MOHD USMAAN MOGAL
36
D15A

EXP 3

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

Theory:

Image Widget:

The Image widget handles decoding and rendering of various image formats, such as JPEG, PNG, GIF, and more.
It can dynamically adapt to different image sources and resolutions.
Image Caching:

Flutter automatically caches images, improving performance by avoiding redundant network requests or file reads.
This caching mechanism helps in efficiently managing and displaying images.
Image Transformations:

The widget allows for transformations like scaling, cropping, and rotation, providing developers with control over the displayed image's appearance.
Placeholder and Error Handling:

Developers can specify placeholder images to display while the actual image is being loaded.
Error images can be configured to handle cases where the intended image cannot be loaded.
Icon Widget:
Semantic Labels:

The Icon widget allows you to set a semantic label, providing accessibility for users with screen readers.
This enhances the user experience for those who rely on assistive technologies.

Icons:

Developers can use custom icons in addition to the extensive set of built-in icons.
Custom icons can be loaded from various sources, such as font-based icons or image files.
Icon Theming:

Icons can be styled and themed using the color property, allowing for consistent design across the app.
Size Adjustment:

The size of the icon can be adjusted based on application requirements, providing flexibility in the UI design.

NetworkImage:

NetworkImage efficiently manages the retrieval of images from network URLs, optimizing the network requests for performance.
Caching Strategies:

It incorporates caching strategies to minimize redundant network requests, ensuring a smooth user experience even in varying network conditions.
Error Handling:

NetworkImage allows developers to handle errors gracefully, specifying fallback images or other actions in case the network image cannot be loaded.
Progressive Loading:

Progressive loading is supported, where images are displayed gradually as they are being downloaded, improving perceived performance.

Code:

```
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';
import 'package:reddit_clone/core/common/loader.dart';
import 'package:reddit_clone/core/common/sign_in_button.dart';
import 'package:reddit_clone/core/constants/constants.dart';
import 'package:reddit_clone/features/auth/controller/auth_controller.dart';

class LoginScreen extends ConsumerWidget {
  const LoginScreen({super.key});

  @override
  Widget build(BuildContext context, WidgetRef ref) {
    final isLoading = ref.watch(authControllerProvider);

    return Scaffold(
      appBar: AppBar(
        title: Image.asset(
          Constants.logoPath,
          height: 40,
        ),
```

```
        actions: [
          TextButton(
              onPressed: () {},
              child: const Text(
                'Skip',
                style:
                    TextStyle(fontWeight: FontWeight.bold, color: Colors.blue),
              ))
        ],
      ),
      body: isLoading
          ? const Loader()
          : Column(children: [
              const SizedBox(height: 30),
              const Text(
                'Dive into anything',
                style: TextStyle(
                    fontSize: 24,
                    fontWeight: FontWeight.bold,
                    letterSpacing: 0.5),
              ),
              const SizedBox(
                height: 30,
              ),
              Padding(
                padding: const EdgeInsets.all(8.0),
                child: Image.asset(
                  Constants.loginEmotePath,
                  height: 400,
                ),
              ),
              const SizedBox(
                height: 20,
              ),
              const SignInButton()
            ]),
    );
  }
}
```

Output: