

End-Term Project Report



Faculty name: Dr. Aarti Sangwan

Team Members:

Geet Khanna 21csu346

Ishita Bindal 21csu493

Lakshay Yadav 21csu342

Disha Taneja 21csu391

Sanchit Bajaj 21csu305

Semester: VI

Group: FS-B

Department of Computer Science and Engineering

The NorthCap University, Gurugram- 122001, India

Session 2023-24

Table of Contents

S.No		Page No.
1.	Project Description	3,4,5,6
2.	Problem Statement	7,8
3.	Analysis 3.1 Hardware Requirements 3.2 Software Requirements	9,10,11,12 ,13
4.	Design 4.1 Data/Input Output Description: 4.2 Algorithmic Approach / Algorithm / DFD / ER diagram/Program Steps	14-25
5.	Implementation and Testing (stage/module wise)	26.27
6.	Output (Screenshots)	28-32
7.	Conclusion and Future Scope	32-35

1. Project Description

FitMe is a cutting-edge fitness and wellness application developed using Flutter, ensuring a seamless and consistent experience across both iOS and Android platforms. Designed to help users achieve their health and fitness goals, FitMe offers personalized workout plans, nutrition guidance, and detailed progress tracking.

Key Features

1.User Login

Using google account

2.Workout Plans:

Provide customizable workout routines including strength training, cardio, and flexibility exercises.

3.Nutrition Guidance:

Deliver personalized meal plans with calorie and macronutrient tracking.

Suggest recipes and generate grocery lists to simplify healthy eating.

4.Progress Tracking:

Enable logging of daily activities, workouts, steps, and calories burned.

Track weight, measurements, and other health metrics with progress charts and milestones.

5. Personalized Fitness Solutions

Traditional fitness programs often follow a one-size-fits-all approach, which may not be effective for everyone. FitMe addresses this by offering personalized workout plans and nutrition guidance tailored to individual goals, fitness levels, and preferences. This customization increases the likelihood of achieving desired results and maintaining long-term engagement.

6. Comprehensive Health Management

Many existing fitness apps focus solely on either exercise or nutrition. FitMe combines both, providing a holistic approach to health and wellness. By integrating workout plans, meal planning, and progress tracking in one platform, users can manage all aspects of their fitness journey conveniently.

2. Problem Statement

The modern lifestyle is characterized by hectic schedules, high stress, and limited time, making it increasingly difficult for individuals to maintain their physical fitness and overall health. Traditional fitness solutions, such as gym memberships and personal trainers, are often expensive, inflexible, and inconvenient, deterring many from achieving their fitness goals.

Key Challenges:

1. Inflexible Schedules and Costly Memberships:

Many individuals find it challenging to adhere to fixed gym schedules due to busy lifestyles. Furthermore, gym memberships and personal training sessions can be prohibitively expensive for a significant portion of the population. These barriers prevent many from pursuing a consistent fitness regimen.

2. Information Overload and Misinformation:

The internet is inundated with fitness and health information, much of which is contradictory or inaccurate. This overwhelming amount of information makes it difficult for users to discern effective fitness strategies and reliable nutritional advice. Consequently, individuals may follow misguided practices that could hinder their progress or even harm their health.

3.Lack of Professional Guidance:

Access to professional fitness trainers and nutritionists is limited, especially in remote or underserved areas. Without expert guidance, individuals struggle to design effective workout plans, track their progress accurately, and adhere to nutritional guidelines, leading to suboptimal results and potential injuries.

4.Low Engagement and Motivation:

Maintaining motivation is a common challenge in long-term fitness journeys. Many existing fitness apps fail to keep users engaged over time due to a lack of personalized features, community support, and real-time feedback.

5.Data Privacy and Trust Issues:

Users are increasingly concerned about the privacy of their personal and health data. Trust issues arise when fitness apps require extensive data collection without transparent communication about data usage, potentially leading to misuse of sensitive information.

3. Analysis

3.1 Hardware Requirements

Development Environment

To develop a cross-platform application like FitMe using Flutter, the development environment should be equipped with adequate hardware to ensure smooth coding, testing, and debugging. Here are the recommended hardware requirements:

Processor (CPU):

Minimum: Intel Core i3 or equivalent.

Recommended: Intel Core i5 or higher for better performance, especially when running emulators or simulators concurrently.

Memory (RAM):

Minimum: 8 GB.

Recommended: 16 GB or more, to handle multiple IDEs, emulators, and other tools simultaneously without performance degradation.

Storage:

Minimum: 256 GB SSD.

Recommended: 512 GB SSD or more for faster read/write speeds and to accommodate project files, libraries, and dependencies.

Graphics:

Minimum: Integrated graphics with support for OpenGL 3.3 or higher.

Recommended: Dedicated GPU (NVIDIA or AMD) for better performance in rendering UI components and running emulators.

Display:

Minimum: 13-inch display with 1080p resolution.

Recommended: 15-inch or larger display with 1080p or higher resolution for better workspace management and clarity.

Target Devices for Testing

To ensure the app works seamlessly across different platforms, testing should be conducted on a variety of devices:

Android Devices:

Low-end to high-end smartphones and tablets with different screen sizes and resolutions.

Example devices: Google Pixel, Samsung Galaxy series, OnePlus, Xiaomi Redmi series.

iOS Devices:

Different models of iPhones and iPads to cover various screen sizes and performance capabilities.

Example devices: iPhone SE, iPhone 11/12/13, iPad Mini, iPad Pro.

Development Tools

Development Machines:

MacOS: Necessary for iOS development. MacBook Pro/Air or Mac Mini with the above hardware specifications.

Windows/Linux: Suitable for Android development. Machines meeting the above hardware specifications.

Additional Hardware:

External Monitors: Dual-monitor setup for enhanced productivity.

Mobile Devices: Multiple Android and iOS devices for real-world testing.

Peripheral Devices: Good quality keyboard and mouse, and possibly a touch screen monitor for testing touch interactions.

3.2 Software Requirements

1. Development Environment

Flutter SDK:

Download and install the latest stable version of the Flutter SDK from the official Flutter website.

Dart SDK:

Dart is the programming language used for Flutter development. The Flutter SDK includes the Dart SDK, so no separate installation is required.

Integrated Development Environment (IDE):

Visual Studio Code: Lightweight and highly customizable with Flutter and Dart plugins.

2. Command Line Tools:

Flutter CLI: Used for creating, building, and managing Flutter projects

Dart CLI: For running and managing Dart code outside the Flutter framework.

Platform-Specific Tools'

3.Android Development:

Android SDK: Includes essential tools for Android app development.

Android Studio: Recommended for managing Android SDK and using the Android emulator.

Android Emulator: For testing the app on different Android devices and configurations.

4.iOS Development:

Xcode: Required for developing and testing iOS apps. Includes the iOS SDK, simulators, and necessary tools for iOS development. Download from the Mac App Store.

CocoaPods: Dependency manager for iOS projects, required for integrating Flutter plugins.

Version Control

5.Git:

Essential for version control and collaboration. Install from Git's official site.

GitHub, GitLab, or Bitbucket for repository hosting and collaboration.

6. Firebase:

For authentication, real-time database, cloud storage, and other backend services. Set up a Firebase project from the Firebase Console.

Postman:

API development and testing tool. Useful for testing backend APIs that the app will interact with.

4. Design

4.1 Data

In a FitMe app developed in Flutter, the data input/output process typically involves capturing user input related to fitness metrics and preferences, processing that data, and displaying relevant output or recommendations. Here's a basic outline of how this might work:

Data Input:

User Profiles: Users input personal information such as age, gender, weight, height, fitness goals, activity level, and any health conditions.

Fitness Metrics: Users input data related to their workouts, such as duration, type of exercise, intensity level, calories burned, distance covered, etc.

Nutritional Information: Users may input their daily food intake, including the type and quantity of food consumed.

Wearable Devices Integration: If the app integrates with wearable devices like fitness trackers or smartwatches, it can automatically collect data such as steps taken, heart rate, sleep patterns, etc.

Data Processing:

Calculations: The app performs calculations based on the input data to generate insights or recommendations. For example, it might calculate the user's daily caloric needs, recommend workout routines based on their goals, analyze their progress towards those goals, etc.

Data Analysis: Machine learning algorithms or predefined rules may analyze the data to provide personalized recommendations or insights. For instance, the app might suggest adjusting the user's diet or exercise regimen based on their progress.

Data Aggregation: The app aggregates data over time to track the user's progress, such as weight loss/gain, improvements in fitness level, adherence to workout plans, etc.

Data Output:

Visualizations: The app presents data to users through visualizations like charts, graphs, progress bars, etc., to help them understand their progress and stay motivated.

Notifications: Users may receive notifications based on their progress or reminders for upcoming workouts or meal times.

Recommendations: The app provides personalized recommendations for workouts, dietary adjustments, or lifestyle changes based on the user's goals and progress.

Feedback: Users may receive feedback on their activities, such as congratulatory messages for achieving a milestone or suggestions for improvement.

Data Storage:

Local Storage: User data is typically stored locally on the device, using tools like SQLite or shared preferences for persistent storage.

Cloud Storage: Optionally, user data can be synced to a cloud server for backup and cross-device synchronization, using services like Firebase Firestore or AWS DynamoDB.

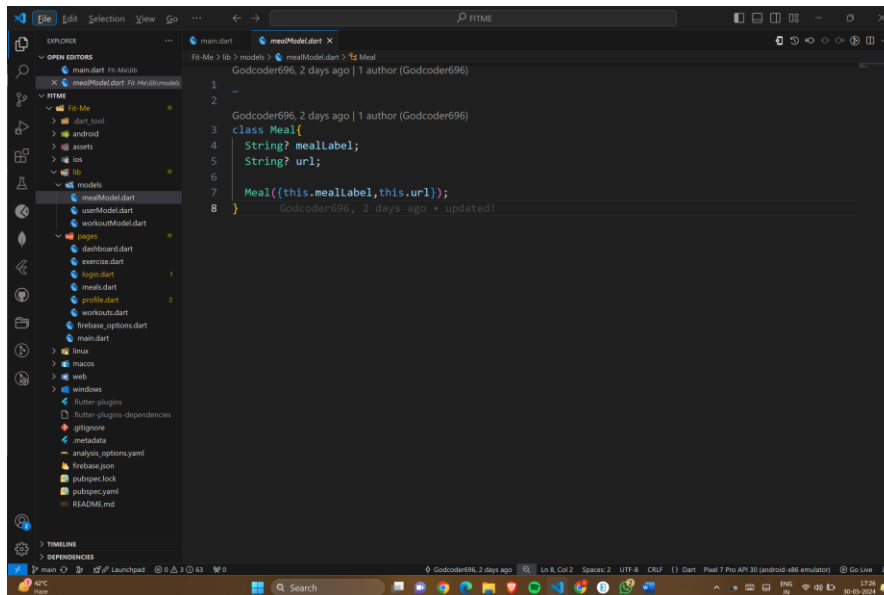
Throughout this process, it's essential to prioritize user privacy and data security, ensuring that sensitive information is handled appropriately and securely.

Additionally, providing users with control over their data, such as the ability to delete or export it, enhances trust and user experience.

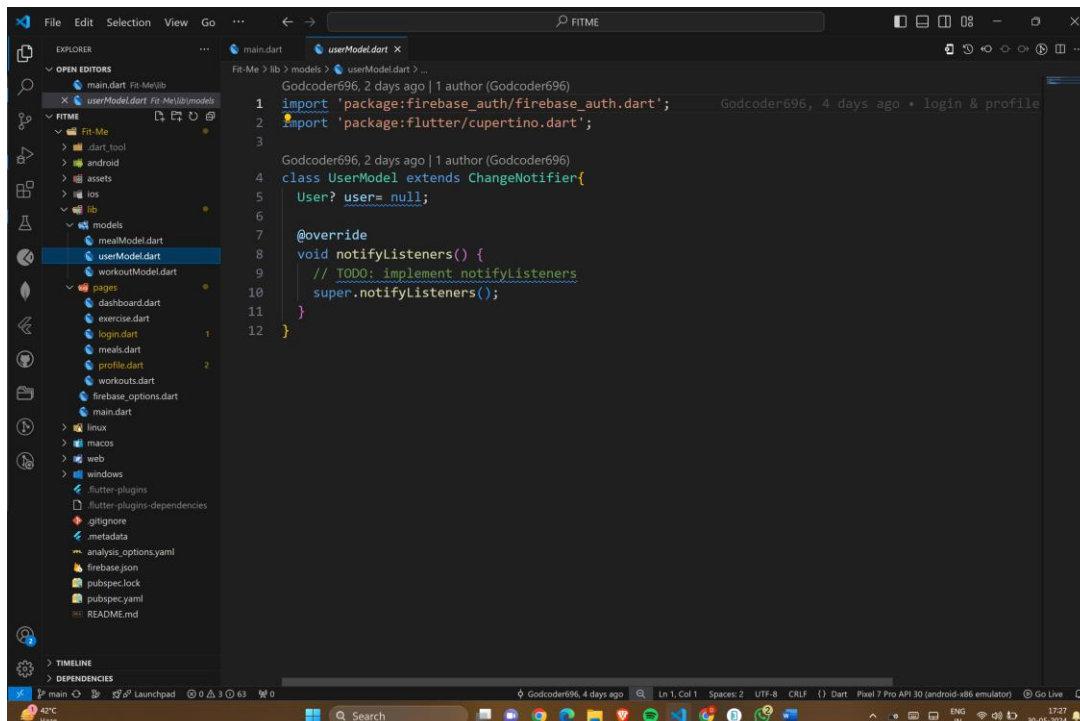
4.2 Algorithm and Code

Models:

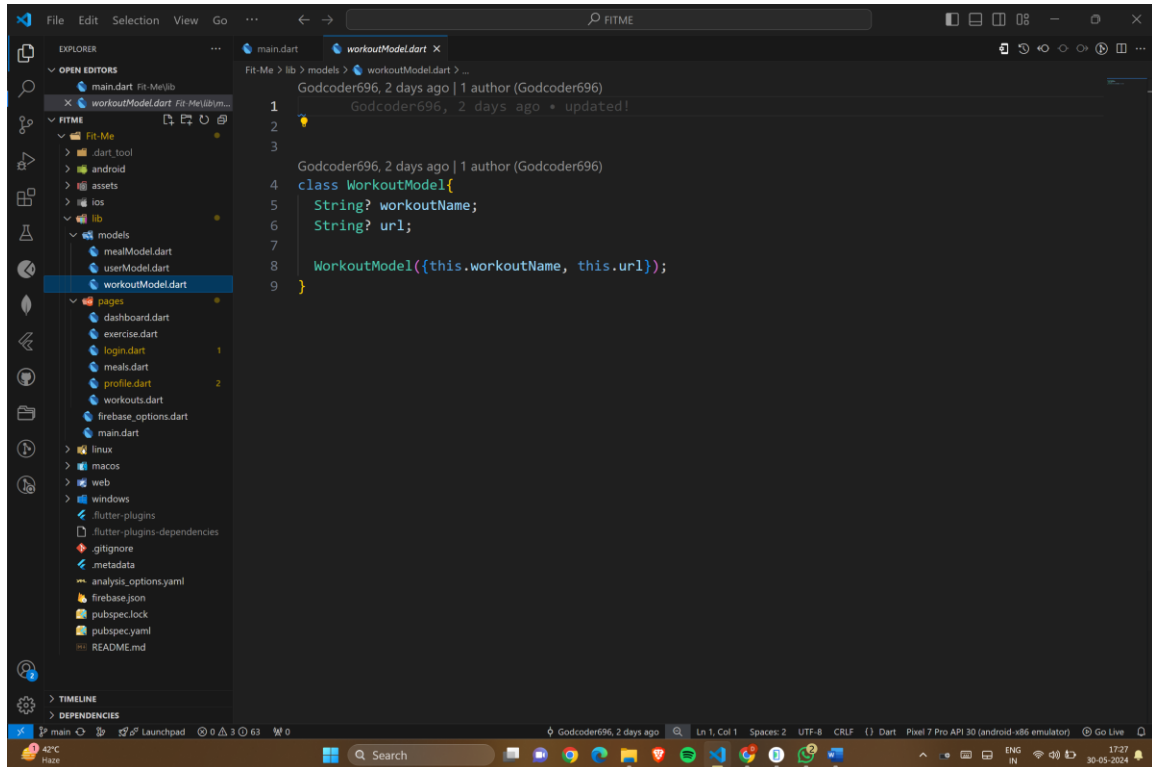
1. mealModel.dart



2. userModel.dart



3. workoutModel.dart



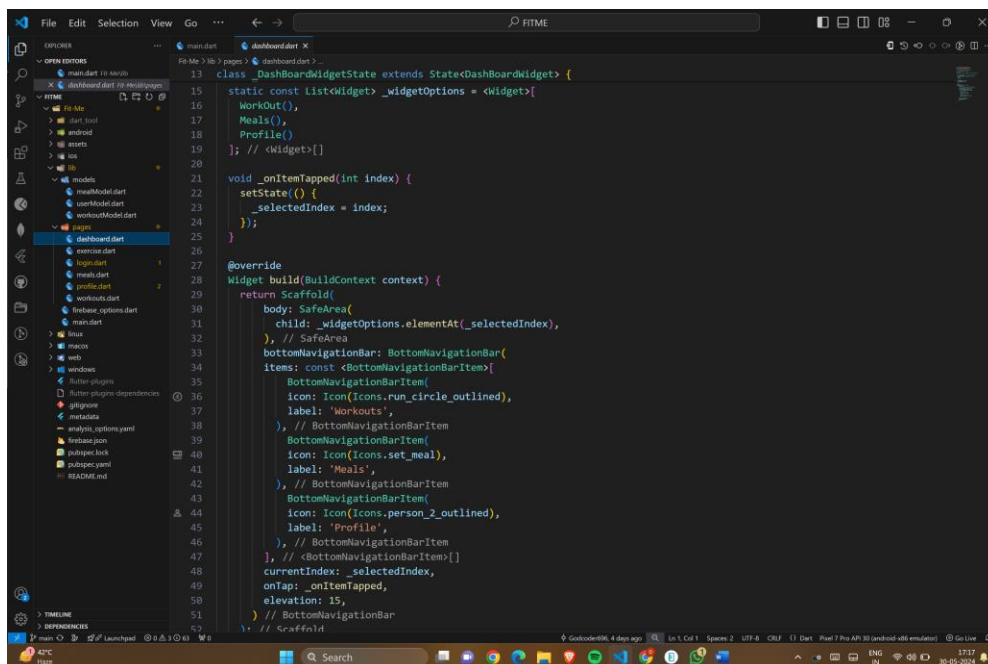
```

1  Godcoder696, 2 days ago | 1 author (Godcoder696)
2  Godcoder696, 2 days ago • updated!
3
4  Godcoder696, 2 days ago | 1 author (Godcoder696)
5  class WorkoutModel{
6    String? workoutName;
7    String? url;
8
9    WorkoutModel({this.workoutName, this.url});

```

Pages:

1. dashboard.dart

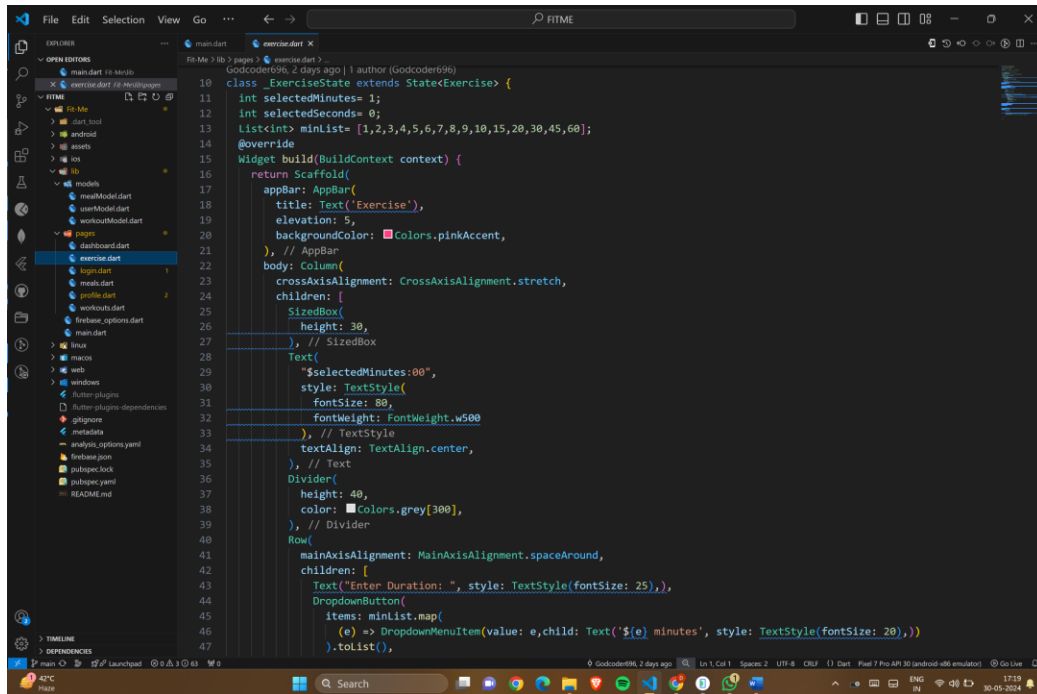


```

13 class _DashboardWidgetState extends StatefulWidget {
14   static const List<Widget> _widgetOptions = <Widget>[
15     Workout(),
16     Meals(),
17     Profile(),
18   ]; // <Widget>[]
19
20   void onItemTapped(int index) {
21     setState(() {
22       _selectedIndex = index;
23     });
24   }
25
26   @override
27   Widget build(BuildContext context) {
28     return Scaffold(
29       body: SafeArea(
30         child: _widgetOptions.elementAt(_selectedIndex),
31       ), // SafeArea
32       bottomNavigationBar: BottomNavigationBar(
33         items: const <BottomNavigationBarItem>[
34           BottomNavigationBarItem(
35             icon: Icon(Icons.run_circle_outlined),
36             label: 'Workouts',
37           ), // BottomNavigationBarItem
38           BottomNavigationBarItem(
39             icon: Icon(Icons.set_meal),
40             label: 'Meals',
41           ), // BottomNavigationBarItem
42           BottomNavigationBarItem(
43             icon: Icon(Icons.person_2_outlined),
44             label: 'Profile',
45           ), // BottomNavigationBarItem
46         ], // <BottomNavigationBarItem>[]
47         currentIndex: _selectedIndex,
48         onTap: onItemTapped,
49         elevation: 15,
50       ), // BottomNavigationBar
51     ); // Scaffold

```


2. exercise.dart

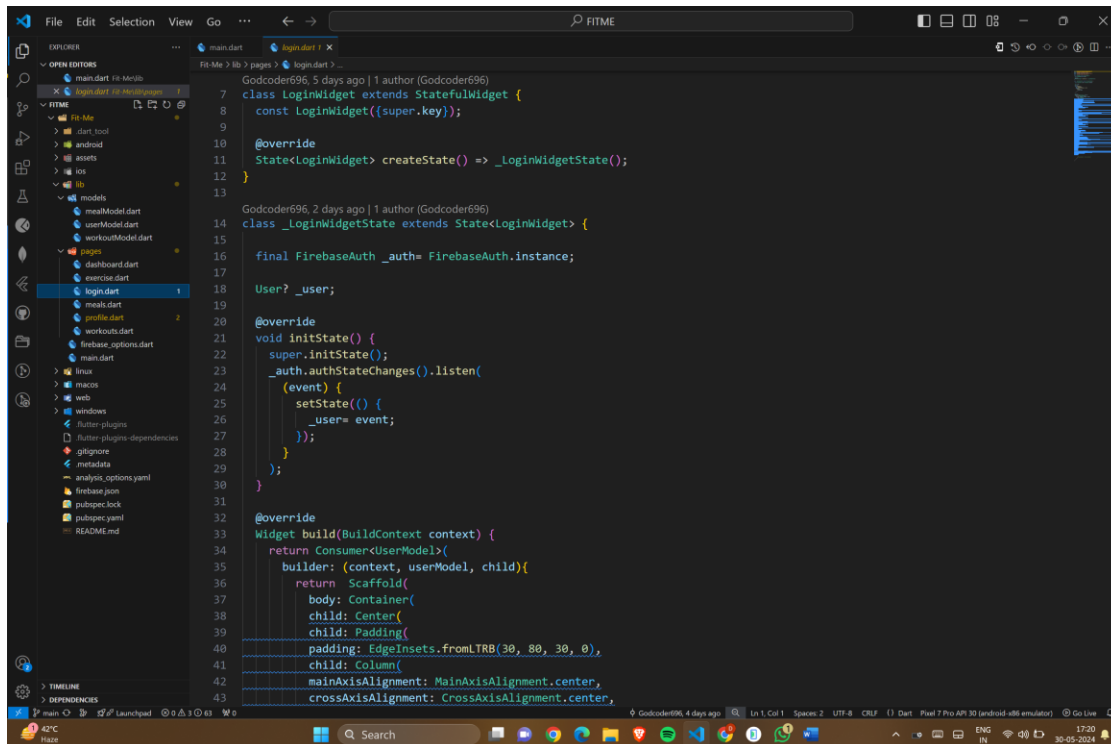


```

10 class _ExerciseState extends StatefulWidget {
11   int selectedMinutes= 1;
12   int selectedSeconds= 0;
13   List<int> minList= [1,2,3,4,5,6,7,8,9,10,15,20,30,45,60];
14   @override
15   Widget build(BuildContext context) {
16     return Scaffold(
17       appBar: AppBar(
18         title: Text('Exercise'),
19         elevation: 5,
20         backgroundColor: Colors.pinkAccent,
21       ), // AppBar
22       body: Column(
23         crossAxisAlignment: CrossAxisAlignment.stretch,
24         children: [
25           SizedBox(
26             height: 30,
27           ), // SizedBox
28           Text(
29             "$selectedMinutes:00",
30             style: TextStyle(
31               fontSize: 80,
32               fontWeight: FontWeight.w500
33             ), // TextStyle
34             textAlign: TextAlign.center,
35           ), // Text
36           Divider(
37             height: 40,
38             color: Colors.grey[300],
39           ), // Divider
40           Row(
41             mainAxisAlignment: MainAxisAlignment.spaceAround,
42             children: [
43               Text("Enter Duration: ", style: TextStyle(fontSize: 25)),
44               DropdownButton<int>(
45                 items: minList.map(
46                   (e) => DropdownMenuItem(value: e, child: Text('$e minutes', style: TextStyle(fontSize: 20))),
47                 ).toList(),

```

3. login.dart



```

7 class LoginWidget extends StatefulWidget {
8   const LoginWidget({super.key});
9
10   @override
11   State<LoginWidget> createState() => _LoginWidgetState();
12 }
13
14 class _LoginWidgetState extends State<LoginWidget> {
15
16   final FirebaseAuth _auth= FirebaseAuth.instance;
17
18   User? _user;
19
20   @override
21   void initState() {
22     super.initState();
23     _auth.authStateChanges().listen(
24       (event) {
25         setState(() {
26           _user= event;
27         });
28       }
29     );
30   }
31
32   @override
33   Widget build(BuildContext context) {
34     return Consumer<UserModel>({
35       builder: (context, userModel, child){
36         return Scaffold(
37           body: Container(
38             child: Center(
39               child: Padding(
40                 padding: EdgeInsets.fromLTRB(30, 80, 30, 0),
41               child: Column(
42                 mainAxisAlignment: MainAxisAlignment.center,
43                 crossAxisAlignment: CrossAxisAlignment.center,

```

[illegible]

The screenshot shows the Android Studio IDE with a Dart file named `login.dart` open. The file contains the following code:

```

class LoginWidgetState extends StatelessWidget {
  Widget build(BuildContext context) {
    GoogleAuthProvider _googleAuthProvider= GoogleAuthProvider();
    await _auth.signInWithProvider(_googleAuthProvider);

    catch(e){
      print(e);
    }

    print(user!.email);
    userModel.user= user;
    if(userModel.user!=null){
      Navigator.pushReplacementNamed(context, '/dashboard');
      print(userModel.user!.email);
    }
  },
  icon: Icon(Icons.login),
  label: Text(
    'Sign in with Google',
    style: TextStyle(fontSize: 18),
  ), // Text
  // ElevatedButton.icon
  SizedBox(height: 20),
  Image.asset('assets/man.png',height: 250),
  ), // Column
  // Padding
  // Center
  decoration: BoxDecoration(
    image: DecorationImage(
      image: AssetImage('assets/hero.jpg'),
      fit: BoxFit.cover
    ), // DecorationImage
    // BoxDecoration
  ), // Container
  // backgroundColor: Colors.pinkAccent,
); // Scaffold
); // Consumer

```

The IDE interface shows the project structure on the left, the code editor in the center, and a toolbar at the top. The status bar at the bottom indicates the device is a Pixel 7 Pro API 30 (android-x86_64) running on a Linux host.

4. meals.dart

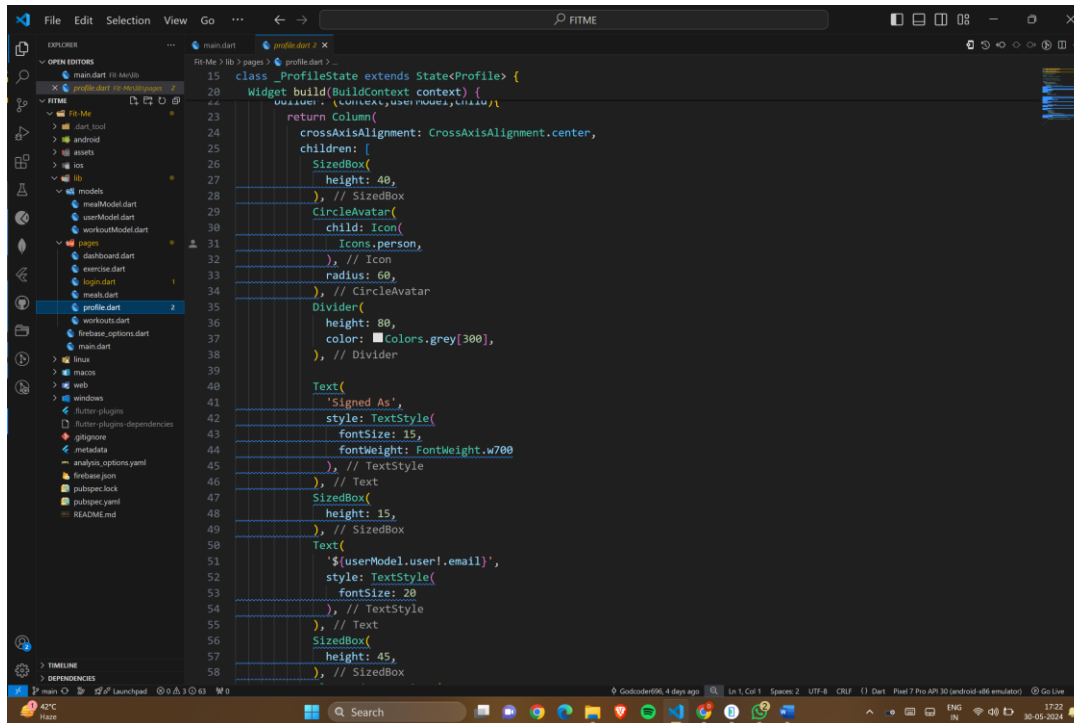
[illegible]

```

class MealsState extends State<Meals> {
  Widget build(BuildContext context) {
    return ListView.builder(
      itemCount: meals.length,
      itemBuilder: (context, index) {
        return Card(
          child: Column(
            children: [
              NetworkImage(
                url: meals[index].url,
              ),
              Text(
                meals[index].mealLabel,
              ),
            ],
          ),
        );
      },
    );
  }
}

```

5. profile.dart

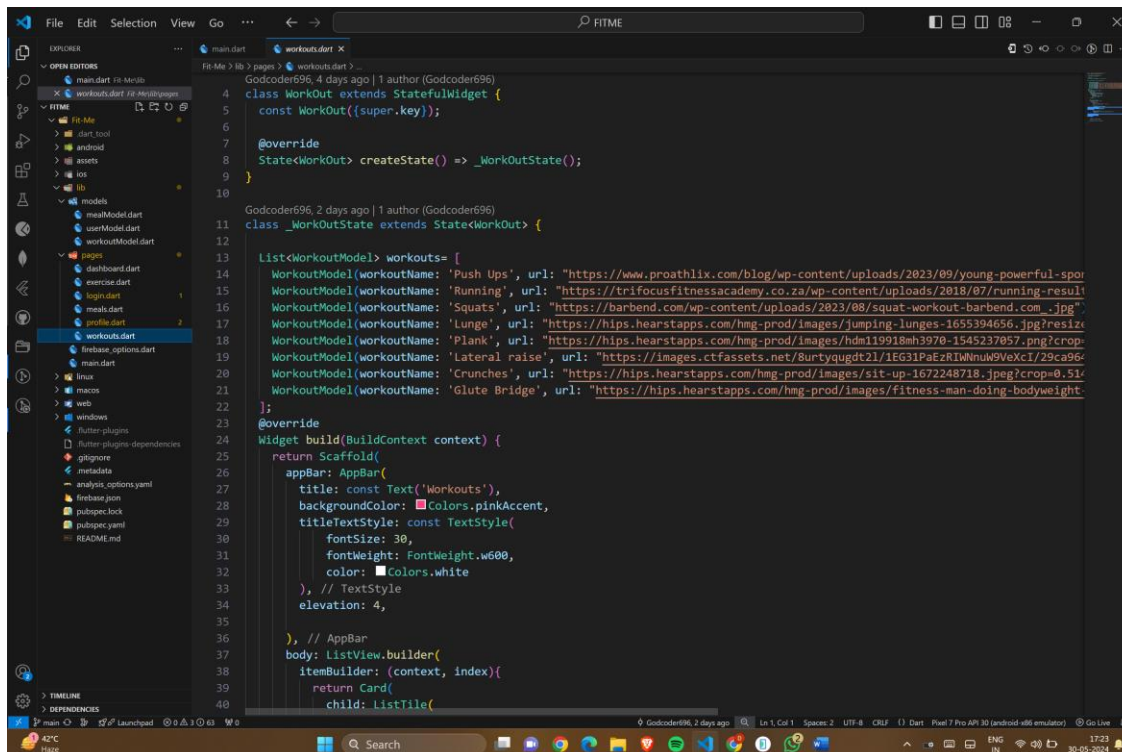


```

15 class _ProfileState extends State<Profile> {
16   Widget build(BuildContext context) {
17     return Column(
18       mainAxisAlignment: MainAxisAlignment.center,
19       children: [
20         SizedBox(
21           height: 40,
22         ), // SizedBox
23         CircleAvatar(
24           child: Icon(
25             Icons.person,
26           ), // Icon
27         ), // CircleAvatar
28         Divider(
29           height: 80,
30           color: Colors.grey[300],
31         ), // Divider
32         Text(
33           'Signed As',
34           style: TextStyle(
35             fontSize: 15,
36             fontWeight: FontWeight.w700
37           ), // TextStyle
38         ), // Text
39         SizedBox(
40           height: 15,
41         ), // SizedBox
42         Text(
43           '${userModel.user!.email}',
44           style: TextStyle(
45             fontSize: 20
46           ), // TextStyle
47         ), // Text
48         SizedBox(
49           height: 45,
50         ), // SizedBox
51       ],
52     );
53   }
54 }

```

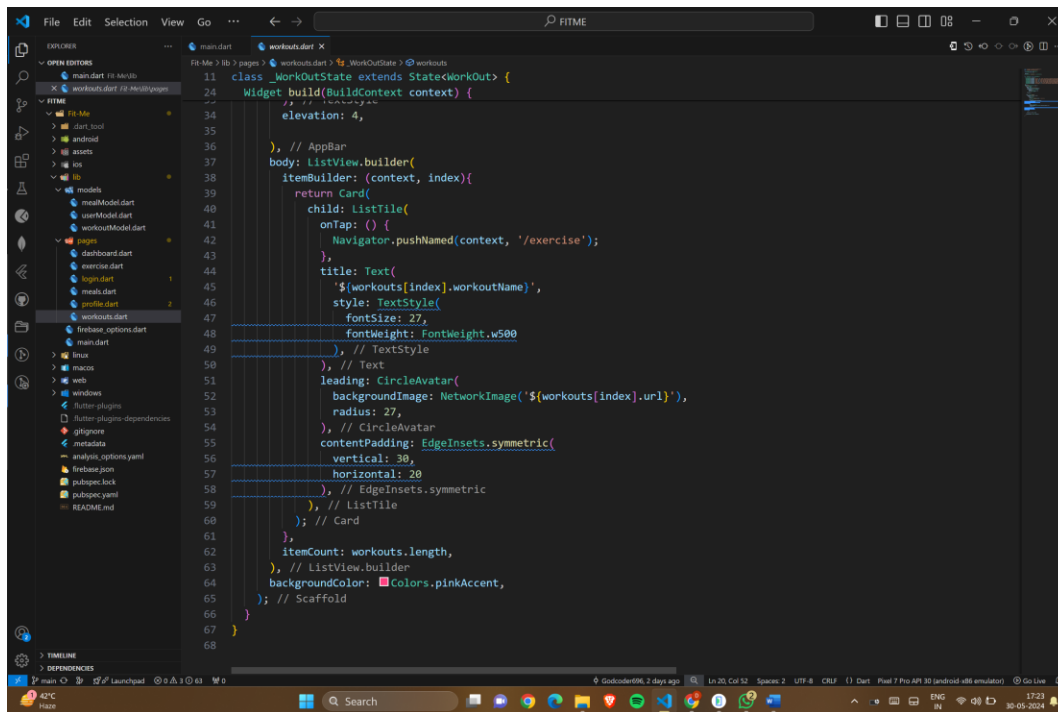
6. workout.dart



```

4 class Workout extends StatefulWidget {
5   const Workout({super.key});
6
7   @override
8   State<Workout> createState() => _WorkOutState();
9 }
10
11 class _WorkOutState extends State<Workout> {
12
13   List<WorkoutModel> workouts= [
14     WorkoutModel(workoutName: 'Push Ups', url: "https://www.proathlix.com/blog/wp-content/uploads/2023/09/young-powerful-sportswoman-doing-push-ups-against-a-white-wall-1000x666.jpg"),
15     WorkoutModel(workoutName: 'Running', url: "https://trifocusfitnessacademy.co.za/wp-content/uploads/2018/07/running-results.jpg"),
16     WorkoutModel(workoutName: 'Squats', url: "https://barbend.com/wp-content/uploads/2023/08/squat-workout-barbend.com .jpg"),
17     WorkoutModel(workoutName: 'Lunge', url: "https://hips.hearstapps.com/hmg-prod/images/jumping-lunges-1655394656.jpg?resizedimage=1"),
18     WorkoutModel(workoutName: 'Plank', url: "https://hips.hearstapps.com/hmg-prod/images/hdm119918mh3970-1545237057.png?crop=0.51x1,1y1o"),
19     WorkoutModel(workoutName: 'Lateral raise', url: "https://images.ctfassets.net/8urtyqugdt21/1EG31PaEzRiWnuuM9VeXc1/29ca96f1e1e1e1e1e1e1e1e1e1e1e1e1/Lateral%20raise.jpg"),
20     WorkoutModel(workoutName: 'Crunches', url: "https://hips.hearstapps.com/hmg-prod/images/sit-up-1672248718.jpeg?crop=0.51x1,1y1o"),
21     WorkoutModel(workoutName: 'Glute Bridge', url: "https://hips.hearstapps.com/hmg-prod/images/fitness-man-doing-bodyweight-glute-bridge-1000x666.jpg"),
22   ];
23
24   @override
25   Widget build(BuildContext context) {
26     return Scaffold(
27       appBar: AppBar(
28         title: const Text('Workouts'),
29         backgroundColor: Colors.pinkAccent,
30         titleTextStyle: const TextStyle(
31           fontSize: 30,
32           fontWeight: FontWeight.w600,
33           color: Colors.white
34         ), // TextStyle
35       ), // AppBar
36       body: ListView.builder(
37         itemBuilder: (context, index){
38           return Card(
39             child: ListTile(

```

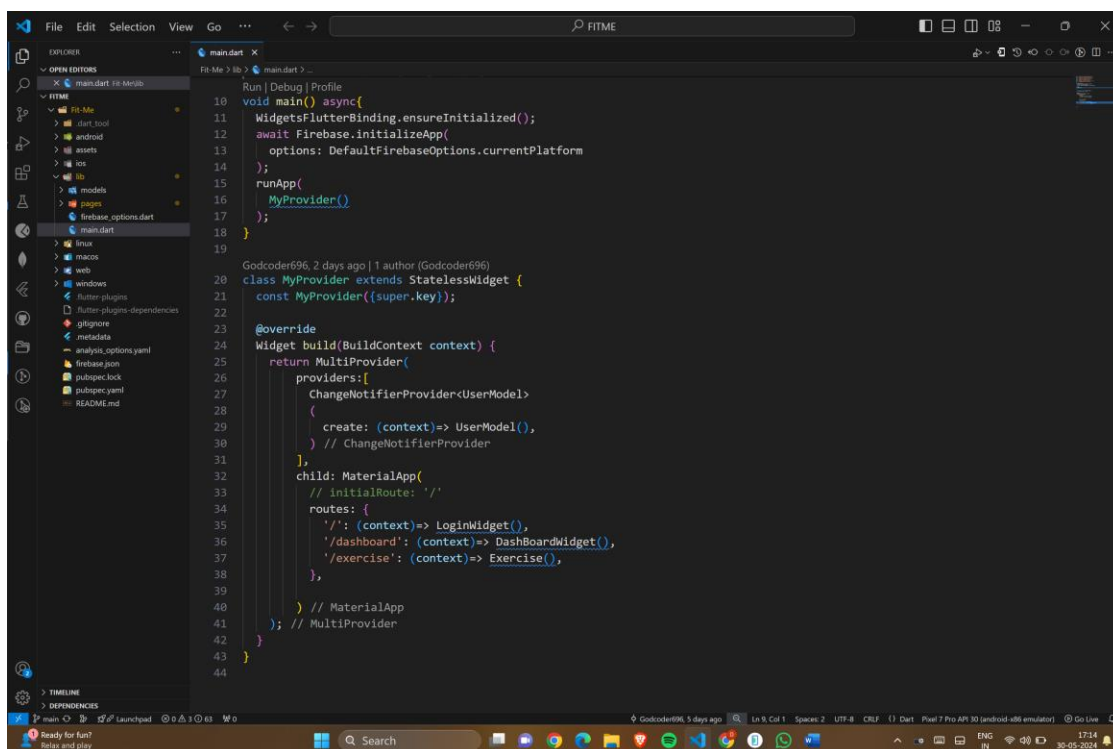


```

11 class _WorkOutState extends StatefulWidget {
12   Widget build(BuildContext context) {
13     // ...
14     elevation: 4,
15   }, // AppBar
16   body: ListView.builder(
17     itemBuilder: (context, index){
18       return Card(
19         child: ListTile(
20           onTap: () {
21             Navigator.pushNamed(context, '/exercise');
22           },
23           title: Text(
24             '${workouts[index].workoutName}',
25             style: TextStyle(
26               fontSize: 27,
27               fontWeight: FontWeight.w500
28             ), // TextStyle
29           ), // Text
30           leading: CircleAvatar(
31             backgroundImage: NetworkImage('${workouts[index].url}'),
32             radius: 27,
33           ), // CircleAvatar
34           contentPadding: EdgeInsets.symmetric(
35             vertical: 30,
36             horizontal: 20
37           ), // EdgeInsets.symmetric
38         ), // ListTile
39       ); // Card
40     },
41     itemCount: workouts.length,
42   ), // ListView.builder
43   backgroundColor: Colors.pinkAccent,
44 ); // Scaffold
45 }
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68

```

main.dart

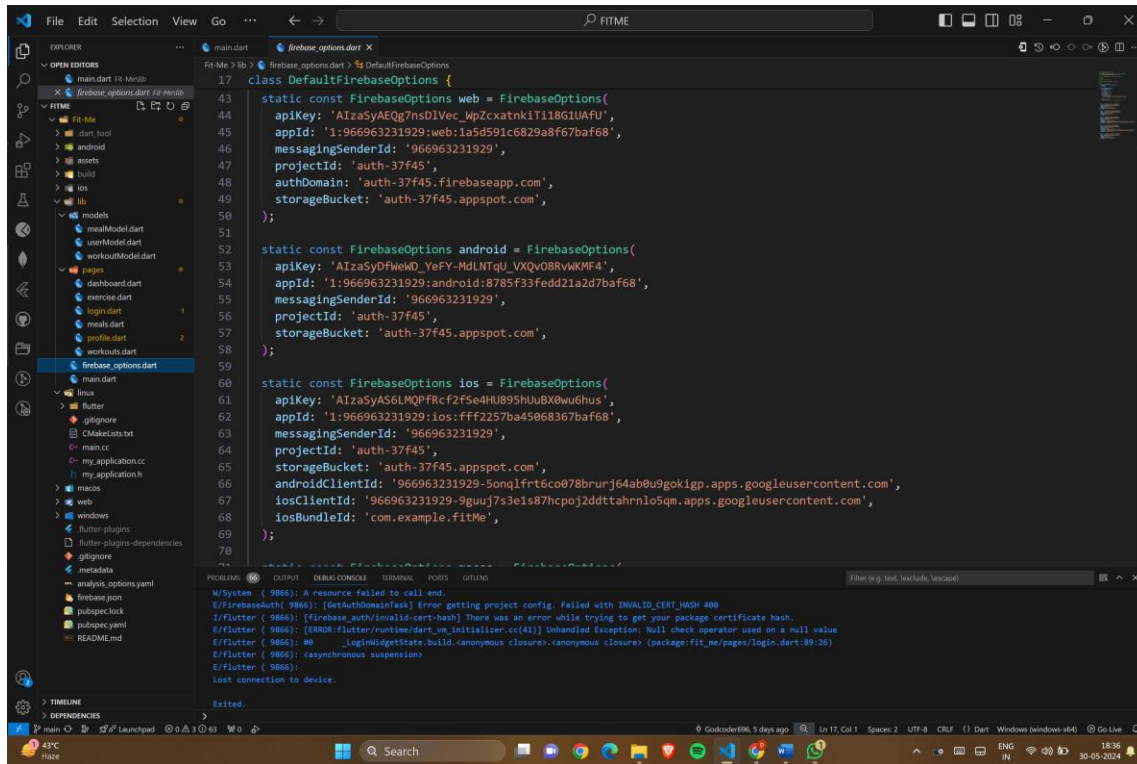


```

10 void main() async{
11   WidgetsFlutterBinding.ensureInitialized();
12   await Firebase.initializeApp(
13     options: DefaultFirebaseOptions.currentPlatform
14   );
15   runApp(
16     MyProvider()
17   );
18 }
19
20 class MyProvider extends StatelessWidget {
21   const MyProvider({super.key});
22
23   @override
24   Widget build(BuildContext context) {
25     return MultiProvider(
26       providers: [
27         ChangeNotifierProvider<UserModel>
28         (
29           create: (context)=> UserModel(),
30           // ChangeNotifierProvider
31         ),
32       ],
33       child: MaterialApp(
34         // initialRoute: '/'
35         routes: {
36           '/': (context)=> LoginWidget(),
37           '/dashboard': (context)=> DashBoardWidget(),
38           '/exercise': (context)=> Exercise(),
39         },
40       ), // MaterialApp
41     ); // MultiProvider
42   }
43 }
44

```

firebase_options.dart



5. Implementation And Testing

1.Feature Planning:

Define the features and functionalities you want to implement in the FitMe app, such as user profiles, workout tracking, nutrition tracking, goal setting, etc.

Break down each feature into smaller tasks or user stories to facilitate development and testing.

Implementation:

Use Flutter and Dart to write code for each feature, following best practices for code organization, readability, and maintainability.

Utilize Flutter packages or custom widgets to build UI components, handle user interactions, and manage state within the app.

Implement backend services or APIs to support features like user authentication, data storage, and synchronization (if applicable).

Integrate any third-party services or APIs, such as fitness trackers or nutritional databases, into the app as needed.

2.Unit Testing:

Write unit tests for individual functions, methods, or classes to verify that they behave as expected.

Use tools like flutter_test package or mockito for mocking dependencies and test package for writing test cases.

Test edge cases and error conditions to ensure robustness and reliability.

3.Integration Testing:

Perform integration tests to verify that different parts of the app work together correctly.

Test user flows, such as registration, login, profile setup, workout tracking, etc., to ensure a seamless user experience.

Use tools like Flutter's integration_test package or frameworks like Flutter Driver for automated UI testing.

User Acceptance Testing (UAT):

Conduct UAT with real users or stakeholders to gather feedback and validate that the app meets their expectations.

Address any issues or usability concerns identified during UAT and incorporate feedback into future iterations.

Performance Testing:

Measure and optimize the app's performance, including startup time, responsiveness, memory usage, and battery consumption.

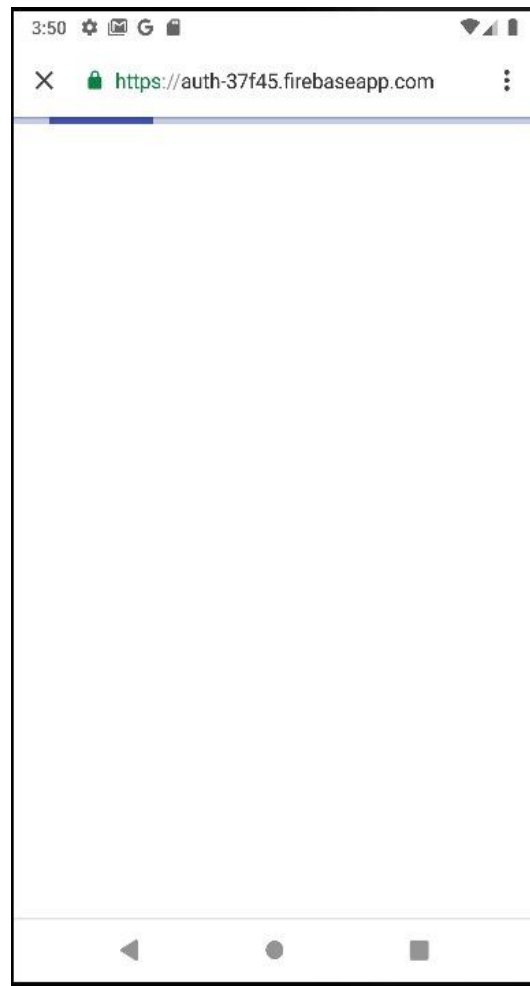
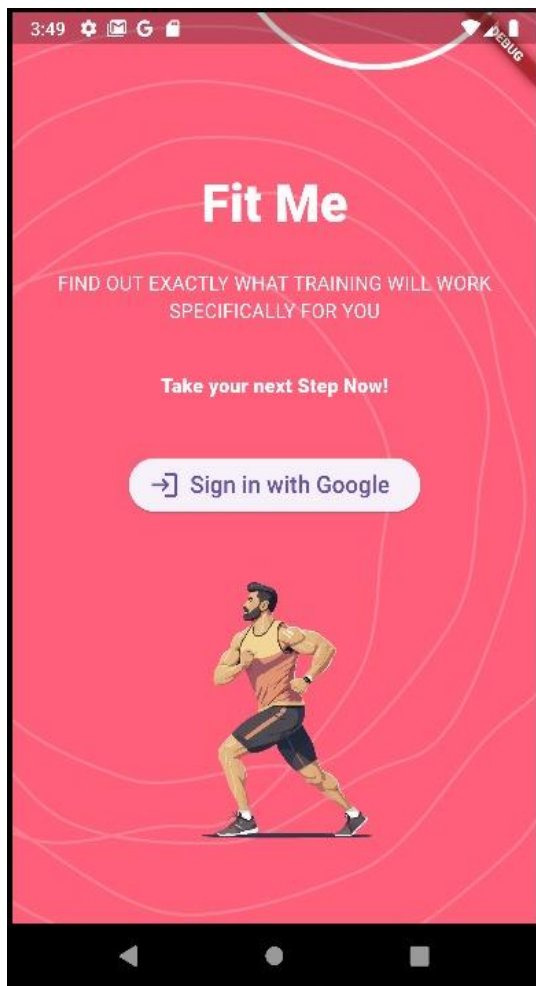
Use profiling tools like Flutter DevTools to identify performance bottlenecks and optimize code accordingly.

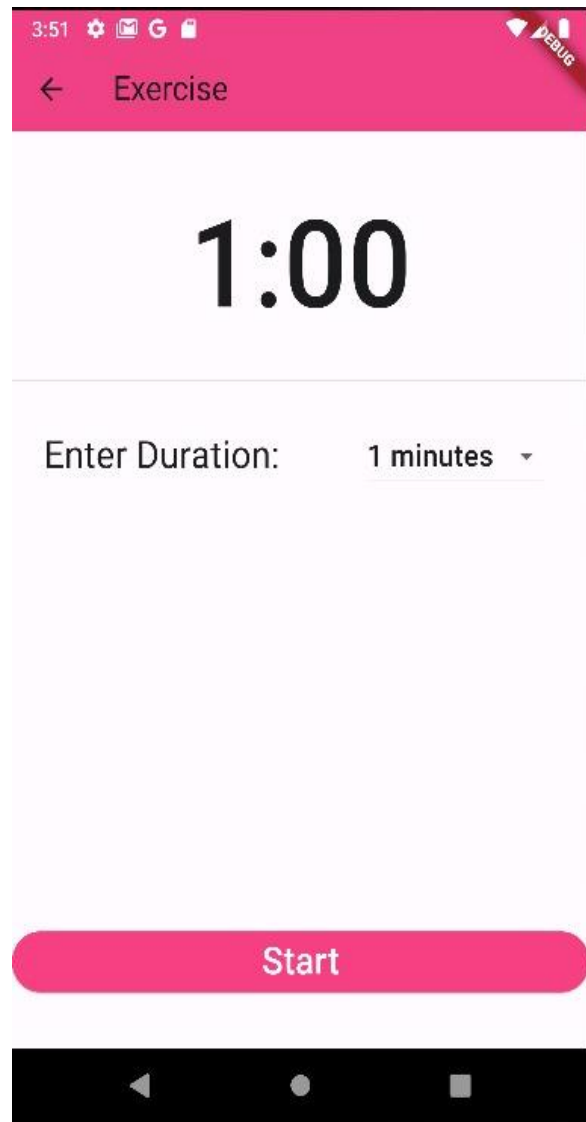
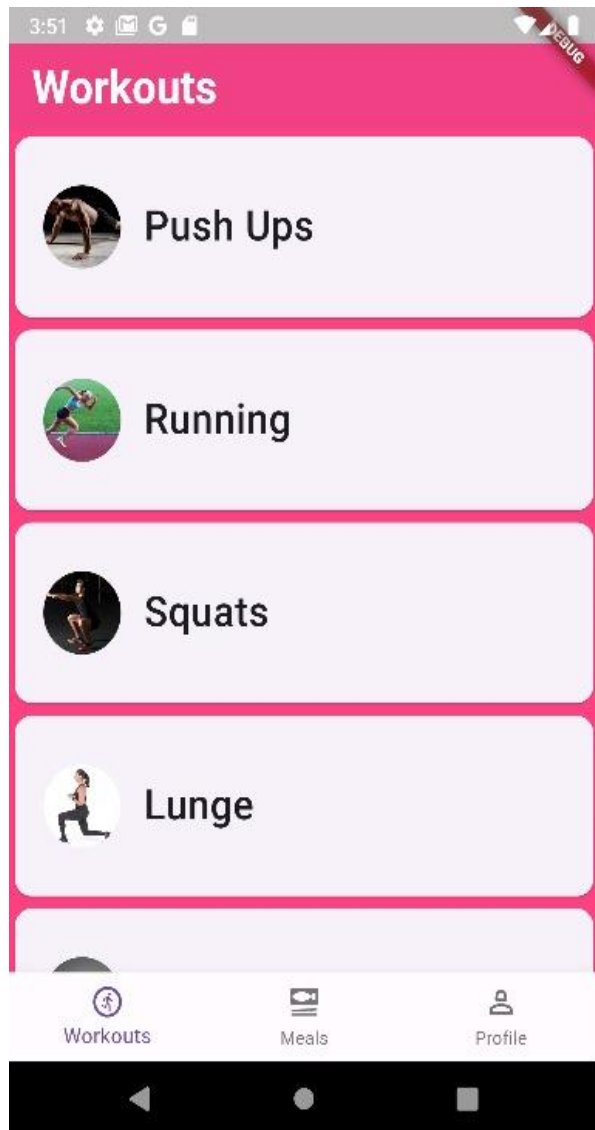
4.Accessibility Testing:

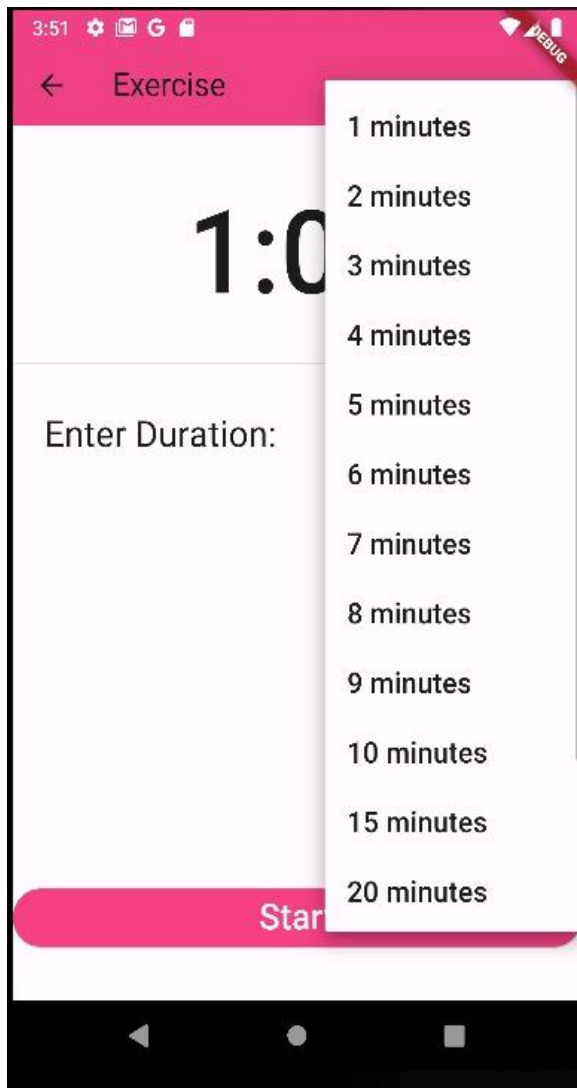
Ensure that the app is accessible to users with disabilities by testing with screen readers, keyboard navigation, and other assistive technologies.

6. Output(Screenshots)

PROTOTYPE AND DESIGN







7. Conclusion and future scope

Conclusion:

The FitMe app, developed in Flutter, serves as a comprehensive platform for users to track their fitness progress, manage their workouts and nutrition, and receive personalized recommendations tailored to their goals. Through careful planning, implementation, and testing, the app provides a seamless and intuitive user experience, empowering individuals to lead healthier lifestyles.

Future Scope:

While the FitMe app has reached a significant milestone in its development, there are several avenues for future enhancement and expansion:

Fit-Me is a health well-being application and in its further versions we tend to introduce the AI support and chat box for enabling much more accuracy of advices and exercises with routine checkups and tracking. Further, more we aim to bring features for personalized and customized workouts that suits your day-to-day routine.

Fit-Me also aims to provide medical and emergency support for the old age people.

GITHUB LINK ON THE NEXT PAGE:

GITHUB LINK:

<https://github.com/geetkhanna223/FitMe-App>

The PPT for the project is also uploaded in this repository along with the code.