**NAME-Prathamesh P Chawande 3rd Year contact-8767729487**

**Case Study-Mobile Developer Flutter**

**Background:-**

Bynry is gearing up to introduce a new feature to its SAAS product, necessitating improvements to its existing mobile application. This application, crafted using Flutter, serves the consumers of utility companies, enabling them to access a range of functionalities from signing up and logging in to managing their accounts and paying bills. The primary goal of this assignment is to implement the necessary enhancements to accommodate the new feature within the mobile application. You are expected to focus on both the front-end and back-end aspects, ensuring a seamless user experience and integration with the existing functionalities.

**1. Set Up the Development Environment**

1.1 Install Flutter

Follow the official Flutter installation guide for your operating system. Ensure you have an IDE like Visual Studio Code or Android Studio with Flutter and Dart plugins installed.

1.2 Clone the Repository

**git clone https://github.com/your-username/bynry-mobile-app.git**

**cd bynry-mobile-app**

**2. Login Screen Improvements**

2.1 Enhance Authentication

Add new authentication fields or methods if required (e.g., two-factor authentication). Modify the login form to include any new fields.

2.2 Input Validation and Error Handling

Use Text Form Field with validators to ensure data integrity and enhance error handling.

We can use the below code:-

**File: lib/screens/login\_screen.dart**

import 'package:flutter/material.dart';

import 'package:firebase\_auth/firebase\_auth.dart';

import 'package:provider/provider.dart';

import '../services/auth\_service.dart';

import '../widgets/custom\_button.dart';

import '../widgets/custom\_text\_field.dart';

class LoginScreen extends StatefulWidget {

  @override

  \_LoginScreenState createState() => \_LoginScreenState();

}

class \_LoginScreenState extends State<LoginScreen> {

  final \_formKey = GlobalKey<FormState>();

  final TextEditingController \_emailController = TextEditingController();

  final TextEditingController \_passwordController = TextEditingController();

  bool \_isLoading = false;

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: Text('Login'),

      ),

      body: Padding(

        padding: const EdgeInsets.all(16.0),

        child: Form(

          key: \_formKey,

          child: Column(

            children: [

              CustomTextField(

                controller: \_emailController,

                labelText: 'Email',

                keyboardType: TextInputType.emailAddress,

                validator: (value) {

                  if (value == null || value.isEmpty) {

                    return 'Please enter your email';

                  }

                  if (!RegExp(r'^[^@]+@[^@]+\.[^@]+').hasMatch(value)) {

                    return 'Please enter a valid email address';

                  }

                  return null;

                },

              ),

              CustomTextField(

                controller: \_passwordController,

                labelText: 'Password',

                obscureText: true,

                validator: (value) {

                  if (value == null || value.isEmpty) {

                    return 'Please enter your password';

                  }

                  if (value.length < 6) {

                    return 'Password must be at least 6 characters long';

                  }

                  return null;

                },

              ),

              SizedBox(height: 20),

              \_isLoading

                  ? CircularProgressIndicator()

                  : CustomButton(

                      text: 'Login',

                      onPressed: \_login,

                    ),

              TextButton(

                onPressed: \_navigateToSignUp,

                child: Text('Don\'t have an account? Sign up'),

              ),

              TextButton(

                onPressed: \_navigateToResetPassword,

                child: Text('Forgot password? Reset'),

              ),

            ],

          ),

        ),

      ),

    );

  }

  Future<void> \_login() async {

    if (\_formKey.currentState!.validate()) {

      setState(() {

        \_isLoading = true;

      });

      try {

        await context.read<AuthService>().signIn(

              email: \_emailController.text,

              password: \_passwordController.text,

            );

        Navigator.pushReplacementNamed(context, '/dashboard');

      } on FirebaseAuthException catch (e) {

        String message = 'An error occurred, please try again';

        if (e.code == 'user-not-found') {

          message = 'No user found for that email.';

        } else if (e.code == 'wrong-password') {

          message = 'Wrong password provided.';

        }

        ScaffoldMessenger.of(context).showSnackBar(

          SnackBar(content: Text(message)),

        );

      } finally {

        setState(() {

          \_isLoading = false;

        });

      }

    }

  }

  void \_navigateToSignUp() {

    Navigator.pushNamed(context, '/signup');

  }

  void \_navigateToResetPassword() {

    Navigator.pushNamed(context, '/reset-password');

  }

}

**File: lib/services/auth\_service.dart**

import 'package:firebase\_auth/firebase\_auth.dart';

class AuthService {

  final FirebaseAuth \_auth = FirebaseAuth.instance;

  Future<User?> signIn({required String email, required String password}) async {

    UserCredential result = await \_auth.signInWithEmailAndPassword(

      email: email,

      password: password,

    );

    return result.user;

  }

  Future<User?> signUp({required String email, required String password}) async {

    UserCredential result = await \_auth.createUserWithEmailAndPassword(

      email: email,

      password: password,

    );

    return result.user;

  }

  Future<void> resetPassword({required String email}) async {

    await \_auth.sendPasswordResetEmail(email: email);

  }

  Future<void> signOut() async {

    await \_auth.signOut();

  }

}

**File: lib/widgets/custom\_button.dart**

import 'package:flutter/material.dart';

class CustomButton extends StatelessWidget {

  final String text;

  final VoidCallback onPressed;

  const CustomButton({

    Key? key,

    required *this*.text,

    required *this*.onPressed,

  }) : *super*(key: key);

  @override

  Widget build(BuildContext context) {

    return ElevatedButton(

      onPressed: onPressed,

      child: Text(text),

    );

  }

}

**File: lib/widgets/custom\_text\_field.dart**

import 'package:flutter/material.dart';

class CustomTextField extends StatelessWidget {

  final TextEditingController controller;

  final String labelText;

  final TextInputType keyboardType;

  final bool obscureText;

  final FormFieldValidator<String> validator;

  const CustomTextField({

    Key? key,

    required *this*.controller,

    required *this*.labelText,

*this*.keyboardType = TextInputType.text,

*this*.obscureText = false,

    required *this*.validator,

  }) : *super*(key: key);

  @override

  Widget build(BuildContext context) {

    return TextFormField(

      controller: controller,

      decoration: InputDecoration(labelText: labelText),

      keyboardType: keyboardType,

      obscureText: obscureText,

      validator: validator,

    );

  }

}

**3. Dashboard Updates**

3.1 UI/UX Design

Update the dashboard to include a new button or card for the new feature. Ensure the design aligns with the overall aesthetic of the application.

3.2 Integration

Implement navigation to the new feature screen.

**File: lib/screens/dashboard\_screen.dart**

import 'package:flutter/material.dart';

import '../widgets/custom\_button.dart';

import '../widgets/custom\_card.dart';

import '../services/auth\_service.dart';

import 'package:provider/provider.dart';

class DashboardScreen extends StatelessWidget {

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: Text('Dashboard'),

        actions: [

          IconButton(

            icon: Icon(Icons.logout),

            onPressed: () async {

              await context.read<AuthService>().signOut();

              Navigator.pushReplacementNamed(context, '/login');

            },

          ),

        ],

      ),

      body: Padding(

        padding: const EdgeInsets.all(16.0),

        child: SingleChildScrollView(

          child: Column(

            children: [

              CustomCard(

                icon: Icons.payment,

                title: 'Pay Bills',

                description: 'Pay your utility bills easily.',

                onTap: () {

                  Navigator.pushNamed(context, '/pay-bills');

                },

              ),

              CustomCard(

                icon: Icons.account\_circle,

                title: 'Account Management',

                description: 'Manage your account settings.',

                onTap: () {

                  Navigator.pushNamed(context, '/account-management');

                },

              ),

              CustomCard(

                icon: Icons.new\_releases,

                title: 'New Feature',

                description: 'Access the new feature here.',

                onTap: () {

                  Navigator.pushNamed(context, '/new-feature');

                },

              ),

            ],

          ),

        ),

      ),

    );

  }

}

**File: lib/widgets/custom\_card.dart**

import 'package:flutter/material.dart';

class CustomCard extends StatelessWidget {

  final IconData icon;

  final String title;

  final String description;

  final VoidCallback onTap;

  const CustomCard({

    Key? key,

    required *this*.icon,

    required *this*.title,

    required *this*.description,

    required *this*.onTap,

  }) : *super*(key: key);

  @override

  Widget build(BuildContext context) {

    return Card(

      elevation: 4.0,

      margin: EdgeInsets.symmetric(vertical: 10.0),

      child: InkWell(

        onTap: onTap,

        child: Padding(

          padding: const EdgeInsets.all(16.0),

          child: Row(

            children: [

              Icon(icon, size: 40.0, color: Theme.of(context).primaryColor),

              SizedBox(width: 20.0),

              Column(

                crossAxisAlignment: CrossAxisAlignment.start,

                children: [

                  Text(

                    title,

                    style: TextStyle(

                      fontSize: 20.0,

                      fontWeight: FontWeight.bold,

                    ),

                  ),

                  SizedBox(height: 10.0),

                  Text(

                    description,

                    style: TextStyle(

                      fontSize: 16.0,

                      color: Colors.grey[700],

                    ),

                  ),

                ],

              ),

            ],

          ),

        ),

      ),

    );

  }

}

**Navigation setup-**

**File: lib/main.dart**

import 'package:flutter/material.dart';

import 'package:provider/provider.dart';

import 'screens/login\_screen.dart';

import 'screens/dashboard\_screen.dart';

import 'screens/pay\_bills\_screen.dart';

import 'screens/account\_management\_screen.dart';

import 'screens/new\_feature\_screen.dart';

import 'services/auth\_service.dart';

void main() {

  runApp(MyApp());

}

class MyApp extends StatelessWidget {

  @override

  Widget build(BuildContext context) {

    return MultiProvider(

      providers: [

        Provider<AuthService>(create: (\_) => AuthService()),

      ],

      child: MaterialApp(

        title: 'Bynry Utility App',

        theme: ThemeData(

          primarySwatch: Colors.blue,

        ),

        initialRoute: '/',

        routes: {

          '/': (context) => LoginScreen(),

          '/login': (context) => LoginScreen(),

          '/dashboard': (context) => DashboardScreen(),

          '/pay-bills': (context) => PayBillsScreen(),

          '/account-management': (context) => AccountManagementScreen(),

          '/new-feature': (context) => NewFeatureScreen(),

        },

      ),

    );

  }

}

**File: lib/screens/new\_feature\_screen.dart**

import 'package:flutter/material.dart';

class NewFeatureScreen extends StatelessWidget {

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: Text('New Feature'),

      ),

      body: Center(

        child: Text(

          'This is the new feature screen',

          style: TextStyle(fontSize: 24.0),

        ),

      ),

    );

  }

}

**Here is a detailed README file that includes instructions on how to set up, run, and test the Flutter application.**

Here is a detailed README file that includes instructions on how to set up, run, and test the Flutter application.

**Bynry Mobile App**

**Introduction**

This Flutter application serves the consumers of utility companies, enabling them to access functionalities such as signing up, logging in, managing accounts, and paying bills. The recent updates include enhanced authentication requirements and access to a new feature from the dashboard.

1. **Features**

- Sign up and log in

- Manage accounts

- Pay bills

- Access new feature

1. **Setup Instructions**

- Ensure you have Flutter installed on your machine. Follow the [official Flutter installation guide](https://flutter.dev/docs/get-started/install) for your operating system.

- You need to have an IDE like Visual Studio Code or Android Studio with Flutter and Dart plugins installed.

Step 1: Clone the Repository

Clone the repository to your local machine using the following command:

git clone https://github.com/your-username/bynry-mobile-app.git

Step 2: Navigate to the Project Directory

Navigate to the project directory:

cd bynry-mobile-app

Step 3: Install Dependencies

Install the required dependencies using the following command:

flutter pub get

Step 4: Configure Firebase (if applicable)

If the app uses Firebase for authentication or other services, ensure you add your `google-services.json` (for Android) and `GoogleService-Info.plist` (for iOS) in the respective directories.

Step 5: Run the Application

Run the application on an emulator or physical device:

flutter run

1. **Project Structure**

The project follows a standard Flutter project structure:

- main.dart Entry point of the application

- screens/ Directory containing screen widgets

- login\_screen.dart

- dashboard\_screen.dart

- new\_feature\_screen.dart

- widgets/ Directory containing reusable widgets

- models/ Directory containing data models

- services/ Directory containing service classes (e.g., authentication, API calls)

1. **Usage Instructions**

Login

1. Open the app.

2. Enter your email and password.

3. Click on 'Login'.

4. If you encounter errors, follow the on-screen instructions to correct them.

Dashboard

1. After logging in, you will see the dashboard.

2. Click on the 'New Feature' button to access the new feature.

3. Follow the instructions provided on the new feature screen.

Testing

Running Unit Tests

To ensure the application is working correctly, you should run the provided unit tests. Use the following command to run the tests:

flutter test

Testing Tips

- Verify that the login validation works correctly.

- Check that the new feature is accessible from the dashboard.

- Ensure there are no runtime errors or warnings.

1. **Technical Documentation**

New Feature Overview

- Feature Name: Enhanced Authentication and Dashboard Access

- Purpose: To improve user authentication and provide access to a new feature directly from the dashboard.

- Integration Points:

- Login Screen: Added validation and error handling.

- Dashboard: Added a button to navigate to the new feature.

Changes to Login Screen

- Added input validation for email and password fields.

- Enhanced error handling for better user experience.

Changes to Dashboard

- Added a button to access the new feature.

- Ensured the design aligns with the app's aesthetic.

Dependencies

- No new dependencies were added.

Login

1. Open the app.

2. Enter your email and password.

3. Click on 'Login'.

4. If you encounter errors, follow the on-screen instructions to correct them.

Dashboard

1. After logging in, you will see the dashboard.

2. Click on the 'New Feature' button to access the new feature.

3. Follow the instructions provided on the new feature screen.

**Contributors**

- Prathamesh P Chawande

**License**

This project is licensed under the MIT License.