Module	Portfolio	Assessment Type
Collaborative Development (5CS024)	1	Individual Report

EXPENSES MANAGEMENT SYSTEM DEVELOPER

Student Id : 2227486

Student Name : Nayan Raj Khanal

Section : L5CG4

Group : L5CG4 Group 5

Role : Developer

Instructor : Mr. Adarsha Khadka

Submitted on : 15th May 2023

Word Count : 3476

Acknowledgement:

None of this would have been possible if it was not for the University of Wolverhampton and Herald College Kathmandu, the two esteemed institutes that gave me the chance to pursue higher education for that I am filled with gratitude.

My heartfelt appreciation also goes to our instructor, Mr Adarsha Khadka, whose dedication to teaching alongside constant guidance and bright ideas inspired me to aim for academic excellence.

I would also like to extend my appreciation to Mr Biraj Dulal, Mr Anmol Adhikari and all the other module members for their commitment to the course and for creating a positive learning environment.

I am also grateful for the internet, which provided me with access to a vast array of study material required for the completion of this report.

Finally, I want to express my heartfelt gratitude to my family and friends for their unwavering love, support, and motivation throughout my academic career.

Table of Contents

Table of Figures

1. Self-appraisal form

Student number	2227486	Name	Nayan Raj Khanal
Project	Expenses Management System	Date	
Role	Developer	Team	L5CG4 (Group 5)
Sprint (1 or 2)	2		

2. Personal objectives – performance measurement

Objectives	**Evidence provided**	**Evaluation** *Student / tutor*	
Implementing functional requirements	**From having zero knowledge about mobile application development to deploying a fully functional one is surreal to me. Being able to support the team by taking on full responsibility for the front-end part makes me so proud of myself. Working day and night and continuously collaborating with my teammates, I feel like I have both enhanced my personal and team skills.** **Provided screenshots of codes, flowchart of the system and commit logs.** **<u>Appendix A1</u>**	**9**	
Tutor feedback:			

Bug Fixing	**Being new to developing apps I was bound to cause errors resulting in bugs. The motivation to prove to myself led me to not be discouraged when faced with bugs and instead look forward to them and fix them. By doing so, I found myself learning a lot more than watching YouTube videos.** **Provided screenshots of White-Box Testing and GitHub Issues.** **<u>Appendix A2</u>**	**9**	
Tutor feedback:			
		/20	/20

3. Collaboration Document

3.1. Evidence of good communication and file sharing

Teamwork makes the dream work. Even though the team had its fair share of problems at the end of the day we were able to complete and submit the project we promised. This would not have been possible without the effort of all the members. Upon completion of this project, I feel like I have learnt a lot about the team environment and in the future, I won't be getting a lot of surprises.

Appendix B1

3.2. Continuing Personal Development (CPD)

No one is perfect, but one can strive for perfection. Looking back, I was a nobody in developing space but now I feel like I am equipped enough to develop apps. I feel like I have improved my technical skillset alongside my willpower and discipline. I was able to develop a mobile app now I look forward to developing a website using the MERN stack, sky is the limit.

Appendix B2

3.3. Issue tracking

No matter how much one tries, errors are bound to follow. Making mistakes is not a bad thing, one should strive to do better. In my case, I made a lot of errors but due to my determination and team, I was able to overcome them. As a team BA and I, we solved numerous errors and were able to deliver a bug-free product and we have to thank GitHub Issues for it.

Appendix B3

4. Appendix A

4.1. Implementing functional requirement

Being one of the developers working on "Kharcha Saathi", I was assigned the entirety of the front-end for the project.

The product backlog is as follows:

Epic	Priority	User stories	Acceptance Criteria (CheckList)	Story Point(Time	Status	Other labels	Story No
User management	high	As a user, I want to be able to create an account and log in to the expense management system	-User can sign up witha valid email and passwordand user can login with the registered email and mpin	15 days	To Do	authentication , signup, login	1
	high	As a user, I want to set an MPIN for added security	-user can set a 4-digit Mpin, MPIN must be validated before being saved	5 days	To Do	security, MPIN	2
Expense management	high	As a user, I want to add my income	-user can enter income details, such as amount, date, description and user can save the income record	5 days	To Do	income, record keeping	3
	high	As a user, I want to add my expenses	- User can enter expense details such as amount, date, category, and description. User can save the expense record	10 days	To Do	expenses,record keeping	4
	high	As a user, I want to view my income and expense records	- User can view all income and expense records in a list view. Records should be sortable and filterable by date and category. Total income and expense should be displayed	10 days	To Do	view, record keeping	5
Expense management	high	As a user, I want to change my MPIN	-User can change the MPIN by entering the old and new MPIN .The new MPIN must be validated before being saved	10 days	To Do	security, MPIN	6
	high	As a user, I want to download a PDF report of my income and expense records	- User can download a PDF report of their income and expense records.The report should include all income and expense records in a table format with total income and expense	15 days	To Do	download report, record keeping	7

Figure 1. Product Backlog

All the product backlog tasks has been checked off. Even though the development process was slower than anticipated, in the end, the product was completed in time and tested successfully, for that, I would like to thank myself, my team, my tutors and all others who were involved.

4.1.1. Use of Version Control

The front-end was developed using Dart as the programming language and Flutter as the framework. Thunder Client was used for building and testing APIs. Git was used as the version control system and GitHub was used as storage for Git repositories. Finally, VS Code was used as a source code editor.

A total of 17 dart files were created for the completion of the project.

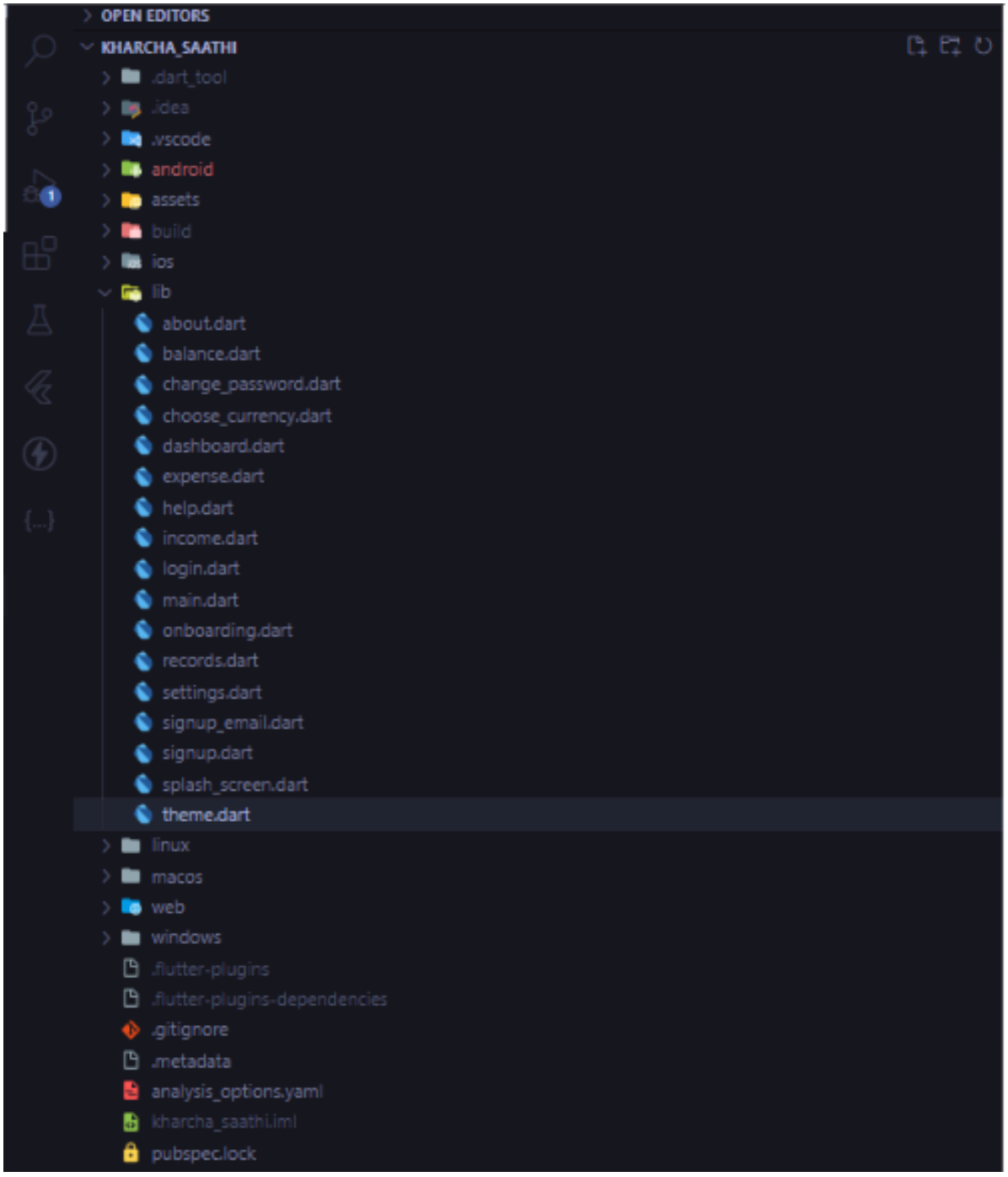


Figure 2. Dart Files

These are all the frames developed according to the product backlog.

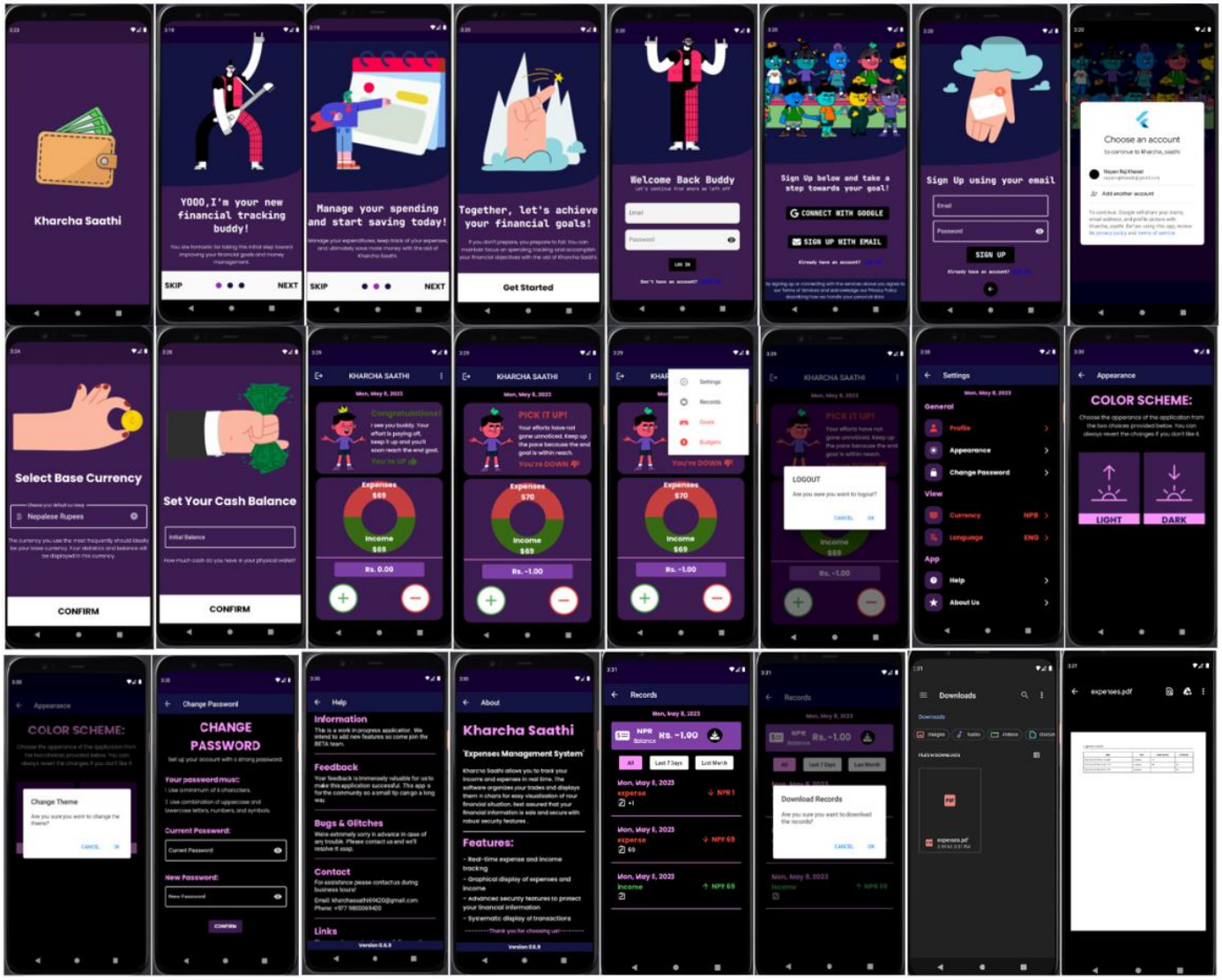


Figure 3. Key Frames

Included below are only the dart files required as per the product backlog with output and proper commented codes alongside brief explanations.

4.1.1.1. login.dart

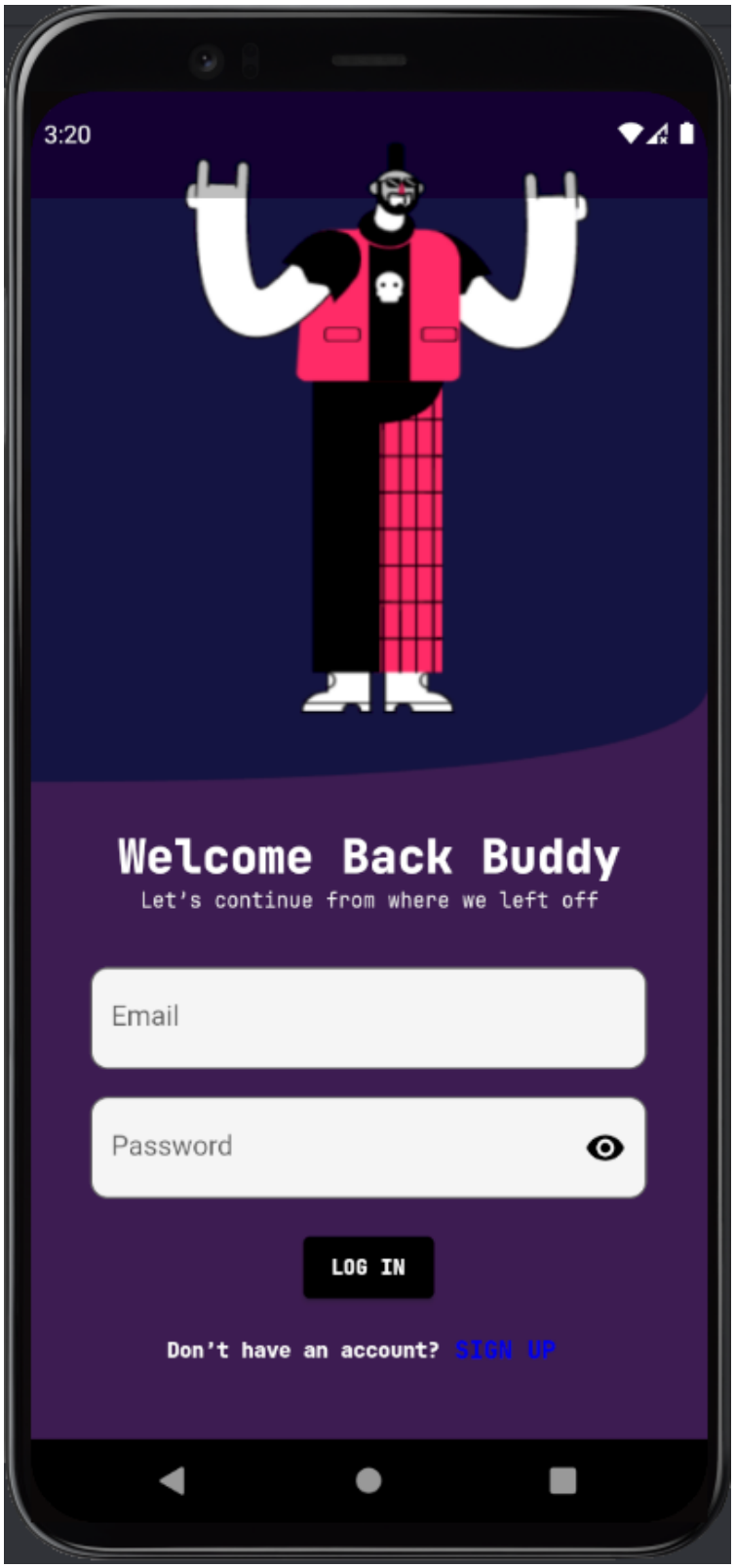


Figure 4. Login (Output)

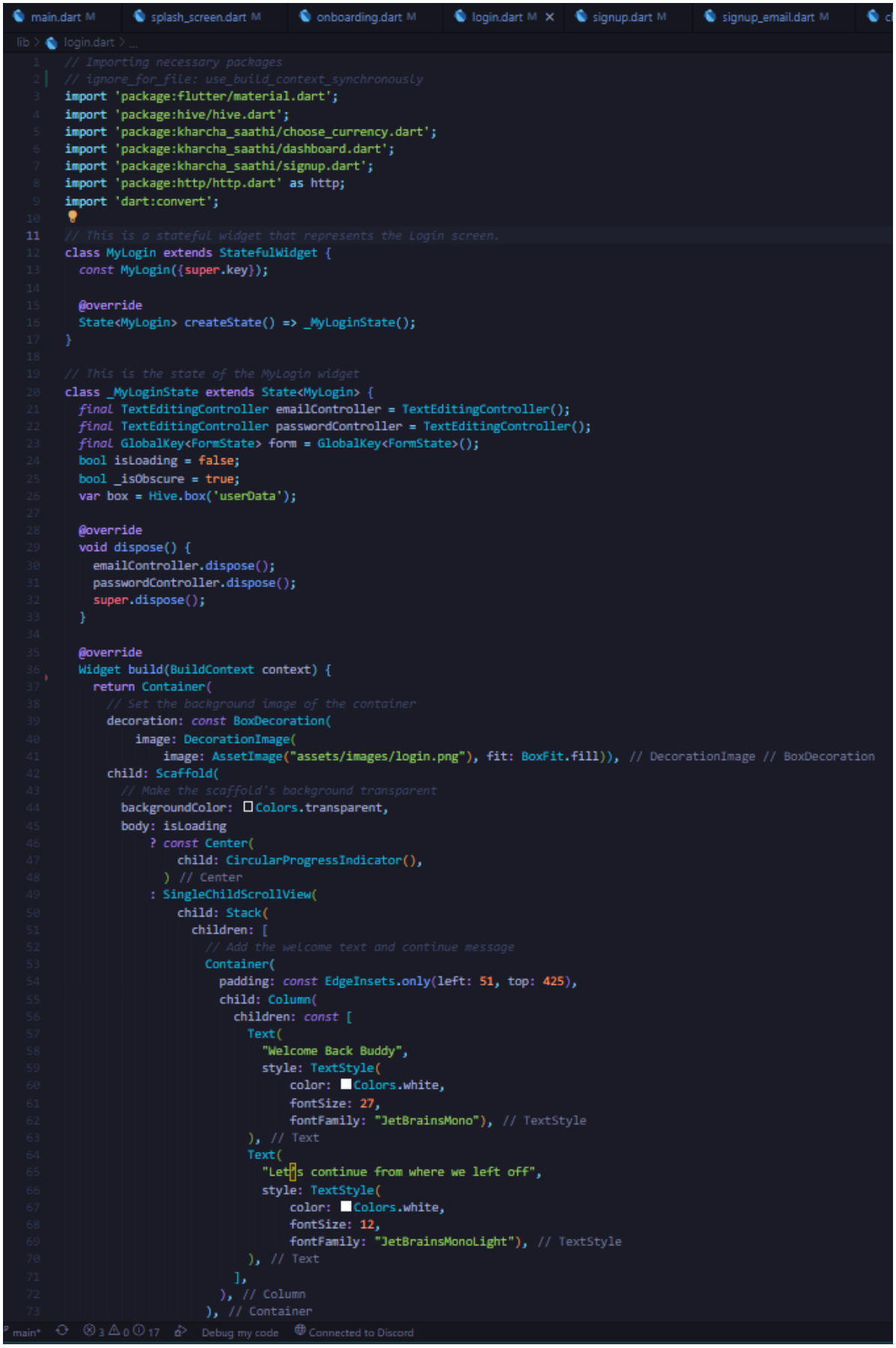


Figure 5. Login (Code 1)

```
Container(
  // Set the top padding to match device height
  padding: EdgeInsets.only(
      top: MediaQuery.of(context).size.height * 0.65,
      right: 35,
      left: 35), // EdgeInsets.only
  child: Column(
    children: [
      // Add the email text field
      TextFormField(
        autovalidateMode:
            AutovalidateMode.onUserInteraction,
        validator: (value) {
          RegExp emailReg = RegExp(
              r"^[a-zA-Z0-9.a-zA-Z0-9.!#$%&'*+-/=?^_`{|}~]+@[a-zA-Z0-9]+\.[a-zA-Z]+"); // RegExp
          if (!emailReg.hasMatch(value.toString())) {
            return "Invalid Email";
          }
          return null;
        },
        controller: emailController,
        decoration: InputDecoration(
            fillColor: Colors.grey.shade100,
            filled: true,
            hintText: "Email",
            border: OutlineInputBorder(
                borderRadius: BorderRadius.circular(10))), // OutlineInputBorder // InputDecoration
      ), // TextFormField
      const SizedBox(
        height: 16,
      ), // SizedBox
      // Add the password text field
      TextFormField(
        autovalidateMode:
            AutovalidateMode.onUserInteraction,
        validator: (value) {
          RegExp passReg = RegExp(
              r'^(?=.*?[A-Z])(?=.*?[a-z])(?=.*?[0-9])(?=.*?[!@#\$&*~]).{8,}$'); // RegExp
          if (!passReg.hasMatch(value.toString())) {
            return "Invalid Password";
          }
          return null;
        },
        controller: passwordController,
        obscureText: _isObscure,
        decoration: InputDecoration(
          suffixIcon: InkWell(
            onTap: () {
              setState(() {
                _isObscure = !_isObscure;
              });
            },
            child: Icon(
              _isObscure
                  ? Icons.visibility
                  : Icons.visibility_off,
              color: Colors.black,
            ), // Icon
          ), // InkWell
          fillColor: Colors.grey.shade100,
          filled: true,
          hintText: "Password",
          border: OutlineInputBorder(
            borderRadius: BorderRadius.circular(10),
          ), // OutlineInputBorder
        ), // InputDecoration
      ), // TextFormField

      const SizedBox(
        height: 16,
      ), // SizedBox
      Column(
        children: [
```

Figure 6. Login (Code 2)

```
ElevatedButton(
  onPressed: () async {
    setState(() {
      isLoading = true;
    });
    http.Response response = await http.post(
        Uri.parse(
            "https://kharchasathi-backend.onrender.com/users/login"),
        body: json.encode({
          "email": emailController.text,
          "password": passwordController.text
        }));

    var jsonResponse = json.decode(response.body);
    if (response.statusCode == 200) {
      ScaffoldMessenger.of(context)
          .showSnackBar(SnackBar(
        content: Text(jsonResponse["message"]),
      )); // SnackBar
      if (jsonResponse["initialCurrency"] == "" &&
          jsonResponse["balance"] == 0) {
        Navigator.of(context).pushReplacement(
          MaterialPageRoute(
              builder: (context) => Mycurrency(
                  accessToken: jsonResponse[
                      "accessToken"])), // Mycurrency // MaterialPageRoute
        );
      } else {
        Navigator.of(context).pushReplacement(
          MaterialPageRoute(
              builder: (context) => MyDashboard(
                    accessToken: jsonResponse[
                        "accessToken"],
                  )), // MyDashboard // MaterialPageRoute
        );
      }
    } else {
      setState(() {
        isLoading = false;
      });
      ScaffoldMessenger.of(context)
          .showSnackBar(SnackBar(
        content: Text(jsonResponse["error"]),
      )); // SnackBar
    }
  },
  style: ElevatedButton.styleFrom(
    backgroundColor: Colors.black,
    textStyle: const TextStyle(
        fontFamily: "JetBrainsMono",
        fontSize: 12), // TextStyle
  ),
  child: const Text("LOG IN"),
), // ElevatedButton
// Add the Sign up text and button
Row(
  mainAxisAlignment: MainAxisAlignment.center,
  children: [
    const Text(
      "Don't have an account?",
      style: TextStyle(
          color: Colors.white,
          fontSize: 12,
          fontFamily: "JetBrainsMono"), // TextStyle
    ), // Text
    TextButton(
        onPressed: () {
          Navigator.of(context).pushReplacement(
            MaterialPageRoute(
                builder: (context) =>
                    const Mysignup()), // MaterialPageRoute
          );
        },
```


Figure 7. Login (Code 3)

Figure 8. Login (Code 4)

This code represents the login screen. It contains a page with email and password text fields for the user to enter their login details. The page also includes validation for the email and password fields, which displays the message accordingly. The user can click on the visibility icon next to the password field to toggle the visibility of the password. The screen has a login button which makes an API call to the backend to authenticate the user. If the user is authenticated successfully, they get navigated to the MyDashboard screen. Otherwise, an error message is displayed. The page also has a 'Sign up' text button which navigates to the MySignup screen.

4.1.1.2. signup_email.dart

Figure 9. Sign Up Email (Output)

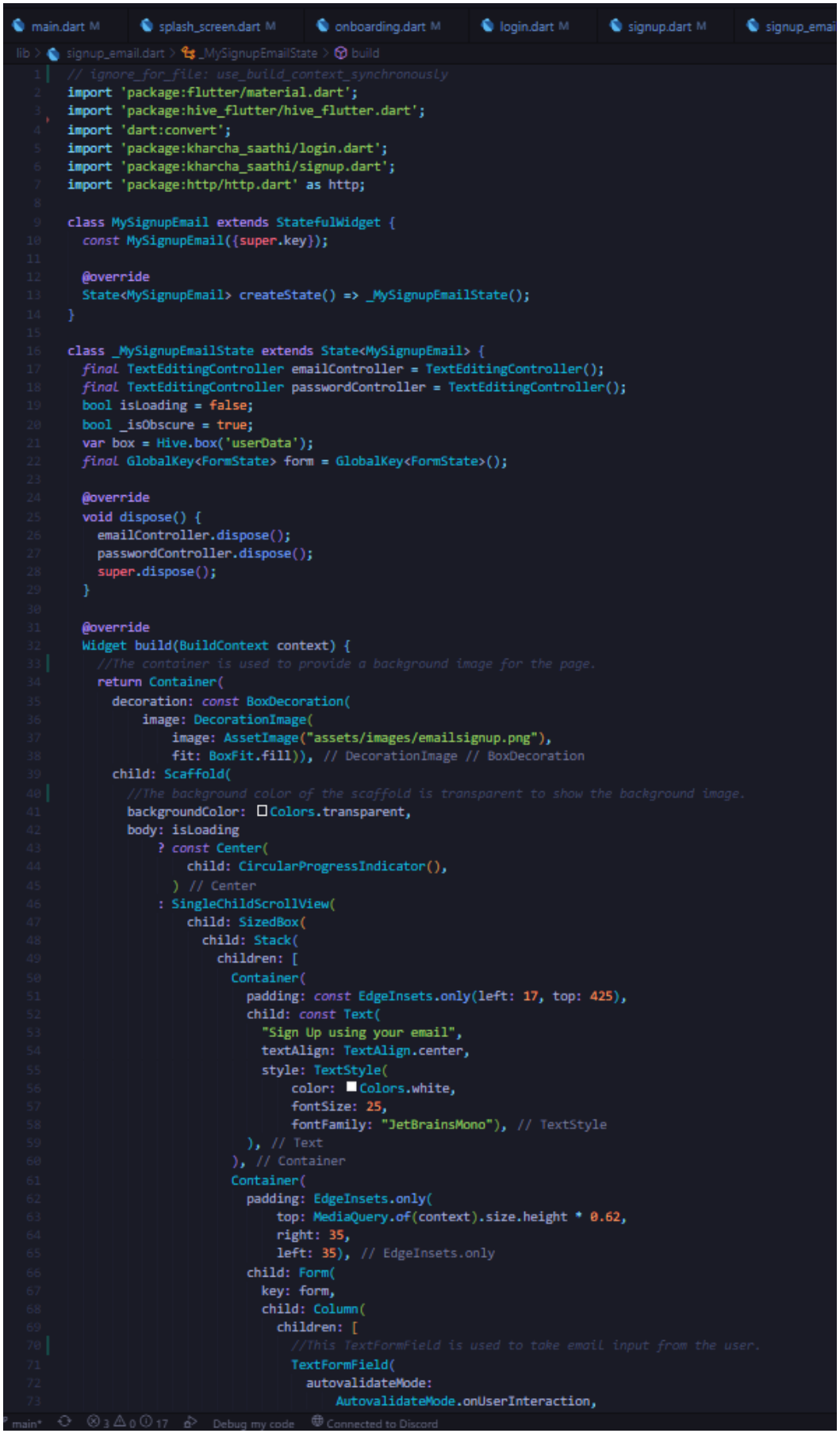


Figure 10. Sign Up Email (Code 1)

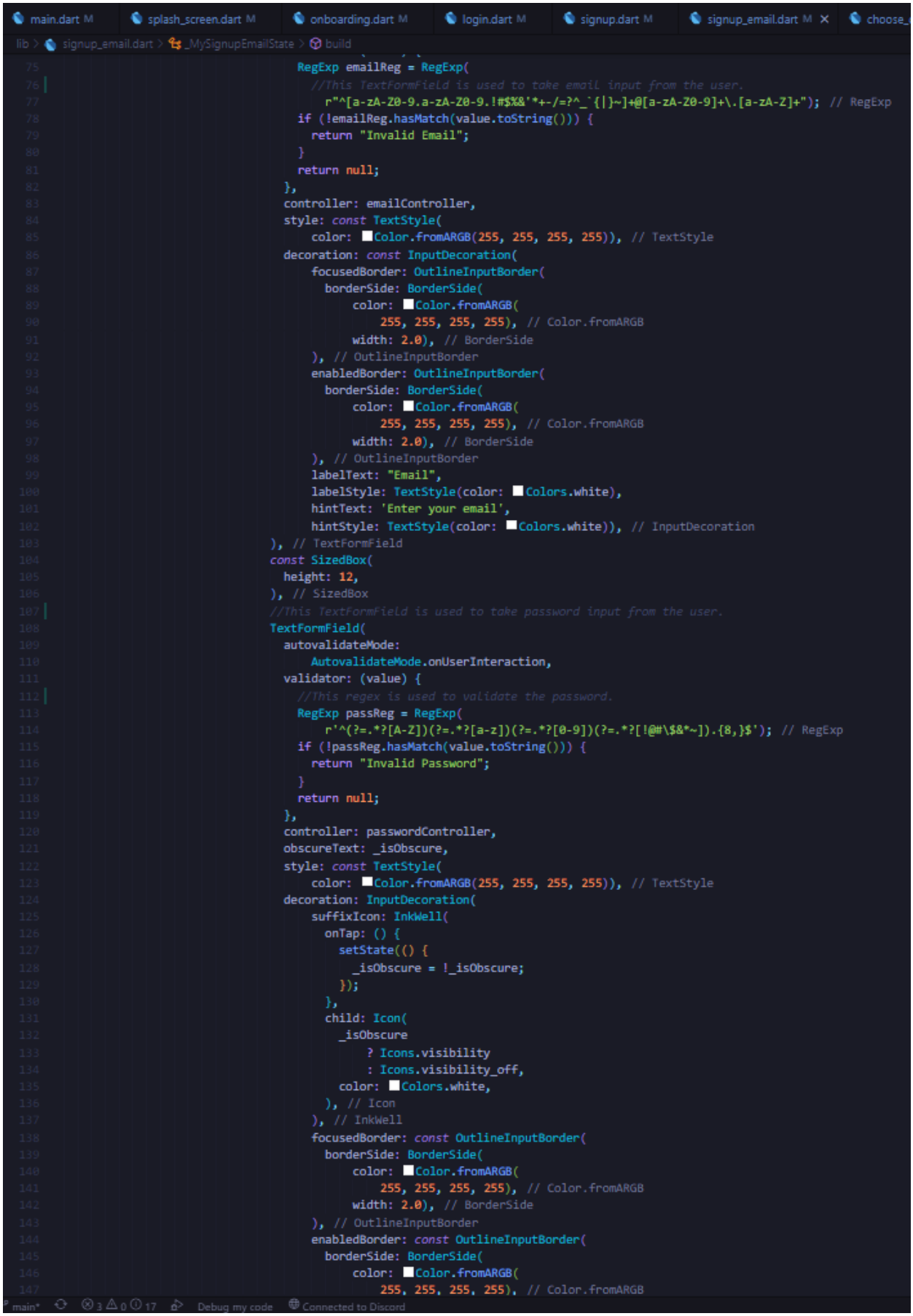


Figure 11. Sign Up Email (Code 2)

Figure 12. Sign Up Email (Code 3)

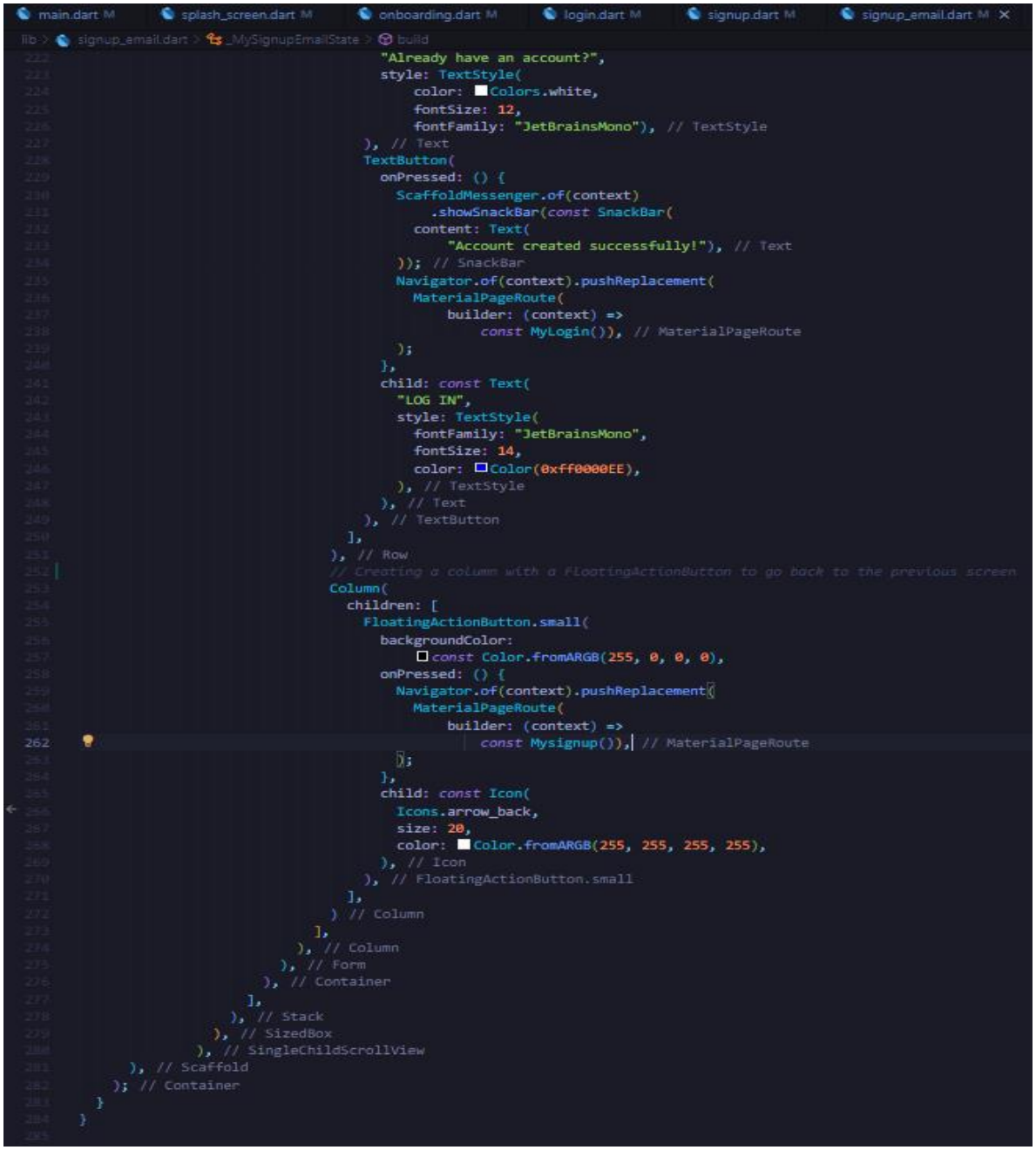


Figure 13. Sign Up Email (Code 4)

This code represents the screen for signing up using email. It contains a page with email and password fields and a sign-up button. The user can enter their email and password, and upon successful validation, they can sign up and create a new account. The form is validated according to the users input and display the message accordingly. The user is redirected to the login screen upon successful sign up. The screen also contains a back button to navigate to the previous screen, if they choose to use a different sign-up method or go back to login if they already have an account and wish to login instead.

4.1.1.3. income.dart

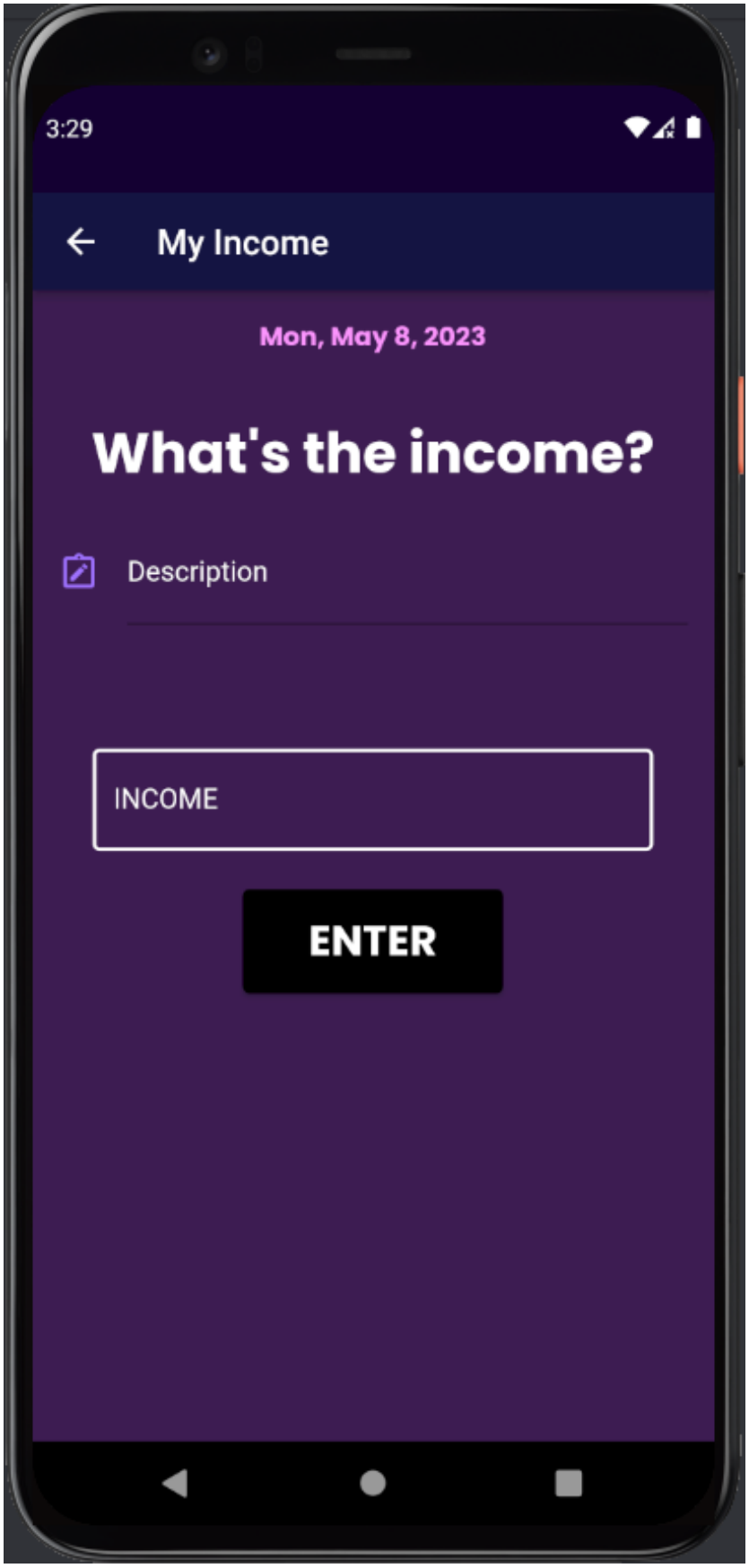


Figure 14. Income (Output)

Figure 15. Input (Code 1)

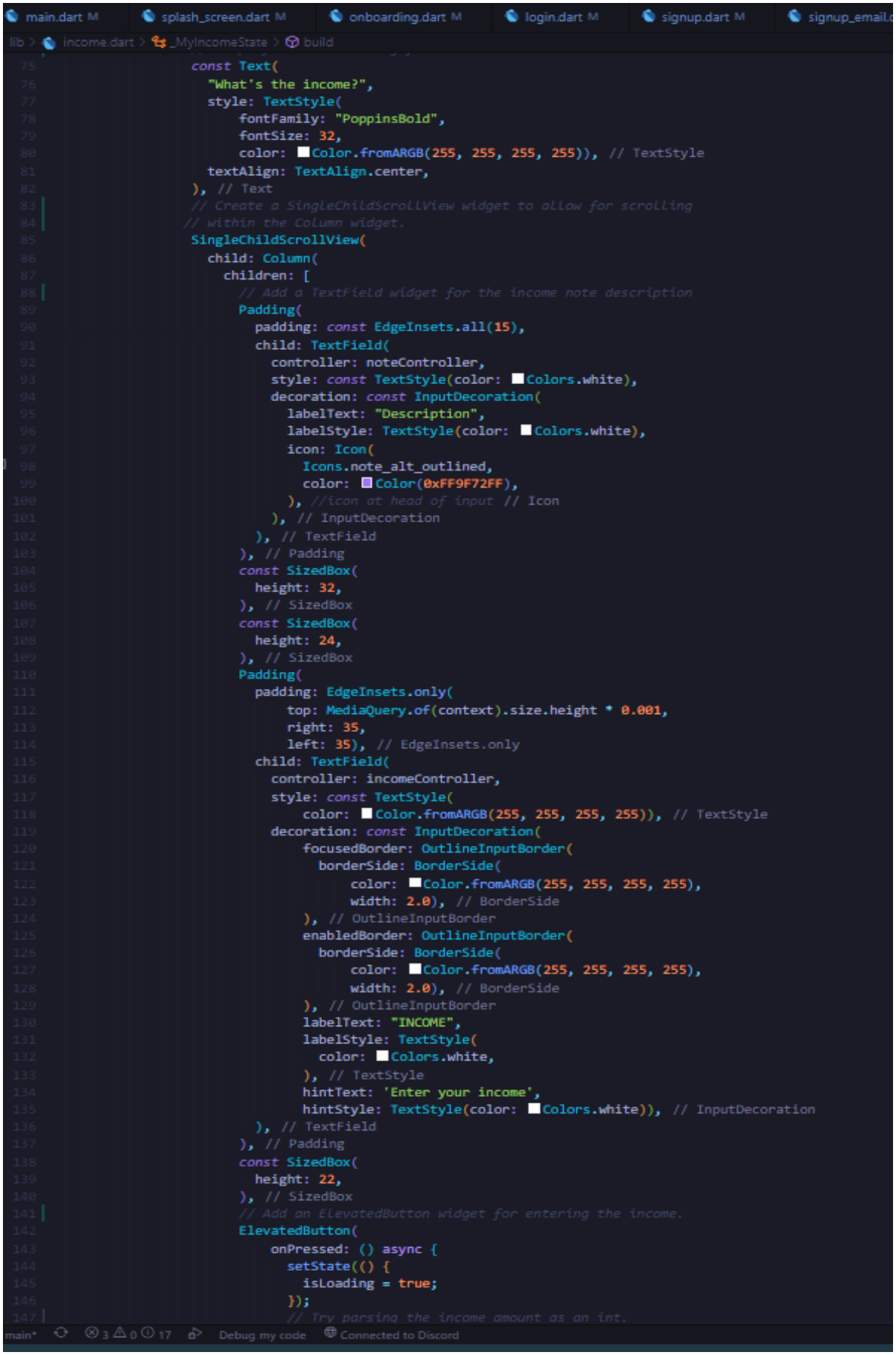


Figure 16. Input (Code 2)

Figure 17. Input (Code 3)

This code represents the income page of the app. The user can input income data such as the description and amount, and the code sends a PATCH request to the server to save the data. It displays a loading indicator while waiting for the response and a snackbar with the appropriate message afterwards.

It uses Hive to get the access token and dispose of text editing controllers properly. It also uses the intl package to format the date displayed at the top of the page, http package to make the PATCH request to the server, and json package to encode and decode the request and response bodies.

4.1.1.4. expense.dart

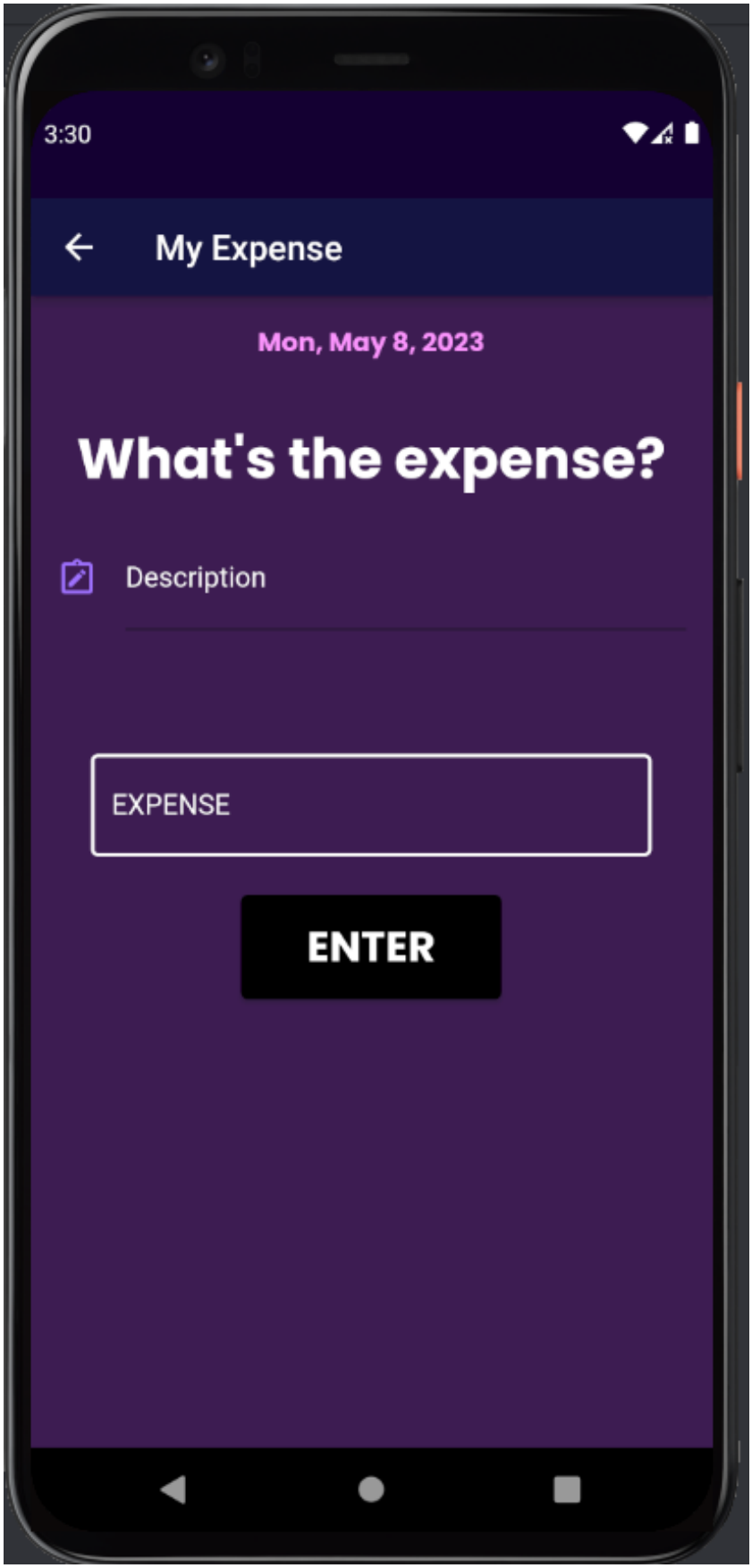


Figure 18. Expense (Output)

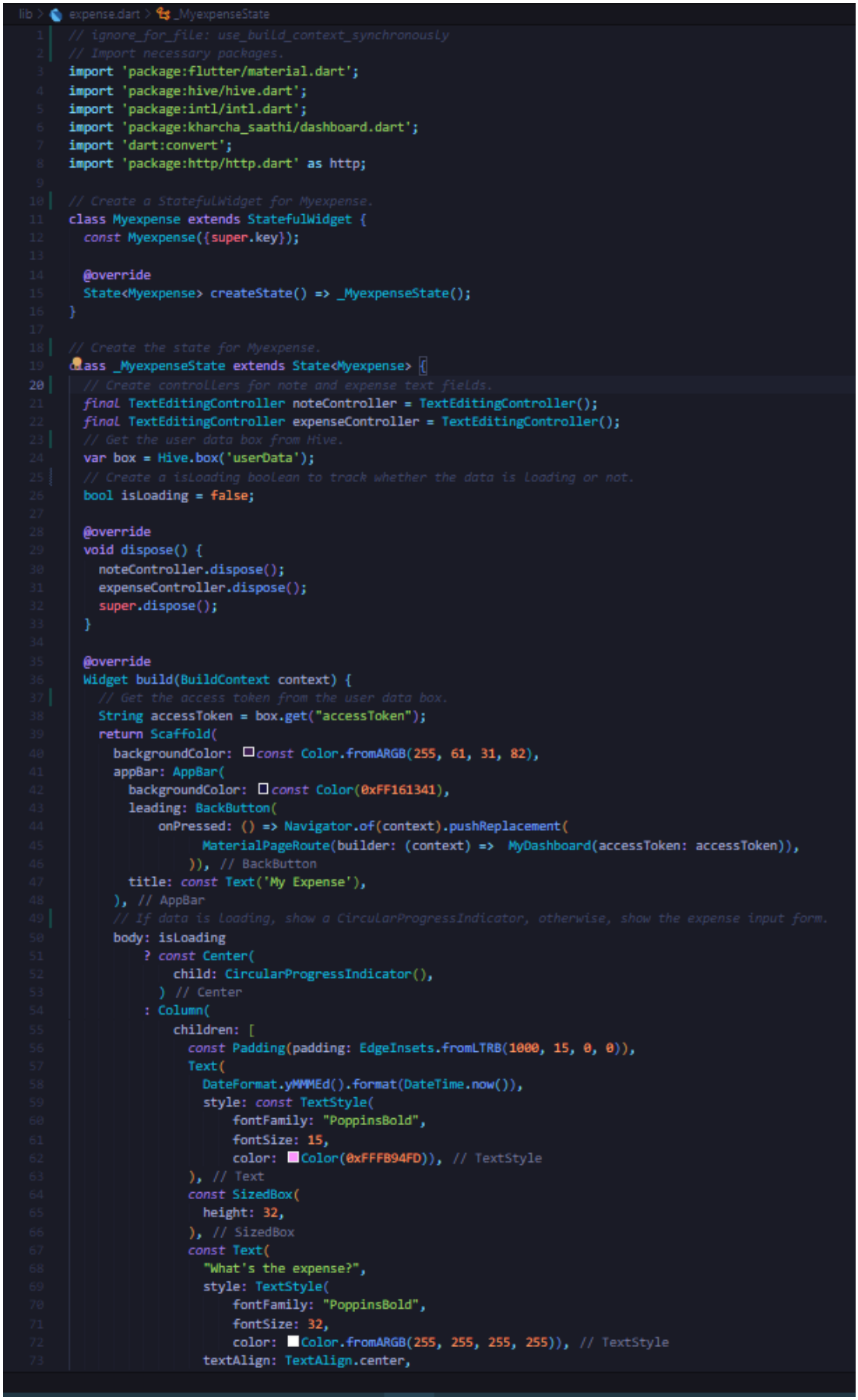


Figure 19. Expense (Code 1)

```
Column(
  children: [
    Padding(
      padding: const EdgeInsets.all(15),
      // Create a text input field for the expense note.
      child: TextField(
        controller: noteController,
        style: const TextStyle(color: Colors.white),
        decoration: const InputDecoration(
          labelText: "Description",
          labelStyle: TextStyle(color: Colors.white),
          icon: Icon(
            Icons.note_alt_outlined,
            color: Color(0xFF9F72FF),
          ), //icon at head of input // Icon
        ), // InputDecoration
      ), // TextField
    ), // Padding
    const SizedBox(
      height: 32,
    ), // SizedBox
    const SizedBox(
      height: 24,
    ), // SizedBox
    // Create a SingleChildScrollView to ensure the text input field is scrollable.
    SingleChildScrollView(
      child: Padding(
        padding: EdgeInsets.only(
            top: MediaQuery.of(context).size.height * 0.001,
            right: 35,
            left: 35), // EdgeInsets.only
        // Create a text input field for the expense amount.
        child: TextField(
          controller: expenseController,
          // Create input decoration for the text input field.
          style: const TextStyle(
              color: Color.fromARGB(255, 255, 255, 255)), // TextStyle
          decoration: const InputDecoration(
              focusedBorder: OutlineInputBorder(
                borderSide: BorderSide(
                    color: Color.fromARGB(255, 255, 255, 255),
                    width: 2.0), // BorderSide
              ), // OutlineInputBorder
              enabledBorder: OutlineInputBorder(
                borderSide: BorderSide(
                    color: Color.fromARGB(255, 255, 255, 255),
                    width: 2.0), // BorderSide
              ), // OutlineInputBorder
              labelText: "EXPENSE",
              labelStyle: TextStyle(
                color: Colors.white,
              ), // TextStyle
              hintText: 'Enter your expense',
              hintStyle: TextStyle(color: Colors.white)), // InputDecoration
        ), // TextField
      ), // Padding
    ), // SingleChildScrollView
    const SizedBox(
      height: 22,
    ), // SizedBox
    // Create an ElevatedButton for submitting the expense.
    ElevatedButton(
        onPressed: () async {
          setState(() {
            isLoading = true;
          });

          // Parse the expense amount from the text input field.
          int? amount = int.tryParse(expenseController.text);
          if (amount == null) {
            ScaffoldMessenger.of(context)
                .showSnackBar(const SnackBar(
              content: Text("Invalid balance value."),
```

Figure 20. Expense (Code 2)

```
                        )); // SnackBar
                        setState(() {
                          isLoading = false;
                        });
                        return;
                      }
                      // Send a PATCH request to the backend API to add the expense.
                      http.Response response = await http.patch(
                          Uri.parse(
                              "https://kharchasathi-backend.onrender.com/users/expense"),
                          headers: {
                            "Authorization":
                                "Bearer $accessToken"
                          },
                          body: json.encode({
                            "description": noteController.text,
                            "amount": amount,
                          }));

                      var jsonResponse = json.decode(response.body);
                      if (response.statusCode == 200) {
                        // Show a success message and navigate back to the dashboard.
                        ScaffoldMessenger.of(context).showSnackBar(SnackBar(
                          content: Text(jsonResponse["message"]),
                        )); // SnackBar
                        Navigator.of(context).pushReplacement(
                          MaterialPageRoute(
                              builder: (context) =>  MyDashboard(accessToken: accessToken)), // MaterialPageRoute
                        );
                      } else {
                        // Show an error message if the request fails.
                        setState(() {
                          isLoading = false;
                        });
                        ScaffoldMessenger.of(context).showSnackBar(SnackBar(
                          content: Text(jsonResponse["error"]),
                        )); // SnackBar
                      }
                    },
                    style: ElevatedButton.styleFrom(
                        backgroundColor: Colors.black,
                        fixedSize: const Size(150, 60)),
                    child: const Text(
                      "ENTER",
                      style: TextStyle(
                          fontFamily: "PoppinsBold",
                          color: Colors.white,
                          fontSize: 24), // TextStyle
                    )) // Text // ElevatedButton
                ],
              ), // Column
            ],
          ), // Column
        ); // Scaffold
      }
    }
```

Figure 21. Expense (Code 3)

This code represents the expense page of the app. The user can input income data such as the description and amount, and the code sends a PATCH request to the server to save the data. It displays a loading indicator while waiting for the response and a snackbar with the appropriate message afterwards.

It uses Hive to get the access token and dispose of text editing controllers properly. It also uses the intl package to format the date displayed at the top of the page, http package to make the PATCH request to the server, and json package to encode and decode the request and response bodies.

4.1.1.5. records.dart

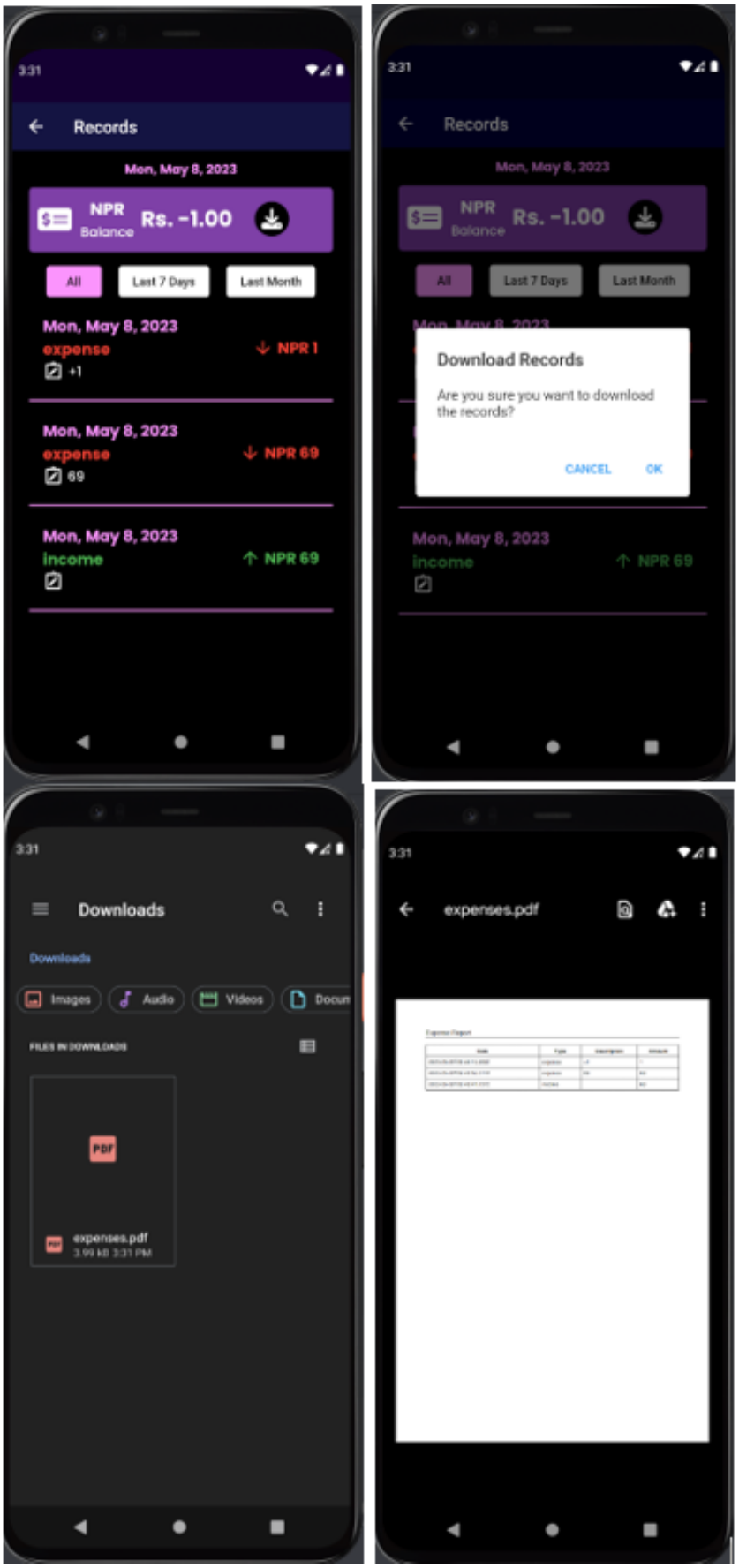


Figure 22. Records (Output)

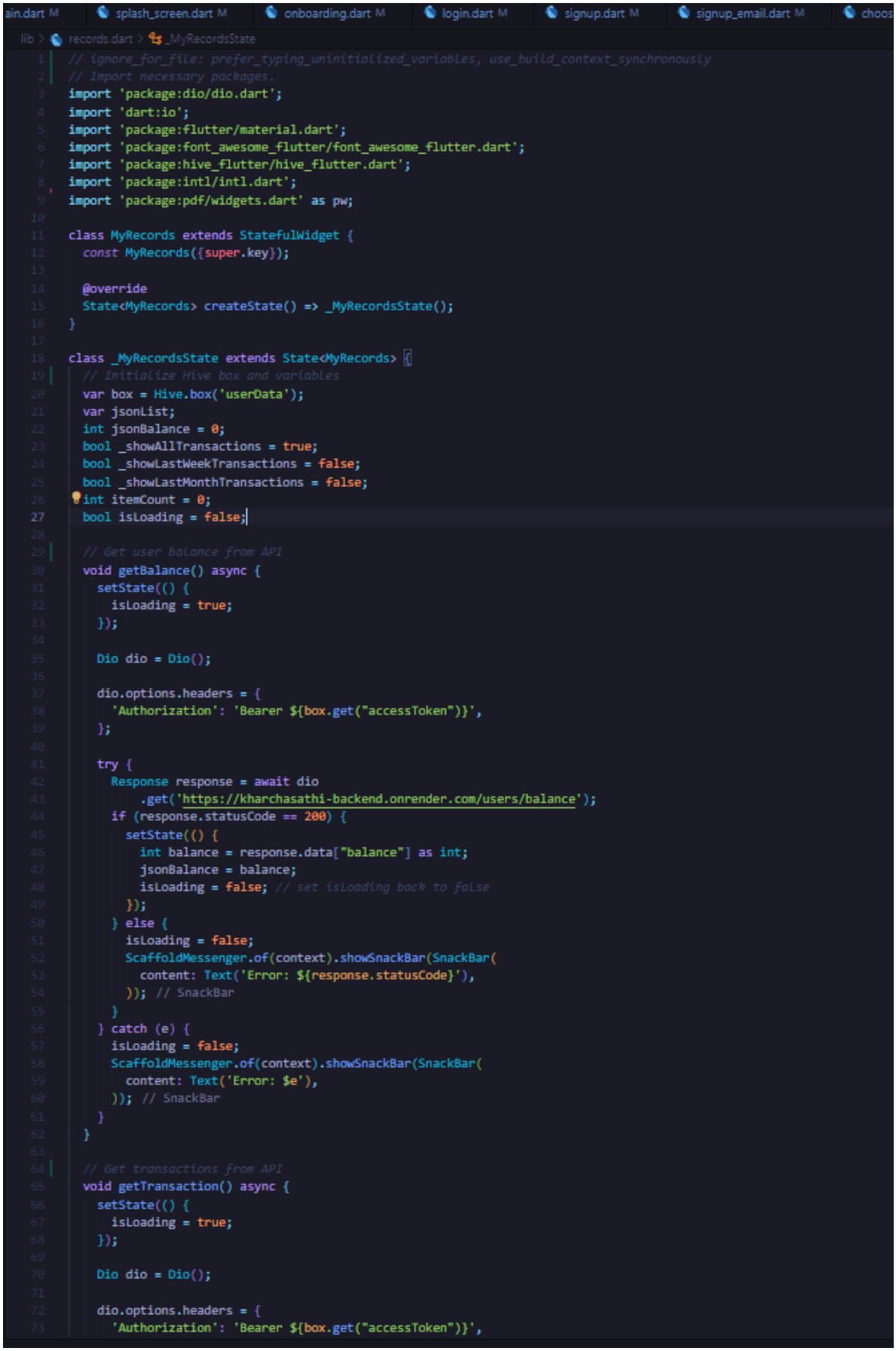


Figure 23. Records (Code 1)

```
    try {
      Response response = await dio
          .get('https://kharchasathi-backend.onrender.com/transaction');
      if (response.statusCode == 200) {
        setState(() {
          jsonList = response.data["transactions"] as List;
          isLoading = false;
        });
      } else {
        isLoading = false;
        ScaffoldMessenger.of(context).showSnackBar(SnackBar(
          content: Text('Error: ${response.statusCode}'),
        )); // SnackBar
      }
    } catch (e) {
      isLoading = false;
      ScaffoldMessenger.of(context).showSnackBar(SnackBar(
        content: Text('Error: $e'),
      )); // SnackBar
    }
  }

  @override
  void initState() {
    getBalance();
    getTransaction();
    super.initState();
  }

  @override
  Widget build(BuildContext context) {
    // Determine the number of transactions to display based on filters
    if (_showLastWeekTransactions) {
      // Filter transactions for the last 7 days
      DateTime now = DateTime.now();
      DateTime lastWeek = now.subtract(const Duration(days: 7));
      itemCount = jsonList
          .where((transaction) =>
              DateTime.parse(transaction['date']).isAfter(lastWeek))
          .length;
    } else if (_showLastMonthTransactions) {
      // Filter transactions for the last month
      DateTime now = DateTime.now();
      DateTime lastMonth = DateTime(now.year, now.month - 1, now.day);
      itemCount = jsonList
          .where((transaction) =>
              DateTime.parse(transaction['date']).isAfter(lastMonth))
          .length;
    } else {
      itemCount = jsonList?.length ?? 0;
    }

    return Scaffold(
      // Set the background color to black
      backgroundColor: Colors.black,
      appBar: AppBar(
        // Set the app bar background color to a dark blue
        backgroundColor: const Color(0xFF161341),
        title: const Text('Records'),
      ), // AppBar
      body: SafeArea(
        child: Padding(
          padding: const EdgeInsets.symmetric(horizontal: 20, vertical: 10),
          child: Column(
            children: [
              // Show today's date in a specific format
              Text(
                DateFormat.yMMMEd().format(DateTime.now()),
                style: const TextStyle(
                  fontFamily: "PoppinsBold",
                  fontSize: 15,
                  color: Color(0xFFFB94FD),
                ), // TextStyle
```

Figure 24. Records (Code 2)

```
const SizedBox(
  height: 10,
), // SizedBox
Container(
  height: 75,
  width: 400,
  decoration: BoxDecoration(
    borderRadius: BorderRadius.circular(5),
    color: const Color(0xff7e3fa7),
  ), // BoxDecoration
  child: Padding(
    padding:
        const EdgeInsets.symmetric(horizontal: 10, vertical: 10),
    child: Column(
      children: [
        Row(
          children: [
            // Show a money icon
            const Icon(
              FontAwesomeIcons.moneyCheckDollar,
              color: Colors.white,
              size: 35,
            ), // Icon
            const SizedBox(
              width: 15,
            ), // SizedBox
            Column(
              children: const [
                // Show "NPR" and "Balance" text
                Text(
                  "NPR",
                  style: TextStyle(
                      fontFamily: "PoppinsBold",
                      fontSize: 20,
                      color: Colors.white), // TextStyle
                ), // Text
                Text(
                  "Balance",
                  style: TextStyle(
                      fontFamily: "PoppinsMedium",
                      fontSize: 15,
                      color: Colors.white), // TextStyle
                ), // Text
              ],
            ), // Column
            Row(
              children: [
                const SizedBox(
                  width: 8,
                ), // SizedBox
                isLoading
                    ? const Center(
                        child: CircularProgressIndicator(),
                      ) // Center
                    : Text(
                        // Show the current balance
                        'Rs. ${jsonBalance.toStringAsFixed(2)}',
                        style: const TextStyle(
                            fontFamily: "PoppinsBold",
                            fontSize: 25,
                            color: Colors.white), // TextStyle
                      ), // Text
                const SizedBox(
                  width: 25,
                ), // SizedBox
                InkWell(
                  onTap: () {
                    // Show a dialog to download records
                    showDialog(
                      context: context,
                      builder: (context) => AlertDialog(
                        title: const Text('Download Records'),
                        content: const Text(
```

Figure 25. Records (Code 3)

Figure 26. Records (Code 4)

Figure 27. Records (Code 5)

```
                                    style: TextStyle(
                                      fontFamily: "PoppinsBold",
                                      fontSize: 18,
                                      color: color,
                                    ), // TextStyle
                                  ), // Text
                                  Row(
                                    children: [
                                      const Icon(
                                        Icons.note_alt_outlined,
                                        color: Colors.white,
                                      ), // Icon
                                      const SizedBox(width: 5),
                                      Text(
                                        jsonList[index]
                                                ["description"] ??
                                            "",
                                        style: const TextStyle(
                                          fontFamily: "PoppinsMedium",
                                          fontSize: 15,
                                          color: Colors.white,
                                        ), // TextStyle
                                      ), // Text
                                    ],
                                  ), // Row
                                ],
                              ), // Column
                            ), // Expanded
                            Row(
                              mainAxisSize: MainAxisSize.min,
                              children: [
                                Icon(
                                  isExpense
                                      ? Icons.arrow_downward
                                      : Icons.arrow_upward,
                                  color: isExpense
                                      ? Colors.red
                                      : Colors.green,
                                ), // Icon
                                const SizedBox(width: 5),
                                Text(
                                  "NPR ${jsonList[index]["amount"] ?? "0.00"}",
                                  style: TextStyle(
                                    fontFamily: "PoppinsBold",
                                    fontSize: 18,
                                    color: isExpense
                                        ? Colors.red
                                        : Colors.green,
                                  ), // TextStyle
                                ), // Text
                              ],
                            ), // Row
                          ],
                        ), // Row
                      ), // ListTile
                      const SizedBox(height: 10),
                      const Divider(
                        thickness: 2,
                        color: Color(0xfffB94FD),
                      ), // Divider
                      const SizedBox(height: 10),
                    ],
                  ); // Column
                },
              ), // ListView.builder
            ), // Expanded
          ],
        ), // Column
      ), // Padding
    ), // SafeArea
  ); // Scaffold
 }
}
```

Figure 28. Records (Code 6)

```dart
Future<void> generatePDFReport(List<dynamic> jsonList, String filePath) async {
  // Creates a new Document object from the pdf library.
  final pdf = pw.Document();
  // Adds a new page to the document using a MultiPage widget from the pdf library.
  pdf.addPage(pw.MultiPage(
      build: (pw.Context context) => <pw.Widget>[
            // Adds a header to the page.
            pw.Header(level: 0, child: pw.Text('Expense Report')),
            // Adds a table to the page using the data from jsonList.
            pw.Table.fromTextArray(context: context, data: <List<String>>[
              <String>['Date', 'Type', 'Description', 'Amount'],
              ...jsonList.map((record) {
                return <String>[
                  record['date'],
                  record['type'],
                  record['description'],
                  record['amount'].toString(),
                ]; // <String>[]
              })
            ]), // <List<String>>[] // pw.Table.fromTextArray
          ])); // <pw.Widget>[] // pw.MultiPage
  // Creates a new File object using the provided filePath.
  final file = File(filePath);
  // Saves the pdf document as bytes to the file.
  await file.writeAsBytes(await pdf.save());
  // Saves the pdf document.
  pdf.save().then((value) {});
}
```

Figure 29. Records (Code 7)

This code represents the records page of the app.

It allows the user to view their balance and transaction history in a listview.

It sends GET requests to the server to retrieve the data and displays a loading indicator while waiting for the response.

It also allows the user to filter transactions by all, last 7 days, or last month.

It uses Hive to get the access token and the intl package to format the date displayed at the top of the page.

It also uses the Dio package to make the GET requests to the server and the pdf package to generate a PDF report of the user's transaction history.

4.1.1.6. change_password.dart

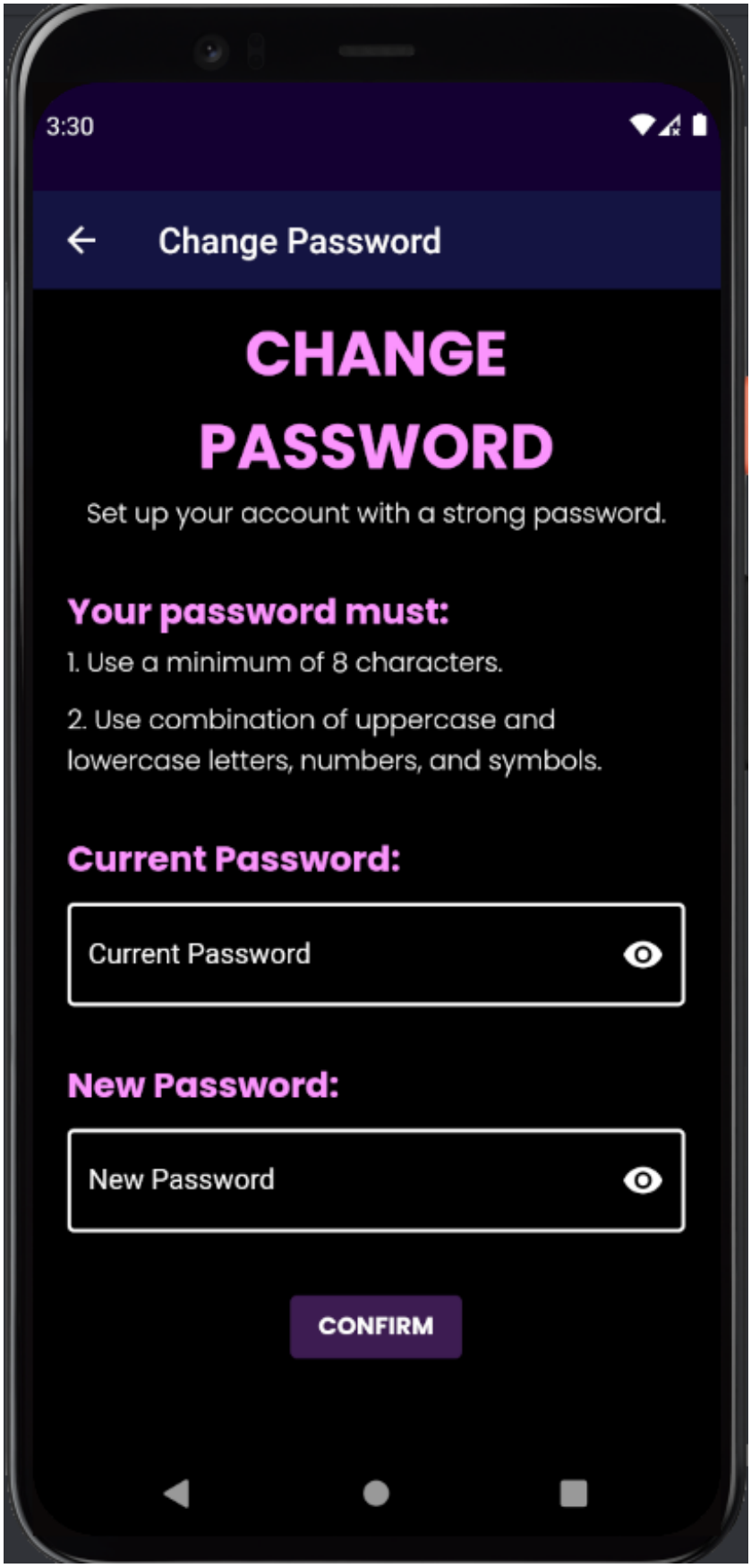


Figure 30. Change Password (Output)

lib > change_password.dart

```dart
// Importing necessary packages
import 'package:flutter/material.dart';
import 'package:hive_flutter/hive_flutter.dart';
import 'package:kharcha_saathi/settings.dart';
import 'package:http/http.dart' as http;
import 'dart:convert';

//StatefulWidget for changing password
class MyChangePassword extends StatefulWidget {
  const MyChangePassword({super.key});

  @override
  State<MyChangePassword> createState() => _MyChangePasswordState();
}

class _MyChangePasswordState extends State<MyChangePassword> {
  //creating two text controllers to retrieve user's input
  final TextEditingController opassController = TextEditingController();
  final TextEditingController npassController = TextEditingController();
  //boolean variable to indicate loading state
  bool isLoading = false;
  //boolean variable to toggle password visibility
  bool _isObscure = true;
  final GlobalKey<FormState> form = GlobalKey<FormState>();
  var box = Hive.box('userData');

  @override
  void dispose() {
    //disposing the text controllers
    opassController.dispose();
    npassController.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    //getting the user's access token from the database
    String accessToken = box.get("accessToken");
    //creating a scaffold widget with a black background and appbar
    return Scaffold(
      backgroundColor: Colors.black,
      appBar: AppBar(
        backgroundColor: const Color(0xFF161341),
        //creating back button to navigate back to settings page
        leading: BackButton(
            onPressed: () => Navigator.of(context).pushReplacement(
                  MaterialPageRoute(builder: (context) => const MySettings()),
                )),
        title: const Text('Change Password'),
```

Figure 31. Change Password (Code 1)

```
50          ), // AppBar
51          //Loading state management using ternary operator
52          body: isLoading
53              ? const Center(
54                  child: CircularProgressIndicator(),
55                ) // Center
56              : SingleChildScrollView(
57                  child: SafeArea(
58                    child: Padding(
59                      padding:
60                          const EdgeInsets.symmetric(horizontal: 20, vertical: 10),
61                      child: Column(
62                        crossAxisAlignment: CrossAxisAlignment.center,
63                        children: [
64                          const Text(
65                            "CHANGE\nPASSWORD",
66                            textAlign: TextAlign.center,
67                            style: TextStyle(
68                              fontFamily: "PoppinsBold",
69                              fontSize: 35,
70                              color: Color(0xffFB94FD),
71                            ), // TextStyle
72                          ), // Text
73                          const Text(
74                            "Set up your account with a strong password.",
75                            textAlign: TextAlign.center,
76                            style: TextStyle(
77                              fontFamily: "PoppinsLight",
78                              fontSize: 15,
79                              color: Colors.white,
80                            ), // TextStyle
81                          ), // Text
82                          const SizedBox(
83                            height: 30,
84                          ), // SizedBox
85                          // Row containing a title for the password requirements
86                          Row(
87                            children: const [
88                              Text(
89                                "Your password must:",
90                                style: TextStyle(
91                                  fontFamily: "PoppinsBold",
92                                  fontSize: 20,
93                                  color: Color(0xffFB94FD),
94                                ), // TextStyle
95                              ), // Text
96                            ],
97                          ), // Row
98                          // Column containing the password requirements
```

Figure 32. Change Password (Code 2)

```
Column(
  crossAxisAlignment: CrossAxisAlignment.start,
  children: [
    const SizedBox(
      height: 2,
    ), // SizedBox
    const Text(
      "1. Use a minimum of 8 characters.",
      style: TextStyle(
        fontFamily: "PoppinsLight",
        fontSize: 15,
        color: Colors.white,
      ), // TextStyle
    ), // Text
    const SizedBox(
      height: 10,
    ), // SizedBox
    const Text(
      "2. Use combination of uppercase and lowercase letters, numbers, and symbols.",
      style: TextStyle(
        fontFamily: "PoppinsLight",
        fontSize: 15,
        color: Colors.white,
      ), // TextStyle
    ), // Text
    const SizedBox(
      height: 30,
    ), // SizedBox
    const Text(
      "Current Password:",
      style: TextStyle(
        fontFamily: "PoppinsBold",
        fontSize: 20,
        color: Color(0xffFB94FD),
      ), // TextStyle
    ), // Text
    const SizedBox(
      height: 10,
    ), // SizedBox
    // A TextFormField for the current password field
    TextFormField(
      autovalidateMode:
          AutovalidateMode.onUserInteraction,
      validator: (value) {
        // Regular expression to validate password
        RegExp passReg = RegExp(
            r'^(?=.*?[A-Z])(?=.*?[a-z])(?=.*?[0-9])(?=.*?[!@#\$&*~]).{8,}$'); // RegExp
        // Check if value matches password regex
        if (!passReg.hasMatch(value.toString())) {
```

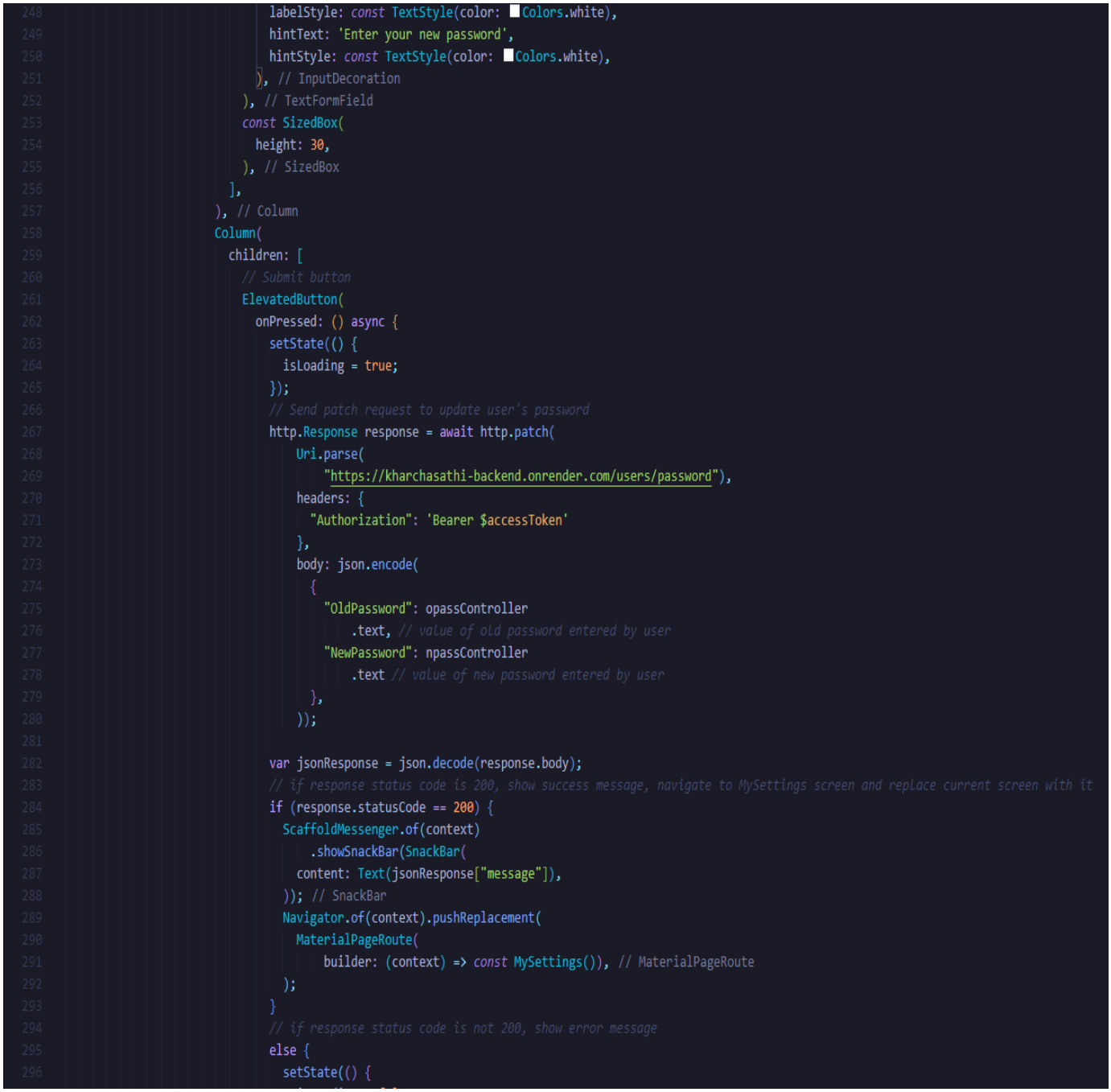
Figure 33. Change Password (Code 3)

```
148                     return "Invalid Password";
149                   }
150                   return null;
151                 },
152                 obscureText: _isObscure,
153                 controller: opassController,
154                 style: const TextStyle(
155                     color: Color.fromARGB(255, 255, 255, 255)), // TextStyle
156                 decoration: InputDecoration(
157                   suffixIcon: InkWell(
158                     onTap: () {
159                       setState(() {
160                         _isObscure = !_isObscure;
161                       });
162                     },
163                     child: Icon(
164                       _isObscure
165                           ? Icons.visibility
166                           : Icons.visibility_off,
167                       color: Colors.white,
168                     ), // Icon
169                   ), // InkWell
170                   focusedBorder: const OutlineInputBorder(
171                     borderSide: BorderSide(
172                       color: Color.fromARGB(255, 255, 255, 255),
173                       width: 2.0,
174                     ), // BorderSide
175                   ), // OutlineInputBorder
176                   enabledBorder: const OutlineInputBorder(
177                     borderSide: BorderSide(
178                       color: Color.fromARGB(255, 255, 255, 255),
179                       width: 2.0,
180                     ), // BorderSide
181                   ), // OutlineInputBorder
182                   labelText: "Current Password",
183                   labelStyle: const TextStyle(color: Colors.white),
184                   hintText: 'Enter your current password',
185                   hintStyle: const TextStyle(color: Colors.white),
186                 ), // InputDecoration
187               ), // TextFormField
188               const SizedBox(
189                 height: 30,
190               ), // SizedBox
191               const Text(
192                 "New Password:",
193                 style: TextStyle(
194                   fontFamily: "PoppinsBold",
195                   fontSize: 20,
196                   color: Color(0xffFB94FD),
```

Figure 34. Change Password (Code 4)

```
                ), // TextStyle
              ), // Text
              const SizedBox(
                height: 10,
              ), // SizedBox
              // Validate new password field
              TextFormField(
                autovalidateMode:
                    AutovalidateMode.onUserInteraction,
                validator: (value) {
                  RegExp passReg = RegExp(
                      r'^(?=.*?[A-Z])(?=.*?[a-z])(?=.*?[0-9])(?=.*?[!@#\$&*~]).{8,}$'); // RegExp
                  // Check if value matches password regex
                  if (!passReg.hasMatch(value.toString())) {
                    return "Invalid Password";
                  }
                  return null;
                },
                obscureText: _isObscure,
                controller: npassController,
                style: const TextStyle(
                    color: Color.fromARGB(255, 255, 255, 255)), // TextStyle
                decoration: InputDecoration(
                  suffixIcon: InkWell(
                    onTap: () {
                      setState(() {
                        _isObscure = !_isObscure;
                      });
                    },
                    child: Icon(
                      _isObscure
                          ? Icons.visibility
                          : Icons.visibility_off,
                      color: Colors.white,
                    ), // Icon
                  ), // InkWell
                  focusedBorder: const OutlineInputBorder(
                    borderSide: BorderSide(
                      color: Color.fromARGB(255, 255, 255, 255),
                      width: 2.0,
                    ), // BorderSide
                  ), // OutlineInputBorder
                  enabledBorder: const OutlineInputBorder(
                    borderSide: BorderSide(
                      color: Color.fromARGB(255, 255, 255, 255),
                      width: 2.0,
                    ), // BorderSide
                  ), // OutlineInputBorder
                  labelText: "New Password",
```

Figure 35. Change Password (Code 5)



```
                labelStyle: const TextStyle(color: Colors.white),
                hintText: 'Enter your new password',
                hintStyle: const TextStyle(color: Colors.white),
              ), // InputDecoration
            ), // TextFormField
            const SizedBox(
              height: 30,
            ), // SizedBox
          ],
        ), // Column
        Column(
          children: [
            // Submit button
            ElevatedButton(
              onPressed: () async {
                setState(() {
                  isLoading = true;
                });
                // Send patch request to update user's password
                http.Response response = await http.patch(
                    Uri.parse(
                        "https://kharchasathi-backend.onrender.com/users/password"),
                    headers: {
                      "Authorization": 'Bearer $accessToken'
                    },
                    body: json.encode(
                      {
                        "OldPassword": opassController
                            .text, // value of old password entered by user
                        "NewPassword": npassController
                            .text // value of new password entered by user
                      },
                    ));

                var jsonResponse = json.decode(response.body);
                // if response status code is 200, show success message, navigate to MySettings screen and replace current screen with it
                if (response.statusCode == 200) {
                  ScaffoldMessenger.of(context)
                      .showSnackBar(SnackBar(
                    content: Text(jsonResponse["message"]),
                  )); // SnackBar
                  Navigator.of(context).pushReplacement(
                    MaterialPageRoute(
                        builder: (context) => const MySettings()), // MaterialPageRoute
                  );
                }
                // if response status code is not 200, show error message
                else {
                  setState(() {
```

Figure 36. Change Password (Code 6)

Figure 37. Change Password (Code 7)

This is a code for change password page. The necessary packages are imported at the beginning. User input is retrieved using two text fields, each has a visibility icon to toggle password visibility. Password validation is done using a regular expression. When the user submits the new password, the widget sends a POST request with the new password and the user's access token to the server which gets validated before showing a message in the snack bar. If the old password is matched with the database, the password will be changed otherwise an error message will be displayed.

4.1.2. Commits

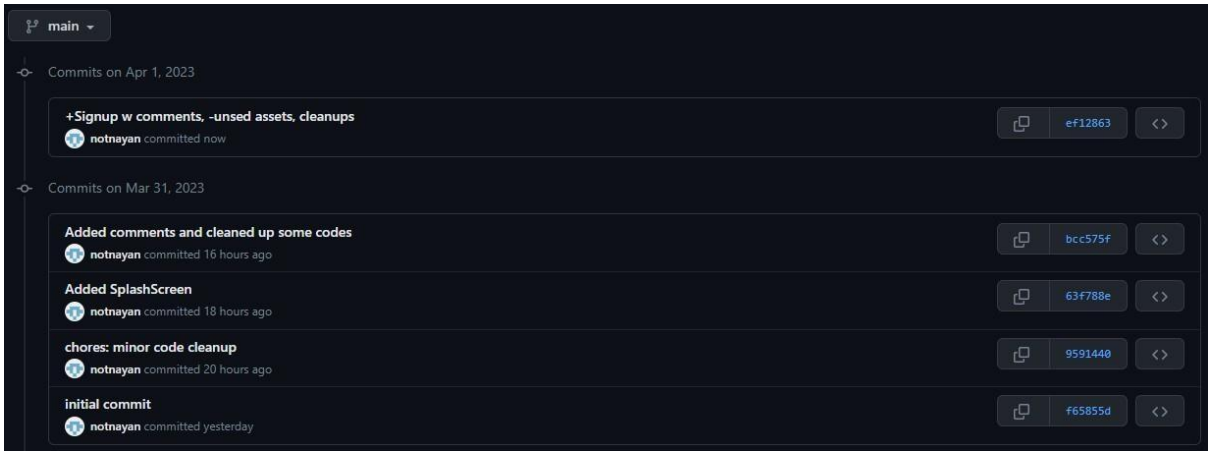


Figure 38. GitHub Commit (Sprint 1)

Daily commits of the codes in GitHub of development for Sprint 1.

For Sprint 2:

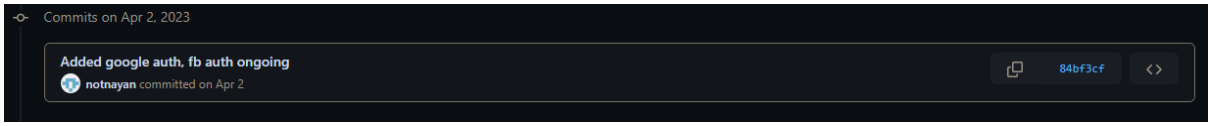


Figure 39. GitHub Commit (April 2)

On April 2nd completed adding google authentication and started working on Facebook authentication.

Figure 40. GitHub Commit (April 5)

On April 5th completed Facebook authentication.

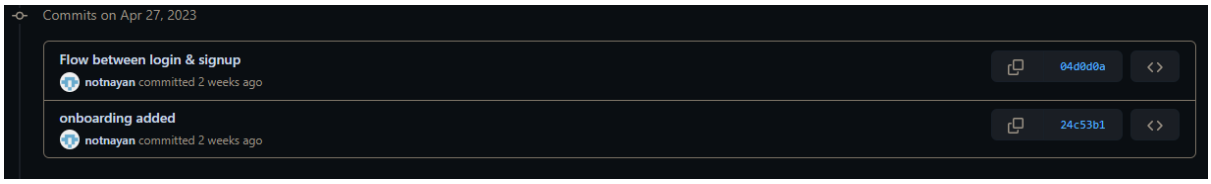


Figure 41. GitHub Commit (April 27)

On April 27th added navigation between login and signup page and finished three onboarding screens.

The huge gap between the last and this commit was due to examination week and HCI's submission in between.

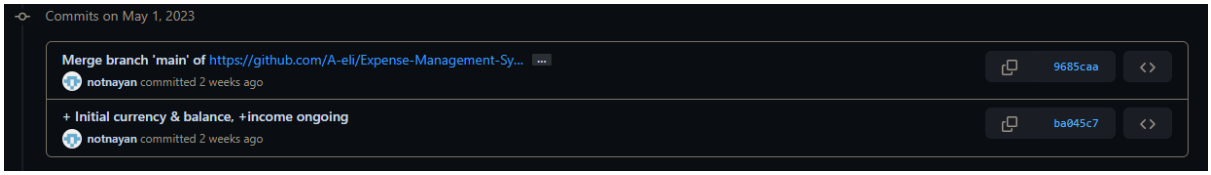


Figure 42. GitHub Commit (May 1)

On May 1st added initial currency and balance set up page for new users and started working on income page.

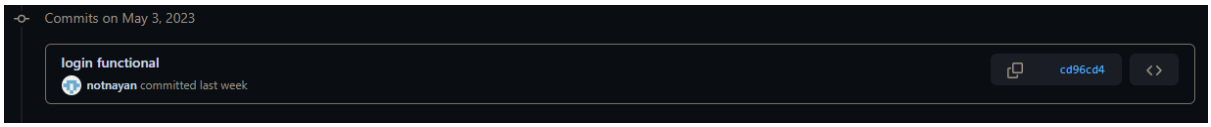


Figure 43. GitHub Commit (May 3)

On May 3rd connected frontend and backend and made the login page functional.

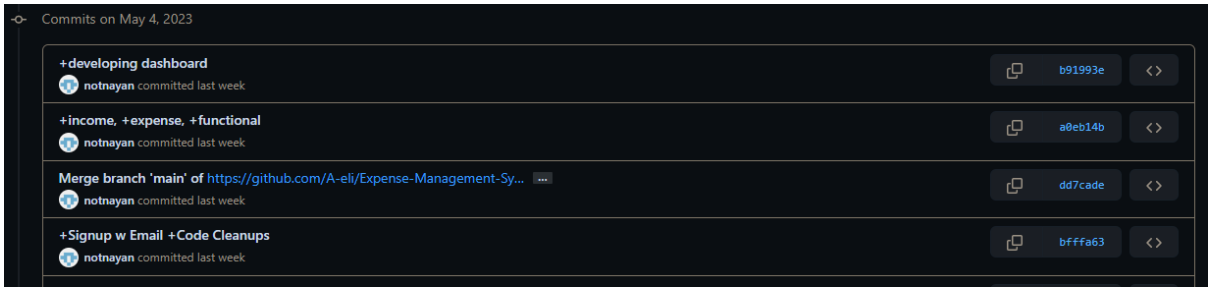


Figure 44. GitHub Commit (May 4)

On May 4th started working on dashboard. Finished income and expense page and connected the frontend and backend. Also, added a new method to sign up with email and password alongside cleaned up some codes.

Figure 45. GitHub Commit (May 5)

On May 5th added settings page alongside with help section in it, pop up menu in the dashboard and cleaned up some codes.

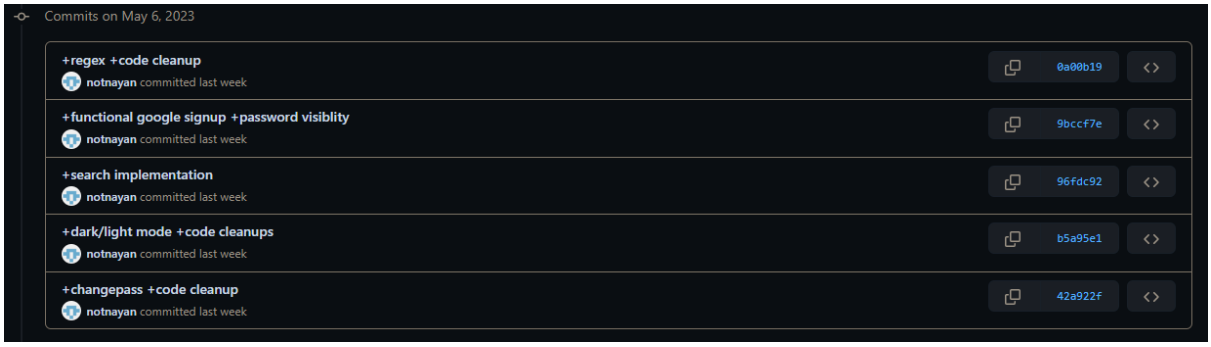


Figure 46. GitHub Commit (May 6)

On 6th May added change password page alongside with regex for login, signup and change password. Also connected the frontend and backend for google signup and included a visibility icon to toggle password visibility. Also added search button for records page alongside with light and dark mode and some code clean-ups.

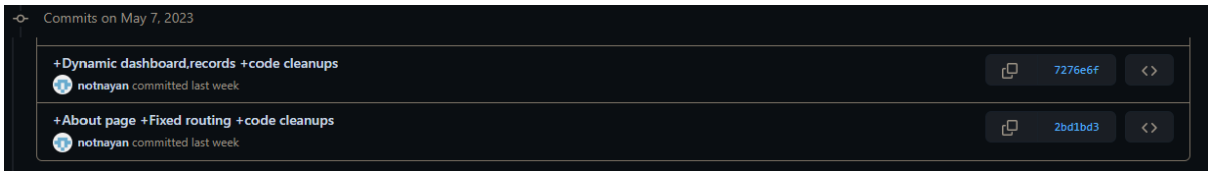


Figure 47. GitHub Commit (May 7)

On 7th May connected the dashboard and records page with backend so that the data changes dynamically and also added about page alongside with code clean-ups and better navigation between pages.

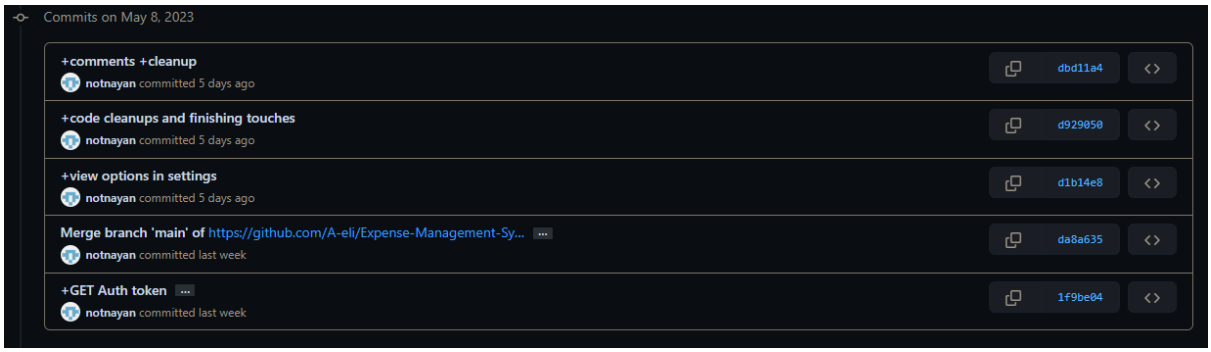


Figure 48. GitHub Commit (May 8)

On 8th May started adding comments to the code and cleaned up codes. Passed Auth token as a parameter for better UX and state management.

Figure 49. GitHub Commit (May 9)

On 9th May finished adding comments and gave the code some final touches.

4.1.3. Flow Chart of the system

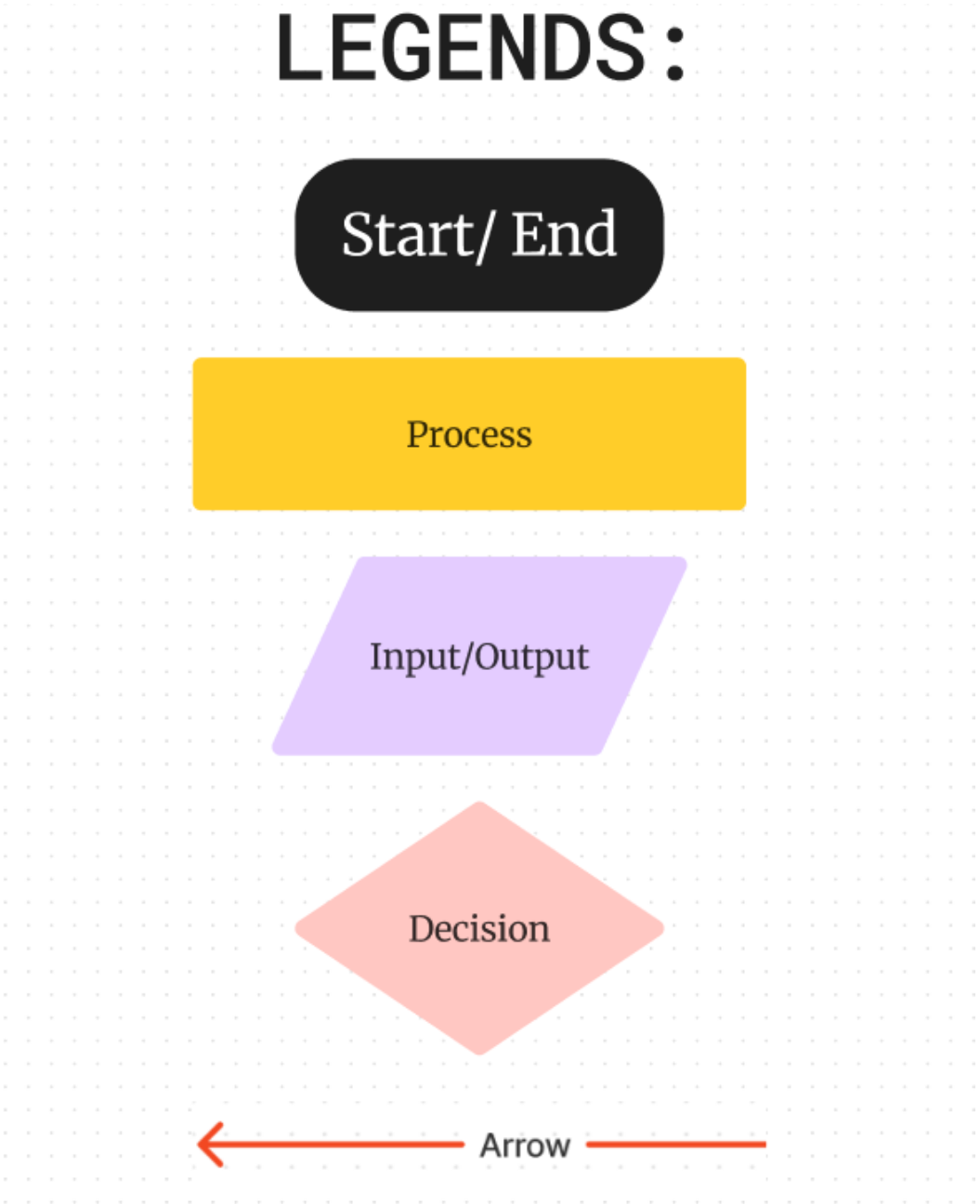


Figure 50. Flowchart Legends

Here,

1. Start/End → Start App and End App
2. Process → Keyframes/ Page
3. Input/Output → User Input
4. Decision → Users' Choices
5. Arrow → Flow

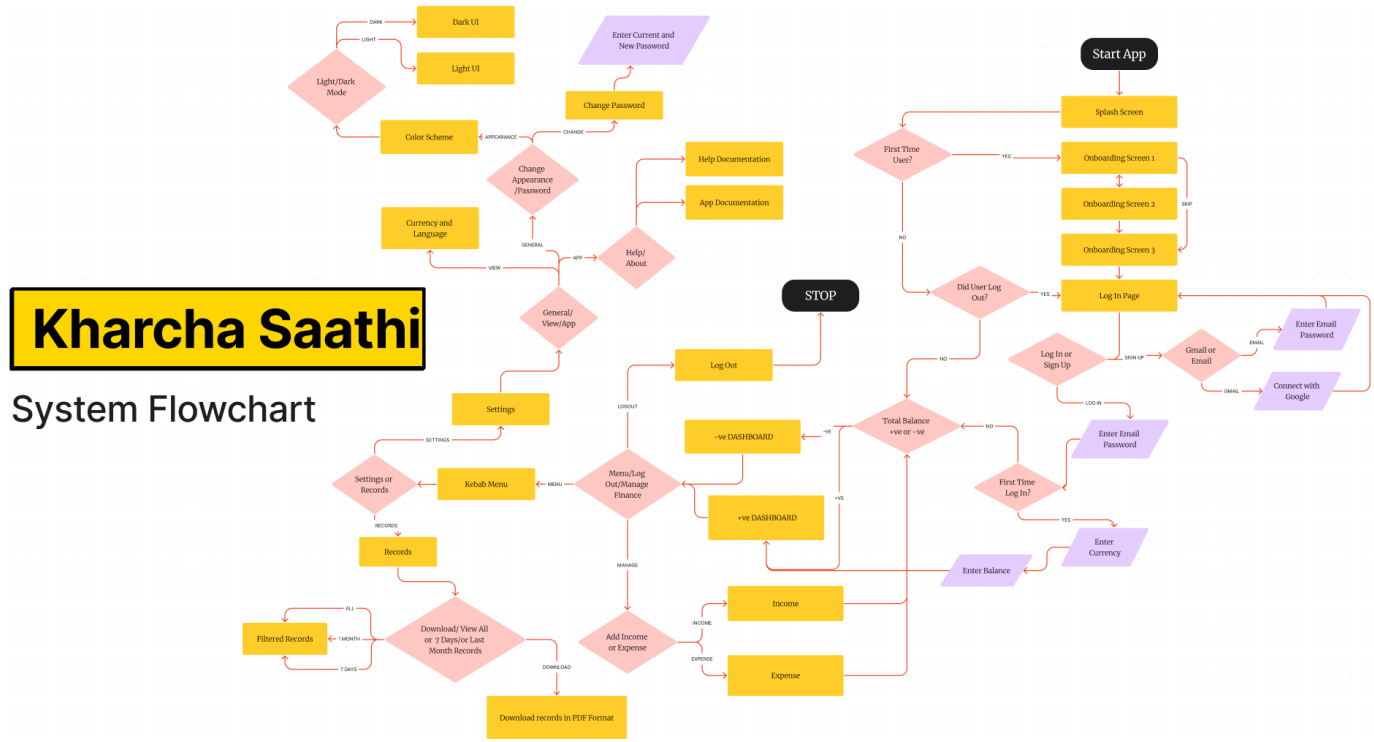


Figure 51. Flowchart of the System

The user boots the app and the splash screen loads. Depending on the users' past activity, either the login page or onboarding page or dashboard loads. If the user didn’t log out then they enter the dashboard page otherwise login page loads and here user can sign up or log in. In signup, they can either set up an account using Google or use email. After setting up their account they can log in in and finish setting up the account by adding currency and their balance. Once they log in, they enter the dashboard page which consists of a text message, a donut chart and total balance text, in this page they can log out, add income or expense, and go to records or setting via the menu. In the records, they can view their transaction history alongside the feature to download the transaction. In the settings, they can change their password or change the theme, and access the help and about page.

4.2. Bug Fixing

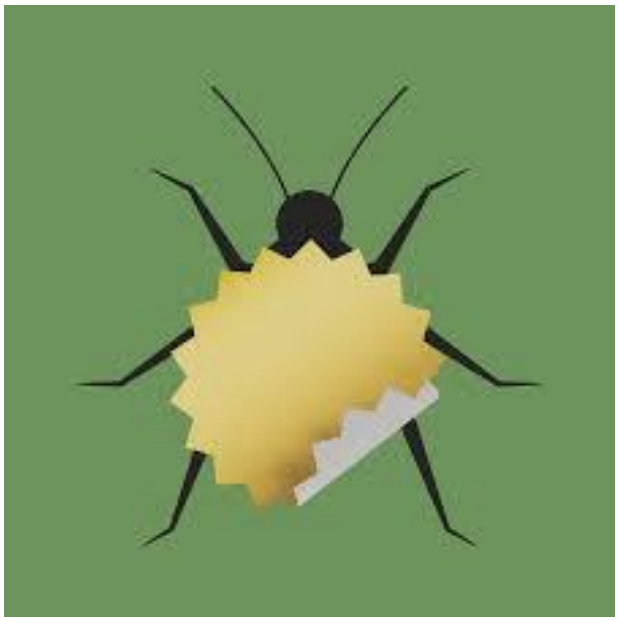
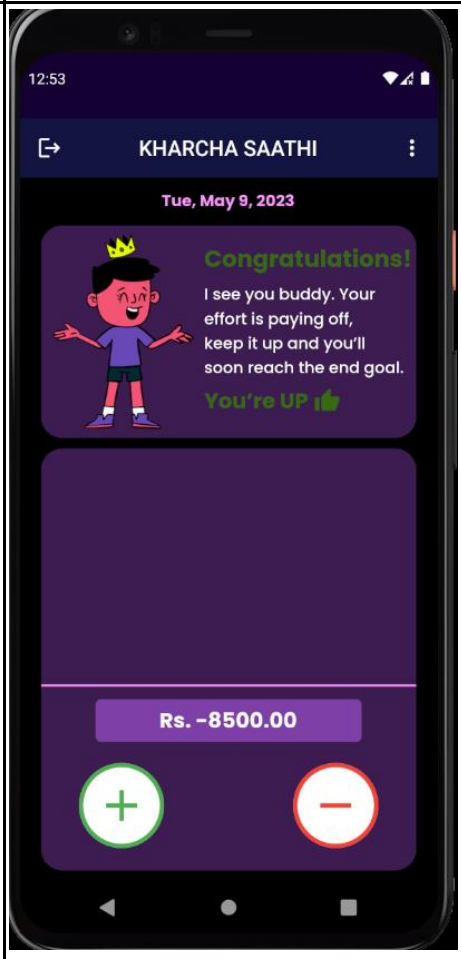


Figure 52. It's not a Bug, it's a Feature

Before deployment of any project, it must be trailed and tested so that the final product can be enjoyed by users without any issues. Similarly, in our project for smooth deployment white-box testing was done. Below are the testing bug reports with proper documentation.

4.2.1. Bug Report 1

ID number	WBT001
Name	Empty Graph
Reporter	Nayan Raj Khanal
Submit Date	7th May, 2023
Summary	After logging in, the total expense VS income donut chart is not being displayed.
Screenshot	 *Figure 53. Bug Report 1 (Bug)*
Phone	Google Pixel 4
Platform	Android 11
Severity	Major
Priority	High

Description

- When I login the container storing the total expense VS income donut chart is not being displayed.

Steps to reproduce

- Log in

Expected result

- Donut chart showing total income and total expense.

Actual result

- The dashboard has an empty space.

Notes

- The problem was occurring because the chart was plotting total income and expense with the values of 0. The GET request was calling for "Expense" and "Income" from the API but the body accepted only "expense" and "income". Hence, the graph was being plotted for 0 VS 0, so nothing was visible.
- Changed "Expense" to "expense" and "Income" to "income".

Fix:

lib > dashboard.dart > _MyDashboardState > getTotal

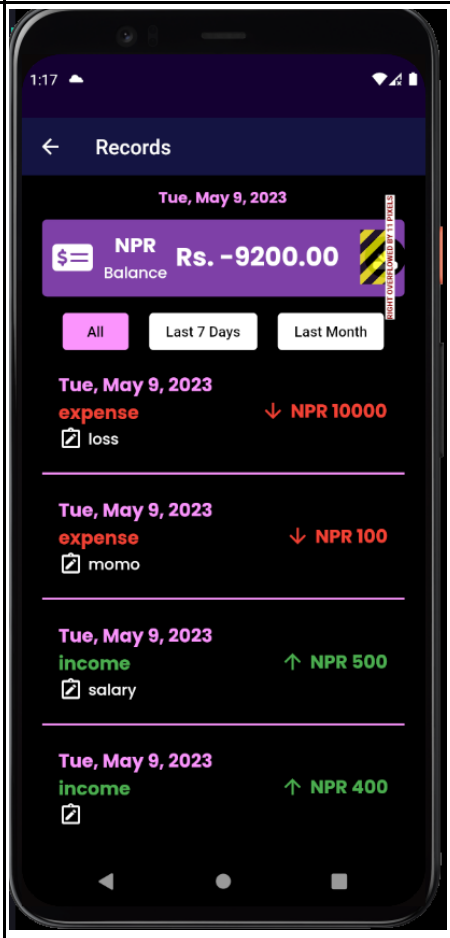
```dart
Future<List<int>> getTotal() async {
  setState(() {
    isContainerLoading = true;
  });
  Dio dio = Dio();
  dio.options.headers = {
    'Authorization':
        'Bearer ${widget.accessToken}', // Set the authorization header with the user's access token.
  };
  try {
    Response response = await dio
        .get('https://kharchasathi-backend.onrender.com/users/Total');
    setState(() {
      isContainerLoading = false; // Set isContainerLoading back to false.
    });
    if (response.statusCode == 200) {
      setState(() {
        expense = int.parse(response.data["expense"].toString());
        income = int.parse(response.data["income"].toString());
      });
      return [expense, income]; // Return the expenses and income as a list.
    } else {
      return [0, 0]; // return 0 for both expense and income on error
    }
  } catch (e) {
    return [0, 0]; // return 0 for both expense and income on exception
  }
}
```

Figure 54. Bug Report 1 (Fix)

Solved:

Figure 55. Bug Report 1 (Proof)

4.2.2. Bug Report 2

ID number	WBT002
Name	Text Overflow
Reporter	Nayan Raj Khanal
Submit Date	7th May, 2023
Summary	After logging in and viewing records, the text that should display the balance overflows.
Screenshot	 *Figure 56. Bug Report 2 (Bug)*
Phone	Google Pixel 4
Platform	Android 11
Severity	Major
Priority	High

Description

- When I login and navigate to records, the container holding balance text that should display the total balance of the user overflows to the right.

Steps to reproduce

- Log in
- Click on kebab menu
- Select records

Expected result

- Balance text fitted inside the container displaying accurate balance.

Actual result

- Balance text displaying overflow error message.

Notes

- The problem was occurring because the text size was too big for the value it was getting back. The GET request was calling for formatted Total Balance from the API but the text size being big could only hold small numbers. Hence, the balance text was overflowing to the right.
- Reduced the text font size so that many numbers could fit.

Fix:

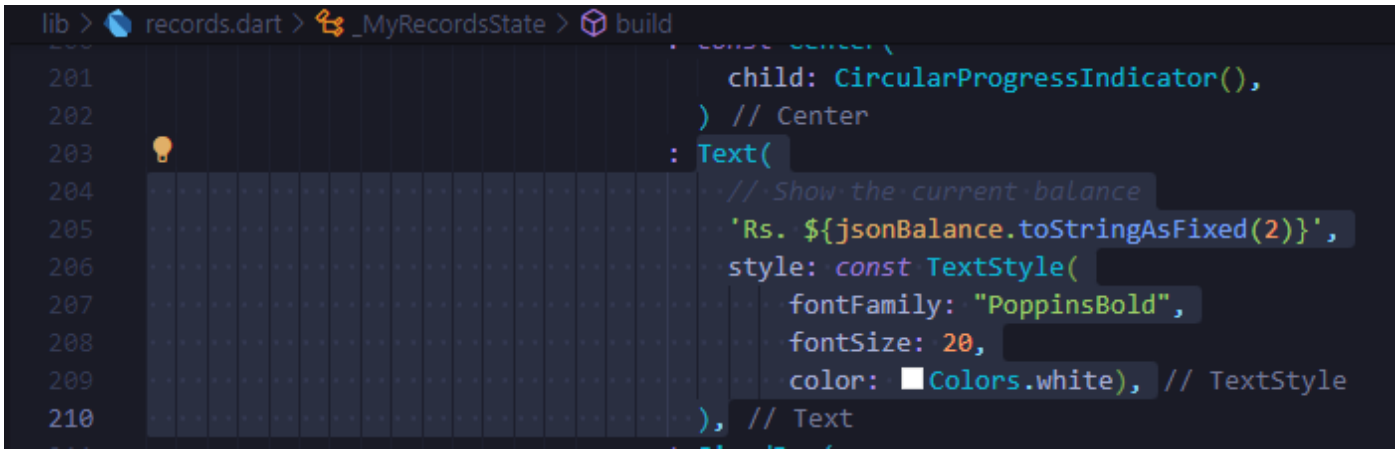


Figure 57. Bug Report 2 (Fix)

Solved:

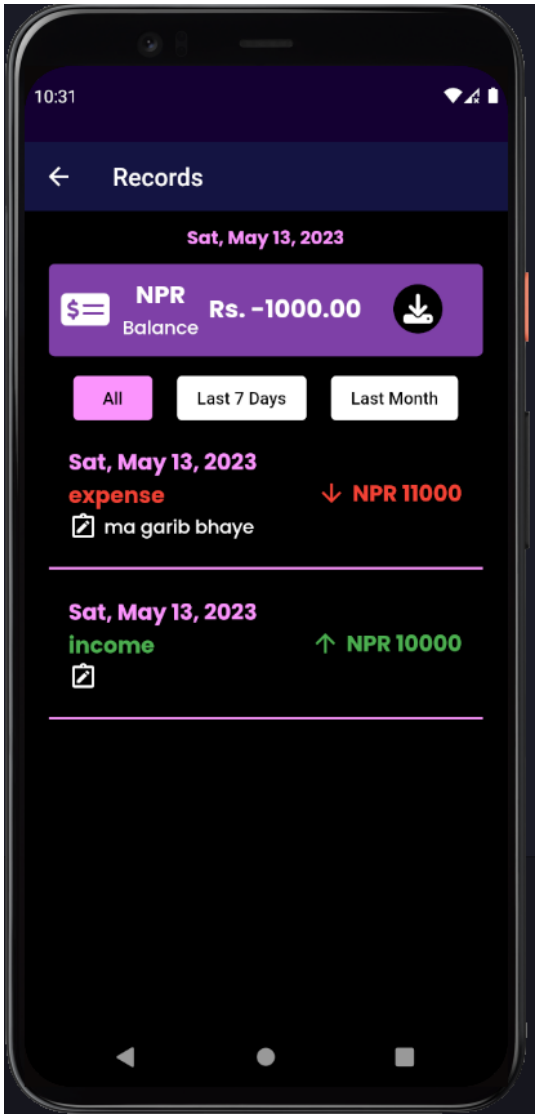
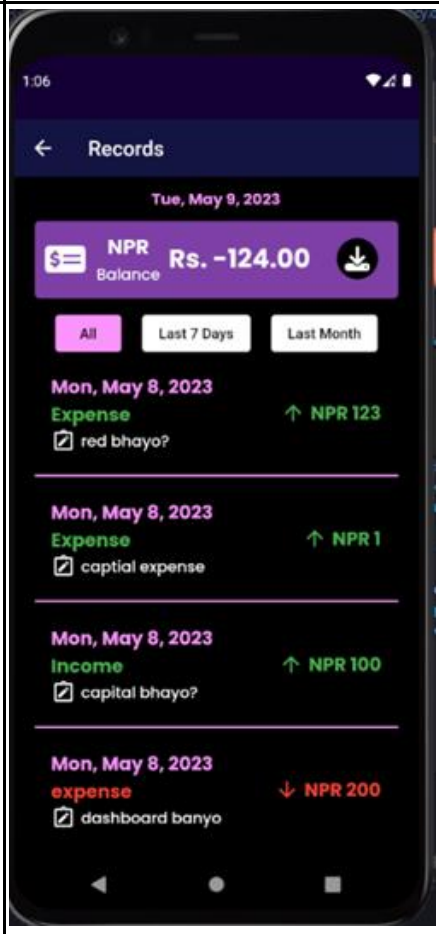


Figure 58. Bug Report 2 (Proof)

4.2.3. Bug Report 3

ID number	WBT003
Name	Colors mismatched
Reporter	Nayan Raj Khanal
Submit Date	8th May, 2023
Summary	Expense on record is meant to be red instead of green whereas income needs to be green
Screenshot	 *Figure 59. Bug Report 3 (Bug)*
Phone	Google Pixel 4
Platform	Android 11
Severity	Major
Priority	High

Description

- When viewing records, the color for expense should be red and the color for income should be green but the colors are swapped.

Steps to reproduce

- Log in
- Open menu
- Go to Records

Expected result

- Expense in red and Income in green

Actual result

- Expense in green

Notes

- The problem was occurring because the data coming from the cluster was changed hence the front end experienced some changes.
- Changed “Changed "expense" to "Expense", hence if the cluster returns Expense, it is red otherwise it is green.

Fix:

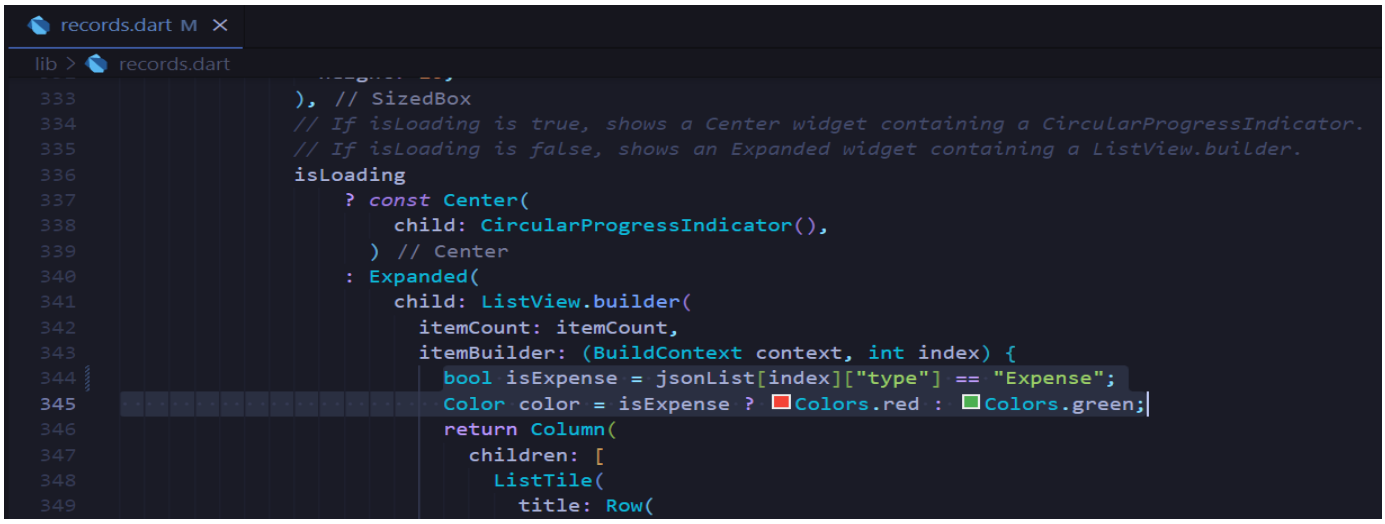


Figure 60. Bug Report 3 (Fix)

Solved:

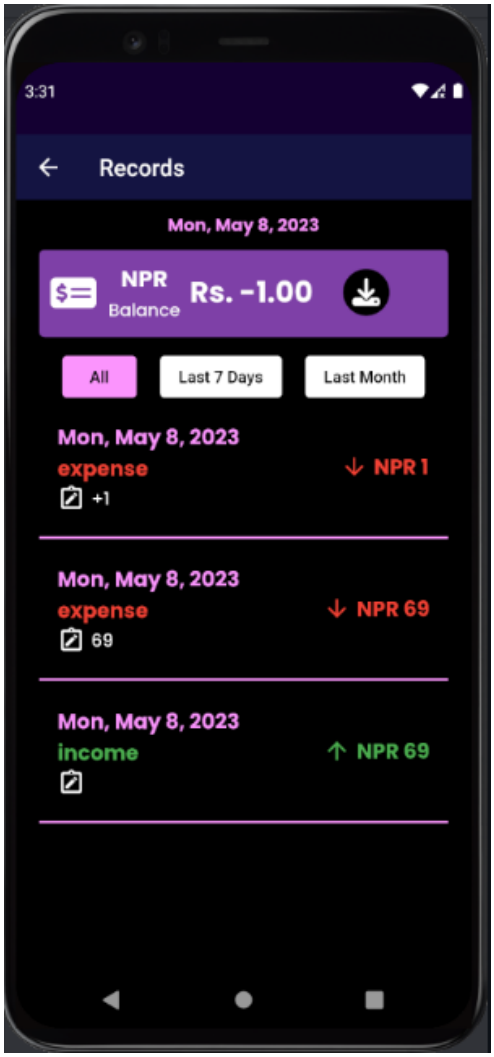
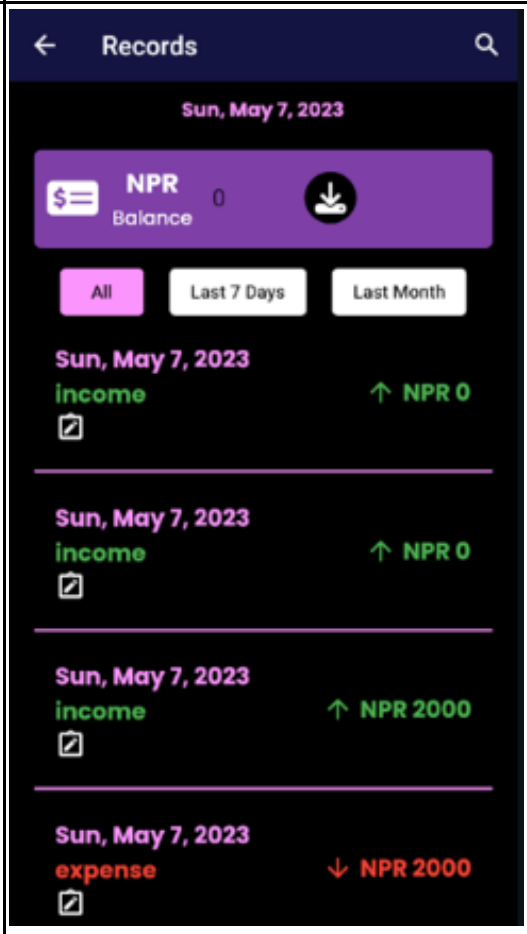


Figure 61. Bug Report 3 (Proof)

4.2.4. Bug Report 4

ID number	WBT004
Name	Empty Balance
Reporter	Nayan Raj Khanal
Submit Date	8th May, 2023
Summary	When viewing records or dashboard the actual balance is not being showed
Screenshot	 *Figure 62. Bug Report 4 (Bug)*
Phone	Google Pixel 4
Platform	Android 11
Severity	Major
Priority	High

Description

- When I click on view records or dashboard the balance is not being shown instead it either have null or 0 value.

Steps to reproduce

- Log in
- View Dashboard/ View Records

Expected result

- View actual current balance of the user

Actual result

- 0 or null value in the balance section

Notes

- The issue was caused by wrong API fetch method.
- Modified the function and changed the PATCH request to GET request.

Fix:

records.dart

lib > records.dart > _MyRecordsState > getTransaction

```dart
  }

  // Get transactions from API
  void getTransaction() async {
    setState(() {
      isLoading = true;
    });

    Dio dio = Dio();

    dio.options.headers = {
      'Authorization': 'Bearer ${box.get("accessToken")}',
    };

    try {
      Response response = await dio
          .get('https://kharchasathi-backend.onrender.com/transaction');
      if (response.statusCode == 200) {
        setState(() {
          jsonList = response.data["transactions"] as List;
          isLoading = false;
        });
      } else {
        isLoading = false;
        ScaffoldMessenger.of(context).showSnackBar(SnackBar(
          content: Text('Error: ${response.statusCode}'),
        )); // SnackBar
      }
    } catch (e) {
      isLoading = false;
      ScaffoldMessenger.of(context).showSnackBar(SnackBar(
        content: Text('Error: $e'),
      )); // SnackBar
    }
  }
```

Figure 63. Bug Report 4 (Fix)

Solved:

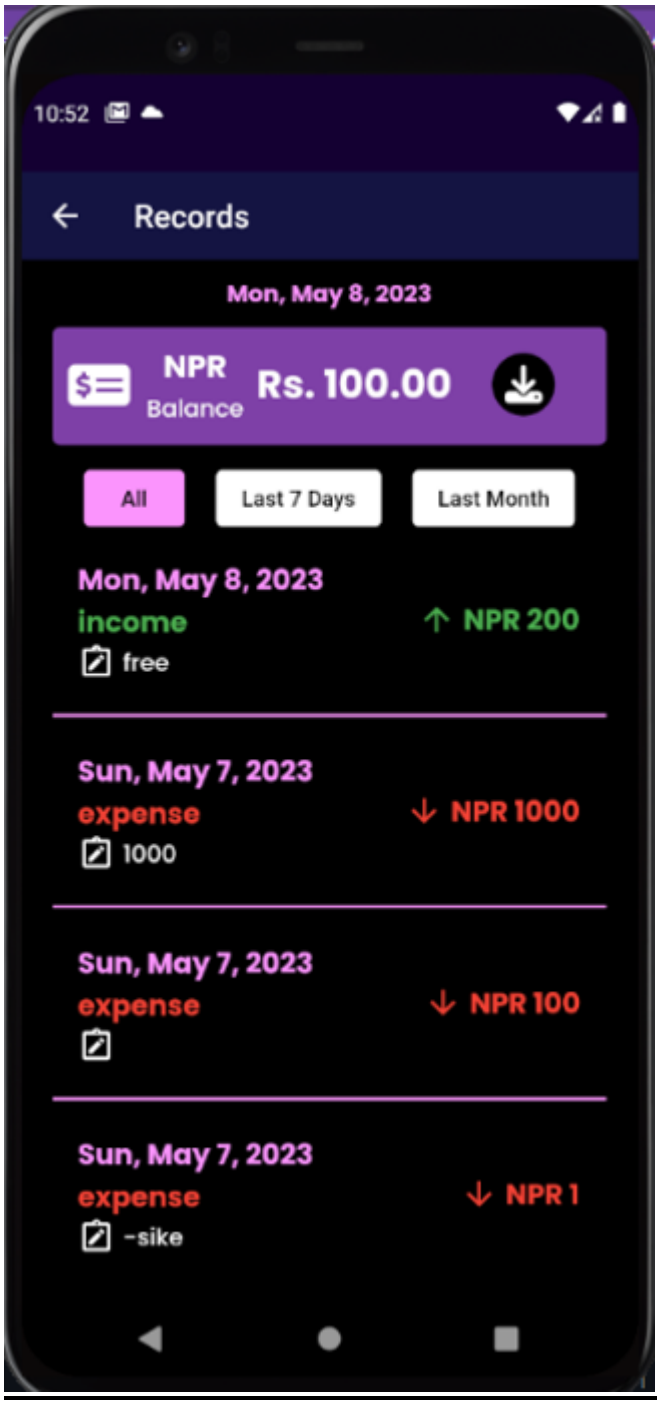
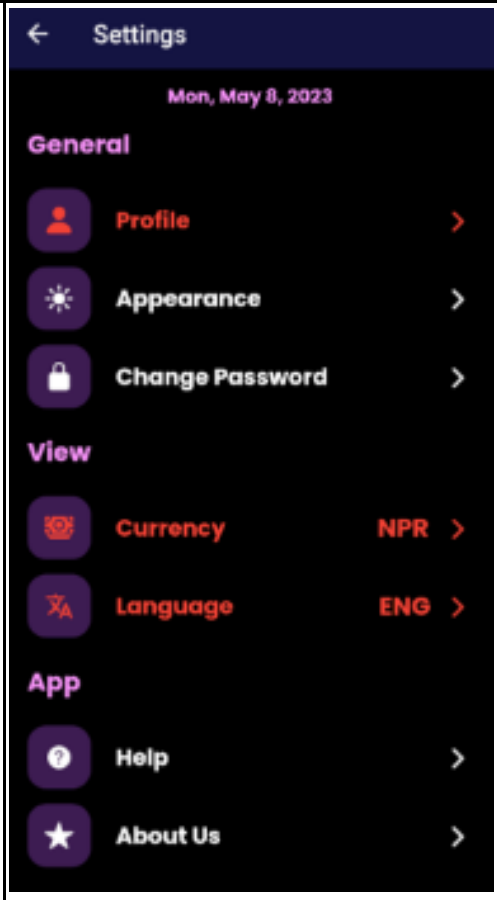


Figure 64. Bug Report 4 (Proof)

4.2.5. Bug Report 5

ID number	WBT005
Name	Return Button Error
Reporter	Nayan Raj Khanal
Submit Date	9th May, 2023
Summary	User can't go back to the fresh dashboard from settings.
Screenshot	 *Figure 65. Bug Report 5 (Bug)*
Phone	Google Pixel 4
Platform	Android 11
Severity	Major
Priority	High

Description

- When trying to go back to dashboard from the Settings page, user gets back to the state when the menu is still open.

Steps to reproduce

- Log in
- Open Menu
- Go to Settings
- Press back button

Expected result

- A new refreshed dashboard

Actual result

- Previous dashboard with the menu still open

Notes

- The problem was occurring because the routing method used did not delete the past action instead the new action was put on top of it, so when going back the old page with menu still open was being displayed.
- Instead of pushReplacement method used Pop to navigate to the settings.

Fix:

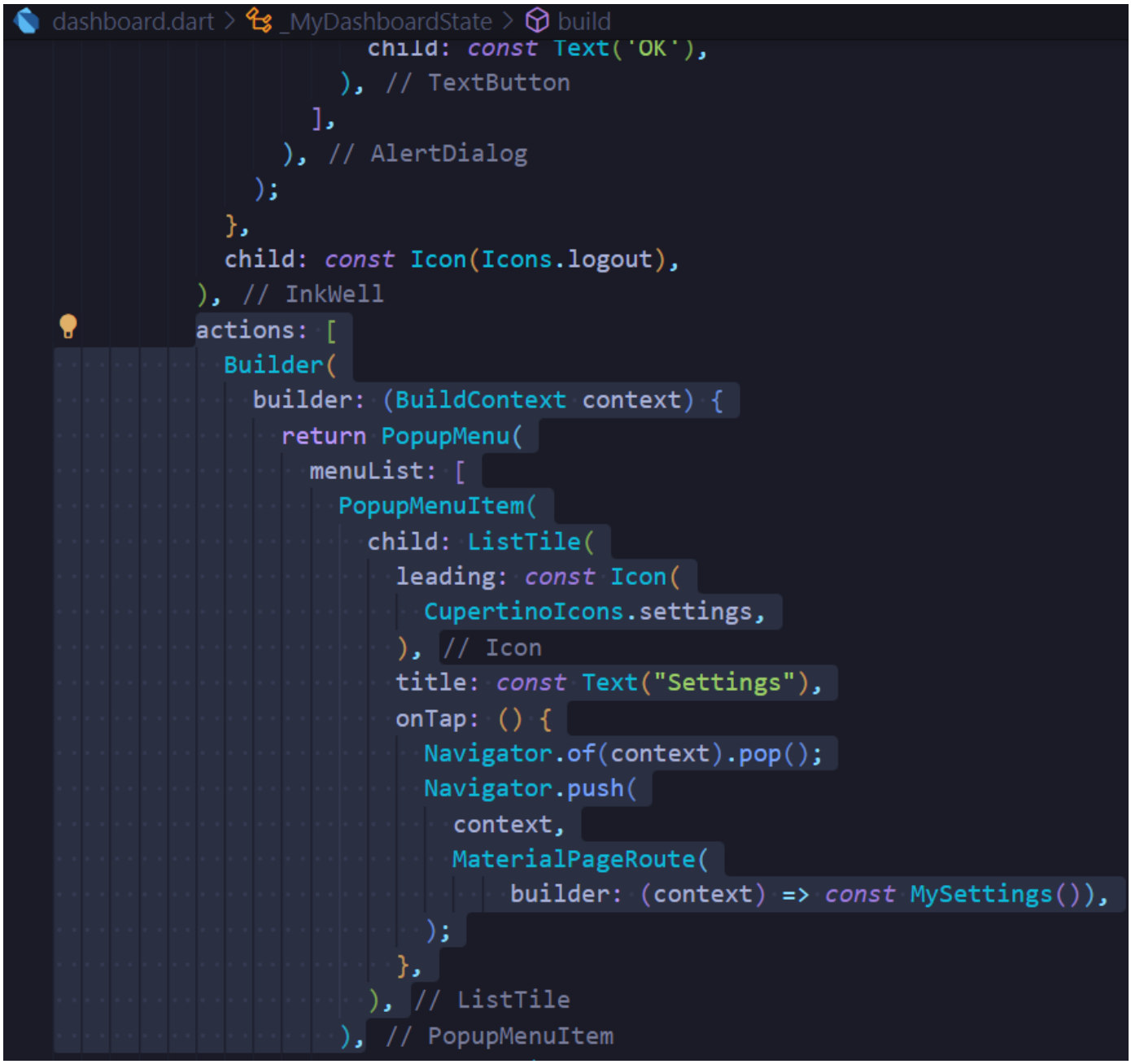


Figure 66. Bug Report 5 (Fix)

Solved:

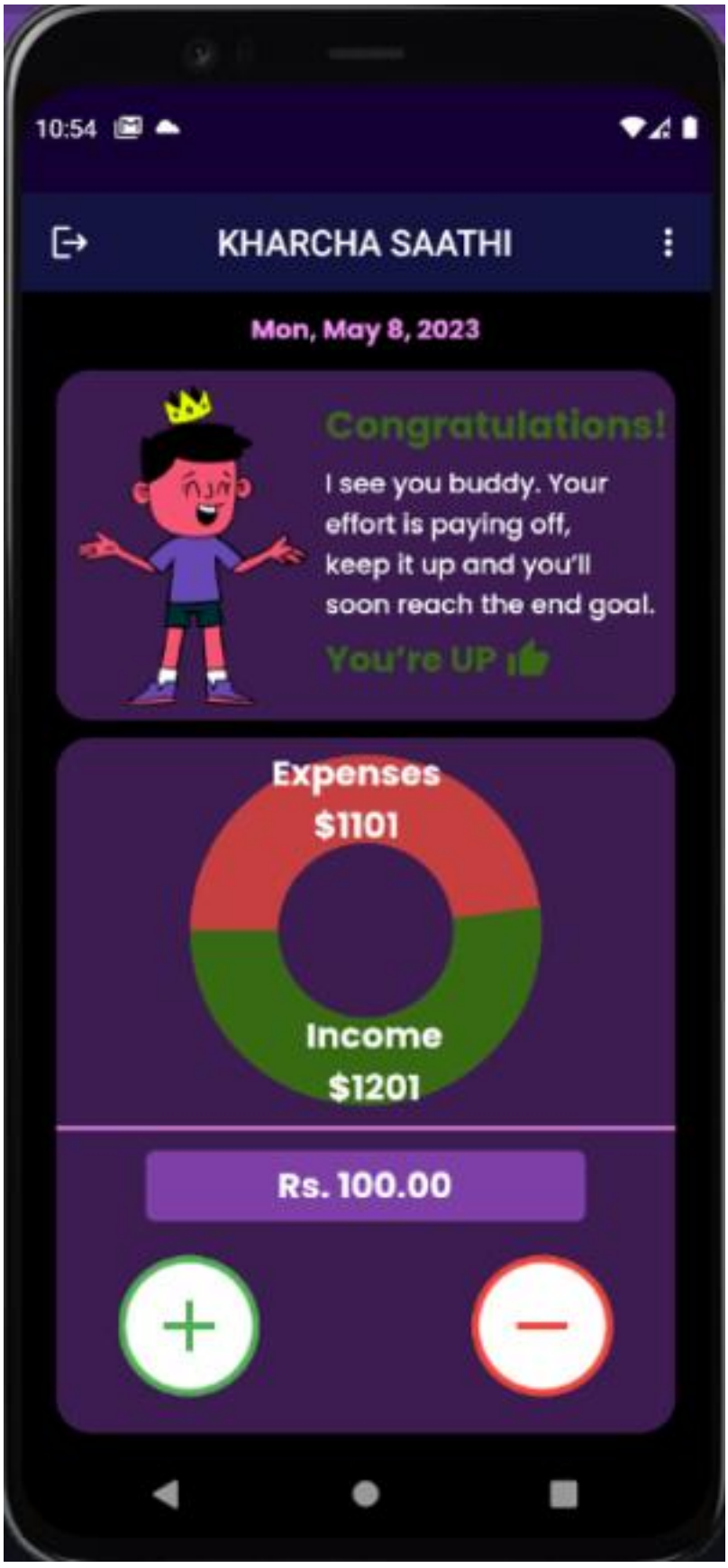


Figure 67. Bug Report 5 (Proof)

5. Appendix B

5.1. Evidences of Good communication and file sharing

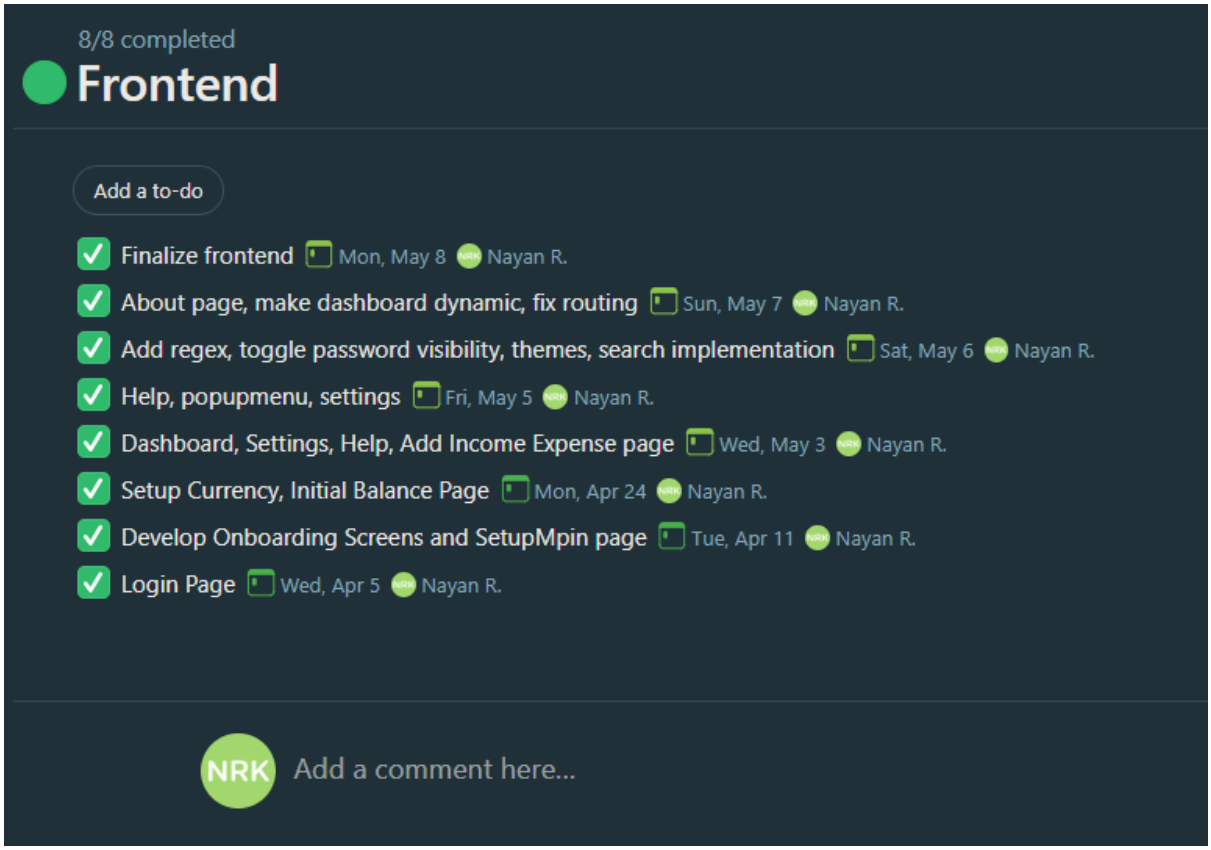


Figure 68. TO DO's Front-End

Timely and diligently checked off all the tasks assigned to me by PM and BA in due time.

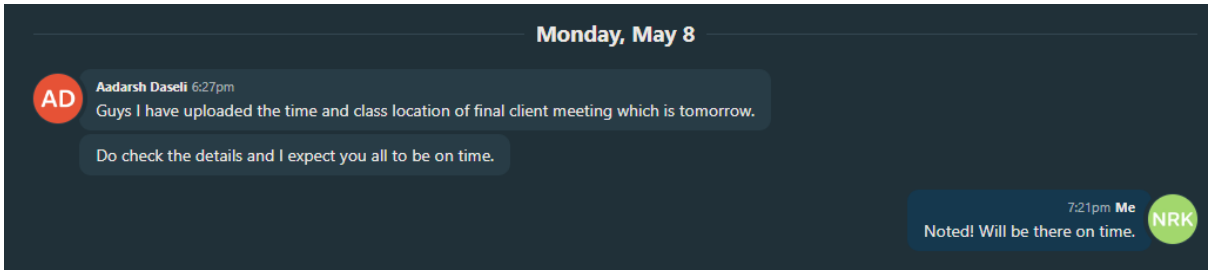


Figure 69. Chats (i)

Responding to messages of team members, in this scenario to the meeting arranged by the PM.

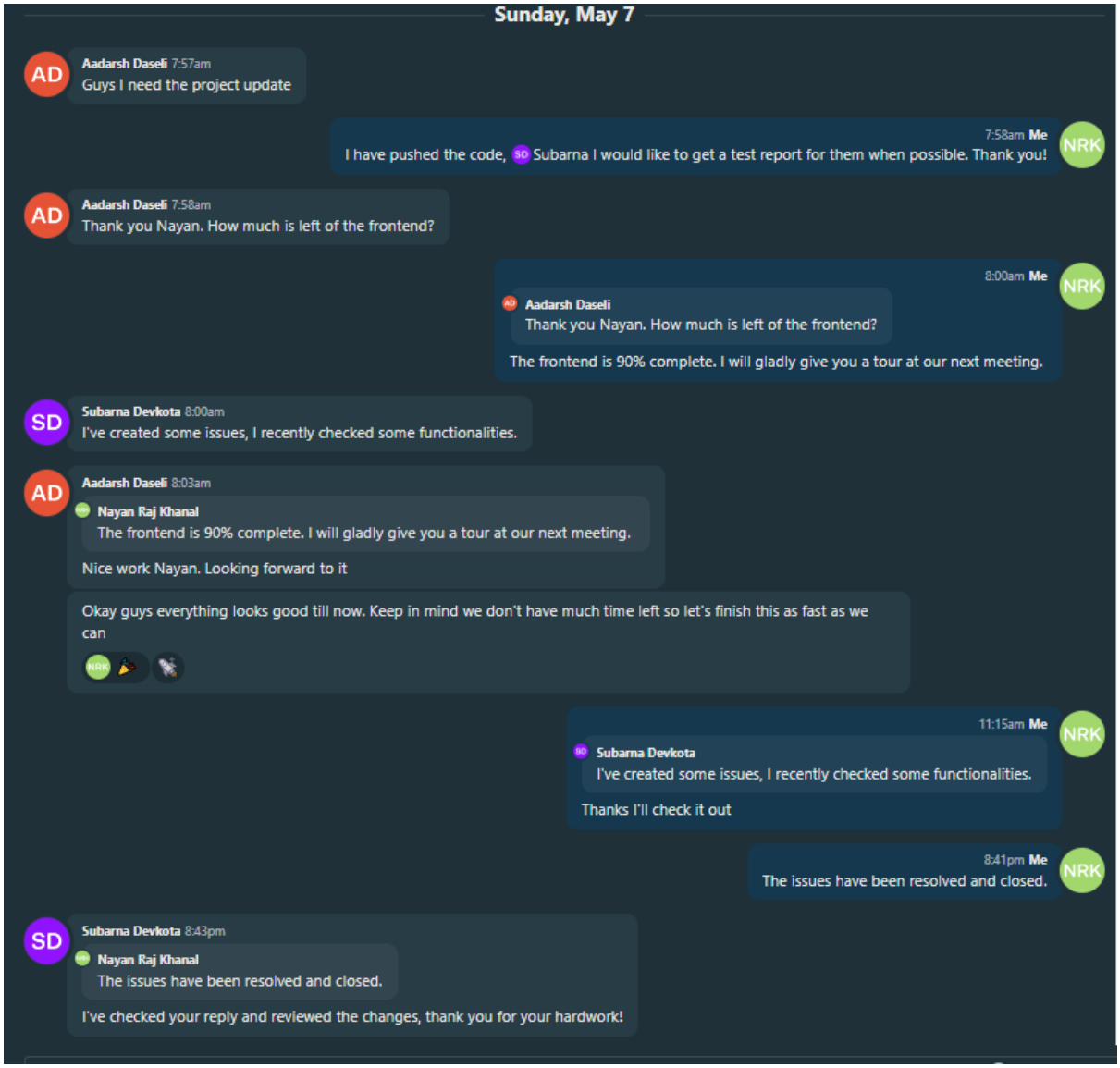


Figure 70. Chats (ii)

- Giving updates of the product to PM.
- Informing BA that the next set of code has been pushed and is waiting for testing.
- Informing the BA that the issues have been resolved and closed.

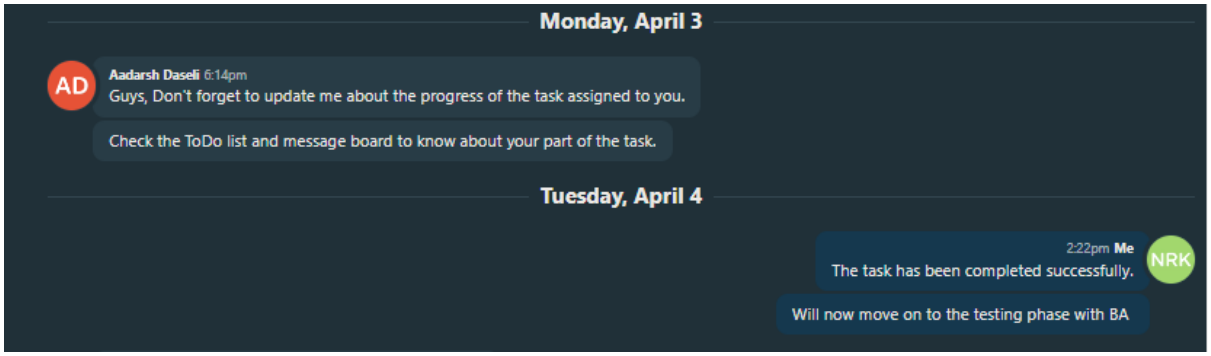


Figure 71. Chats (iii)

Replying to PM regarding completion of the tasks assigned to me.

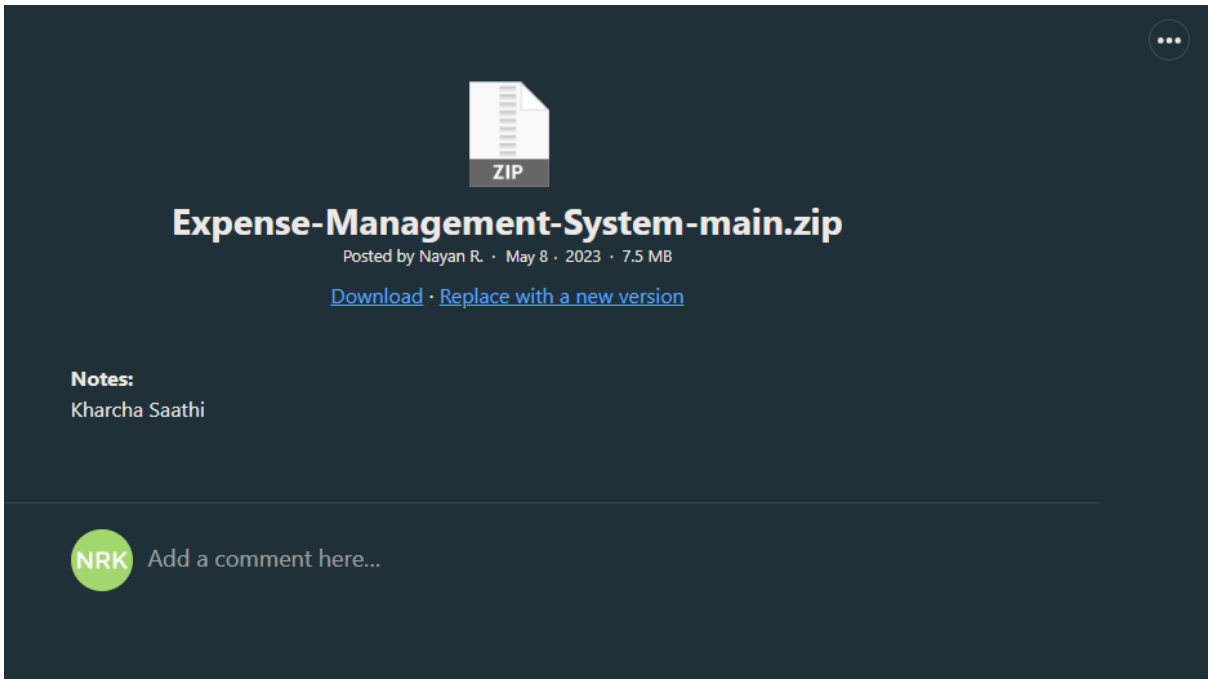


Figure 72. File Sharing

Sharing the final product with all the team members.

5.2. Evidences of Continuing Personal Development (CPD)

"Kharcha Saathi" is a collaborative development between five team members. Even though it is a group project, it would have not been completed if it was not for the personal efforts of each member. Everyone gave it their best that's why this project was a huge success.

I was the developer for this project. As per the discussion by the team, I sorted the entirety of the front end. It was a huge responsibility for someone who knew nothing about mobile app development. In Sprint 1, as the development part was almost zero, I just spent my time designing UI and learning about Flutter. But in Sprint 2 I had to convert my theoretical knowledge into practical knowledge and start developing the app. Luckily, I was able to find plenty of resources on the internet, many tutorials, lectures, live streams etc. Instead of developing the app in one go, I broke it into parts and for each part, I looked for various resources on the web. By doing so, I was able to reduce the workload and finish my assigned task on time. I must also thank my friends who assisted me when I saw myself getting lost. In Sprint 1, I noticed myself taking notes and bookmarking videos a lot and that certainly helped me in Sprint 2 as I caught myself reopening those tabs and rewatching those videos alongside scrolling through Stack Overflow and Flutter Documentations.

I even completed the lengthy tutorial videos that I had started in Sprint 1, especially those freecodecamp videos. I would also like to put Mitch Koko, HeyFlutter and Coding with Tea in the spotlight as these three channels were my go-to spot for any tutorials I needed.

The Internet can be the most useful platform or the most dangerous one, it all depends on the user. I was able to learn Flutter from scratch just from the internet. Now, if someone asks me to develop something for them in Flutter, I am confident that I can produce something that will be accepted in the community. I feel like this project has boosted my morale and skills tremendously and I am so very grateful for everyone involved in this module.

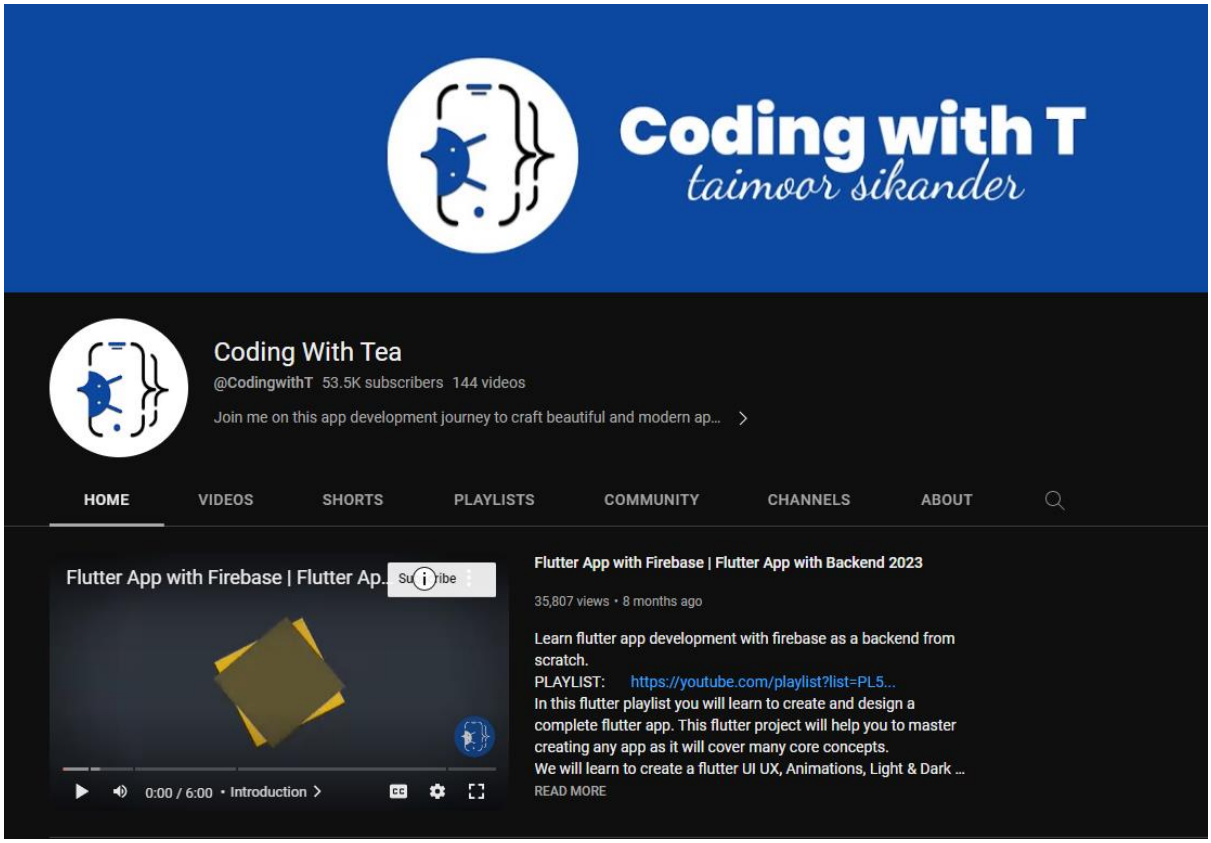


Figure 73. CPD (Evidence 1)

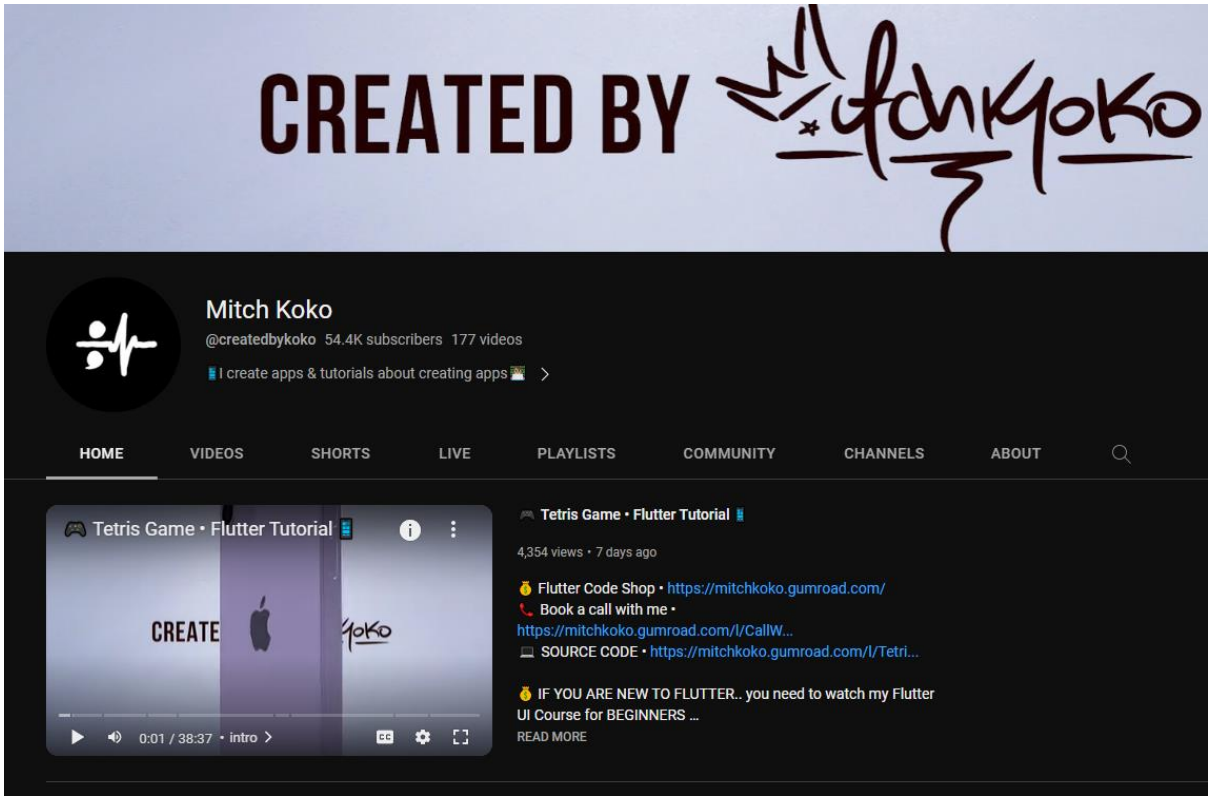


Figure 74. CPD (Evidence 2)

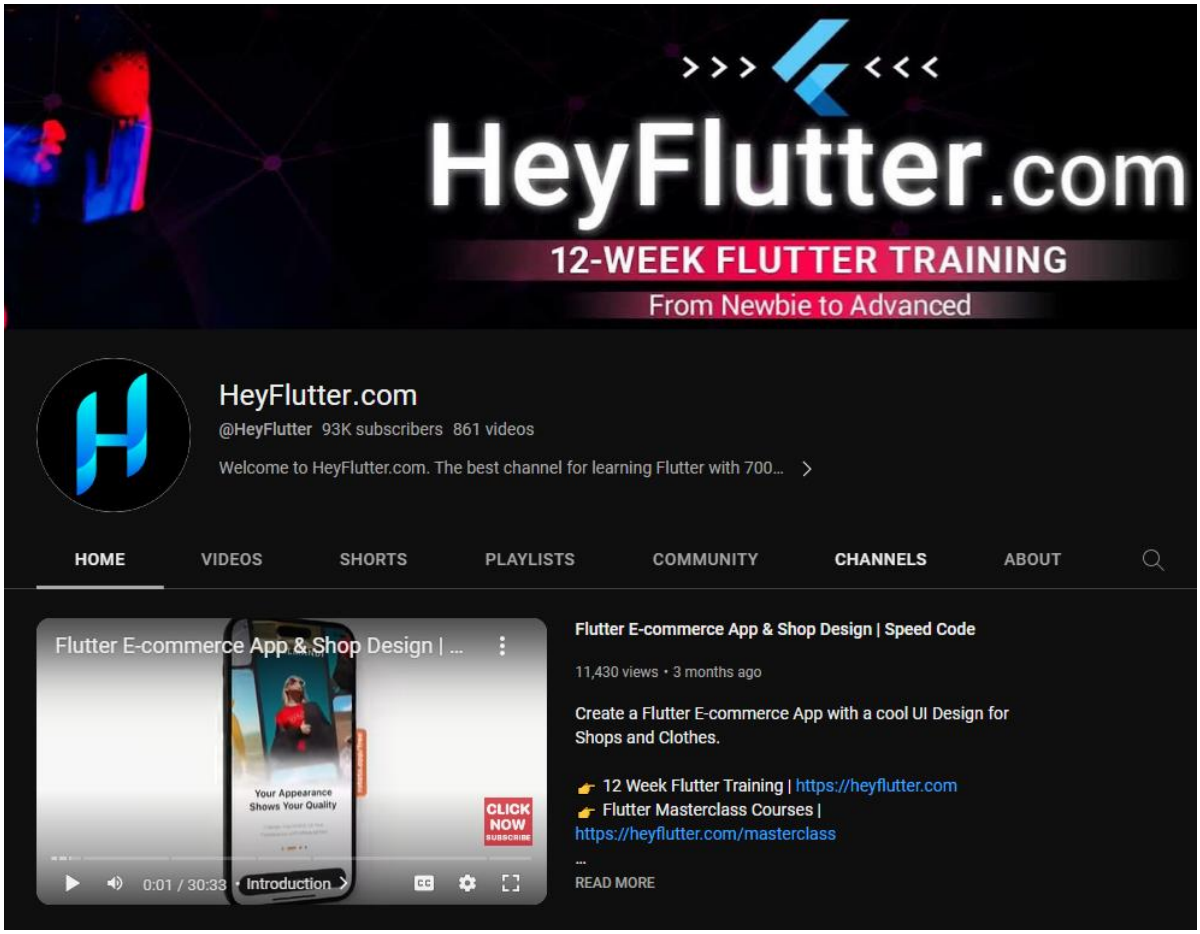


Figure 75. CPD (Evidence 3)

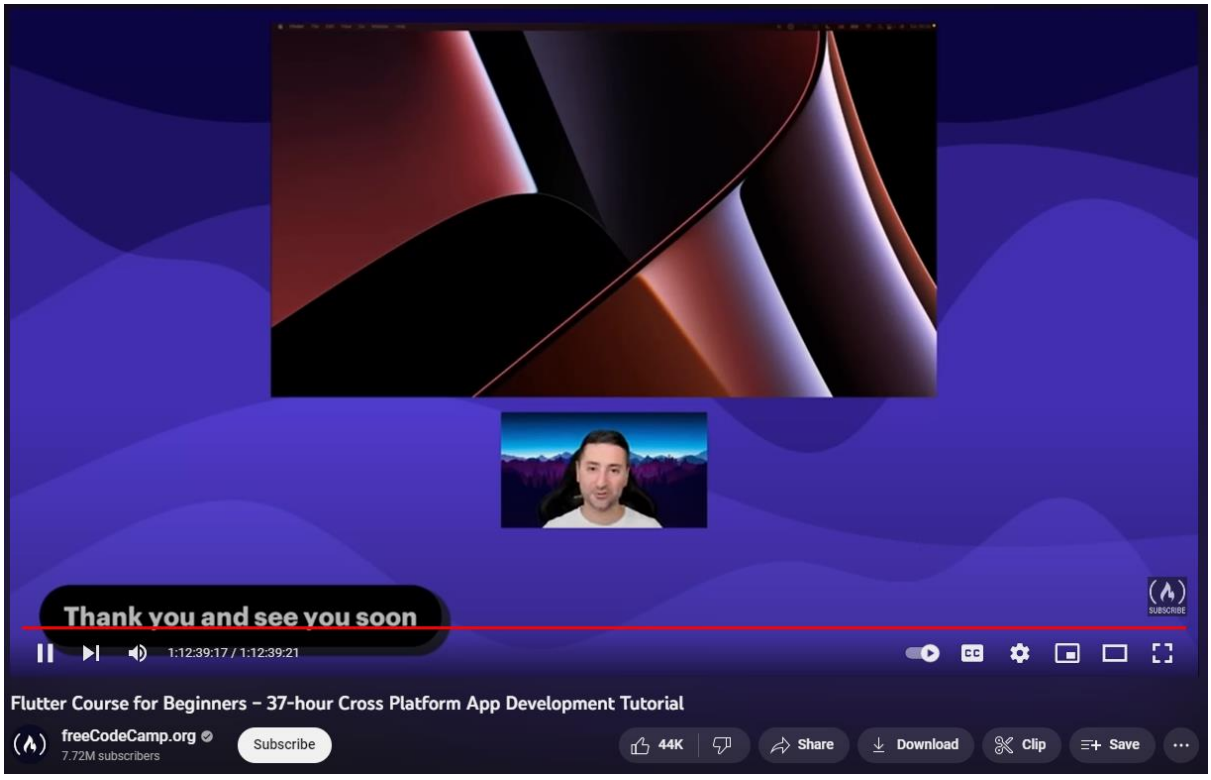


Figure 76. CPD (Evidence 4)

5.3. Evidences of Issue tracking

In order for a fabulous user experience, the application must have zero bugs. This solely relies on the hand of the developer as I have to find the bug and fix it but in this case the burden is shared between the BA and me as they are also responsible for testing and pointing out the bugs but at the end it is up to me to solve them and deploy the app bug free.

Testing for the front end was done between BA and me. For effective issue tracking we used GitHub Issues. This allowed BA to effortlessly point out bugs, create issues and assign them to me. And I could easily sort them and provide fixes. Without Issues, the collaboration between me and the BA would not have been easy and there was a chance of deploying the app with bugs and errors.

The evidence pointed out below is all between me and the BA. It consists of BA pointing out the bug and assigning the task to fix it to me. I on the other hand post the cause of the error and provide the proof that the bug has been resolved and then close the issue.

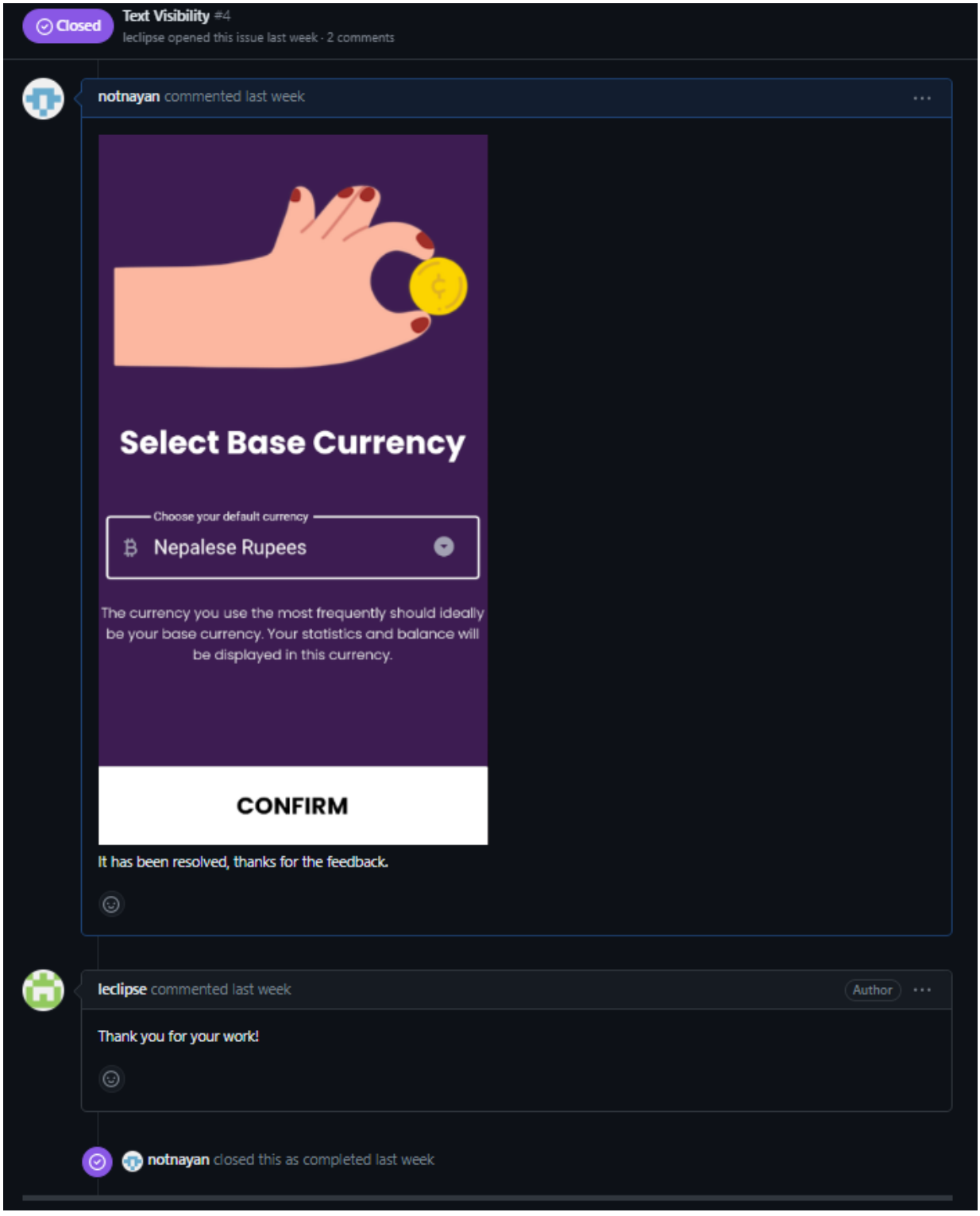


Figure 77. Issue Tracking (Evidence 1)

Here, BA raised the issue of visibility and asked me to change the color of the icons, dropdown menu and the button. I provided the fix by changing the colors to white and closed the issue in due time.

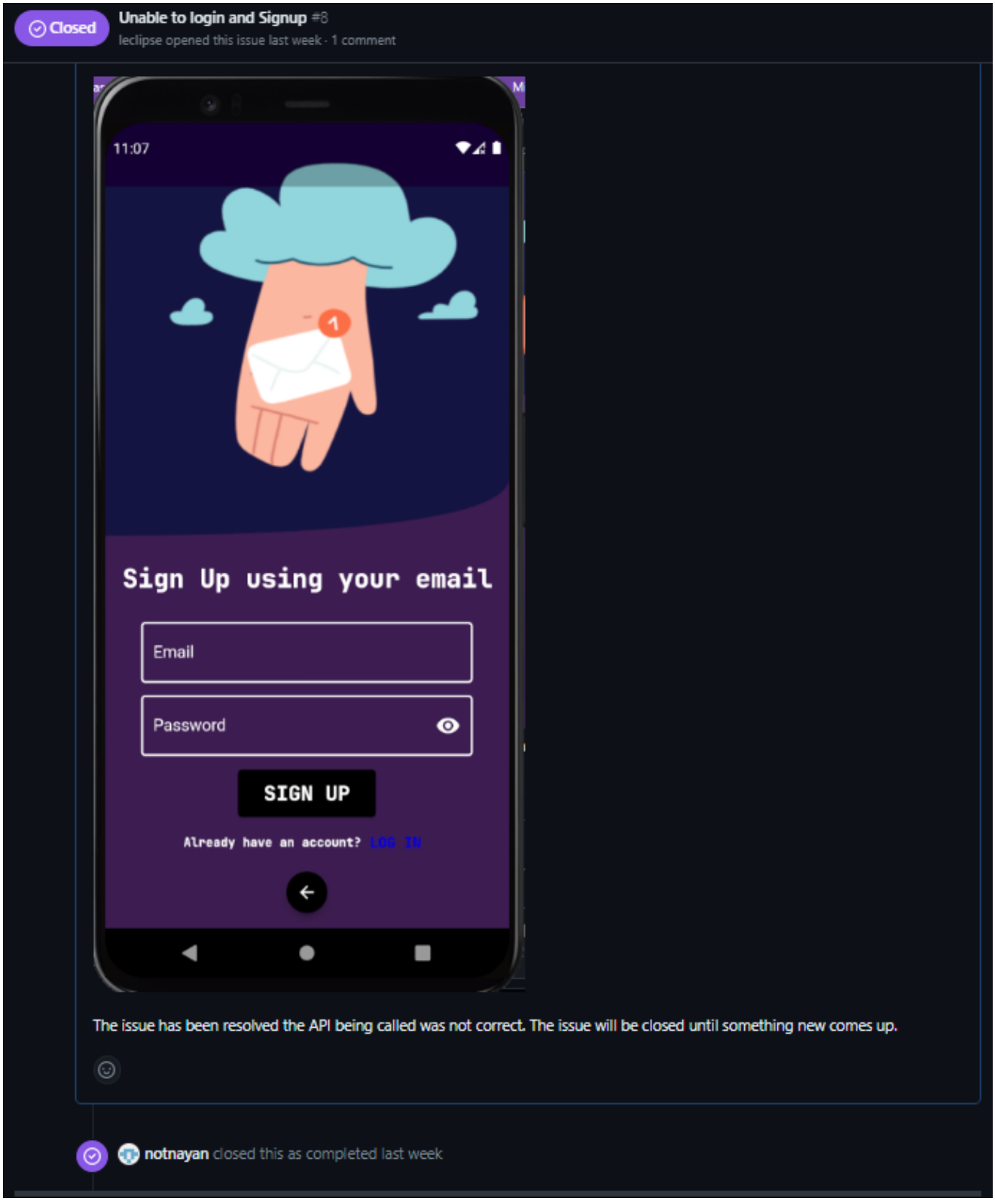


Figure 78. Issue Tracking (Evidence 2)

Here, BA raised the issue of not being able to login or sign up and assigned me the task to fix the issue. The error was occurring because of a faulty URL which I fixed and closed the issue.

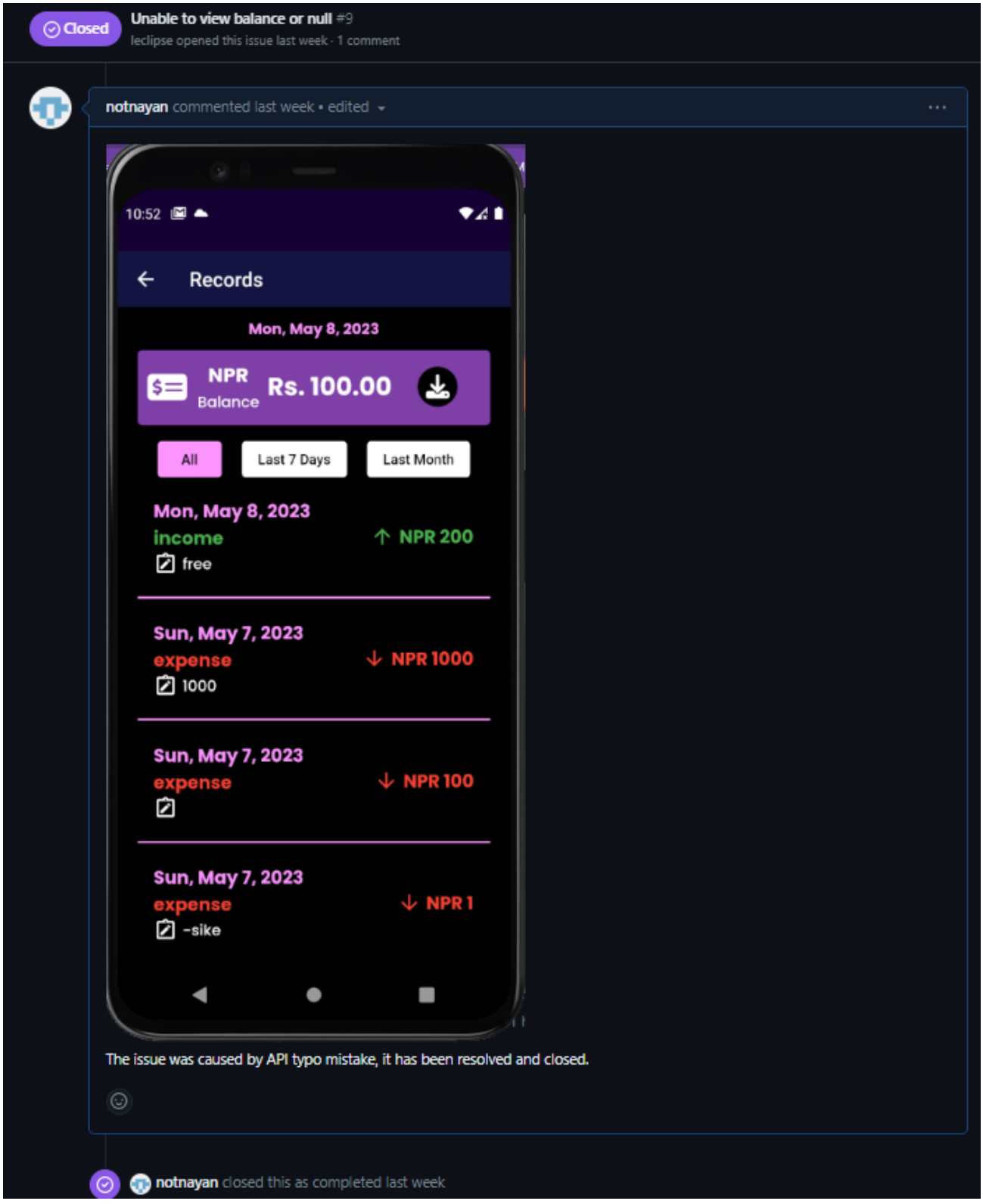


Figure 79. Issue Tracking (Evidence 3)

Here, BA raised the issue of not being able to view the balance in the records page. Here, again the API being called was mistyped which was causing the error so it was fixed in due time and the issue was closed.

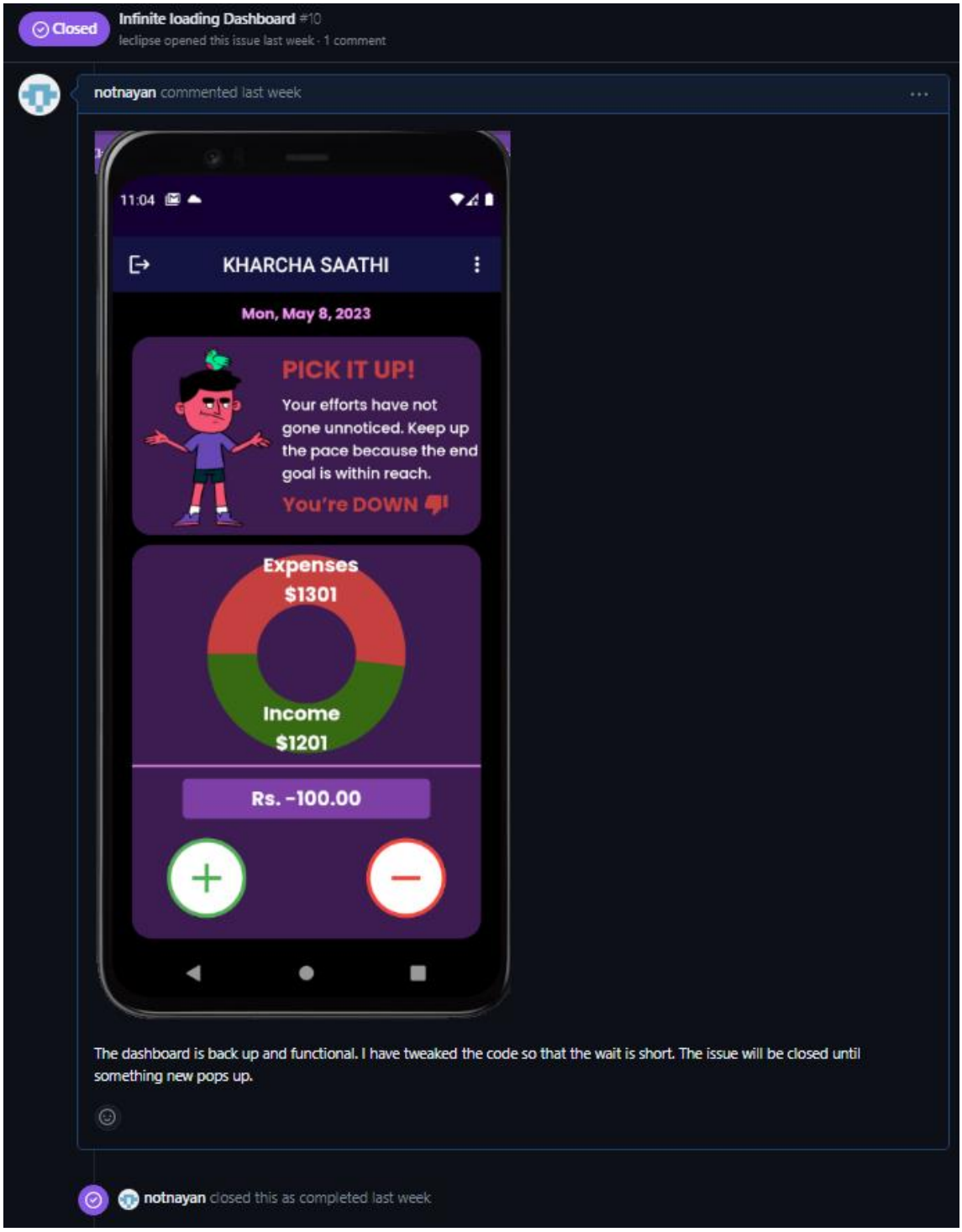


Figure 80. Issue Tracking (Evidence 4)

Here, the BA pointed out that the loading of text and chart was taking too long. Upon checking the bug was born due to the slow response time from the backend, to tackle this instead of calling the whole container I just called the variable so that the conditional statements could operate quickly and reduce the loading time and upon fixing the bug closed the issue.

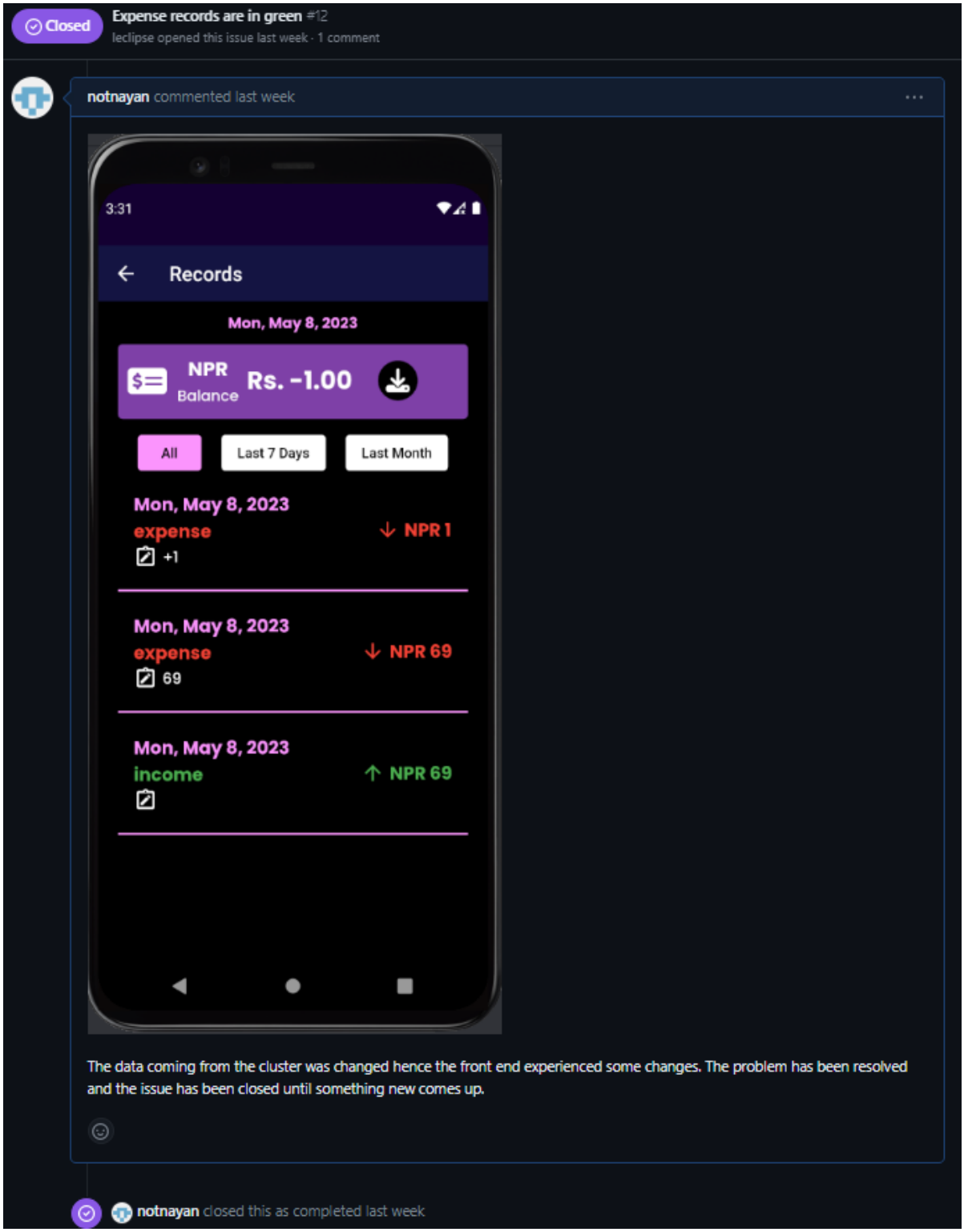


Figure 81. Issue Tracking (Evidence 5)

Here, the BA opened an issue stating that the color code for income and expense were opposite. This was occurring because the data coming from the backend was altered hence the front end was affected and need some changes too. Upon slight modification it was resolved and the issue was closed.

6. APPENDIX

6.1. Link to Flowchart

https://www.figma.com/file/yiVcIKuWjxbivK5asCNqpU/Nayan_System_Flowchart?type=whiteboard&node-id=0%3A1&t=UDxogbyZqDSX11h7-1

6.2. Link to Developer Presentation

https://www.canva.com/design/DAFiVHssgq4/ECangxZbRjm_eF3h4vRJbQ/edit?utm_content=DAFiVHssgq4&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton

6.3. Link to the Four Channels

freeCodeCamp: https://www.youtube.com/watch?v=VPvVD8t02U8

Mitch Koko: https://www.youtube.com/@createdbykoko

HeyFlutter; https://www.youtube.com/@HeyFlutter

Coding with Tea: https://www.youtube.com/@CodingwithT

6.4. Link to the GitHub Repository

https://github.com/A-eli/Expense-Management-System