Sri Lanka Institute of Information Technology

Linkup

Current Trends in Software Engineering- SE4010

2022

Submitted by:

Lasal Sandeepa Hettiarachchi

IT19132310

User profile management

Table of Contents

1. Background

Linkup is a social networking platform mainly focused on connecting skilled professionals with employers. In a highly competitive job market, it is essential that employers are given the ability to filter out the talent that is required and suited for the jobs. Also, the applicants must be able to showcase their knowledge and capabilities. Linkup takes the essential foundation of social networking applications and elevates it by adding extensive capabilities for job posting, application management, etc. It simplifies the process of professional networking.

In the application, there is one single user role. Unlike the competing applications where separate user roles are created for posting jobs and applying/exploring for jobs, Link up takes a different approach. In this particular application, the user has a single login and from there, they can either post jobs, edit jobs, manage the job applications that are coming, etc. (job posting functionality) while also being able to explore jobs and apply for those ones that they prefer. Based on that, the users can be divided logically as,

- Employers
- Applicants

But in essence, a user can take both roles at once.

User profile management is a highly essential feature in a professional networking application like this as the target group of the app is mainly job seekers. From an applicant's perspective, they should be able to emphasize their skills and experiences and highlight what sets them apart from the rest so that they can stand out to employers. Therefore, the ability to add employment history, skills, etc. to the profile is essential. In linkup, highly interactive modules and forms have been provided in order to achieve this (further emphasized in the following sections).

Apart from this, it is essential to securely authenticate and authorize the users for the functionalities of this application. A token-based authentication system is used to authenticate the users while an email password combination is used to authorize them. The user cannot access the functionalities of the application without logging in to the system and taking the authorization privileges. The token is created when the user enters the correct combination of email and password and is sent from the server to the flutter application. The token is stored in the secure storage (discussed under flutter features) and accessed when calling APIs that require the authentication tokens to access. This approach ensures that unauthorized users or other third parties cannot access the endpoints which improves the security of the application. The authentication functionality is used in myPosts and myJobs functions where jobs and the posts inserted by a particular user is retrieved.

Other than this, the application allows the users to create accounts and register to the system dynamically. This is done through a registration form where details regarding the user are taken. The form fields are validated and have capabilities to add profile pictures either from the storage or the camera. The profile management is therefore can be summarized as,

- Login / Token based authentication

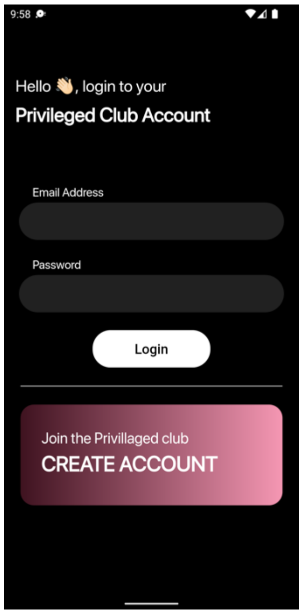
- Registration
- Edit profile
- Add Past experiences, skills etc. to the profile
- Delete profile.

In the following section we will discuss each functionality together with the associated flutter features.

2. Functionalities

Login / Token-based authentication

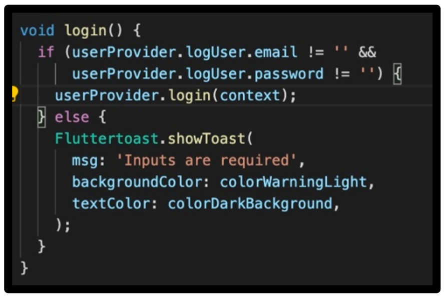
Login functionality is an essential feature of any application. It allows the user to be authorized such that the appropriate authentications can be provided. In linkup, the authorization is done



through an email password combination. When logging in, the user has to provide an email can a password which can be used in successive logins to get authorization. The interface of the login function is as follows. The interface is implemented in a way that it is easily navigable in all devices and a proper flow is maintained. White is used as the accent color in this since the app is implemented in dark mode. Labels are given properly so that the text fields are identified easily (A custom text field is implemented for the application).

The create account button is created by adding custom styling so that it manages to grab the attention of new users easily. A linear gradient is added by defining appropriate colors in the application. The structuring of the layout elements is done such that there is a proper flow in the interface. Elements such as dividers are used to properly structure the UI elements while the widths and the heights are defined by querying device height and device width. Therefore, the application is responsive for all the device instances despite different screen resolutions and aspect ratios.

When the user enters the username password it is updated to the state of the widget using an

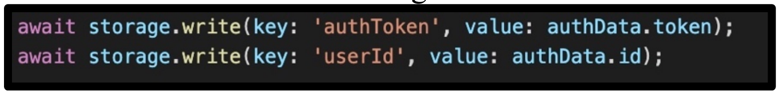


```
void login() {
  if (userProvider.logUser.email != '' &&
      userProvider.logUser.password != '') {
    userProvider.login(context);
  } else {
    Fluttertoast.showToast(
      msg: 'Inputs are required',
      backgroundColor: colorWarningLight,
      textColor: colorDarkBackground,
    );
  }
}
```

onChange function and will be passed to the login method implemented in the user provider to query the backend API. If the username password combination is correct, the backend returns an access token which is set in the flutter secure storage. This is all handled asynchronously using Futures.
If the combination is correct the user is redirected by pushing it through to the named routes and showing a toast message. If the authorization is failed a similar toast will be shown.

Flutter secure storage

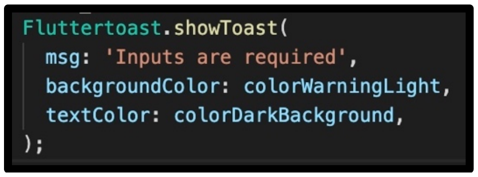
Flutter secure storage is an API that uses device storage to store application critical information securely. It uses keychain on IOS platform while keystore based solution is used on android. The values are stored as key value pairs that are encrypted. These are usually used to store information such as auth token. The set and get functions can be used to manipulate the values



```
await storage.write(key: 'authToken', value: authData.token);
await storage.write(key: 'userId', value: authData.id);
```

Flutter toast

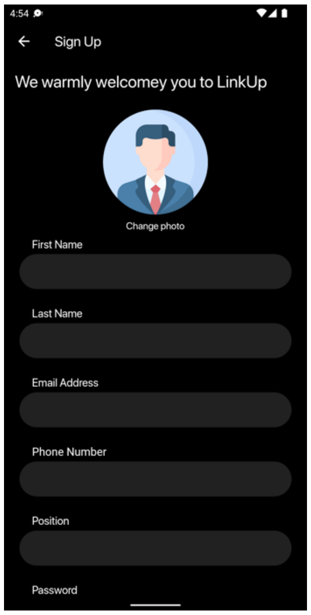
Flutter toasts are a dynamic way of providing feedback to the users, Short messages can be given to the user such as ‘Login successful’ or ‘password incorrect’. These can easily be configured.



```
Fluttertoast.showToast(
  msg: 'Inputs are required',
  backgroundColor: colorWarningLight,
  textColor: colorDarkBackground,
);
```

Registration

The registration goes hand in hand with the login functionality. This allows the user to enroll with the functionalities provided by the platform and receive the advantages. Similar to the login interface, the registration interface is properly structured with a proper flow in UI elements. Custom input fields are used here as well. Since multiple inputs are taken and it overflows from the screen, a single child scroll view is used so that the user can scroll down to the bottom elements of the interface.



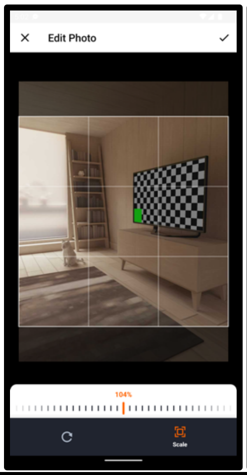
The application provides the capability of adding images when registering. This allows individuality and diversity within the users in the system. This is achieved through a combination of packages provided. There are 2 main ways the user can upload images to the application. They can either choose an existing image from the file system or take a new picture by accessing the camera. The uploaded images will be sent to the firebase storage and will be used throughout the application as the user’s profile image.

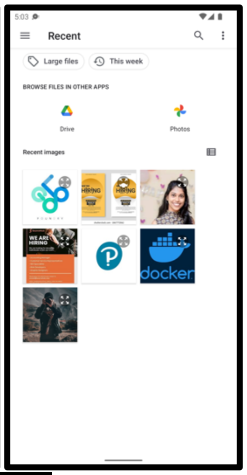
The image upload capability is achieved with a combination of ,

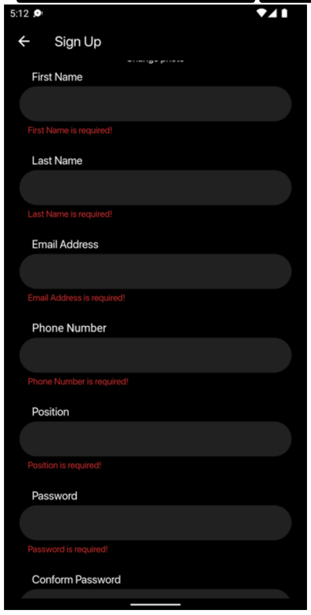
- 'package:firebase_storage/firebase_storage.dart' ,
- 'package:image_picker/image_picker.dart',
- 'package:image_cropper/image_cropper.dart'

packages.

The demonstration of the image upload capability is as follows. In there the cropper allows to scale the image, rotate, or crop to the required sizes.







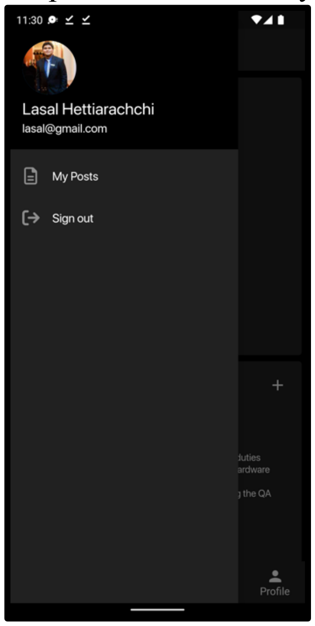
Apart from that, all input fields are validated in the registration form so that no null values are provided to the information of the application that is required from the user. Whenever the user clicks the signup button without providing all the necessary information, a red subscript will be added letting the user know that those fields are required. This feedback allows the user to see the error.

Another thing that is validated is password confirmation. When registering the user has to enter a password. When the user enters a mismatching password to the ‘password’ and ‘confirm password’ fields, the error will be shown using a toast message to the user.

When proper registration details are entered, the ‘create’ function in the user provider is called which intern sends a request to the backend application and allows the customer to be navigated to the application by pushing the named route to the application context.

Side navigation

Apart from the bottom navigation, side navigation is implemented to add navigation to extensive capabilities such as myPosts. This side navigation changes according to the logged-in user and the user profile image along with the email and the name will be displayed in the side navigation. This is done using the authid set when logging in. The backend is queried for the user using the implemented provider and all the user information is retrieved to the mobile application. The side navigation also has the logout button which allows the user to log out from the account and destroys the auth token that is created and stored in the secure storage. When logged out the application functionalities are no longer accessible until logged in again.



The side navbar is implemented using the drawer widget that is provided under flutter widgets. The drawer can be set as the drawer property of the scaffold widget which is the parent widget of most of the screen widgets in the application.

When the drawer is not required (in screens such as forms) The drawer property is not required to be provided

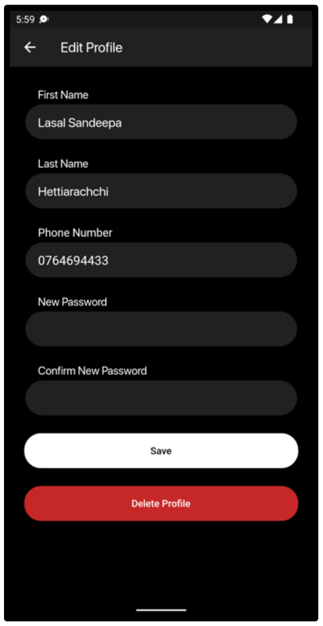
Edit profile

The users must have the capability of changing the information that is provided when registering. The edit profile functionality allows that capability. The user can go to their profile and edit their information. There are 2 types of edits a user can perform.

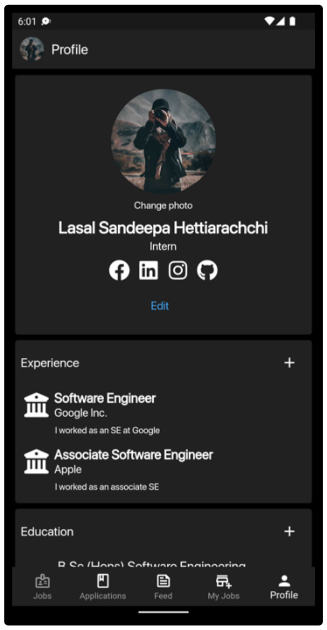
- A user can simply update their profile picture
- A user can edit other information such as name, phone number, etc.

The user can edit the profile picture by clicking the change button under the profile image. When that button is clicked, similar to when uploading, the user is asked whether to upload from the file storage or to take using the camera. The same image upload module is used here.

On the other instance, the user is able to change information that is provided at registration time such as name, phone number etc. When the form is loaded, the data that are in the backend application are fetched using the providers and set as values in the form. This information can be updated and the changed details will be sent to the backend with the endpoint accessed through



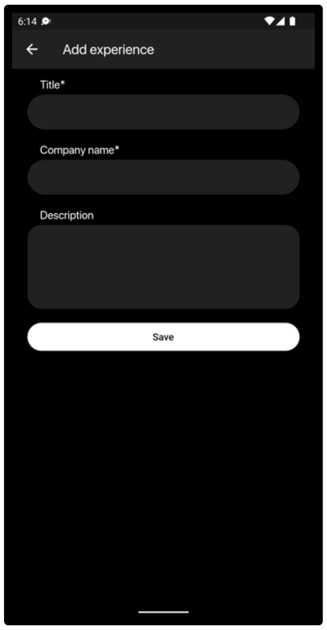
First Name
Lasal Sandeepa
Last Name
Hettiarachchi
Phone Number
0764694433
New Password
Confirm New Password



Lasal Sandeepa Hettiarachchi
Intern

Experience +
Software Engineer
Google Inc.
I worked as an SE at Google
Associate Software Engineer
Apple
I worked as an associate SE

Education +
B.Sc (Hons) Software Engineering



Title*
Company name*
Description

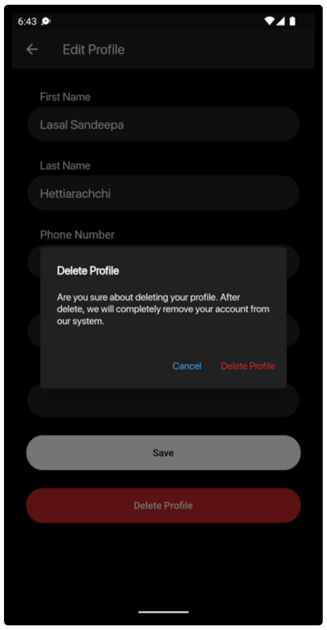
the provider.

These edit functionalities are accessed through the profile page as shown above. The profile page also displays the information of the logged-in user. This information such as the name and social handles is displayed, but the vital information is in the modules that are implemented to showcase the talent of the user along with their past experiences.
Several custom modules are written to add education, past experiences, and skills that can showcase the applicant's proficiency to the employers. The '+' icon can be used to add new information to the modules.

The experience section module allows the user to add their past experience and the organization that they have worked at. They can add the organization, the position title, and a small description of the experiences. After the experiences are added to the profile, the user can pull to refresh the page and the new experience will be displayed in the profile.

Delete profile

The user must be provided the capability to delete their profile along with all the other information registered in the application from the database. This is an extremely essential feature for any networking application given the circumstances associated with collecting user information and not deleting them when the account is deleted. The delete profile functionality can also be accessed under the edit button which the user can navigate from the profile. From there, when the user clicks on the delete button a dialog box will appear confirming the user about the deletion.



6:43

Edit Profile

First Name

Lasal Sandeepa

Last Name

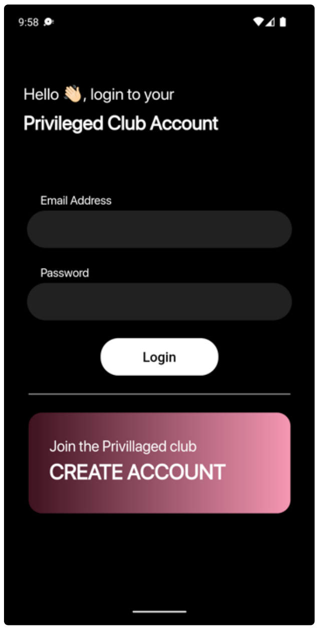
Hettiarachchi

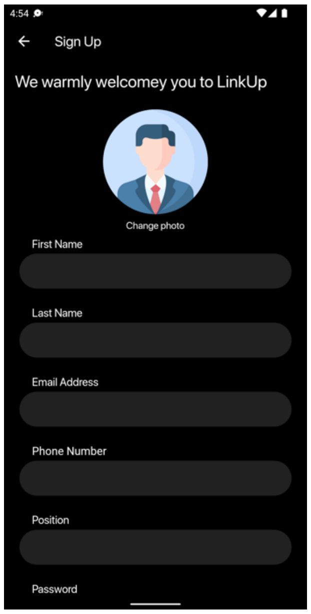
Phone Number

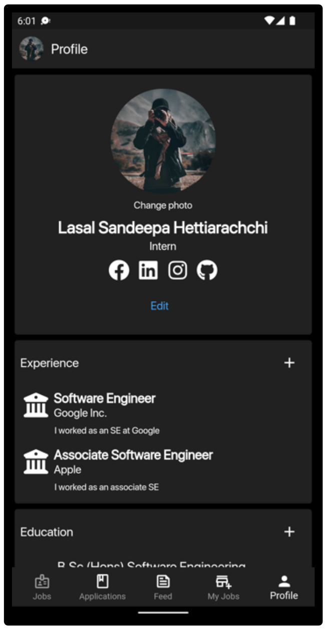
Delete Profile

Are you sure about deleting your profile. After delete, we will completely remove your account from our system.

3. User interfaces


Hello 👋, login to your
Privileged Club Account


We warmly welcomey you to LinkUp

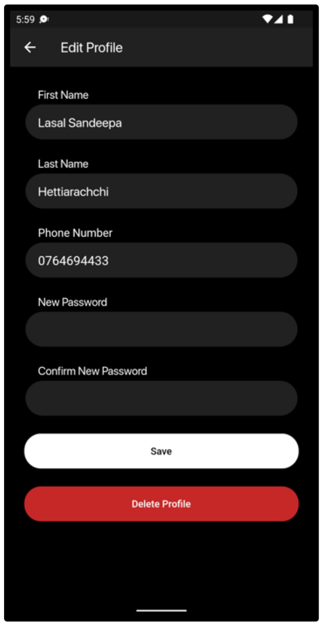

Lasal Sandeepa Hettiarachchi
Intern

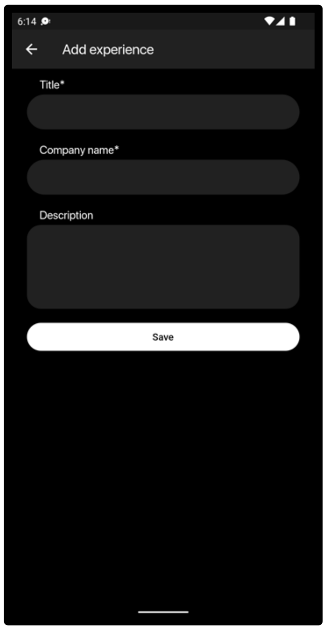
Experience

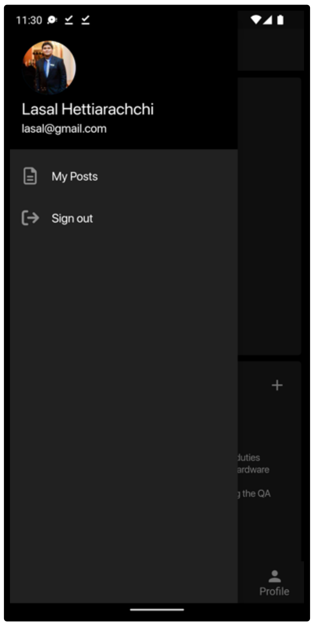
Software Engineer
Google Inc.
I worked as an SE at Google

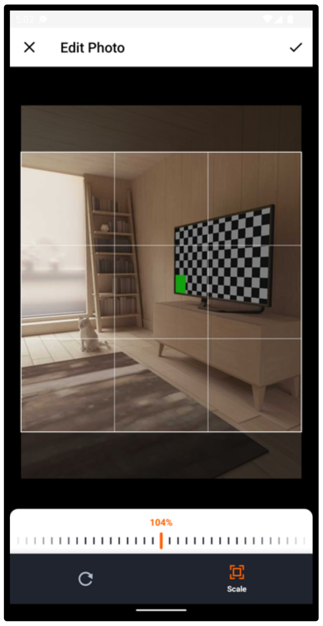
Associate Software Engineer
Apple
I worked as an associate SE

Education
B.Sc (Hons) Software Engineering


First Name: Lasal Sandeepa
Last Name: Hettiarachchi
Phone Number: 0764694433

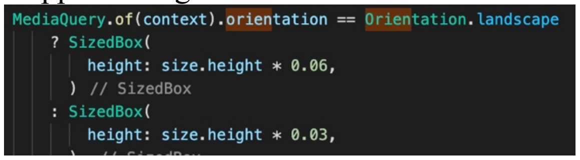



Lasal Hettiarachchi
lasal@gmail.com



Responsiveness of the interface

All the interfaces are developed as responsive as possible. In some interfaces where both horizontal and vertical orientations are viable, the height and the width of the UI elements are given accordingly. This is done by querying the device height and width in each orientation and conditionally providing the dimensions of the UI elements based on the data. The following code snippets will give an intuition on to how that is achieved.



```
MediaQuery.of(context).orientation == Orientation.landscape
    ? SizedBox(
        height: size.height * 0.06,
      ) // SizedBox
    : SizedBox(
        height: size.height * 0.03,
```

4. Flutter features used

The following special features in the Flutter framework and 3rd party libraries in pub dev were used. The following list provides the references corresponding to each feature and where they were used.

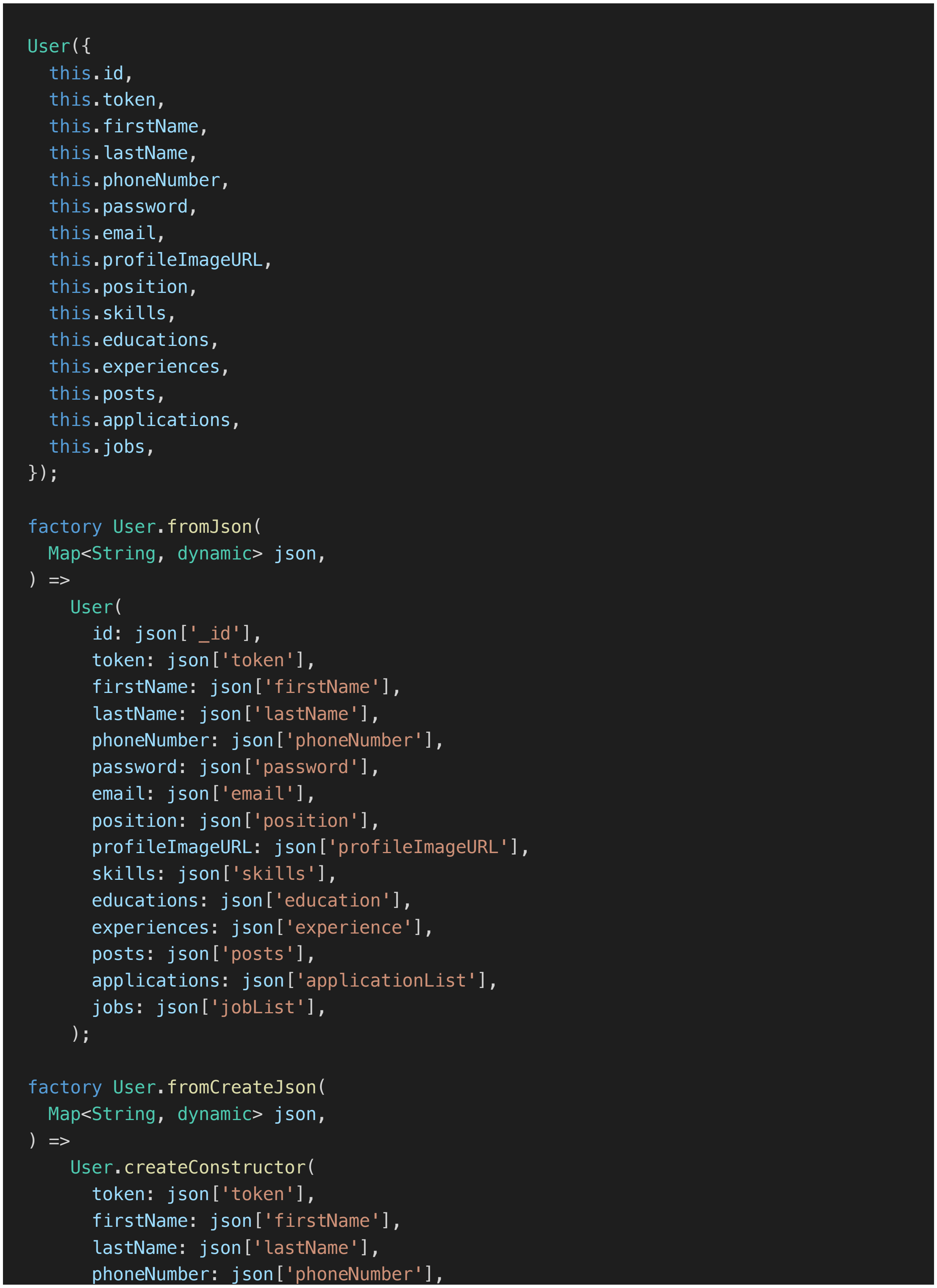
- Image upload
  - Instances used: Registration, Edit profile.
  - References :
    - https://firebase.google.com/docs/flutter/setup?platform=android
    - https://pub.dev/packages/firebase
    - https://pub.dev/packages/firebase_storage
    - https://pub.dev/packages/image_cropper
    - https://pub.dev/packages/image_picker
    - https://pub.dev/packages/image_cropper
- Toast
  - Instances used: Login , registration (providing feedback)
  - References:
    - https://pub.dev/packages/fluttertoast
- Dialog box
  - Instances used: Delete profile
  - References:
    - https://www.youtube.com/watch?v=75CsnyRXf5I
    - https://api.flutter.dev/flutter/material/AlertDialog-class.html
- Drawer
  - Instances used: Side navigation of scaffold
  - References:
    - https://docs.flutter.dev/cookbook/design/drawer
    - https://www.youtube.com/watch?v=WRj86iHihgY
- Pull to refresh
  - Instances used: Profile , to refresh experiences and other modules
  - References:
    - https://pub.dev/packages/pull_to_refresh
- Secure storage
  - Instances used: User authentication
  - References:
    - https://pub.dev/packages/flutter_secure_storage

5. Code

Models

User_model.dart

```dart
class User {
  String id;
  String token;
  String firstName;
  String lastName;
  String phoneNumber;
  String password;
  String email;
  String position;
  String profileImageURL;
  List<dynamic> skills;
  List<dynamic> educations;
  List<dynamic> experiences;
  List<dynamic> posts;
  List<dynamic> applications;
  List<dynamic> jobs;

  User.createConstructor({
    this.firstName,
    this.lastName,
    this.phoneNumber,
    this.email,
    this.position,
    this.password,
    this.profileImageURL,
    this.token,
  });

  User.updateConstructor({
    this.firstName,
    this.lastName,
    this.phoneNumber,
    this.email,
    this.position,
    this.password,
  });

  User.loginConstructor({
    this.email,
    this.password,
  });
```



```dart
User({
  this.id,
  this.token,
  this.firstName,
  this.lastName,
  this.phoneNumber,
  this.password,
  this.email,
  this.profileImageURL,
  this.position,
  this.skills,
  this.educations,
  this.experiences,
  this.posts,
  this.applications,
  this.jobs,
});

factory User.fromJson(
  Map<String, dynamic> json,
) =>
    User(
      id: json['_id'],
      token: json['token'],
      firstName: json['firstName'],
      lastName: json['lastName'],
      phoneNumber: json['phoneNumber'],
      password: json['password'],
      email: json['email'],
      position: json['position'],
      profileImageURL: json['profileImageURL'],
      skills: json['skills'],
      educations: json['education'],
      experiences: json['experience'],
      posts: json['posts'],
      applications: json['applicationList'],
      jobs: json['jobList'],
    );

factory User.fromCreateJson(
  Map<String, dynamic> json,
) =>
    User.createConstructor(
      token: json['token'],
      firstName: json['firstName'],
      lastName: json['lastName'],
      phoneNumber: json['phoneNumber'],
        position: json['position'],
        password: json['password'],
        email: json['email'],
        profileImageURL: json['profileImageURL'],
      );

  factory User.fromModifyJson(
    Map<String, String> json,
  ) =>
      User.updateConstructor(
        firstName: json['firstName'],
        lastName: json['lastName'],
        phoneNumber: json['phoneNumber'],
        position: json['position'],
        password: json['password'],
        email: json['email'],
      );

  Map<String, dynamic> toJson() => {
        'id': id,
        'firstName': firstName,
        'lastName': lastName,
        'phoneNumber': phoneNumber,
        'password': password,
        'email': email,
        'position': position,
        'profileImageURL': profileImageURL,
        'skills': skills,
        'educations': educations,
        'experiences': experiences,
        'applications': applications,
        'jobs': jobs,
      };
}
```

experience_model.dart

```
class Experience {
```

```dart
  String id;
  String position;
  String companyName;
  String description;

  Experience({
    this.id,
    this.companyName,
    this.position,
    this.description,
  });

  factory Experience.fromJson(
    Map<String, dynamic> json,
  ) =>
      Experience(
        id: json['_id'],
        position: json['position'],
        companyName: json['companyName'],
        description: json['description'],
      );

  Map<String, dynamic> toJson() => {
        '_id': id,
        'position': position,
        'companyName': companyName,
        'description': description,
      };
}
```

Components

experience_card.dart

```dart
import "package:flutter/material.dart";
import 'package:font_awesome_flutter/font_awesome_flutter.dart';
import 'package:linkup/constants.dart';

class ExperienceCard extends StatelessWidget {
  final String position;
  final String companyName;
  final String description;

  const ExperienceCard({
    Key key,
    this.companyName,
    this.position,
    this.description,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    final size = MediaQuery.of(context).size;
    final orientation = MediaQuery.of(context).orientation;

    return Container(
      width: size.width,
      padding: const EdgeInsets.only(top: 15, left: 5),
      child: Column(
        children: [
          Row(
            children: [
              const Icon(
                FontAwesomeIcons.bank,
                size: 35,
                color: colorTextPrimary,
              ),
              Padding(
                padding: const EdgeInsets.only(left: 8),
                child: Column(
                  crossAxisAlignment: CrossAxisAlignment.start,
                  children: [
                    Text(
                      position,
                      style: const TextStyle(
                        fontFamily: fontFamilySFPro,
                        fontSize: 19,
                        fontWeight: FontWeight.bold,
                        color: colorTextPrimary,
                      ),
                    ),
                    Text(
                      companyName,
                      style: const TextStyle(
                        fontFamily: fontFamilySFPro,
                        fontSize: 16,
                        color: colorTextPrimary,
                      ),
                    ),
                  ],
                ),
              )
            ],
          ),
          Row(
            children: [
              if (description != null)
                Container(
                  padding: const EdgeInsets.only(left: 44, top: 8),
                  width: orientation == Orientation.landscape
                      ? size.width * 0.65
                      : size.width * 0.85,
                  child: Text(
                    description,
                    style: const TextStyle(
                      fontFamily: fontFamilySFPro,
                      fontSize: 13,
                      color: colorTextPrimary,
                    ),
                  ),
                ),
            ],
          )
        ],
      ),
    );
  }
}
```

Image_upload.dart

```
import 'dart:io';
```

```dart
import 'package:firebase_core/firebase_core.dart';
import "package:flutter/material.dart";
import 'package:image_cropper/image_cropper.dart';
import 'package:image_picker/image_picker.dart';
import 'package:linkup/components/user_image_upload.dart';
import 'package:linkup/constants.dart';
import "package:path/path.dart" as p;
import 'package:firebase_storage/firebase_storage.dart' as storage;

class ImageUpload extends StatefulWidget {
  final Function(String imageURL) onFileChanged;

  const ImageUpload({
    Key key,
    this.onFileChanged,
  }) : super(key: key);

  @override
  _ImageUploadState createState() => _ImageUploadState();
}

class _ImageUploadState extends State<ImageUpload> {
  final ImagePicker _picker = ImagePicker();
  String imagePath;

  @override
  Widget build(BuildContext context) {
    return Column(
      children: [
        if (imagePath == null)
          Container(
            child: const Icon(
              Icons.camera,
              size: 150,
              color: colorDarkForground,
            ),
          ),
        if (imagePath != null)
          InkWell(
            splashColor: Colors.transparent,
            highlightColor: Colors.transparent,
            onTap: () => _selectPhoto(),
            child: AppRoundedImage.url(
              imagePath,
              height: 150,
              width: 150,
            ),
          ),
        InkWell(
          onTap: () => _selectPhoto(),
          child: Padding(
            padding: const EdgeInsets.all(8.0),
            child: Text(
              imagePath != null ? "Change photo" : "Select photo",
              style: const TextStyle(
                fontFamily: fontFamilySFPro,
                color: Colors.white,
              ),
            ),
          ),
        )
      ],
    );
  }

  Future _selectPhoto() async {
    await showModalBottomSheet(
      context: context,
      builder: (context) => BottomSheet(
        builder: (context) => Column(
          mainAxisSize: MainAxisSize.min,
          children: [
            ListTile(
              horizontalTitleGap: 0,
              leading: const Icon(Icons.camera_alt_outlined),
              title: const Text(
                "Camera",
                style: TextStyle(
                  fontFamily: fontFamilySFPro,
                ),
              ),
              onTap: () {
                Navigator.of(context).pop();
                _pickImage(ImageSource.camera);
              },
            ),
            ListTile(
              horizontalTitleGap: 0,
              leading: const Icon(Icons.folder_outlined),
              title: const Text(
                "Pick a photo",
                style: TextStyle(
                  fontFamily: fontFamilySFPro,
                ),
              ),
              onTap: () {
                Navigator.of(context).pop();
                _pickImage(ImageSource.gallery);
              },
            ),
          ],
        ),
        onClosing: () {},
      ),
    );
  }

  Future _pickImage(ImageSource source) async {
    final pickerFile =
        await _picker.pickImage(source: source, imageQuality: 50);
    if (pickerFile == null) {
      return;
    }

    var file = await ImageCropper().cropImage(
      sourcePath: pickerFile.path,
      aspectRatio: const CropAspectRatio(
        ratioX: 1,
        ratioY: 1,
      ),
    );
    if (file == null) {
      return;
    }

    await _uploadFile(file.path);
  }

  Future _uploadFile(String path) async {
    await Firebase.initializeApp();
    final ref = storage.FirebaseStorage.instance
        .ref()
        .child("images")
        .child(DateTime.now().toIso8601String() + p.basename(path));

    final result = await ref.putFile(File(path));
    final fileUrl = await result.ref.getDownloadURL();
    print("Test URL: " + fileUrl);

    setState(() {
      imagePath = fileUrl;
    });

    widget.onFileChanged(fileUrl);
  }
}
```

User_image_upload.dart

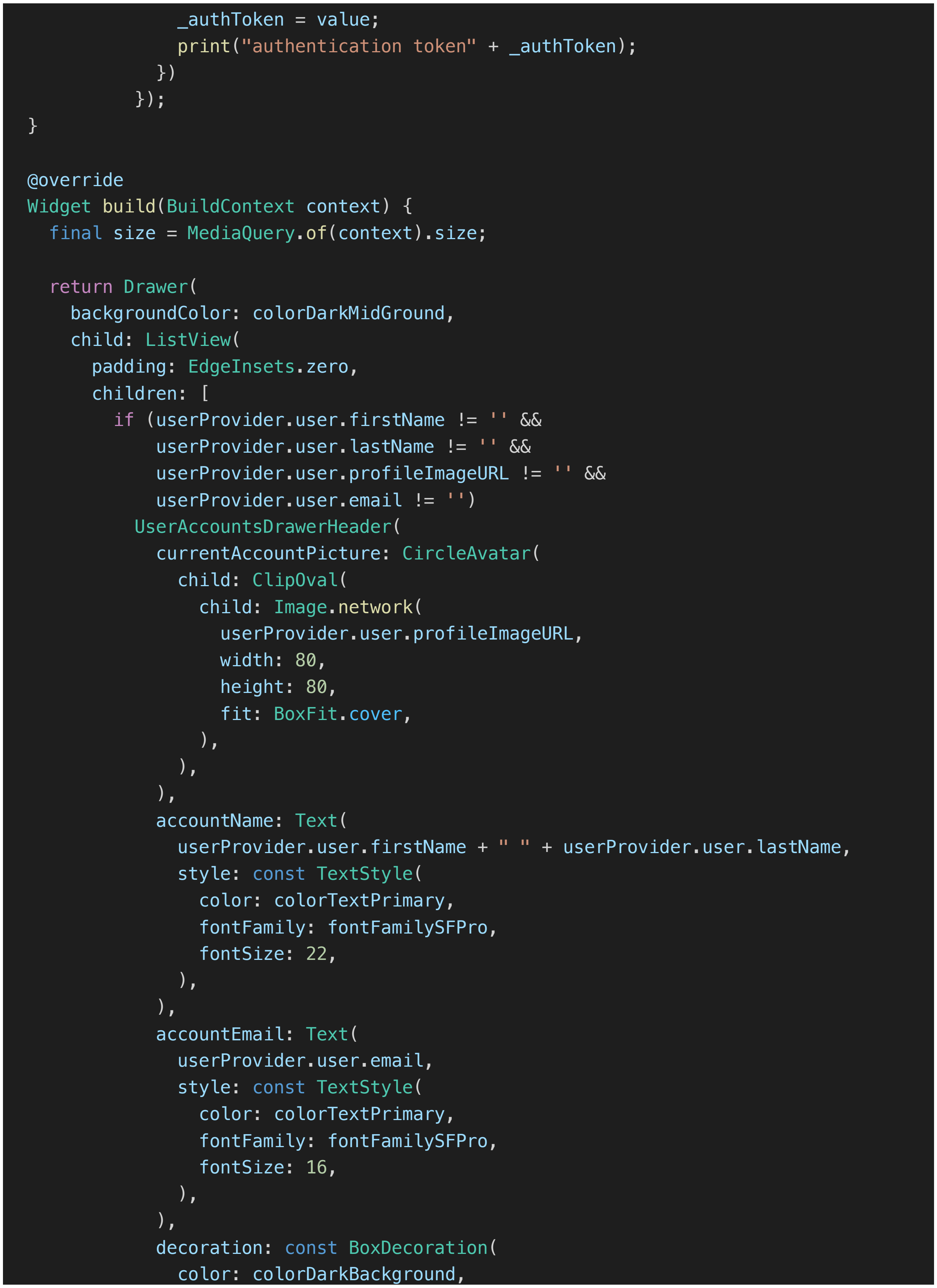
```
import 'dart:io';
import 'dart:typed_data';

import 'package:firebase_core/firebase_core.dart';
import "package:flutter/material.dart";
import 'package:image_cropper/image_cropper.dart';
import 'package:image_picker/image_picker.dart';
import 'package:linkup/constants.dart';
import "package:path/path.dart" as p;
import 'package:firebase_storage/firebase_storage.dart' as storage;

class UserImageUpload extends StatefulWidget {
  final Function(String imageURL) onFileChanged;
  String imageURL;

  UserImageUpload({
    Key key,
    this.imageURL,
    this.onFileChanged,
  }) : super(key: key);

  @override
  _UserImageUploadState createState() => _UserImageUploadState();
}

class _UserImageUploadState extends State<UserImageUpload> {
  final ImagePicker _picker = ImagePicker();

  @override
  Widget build(BuildContext context) {
    return Column(
      children: [
        if (widget.imageURL == null)
          Container(
            decoration: BoxDecoration(
              border: Border.all(width: 2, color: colorDarkForground),
              borderRadius: BorderRadius.circular(100),
            ),
            child: const Icon(
              Icons.person_rounded,
              size: 150,
              color: colorDarkForground,
          ),
        ),
      if (widget.imageURL != null)
        InkWell(
          splashColor: Colors.transparent,
          highlightColor: Colors.transparent,
          onTap: () => _selectPhoto(),
          child: AppRoundedImage.url(
            widget.imageURL,
            height: 150,
            width: 150,
          ),
        ),
      InkWell(
        onTap: () => _selectPhoto(),
        child: Padding(
          padding: const EdgeInsets.all(8.0),
          child: Text(
            widget.imageURL != null ? "Change photo" : "Select photo",
            style: const TextStyle(
              fontFamily: fontFamilySFPro,
              color: Colors.white,
            ),
          ),
        ),
      )
    ],
  );
}

Future _selectPhoto() async {
  await showModalBottomSheet(
    context: context,
    builder: (context) => BottomSheet(
      builder: (context) => Column(
        mainAxisSize: MainAxisSize.min,
        children: [
          ListTile(
            horizontalTitleGap: 0,
            leading: const Icon(Icons.camera_alt_outlined),
            title: const Text(
              "Camera",
              style: TextStyle(
                fontFamily: fontFamilySFPro,
              ),
            ),
            onTap: () {
              Navigator.of(context).pop();
```

```dart
              _pickImage(ImageSource.camera);
            },
          ),
          ListTile(
            horizontalTitleGap: 0,
            leading: const Icon(Icons.folder_outlined),
            title: const Text(
              "Pick a photo",
              style: TextStyle(
                fontFamily: fontFamilySFPro,
              ),
            ),
            onTap: () {
              Navigator.of(context).pop();
              _pickImage(ImageSource.gallery);
            },
          ),
        ],
      ),
      onClosing: () {},
    ),
  );
}

Future _pickImage(ImageSource source) async {
  final pickerFile =
      await _picker.pickImage(source: source, imageQuality: 50);
  if (pickerFile == null) {
    return;
  }

  var file = await ImageCropper().cropImage(
    sourcePath: pickerFile.path,
    aspectRatio: const CropAspectRatio(
      ratioX: 1,
      ratioY: 1,
    ),
  );
  if (file == null) {
    return;
  }

  await _uploadFile(file.path);
}

Future _uploadFile(String path) async {
  await Firebase.initializeApp();
  final ref = storage.FirebaseStorage.instance
```

```dart
        .ref()
        .child("images")
        .child(DateTime.now().toIso8601String() + p.basename(path));

    final result = await ref.putFile(File(path));
    final fileUrl = await result.ref.getDownloadURL();

    setState(() {
      widget.imageURL = fileUrl;
    });

    widget.onFileChanged(fileUrl);
  }
}

class AppRoundedImage extends StatelessWidget {
  final ImageProvider provider;
  final double height;
  final double width;

  const AppRoundedImage(
    this.provider, {
    Key key,
    this.height,
    this.width,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return ClipRRect(
      borderRadius: BorderRadius.circular(height / 2),
      child: Image(
        image: provider,
        height: height,
        width: width,
      ),
    );
  }

  factory AppRoundedImage.url(
    String url, {
    double height,
    double width,
  }) {
    return AppRoundedImage(
      NetworkImage(url),
      height: height,
      width: width,
    );
  }

  factory AppRoundedImage.memory(
    Uint8List data, {
    double height,
    double width,
  }) {
    return AppRoundedImage(
      MemoryImage(data),
      height: height,
      width: width,
    );
  }
}
```

Side_navbar.dart

```
import "package:flutter/material.dart";
import 'package:flutter_secure_storage/flutter_secure_storage.dart';
import 'package:font_awesome_flutter/font_awesome_flutter.dart';
import 'package:linkup/constants.dart';
import 'package:linkup/providers/user_provider.dart';
import 'package:provider/provider.dart';

class SideNavbar extends StatefulWidget {
  const SideNavbar({Key key}) : super(key: key);

  @override
  _SideNavbarState createState() => _SideNavbarState();
}

class _SideNavbarState extends State<SideNavbar> {
  UserProvider userProvider;
  String _authToken;

  @override
  void initState() {
    super.initState();
    userProvider = context.read<UserProvider>();
    context
        .read<UserProvider>()
        .storage
        .read(key: 'authToken')
        .then((value) => {
              setState(() {
```



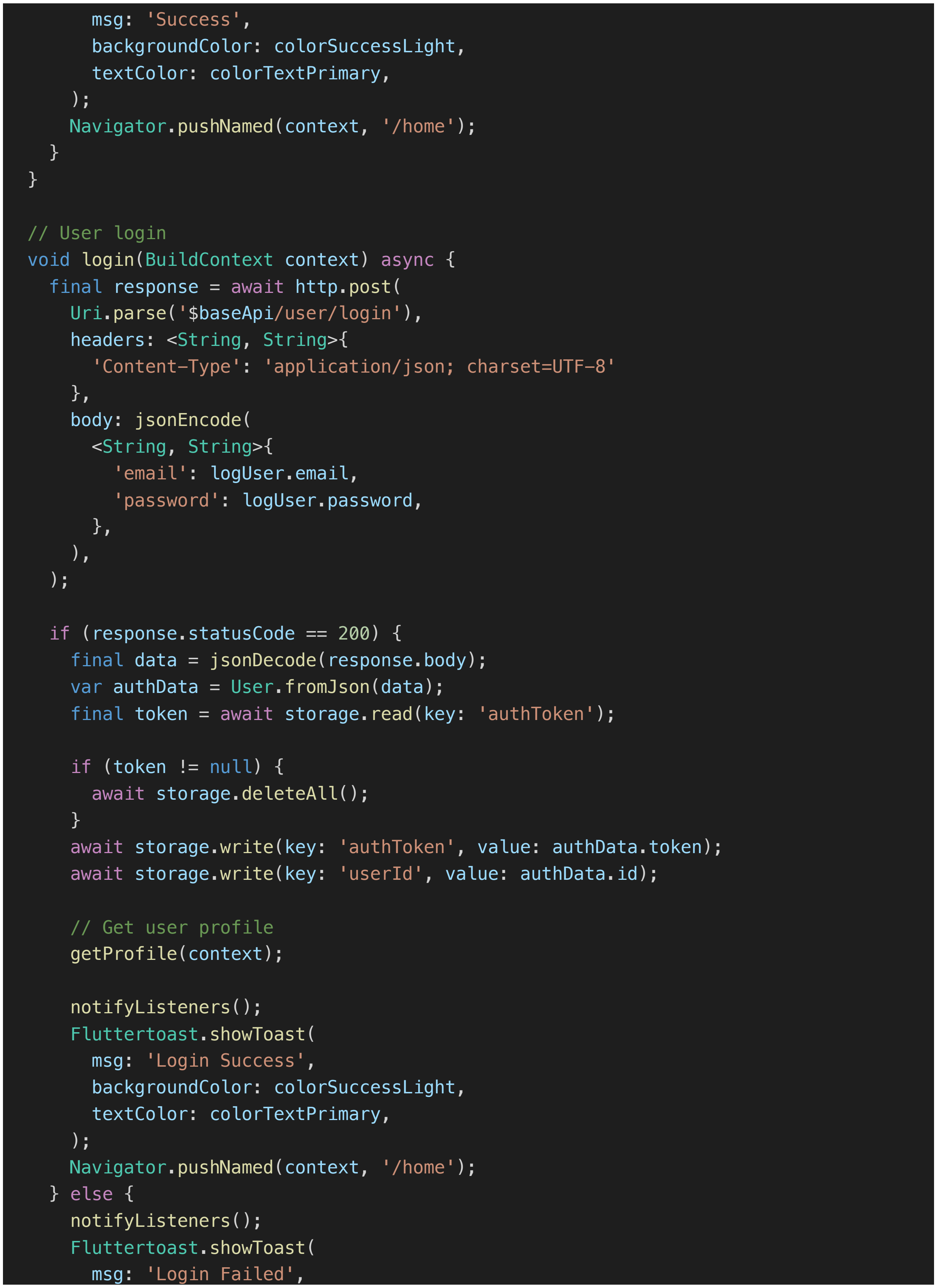
```
              _authToken = value;
              print("authentication token" + _authToken);
            })
          });
}

@override
Widget build(BuildContext context) {
  final size = MediaQuery.of(context).size;

  return Drawer(
    backgroundColor: colorDarkMidGround,
    child: ListView(
      padding: EdgeInsets.zero,
      children: [
        if (userProvider.user.firstName != '' &&
            userProvider.user.lastName != '' &&
            userProvider.user.profileImageURL != '' &&
            userProvider.user.email != '')
          UserAccountsDrawerHeader(
            currentAccountPicture: CircleAvatar(
              child: ClipOval(
                child: Image.network(
                  userProvider.user.profileImageURL,
                  width: 80,
                  height: 80,
                  fit: BoxFit.cover,
                ),
              ),
            ),
            accountName: Text(
              userProvider.user.firstName + " " + userProvider.user.lastName,
              style: const TextStyle(
                color: colorTextPrimary,
                fontFamily: fontFamilySFPro,
                fontSize: 22,
              ),
            ),
            accountEmail: Text(
              userProvider.user.email,
              style: const TextStyle(
                color: colorTextPrimary,
                fontFamily: fontFamilySFPro,
                fontSize: 16,
              ),
            ),
            decoration: const BoxDecoration(
              color: colorDarkBackground,
            ),
          ),
        if (_authToken == null)
          const SizedBox(
            height: 25,
          ),
        if (_authToken == null)
          _NavbarItem(
            text: "Sign Up",
            onClick: () {
              Navigator.pop(context);
              Navigator.pushNamed(context, "/signup");
            },
            icon: FontAwesomeIcons.user,
          ),
        if (_authToken == null)
          _NavbarItem(
            text: "Login",
            onClick: () {
              Navigator.pop(context);
              Navigator.pushNamed(context, "/login");
            },
            icon: FontAwesomeIcons.signIn,
          ),
        if (_authToken != null)
          _NavbarItem(
            text: "My Posts",
            onClick: () {
              Navigator.pop(context);
              Navigator.pushNamed(context, "/my-posts");
            },
            icon: FontAwesomeIcons.fileLines,
          ),
        if (_authToken != null)
          Align(
            alignment: Alignment.bottomCenter,
            child: _NavbarItem(
              text: "Sign out",
              onClick: () {
                context.read<UserProvider>().storage.deleteAll();
                Navigator.pushNamedAndRemoveUntil(
                  context,
                  "/login",
                  (Route<dynamic> route) => false,
                );
              },
              icon: FontAwesomeIcons.signOut,
            ),
```

```dart
            ),
          ],
        ),
      );
  }
}

class _NavbarItem extends StatelessWidget {
  final String text;
  final VoidCallback onClick;
  final IconData icon;

  const _NavbarItem({
    this.text,
    this.onClick,
    this.icon,
  });

  @override
  Widget build(BuildContext context) {
    return ListTile(
      iconColor: colorPrimary,
      leading: Icon(
        icon,
        size: 23,
        color: colorTextDisabled,
      ),
      horizontalTitleGap: 0,
      title: Text(
        text,
        style: const TextStyle(
          fontFamily: fontFamilySFPro,
          fontSize: 16,
          color: colorTextPrimary,
        ),
      ),
      onTap: onClick,
    );
  }
}
```

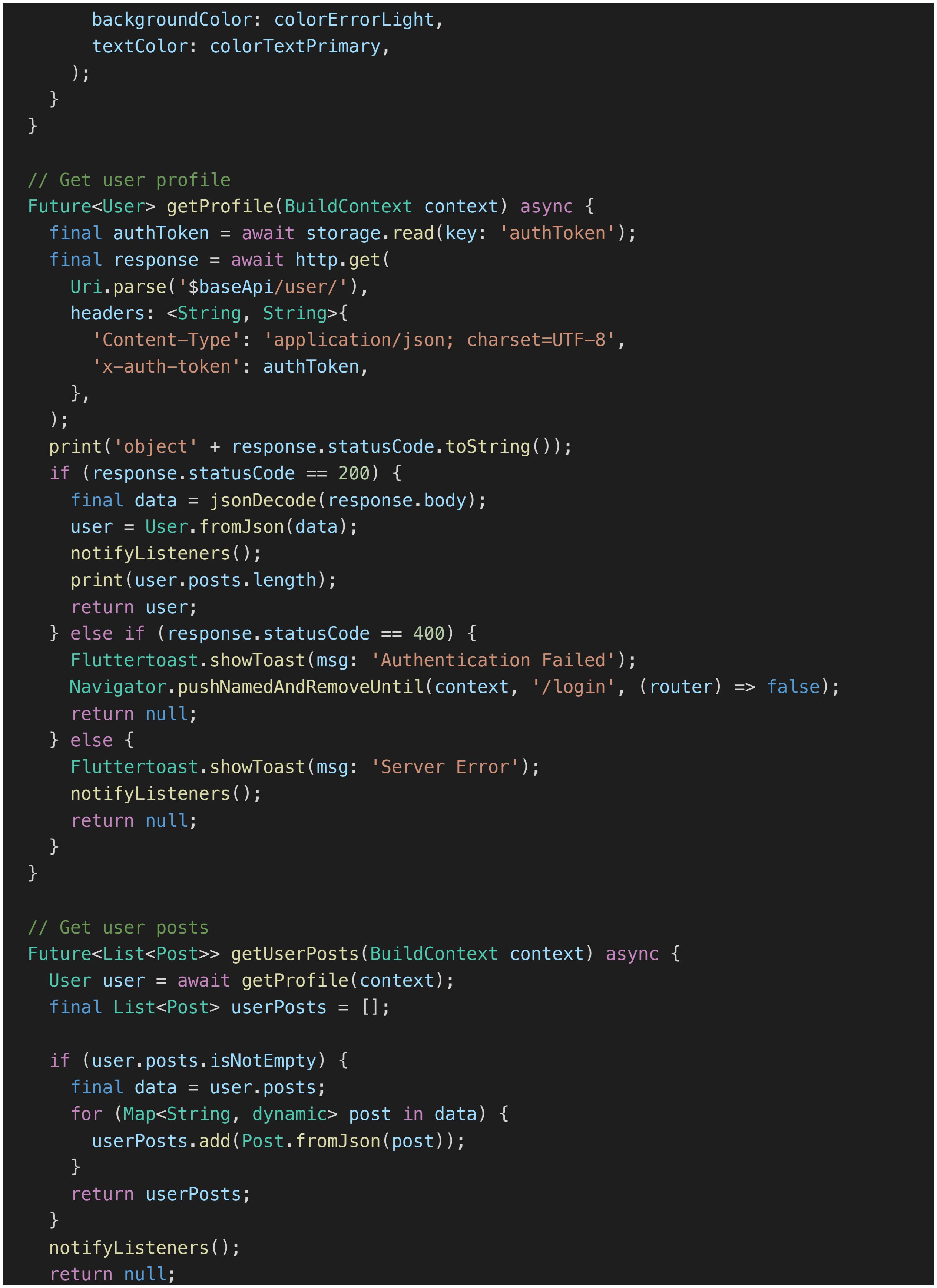
Providers

User_provider.dart

```dart
import 'dart:convert';

import 'package:flutter/material.dart';
import 'package:flutter_secure_storage/flutter_secure_storage.dart';
import 'package:fluttertoast/fluttertoast.dart';
import 'package:linkup/constants.dart';
import 'package:linkup/models/job_model.dart';
import 'package:linkup/models/post_model.dart';
import 'package:linkup/models/user_model.dart';
import 'package:http/http.dart' as http;

class UserProvider extends ChangeNotifier {
  User newUser = User.createConstructor(
    firstName: '',
    lastName: '',
    phoneNumber: '',
    email: '',
    position: '',
    password: '',
    profileImageURL:
        'https://firebasestorage.googleapis.com/v0/b/linkup-
31422.appspot.com/o/images%2Fuser_profile_default.png?alt=media&token=c2575581-3695-
44fa-a30e-02f795f6f669',
  );
  User modifyUser = User.updateConstructor(
    firstName: '',
    lastName: '',
    phoneNumber: '',
    email: '',
    position: '',
    password: '',
  );
  User logUser = User.loginConstructor(
    email: '',
    password: '',
  );
  User user = User(
    firstName: '',
    lastName: '',
    phoneNumber: '',
    email: '',
    password: '',
    profileImageURL: '',
    token: '',
    position: '',
    applications: [],
    educations: [],
    experiences: [],
    id: '',
    jobs: [],
    posts: [],
    skills: [],
  );
  final storage = const FlutterSecureStorage();

  // Create new user profile
  void create(BuildContext context) async {
    final response = await http.post(
      Uri.parse('$baseApi/user/register'),
      headers: <String, String>{
        'Content-Type': 'application/json; charset=UTF-8'
      },
      body: jsonEncode(
        <String, String>{
          'firstName': newUser.firstName,
          'lastName': newUser.lastName,
          'phoneNumber': newUser.phoneNumber,
          'password': newUser.password,
          'position': newUser.position,
          'email': newUser.email,
          'profileImageURL': newUser.profileImageURL,
        },
      ),
    );

    if (response.statusCode == 200) {
      final data = jsonDecode(response.body);
      var authData = User.fromCreateJson(data);
      final token = await storage.read(key: 'authToken');

      if (token != null) {
        await storage.deleteAll();
      }
      await storage.write(key: 'authToken', value: authData.token);
      await storage.write(key: 'userId', value: authData.id);

      // Get user profile
      getProfile(context);

      notifyListeners();
      Fluttertoast.showToast(
```



```dart
      msg: 'Success',
      backgroundColor: colorSuccessLight,
      textColor: colorTextPrimary,
    );
    Navigator.pushNamed(context, '/home');
  }
}

// User login
void login(BuildContext context) async {
  final response = await http.post(
    Uri.parse('$baseApi/user/login'),
    headers: <String, String>{
      'Content-Type': 'application/json; charset=UTF-8'
    },
    body: jsonEncode(
      <String, String>{
        'email': logUser.email,
        'password': logUser.password,
      },
    ),
  );

  if (response.statusCode == 200) {
    final data = jsonDecode(response.body);
    var authData = User.fromJson(data);
    final token = await storage.read(key: 'authToken');

    if (token != null) {
      await storage.deleteAll();
    }
    await storage.write(key: 'authToken', value: authData.token);
    await storage.write(key: 'userId', value: authData.id);

    // Get user profile
    getProfile(context);

    notifyListeners();
    Fluttertoast.showToast(
      msg: 'Login Success',
      backgroundColor: colorSuccessLight,
      textColor: colorTextPrimary,
    );
    Navigator.pushNamed(context, '/home');
  } else {
    notifyListeners();
    Fluttertoast.showToast(
      msg: 'Login Failed',
```



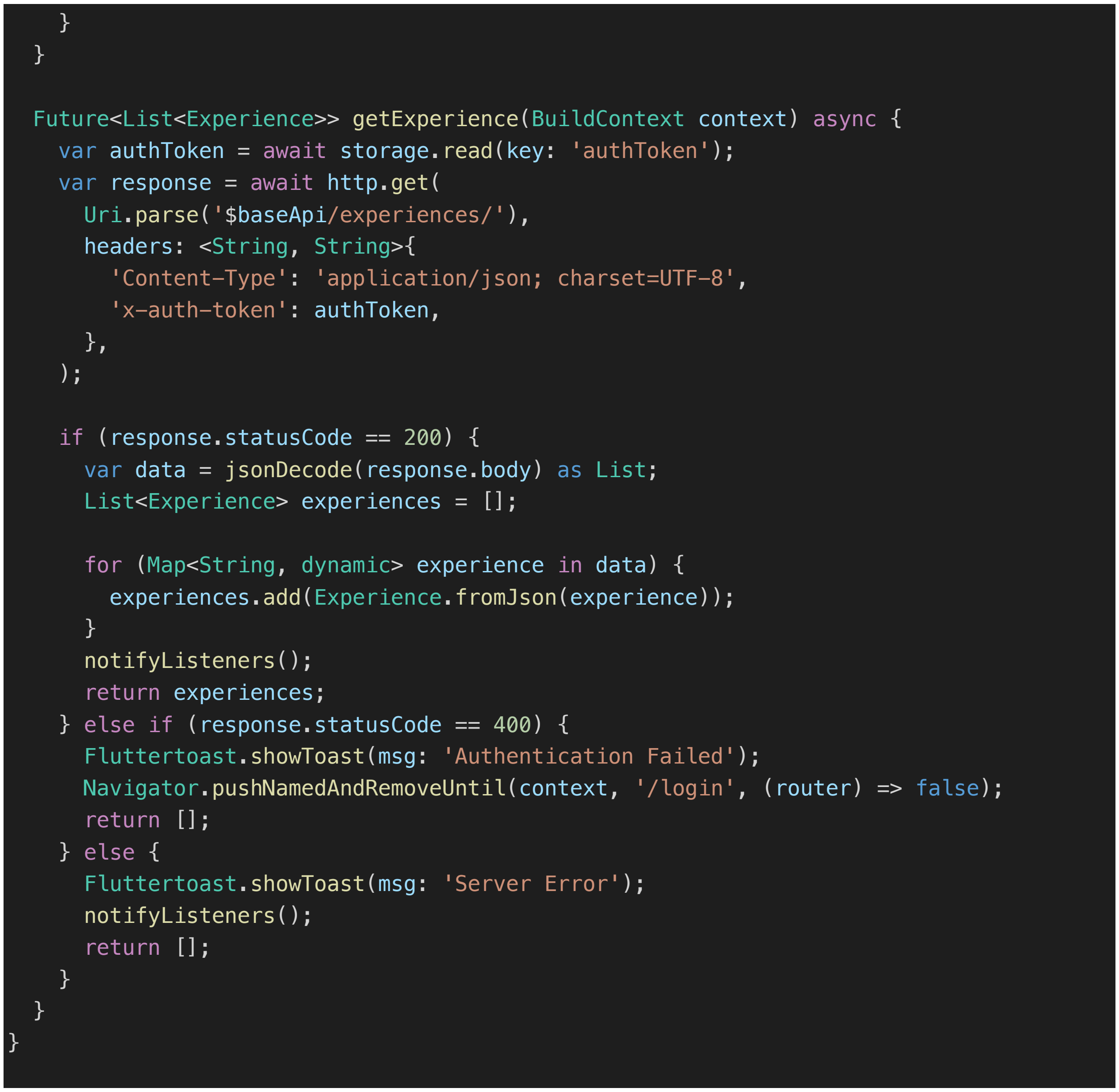
```dart
      backgroundColor: colorErrorLight,
      textColor: colorTextPrimary,
    );
  }
}

// Get user profile
Future<User> getProfile(BuildContext context) async {
  final authToken = await storage.read(key: 'authToken');
  final response = await http.get(
    Uri.parse('$baseApi/user/'),
    headers: <String, String>{
      'Content-Type': 'application/json; charset=UTF-8',
      'x-auth-token': authToken,
    },
  );
  print('object' + response.statusCode.toString());
  if (response.statusCode == 200) {
    final data = jsonDecode(response.body);
    user = User.fromJson(data);
    notifyListeners();
    print(user.posts.length);
    return user;
  } else if (response.statusCode == 400) {
    Fluttertoast.showToast(msg: 'Authentication Failed');
    Navigator.pushNamedAndRemoveUntil(context, '/login', (router) => false);
    return null;
  } else {
    Fluttertoast.showToast(msg: 'Server Error');
    notifyListeners();
    return null;
  }
}

// Get user posts
Future<List<Post>> getUserPosts(BuildContext context) async {
  User user = await getProfile(context);
  final List<Post> userPosts = [];

  if (user.posts.isNotEmpty) {
    final data = user.posts;
    for (Map<String, dynamic> post in data) {
      userPosts.add(Post.fromJson(post));
    }
    return userPosts;
  }
  notifyListeners();
  return null;
```

```
}

// Get user posts
Future<List<Job>> getUserJobs(BuildContext context) async {
  User user = await getProfile(context);
  final List<Job> userJobs = [];

  if (user.jobs.isNotEmpty) {
    final data = user.jobs;

    for (Map<String, dynamic> job in data) {
      userJobs.add(Job.fromJson(job));
      print(Job.fromJson(job).companyName);
    }
    return userJobs;
  }
  notifyListeners();
  return null;
}

// Update user profile image
void updateProfileImage(BuildContext context) async {
  final authToken = await storage.read(key: 'authToken');
  final response = await http.put(
    Uri.parse('$baseApi/user/edit'),
    headers: <String, String>{
      'Content-Type': 'application/json; charset=UTF-8',
      'x-auth-token': authToken,
    },
    body: jsonEncode(
      <String, String>{
        'Id': user.id,
        'profileImageURL': user.profileImageURL,
      },
    ),
  );

  if (response.statusCode == 200) {
    // Get updated user profile
    getProfile(context);

    notifyListeners();
    Fluttertoast.showToast(
      msg: 'Update Success',
      backgroundColor: colorSuccess,
      textColor: colorTextPrimary,
    );
  } else if (response.statusCode == 400) {
```

```dart
      Fluttertoast.showToast(
        msg: 'Authentication Failed',
        backgroundColor: colorError,
        textColor: colorTextPrimary,
      );
      notifyListeners();
    } else {
      Fluttertoast.showToast(
        msg: 'Server Error',
        backgroundColor: colorError,
        textColor: colorTextPrimary,
      );
      notifyListeners();
    }
  }

  // Update user profile
  Future<User> updateUser(BuildContext context) async {
    final authToken = await storage.read(key: 'authToken');
    final response = await http.put(
      Uri.parse('$baseApi/user/edit'),
      headers: <String, String>{
        'Content-Type': 'application/json; charset=UTF-8',
        'x-auth-token': authToken,
      },
      body: jsonEncode(
        <String, String>{
          'Id': modifyUser.id,
          'firstName': modifyUser.firstName,
          'lastName': modifyUser.lastName,
          'phoneNumber': modifyUser.phoneNumber,
          'position': modifyUser.position,
          'password': modifyUser.password,
          'email': modifyUser.email,
        },
      ),
    );

    if (response.statusCode == 200) {
      // Get updated user profile
      getProfile(context);

      notifyListeners();
      Fluttertoast.showToast(
        msg: 'Update Success',
        backgroundColor: colorSuccess,
        textColor: colorTextPrimary,
      );
      Navigator.pushNamed(context, '/home');
      return user;
    } else if (response.statusCode == 400) {
      Fluttertoast.showToast(
        msg: 'Authentication Failed',
        backgroundColor: colorError,
        textColor: colorTextPrimary,
      );
      notifyListeners();
      return null;
    } else {
      Fluttertoast.showToast(
        msg: 'Server Error',
        backgroundColor: colorError,
        textColor: colorTextPrimary,
      );
      notifyListeners();
      return null;
    }
  }

  void deleteUser(BuildContext context) async {
    var userId = user.id;
    final authToken = await storage.read(key: 'authToken');
    final response = await http.delete(
      Uri.parse('$baseApi/user/remove/$userId'),
      headers: <String, String>{
        'Content-Type': 'application/json; charset=UTF-8',
        'x-auth-token': authToken,
      },
    );

    if (response.statusCode == 200) {
      await storage.deleteAll();
      Navigator.pushNamed(context, '/signup');
    } else if (response.statusCode == 400) {
      Fluttertoast.showToast(msg: 'Authentication Failed');
      notifyListeners();
      return null;
    } else {
      print(response.statusCode);
      Fluttertoast.showToast(msg: 'Server Error');
      notifyListeners();
      return null;
    }
  }
}
```

experience_provider.dart

```dart
import 'dart:convert';

import 'package:flutter/material.dart';
import 'package:fluttertoast/fluttertoast.dart';
import 'package:http/http.dart' as http;
import 'package:flutter_secure_storage/flutter_secure_storage.dart';
import 'package:linkup/constants.dart';
import 'package:linkup/models/experience_model.dart';

class ExperienceProvider extends ChangeNotifier {
  List<Experience> experiences = [];
  FlutterSecureStorage storage = const FlutterSecureStorage();

  void addExperience(
    String position,
    String company,
    String description,
    BuildContext context,
  ) async {
    var userId = await storage.read(key: 'userId');
    var authToken = await storage.read(key: 'authToken');
    var response = await http.post(
      Uri.parse('$baseApi/experiences/user/$userId'),
      headers: <String, String>{
        'Content-Type': 'application/json; charset=UTF-8',
        'x-auth-token': authToken,
      },
      body: jsonEncode(<String, String>{
        'position': position,
        'description': description,
        'companyName': company,
      }),
    );

    if (response.statusCode == 200) {
      getExperience(context);
      Navigator.pop(context);
      notifyListeners();
    } else if (response.statusCode == 400) {
      Fluttertoast.showToast(msg: 'Authentication Failed');
      Navigator.pushNamedAndRemoveUntil(context, '/login', (router) => false);
    } else {
      Fluttertoast.showToast(msg: 'Server Error');
      notifyListeners();
```

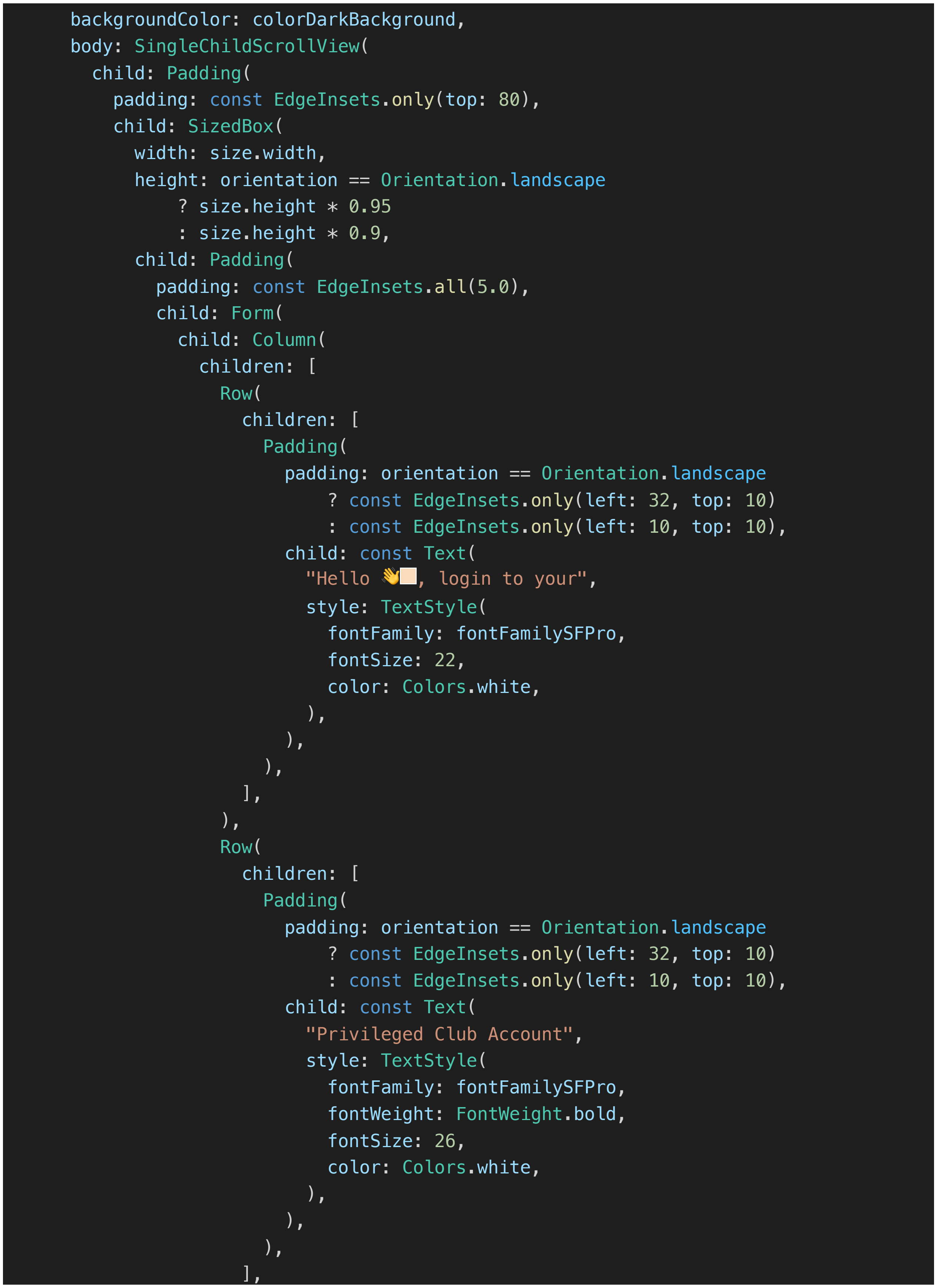


```dart
    }
  }

  Future<List<Experience>> getExperience(BuildContext context) async {
    var authToken = await storage.read(key: 'authToken');
    var response = await http.get(
      Uri.parse('$baseApi/experiences/'),
      headers: <String, String>{
        'Content-Type': 'application/json; charset=UTF-8',
        'x-auth-token': authToken,
      },
    );

    if (response.statusCode == 200) {
      var data = jsonDecode(response.body) as List;
      List<Experience> experiences = [];

      for (Map<String, dynamic> experience in data) {
        experiences.add(Experience.fromJson(experience));
      }
      notifyListeners();
      return experiences;
    } else if (response.statusCode == 400) {
      Fluttertoast.showToast(msg: 'Authentication Failed');
      Navigator.pushNamedAndRemoveUntil(context, '/login', (router) => false);
      return [];
    } else {
      Fluttertoast.showToast(msg: 'Server Error');
      notifyListeners();
      return [];
    }
  }
}
```

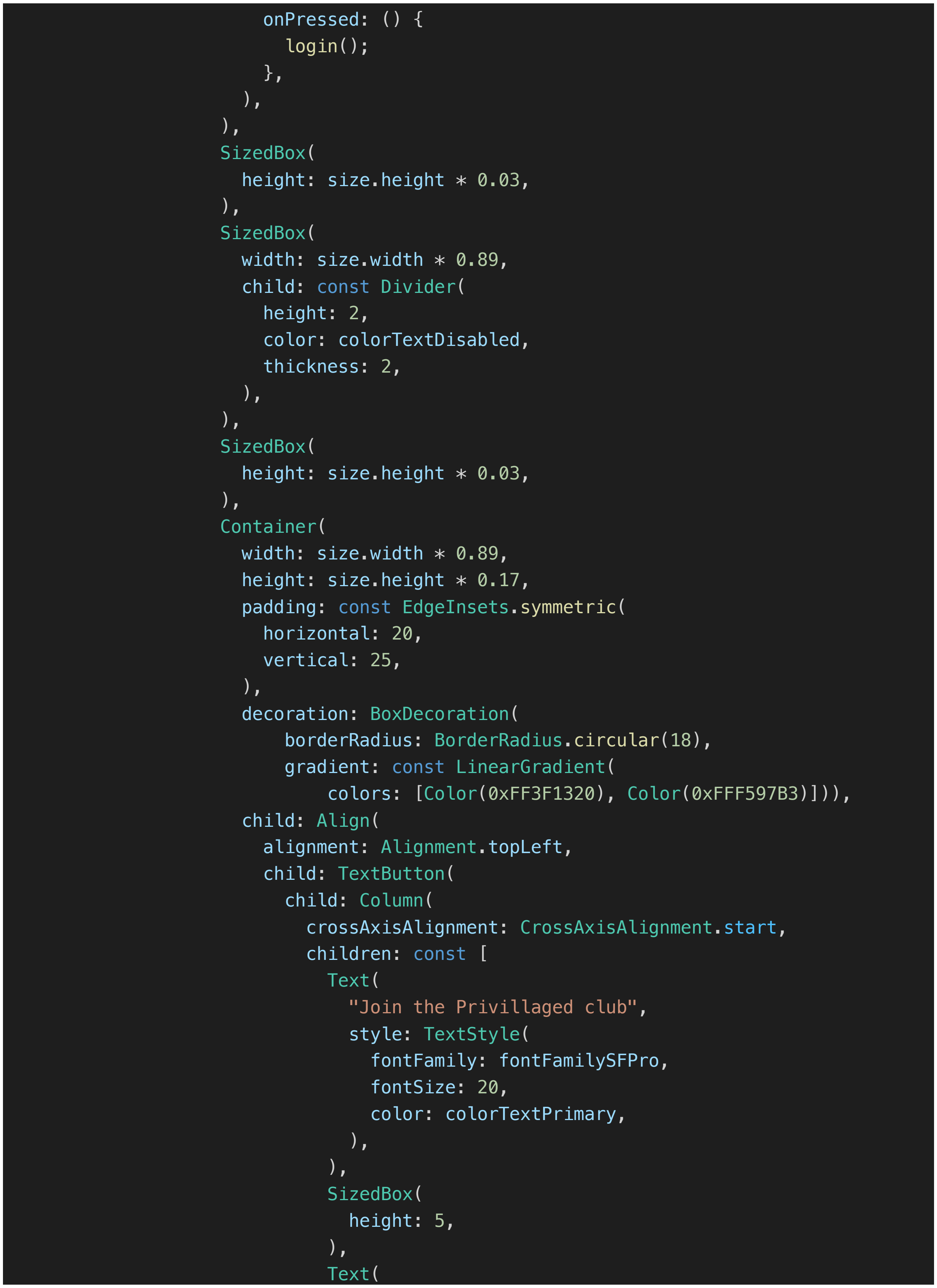
Screens

login_screen.dart

```dart
import "package:flutter/material.dart";
import 'package:fluttertoast/fluttertoast.dart';
import 'package:linkup/components/rounded_text_field.dart';
import 'package:linkup/providers/user_provider.dart';

import '../../components/rounded_button.dart';
import '../../constants.dart';
import 'package:provider/provider.dart';

class LoginScreen extends StatefulWidget {
  const LoginScreen({Key key}) : super(key: key);

  @override
  _LoginScreenState createState() => _LoginScreenState();
}

class _LoginScreenState extends State<LoginScreen> {
  UserProvider userProvider;

  @override
  void initState() {
    super.initState();
    userProvider = context.read<UserProvider>();
  }

  void login() {
    if (userProvider.logUser.email != '' &&
        userProvider.logUser.password != '') {
      userProvider.login(context);
    } else {
      Fluttertoast.showToast(
        msg: 'Inputs are required',
        backgroundColor: colorWarningLight,
        textColor: colorDarkBackground,
      );
    }
  }

  @override
  Widget build(BuildContext context) {
    final size = MediaQuery.of(context).size;
    final orientation = MediaQuery.of(context).orientation;

    return Scaffold(
```



```
      backgroundColor: colorDarkBackground,
      body: SingleChildScrollView(
        child: Padding(
          padding: const EdgeInsets.only(top: 80),
          child: SizedBox(
            width: size.width,
            height: orientation == Orientation.landscape
                ? size.height * 0.95
                : size.height * 0.9,
            child: Padding(
              padding: const EdgeInsets.all(5.0),
              child: Form(
                child: Column(
                  children: [
                    Row(
                      children: [
                        Padding(
                          padding: orientation == Orientation.landscape
                              ? const EdgeInsets.only(left: 32, top: 10)
                              : const EdgeInsets.only(left: 10, top: 10),
                          child: const Text(
                            "Hello 👋🏻, login to your",
                            style: TextStyle(
                              fontFamily: fontFamilySFPro,
                              fontSize: 22,
                              color: Colors.white,
                            ),
                          ),
                        ),
                      ],
                    ),
                    Row(
                      children: [
                        Padding(
                          padding: orientation == Orientation.landscape
                              ? const EdgeInsets.only(left: 32, top: 10)
                              : const EdgeInsets.only(left: 10, top: 10),
                          child: const Text(
                            "Privileged Club Account",
                            style: TextStyle(
                              fontFamily: fontFamilySFPro,
                              fontWeight: FontWeight.bold,
                              fontSize: 26,
                              color: Colors.white,
                            ),
                          ),
                        ),
                      ],
),
SizedBox(
  height: size.height * 0.1,
),
RoundedTextField(
  text: "Email Address",
  type: "email",
  onChange: (value) {
    setState(() {
      userProvider.logUser.email = value;
    });
  },
  value: userProvider.logUser.email,
  isRequired: true,
  backgroundColor: colorDarkBackground,
  textAreaColor: colorDarkMidGround,
),
SizedBox(
  height: size.height * 0.03,
),
RoundedTextField(
  text: "Password",
  type: "password",
  onChange: (value) {
    setState(() {
      userProvider.logUser.password = value;
    });
  },
  value: userProvider.logUser.password,
  isRequired: true,
  backgroundColor: colorDarkBackground,
  textAreaColor: colorDarkMidGround,
),
SizedBox(
  height: size.height * 0.03,
),
Padding(
  padding: const EdgeInsets.only(
    left: 15,
    right: 15,
  ),
  child: RoundedButton(
    color: colorTextPrimary,
    textColor: colorDarkBackground,
    fontSize: 18,
    height: 50,
    width: size.width * 0.4,
    text: "Login",
```

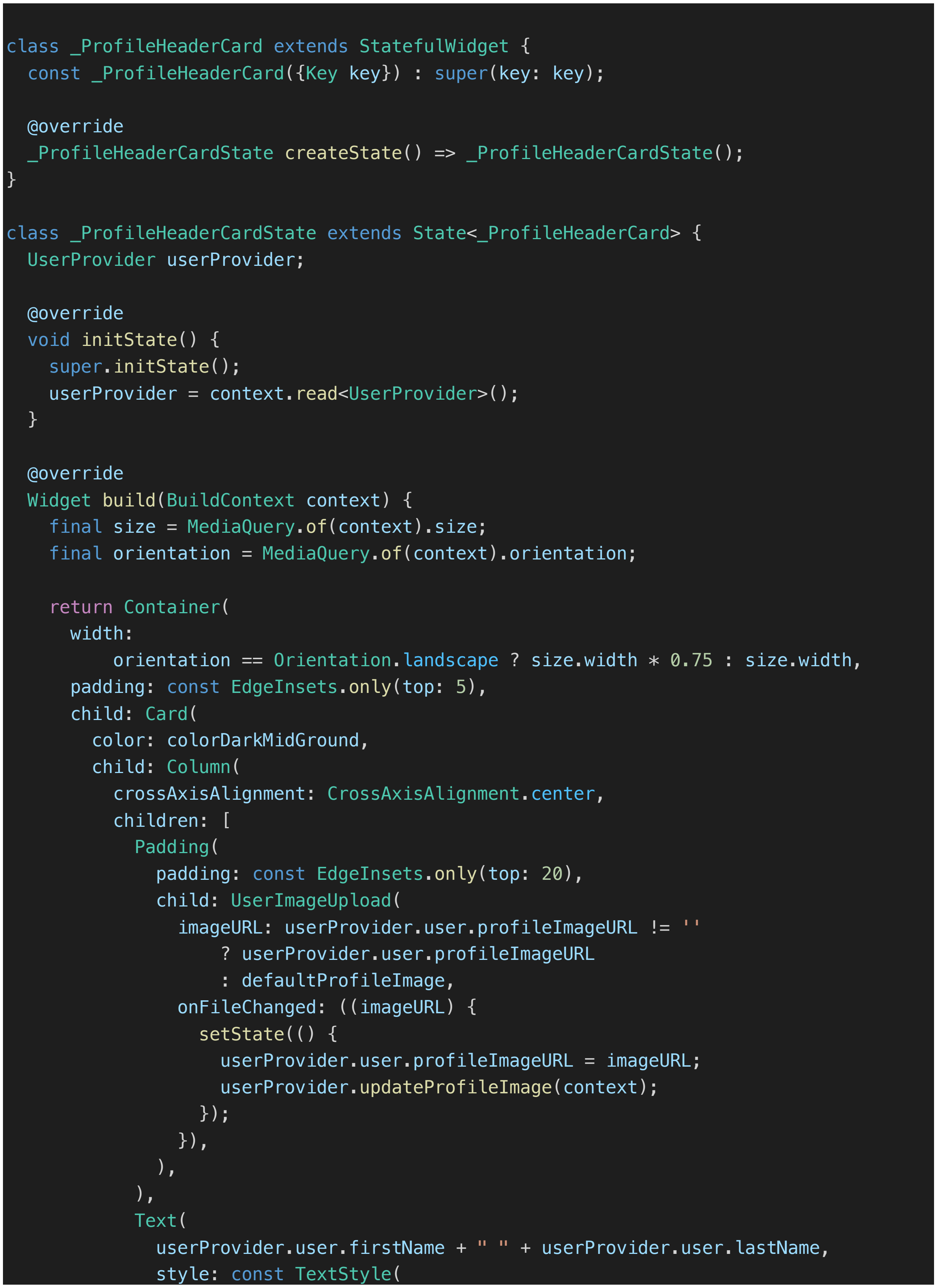


```dart
                        onPressed: () {
                          login();
                        },
                      ),
                    ),
                    SizedBox(
                      height: size.height * 0.03,
                    ),
                    SizedBox(
                      width: size.width * 0.89,
                      child: const Divider(
                        height: 2,
                        color: colorTextDisabled,
                        thickness: 2,
                      ),
                    ),
                    SizedBox(
                      height: size.height * 0.03,
                    ),
                    Container(
                      width: size.width * 0.89,
                      height: size.height * 0.17,
                      padding: const EdgeInsets.symmetric(
                        horizontal: 20,
                        vertical: 25,
                      ),
                      decoration: BoxDecoration(
                          borderRadius: BorderRadius.circular(18),
                          gradient: const LinearGradient(
                              colors: [Color(0xFF3F1320), Color(0xFFF597B3)])),
                      child: Align(
                        alignment: Alignment.topLeft,
                        child: TextButton(
                          child: Column(
                            crossAxisAlignment: CrossAxisAlignment.start,
                            children: const [
                              Text(
                                "Join the Privillaged club",
                                style: TextStyle(
                                  fontFamily: fontFamilySFPro,
                                  fontSize: 20,
                                  color: colorTextPrimary,
                                ),
                              ),
                              SizedBox(
                                height: 5,
                              ),
                              Text(
                                "CREATE ACCOUNT",
                                style: TextStyle(
                                  fontFamily: fontFamilySFPro,
                                  fontSize: 28,
                                  color: colorTextPrimary,
                                  fontWeight: FontWeight.bold,
                                ),
                              ),
                            ],
                          ),
                          onPressed: () =>
                              {Navigator.pushNamed(context, "/signup")},
                        ),
                      ),
                    )
                  ],
                ),
              ),
            ),
          ),
        ),
      ),
    );
  }
}
```

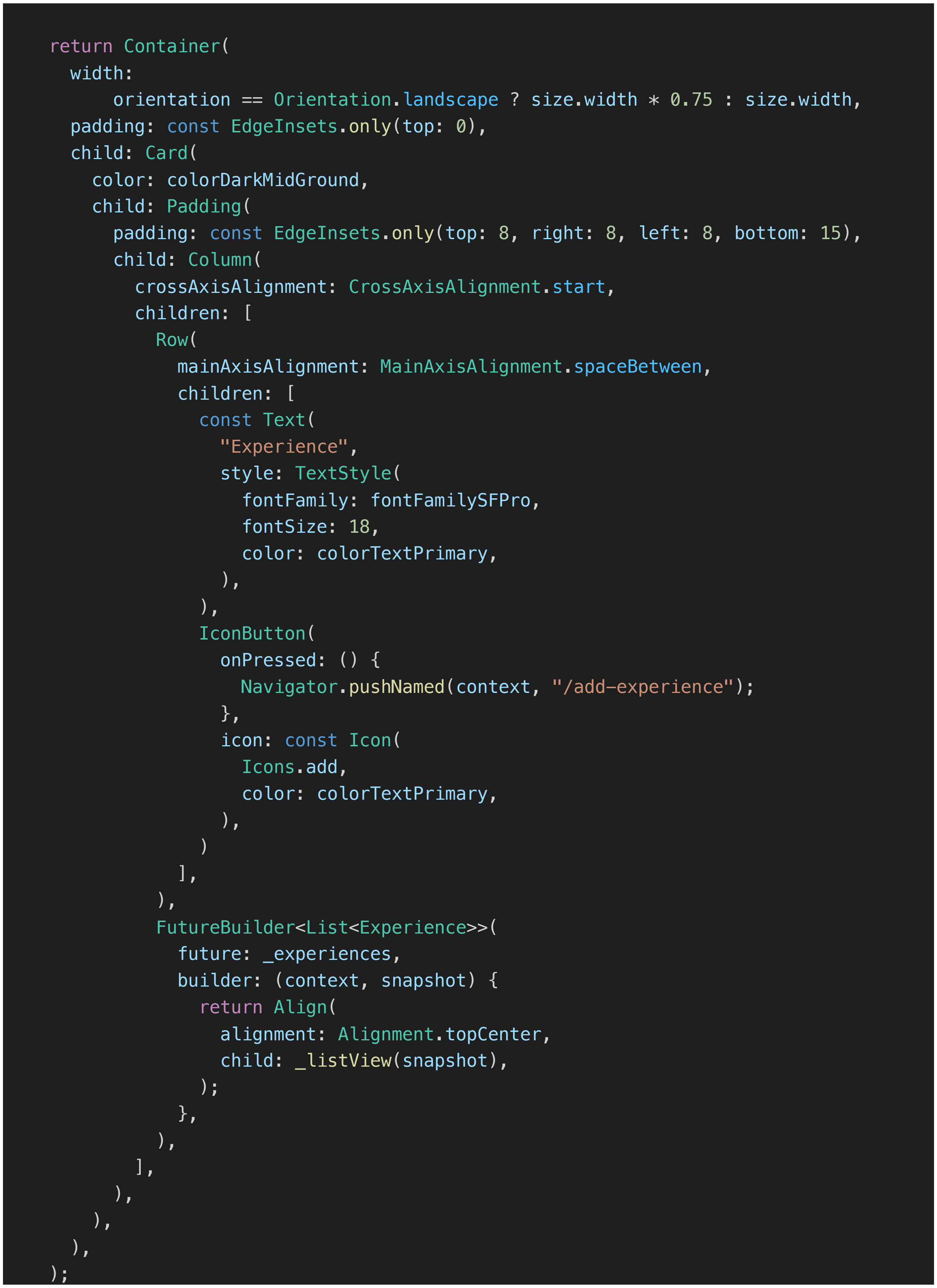
profile_screen.dart

```
import "package:flutter/material.dart";
import 'package:linkup/components/education_card.dart';
import 'package:linkup/components/experience_card.dart';
import 'package:linkup/components/skills_card.dart';
import 'package:linkup/components/user_image_upload.dart';
import 'package:linkup/constants.dart';
import 'package:font_awesome_flutter/font_awesome_flutter.dart';
import 'package:linkup/models/experience_model.dart';
import 'package:linkup/providers/experience_provider.dart';
import 'package:linkup/providers/user_provider.dart';
import 'package:provider/provider.dart';

Future<List<Experience>> _experiences;

class ProfileScreen extends StatefulWidget {
  const ProfileScreen({Key key}) : super(key: key);

  @override
```

```dart
  _ProfileScreenState createState() => _ProfileScreenState();
}

class _ProfileScreenState extends State<ProfileScreen> {
  UserProvider userProvider;
  ExperienceProvider _expProvider;


  @override
  void initState() {
    super.initState();
    _expProvider = context.read<ExperienceProvider>();
    _experiences = _expProvider.getExperience(context);
    userProvider = context.read<UserProvider>();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: colorDarkBackground,
      body: Align(
        alignment: Alignment.topCenter,
        child: RefreshIndicator(
          onRefresh: _pullRefresh,
          child: SingleChildScrollView(
            child: Column(
              crossAxisAlignment: CrossAxisAlignment.center,
              children: [
                const _ProfileHeaderCard(),
                _ExperienceSection(),
                _EducationSection(),
                _SkillsSection(),
              ],
            ),
          ),
        ),
      ),
    );
  }

  Future<void> _pullRefresh() async {
    await Future.delayed(const Duration(seconds: 1));
    setState(() {
      _experiences = Provider.of<ExperienceProvider>(context, listen: false)
          .getExperience(context);
    });
  }
}
```

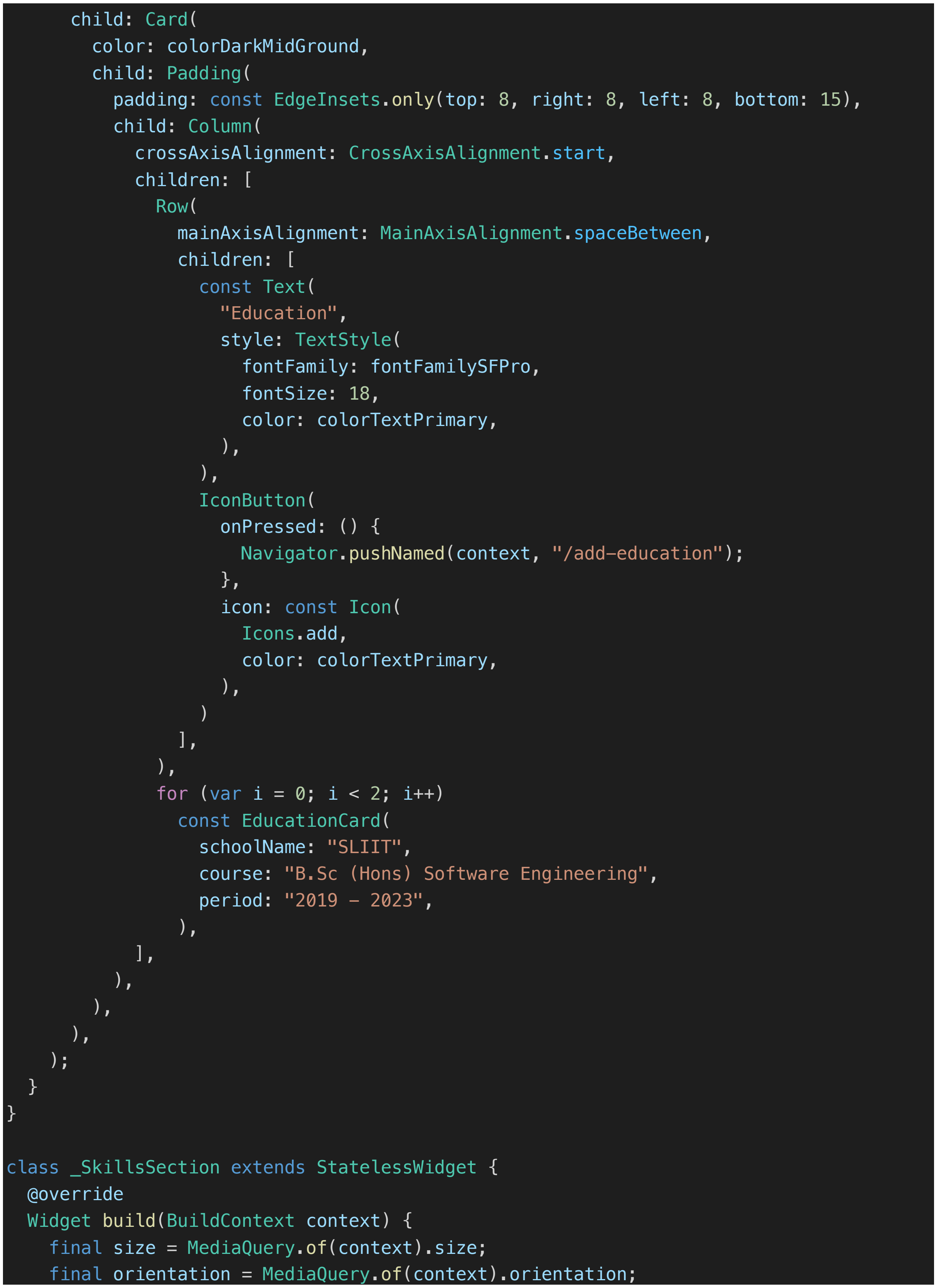


```dart
class _ProfileHeaderCard extends StatefulWidget {
  const _ProfileHeaderCard({Key key}) : super(key: key);

  @override
  _ProfileHeaderCardState createState() => _ProfileHeaderCardState();
}

class _ProfileHeaderCardState extends State<_ProfileHeaderCard> {
  UserProvider userProvider;

  @override
  void initState() {
    super.initState();
    userProvider = context.read<UserProvider>();
  }

  @override
  Widget build(BuildContext context) {
    final size = MediaQuery.of(context).size;
    final orientation = MediaQuery.of(context).orientation;

    return Container(
      width:
          orientation == Orientation.landscape ? size.width * 0.75 : size.width,
      padding: const EdgeInsets.only(top: 5),
      child: Card(
        color: colorDarkMidGround,
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.center,
          children: [
            Padding(
              padding: const EdgeInsets.only(top: 20),
              child: UserImageUpload(
                imageURL: userProvider.user.profileImageURL != ''
                    ? userProvider.user.profileImageURL
                    : defaultProfileImage,
                onFileChanged: ((imageURL) {
                  setState(() {
                    userProvider.user.profileImageURL = imageURL;
                    userProvider.updateProfileImage(context);
                  });
                }),
              ),
            ),
            Text(
              userProvider.user.firstName + " " + userProvider.user.lastName,
              style: const TextStyle(
                fontFamily: fontFamilySFPro,
                fontSize: 24,
                color: colorTextPrimary,
                fontWeight: FontWeight.bold,
              ),
            ),
            const SizedBox(
              height: 3,
            ),
            Text(
              userProvider.user.position,
              style: const TextStyle(
                fontFamily: fontFamilySFPro,
                fontSize: 16,
                color: colorTextPrimary,
              ),
            ),
            const SizedBox(
              height: 10,
            ),
            Row(
              mainAxisAlignment: MainAxisAlignment.center,
              children: const [
                Icon(
                  FontAwesomeIcons.facebook,
                  color: colorTextPrimary,
                  size: 30,
                ),
                SizedBox(
                  width: 12,
                ),
                Icon(
                  FontAwesomeIcons.linkedin,
                  color: colorTextPrimary,
                  size: 30,
                ),
                SizedBox(
                  width: 12,
                ),
                Icon(
                  FontAwesomeIcons.instagram,
                  color: colorTextPrimary,
                  size: 30,
                ),
                SizedBox(
                  width: 12,
                ),
                Icon(
                  FontAwesomeIcons.github,
                  color: colorTextPrimary,
                  size: 30,
                )
              ],
            ),
            const SizedBox(
              height: 12,
            ),
            Align(
              alignment: Alignment.center,
              child: Padding(
                padding: const EdgeInsets.only(right: 10, bottom: 5),
                child: TextButton(
                  onPressed: () {
                    Navigator.pushNamed(context, '/edit-profile');
                  },
                  child: const Text(
                    "Edit",
                    style: TextStyle(
                      fontFamily: fontFamilySFPro,
                      fontSize: 16,
                      color: colorPrimaryLight,
                    ),
                  ),
                ),
              ),
            ),
            const SizedBox(
              height: 12,
            ),
          ],
        ),
      ),
    );
  }
}

class _ExperienceSection extends StatefulWidget {
  @override
  _ExperienceSectionState createState() => _ExperienceSectionState();
}

class _ExperienceSectionState extends State<_ExperienceSection> {
  @override
  Widget build(BuildContext context) {
    final size = MediaQuery.of(context).size;
    final orientation = MediaQuery.of(context).orientation;
```

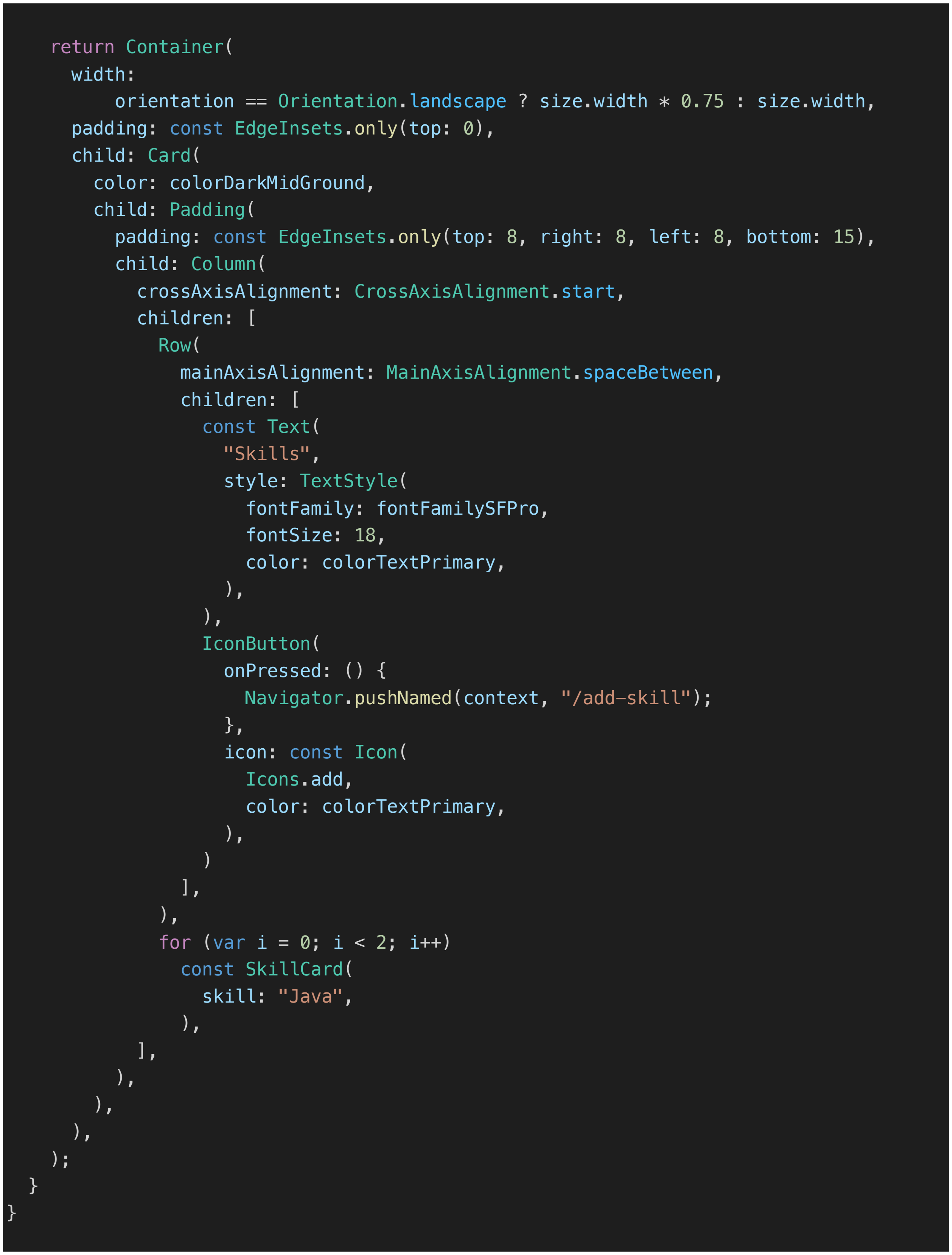


```dart
return Container(
  width:
      orientation == Orientation.landscape ? size.width * 0.75 : size.width,
  padding: const EdgeInsets.only(top: 0),
  child: Card(
    color: colorDarkMidGround,
    child: Padding(
      padding: const EdgeInsets.only(top: 8, right: 8, left: 8, bottom: 15),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          Row(
            mainAxisAlignment: MainAxisAlignment.spaceBetween,
            children: [
              const Text(
                "Experience",
                style: TextStyle(
                  fontFamily: fontFamilySFPro,
                  fontSize: 18,
                  color: colorTextPrimary,
                ),
              ),
              IconButton(
                onPressed: () {
                  Navigator.pushNamed(context, "/add-experience");
                },
                icon: const Icon(
                  Icons.add,
                  color: colorTextPrimary,
                ),
              )
            ],
          ),
          FutureBuilder<List<Experience>>(
            future: _experiences,
            builder: (context, snapshot) {
              return Align(
                alignment: Alignment.topCenter,
                child: _listView(snapshot),
              );
            },
          ),
        ],
      ),
    ),
  ),
);
```

```
  }

  Widget _listView(AsyncSnapshot snapshot) {
    if (snapshot.hasData) {
      return Column(
        children: [
          for (var i = 0; i < snapshot.data.length; i++)
            ExperienceCard(
              companyName: snapshot.data[i].companyName,
              position: snapshot.data[i].position,
              description: snapshot.data[i].description,
            )
        ],
      );
    } else if (snapshot.hasError) {
      return const Text(
        'Error with fetch experiences',
        style: TextStyle(
          fontFamily: fontFamilySFPro,
          fontSize: 16,
          color: colorTextPrimary,
        ),
      );
    }
    return const Padding(
      padding: EdgeInsets.only(top: 20),
      child: SizedBox(
        width: 30,
        height: 30,
        child: CircularProgressIndicator(
          color: colorTextPrimary,
          strokeWidth: 2,
        ),
      ),
    );
  }
}

class _EducationSection extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    final size = MediaQuery.of(context).size;
    final orientation = MediaQuery.of(context).orientation;

    return Container(
      width:
          orientation == Orientation.landscape ? size.width * 0.75 : size.width,
      padding: const EdgeInsets.only(top: 0),
```



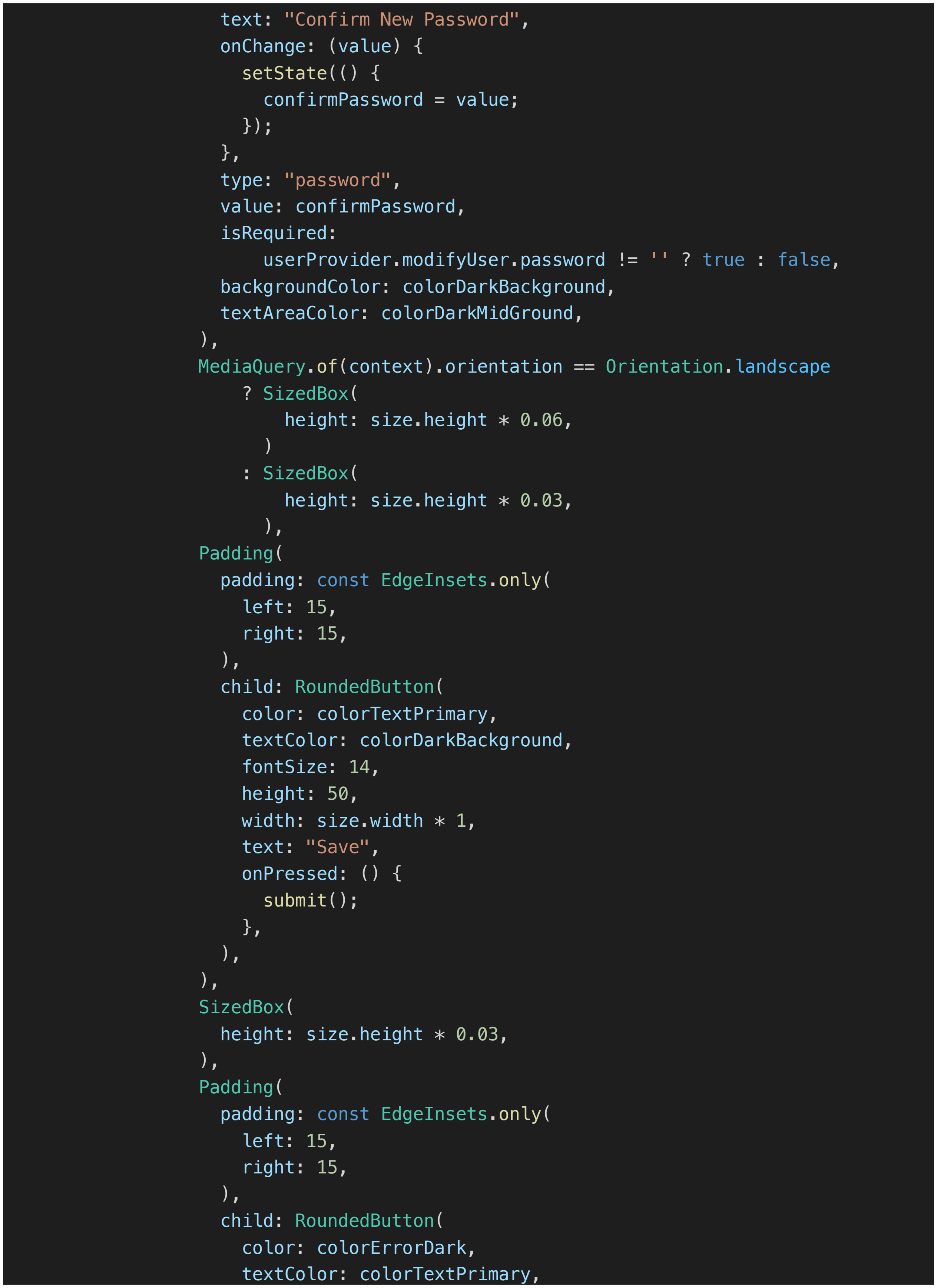
```
      child: Card(
        color: colorDarkMidGround,
        child: Padding(
          padding: const EdgeInsets.only(top: 8, right: 8, left: 8, bottom: 15),
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              Row(
                mainAxisAlignment: MainAxisAlignment.spaceBetween,
                children: [
                  const Text(
                    "Education",
                    style: TextStyle(
                      fontFamily: fontFamilySFPro,
                      fontSize: 18,
                      color: colorTextPrimary,
                    ),
                  ),
                  IconButton(
                    onPressed: () {
                      Navigator.pushNamed(context, "/add-education");
                    },
                    icon: const Icon(
                      Icons.add,
                      color: colorTextPrimary,
                    ),
                  )
                ],
              ),
              for (var i = 0; i < 2; i++)
                const EducationCard(
                  schoolName: "SLIIT",
                  course: "B.Sc (Hons) Software Engineering",
                  period: "2019 - 2023",
                ),
            ],
          ),
        ),
      ),
    );
  }
}

class _SkillsSection extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    final size = MediaQuery.of(context).size;
    final orientation = MediaQuery.of(context).orientation;
```



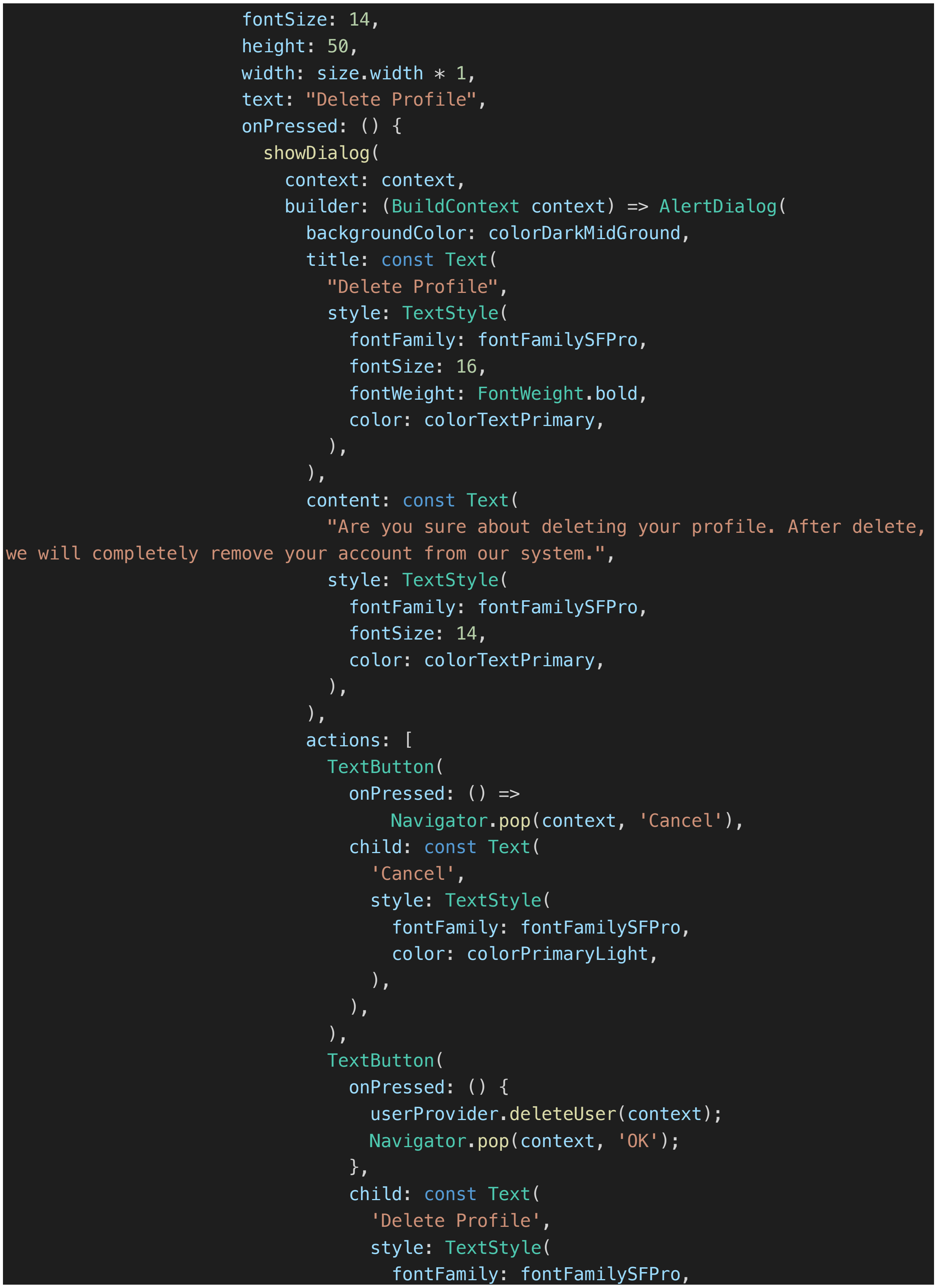
```
    return Container(
      width:
          orientation == Orientation.landscape ? size.width * 0.75 : size.width,
      padding: const EdgeInsets.only(top: 0),
      child: Card(
        color: colorDarkMidGround,
        child: Padding(
          padding: const EdgeInsets.only(top: 8, right: 8, left: 8, bottom: 15),
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              Row(
                mainAxisAlignment: MainAxisAlignment.spaceBetween,
                children: [
                  const Text(
                    "Skills",
                    style: TextStyle(
                      fontFamily: fontFamilySFPro,
                      fontSize: 18,
                      color: colorTextPrimary,
                    ),
                  ),
                  IconButton(
                    onPressed: () {
                      Navigator.pushNamed(context, "/add-skill");
                    },
                    icon: const Icon(
                      Icons.add,
                      color: colorTextPrimary,
                    ),
                  )
                ],
              ),
              for (var i = 0; i < 2; i++)
                const SkillCard(
                  skill: "Java",
                ),
            ],
          ),
        ),
      ),
    );
  }
}
```

edit_profile_screen.dart

```
import "package:flutter/material.dart";
import 'package:linkup/components/rounded_button.dart';
import 'package:linkup/components/rounded_number_field.dart';
import 'package:linkup/components/rounded_text_field.dart';
import 'package:linkup/constants.dart';
import 'package:linkup/providers/user_provider.dart';
import 'package:provider/provider.dart';

class EditProfileScreen extends StatefulWidget {
  const EditProfileScreen({Key key}) : super(key: key);

  @override
  _EditProfileScreenState createState() => _EditProfileScreenState();
}

class _EditProfileScreenState extends State<EditProfileScreen> {
  UserProvider userProvider;
  GlobalKey<FormState> editProfileFormKey = GlobalKey();
  String confirmPassword = '';

  @override
  void initState() {
    super.initState();
    userProvider = context.read<UserProvider>();
    userProvider.modifyUser.id = userProvider.user.id;
    userProvider.modifyUser.firstName = userProvider.user.firstName;
    userProvider.modifyUser.lastName = userProvider.user.lastName;
    userProvider.modifyUser.email = userProvider.user.email;
    userProvider.modifyUser.phoneNumber = userProvider.user.phoneNumber;
  }

  void submit() {
    if (editProfileFormKey.currentState.validate()) {
      userProvider.updateUser(context);
    }
  }

  @override
  Widget build(BuildContext context) {
    final size = MediaQuery.of(context).size;

    return Scaffold(
      appBar: AppBar(
        title: const Text(
          "Edit Profile",
          style: TextStyle(fontFamily: fontFamilySFPro),
        ),
        backgroundColor: colorDarkMidGround,
        elevation: 0.0,
      ),
      backgroundColor: colorDarkBackground,
      body: SingleChildScrollView(
        child: Container(
          width: size.width,
          padding: const EdgeInsets.all(0.0),
          child: Padding(
            padding: const EdgeInsets.all(5.0),
            child: Form(
              key: editProfileFormKey,
              child: Column(
                children: [
                  SizedBox(
                    height: size.height * 0.03,
                  ),
                  RoundedTextField(
                    text: "First Name",
                    onChange: (value) {
                      setState(() {
                        print(value);
                        userProvider.modifyUser.firstName = value;
                      });
                    },
                    value: userProvider.modifyUser.firstName,
                    isRequired: true,
                    backgroundColor: colorDarkBackground,
                    textAreaColor: colorDarkMidGround,
                  ),
                  SizedBox(
                    height: size.height * 0.03,
                  ),
                  RoundedTextField(
                    text: "Last Name",
                    onChange: (value) {
                      setState(() {
                        userProvider.modifyUser.lastName = value;
                      });
                    },
                    value: userProvider.modifyUser.lastName,
                    isRequired: true,
                    backgroundColor: colorDarkBackground,
                    textAreaColor: colorDarkMidGround,
                  ),
                  SizedBox(
                    height: size.height * 0.03,
                  ),
                  // RoundedTextField(
                  //   text: "Email Address",
                  //   type: "email",
                  //   onChange: (value) {
                  //     setState(() {
                  //       userProvider.modifyUser.email = value;
                  //     });
                  //   },
                  //   value: userProvider.modifyUser.email,
                  //   isRequired: true,
                  //   backgroundColor: colorDarkBackground,
                  //   textAreaColor: colorDarkMidGround,
                  // ),
                  // SizedBox(
                  //   height: size.height * 0.03,
                  // ),
                  RoundedNumberField(
                    text: "Phone Number",
                    type: "phone",
                    onChange: (value) {
                      setState(() {
                        userProvider.modifyUser.phoneNumber = value;
                      });
                    },
                    value: userProvider.modifyUser.phoneNumber,
                    isRequired: true,
                  ),
                  SizedBox(
                    height: size.height * 0.03,
                  ),
                  RoundedTextField(
                    text: "New Password",
                    onChange: (value) {
                      setState(() {
                        userProvider.modifyUser.password = value;
                      });
                    },
                    type: "password",
                    value: userProvider.modifyUser.password,
                    isRequired: false,
                    backgroundColor: colorDarkBackground,
                    textAreaColor: colorDarkMidGround,
                  ),
                  SizedBox(
                    height: size.height * 0.03,
                  ),
                  RoundedTextField(
```



```dart
  text: "Confirm New Password",
  onChange: (value) {
    setState(() {
      confirmPassword = value;
    });
  },
  type: "password",
  value: confirmPassword,
  isRequired:
      userProvider.modifyUser.password != '' ? true : false,
  backgroundColor: colorDarkBackground,
  textAreaColor: colorDarkMidGround,
),
MediaQuery.of(context).orientation == Orientation.landscape
    ? SizedBox(
        height: size.height * 0.06,
      )
    : SizedBox(
        height: size.height * 0.03,
      ),
Padding(
  padding: const EdgeInsets.only(
    left: 15,
    right: 15,
  ),
  child: RoundedButton(
    color: colorTextPrimary,
    textColor: colorDarkBackground,
    fontSize: 14,
    height: 50,
    width: size.width * 1,
    text: "Save",
    onPressed: () {
      submit();
    },
  ),
),
SizedBox(
  height: size.height * 0.03,
),
Padding(
  padding: const EdgeInsets.only(
    left: 15,
    right: 15,
  ),
  child: RoundedButton(
    color: colorErrorDark,
    textColor: colorTextPrimary,
```

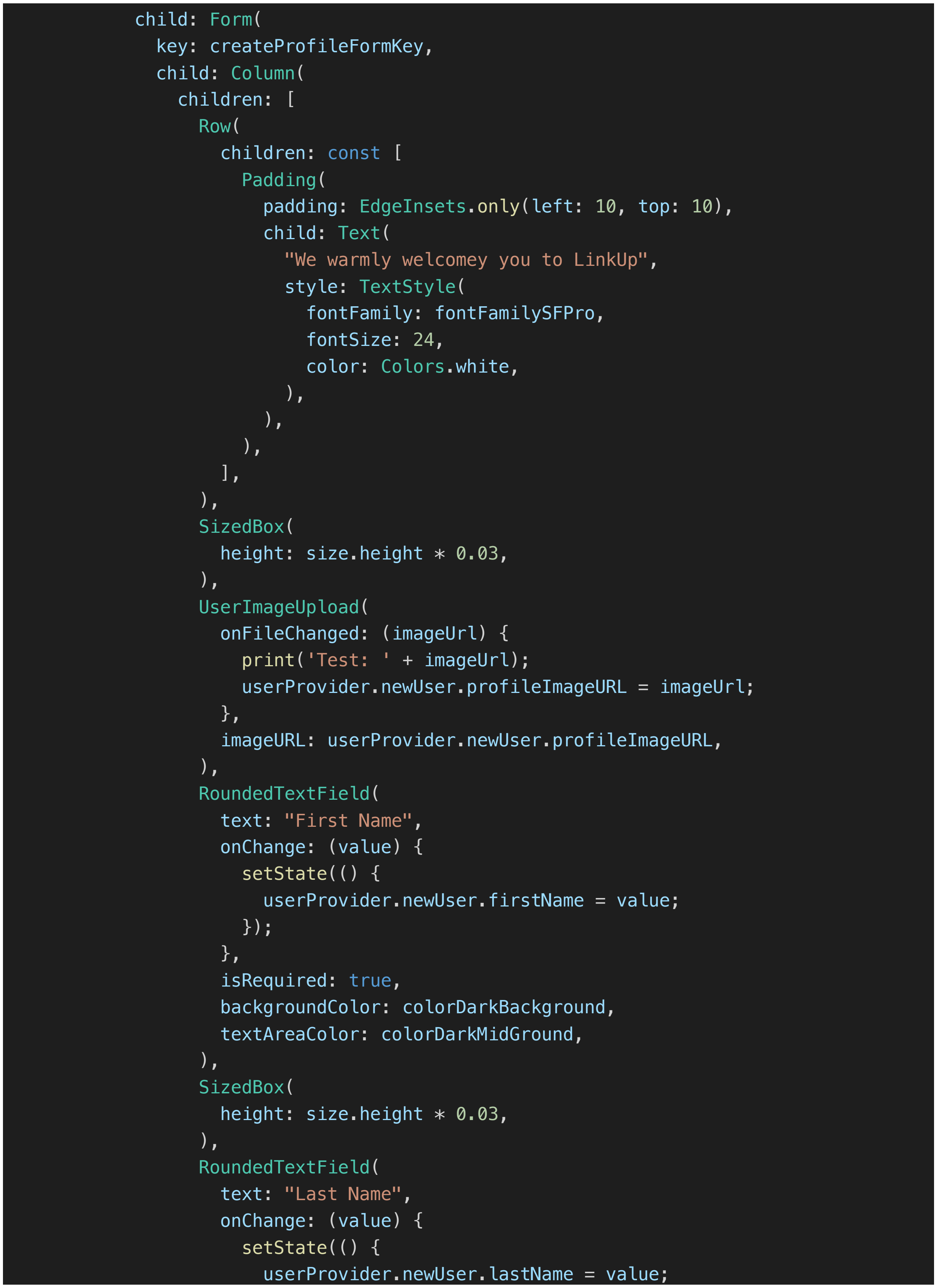


```
                      fontSize: 14,
                      height: 50,
                      width: size.width * 1,
                      text: "Delete Profile",
                      onPressed: () {
                        showDialog(
                          context: context,
                          builder: (BuildContext context) => AlertDialog(
                            backgroundColor: colorDarkMidGround,
                            title: const Text(
                              "Delete Profile",
                              style: TextStyle(
                                fontFamily: fontFamilySFPro,
                                fontSize: 16,
                                fontWeight: FontWeight.bold,
                                color: colorTextPrimary,
                              ),
                            ),
                            content: const Text(
                              "Are you sure about deleting your profile. After delete,
we will completely remove your account from our system.",
                              style: TextStyle(
                                fontFamily: fontFamilySFPro,
                                fontSize: 14,
                                color: colorTextPrimary,
                              ),
                            ),
                            actions: [
                              TextButton(
                                onPressed: () =>
                                    Navigator.pop(context, 'Cancel'),
                                child: const Text(
                                  'Cancel',
                                  style: TextStyle(
                                    fontFamily: fontFamilySFPro,
                                    color: colorPrimaryLight,
                                  ),
                                ),
                              ),
                              TextButton(
                                onPressed: () {
                                  userProvider.deleteUser(context);
                                  Navigator.pop(context, 'OK');
                                },
                                child: const Text(
                                  'Delete Profile',
                                  style: TextStyle(
                                    fontFamily: fontFamilySFPro,
                                    color: colorError,
                                  ),
                                ),
                              ),
                            ],
                          ),
                        );
                      },
                    ),
                  ),
                  SizedBox(
                    height: size.height * 0.03,
                  ),
                ],
              ),
            ),
          ),
        ),
      ),
    );
  }
}
```

signup_screen.dart

```
import "package:flutter/material.dart";
import 'package:fluttertoast/fluttertoast.dart';
import 'package:linkup/components/rounded_button.dart';
import 'package:linkup/components/rounded_number_field.dart';
import 'package:linkup/components/rounded_text_field.dart';
import 'package:linkup/components/side_navbar.dart';
import 'package:linkup/constants.dart';
import 'package:linkup/providers/user_provider.dart';
import 'package:provider/provider.dart';

import '../../components/user_image_upload.dart';

class SignUpScreen extends StatefulWidget {
  const SignUpScreen({Key key}) : super(key: key);

  @override
  _SignUpScreenState createState() => _SignUpScreenState();
}

class _SignUpScreenState extends State<SignUpScreen> {
  UserProvider userProvider;
```

```dart
GlobalKey<FormState> createProfileFormKey = GlobalKey();
String conformPassword;

@override
void initState() {
  super.initState();
  userProvider = context.read<UserProvider>();
}

void submit() {
  if (createProfileFormKey.currentState.validate()) {
    if (userProvider.newUser.password == conformPassword) {
      userProvider.create(context);
    } else {
      Fluttertoast.showToast(
        msg: 'Password not matched',
        backgroundColor: colorWarningLight,
        textColor: colorDarkBackground,
      );
    }
  }
}

@override
Widget build(BuildContext context) {
  final size = MediaQuery.of(context).size;

  return Scaffold(
    appBar: AppBar(
      backgroundColor: colorDarkBackground,
      iconTheme: const IconThemeData(color: Colors.white),
      elevation: 0.0,
      title: const Text(
        "Sign Up",
        style: TextStyle(
          fontFamily: "SF-Pro",
          color: Colors.white,
        ),
      ),
    ),
    backgroundColor: colorDarkBackground,
    // drawer: const SideNavbar(),
    body: SingleChildScrollView(
      child: Container(
        width: size.width,
        padding: const EdgeInsets.all(0.0),
        child: Padding(
          padding: const EdgeInsets.all(5.0),
```



```dart
            child: Form(
              key: createProfileFormKey,
              child: Column(
                children: [
                  Row(
                    children: const [
                      Padding(
                        padding: EdgeInsets.only(left: 10, top: 10),
                        child: Text(
                          "We warmly welcomey you to LinkUp",
                          style: TextStyle(
                            fontFamily: fontFamilySFPro,
                            fontSize: 24,
                            color: Colors.white,
                          ),
                        ),
                      ),
                    ],
                  ),
                  SizedBox(
                    height: size.height * 0.03,
                  ),
                  UserImageUpload(
                    onFileChanged: (imageUrl) {
                      print('Test: ' + imageUrl);
                      userProvider.newUser.profileImageURL = imageUrl;
                    },
                    imageURL: userProvider.newUser.profileImageURL,
                  ),
                  RoundedTextField(
                    text: "First Name",
                    onChange: (value) {
                      setState(() {
                        userProvider.newUser.firstName = value;
                      });
                    },
                    isRequired: true,
                    backgroundColor: colorDarkBackground,
                    textAreaColor: colorDarkMidGround,
                  ),
                  SizedBox(
                    height: size.height * 0.03,
                  ),
                  RoundedTextField(
                    text: "Last Name",
                    onChange: (value) {
                      setState(() {
                        userProvider.newUser.lastName = value;
                      });
                    },
                    isRequired: true,
                    backgroundColor: colorDarkBackground,
                    textAreaColor: colorDarkMidGround,
                  ),
                  SizedBox(
                    height: size.height * 0.03,
                  ),
                  RoundedTextField(
                    text: "Email Address",
                    type: "email",
                    onChange: (value) {
                      setState(() {
                        userProvider.newUser.email = value;
                      });
                    },
                    isRequired: true,
                    backgroundColor: colorDarkBackground,
                    textAreaColor: colorDarkMidGround,
                  ),
                  SizedBox(
                    height: size.height * 0.03,
                  ),
                  RoundedNumberField(
                    text: "Phone Number",
                    type: "phone",
                    onChange: (value) {
                      setState(() {
                        userProvider.newUser.phoneNumber = value;
                      });
                    },
                    isRequired: true,
                  ),
                  SizedBox(
                    height: size.height * 0.03,
                  ),
                  RoundedTextField(
                    text: "Position",
                    onChange: (value) {
                      setState(() {
                        userProvider.newUser.position = value;
                      });
                    },
                    isRequired: true,
                    backgroundColor: colorDarkBackground,
                    textAreaColor: colorDarkMidGround,
                  ),
SizedBox(
  height: size.height * 0.03,
),
RoundedTextField(
  text: "Password",
  onChange: (value) {
    setState(() {
      userProvider.newUser.password = value;
    });
  },
  type: "password",
  isRequired: true,
  backgroundColor: colorDarkBackground,
  textAreaColor: colorDarkMidGround,
),
SizedBox(
  height: size.height * 0.03,
),
RoundedTextField(
  text: "Conform Password",
  onChange: (value) {
    setState(() {
      conformPassword = value;
    });
  },
  type: "password",
  isRequired: true,
  backgroundColor: colorDarkBackground,
  textAreaColor: colorDarkMidGround,
),
MediaQuery.of(context).orientation == Orientation.landscape
    ? SizedBox(
        height: size.height * 0.06,
      )
    : SizedBox(
        height: size.height * 0.03,
      ),
Padding(
  padding: const EdgeInsets.only(
    left: 15,
    right: 15,
  ),
  child: RoundedButton(
    color: colorDarkForground,
    fontSize: 14,
    height: 50,
    width: size.width * 0.4,
    text: "Sign Up",
                      onPressed: () {
                        print("Button clicked");
                        submit();
                      },
                    ),
                  ),
                  SizedBox(
                    height: size.height * 0.03,
                  ),
                ],
              ),
            ),
          ),
        ),
      ),
    );
  }
}
```