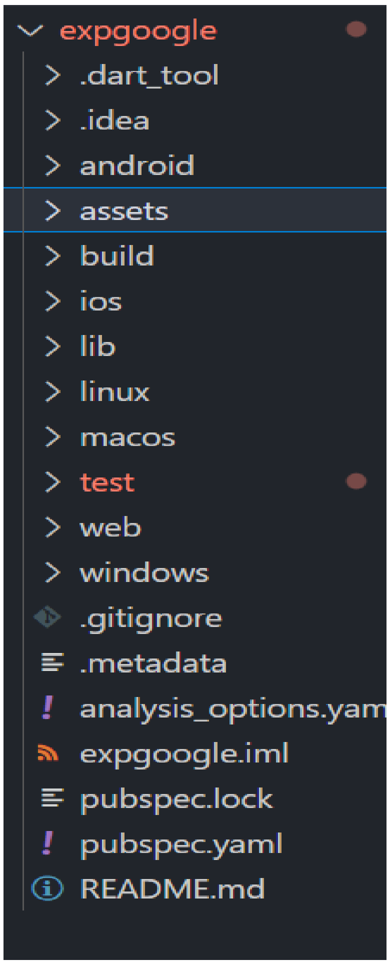
Fonts

Custom fonts contribute to brand identity and enhance the overall aesthetics of an application. Flutter enables the integration of custom fonts, allowing developers to align the typography with the app's design principles.

Principles:

1. Adding Font Files: Placing font files within a 'fonts' folder and updating the pubspec.yaml file establishes a clear structure for font assets. This practice simplifies the inclusion of custom fonts in Flutter projects.
2. Loading and Applying Fonts: Integrating custom fonts involves specifying the font family and style using the TextStyle widget. This ensures consistent and visually appealing text elements throughout the application.

File structure:



```
expgoogle
  .dart_tool
  .idea
  android
  assets
  build
  ios
  lib
  linux
  macos
  test
  web
  windows
  .gitignore
  .metadata
  analysis_options.yam
  expgoogle.iml
  pubspec.lock
  pubspec.yaml
  README.md
```

Code:

//main.dart

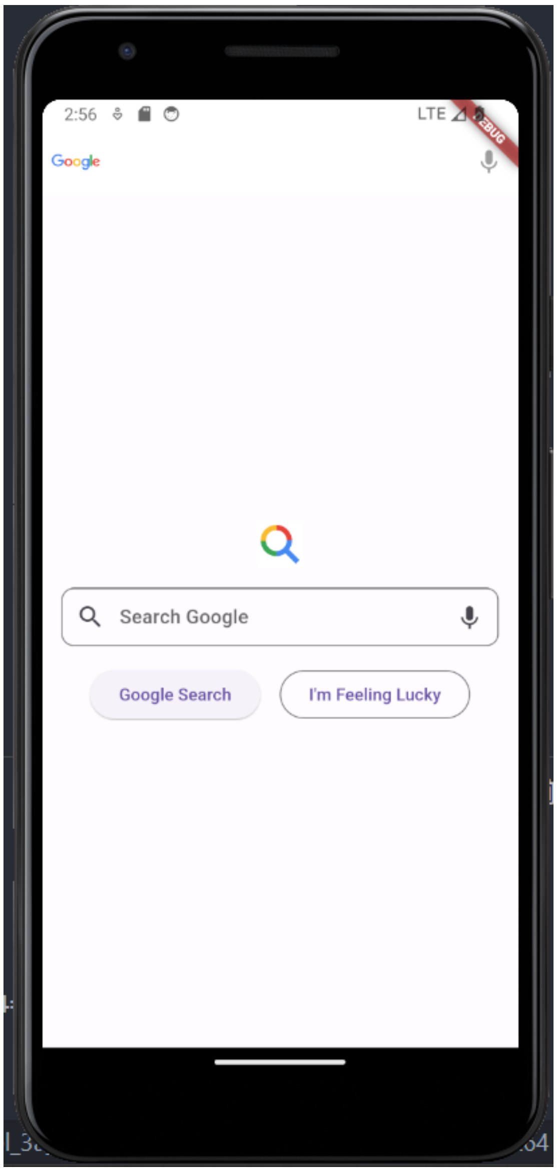
```
import 'package:flutter/material.dart';

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: GoogleSearchScreen(),
    );
  }
}

class GoogleSearchScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor: Colors.white,
        elevation: 0,
        leading: Image.asset(
          'assets/google_logo.png', // Add your Google logo image in the assets folder
          width: 120,
          height: 40,
        ),
        actions: [
          IconButton(
            icon: Icon(Icons.mic),
            color: Colors.grey,
            onPressed: () {
              // Handle microphone button press
            },
          ),
        ],
      ),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            Image.asset(
              'assets/search_icon.png', // Add your search icon image in the assets folder
              width: 40,
              height: 40,
            ),
            SizedBox(height: 16),
            TextField(
              decoration: InputDecoration(
                hintText: 'Search Google',
                border: OutlineInputBorder(
                  borderRadius: BorderRadius.circular(10),
                ),
                contentPadding: EdgeInsets.symmetric(vertical: 10),
                prefixIcon: Icon(Icons.search),
                suffixIcon: Icon(Icons.keyboard_voice),
              ),
            ),
            SizedBox(height: 16),
            Row(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
                TextButton(
                  onPressed: () {
                    // Handle "Google Search" button press
                  },
                  child: Text('Google Search'),
                ),
                SizedBox(width: 16),
                TextButton(
                  onPressed: () {
                    // Handle "I'm Feeling Lucky" button press
                  },
                  child: Text("I'm Feeling Lucky"),
                ),
              ],
            ),
          ],
        ),
      ),
    );
  }
}
```

Output:



Conclusion:

In conclusion, the integration of icons, images, and fonts in Flutter applications is foundational to creating visually stunning and engaging user interfaces. By adhering to best practices, developers can uphold design consistency, enhance user experience, and bring forth applications that resonate aesthetically with their target audience.