

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

1.	Self-appraisal form	1
2.	Personal objectives – performance measurement	1
3.	Collaboration Document.....	3
3.1.	Evidence of good communication and file sharing	3
3.2.	Continuing Personal Development (CPD)	3
3.3.	Issue tracking.....	3
4.	Appendix A	4
4.1.	Implementing functional requirement.....	4
4.1.1.	Use of Version Control	5
4.1.2.	Commits	41
4.1.3.	Flow Chart of the system	44
4.2.	Bug Fixing	46
4.2.1.	Bug Report 1.....	47
4.2.2.	Bug Report 2.....	51
4.2.3.	Bug Report 3.....	54
4.2.4.	Bug Report 4.....	57
4.2.5.	Bug Report 5.....	61
5.	Appendix B	65
5.1.	Evidences of Good communication and file sharing	65
5.2.	Evidences of Continuing Personal Development (CPD)	68
5.3.	Evidences of Issue tracking	71
6.	APPENDIX.....	77
6.1.	Link to Flowchart	77
6.2.	Link to Developer Presentation	77
6.3.	Link to the Four Channels.....	77
6.4.	Link to the GitHub Repository	77

Table of Figures

Figure 1. Product Backlog	4
Figure 2. Dart Files.....	5
Figure 3. Key Frames	6
Figure 4. Login (Output).....	7
Figure 5. Login (Code 1).....	8
Figure 6. Login (Code 2).....	9
Figure 7. Login (Code 3).....	10
Figure 8. Login (Code 4).....	11
Figure 9. Sign Up Email (Output)	12
Figure 10. Sign Up Email (Code 1)	13
Figure 11. Sign Up Email (Code 2)	14
Figure 12. Sign Up Email (Code 3)	15
Figure 13. Sign Up Email (Code 4)	16
Figure 14. Income (Output)	17
Figure 15. Input (Code 1).....	18
Figure 16. Input (Code 2).....	19
Figure 17. Input (Code 3).....	20
Figure 18. Expense (Output).....	21
Figure 19. Expense (Code 1).....	22
Figure 20. Expense (Code 2).....	23
Figure 21. Expense (Code 3).....	24
Figure 22. Records (Output)	25
Figure 23. Records (Code 1)	26
Figure 24. Records (Code 2)	27
Figure 25. Records (Code 3)	28
Figure 26. Records (Code 4)	29
Figure 27. Records (Code 5)	30
Figure 28. Records (Code 6)	31
Figure 29. Records (Code 7)	32
Figure 30. Change Password (Output)	33
Figure 31. Change Password (Code 1).....	34
Figure 32. Change Password (Code 2).....	35
Figure 33. Change Password (Code 3).....	36
Figure 34. Change Password (Code 4).....	37
Figure 35. Change Password (Code 5).....	38
Figure 36. Change Password (Code 6).....	39
Figure 37. Change Password (Code 7).....	40
Figure 38. GitHub Commit (Sprint 1)	41
Figure 39. GitHub Commit (April 2)	41

Figure 40. GitHub Commit (April 5)	41
Figure 41. GitHub Commit (April 27)	41
Figure 42. GitHub Commit (May 1).....	42
Figure 43. GitHub Commit (May 3).....	42
Figure 44. GitHub Commit (May 4).....	42
Figure 45. GitHub Commit (May 5).....	42
Figure 46. GitHub Commit (May 6).....	43
Figure 47. GitHub Commit (May 7).....	43
Figure 48. GitHub Commit (May 8).....	43
Figure 49. GitHub Commit (May 9).....	43
Figure 50. Flowchart Legends	44
Figure 51. Flowchart of the System	45
Figure 52. It's not a Bug, it's a Feature.....	46
Figure 53. Bug Report 1 (Bug)	47
Figure 54. Bug Report 1 (Fix).....	49
Figure 55. Bug Report 1 (Proof).....	50
Figure 56. Bug Report 2 (Bug)	51
Figure 57. Bug Report 2 (Fix).....	53
Figure 58. Bug Report 2 (Proof)	53
Figure 59. Bug Report 3 (Bug)	54
Figure 60. Bug Report 3 (Fix).....	56
Figure 61. Bug Report 3 (Proof)	56
Figure 62. Bug Report 4 (Bug)	57
Figure 63. Bug Report 4 (Fix).....	59
Figure 64. Bug Report 4 (Proof)	60
Figure 65. Bug Report 5 (Bug)	61
Figure 66. Bug Report 5 (Fix).....	63
Figure 67. Bug Report 5 (Proof)	64
Figure 68. TO DO's Front-End	65
Figure 69. Chats (i)	65
Figure 70. Chats (ii)	66
Figure 71. Chats (iii).....	67
Figure 72. File Sharing	67
Figure 73. CPD (Evidence 1)	69
Figure 74. CPD (Evidence 2)	69
Figure 75. CPD (Evidence 3)	70
Figure 76. CPD (Evidence 4)	70
Figure 77. Issue Tracking (Evidence 1).....	72
Figure 78. Issue Tracking (Evidence 2).....	73
Figure 79. Issue Tracking (Evidence 3).....	74
Figure 80. Issue Tracking (Evidence 4).....	75
Figure 81. Issue Tracking (Evidence 5).....	76

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 <i>Student / tutor</i>
Implementing functional requirements	<p>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.</p> <p>Provided screenshots of codes, flowchart of the system and commit logs.</p> <p><u>Appendix A1</u></p>	9

Tutor feedback:

Bug Fixing	<p>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.</p> <p>Provided screenshots of White-Box Testing and GitHub Issues.</p> <p><u>Appendix A2</u></p>	9	
<i>Tutor feedback:</i>			
		/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 with a valid email and password and 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
Expense management	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
	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
		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 have 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.

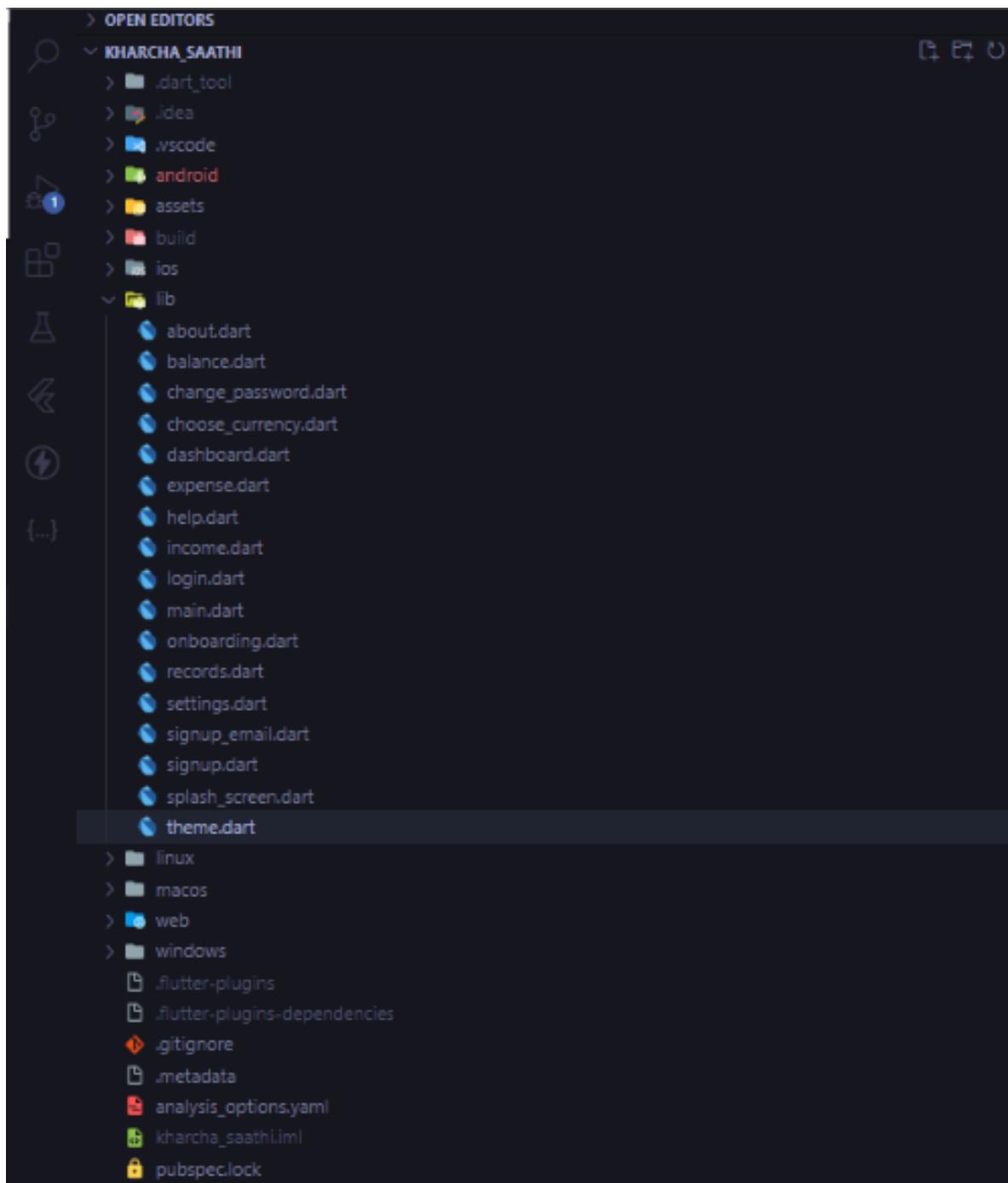


Figure 2. Dart Files

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

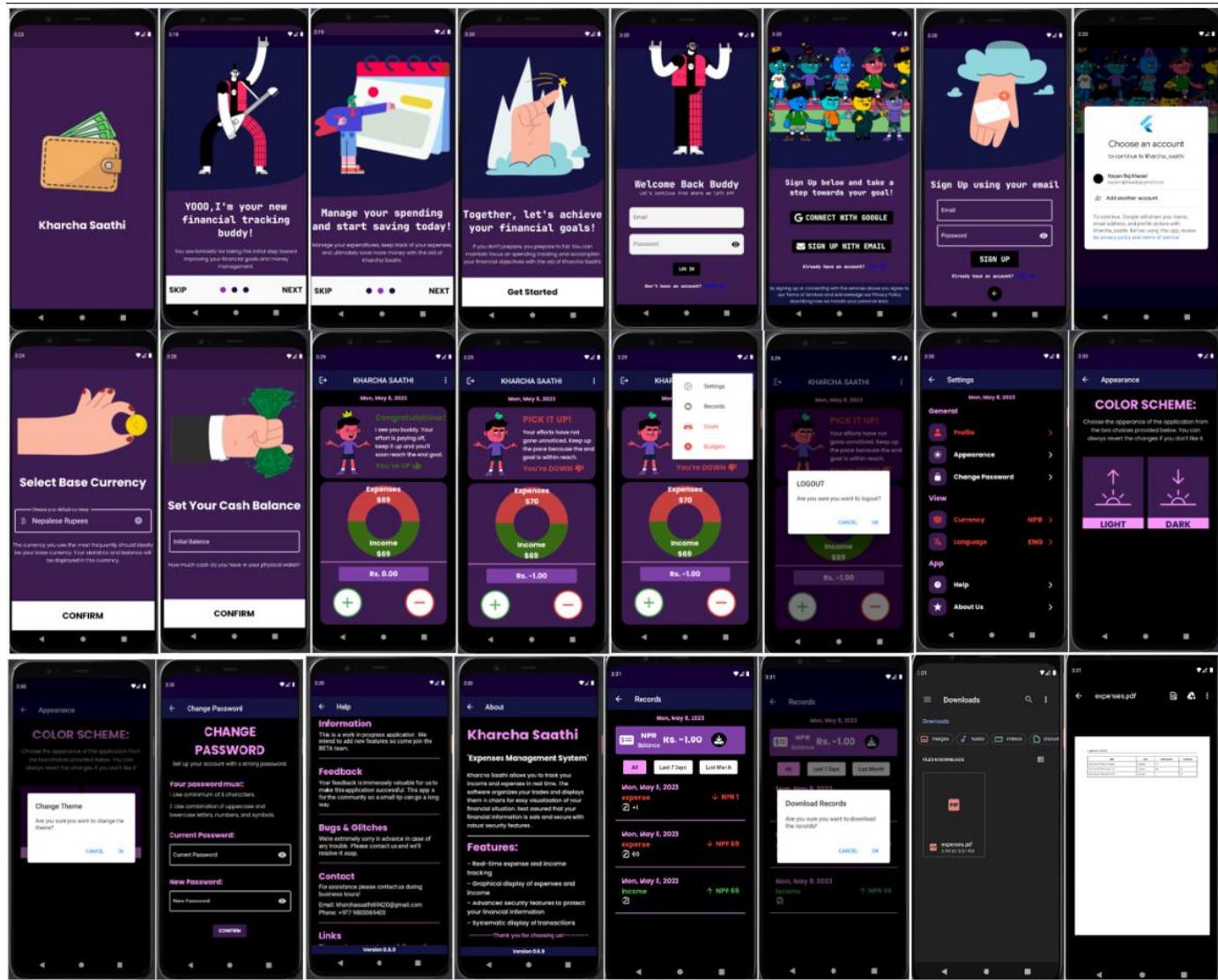


Figure 3. Key Frames

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

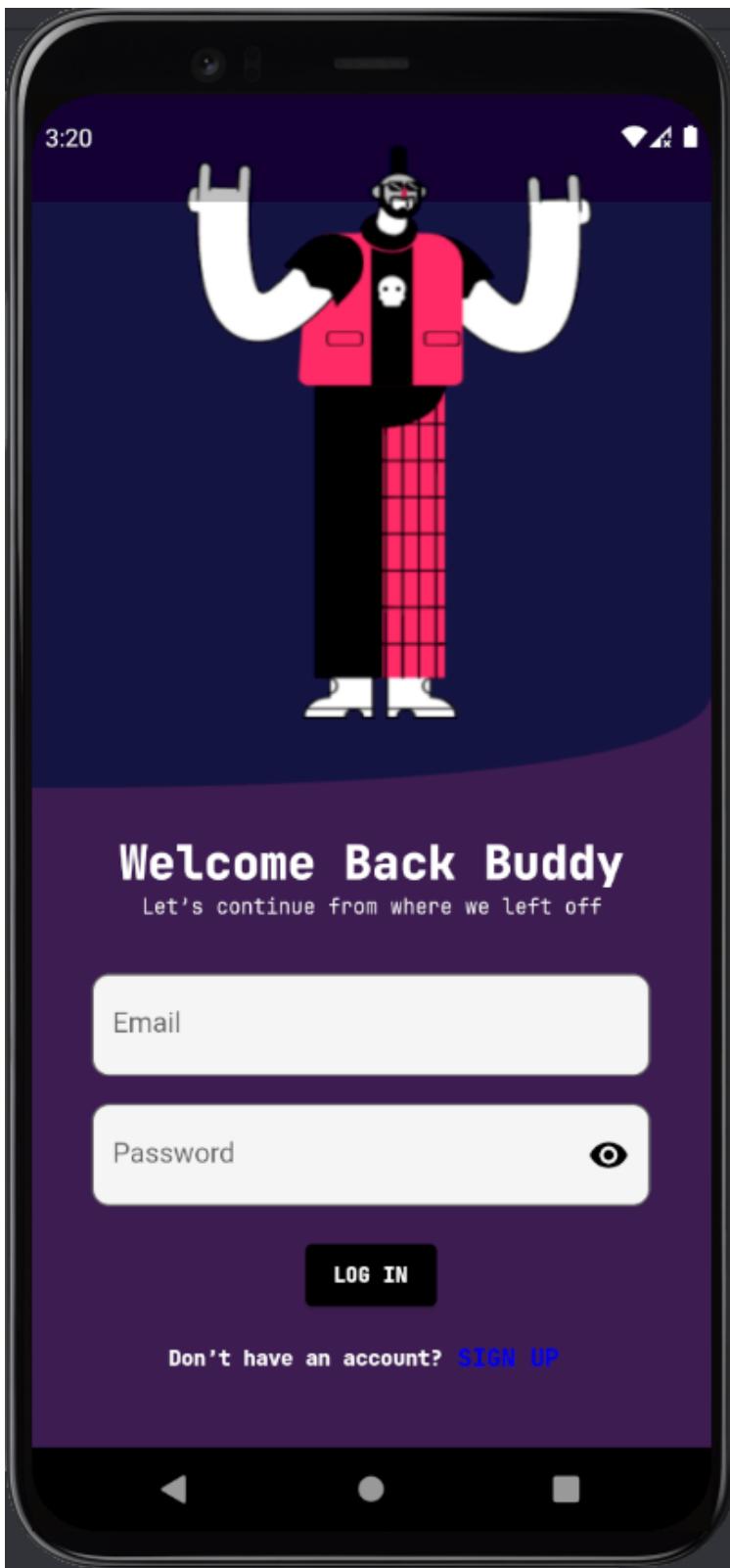


Figure 4. Login (Output)

```
lib > login.dart > ...
1 // Importing necessary packages
2 // ignore_for_file: use_build_context_synchronously
3 import 'package:flutter/material.dart';
4 import 'package:hive/hive.dart';
5 import 'package:kharcha_saathi/choose_currency.dart';
6 import 'package:kharcha_saathi/dashboard.dart';
7 import 'package:kharcha_saathi/signup.dart';
8 import 'package:http/http.dart' as http;
9 import 'dart:convert';
10
11 // This is a stateful widget that represents the Login screen.
12 class MyLogin extends StatefulWidget {
13   const MyLogin({super.key});
14
15   @override
16   State<MyLogin> createState() => _MyLoginState();
17 }
18
19 // This is the state of the MyLogin widget
20 class _MyLoginState extends State<MyLogin> {
21   final TextEditingController emailController = TextEditingController();
22   final TextEditingController passwordController = TextEditingController();
23   final GlobalKey<FormState> form = GlobalKey<FormState>();
24   bool isLoading = false;
25   bool _isObscure = true;
26   var box = Hive.box('userData');
27
28   @override
29   void dispose() {
30     emailController.dispose();
31     passwordController.dispose();
32     super.dispose();
33   }
34
35   @override
36   Widget build(BuildContext context) {
37     return Container(
38       // Set the background image of the container
39       decoration: const BoxDecoration(
40         image: DecorationImage(
41           image: AssetImage("assets/images/login.png"), fit: BoxFit.fill)), // DecorationImage // BoxDecoration
42       child: Scaffold(
43         // Make the scaffold's background transparent
44         backgroundColor: Colors.transparent,
45         body: isLoading
46           ? const Center(
47             child: CircularProgressIndicator(),
48           ) // Center
49           : SingleChildScrollView(
50             child: Stack(
51               children: [
52                 // Add the welcome text and continue message
53                 Container(
54                   padding: const EdgeInsets.only(left: 51, top: 425),
55                   child: Column(
56                     children: const [
57                       Text(
58                         "Welcome Back Buddy",
59                         style: TextStyle(
60                           color: Colors.white,
61                           fontSize: 27,
62                           fontFamily: "JetBrainsMono"), // TextStyle
63                         ), // Text
64                         Text(
65                           "Let's continue from where we left off",
66                           style: TextStyle(
67                             color: Colors.white,
68                             fontSize: 12,
69                             fontFamily: "JetBrainsMonoLight"), // TextStyle
70                           ), // Text
71                         ],
72                   ), // Column
73                 ), // Container

```

Figure 5. Login (Code 1)

```

75     Container(
76       // Set the top padding to match device height
77       padding: EdgeInsets.only(
78         top: MediaQuery.of(context).size.height * 0.65,
79         right: 35,
80         left: 35), // EdgeInsets.only
81     child: Column(
82       children: [
83         // Add the email text field
84         TextFormField(
85           autovalidateMode:
86             AutovalidateMode.onUserInteraction,
87           validator: (value) {
88             RegExp emailReg = RegExp(
89               r"^[a-zA-Z0-9.a-zA-Z0-9.!#$%&'*+=?^_`{|}~]+@[a-zA-Z0-9]+\.[a-zA-Z]+");
90             if (!emailReg.hasMatch(value.toString())) {
91               return "Invalid Email";
92             }
93             return null;
94           },
95           controller: emailController,
96           decoration: InputDecoration(
97             fillColor: Colors.grey.shade100,
98             filled: true,
99             hintText: "Email",
100            border: OutlineInputBorder(
101              borderRadius: BorderRadius.circular(10)), // OutlineInputBorder // InputDecoration
102          ), // TextFormField
103          const SizedBox(
104            height: 16,
105          ), // SizedBox
106          // Add the password text field
107          TextFormField(
108            autovalidateMode:
109              AutovalidateMode.onUserInteraction,
110            validator: (value) {
111              RegExp passReg = RegExp(
112                r"^(?=.*?[A-Z])(?=.*?[a-z])(?=.*?[0-9])(?=.*[!@#$%^&*~]).{8,}$");
113              if (!passReg.hasMatch(value.toString())) {
114                return "Invalid Password";
115              }
116              return null;
117            },
118            controller: passwordController,
119            obscureText: _isObscure,
120            decoration: InputDecoration(
121              suffixIcon: InkWell(
122                onTap: () {
123                  setState(() {
124                    _isObscure = !_isObscure;
125                  });
126                },
127                child: Icon(
128                  _isObscure
129                    ? Icons.visibility
130                    : Icons.visibility_off,
131                  color: Colors.black,
132                ), // Icon
133              ), // InkWell
134              fillColor: Colors.grey.shade100,
135              filled: true,
136              hintText: "Password",
137              border: OutlineInputBorder(
138                borderRadius: BorderRadius.circular(10),
139              ), // OutlineInputBorder
140            ), // InputDecoration
141          ), // TextFormField
142
143          const SizedBox(
144            height: 16,
145          ), // SizedBox
146          Column(
147            children: [

```

Figure 6. Login (Code 2)

```

149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
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)

```

222             child: const Text("SIGN UP",
223               style: TextStyle(
224                 fontFamily: "JetBrainsMono",
225                 fontSize: 14,
226                 color: Color(0xFF0000EE))), // TextStyle // Text // TextButton
227           ],
228         ),
229       ],
230     ],
231   ],
232   ],
233   ],
234   ],
235   ],
236   ],
237   );
238 );
239 }
240 }
241

```

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

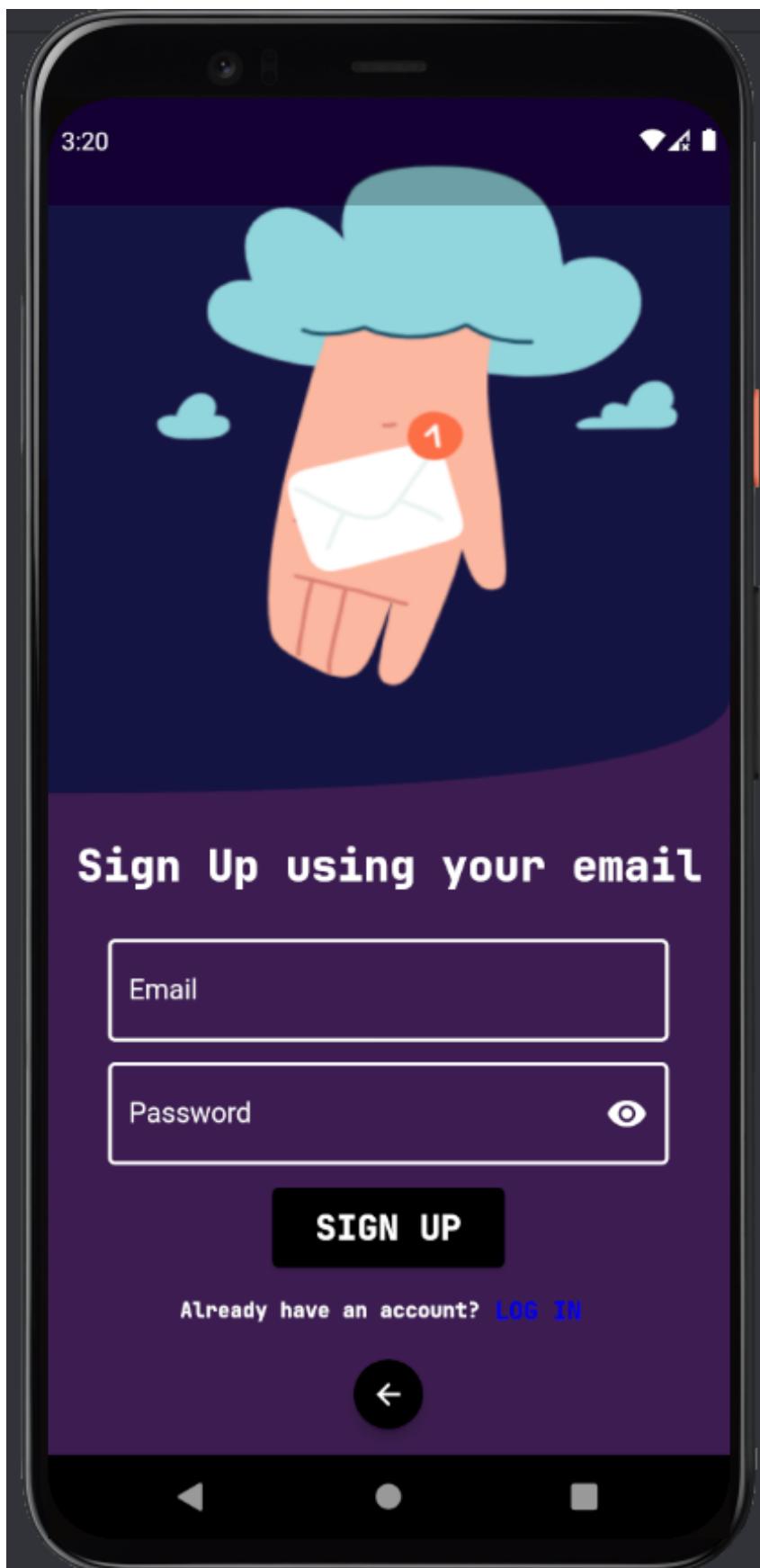


Figure 9. Sign Up Email (Output)

The screenshot shows a mobile application development environment with a code editor and various toolbars. The code editor displays Dart code for a sign-up email screen. The code includes imports for Flutter material, hive_flutter, dart:convert, kharcha_saathi/login, kharcha_saathi/signup, and http. It defines a MySignupEmail StatefulWidget and its _MySignupEmailState. The state handles TextEditingController for email and password, manages loading status, and retrieves user data from Hive. The build method creates a Scaffold with a background image, a CircularProgressIndicator if loading, and a Stack containing a Text widget and a Form with a TextFormField for email input.

```
1 // ignore_for_file: use_build_context_synchronously
2 import 'package:flutter/material.dart';
3 import 'package:hive_flutter/hive_flutter.dart';
4 import 'dart:convert';
5 import 'package:kharcha_saathi/login.dart';
6 import 'package:kharcha_saathi/signup.dart';
7 import 'package:http/http.dart' as http;
8
9 class MySignupEmail extends StatefulWidget {
10   const MySignupEmail({super.key});
11
12   @override
13   State<MySignupEmail> createState() => _MySignupEmailState();
14 }
15
16 class _MySignupEmailState extends State<MySignupEmail> {
17   final TextEditingController emailController = TextEditingController();
18   final TextEditingController passwordController = TextEditingController();
19   bool isLoading = false;
20   bool _isObscure = true;
21   var box = Hive.box('userData');
22   final GlobalKey<FormState> form = GlobalKey<FormState>();
23
24   @override
25   void dispose() {
26     emailController.dispose();
27     passwordController.dispose();
28     super.dispose();
29   }
30
31   @override
32   Widget build(BuildContext context) {
33     //The container is used to provide a background image for the page.
34     return Container(
35       decoration: const BoxDecoration(
36         image: DecorationImage(
37           image: AssetImage("assets/images/emailsignup.png"),
38           fit: BoxFit.fill)), // DecorationImage // BoxDecoration
39       child: Scaffold(
40         //The background color of the scaffold is transparent to show the background image.
41         backgroundColor: Colors.transparent,
42         body: isLoading
43             ? const Center(
44                 child: CircularProgressIndicator(),
45             ) // Center
46             : SingleChildScrollView(
47               child: SizedBox(
48                 child: Stack(
49                   children: [
50                     Container(
51                       padding: const EdgeInsets.only(left: 17, top: 425),
52                       child: const Text(
53                         "Sign Up using your email",
54                         textAlign: TextAlign.center,
55                         style: TextStyle(
56                           color: Colors.white,
57                           fontSize: 25,
58                           fontFamily: "JetBrainsMono"), // TextStyle
59                     ), // Text
60                   ), // Container
61                   Container(
62                     padding: EdgeInsets.only(
63                       top: MediaQuery.of(context).size.height * 0.62,
64                       right: 35,
65                       left: 35), // EdgeInsets.only
66                     child: Form(
67                       key: form,
68                       child: Column(
69                         children: [
70                           //This TextFormField is used to take email input from the user.
71                           TextFormField(
72                             autovalidateMode:
73                               AutovalidateMode.onUserInteraction,
```

Figure 10. Sign Up Email (Code 1)

The screenshot shows a mobile application development interface with several tabs at the top: main.dart M, splash_screen.dart M, onboarding.dart M, login.dart M, signup.dart M, signup_email.dart M X, and choose_. Below the tabs, the code for `signup_email.dart` is displayed in a code editor. The code defines two `TextFormField` widgets for email and password input. The email input has a regular expression validator for email format and a color-coded style. The password input has a regular expression validator for password complexity and a suffix icon for toggling visibility.

```
75 |           RegExp emailReg = RegExp(          //This TextFormField is used to take email input from the user.
76 |             r'^[a-zA-Z0-9.a-zA-Z0-9.!#$%&^*+=?^`{|}~]+@[a-zA-Z0-9]+\.[a-zA-Z]+'); // RegExp
77 |           if (!emailReg.hasMatch(value.toString())) {          return "Invalid Email";
78 |             }
79 |           return null;
80 |         },
81 |       ),
82 |       controller: emailController,
83 |       style: const TextStyle(
84 |         color: Color.fromARGB(255, 255, 255, 255), // TextStyle
85 |       decoration: const InputDecoration(
86 |         focusedBorder: OutlineInputBorder(
87 |           borderSide: BorderSide(
88 |             color: Color.fromARGB(
89 |               255, 255, 255, 255), // Color.fromARGB
90 |               width: 2.0), // BorderSide
91 |             ), // OutlineInputBorder
92 |             enabledBorder: OutlineInputBorder(
93 |               borderSide: BorderSide(
94 |                 color: Color.fromARGB(
95 |                   255, 255, 255, 255), // Color.fromARGB
96 |                   width: 2.0), // BorderSide
97 |                 ), // OutlineInputBorder
98 |                 labelText: "Email",
99 |                 labelStyle: TextStyle(color: Colors.white),
100 |                 hintText: 'Enter your email',
101 |                 hintStyle: TextStyle(color: Colors.white)), // InputDecoration
102 |               ), // TextFormField
103 |               const SizedBox(
104 |                 height: 12,
105 |               ), // SizedBox
106 |               //This TextFormField is used to take password input from the user.
107 |               TextFormField(
108 |                 autovalidateMode:
109 |                   AutovalidateMode.onUserInteraction,
110 |                 validator: (value) {
111 |                   //This regex is used to validate the password.
112 |                   RegExp passReg = RegExp(
113 |                     r'^(?=.*?[A-Z])(?=.*?[a-z])(?=.*?[0-9])(?=.*?[!@#\$&*~]).{8,}$'); // RegExp
114 |                     if (!passReg.hasMatch(value.toString())) {
115 |                       return "Invalid Password";
116 |                     }
117 |                     return null;
118 |                   },
119 |                   controller: passwordController,
120 |                   obscureText: _isObscure,
121 |                   style: const TextStyle(
122 |                     color: Color.fromARGB(255, 255, 255, 255), // TextStyle
123 |                   decoration: InputDecoration(
124 |                     suffixIcon: InkWell(
125 |                       onTap: () {
126 |                         setState(() {
127 |                           _isObscure = !_isObscure;
128 |                         });
129 |                       },
130 |                       child: Icon(
131 |                         _isObscure
132 |                           ? Icons.visibility
133 |                           : Icons.visibility_off,
134 |                           color: Colors.white,
135 |                         ), // Icon
136 |                         ), // InkWell
137 |                         focusedBorder: const OutlineInputBorder(
138 |                           borderSide: BorderSide(
139 |                             color: Color.fromARGB(
140 |                               255, 255, 255, 255), // Color.fromARGB
141 |                               width: 2.0), // BorderSide
142 |                             ), // OutlineInputBorder
143 |                             enabledBorder: const OutlineInputBorder(
144 |                               borderSide: BorderSide(
145 |                                 color: Color.fromARGB(
146 |                                   255, 255, 255, 255), // Color.fromARGB
147 |                                 
```

Figure 11. Sign Up Email (Code 2)

```
lib > signup_email.dart > _MySignupEmailState > build
148           width: 2.0), // BorderSide
149           ), // OutlineInputBorder
150           labelText: "Password",
151           labelStyle:
152             const TextStyle(color: Colors.white),
153             hintText: 'Enter your password',
154             hintStyle:
155               const TextStyle(color: Colors.white), // InputDecoration
156             ), // TextFormField
157             const SizedBox(
158               height: 12,
159             ), // SizedBox
160             // Creating an ElevatedButton to submit the form
161             ElevatedButton(
162               onPressed: () async {
163                 if (form.currentState!.validate()) {
164                   setState(() {
165                     isLoading = true;
166                   });
167
168                   http.Response response = await http.post(
169                     Uri.parse(
170                       // Sending a POST request to create a new user
171                       "https://kharchasathi-backend.onrender.com/users"),
172                     headers: {
173                       "Authorization":
174                         "Bearer ${box.get("accessToken")}"
175                     },
176                     body: json.encode({
177                       "email": emailController.text,
178                       "password": passwordController.text
179                     }));
180
181                   var jsonResponse =
182                     json.decode(response.body);
183                   if (response.statusCode == 200) {
184                     // Displaying a success message and navigating to the Login screen
185                     ScaffoldMessenger.of(context)
186                       .showSnackBar(SnackBar(
187                         content: Text(jsonResponse["message"]),
188                       )); // SnackBar
189
190                     Navigator.of(context).pushReplacement(
191                       MaterialPageRoute(
192                         builder: (context) =>
193                           const MyLogin()), // MaterialPageRoute
194                       );
195                   } else {
196                     setState(() {
197                       isLoading = false;
198                     });
199                     ScaffoldMessenger.of(context)
200                       .showSnackBar(SnackBar(
201                         content: Text(jsonResponse["error"]),
202                       )); // SnackBar
203                   }
204                 }
205               },
206               // Styling the ElevatedButton
207               style: ElevatedButton.styleFrom(
208                 backgroundColor:
209                   const Color.fromARGB(255, 0, 0, 0),
210                 padding: const EdgeInsets.symmetric(
211                   horizontal: 25, vertical: 10), // EdgeInsets.symmetric
212                 textStyle: const TextStyle(
213                   fontSize: 20,
214                   fontFamily: "JetBrainsMono")), // TextStyle
215                 child: const Text('SIGN UP'),
216               ), // ElevatedButton
217               // Creating a row with a Text and a TextButton to navigate to the Login screen
218               Row(
219                 mainAxisAlignment: MainAxisAlignment.center,
220                 children: [
221                   Text("Don't have an account?"),
222                   TextButton(
223                     onPressed: () {
224                       Navigator.pushNamed(context, "/login");
225                     },
226                     child: Text("Log In"),
227                   ),
228                 ],
229               ),
230             ],
231           ),
232         ],
233       ),
234     ],
235   ),
236 
```

Figure 12. Sign Up Email (Code 3)

The screenshot shows a mobile application development environment with several tabs at the top: main.dart, splash_screen.dart, onboarding.dart, login.dart, signup.dart, and signup_email.dart. The current file is signup_email.dart. The code is a Dart file containing the implementation for the sign-up email screen. It uses the Flutter framework's UI components like Scaffold, Column, Form, and TextFormField. The code includes logic for handling user input, validating email and password, and displaying success or error messages via SnackBar. It also handles navigation between screens using Navigator.pushReplacement.

```
110: > 111:     "Already have an account?",  
112:     style: TextStyle(  
113:         color: Colors.white,  
114:         fontSize: 12,  
115:         fontFamily: "JetBrainsMono"), // TextStyle  
116:     ), // Text  
117:     TextButton(  
118:         onPressed: () {  
119:             ScaffoldMessenger.of(context)  
120:                 .showSnackBar(const SnackBar(  
121:                     content: Text(  
122:                         "Account created successfully!"), // Text  
123:                 )); // SnackBar  
124:             Navigator.of(context).pushReplacement(  
125:                 MaterialPageRoute(  
126:                     builder: (context) =>  
127:                         const MyLogin(), // MaterialPageRoute  
128:                 ),  
129:             ),  
130:             child: const Text(  
131:                 "LOG IN",  
132:                 style: TextStyle(  
133:                     fontFamily: "JetBrainsMono",  
134:                     fontSize: 14,  
135:                     color: Color(0xff0000EE),  
136:                 ), // TextStyle  
137:             ), // Text  
138:         ), // TextButton  
139:     ],  
140: ), // Row  
141: // Creating a column with a FloatingActionButton to go back to the previous screen  
142: Column(  
143:     children: [  
144:         FloatingActionButton.small(  
145:             backgroundColor:  
146:                 const Color.fromARGB(255, 0, 0, 0),  
147:             onPressed: () {  
148:                 Navigator.of(context).pushReplacement(  
149:                     MaterialPageRoute(  
150:                         builder: (context) =>  
151:                             const Mysignup(), // MaterialPageRoute  
152:                     ),  
153:                 );  
154:             },  
155:             child: const Icon(  
156:                 Icons.arrow_back,  
157:                 size: 20,  
158:                 color: Color.fromARGB(255, 255, 255, 255),  
159:             ), // Icon  
160:         ), // FloatingActionButton.small  
161:     ],  
162: ), // Column  
163:     ), // Column  
164:     ), // Form  
165:     ), // Container  
166:     ],  
167: ), // Stack  
168: ), // SizedBox  
169: ), // SingleChildScrollView  
170: ), // Scaffold  
171: ); // Container  
172: }  
173: }  
174: }  
175: }  
176: }  
177: }  
178: }  
179: }  
180: }  
181: }  
182: }  
183: }  
184: }  
185: }
```

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

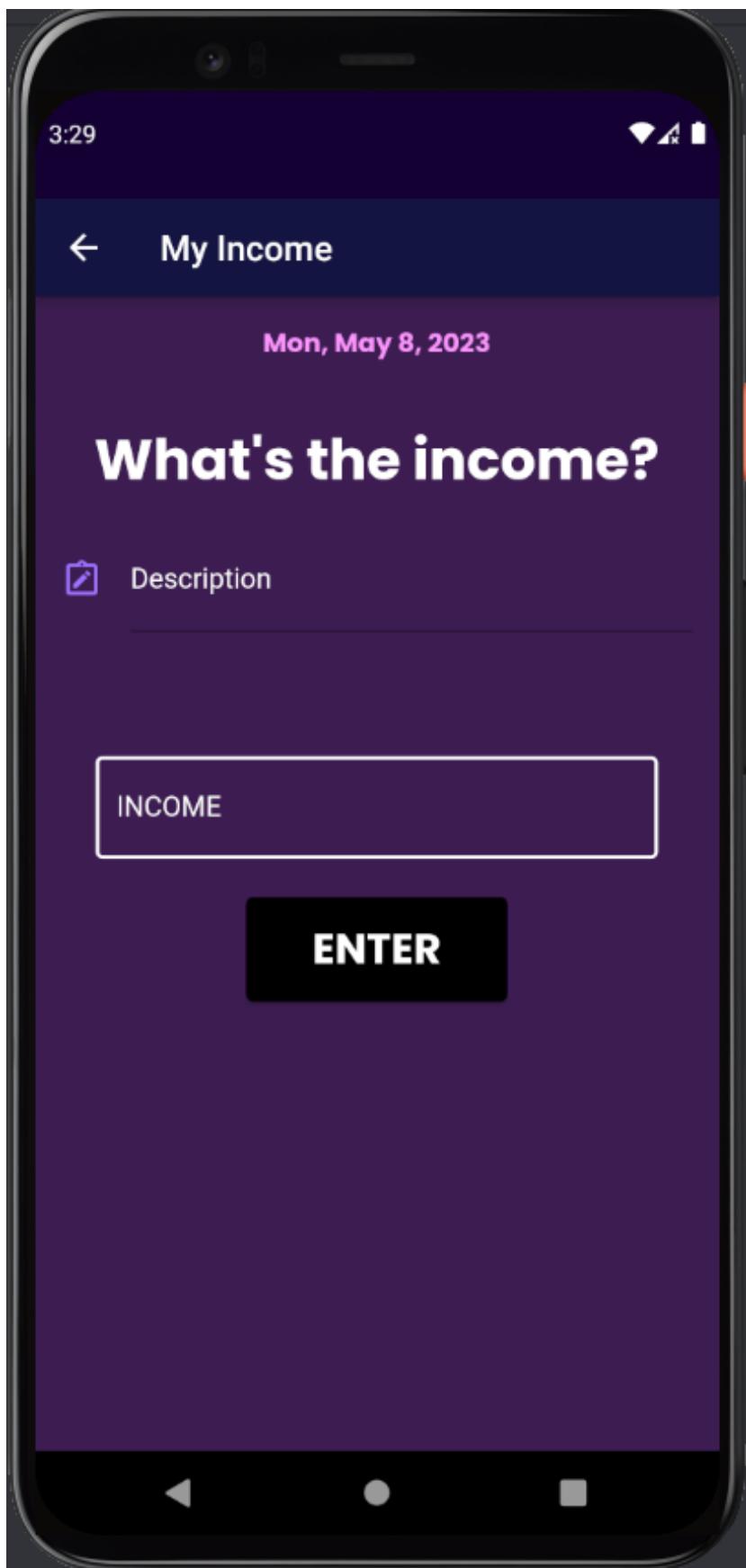


Figure 14. Income (Output)

The screenshot shows a code editor with multiple tabs at the top: main.dart M, splash_screen.dart M, onboarding.dart M, login.dart M, and signup.dart M. The main.dart tab is active. The code in the editor is as follows:

```
lib > income.dart > _MyIncomeState > dispose
1 // ignore_for_file: use_build_context_synchronously
2 // Importing necessary packages
3 import 'package:flutter/material.dart';
4 import 'package:hive/hive.dart';
5 import 'package:intl/intl.dart';
6 import 'package:kharcha_saathi/dashboard.dart';
7 import 'dart:convert';
8 import 'package:http/http.dart' as http;
9
10 // Define a widget called MyIncome and make it a StatefulWidget.
11 class MyIncome extends StatefulWidget {
12   const MyIncome({super.key});
13
14   // Override createState() to create an instance of _MyIncomeState.
15   @override
16   State<MyIncome> createState() => _MyIncomeState();
17 }
18
19 // Define a widget called _MyIncomeState and make it a State<MyIncome>.
20 class _MyIncomeState extends State<MyIncome> {
21   // Define noteController and incomeController as TextEditingController.
22   final TextEditingController noteController = TextEditingController();
23   final TextEditingController incomeController = TextEditingController();
24   // Define isLoading as bool and box as a Hive box.
25   bool isLoading = false;
26   var box = Hive.box('userData');
27   // Dispose of all controllers.
28   @override
29   void dispose() {
30     noteController.dispose();
31     incomeController.dispose();
32     super.dispose();
33   }
34
35   @override
36   Widget build(BuildContext context) {
37     // Get accessToken from Hive box.
38     String accessToken = box.get("accessToken");
39     // Create a scaffold with backgroundColor, appBar, and body widgets.
40     return Scaffold(
41       backgroundColor: const Color.fromARGB(255, 61, 31, 82),
42       appBar: AppBar(
43         backgroundColor: const Color(0xFF161341),
44         leading: BackButton(
45           onPressed: () => Navigator.of(context).pushReplacement(
46             MaterialPageRoute(
47               builder: (context) => MyDashboard(
48                 accessToken: accessToken,
49               ), // MyDashboard // MaterialPageRoute
50             ), // BackButton
51             title: const Text('My Income'),
52           ), // AppBar
53           // If isLoading is true, show a centered CircularProgressIndicator widget.
54           body: isLoading
55             ? const Center(
56               child: CircularProgressIndicator(),
57             ) // Center
58             // Otherwise, show a Column widget with various child widgets.
59             : Column(
60               children: [
61                 // Add Padding widget with EdgeInsets to create some space.
62                 const Padding(padding: EdgeInsets.fromLTRB(100, 15, 0, 0)),
63                 // Show today's date using DateFormat and TextStyle.
64                 Text(
65                   DateFormat.yMMMd().format(DateTime.now()),
66                   style: const TextStyle(
67                     fontFamily: "PoppinsBold",
68                     fontSize: 15,
69                     color: const Color(0xFFFFB94FD)), // TextStyle
70                 ), // Text
71                 const SizedBox(
72                   height: 32,
73                 ), // SizedBox
74               ],
75             ),
76           );
77         );
78       );
79     );
80   }
81 }
```

Figure 15. Input (Code 1)

The screenshot shows a code editor with several tabs at the top: main.dart, splash_screen.dart, onboarding.dart, login.dart, signup.dart, and signup_email.dart. The current file is income.dart. The code is part of a class named '_MyIncomeState' and is building a UI for entering income. It includes a Text widget for the question "What's the income?", a SingleChildScrollView containing a Column with a Padding and a TextField for the income note description, and another Column with a Padding and a TextField for the income amount. The income amount TextField has a focusedBorder and enabledBorder defined with OutlineInputBorder and borderSide properties.

```
const Text(
    "What's the income?",
    style: TextStyle(
        fontFamily: "PoppinsBold",
        fontSize: 32,
        color: Color.fromARGB(255, 255, 255, 255)), // TextStyle
    textAlign: TextAlign.center,
),
// Text
// Create a SingleChildScrollView widget to allow for scrolling
// within the Column widget.
SingleChildScrollView(
    child: Column(
        children: [
            // Add a TextField widget for the income note description
            Padding(
                padding: const EdgeInsets.all(15),
                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
            Padding(
                padding: EdgeInsets.only(
                    top: MediaQuery.of(context).size.height * 0.001,
                    right: 35,
                    left: 35), // EdgeInsets.only
                child: TextField(
                    controller: incomeController,
                    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: "INCOME",
                        labelStyle: TextStyle(
                            color: Colors.white,
                        ), // TextStyle
                        hintText: 'Enter your income',
                        hintStyle: TextStyle(color: Colors.white)), // InputDecoration
                ), // TextField
            ), // Padding
            const SizedBox(
                height: 22,
            ), // SizedBox
            // Add an ElevatedButton widget for entering the income.
            ElevatedButton(
                onPressed: () async {
                    setState(() {
                        isLoading = true;
                    });
                },
            ), // Try parsing the income amount as an int.
        ],
    ),
)
```

Figure 16. Input (Code 2)

```

lib > income.dart > _MyIncomeState > build
147 |   // Try parsing the income amount as an int.
148 |   int? amount = int.tryParse(incomeController.text);
149 |   // If the amount is null, show a SnackBar and set isLoading to false.
150 |   if (amount == null) {
151 |     ScaffoldMessenger.of(context)
152 |       .showSnackBar(const SnackBar(
153 |         content: Text("Invalid balance value."),
154 |       )); // SnackBar
155 |     setState(() {
156 |       isLoading = false;
157 |     });
158 |   }
159 |
160 |   // Send a PATCH request to the API with the income note description and amount.
161 |   http.Response response = await http.patch(
162 |     Uri.parse(
163 |       "https://kharchasathi-backend.onrender.com/users/income"),
164 |     headers: {
165 |       "Authorization": 'Bearer $accessToken'
166 |     },
167 |     body: json.encode({
168 |       "description": noteController.text,
169 |       "amount": amount,
170 |     }));
171 |
172 |   var jsonResponse = json.decode(response.body);
173 |   // If the response status code is 200, show a SnackBar with the message and navigate to MyDashboard.
174 |   if (response.statusCode == 200) {
175 |     ScaffoldMessenger.of(context)
176 |       .showSnackBar(SnackBar(
177 |         content: Text(jsonResponse["message"]),
178 |       )); // SnackBar
179 |     Navigator.of(context).pushReplacement(
180 |       MaterialPageRoute(
181 |         builder: (context) => MyDashboard(
182 |           accessToken: accessToken,
183 |         ), // MyDashboard // MaterialPageRoute
184 |       );
185 |     }
186 |   } // Otherwise, show a SnackBar with the error message and set isLoading to false.
187 |   else {
188 |     setState(() {
189 |       isLoading = false;
190 |     });
191 |     ScaffoldMessenger.of(context)
192 |       .showSnackBar(SnackBar(
193 |         content: Text(jsonResponse["error"]),
194 |       )); // SnackBar
195 |   }
196 | },
197 | style: ElevatedButton.styleFrom(
198 |   backgroundColor: Colors.black,
199 |   fixedSize: const Size(150, 60)),
200 | child: const Text(
201 |   "ENTER",
202 |   style: TextStyle(
203 |     fontFamily: "PoppinsBold",
204 |     color: Colors.white,
205 |     fontSize: 24), // TextStyle
206 |   )) // Text // ElevatedButton
207 | ],
208 | ), // Column
209 | ), // SingleChildScrollView
210 | ],
211 | ), // Column
212 | ); // Scaffold
213 |
214 }

```

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

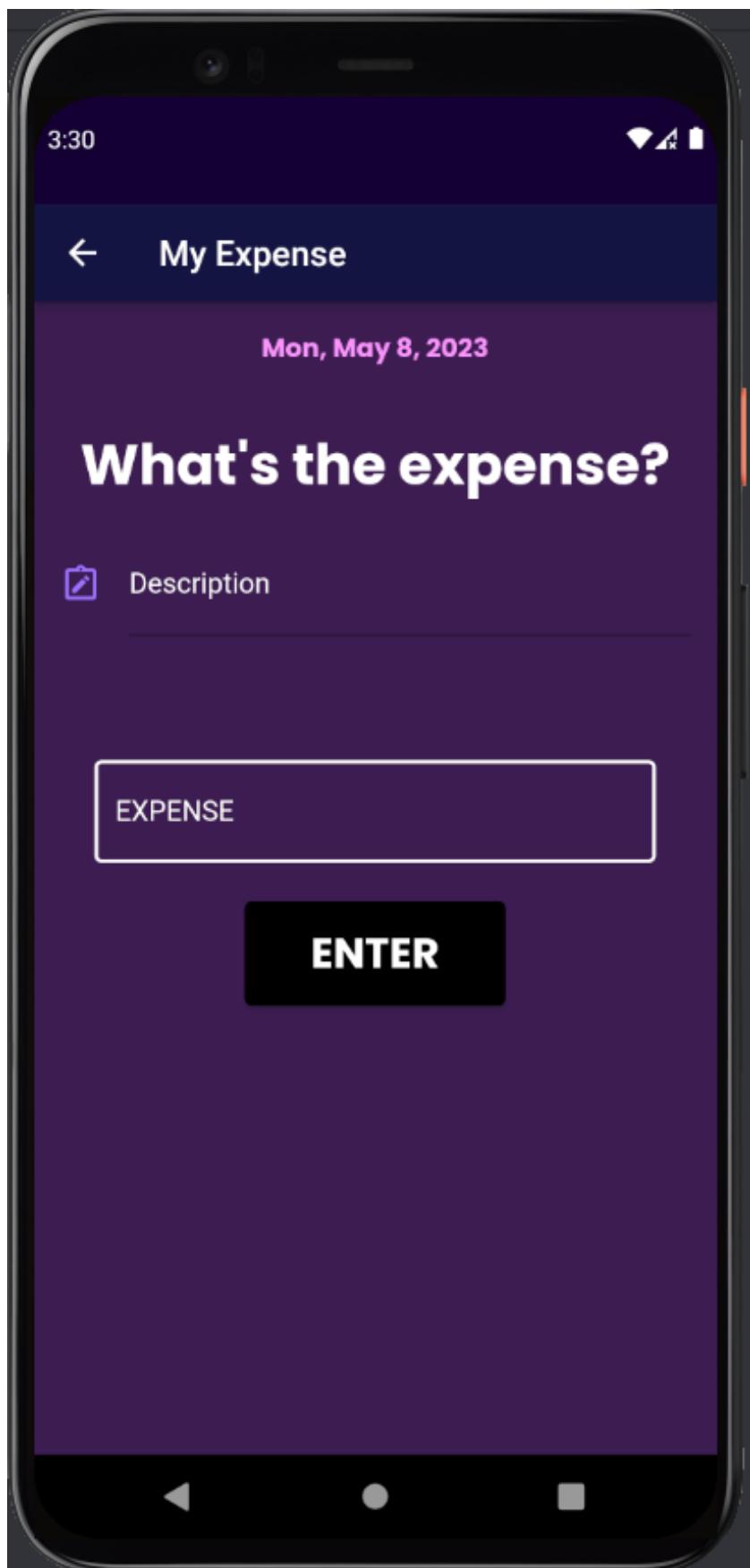


Figure 18. Expense (Output)

```

lib > expense.dart > _MyexpenseState
1 // ignore_for_file: use_build_context_synchronously
2 // Import necessary packages.
3 import 'package:flutter/material.dart';
4 import 'package:hive/hive.dart';
5 import 'package:intl/intl.dart';
6 import 'package:kharcha_saathi/dashboard.dart';
7 import 'dart:convert';
8 import 'package:http/http.dart' as http;
9
10 // Create a StatefulWidget for Myexpense.
11 class Myexpense extends StatefulWidget {
12   const Myexpense({super.key});
13
14   @override
15   State<Myexpense> createState() => _MyexpenseState();
16 }
17
18 // Create the state for Myexpense.
19 class _MyexpenseState extends State<Myexpense> {
20   // Create controllers for note and expense text fields.
21   final TextEditingController noteController = TextEditingController();
22   final TextEditingController expenseController = TextEditingController();
23   // Get the user data box from Hive.
24   var box = Hive.box('userData');
25   // Create a isLoading boolean to track whether the data is Loading or not.
26   bool isLoading = false;
27
28   @override
29   void dispose() {
30     noteController.dispose();
31     expenseController.dispose();
32     super.dispose();
33   }
34
35   @override
36   Widget build(BuildContext context) {
37     // Get the access token from the user data box.
38     String accessToken = box.get("accessToken");
39     return Scaffold(
40       backgroundColor: const Color.fromARGB(255, 61, 31, 82),
41       appBar: AppBar(
42         backgroundColor: const Color(0xFF161341),
43         leading: BackButton(
44           onPressed: () => Navigator.of(context).pushReplacement(
45             MaterialPageRoute(builder: (context) => MyDashboard(accessToken: accessToken)),
46           ), // BackButton
47           title: const Text('My Expense'),
48         ), // AppBar
49       // If data is Loading, show a CircularProgressIndicator, otherwise, show the expense input form.
50       body: isLoading
51         ? const Center(
52             child: CircularProgressIndicator(),
53           ) // Center
54         : Column(
55           children: [
56             const Padding(padding: EdgeInsets.fromLTRB(1000, 15, 0, 0)),
57             Text(
58               DateFormat.yMMMd().format(DateTime.now()),
59               style: const TextStyle(
60                 fontFamily: "PoppinsBold",
61                 fontSize: 15,
62                 color: Color(0xFFFFB94FD)), // TextStyle
63             ), // Text
64             const SizedBox(
65               height: 32,
66             ), // SizedBox
67             const Text(
68               "What's the expense?",
69               style: TextStyle(
70                 fontFamily: "PoppinsBold",
71                 fontSize: 32,
72                 color: Color.fromARGB(255, 255, 255, 255)), // TextStyle
73               textAlign: TextAlign.center,

```

Figure 19. Expense (Code 1)

```

75     // Create a Column
76     Column(
77       children: [
78         Padding(
79           padding: const EdgeInsets.all(15),
80           // Create a text input field for the expense note.
81           child: TextField(
82             controller: noteController,
83             style: const TextStyle(color: Colors.white),
84             decoration: const InputDecoration(
85               labelText: "Description",
86               labelStyle: TextStyle(color: Colors.white),
87               icon: Icon(
88                 Icons.note_alt_outlined,
89                 color: Color(0xFF9F72FF),
90               ), // icon at head of input // Icon
91             ), // InputDecoration
92           ), // TextField
93         ), // Padding
94         const SizedBox(
95           height: 32,
96         ), // SizedBox
97         const SizedBox(
98           height: 24,
99         ), // SizedBox
100        // Create a SingleChildScrollView to ensure the text input field is scrollable.
101        SingleChildScrollView(
102          child: Padding(
103            padding: EdgeInsets.only(
104              top: MediaQuery.of(context).size.height * 0.001,
105              right: 35,
106              left: 35), // EdgeInsets.only
107            // Create a text input field for the expense amount.
108            child: TextField(
109              controller: expenseController,
110              // Create input decoration for the text input field.
111              style: const TextStyle(
112                color: Color.fromRGBO(255, 255, 255, 255)), // TextStyle
113              decoration: const InputDecoration(
114                focusedBorder: OutlineInputBorder(
115                  borderSide: BorderSide(
116                    color: Color.fromRGBO(255, 255, 255, 255),
117                    width: 2.0), // BorderSide
118                ), // OutlineInputBorder
119                enabledBorder: OutlineInputBorder(
120                  borderSide: BorderSide(
121                    color: Color.fromRGBO(255, 255, 255, 255),
122                    width: 2.0), // BorderSide
123                ), // OutlineInputBorder
124                labelText: "EXPENSE",
125                labelStyle: TextStyle(
126                  color: Colors.white,
127                ), // TextStyle
128                hintText: 'Enter your expense',
129                hintStyle: TextStyle(color: Colors.white), // InputDecoration
130              ), // TextField
131            ), // Padding
132          ), // SingleChildScrollView
133          const SizedBox(
134            height: 22,
135          ), // SizedBox
136          // Create an ElevatedButton for submitting the expense.
137          ElevatedButton(
138            onPressed: () async {
139              setState(() {
140                isLoading = true;
141              });
142              // Parse the expense amount from the text input field.
143              int? amount = int.tryParse(expenseController.text);
144              if (amount == null) {
145                ScaffoldMessenger.of(context)
146                  .showSnackBar(const SnackBar(
147                    content: Text("Invalid balance value."),

```

Figure 20. Expense (Code 2)

```

148          });
149          setState(() {
150              isLoading = false;
151          });
152          return;
153      }
154      // Send a PATCH request to the backend API to add the expense.
155      http.Response response = await http.patch(
156          Uri.parse(
157              "https://kharchasathi-backend.onrender.com/users/expense"),
158          headers: {
159              "Authorization":
160                  "Bearer $accessToken"
161          },
162          body: json.encode({
163              "description": noteController.text,
164              "amount": amount,
165          }));
166
167      var jsonResponse = json.decode(response.body);
168      if (response.statusCode == 200) {
169          // Show a success message and navigate back to the dashboard.
170          ScaffoldMessenger.of(context).showSnackBar(SnackBar(
171              content: Text(jsonResponse["message"]),
172          )); // SnackBar
173          Navigator.of(context).pushReplacement(
174              MaterialPageRoute(
175                  builder: (context) => MyDashboard(token: accessToken)), // MaterialPageRoute
176          );
177      } else {
178          // Show an error message if the request fails.
179          setState(() {
180              isLoading = false;
181          });
182          ScaffoldMessenger.of(context).showSnackBar(SnackBar(
183              content: Text(jsonResponse["error"]),
184          )); // SnackBar
185      }
186  },
187  style: ElevatedButton.styleFrom(
188      backgroundColor: Colors.black,
189      fixedSize: const Size(150, 60)),
190  child: const Text(
191      "ENTER",
192      style: TextStyle(
193          fontFamily: "PoppinsBold",
194          color: Colors.white,
195          fontSize: 24), // TextStyle
196      )) // Text // ElevatedButton
197  ],
198  ),
199  ],
200  ), // Column
201  ), // Column
202 ); // Scaffold
203 }
204

```

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

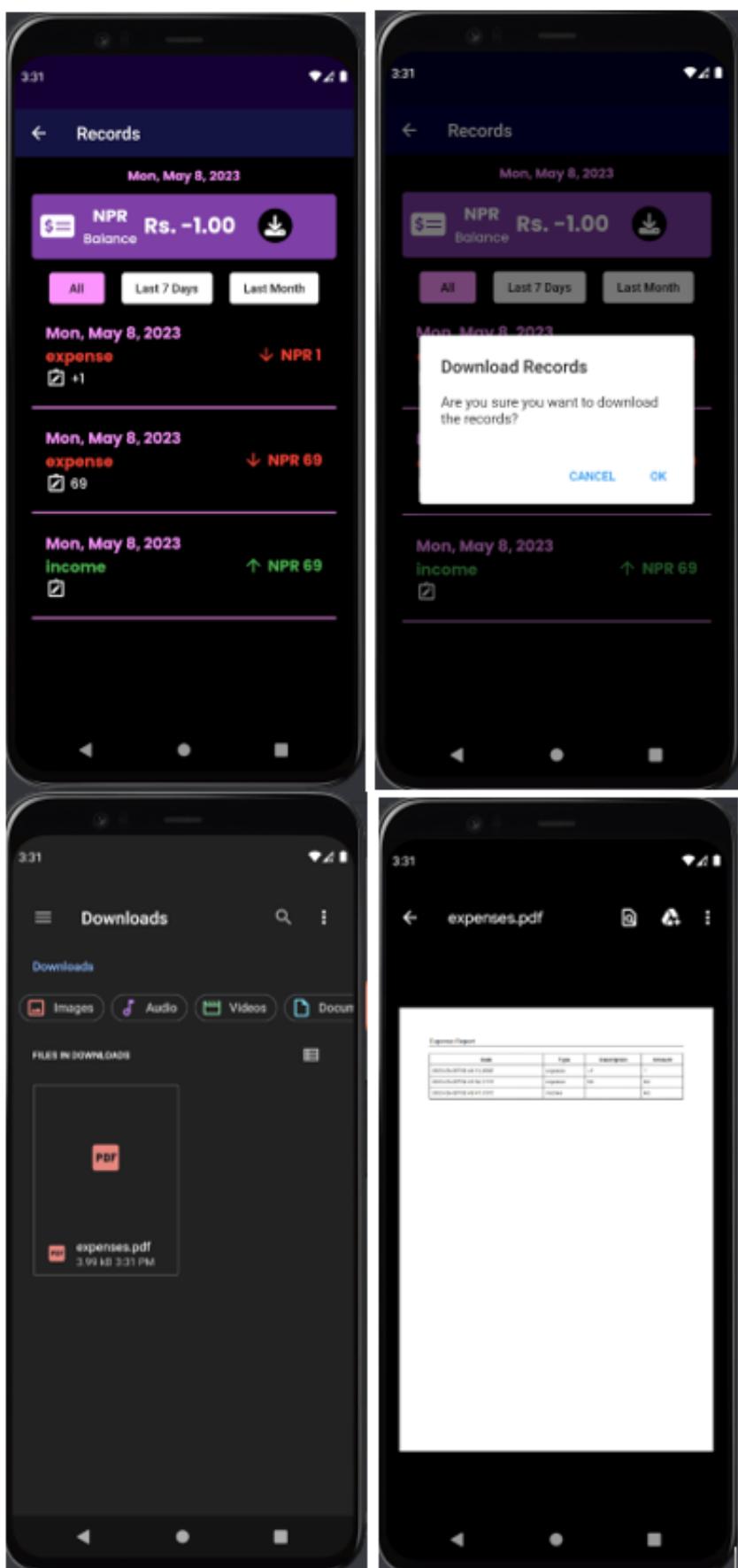


Figure 22. Records (Output)

```
lib > records.dart > _MyRecordsState
1 // ignore_for_file: prefer_typing_uninitialized_variables, use_build_context_synchronously
2 // Import necessary packages.
3 import 'package:dio/dio.dart';
4 import 'dart:io';
5 import 'package:flutter/material.dart';
6 import 'package:font_awesome_flutter/font_awesome_flutter.dart';
7 import 'package:hive_flutter/hive_flutter.dart';
8 import 'package:intl/intl.dart';
9 import 'package:pdf/widgets.dart' as pw;
10
11 class MyRecords extends StatefulWidget {
12   const MyRecords({super.key});
13
14   @override
15   State<MyRecords> createState() => _MyRecordsState();
16 }
17
18 class _MyRecordsState extends State<MyRecords> {
19   // Initialize Hive box and variables
20   var box = Hive.box('userData');
21   var jsonList;
22   int jsonBalance = 0;
23   bool _showAllTransactions = true;
24   bool _showLastWeekTransactions = false;
25   bool _showLastMonthTransactions = false;
26   int itemCount = 0;
27   bool isLoading = false;
28
29   // Get user balance from API
30   void getBalance() async {
31     setState(() {
32       isLoading = true;
33     });
34
35     Dio dio = Dio();
36
37     dio.options.headers = {
38       'Authorization': 'Bearer ${box.get("accessToken")}',
39     };
34
40     try {
41       Response response = await dio
42         .get('https://kharchasathi-backend.onrender.com/users/balance');
43       if (response.statusCode == 200) {
44         setState(() {
45           int balance = response.data["balance"] as int;
46           jsonBalance = balance;
47           isLoading = false; // set isLoading back to false
48         });
49       } else {
50         isLoading = false;
51         ScaffoldMessenger.of(context).showSnackBar(SnackBar(
52           content: Text('Error: ${response.statusCode}'),
53         )); // SnackBar
54       }
55     } catch (e) {
56       isLoading = false;
57       ScaffoldMessenger.of(context).showSnackBar(SnackBar(
58         content: Text('Error: $e'),
59       )); // SnackBar
60     }
61   }
62
63
64   // Get transactions from API
65   void getTransaction() async {
66     setState(() {
67       isLoading = true;
68     });
69
70     Dio dio = Dio();
71
72     dio.options.headers = {
73       'Authorization': 'Bearer ${box.get("accessToken")}',
```

Figure 23. Records (Code 1)

```

76    try {
77      Response response = await dio
78        .get('https://kharchasathi-backend.onrender.com/transaction');
79      if (response.statusCode == 200) {
80        setState(() {
81          jsonList = response.data["transactions"] as List;
82          isLoading = false;
83        });
84      } else {
85        isLoading = false;
86        ScaffoldMessenger.of(context).showSnackBar(SnackBar(
87          content: Text('Error: ${response.statusCode}'),
88        )); // SnackBar
89      }
90    } catch (e) {
91      isLoading = false;
92      ScaffoldMessenger.of(context).showSnackBar(SnackBar(
93        content: Text('Error: $e'),
94      )); // SnackBar
95    }
96  }
97
98  @override
99  void initState() {
100   getBalance();
101   getTransaction();
102   super.initState();
103 }
104
105 @override
106 Widget build(BuildContext context) {
107   // Determine the number of transactions to display based on filters
108   if (_showLastWeekTransactions) {
109     // Filter transactions for the last 7 days
110     DateTime now = DateTime.now();
111     DateTime lastWeek = now.subtract(const Duration(days: 7));
112     itemCount = jsonList
113       .where((transaction) =>
114         DateTime.parse(transaction['date']).isAfter(lastWeek))
115       .length;
116   } else if (_showLastMonthTransactions) {
117     // Filter transactions for the last month
118     DateTime now = DateTime.now();
119     DateTime lastMonth = DateTime(now.year, now.month - 1, now.day);
120     itemCount = jsonList
121       .where((transaction) =>
122         DateTime.parse(transaction['date']).isAfter(lastMonth))
123       .length;
124   } else {
125     itemCount = jsonList?.length ?? 0;
126   }
127
128   return Scaffold(
129     // Set the background color to black
130     backgroundColor: Colors.black,
131     appBar: AppBar(
132       // Set the app bar background color to a dark blue
133       backgroundColor: const Color(0xFF161341),
134       title: const Text('Records'),
135     ), // AppBar
136     body: SafeArea(
137       child: Padding(
138         padding: const EdgeInsets.symmetric(horizontal: 20, vertical: 10),
139         child: Column(
140           children: [
141             // Show today's date in a specific format
142             Text(
143               DateFormat.yMMEd().format(DateTime.now()),
144               style: const TextStyle(
145                 fontFamily: "PoppinsBold",
146                 fontSize: 15,
147                 color: Color(0xFFFFB94FD),
148               ), // TextStyle

```

Figure 24. Records (Code 2)

```

158         const SizedBox(
159             height: 10,
160         ), // SizedBox
161     Container(
162         height: 75,
163         width: 400,
164         decoration: BoxDecoration(
165             borderRadius: BorderRadius.circular(5),
166             color: const Color(0xFF7e3fa7),
167         ), // BoxDecoration
168         child: Padding(
169             padding:
170                 const EdgeInsets.symmetric(horizontal: 10, vertical: 10),
171             child: Column(
172                 children: [
173                     Row(
174                         children: [
175                             // Show a money icon
176                             const Icon(
177                                 FontAwesomeIcons.moneyCheckDollar,
178                                 color: Colors.white,
179                                 size: 35,
180                             ), // Icon
181                             const SizedBox(
182                                 width: 15,
183                             ), // SizedBox
184                             Column(
185                                 children: const [
186                                     // Show "NPR" and "Balance" text
187                                     Text(
188                                         "NPR",
189                                         style: TextStyle(
190                                             fontFamily: "PoppinsBold",
191                                             fontSize: 20,
192                                             color: Colors.white), // TextStyle
193                                         ), // Text
194                                         Text(
195                                             "Balance",
196                                             style: TextStyle(
197                                                 fontFamily: "PoppinsMedium",
198                                                 fontSize: 15,
199                                                 color: Colors.white), // TextStyle
200                                         ), // Text
201                                     ],
202                                 ), // Column
203                                 Row(
204                                     children: [
205                                         const SizedBox(
206                                         width: 8,
207                                         ), // SizedBox
208                                         isLoading
209                                         ? const Center(
210                                             child: CircularProgressIndicator(),
211                                         ) // Center
212                                         : Text(
213                                             // Show the current balance
214                                             'Rs. ${jsonBalance.toStringAsFixed(2)}',
215                                             style: const TextStyle(
216                                                 fontFamily: "PoppinsBold",
217                                                 fontSize: 25,
218                                                 color: Colors.white), // TextStyle
219                                         ), // Text
220                                         const SizedBox(
221                                         width: 25,
222                                         ), // SizedBox
223                                         InkWell(
224                                             onTap: () {
225                                                 // Show a dialog to download records
226                                                 showDialog(
227                                                     context: context,
228                                                     builder: (context) => AlertDialog(
229                                                         title: const Text('Download Records'),
230                                                         content: const Text(

```

Figure 25. Records (Code 3)

```

223     'Are you sure you want to download the records? ', // Text
224     actions: [
225       TextButton(
226         onPressed: () =>
227           Navigator.pop(context, false),
228         child: const Text('CANCEL'),
229       ), // TextButton
230       TextButton(
231         onPressed: () async {
232           Navigator.pop(context, true);
233           ScaffoldMessenger.of(context)
234             .showSnackBar(
235               const SnackBar(
236                 content: Text(
237                   "Records have been downloaded successfully!"), // Text
238                 ), // SnackBar
239               );
240             await generatePDFReport(jsonList,
241               '/storage/emulated/0/Download/expenses.pdf');
242           },
243           child: const Text('OK'),
244         ), // TextButton
245       ],
246     ), // AlertDialog
247     .then((value) {
248       if (value == true) {}
249     });
250   },
251   child: const CircleAvatar(
252     radius: 20,
253     backgroundColor: Colors.black,
254     child: Icon(
255       FontAwesomeIcons.download,
256       color: Colors.white,
257     ), // Icon
258   ), // CircleAvatar
259 ) // InkWell
260 ],
261 ) // Row
262 ],
263 ), // Row
264 ],
265 ), // Column
266 ), // Padding
267 ), // Container
268 const SizedBox(
269   height: 10,
270 ), // SizedBox
271 // Create three buttons to filter the records
272 Row(
273   mainAxisAlignment: MainAxisAlignment.spaceEvenly,
274   children: [
275     ElevatedButton(
276       onPressed: () {
277         setState(() {
278           _showAllTransactions = true;
279           _showLastWeekTransactions = false;
280           _showLastMonthTransactions = false;
281         });
282       },
283       style: ElevatedButton.styleFrom(
284         backgroundColor: _showAllTransactions
285           ? const Color(0xFFFFB94FD)
286           : Colors.white,
287       ),
288       child: const Text(
289         'All',
290         style: TextStyle(color: Colors.black),
291       ), // Text
292     ), // ElevatedButton
293     ElevatedButton(
294       onPressed: () {
295         setState(() {

```

Figure 26. Records (Code 4)

```

296         _showAllTransactions = false;
297         _showLastWeekTransactions = true;
298         _showLastMonthTransactions = false;
299     });
300 },
301 style: ElevatedButton.styleFrom(
302     backgroundColor: _showLastWeekTransactions
303         ? const Color(0xFFFFB94FD)
304         : Colors.white,
305 ),
306 child: const Text(
307     'Last 7 Days',
308     style: TextStyle(color: Colors.black),
309 ), // Text
310 ), // ElevatedButton
311 ElevatedButton(
312     onPressed: () {
313         setState(() {
314             _showAllTransactions = false;
315             _showLastWeekTransactions = false;
316             _showLastMonthTransactions = true;
317         });
318     },
319     style: ElevatedButton.styleFrom(
320         backgroundColor: _showLastMonthTransactions
321             ? const Color(0xFFFFB94FD)
322             : Colors.white,
323     ),
324     child: const Text(
325         'Last Month',
326         style: TextStyle(color: Colors.black),
327     ), // Text
328 ), // ElevatedButton
329 ],
330 ), // Row
331 const SizedBox(
332     height: 10,
333 ), // SizedBox
334 // If isLoading is true, shows a Center widget containing a CircularProgressIndicator.
335 // If isLoading is false, shows an Expanded widget containing a ListView.builder.
336 isLoading
337     ? const Center(
338         child: CircularProgressIndicator(),
339     ) // Center
340     : Expanded(
341         child: ListView.builder(
342             itemCount: itemCount,
343             itemBuilder: (BuildContext context, int index) {
344                 bool isExpense = jsonList[index]["type"] == "expense";
345                 Color color = isExpense ? Colors.red : Colors.green;
346                 return Column(
347                     children: [
348                         ListTile(
349                             title: Row(
350                                 children: [
351                                     Expanded(
352                                         child: Column(
353                                             crossAxisAlignment:
354                                                 CrossAxisAlignment.start,
355                                             children: [
356                                                 Text(
357                                                     DateFormat('E, MMM d, yyyy').format(
358                                                         DateTime.parse(
359                                                             jsonList[index]["date"]),
360                                                     ),
361                                                     style: const TextStyle(
362                                                         fontFamily: "PoppinsBold",
363                                                         fontSize: 18,
364                                                         color: Color(0xFFFFB94FD),
365                                                     ), // TextStyle
366                                                 Text(
367                                                     jsonList[index]["type"] ?? "",
```

Figure 27. Records (Code 5)

```

369     style: TextStyle(
370       fontFamily: "PoppinsBold",
371       fontSize: 18,
372       color: color,
373     ), // TextStyle
374   ],
375   // Text
376   Row(
377     children: [
378       const Icon(
379         Icons.note_alt_outlined,
380         color: Colors.white,
381       ), // Icon
382       const SizedBox(width: 5),
383       Text(
384         jsonList[index]
385           ["description"] ??
386             "",
387           style: const TextStyle(
388             fontFamily: "PoppinsMedium",
389             fontSize: 15,
390             color: Colors.white,
391           ), // TextStyle
392           ), // Text
393         ],
394       ), // Row
395     ],
396   ), // Column
397   ), // Expanded
398   Row(
399     mainAxisAlignment: MainAxisAlignment.min,
400     children: [
401       Icon(
402         isExpense
403           ? Icons.arrow_downward
404           : Icons.arrow_upward,
405         color: isExpense
406           ? Colors.red
407           : Colors.green,
408       ), // Icon
409       const SizedBox(width: 5),
410       Text(
411         "NPR ${jsonList[index]["amount"]} ?? "0.00"),
412         style: TextStyle(
413           fontFamily: "PoppinsBold",
414           fontSize: 18,
415           color: isExpense
416             ? Colors.red
417             : Colors.green,
418           ), // TextStyle
419           ), // Text
420         ],
421       ), // Row
422     ),
423   ), // ListTile
424   const SizedBox(height: 10),
425   const Divider(
426     thickness: 2,
427     color: Color(0xFFFFB94FD),
428   ), // Divider
429   const SizedBox(height: 10),
430   ],
431   ); // Column
432 },
433 ), // ListView.builder
434 ), // Expanded
435 ],
436 ), // Column
437 ), // Padding
438 ), // SafeArea
439 ); // Scaffold
440 }
441 }

```

Figure 28. Records (Code 6)

```

443 Future<void> generatePDFReport(List<dynamic> jsonList, String filePath) async {
444   // Creates a new Document object from the pdf library.
445   final pdf = pw.Document();
446   // Adds a new page to the document using a MultiPage widget from the pdf library.
447   pdf.addPage(pw.MultiPage(
448     build: (pw.Context context) => <pw.Widget>[
449       // Adds a header to the page,
450       pw.Header(level: 0, child: pw.Text('Expense Report')),
451       // Adds a table to the page using the data from jsonList.
452       pw.Table.fromTextArray(context: context, data: <List<String>>[
453         <String>['Date', 'Type', 'Description', 'Amount'],
454         ...jsonList.map((record) {
455           return <String>[
456             record['date'],
457             record['type'],
458             record['description'],
459             record['amount'].toString(),
460           ]; // <String>[]
461         })
462       ], // <List<String>>[] // pw.Table.fromTextArray
463     ]); // <pw.Widget>[] // pw.MultiPage
464   // Creates a new File object using the provided filePath.
465   final file = File(filePath);
466   // Saves the pdf document as bytes to the file.
467   await file.writeAsBytes(await pdf.save());
468   // Saves the pdf document.
469   pdf.save().then((value) {});
470 }
471

```

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

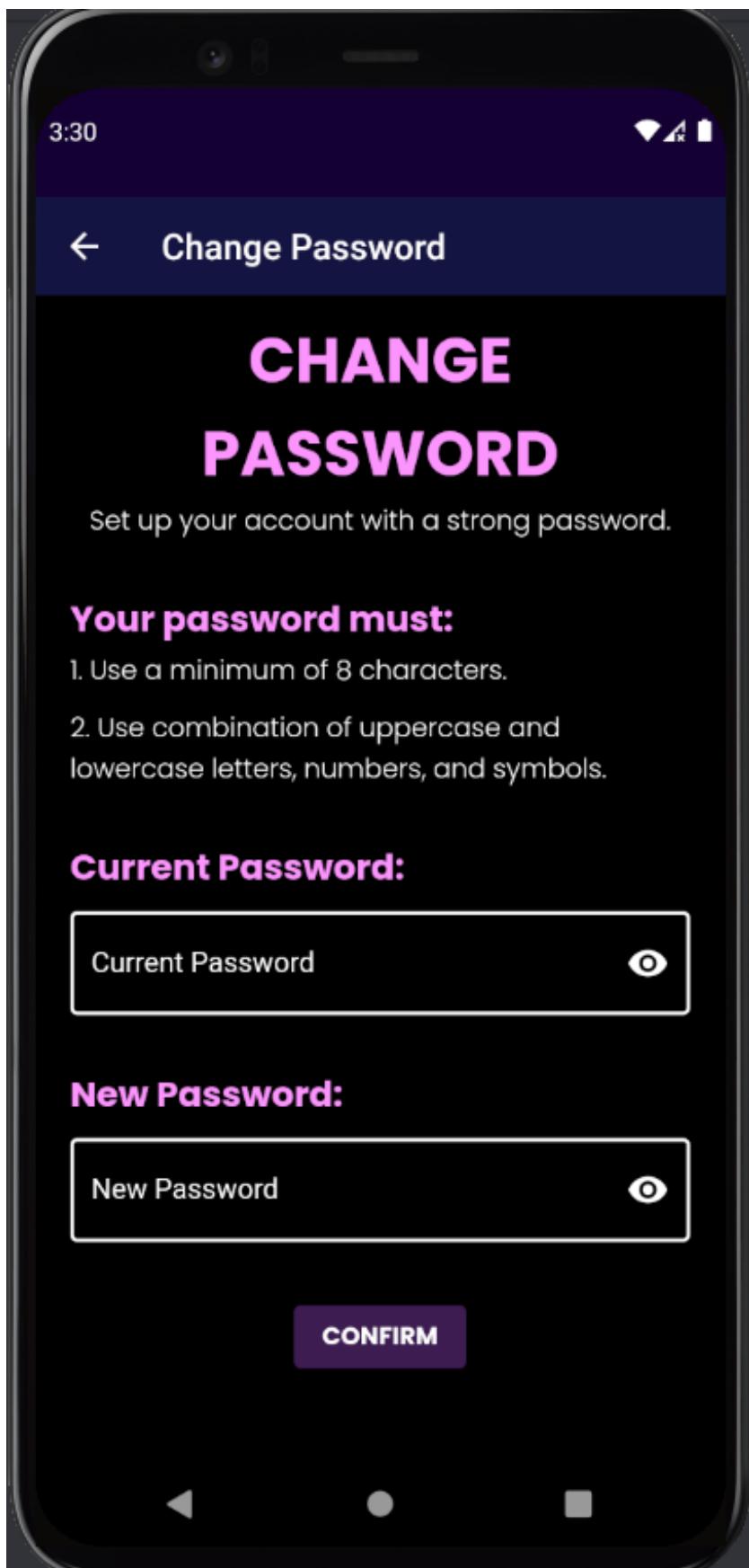


Figure 30. Change Password (Output)

```

lib > change_password.dart
1 // Importing necessary packages
2 import 'package:flutter/material.dart';
3 import 'package:hive_flutter/hive_flutter.dart';
4 import 'package:kharcha_saathi/settings.dart';
5 import 'package:http/http.dart' as http;
6 import 'dart:convert';
7
8 // StatefulWidget for changing password
9 class MyChangePassword extends StatefulWidget {
10   const MyChangePassword({super.key});
11
12   @override
13   State<MyChangePassword> createState() => _MyChangePasswordState();
14 }
15
16 class _MyChangePasswordState extends State<MyChangePassword> {
17   //creating two text controllers to retrieve user's input
18   final TextEditingController opassController = TextEditingController();
19   final TextEditingController npassController = TextEditingController();
20   //boolean variable to indicate Loading state
21   bool isLoading = false;
22   //boolean variable to toggle password visibility
23   bool _isObscure = true;
24   final GlobalKey<FormState> form = GlobalKey<FormState>();
25   var box = Hive.box('userData');
26
27   @override
28   void dispose() {
29     //disposing the text controllers
30     opassController.dispose();
31     npassController.dispose();
32     super.dispose();
33   }
34
35   @override
36   Widget build(BuildContext context) {
37     //getting the user's access token from the database
38     String accessToken = box.get("accessToken");
39     //creating a scaffold widget with a black background and appBar
40     return Scaffold(
41       backgroundColor: Colors.black,
42       appBar: AppBar(
43         backgroundColor: const Color(0xFF161341),
44         //creating back button to navigate back to settings page
45         leading: BackButton(
46           onPressed: () => Navigator.of(context).pushReplacement(
47             MaterialPageRoute(builder: (context) => const MySettings()),
48           )),
49         title: const Text('Change Password'),

```

Figure 31. Change Password (Code 1)

```

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

```

Figure 32. Change Password (Code 2)

```

99      Column(
100        crossAxisAlignment: CrossAxisAlignment.start,
101        children: [
102          const SizedBox(
103            height: 2,
104          ), // SizedBox
105          const Text(
106            "1. Use a minimum of 8 characters.",
107            style: TextStyle(
108              fontFamily: "PoppinsLight",
109              fontSize: 15,
110              color: Colors.white,
111            ), // TextStyle
112          ), // Text
113          const SizedBox(
114            height: 10,
115          ), // SizedBox
116          const Text(
117            "2. Use combination of uppercase and lowercase letters, numbers, and symbols.",
118            style: TextStyle(
119              fontFamily: "PoppinsLight",
120              fontSize: 15,
121              color: Colors.white,
122            ), // TextStyle
123          ), // Text
124          const SizedBox(
125            height: 30,
126          ), // SizedBox
127          const Text(
128            "Current Password:",
129            style: TextStyle(
130              fontFamily: "PoppinsBold",
131              fontSize: 20,
132              color: Color(0xFFB94FD),
133            ), // TextStyle
134          ), // Text
135          const SizedBox(
136            height: 10,
137          ), // SizedBox
138          // A TextFormField for the current password field
139          TextFormField(
140            autovalidateMode:
141              AutovalidateMode.onUserInteraction,
142            validator: (value) {
143              // Regular expression to validate password
144              RegExp passReg = RegExp(
145                r'^(?=.*?[A-Z])(?=.*?[a-z])(?=.*?[0-9])(?=.*?[!@#\$&*~]).{8,}$'); // RegExp
146              // Check if value matches password regex
147              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(0xfffffb94fd),

```

Figure 34. Change Password (Code 4)

```

197     ],
198   ),
199   const SizedBox(
200     height: 10,
201   ),
202   // Validate new password field
203   TextFormField(
204     autovalidateMode:
205       AutovalidateMode.onUserInteraction,
206     validator: (value) {
207       RegExp passReg = RegExp(
208         r'^.*?[A-Z](?=.*?[a-z])(?=.*?[0-9])(?=.*?[^@#$&*~]).{8,}$'); // RegExp
209       // Check if value matches password regex
210       if (!passReg.hasMatch(value.toString())) {
211         return "Invalid Password";
212       }
213       return null;
214     },
215     obscureText: _isObscure,
216     controller: npassController,
217     style: const TextStyle(
218       color: Color.fromARGB(255, 255, 255, 255), // TextStyle
219     decoration: InputDecoration(
220       suffixIcon: InkWell(
221         onTap: () {
222           setState(() {
223             _isObscure = !_isObscure;
224           });
225         },
226         child: Icon(
227           _isObscure
228             ? Icons.visibility
229             : Icons.visibility_off,
230             color: Colors.white,
231           ), // Icon
232         ), // InkWell
233         focusedBorder: const OutlineInputBorder(
234           borderSide: BorderSide(
235             color: Color.fromARGB(255, 255, 255, 255),
236             width: 2.0,
237           ), // BorderSide
238         ), // OutlineInputBorder
239         enabledBorder: const OutlineInputBorder(
240           borderSide: BorderSide(
241             color: Color.fromARGB(255, 255, 255, 255),
242             width: 2.0,
243           ), // BorderSide
244         ), // OutlineInputBorder
245         labelText: "New Password",

```

Figure 35. Change Password (Code 5)

```

248   labelStyle: const TextStyle(color: Colors.white),
249   hintText: 'Enter your new password',
250   hintStyle: const TextStyle(color: Colors.white),
251   ],
252   ),
253   const SizedBox(
254     height: 30,
255   ),
256   ],
257   ),
258   Column(
259     children: [
260       // Submit button
261       ElevatedButton(
262         onPressed: () async {
263           setState(() {
264             isLoading = true;
265           });
266           // Send patch request to update user's password
267           http.Response response = await http.patch(
268             Uri.parse(
269               "https://kharchasathi-backend.onrender.com/users/password"),
270             headers: {
271               "Authorization": 'Bearer $accessToken'
272             },
273             body: json.encode(
274             {
275               "OldPassword": opassController
276                 .text, // value of old password entered by user
277               "NewPassword": npassController
278                 .text // value of new password entered by user
279             },
280           ));
281
282           var jsonResponse = json.decode(response.body);
283           // if response status code is 200, show success message, navigate to MySettings screen and replace current screen with it
284           if (response.statusCode == 200) {
285             ScaffoldMessenger.of(context)
286               .showSnackBar(SnackBar(
287                 content: Text(jsonResponse["message"]),
288               )); // SnackBar
289             Navigator.of(context).pushReplacement(
290               MaterialPageRoute(
291                 builder: (context) => const MySettings(), // MaterialPageRoute
292               );
293             }
294           // if response status code is not 200, show error message
295           else {
296             setState(() {

```

Figure 36. Change Password (Code 6)

```

297         isLoading = false;
298     });
299     ScaffoldMessenger.of(context)
300         .showSnackBar(SnackBar(
301             content: Text(jsonResponse["error"]),
302         )); // SnackBar
303     }
304 },
305 style: ElevatedButton.styleFrom(
306     backgroundColor: const Color(0xff3E1F52),
307 ),
308 child: const Text(
309     "CONFIRM",
310     style: TextStyle(fontFamily: "PoppinsBold"),
311 ), // Text
312 ) // ElevatedButton
313 ],
314 ) // Column
315 ],
316 ), // Column
317 ), // Padding
318 ), // SafeArea
319 ), // SingleChildScrollView
320 ); // Scaffold
321 }
322 }

```

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

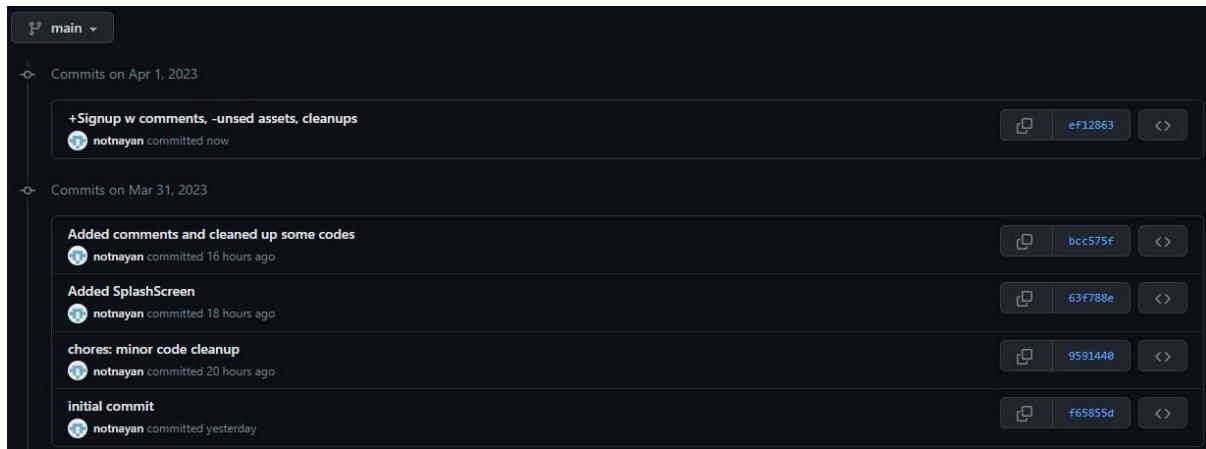


Figure 38. GitHub Commit (Sprint 1)

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

For Sprint 2:

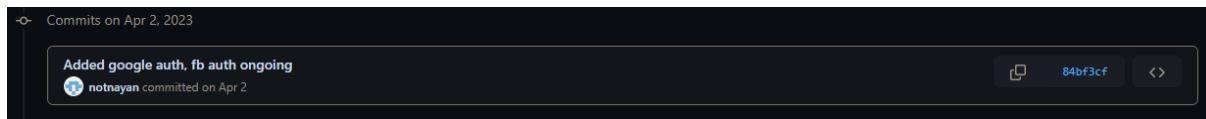


Figure 39. GitHub Commit (April 2)

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

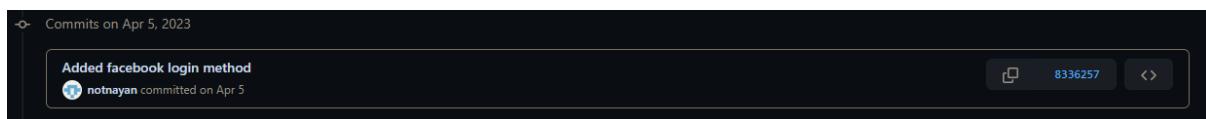


Figure 40. GitHub Commit (April 5)

On April 5th completed Facebook authentication.

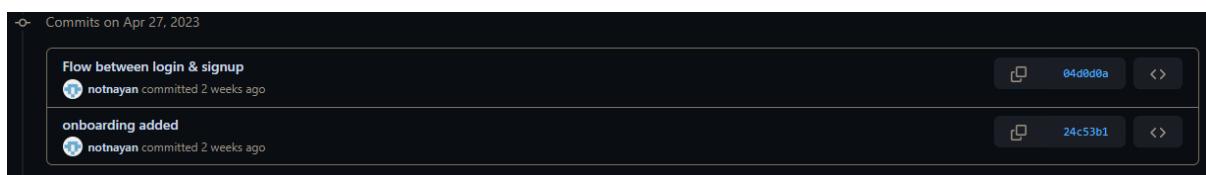


Figure 41. GitHub Commit (April 27)

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.

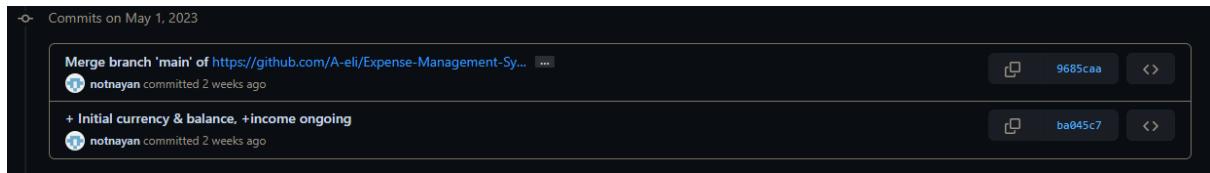


Figure 42. GitHub Commit (May 1)

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

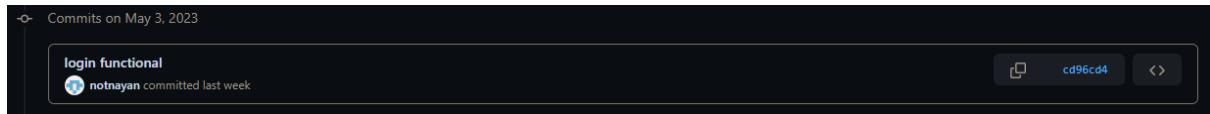


Figure 43. GitHub Commit (May 3)

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

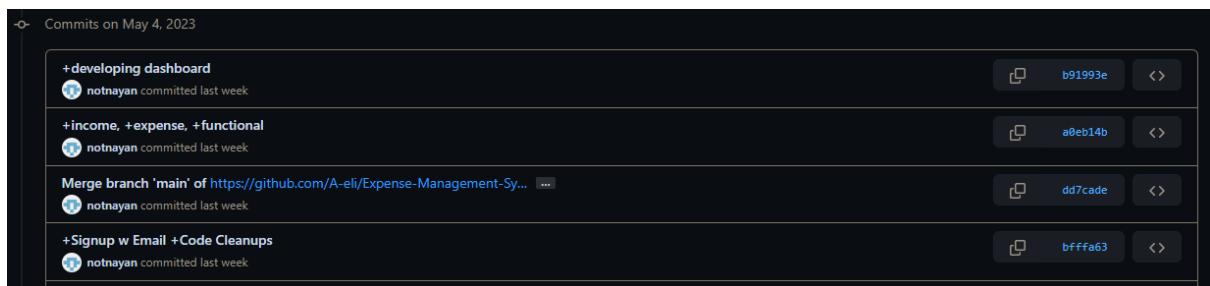


Figure 44. GitHub Commit (May 4)

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.

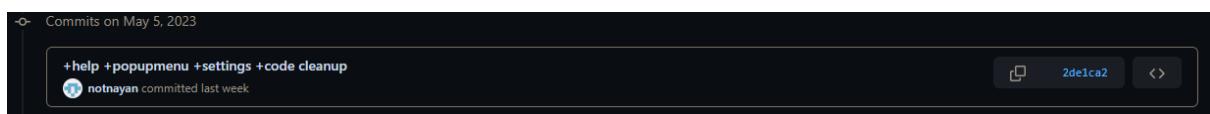


Figure 45. GitHub Commit (May 5)

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

A screenshot of a GitHub commit history for May 6, 2023. The commits are listed in chronological order from top to bottom:

- +regex +code cleanup
notnayan committed last week
- +functional google signup +password visibility
notnayan committed last week
- +search implementation
notnayan committed last week
- +dark/light mode +code cleanups
notnayan committed last week
- +changepass +code cleanup
notnayan committed last week

Each commit has a copy icon, a commit hash (e.g., 0a00b19, 9bccf7e, 96fdc92, b5a95e1, 42a922f), and a diff icon.

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.

A screenshot of a GitHub commit history for May 7, 2023. The commits are listed in chronological order from top to bottom:

- +Dynamic dashboard.records +code cleanups
notnayan committed last week
- +About page +Fixed routing +code cleanups
notnayan committed last week

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.

A screenshot of a GitHub commit history for May 8, 2023. The commits are listed in chronological order from top to bottom:

- +comments +cleanup
notnayan committed 5 days ago
- +code cleanups and finishing touches
notnayan committed 5 days ago
- +view options in settings
notnayan committed 5 days ago
- Merge branch 'main' of https://github.com/A-eli/Expense-Management-Sy...
notnayan committed last week
- +GET Auth token ...
notnayan committed last week

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.

A screenshot of a GitHub commit history for May 9, 2023. The commits are listed in chronological order from top to bottom:

- +comments +code cleanups
notnayan committed 4 days ago

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

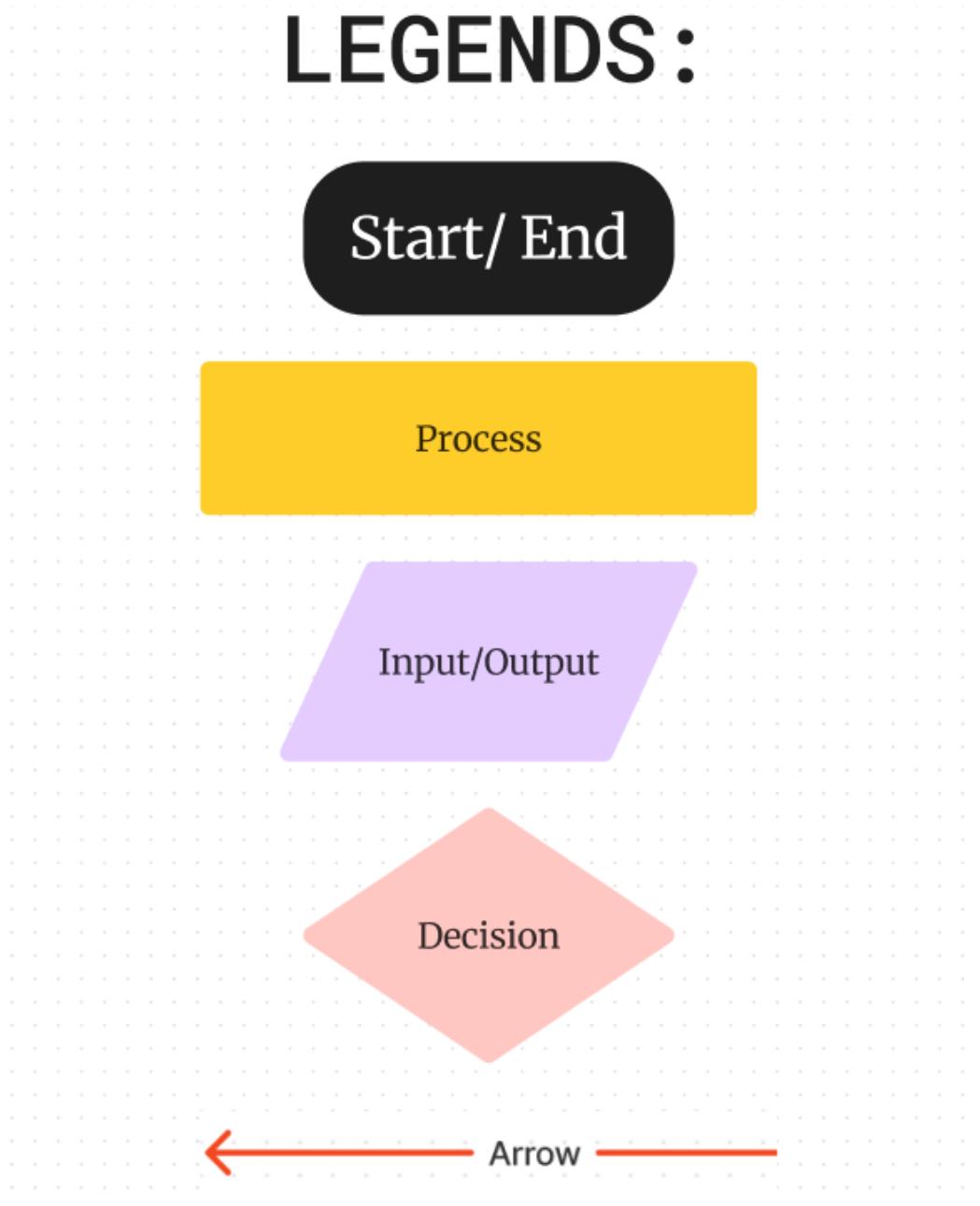


Figure 50. Flowchart Legends

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

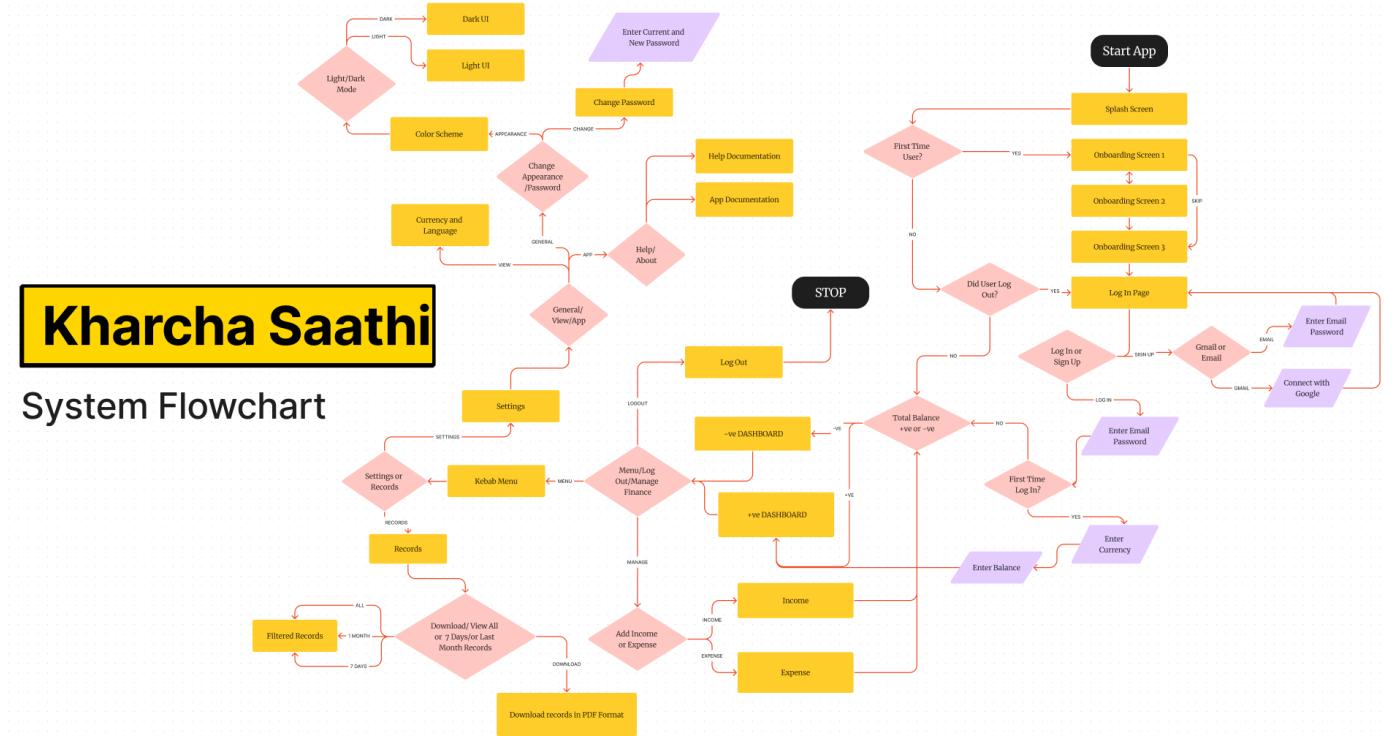


Figure 51. Flowchart of the System

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 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.

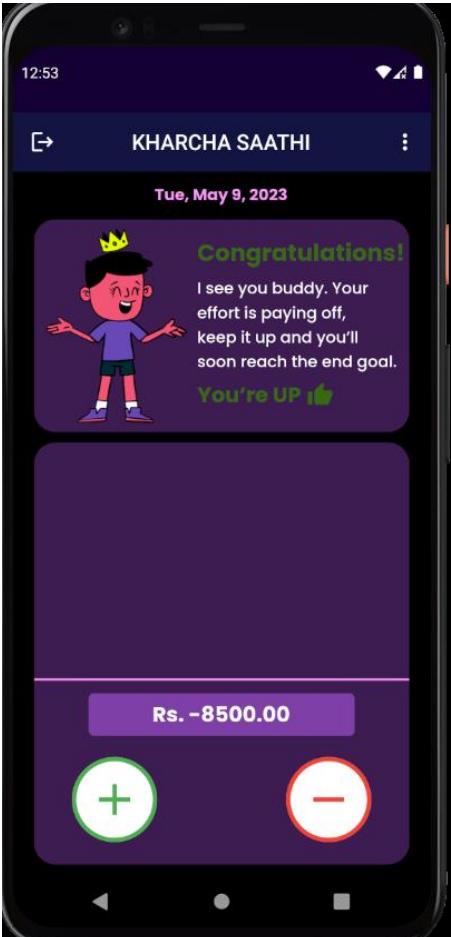
4.2. Bug Fixing



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	7 th May, 2023
Summary	After logging in, the total expense VS income donut chart is not being displayed.
Screenshot	 <p>The screenshot shows a mobile application interface for 'KHARCHA SAATHI'. At the top, it displays the time '12:53' and the date 'Tue, May 9, 2023'. Below this, there is a congratulatory message: 'Congratulations! I see you buddy. Your effort is paying off, keep it up and you'll soon reach the end goal. You're UP!' accompanied by a cartoon character wearing a crown. Below the message is a large, empty purple rectangular area. At the bottom of this area, there is a purple button with the text 'Rs. -8500.00' and two circular buttons, one with a green plus sign and one with a red minus sign. The overall background of the app interface is dark purple.</p>
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.

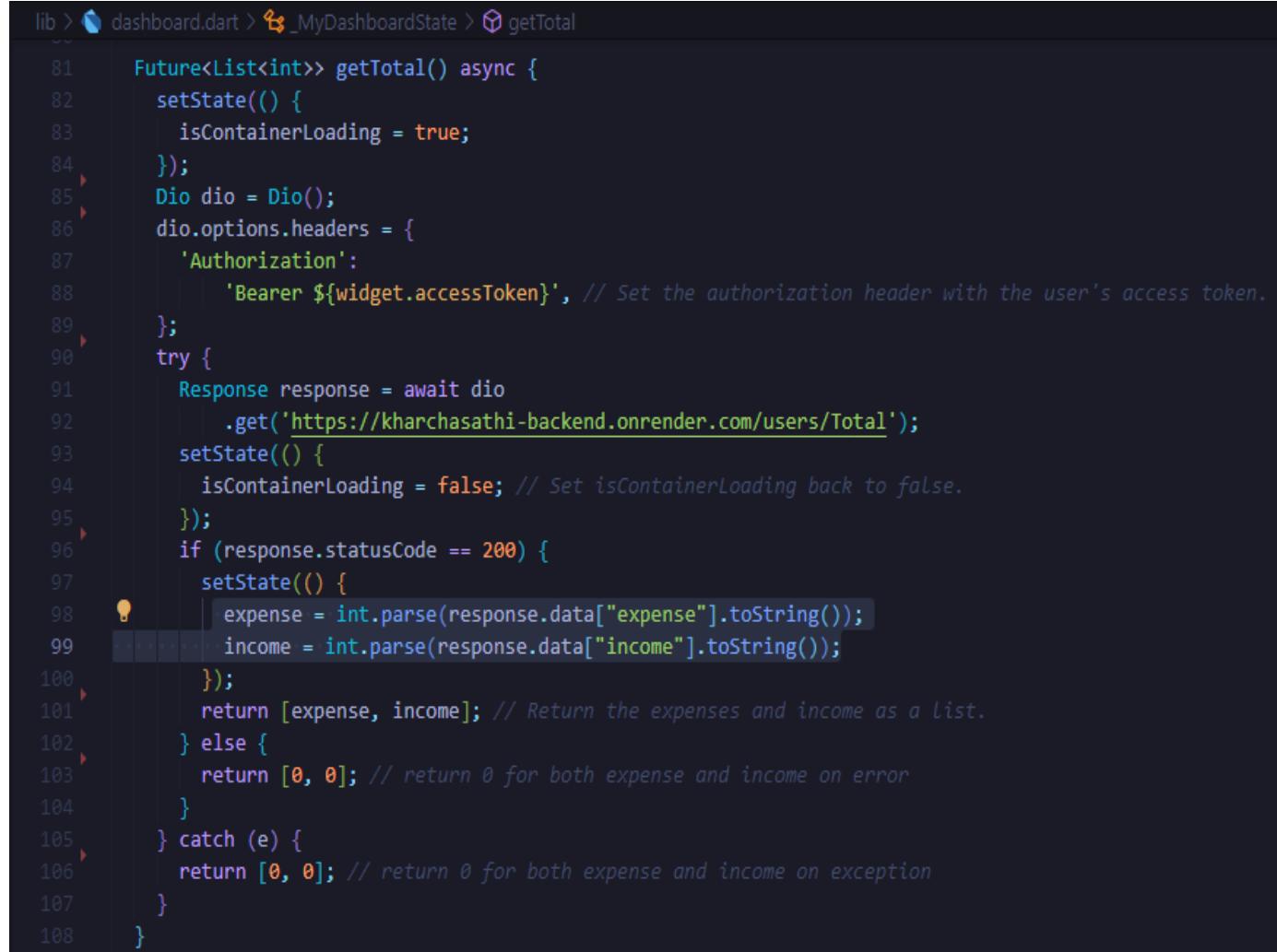
Actual result

- 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
...
81     Future<List<int>> getTotal() async {
82         setState(() {
83             isContainerLoading = true;
84         });
85         Dio dio = Dio();
86         dio.options.headers = {
87             'Authorization':
88                 'Bearer ${widget.accessToken}', // Set the authorization header with the user's access token.
89         };
90         try {
91             Response response = await dio
92                 .get('https://kharchasathi-backend.onrender.com/users/Total');
93             setState(() {
94                 isContainerLoading = false; // Set isContainerLoading back to false.
95             });
96             if (response.statusCode == 200) {
97                 setState(() {
98                     expense = int.parse(response.data["expense"].toString());
99                     income = int.parse(response.data["income"].toString());
100                });
101            return [expense, income]; // Return the expenses and income as a list.
102        } else {
103            return [0, 0]; // return 0 for both expense and income on error
104        }
105    } catch (e) {
106        return [0, 0]; // return 0 for both expense and income on exception
107    }
108}
```

Figure 54. Bug Report 1 (Fix)

Solved:

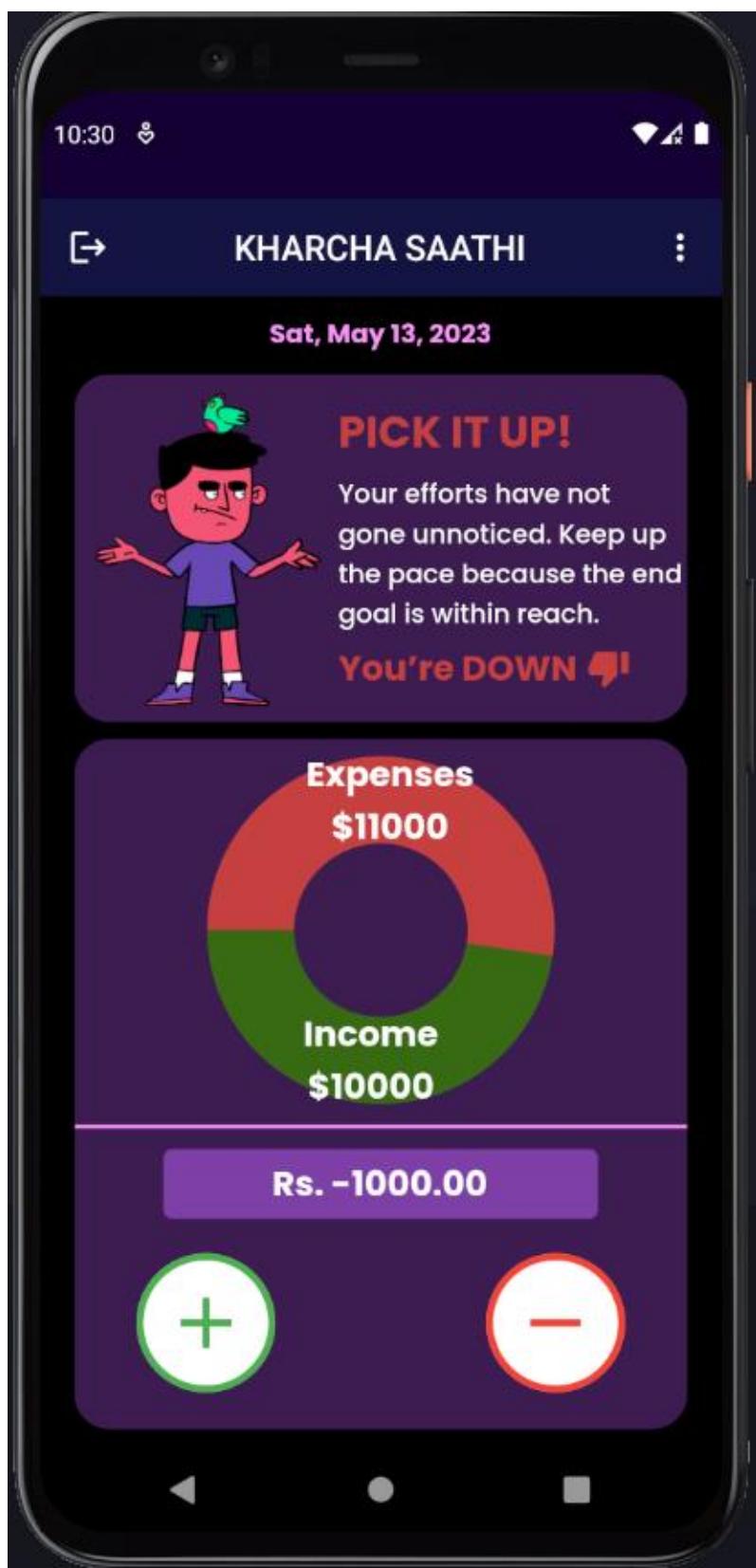
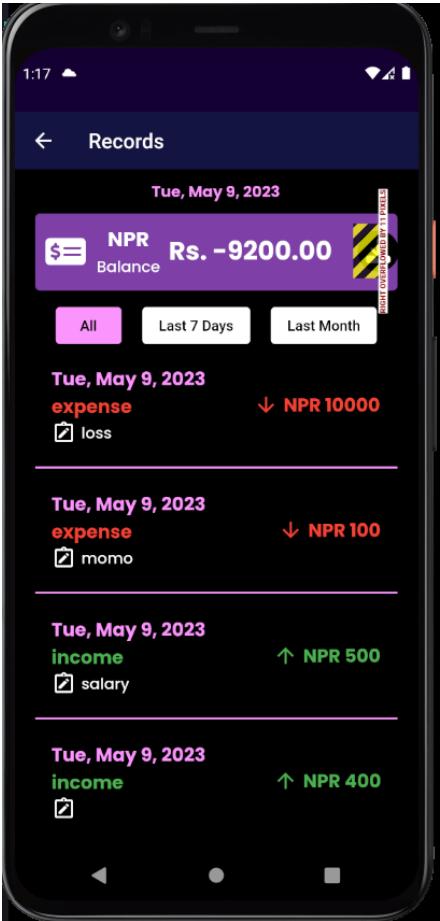


Figure 55. Bug Report 1 (Proof)

4.2.2. Bug Report 2

ID number	WBT002
Name	Text Overflow
Reporter	Nayan Raj Khanal
Submit Date	7 th May, 2023
Summary	After logging in and viewing records, the text that should display the balance overflows.
Screenshot	
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:



```
lib > records.dart > _MyRecordsState > build
201           child: CircularProgressIndicator(),
202         ) // Center
203       : Text(
204         // Show the current balance
205         'Rs. ${jsonBalance.toStringAsFixed(2)}',
206         style: const TextStyle(
207           fontFamily: "PoppinsBold",
208           fontSize: 20,
209           color: Colors.white), // TextStyle
210       ), // Text
```

Figure 57. Bug Report 2 (Fix)

Solved:

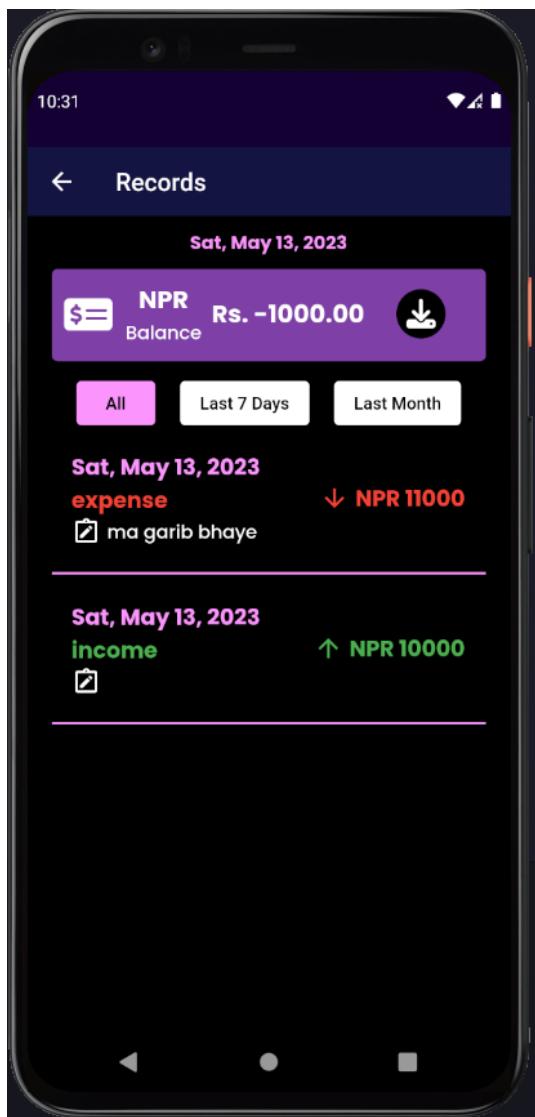
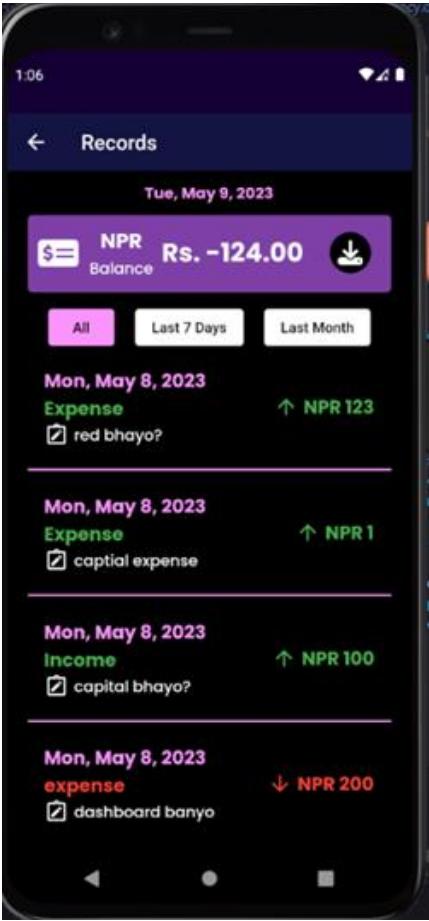


Figure 58. Bug Report 2 (Proof)

4.2.3. Bug Report 3

ID number	WBT003
Name	Colors mismatched
Reporter	Nayan Raj Khanal
Submit Date	8 th May, 2023
Summary	Expense on record is meant to be red instead of green whereas income needs to be green
Screenshot	
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

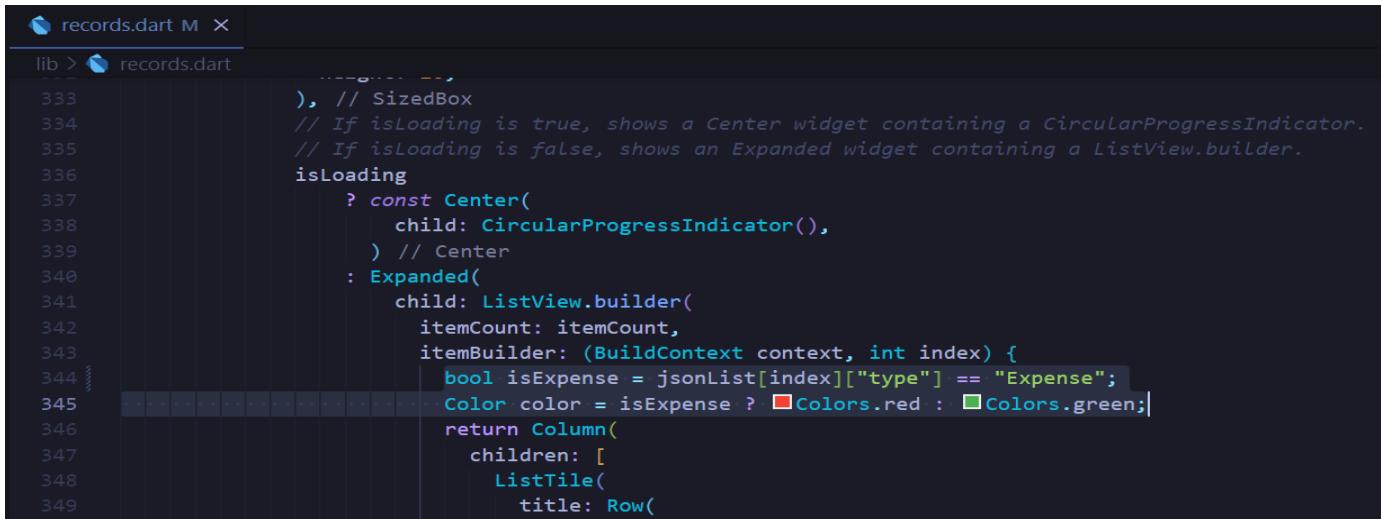
Actual result

- 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:



```
records.dart M X
lib > records.dart
333     ), // SizedBox
334     // If isLoading is true, shows a Center widget containing a CircularProgressIndicator.
335     // If isLoading is false, shows an Expanded widget containing a ListView.builder.
336     isLoading
337         ? const Center(
338             child: CircularProgressIndicator(),
339         ) // Center
340         : Expanded(
341             child: ListView.builder(
342                 itemCount: itemCount,
343                 itemBuilder: (BuildContext context, int index) {
344                     bool isExpense = jsonList[index]["type"] == "Expense";
345                     Color color = isExpense ? Colors.red : Colors.green;
346                     return Column(
347                         children: [
348                             ListTile(
349                                 title: Row(
```

Figure 60. Bug Report 3 (Fix)

Solved:

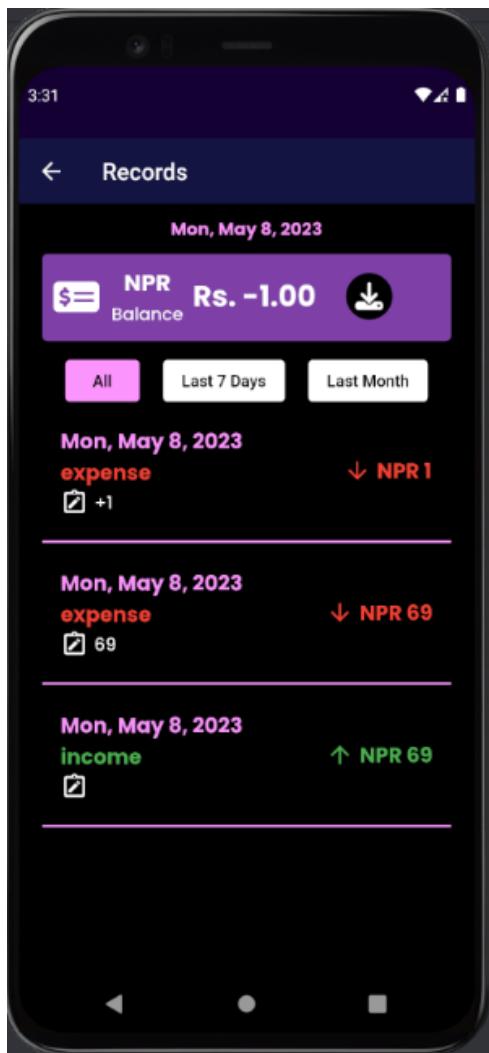
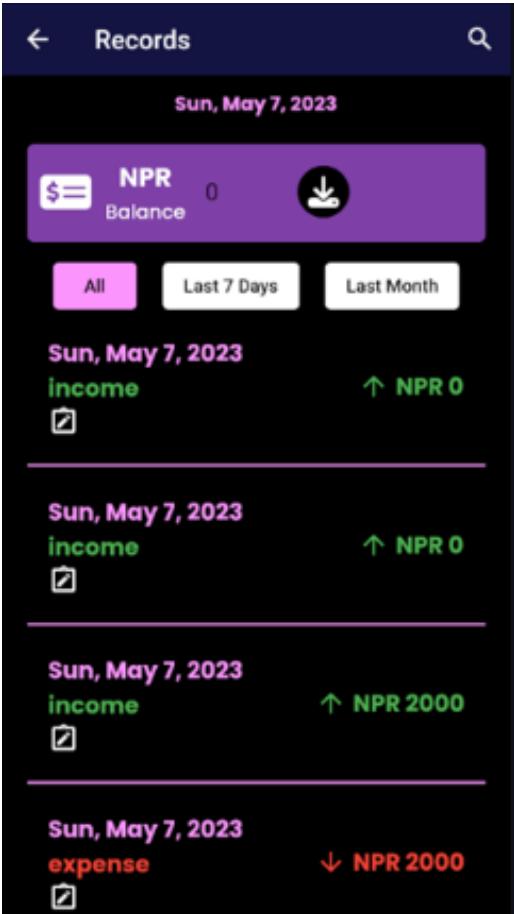


Figure 61. Bug Report 3 (Proof)

4.2.4. Bug Report 4

ID number	WBT004
Name	Empty Balance
Reporter	Nayan Raj Khanal
Submit Date	8 th May, 2023
Summary	When viewing records or dashboard the actual balance is not being showed
Screenshot	 <p>Figure 62. Bug Report 4 (Bug)</p>
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:

The screenshot shows a code editor window with the file 'records.dart' open. The code is written in Dart and defines a class '_MyRecordsState' with a method 'getTransaction'. The code uses a Dio client to make a GET request to an API endpoint. It handles the response and shows an error message if the status code is not 200. The code is annotated with comments and includes several imports at the top.

```
records.dart X
lib > records.dart > _MyRecordsState > getTransaction
  ...
62     }
63
64     // Get transactions from API
65     void getTransaction() async {
66         setState(() {
67             isLoading = true;
68         });
69
70         Dio dio = Dio();
71
72         dio.options.headers = {
73             'Authorization': 'Bearer ${box.get("accessToken")}',
74         };
75
76         try {
77             Response response = await dio
78                 .get('https://kharchasathi-backend.onrender.com/transaction');
79             if (response.statusCode == 200) {
80                 setState(() {
81                     jsonList = response.data["transactions"] as List;
82                     isLoading = false;
83                 });
84             } else {
85                 isLoading = false;
86                 ScaffoldMessenger.of(context).showSnackBar(SnackBar(
87                     content: Text('Error: ${response.statusCode}'),
88                 )); // SnackBar
89             }
90         } catch (e) {
91             isLoading = false;
92             ScaffoldMessenger.of(context).showSnackBar(SnackBar(
93                 content: Text('Error: $e'),
94             )); // SnackBar
95         }
96     }
97 }
```

Figure 63. Bug Report 4 (Fix)

Solved:

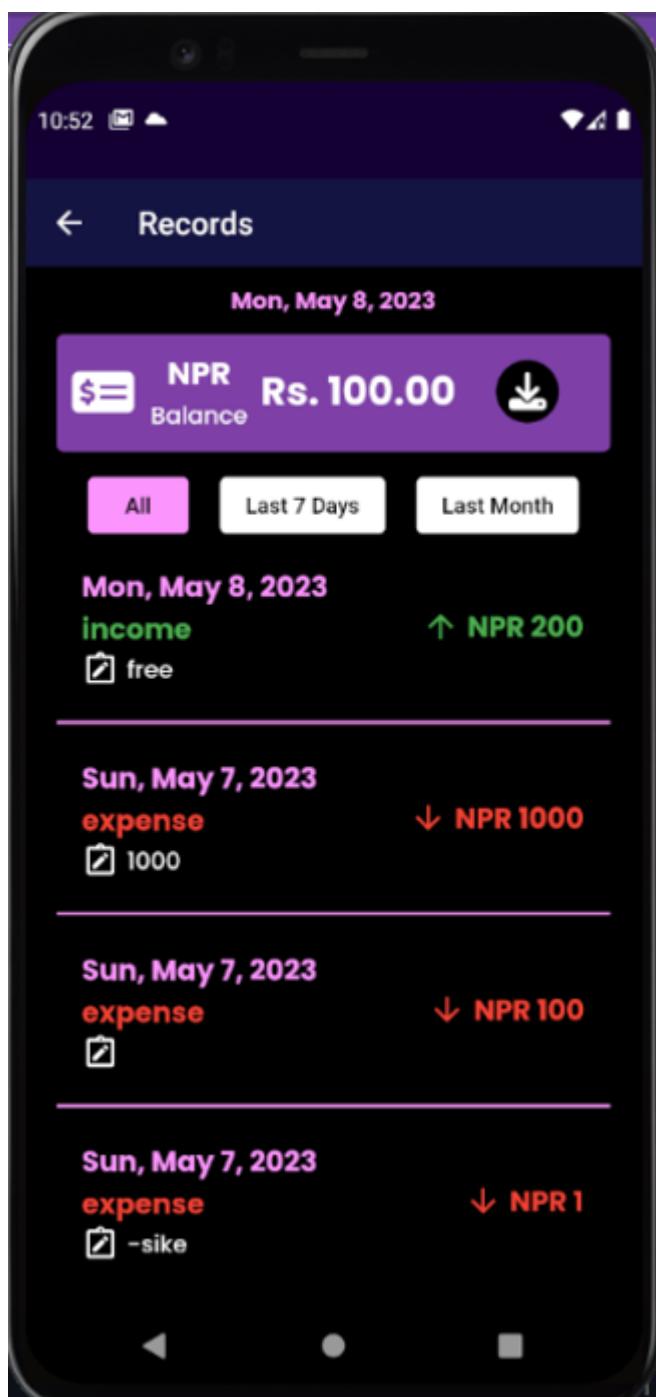
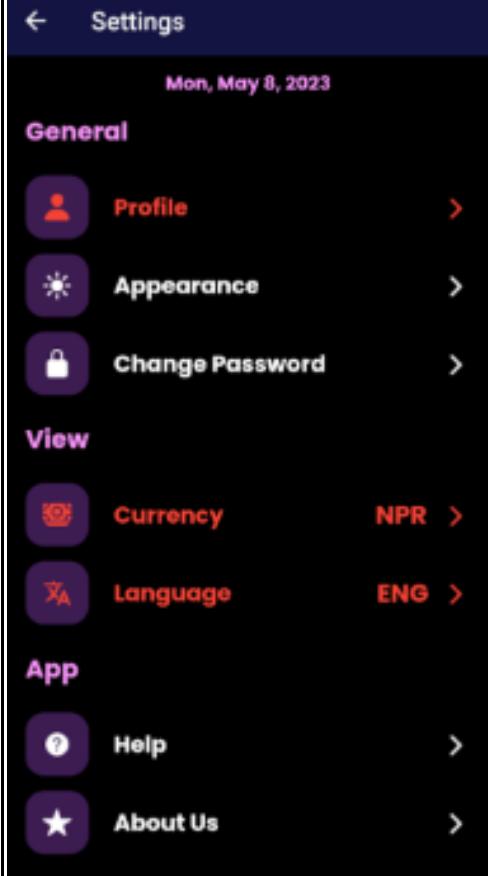


Figure 64. Bug Report 4 (Proof)

4.2.5. Bug Report 5

ID number	WBT005
Name	Return Button Error
Reporter	Nayan Raj Khanal
Submit Date	9 th May, 2023
Summary	User can't go back to the fresh dashboard from settings.
Screenshot	
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:

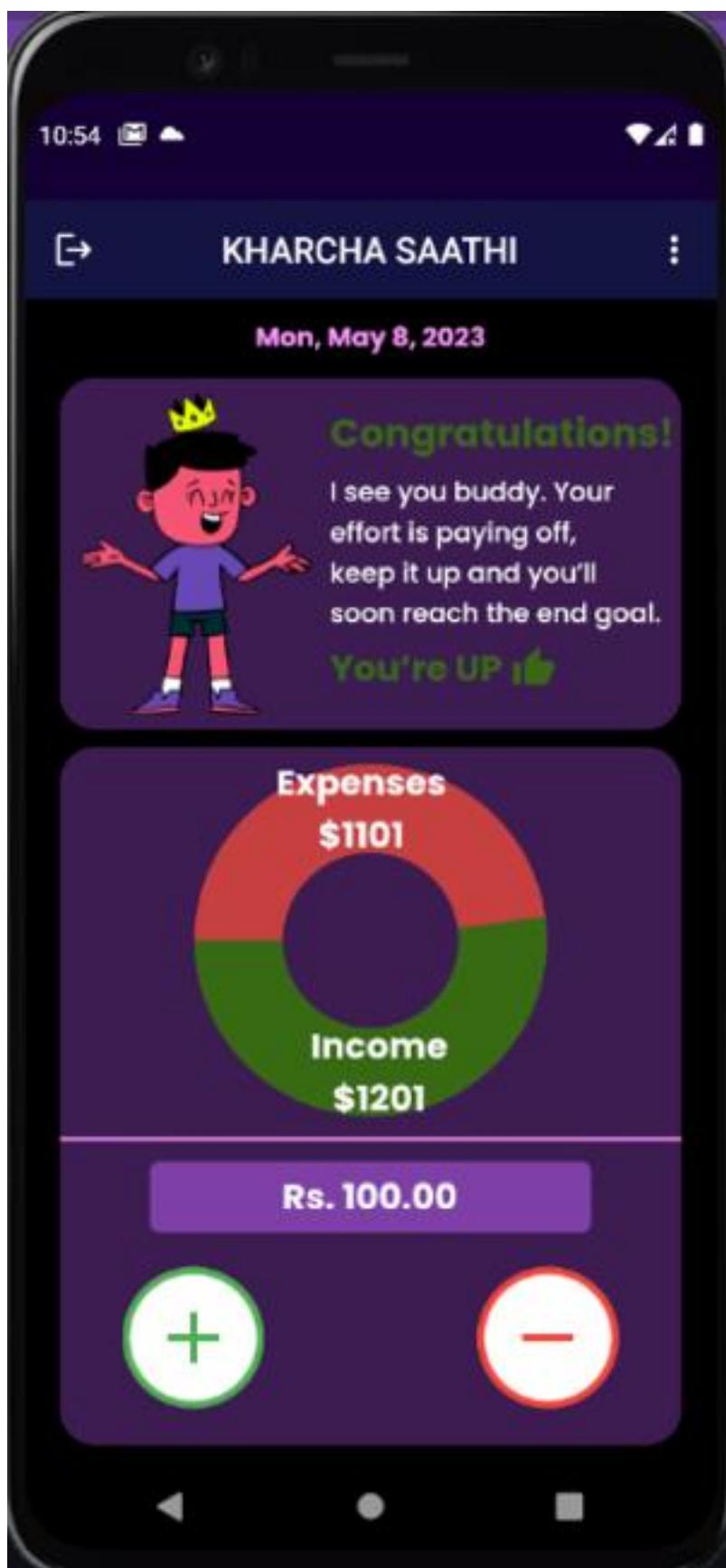


Figure 67. Bug Report 5 (Proof)

5. Appendix B

5.1. Evidences of Good communication and file sharing

The screenshot shows a dark-themed To-Do list application. At the top, it says "8/8 completed". Below that is a green circular icon with the letters "NRK" and the title "Frontend". A button labeled "Add a to-do" is visible. The list contains eight tasks, each with a checkmark, a due date, and the name "Nayan R." followed by a small green profile picture. The tasks are:

- ✓ Finalize frontend [square] Mon, May 8 NRK Nayan R.
- ✓ About page, make dashboard dynamic, fix routing [square] Sun, May 7 NRK Nayan R.
- ✓ Add regex, toggle password visibility, themes, search implementation [square] Sat, May 6 NRK Nayan R.
- ✓ Help, popupmenu, settings [square] Fri, May 5 NRK Nayan R.
- ✓ Dashboard, Settings, Help, Add Income Expense page [square] Wed, May 3 NRK Nayan R.
- ✓ Setup Currency, Initial Balance Page [square] Mon, Apr 24 NRK Nayan R.
- ✓ Develop Onboarding Screens and SetupMpin page [square] Tue, Apr 11 NRK Nayan R.
- ✓ Login Page [square] Wed, Apr 5 NRK Nayan R.

At the bottom, there is a placeholder text "Add a comment here..." next to a green circular icon with "NRK".

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.

The screenshot shows a team chat interface on Monday, May 8. A message from "AD" at 6:27pm reads: "Guys I have uploaded the time and class location of final client meeting which is tomorrow." A response from "Me" at 7:21pm reads: "Do check the details and I expect you all to be on time." Another message from "Me" at 7:21pm reads: "Noted! Will be there on time." The messages are timestamped and show the participants' initials (AD, Me) and names (Aadarsh Daseli, Nayan R.).

Figure 69. Chats (i)

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

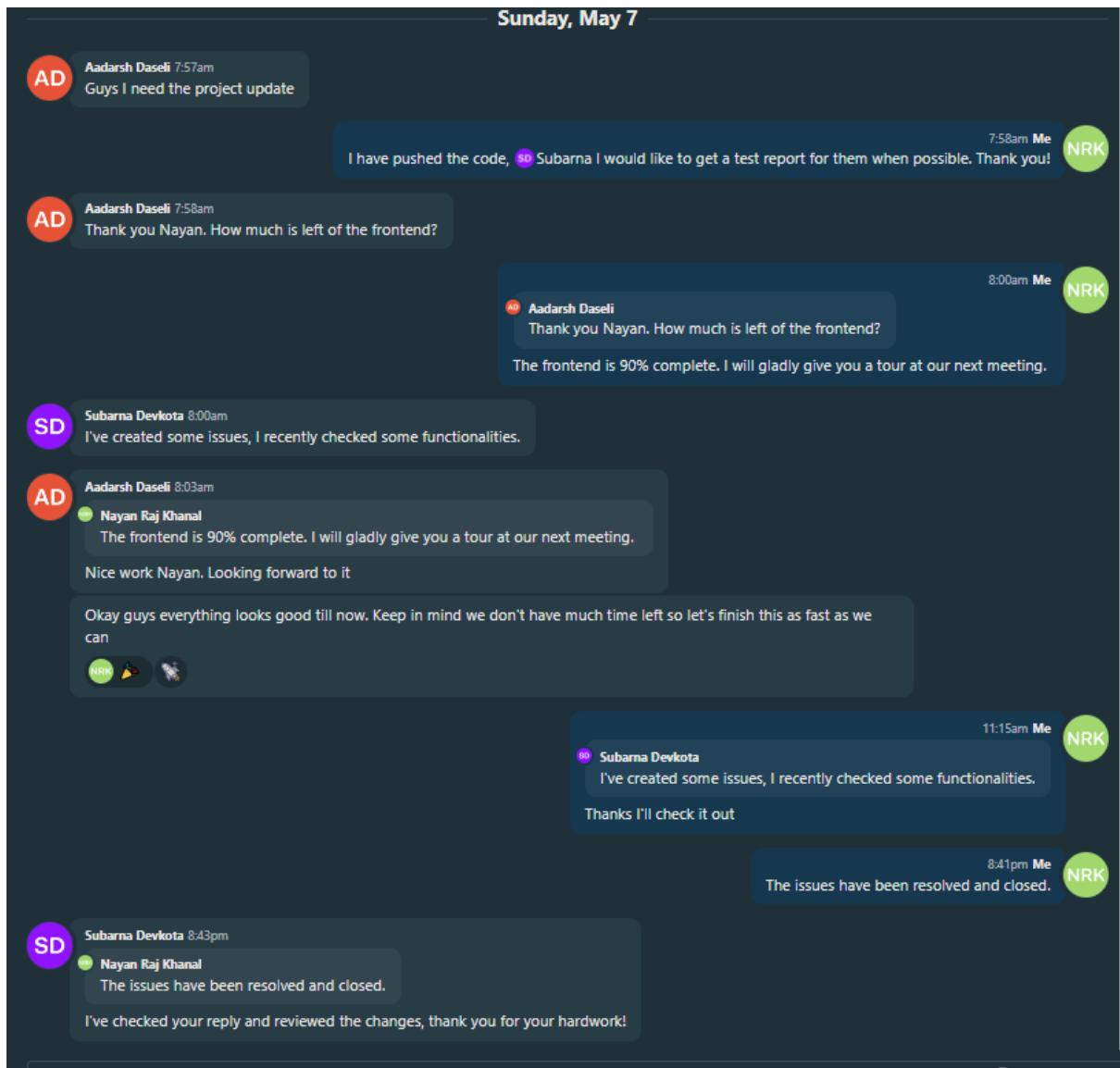


Figure 70. Chats (ii)

- 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.

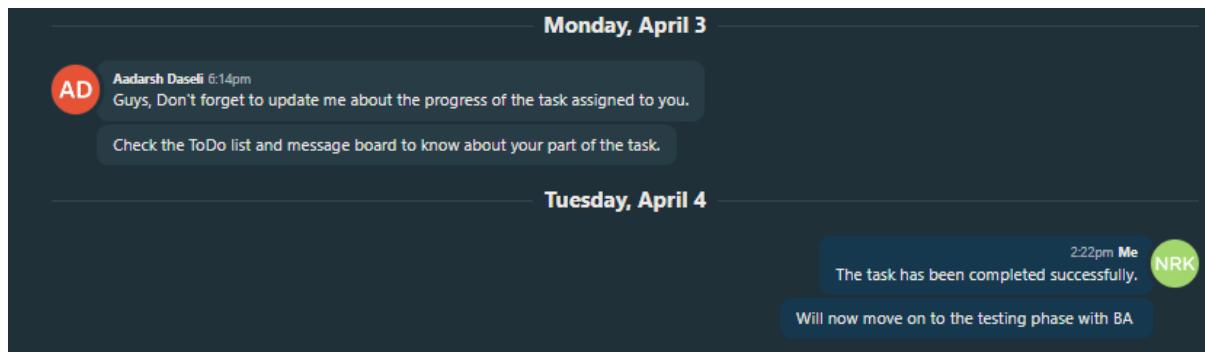


Figure 71. Chats (iii)

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

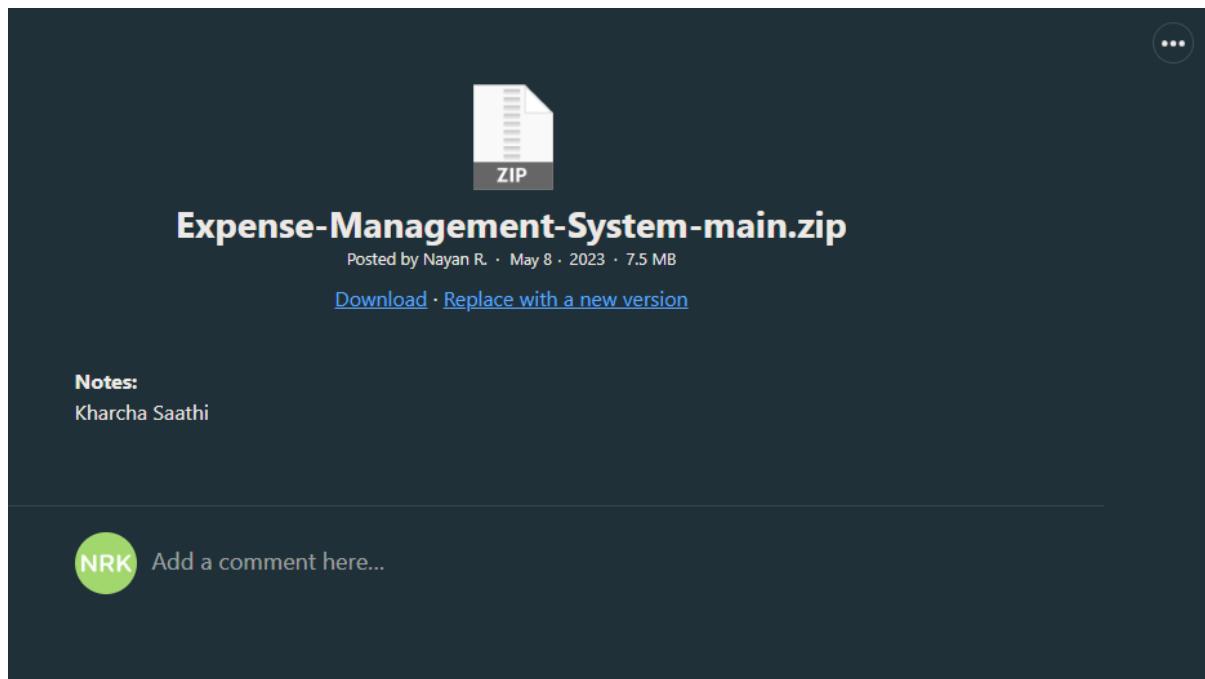


Figure 72. File Sharing

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.

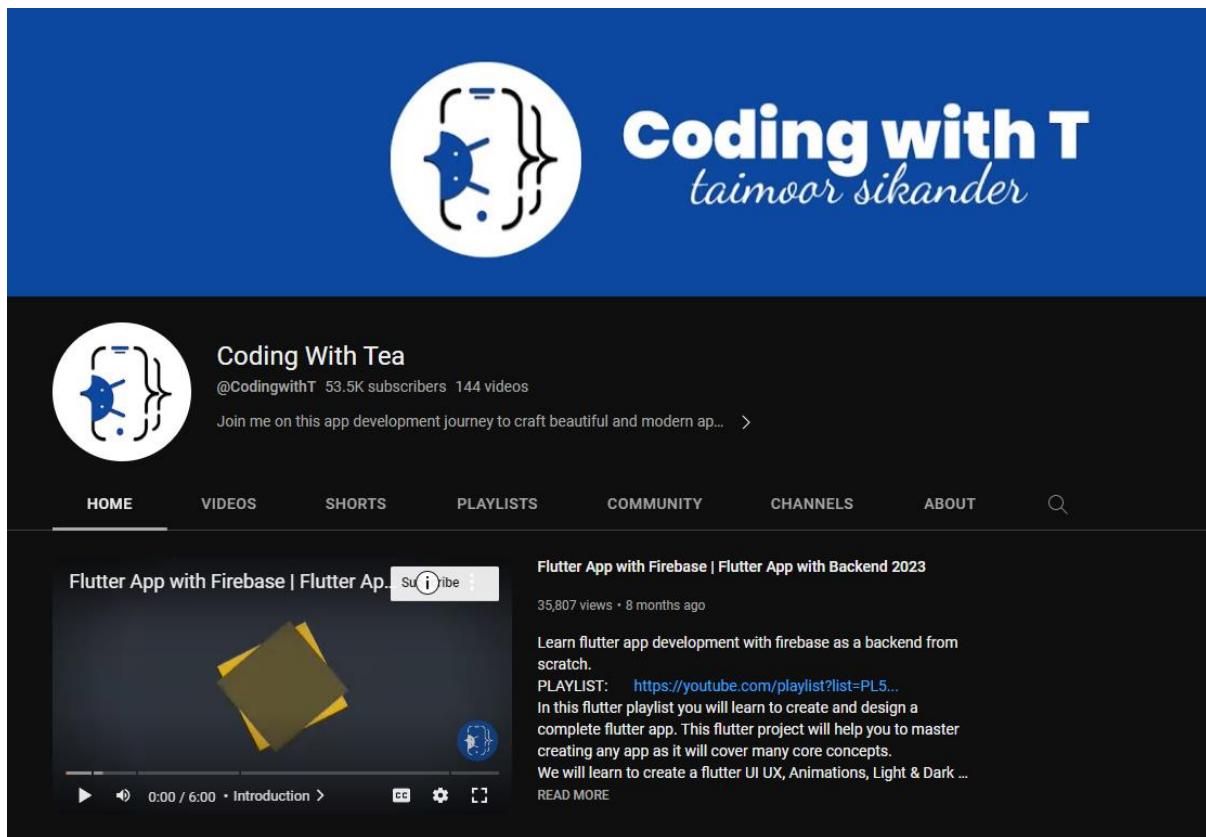


Figure 73. CPD (Evidence 1)

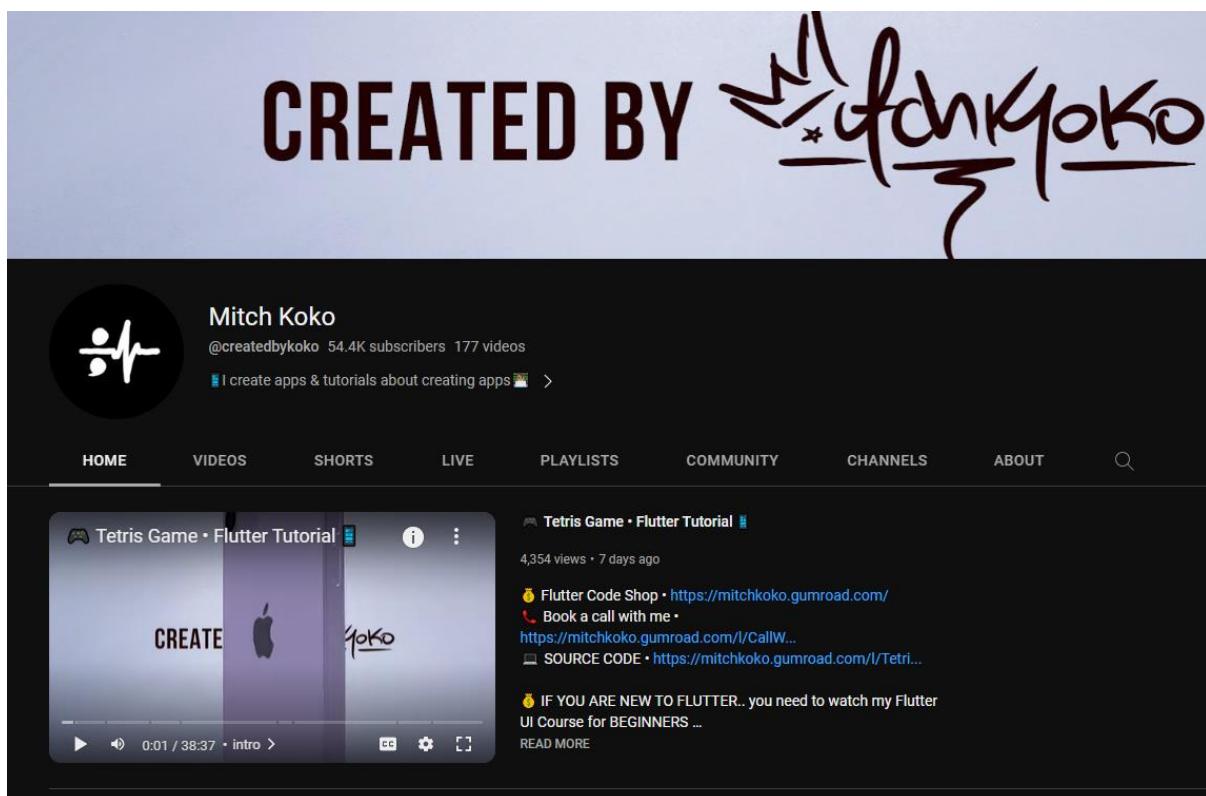


Figure 74. CPD (Evidence 2)

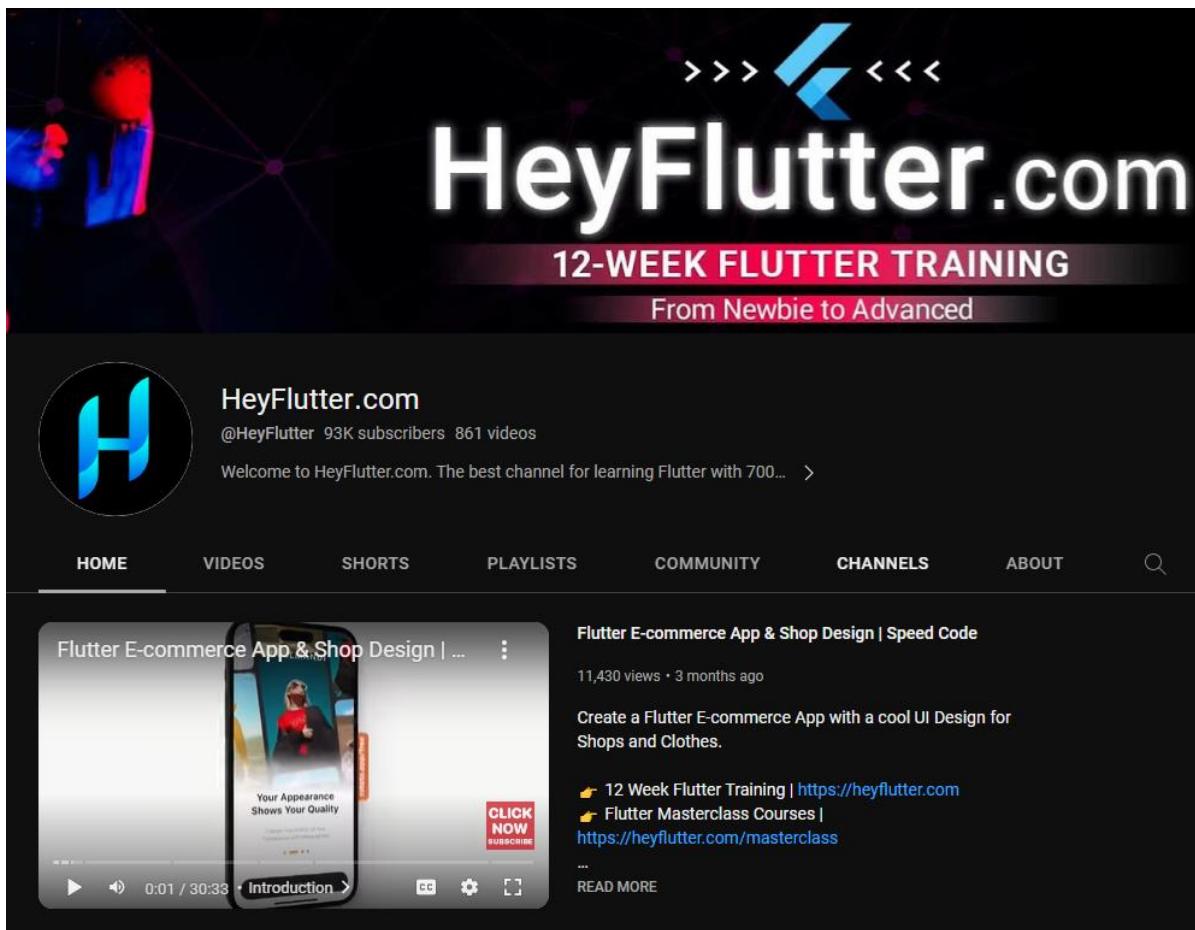


Figure 75. CPD (Evidence 3)

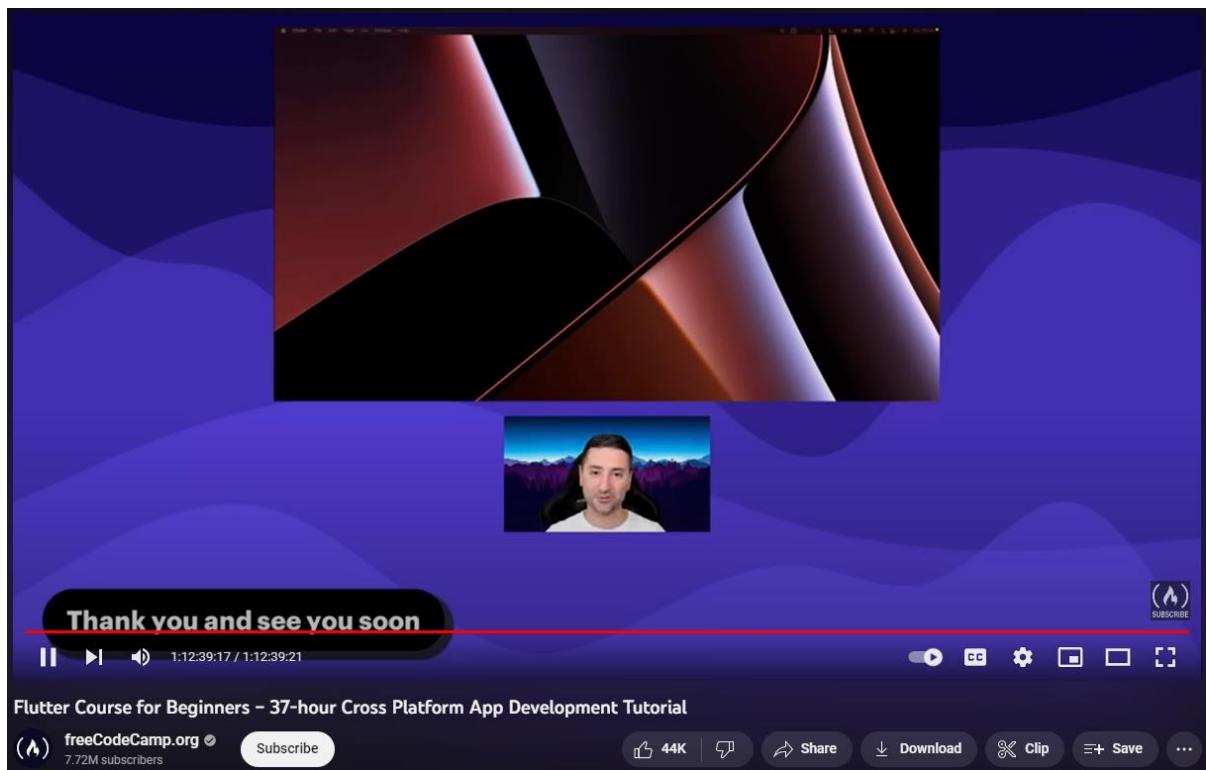


Figure 76. CPD (Evidence 4)

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.

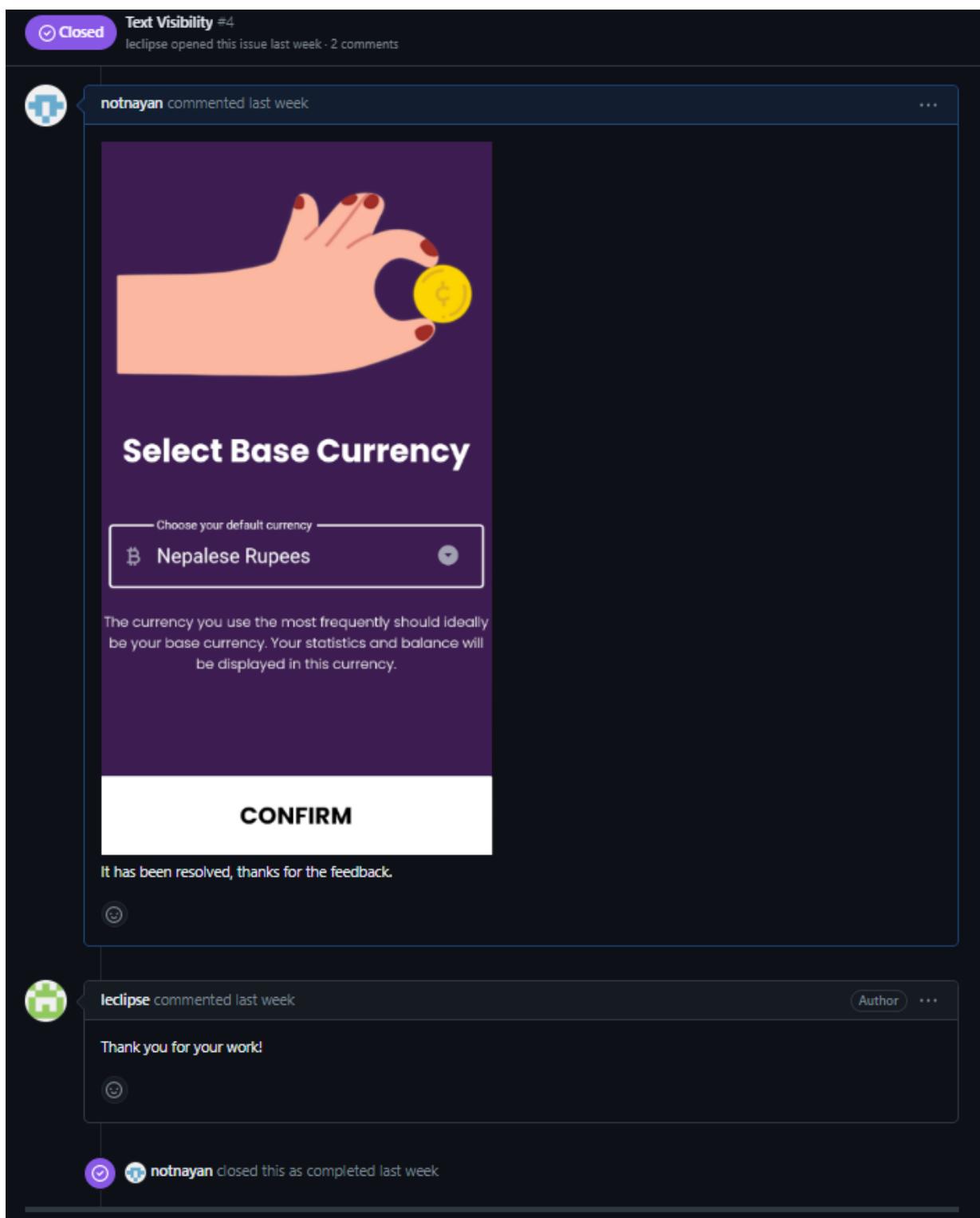


Figure 77. Issue Tracking (Evidence 1)

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.

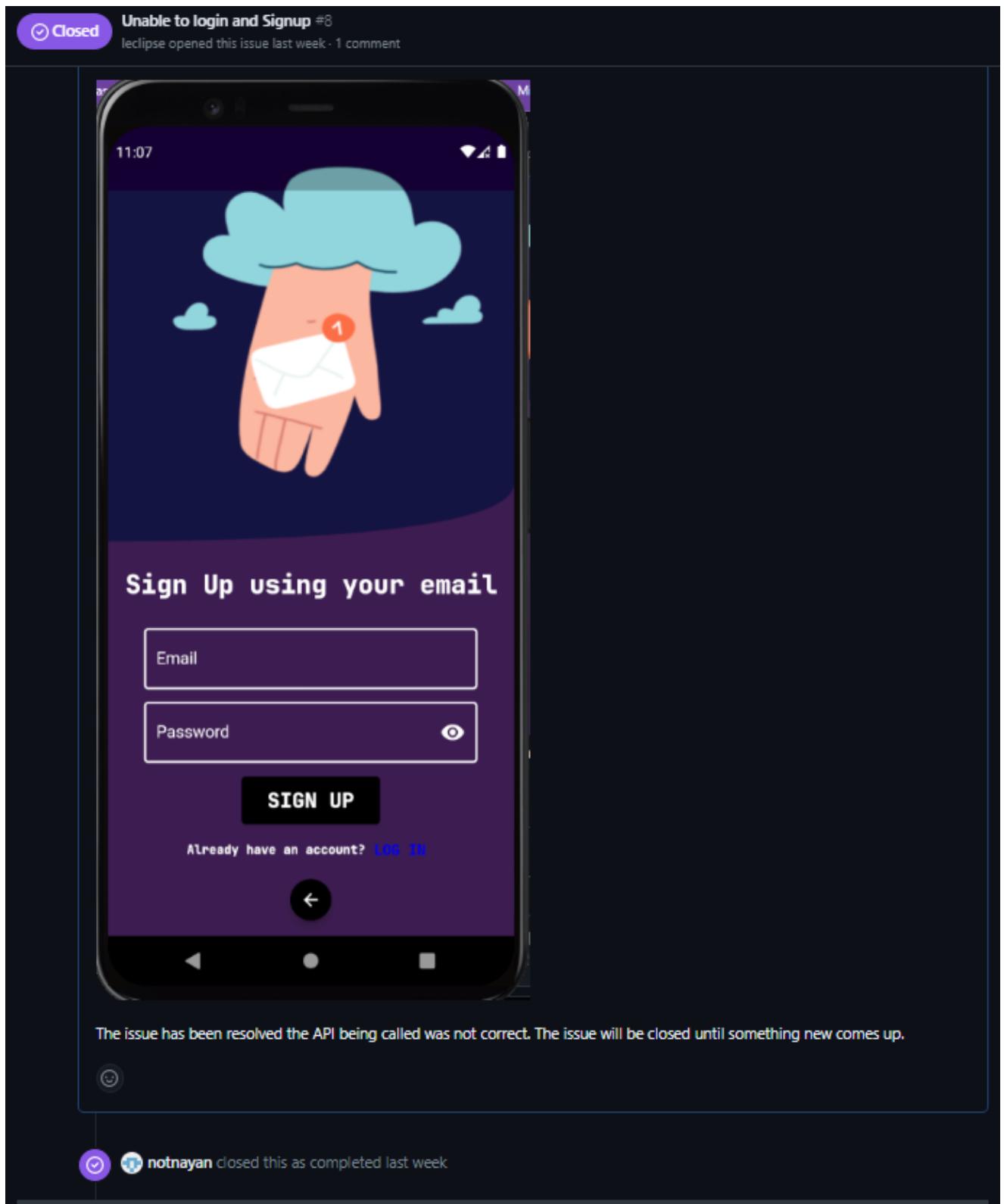


Figure 78. Issue Tracking (Evidence 2)

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.

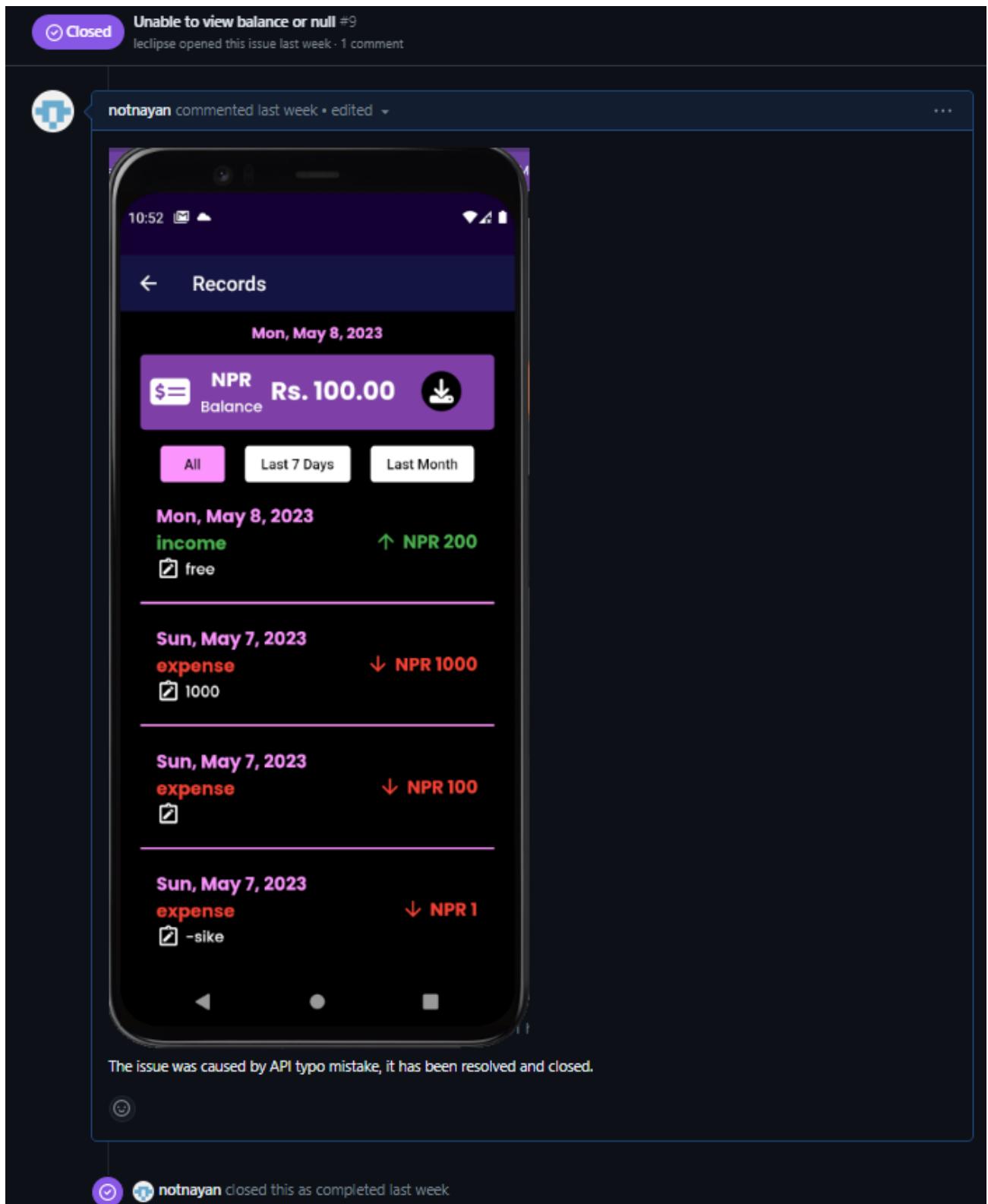


Figure 79. Issue Tracking (Evidence 3)

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.

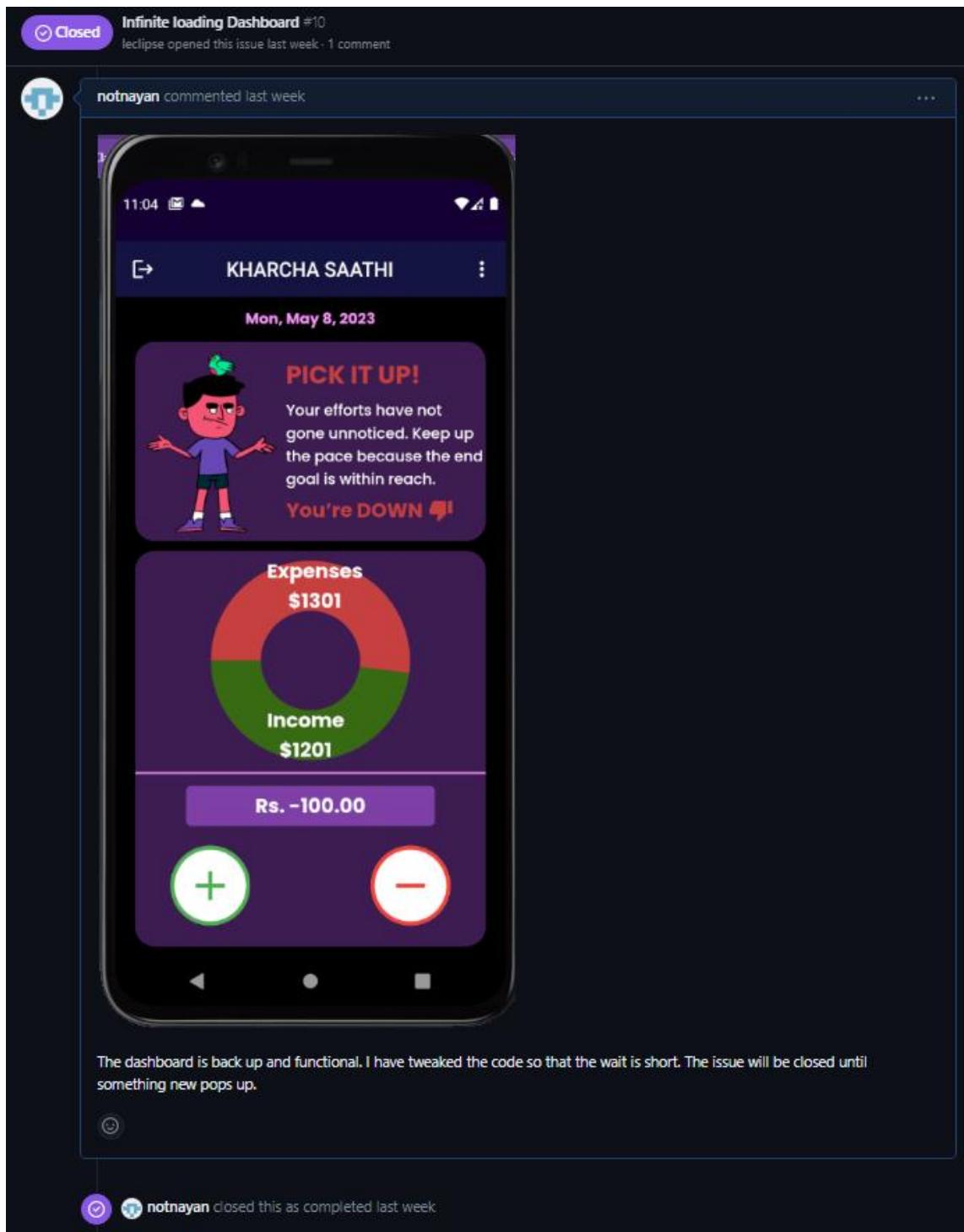


Figure 80. Issue Tracking (Evidence 4)

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.

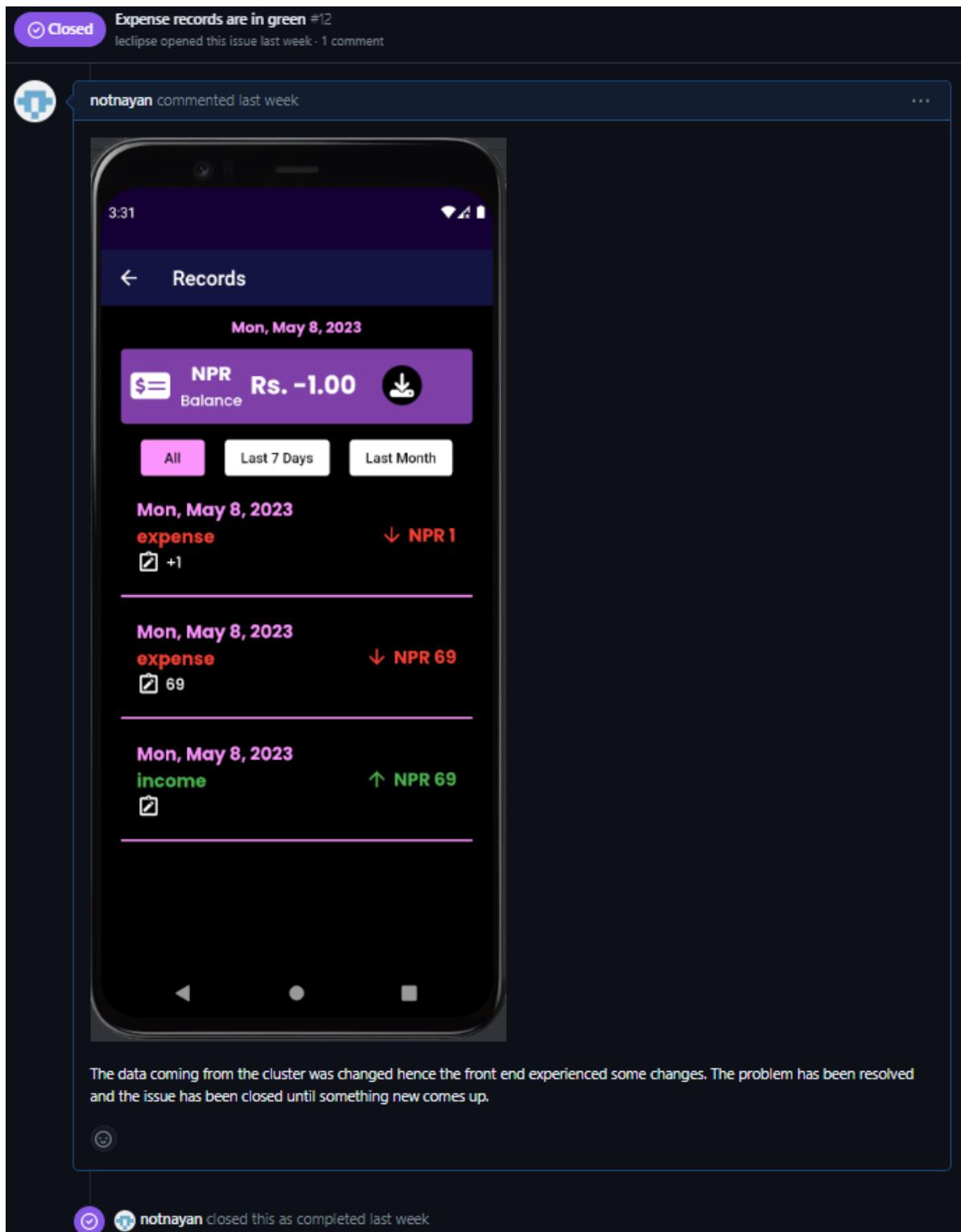


Figure 81. Issue Tracking (Evidence 5)

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/DAFiVHssqq4/ECangxZbRjm_eF3h4vRJbQ/edit?utm_content=DAFiVHssqq4&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>