Sri Lanka Institute of Information Technology

LinkUp

Current Trends in Software Engineering

2022

Submitted by:

Bandara G.B.M.A.G.R.A.V.

IT19104218

User Posts and News feed Management

Table of Contents

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

Adding posts is another important functionality of this application. Users can post and share content in LinkUp and once they add a post it will be shared publicly in a feed that is visible to all users. This gives the user the ability to give an update to the community on the things that they are learning, important information, and new findings via posts. In this application users are allowed to share only images and text and both types of users are given privilege to share content in their respective profiles.

The main features of the post feed functionality are as follows,

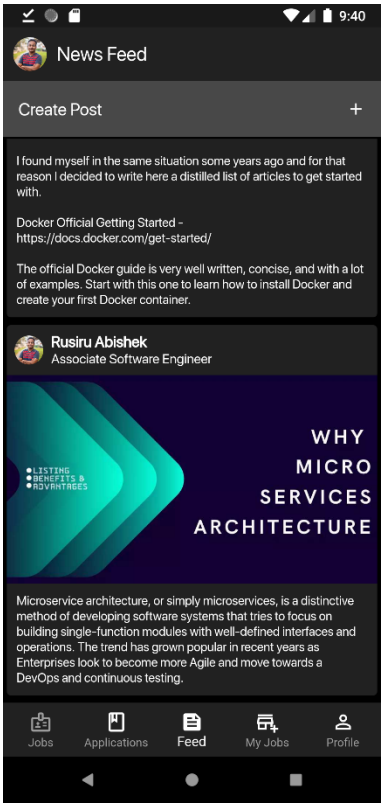
- Ability to insert a post to the general feed with or without images
- Ability to update the information provided under the posts
- Ability to delete posts.

There are many ways the user can interact with the image upload feature. The user can either access images from the file storage or by capturing an image from the camera. In the following section we will discuss the individual functionalities of the system.

Functionalities

Insert post

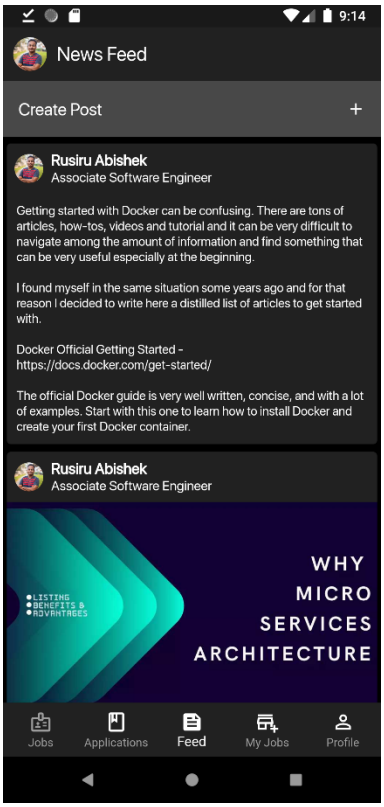
The insert post functionality allows the user to add a post to the general post feed. The posts in the feed can be viewed by all the users registered under the platform. So this gives the user the ability to show proficiency of the knowledge on the subject. To insert a new post, the user has to navigate to the general post feed and click on the expansion tile fixed at the top. The flow of the interface is implemented such that the inserting component is at the top and the general post feed is under that



The expandable tile allows the user to expand the component when it is required to add new posts and after that, it is in the collapsed state. This intuitive design approach makes it easier for the user to interact with the application easily.

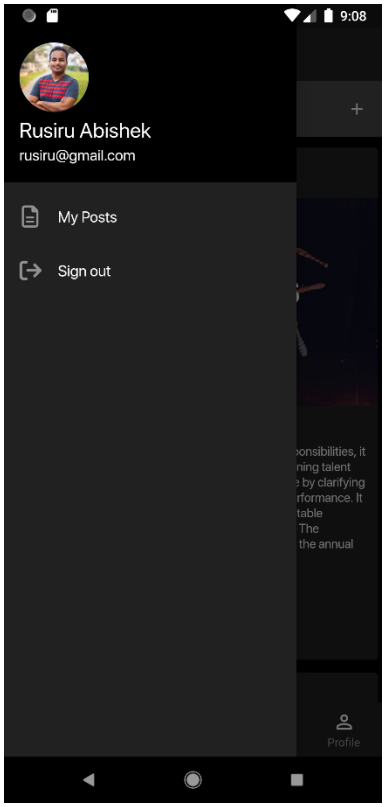
View News feed.

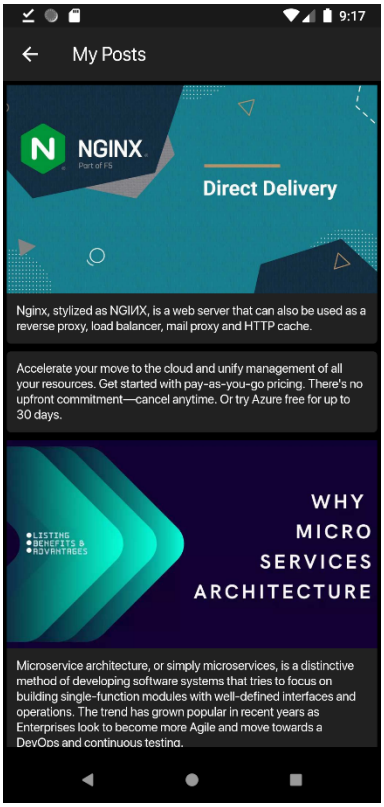
The inserted post is visible under the expandible tile. Here the user can view the general feed which includes all the posts added by different users. The information of the users that added the post is also displayed along with the post caption and other information. This is done by populating the post information that is retrieved from the backend to have nested references to the user information.



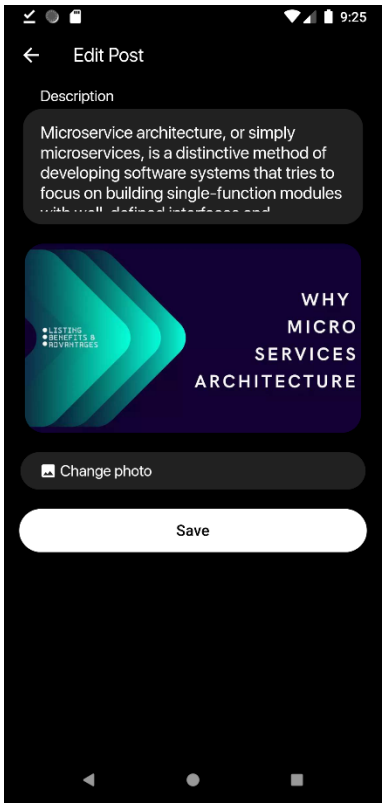
Update post

Update user post functionality is another key feature in LinkUp application. This feature allows users to modify the posts that they have published. Before update a post, authenticated user needs to select **My Posts** from the side navigation menu. The following image illustrates the user interface of the user published posts.





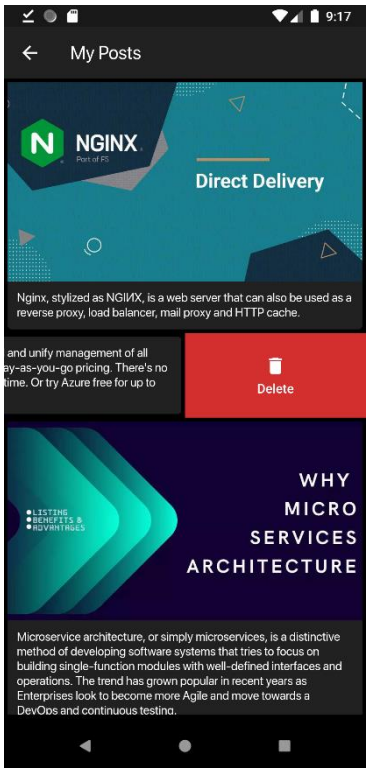
To update a post, user have to swipe from left to right. Then it will open a new screen with the update post form. All the previous values are automatically filled to the relevant input fields. Moreover, previous uploaded image is also displayed on the image previewer



To update a post, user have to swipe from left to right. Then it will open a new screen with the update post form. All the preivous values are automatically filled to the relevant input fields. Moreover, previous image is also displayed on the image previewer. After click on the **Save** button, user can swipe down to refresh the post list

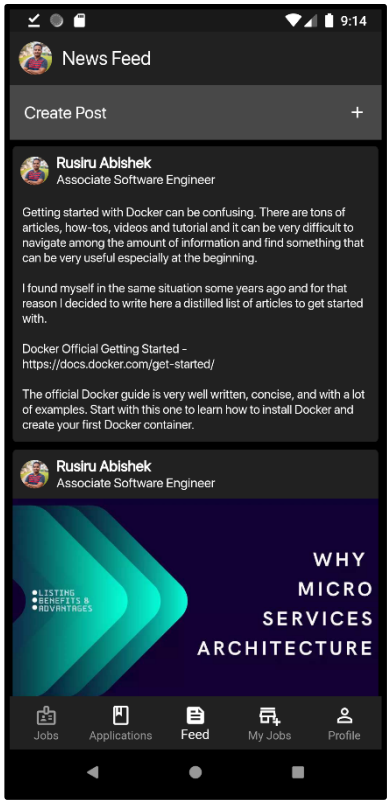
After click on the **Save** button, user can swipe down to refresh the updated post list.

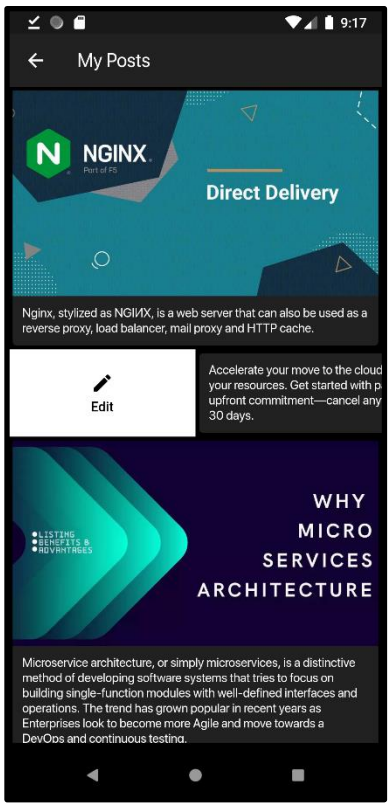
Delete post

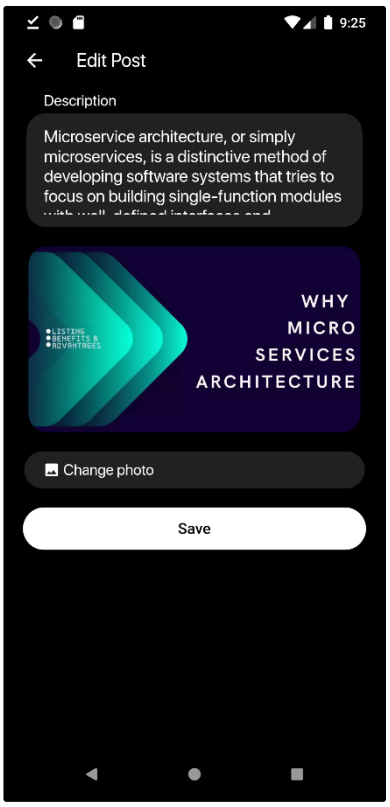


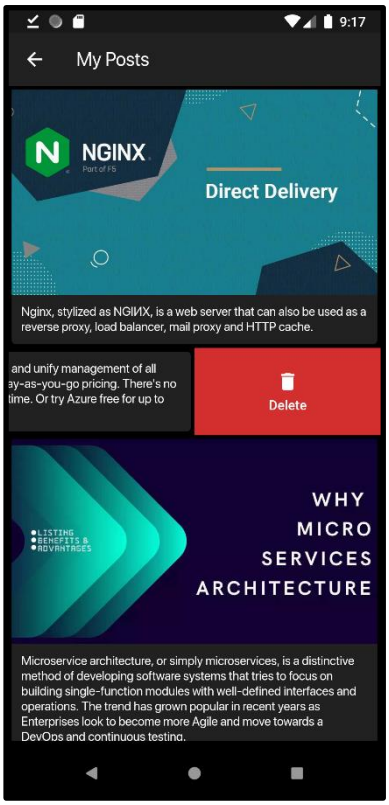
The user has the ability to delete the posts that they have published in the general news feed. This functionality can only be accessed for their particular posts and not the posts that others have submitted. The user can access the posts that they have uploaded by navigating to my posts as explained in the above sections. From there, the user can swipe the expansion tile from the right to the left to delete the post. The expansion tile allows the user to only access the information that is necessary at that time. Further, the swiping capabilities reduce the need for buttons to be implemented which clutters the UI.

User interfaces









Flutter features

- Slider

  This feature is used when deleting or updating a post. This was used to improve the user experience of the application.

  - Instances : Delete and edit post
    - References : https://api.flutter.dev/flutter/material/Slider-class.html
- 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
- Pull to refresh
  - Instances used: Profile , to refresh experiences and other modules
  - References:
    - https://pub.dev/packages/pull_to_refresh

Screens

News Feed Screen

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

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

class NewsFeedScreenState extends State<NewsFeedScreen> {
  Future<List<Post>> posts;

  @override
  void initState() {
    super.initState();
    posts = Provider.of<PostProvider>(context, listen: false).getPosts();
  }

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

    return Scaffold(
      backgroundColor: colorDarkBackground,
      body: Align(
        alignment: Alignment.topCenter,
        child: Stack(
          children: [
            Padding(
              padding: const EdgeInsets.only(top: 60),
              child: FutureBuilder<List<Post>>(
                future: posts,
                builder: (context, snapshot) {
                  return Align(
                    alignment: Alignment.topCenter,
                    child: RefreshIndicator(
                      child: _listView(snapshot),
                      onRefresh: _pullRefresh,
```

```
                ),
              );
            },
          ),
        ),
        const AddPostFeed(),
      ],
    ),
  ),
);
}

Widget _listView(AsyncSnapshot snapshot) {
  if (snapshot.hasData) {
    return ListView.builder(
      itemCount: snapshot.data.length,
      itemBuilder: ((context, index) {
        return PostCard(
          fullName: snapshot.data[index].fullName,
          description: snapshot.data[index].description,
          position: snapshot.data[index].position,
          postImage: snapshot.data[index].postImage,
          profileImageURL: snapshot.data[index].profileImageURL,
        );
      }),
    );
  } else if (snapshot.hasError) {
    return const Text(
      'Error with load posts',
      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,
        ),
      ),
    );
  }

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

My Post Screen

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

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

class _MyPostScreenState extends State<MyPostScreen> {
  Future<List<Post>> userPosts;
  PostProvider _postProvider;
  UserProvider _userProvider;

  @override
  void initState() {
    super.initState();
    _userProvider = context.read<UserProvider>();
    _postProvider = context.read<PostProvider>();

    userPosts =
```

```
      Provider.of<UserProvider>(context, listen: false).getUserPosts(context);
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Text(
        "My Posts",
        style: TextStyle(fontFamily: fontFamilySFPro),
      ),
      backgroundColor: colorDarkMidGround,
      elevation: 0.0,
    ),
    backgroundColor: colorDarkBackground,
    body: Align(
      alignment: Alignment.topCenter,
      child: Padding(
        padding: const EdgeInsets.only(top: 0, bottom: 10),
        child: FutureBuilder<List<Post>>(
          future: userPosts,
          builder: (context, snapshot) {
            return Align(
              alignment: Alignment.topCenter,
              child: RefreshIndicator(
                child: _listView(snapshot),
                onRefresh: _pullRefresh,
              ),
            );
          },
        ),
      ),
    ),
  );
}

Widget _listView(AsyncSnapshot snapshot) {
  if (snapshot.hasData) {
    return ListView.builder(
```

```
itemCount: snapshot.data.length,
itemBuilder: ((context, index) {
  return Slidable(
    closeOnScroll: true,
    key: Key(snapshot.data[index].id),
    child: MyPostCard(
      fullName: snapshot.data[index].fullName,
      description: snapshot.data[index].description,
      position: snapshot.data[index].position,
      postImage: snapshot.data[index].postImage,
      profileImageURL: snapshot.data[index].profileImageURL,
    ),
    endActionPane: ActionPane(
      dismissible: DismissiblePane(
        onDismissed: () {
          _postProvider.deletePost(context, snapshot.data[index].id);
          _userProvider.getProfile(context);
        },
      ),
      children: [
        SlidableAction(
          onPressed: (context) {
            print("Delete");
          },
          backgroundColor: colorError,
          foregroundColor: colorTextPrimary,
          spacing: 5,
          icon: Icons.delete,
          label: 'Delete',
        ),
      ],
      motion: const DrawerMotion(),
    ),
    startActionPane: ActionPane(
      motion: const DrawerMotion(),
      children: [
        SlidableAction(
          onPressed: (context) {
            Navigator.push(
```

```
                    context,
                    MaterialPageRoute(
                      builder: (context) => EditPost(
                        id: snapshot.data[index].id,
                        description: snapshot.data[index].description,
                        imageURL: snapshot.data[index].postImage,
                      ),
                    ),
                  );
                },
                backgroundColor: colorTextPrimary,
                foregroundColor: colorDarkBackground,
                spacing: 5,
                icon: Icons.edit,
                label: 'Edit',
              ),
            ],
          ),
        );
      }),
    );
  } else if (snapshot.hasError) {
    return const Text(
      'Error with load posts',
      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,
```

```
        ),
      ),
    );
  }

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

Edit Post Screen

```
class EditPost extends StatefulWidget {
  final String id;
  final String description;
  final String imageURL;

  const EditPost({
    Key key,
    this.id,
    this.description,
    this.imageURL,
  }) : super(key: key);

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

class _EditPostState extends State<EditPost> {
  String _newImageUrl = '';
  String _newDescription = '';
  String _id = '';
  PostProvider _postProvider;

  @override
  void initState() {
```

```
  super.initState();
  _id = widget.id;
  _newDescription = widget.description;
  _newImageUrl = widget.imageURL;
  _postProvider = context.read<PostProvider>();
}

void submit() {
  _postProvider.updatePost(_id, _newDescription, _newImageUrl, context);
}

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

  return Scaffold(
    appBar: AppBar(
      backgroundColor: colorDarkBackground,
      iconTheme: const IconThemeData(color: Colors.white),
      elevation: 0.0,
      title: const Text(
        "Edit Post",
        style: TextStyle(
          fontFamily: "SF-Pro",
          color: colorTextPrimary,
        ),
      ),
    ),
    backgroundColor: colorDarkBackground,
    body: SingleChildScrollView(
      child: SizedBox(
        width: size.width,
        height: orientation == Orientation.landscape
            ? size.height * 0.95
            : size.height * 0.7,
        child: Padding(
          padding: const EdgeInsets.all(5),
          child: Form(
```

```
      child: Column(children: [
        RoundedTextArea(
          backgroundColor: colorDarkBackground,
          textAreaColor: colorDarkMidGround,
          text: 'Description',
          value: _newDescription,
          onChange: (value) {
            setState(() {
              _newDescription = value;
            });
          },
        ),
        const SizedBox(
          height: 20,
        ),
        _PostEditImageUpload(
          imageUrl: _newImageUrl,
          onFileChanged: (imageUrl) {
            setState(() {
              _newImageUrl = imageUrl;
            });
          },
        ),
        const SizedBox(
          height: 20,
        ),
        RoundedButton(
          color: colorTextPrimary,
          textColor: colorDarkBackground,
          fontSize: 16,
          height: 45,
          width: size.width * 0.92,
          text: 'Save',
          onPressed: () {
            submit();
          },
        )
      ]),
    ),
```

```
          ),
        ),
      ),
    );
  }
}

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

  _PostEditImageUpload({
    Key key,
    this.imageUrl,
    this.onFileChanged,
  }) : super(key: key);

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

class _PostImageUploadState extends State<_PostEditImageUpload> {
  final ImagePicker _picker = ImagePicker();
  String _newImage = '';

  @override
  void initState() {
    super.initState();
    _newImage = widget.imageUrl;
  }

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

    return Column(
      children: [
        if (_newImage != '')
          InkWell(
```

```
    splashColor: Colors.transparent,
    highlightColor: Colors.transparent,
    onTap: () => _selectPhoto(),
    child: ClipRRect(
      borderRadius: BorderRadius.circular(20),
      child: Image.network(
        _newImage,
        width: size.width * 0.89,
      ),
    ),
  ),
if (_newImage != '')
  const SizedBox(
    height: 20,
  ),
Align(
  alignment: Alignment.topLeft,
  child: Padding(
    padding: const EdgeInsets.only(left: 12, right: 12),
    child: Container(
      decoration: BoxDecoration(
        borderRadius: BorderRadius.circular(
          20,
        ),
        color: colorDarkMidGround,
      ),
      child: InkWell(
        onTap: () => _selectPhoto(),
        child: Padding(
          padding: const EdgeInsets.only(
            top: 10,
            bottom: 10,
            left: 20,
            right: 20,
          ),
          child: Row(
            children: [
              const Icon(
                Icons.image,
```

```
                    size: 18,
                    color: colorTextPrimary,
                  ),
                  Text(
                    widget.imageUrl != null
                        ? " Change photo"
                        : " Select photo",
                    style: const TextStyle(
                      fontFamily: fontFamilySFPro,
                      fontSize: 16,
                      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: 16,
      ratioY: 9,
```

```
      ),
    );
    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(() {
      _newImage = fileUrl;
    });

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

My Post Card

```
class MyPostCard extends StatefulWidget {
  final String fullName;
  final String position;
  final String profileImageURL;
  final String postImage;
  final String description;

  const MyPostCard({
    Key key,
```

```
    this.fullName,
    this.position,
    this.profileImageURL,
    this.postImage,
    this.description,
  }) : super(key: key);

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

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

    return Column(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        Container(
          width: orientation == Orientation.landscape
              ? size.width * 0.75
              : size.width,
          padding: const EdgeInsets.only(top: 0),
          child: Card(
            color: colorDarkMidGround,
            child: Column(
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                if (widget.postImage != '') Image.network(widget.postImage),
                Padding(
                  padding: const EdgeInsets.all(10.0),
                  child: Text(
                    widget.description,
                    style: const TextStyle(
                      fontFamily: fontFamilySFPro,
                      fontSize: 13,
                      color: Colors.white,
```

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

Models

Post Model

```
class Post {
  String id;
  String fullName;
  String position;
  String profileImageURL;
  String postImage;
  String description;

  Post.createPostConstructor({
    this.id,
    this.fullName,
    this.position,
    this.profileImageURL,
    this.postImage,
    this.description,
  });

  Post({
    this.id,
    this.fullName,
    this.position,
    this.profileImageURL,
```

```
    this.postImage,
    this.description,
  });

  factory Post.fromJson(
    Map<String, dynamic> json,
  ) =>
      Post(
        id: json['_id'],
        fullName: json['fullName'],
        position: json['position'],
        profileImageURL: json['profileImageURL'],
        postImage: json['postImage'],
        description: json['description'],
      );

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

Providers

Post Provider

```
class PostProvider extends ChangeNotifier {
  Post post = Post.createPostConstructor(
    description: '',
    fullName: '',
    position: '',
    id: '',
    postImage: '',
    profileImageURL: '',
  );
  List<Post> posts = [];
```

```dart
final storage = const FlutterSecureStorage();

Future createPost(
  BuildContext context,
  String firstName,
  String lastName,
  String position,
  String profileImageURL,
) async {
  var userId = await storage.read(key: 'userId');
  var authToken = await storage.read(key: 'authToken');

  final response = await http.post(
    Uri.parse('$baseApi/posts/user/$userId'),
    headers: <String, String>{
      'Content-Type': 'application/json; charset=UTF-8',
      'x-auth-token': authToken,
    },
    body: jsonEncode(
      <String, String>{
        'fullName': firstName + " " + lastName,
        'position': position,
        'profileImageURL': profileImageURL,
        'postImage': post.postImage,
        'description': post.description,
      },
    ),
  );

  if (response.statusCode == 200) {
    final data = jsonDecode(response.body);
    post = Post.fromJson(data);
    posts.add(post);
    post.description = '';
    post.postImage = '';
    notifyListeners();
  } else if (response.statusCode == 400) {
    Fluttertoast.showToast(msg: 'Authentication Failed');
    Navigator.pushNamed(context, '/login');
```

```
    notifyListeners();
  } else {
    Fluttertoast.showToast(msg: 'Server Error');
    notifyListeners();
  }
}

Future<List<Post>> getPosts() async {
  final response = await http.get(
    Uri.parse('$baseApi/posts/'),
    headers: <String, String>{
      'Content-Type': 'application/json; charset=UTF-8',
    },
  );

  if (response.statusCode == 200) {
    posts.clear();
    final data = jsonDecode(response.body) as List;
    final List<Post> newPosts = [];

    for (Map<String, dynamic> post in data) {
      newPosts.add(Post.fromJson(post));
    }

    return newPosts;
  } else {
    Fluttertoast.showToast(msg: 'Server Error');
  }
  notifyListeners();
  return null;
}

void deletePost(BuildContext context, String postId) async {
  var userId = await storage.read(key: 'userId');
  var authToken = await storage.read(key: 'authToken');
  final response = await http.delete(
    Uri.parse('$baseApi/posts/remove/$userId/$postId'),
    headers: <String, String>{
      'Content-Type': 'application/json; charset=UTF-8',
```

```
      'x-auth-token': authToken,
    },
  );

  if (response.statusCode == 200) {
    notifyListeners();
  } else if (response.statusCode == 400) {
    Fluttertoast.showToast(msg: 'Authentication Failed');
    Navigator.pushNamed(context, '/login');
    notifyListeners();
  } else {
    Fluttertoast.showToast(msg: 'Server Error');
    notifyListeners();
  }
}

void updatePost(
  String id,
  String description,
  String imageURL,
  BuildContext context,
) async {
  var authToken = await storage.read(key: 'authToken');
  final response = await http.put(
    Uri.parse('$baseApi/posts/edit/$id'),
    headers: <String, String>{
      'Content-Type': 'application/json; charset=UTF-8',
      'x-auth-token': authToken,
    },
    body: jsonEncode(
      <String, String>{
        'postImage': imageURL,
        'description': description,
      },
    ),
  );

  if (response.statusCode == 200) {
    Navigator.pop(context);
```

```
      notifyListeners();
    } else if (response.statusCode == 400) {
      Fluttertoast.showToast(msg: 'Authentication Failed');
      Navigator.pushNamed(context, '/login');
      notifyListeners();
    } else {
      Fluttertoast.showToast(msg: 'Server Error');
      notifyListeners();
    }
  }
}
```

Components

Add Post Feed

```
class AddPostFeed extends StatefulWidget {
  const AddPostFeed({
    Key key,
  }) : super(key: key);

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

class _AddPostFeedState extends State<AddPostFeed> {
  String description = "";
  PostProvider postProvider;
  UserProvider userProvider;

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

  @override
  Widget build(BuildContext context) {
```

```
final size = MediaQuery.of(context).size;
return SingleChildScrollView(
  child: ExpansionTile(
    initiallyExpanded: false,
    collapsedBackgroundColor: colorDarkForground,
    backgroundColor: colorDarkForground,
    trailing: const Icon(
      Icons.add,
      color: colorTextPrimary,
      size: 20,
    ),
    title: const Text(
      'Create Post',
      style: TextStyle(
        fontFamily: fontFamilySFPro,
        color: colorTextPrimary,
        fontSize: 18,
      ),
    ),
    children: [
      Column(
        children: [
          Card(
            color: colorDarkForground,
            child: Container(
              padding: const EdgeInsets.only(
                top: 10,
                left: 25,
                right: 25,
                bottom: 5,
              ),
              child: Column(
                children: [
                  RoundedTextArea(
                    backgroundColor: colorDarkForground,
                    textAreaColor: colorDarkMidGround,
                    text: "Description",
                    onChange: (value) {
                      setState(() {
```

```
            postProvider.post.description = value;
          });
        },
        value: postProvider.post.description,
      ),
      SizedBox(
        height: size.height * 0.03,
      ),
      PostImageUpload(
        onFileChanged: (imageUrl) {
          setState(() {
            postProvider.post.postImage = imageUrl;
          });
        },
      ),
      SizedBox(
        height: size.height * 0.03,
      ),
      RoundedButton(
        color: colorTextPrimary,
        textColor: colorDarkBackground,
        fontSize: 14,
        height: 35,
        width: size.width * 0.85,
        text: "Post",
        onPressed: () {
          postProvider.createPost(
            context,
            userProvider.user.firstName,
            userProvider.user.lastName,
            userProvider.user.position,
            userProvider.user.profileImageURL,
          );
        },
      ),
      SizedBox(
        height: size.height * 0.03,
      ),
    ],
```

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

Post Card

```
class PostCard extends StatelessWidget {
  final String fullName;
  final String position;
  final String profileImageURL;
  final String postImage;
  final String description;

  const PostCard({
    Key key,
    this.fullName,
    this.position,
    this.profileImageURL,
    this.postImage,
    this.description,
  }) : super(key: key);

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

    return Column(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        Container(
          width: orientation == Orientation.landscape
```

```
      ? size.width * 0.75
      : size.width,
  padding: const EdgeInsets.only(top: 0),
  child: Card(
    color: colorDarkMidGround,
    child: Column(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        Padding(
          padding: const EdgeInsets.all(8.0),
          child: Row(
            // Header section
            children: [
              ClipRRect(
                borderRadius: BorderRadius.circular(100),
                child: Image.network(
                  profileImageURL,
                  fit: BoxFit.cover,
                  height: 30,
                  width: 30,
                ),
              ),
              Padding(
                padding: const EdgeInsets.only(left: 8),
                child: Column(
                  crossAxisAlignment: CrossAxisAlignment.start,
                  children: [
                    Text(
                      fullName,
                      style: const TextStyle(
                        fontFamily: fontFamilySFPro,
                        fontWeight: FontWeight.bold,
                        fontSize: 16,
                        color: Colors.white,
                      ),
                    ),
                    Text(
                      position,
                      style: const TextStyle(
```

```
                                fontFamily: fontFamilySFPro,
                                fontSize: 14,
                                color: Colors.white,
                              ),
                            )
                          ],
                        ),
                      )
                    ],
                  ),
                ),
                if (postImage != '') Image.network(postImage),
                Padding(
                  padding: const EdgeInsets.all(10.0),
                  child: Text(
                    description,
                    style: const TextStyle(
                      fontFamily: fontFamilySFPro,
                      fontSize: 13,
                      color: Colors.white,
                    ),
                  ),
                ),
              ],
            ),
          ),
        )
      ],
    );
  }
}
```

Post Image Upload

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

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

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

class _PostImageUploadState extends State<PostImageUpload> {
  final ImagePicker _picker = ImagePicker();
  String imagePath;

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

    return Column(
      children: [
        if (imagePath != null)
          InkWell(
            splashColor: Colors.transparent,
            highlightColor: Colors.transparent,
            onTap: () => _selectPhoto(),
```

```
      child: ClipRRect(
        borderRadius: BorderRadius.circular(20),
        child: Image.network(
          imagePath,
          width: size.width * 0.83,
        ),
      ),
    ),
  Align(
    alignment: Alignment.topLeft,
    child: Padding(
      padding: const EdgeInsets.only(left: 4, right: 4, top: 10),
      child: Container(
        decoration: BoxDecoration(
          borderRadius: BorderRadius.circular(
            20,
          ),
          color: colorDarkMidGround,
        ),
        child: InkWell(
          onTap: () => _selectPhoto(),
          child: Padding(
            padding: const EdgeInsets.only(
              top: 10,
              bottom: 10,
              left: 20,
              right: 20,
            ),
            child: Row(
              children: [
                const Icon(
                  Icons.image,
                  size: 18,
                  color: colorTextPrimary,
                ),
                Text(
                  imagePath != null ? " Change photo" : " Select photo",
                  style: const TextStyle(
                    fontFamily: fontFamilySFPro,
```

```
                        fontSize: 16,
                        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: 16,
      ratioY: 9,
    ),
  );
  if (file == null) {
    return;
  }

  await _uploadFile(file.path);
}
```

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