PROJECT-TITLE: “FitMe360”

1. App idea and objectives:

We have made FitMe360 application with the help of flutter framework.The purpose of this application is to provide a how to make you fit easily.We have made a simple user friendly interface.The main objective of this app is to provide a balance chart and diet plan graph to user and pedometer timer which tell how much step you have done?.The functional requirement includes database managing,provide tracker,graph for visualization,alarm for notification,notepad and diet plan and fact of foods are also provided.

1. Functional description:

We used shared\_preferences package to store date ,youtube\_player widget ,container,text-widget,elevated button,image widget,sound widget,form widget and other widgets and many other widgets.Here I am highlighting important widgets that I have used:

1. **Widgets:**

* Table:Table is another widget that is important to give data in rows and columns.
* Text:Text widget is another important widget that is important to create a text-field print text.
* Image:Image widget is used to import image I have use list and column widget to align image.
* Button is another important widget used for clicking functionality elevated button used
* Tab Naviation :This widget used to goto different screen whenyou want to see and thing one tab to another.I use three tab one home,foot\_step\_timer and notepad.
* Drawer :This widget used for drawer navigation it include hamburger which included Dashboard,Food,Profile,Signout.

1. UI/UX Code:

1**.main.dart:**

|  |
| --- |
| import 'package:flutter/material.dart';  import 'splash\_screen.dart';  import 'login.dart';  import 'signup\_page.dart';  void **main**() {  **runApp**(const **FitmeApp**());}  class **FitmeApp** extends **StatelessWidget** {  const **FitmeApp**({super.key});  @override  **Widget** **build**(**BuildContext** context) {  return **MaterialApp**(  title: 'Fitme360',  debugShowCheckedModeBanner: false,  theme: **ThemeData**(  primarySwatch: **Colors**.blue,),  home: const **SplashScreen**(),  routes: {  '/login': (context) => const **LoginPage**(),  '/signup': (context) => const **SignUpPage**(),  '/splash': (context) => const **SplashScreen**(),},);}} |

2.**database\_helper.dart**

|  |
| --- |
| import 'dart:async';  import 'package:sqflite/sqflite.dart';  import 'package:path/path.dart';  class DatabaseHelper {  static final DatabaseHelper \_instance = DatabaseHelper.\_internal();  factory DatabaseHelper() => \_instance;  static Database? \_db;  DatabaseHelper.\_internal();  Future<Database> get db async {  if (\_db != null) return \_db!;  \_db = await \_initDb();  return \_db!; }  Future<Database> \_initDb() async {  String path = join(await getDatabasesPath(), 'fitme360.db');  return await openDatabase(path, version: 1, onCreate: \_onCreate);}  void \_onCreate(Database db, int version) async {  await db.execute('''  CREATE TABLE users (  id INTEGER PRIMARY KEY AUTOINCREMENT,  name TEXT,  fathername TEXT,  gender TEXT,  weight REAL,  height REAL,  age INTEGER,  dob TEXT,  email TEXT UNIQUE,  password TEXT) ''');  await db.execute('''  CREATE TABLE todos (  id INTEGER PRIMARY KEY AUTOINCREMENT,  task TEXT ) ''');}  Future<int> insertUser(Map<String, dynamic> user) async {  var dbClient = await db;  return await dbClient.insert('users', user);}  Future<Map<String, dynamic>?> getUser(String email, String password) async {  var dbClient = await db;  List<Map<String, dynamic>> result = await dbClient.query(  'users',  where: 'email = ? AND password = ?',  whereArgs: [email, password],);  if (result.isNotEmpty) return result.first;  return null;}  Future<void> createTodoTable() async {  var dbClient = await db;  await dbClient.execute('''  CREATE TABLE IF NOT EXISTS todos ( id INTEGER PRIMARY KEY AUTOINCREMENT,  task TEXT) ''');}  Future<int> insertTodo(String task) async {  var dbClient = await db;  return await dbClient.insert('todos', {'task': task}); }  Future<List<Map<String, dynamic>>> getTodos() async {  var dbClient = await db;  return await dbClient.query('todos'); }  Future<void> updateTodo(int id, String newTask) async {  var dbClient = await db;  await dbClient.update(  'todos',  {'task': newTask},  where: 'id = ?',  whereArgs: [id], ); }  Future<void> deleteTodo(int id) async {  var dbClient = await db;  await dbClient.delete(  'todos',  where: 'id = ?',  whereArgs: [id],);}  Future<void> deleteAllTodos() async {  var dbClient = await db;  await dbClient.delete('todos');  }} |

3.**dashboard.dart:**

|  |
| --- |
| import 'dart:io';  import 'package:flutter/material.dart';  import 'package:shared\_preferences/shared\_preferences.dart';  import 'homescreen.dart';  import 'footsteptimer.dart';  import 'rememberlist.dart';  import 'login.dart';  import 'dietplan.dart';  import 'profilestatus.dart';  class DashboardPage extends StatefulWidget {  const DashboardPage({super.key});  @override  State<DashboardPage> createState() => \_DashboardPageState();  }  class \_DashboardPageState extends State<DashboardPage> {  int \_currentIndex = 0;  final List<Widget> \_pages = const [  HomeScreen(),  FootstepTimer(),  RememberList(),  ];  @override  Widget build(BuildContext context) {  return Scaffold(  backgroundColor: Colors.grey[100],  appBar: AppBar(  backgroundColor: Colors.white,  elevation: 2,  shadowColor: Colors.black26,  title: const Text(  'Dashboard',  style: TextStyle(  color: Colors.black87,  fontWeight: FontWeight.w600,  letterSpacing: 0.5,  ),  ),  iconTheme: const IconThemeData(color: Colors.black87),  ),  drawer: buildAppDrawer(context),  body: Padding(  padding: const EdgeInsets.symmetric(horizontal: 16, vertical: 12),  child: \_pages[\_currentIndex],  ),  bottomNavigationBar: Container(  decoration: BoxDecoration(  color: Colors.white,  boxShadow: [  BoxShadow(  color: Colors.black.withOpacity(0.1),  blurRadius: 12,  offset: const Offset(0, -3),  ),  ],  borderRadius: const BorderRadius.only(  topLeft: Radius.circular(16),  topRight: Radius.circular(16),  ),  ),  child: SafeArea(  top: false,  child: BottomNavigationBar(  currentIndex: \_currentIndex,  onTap: (index) {  setState(() {  \_currentIndex = index;  });  },  backgroundColor: Colors.white,  selectedItemColor: Colors.blue.shade700,  unselectedItemColor: Colors.grey.shade600,  showSelectedLabels: false, // remove label text  showUnselectedLabels: false,  type: BottomNavigationBarType.fixed,  elevation: 0,  items: const [  BottomNavigationBarItem(  icon: Icon(Icons.home\_outlined, size: 32),  activeIcon: Icon(Icons.home, size: 36),  label: '',  ),  BottomNavigationBarItem(  icon: Icon(Icons.directions\_run\_outlined, size: 32),  activeIcon: Icon(Icons.directions\_run, size: 36),  label: '',  ),  BottomNavigationBarItem(  icon: Icon(Icons.note\_outlined, size: 32),  activeIcon: Icon(Icons.note, size: 36),  label: '',  ),  ],  ),  ),  ),  );  }  }  // Reusable drawer widget with profile image and name  Widget buildAppDrawer(BuildContext context) {  return Drawer(  child: ListView(  padding: EdgeInsets.zero,  children: [  DrawerHeader(  decoration: const BoxDecoration(color: Colors.blue),  child: FutureBuilder<Map<String, String>>(  future: \_getUserInfo(),  builder: (context, snapshot) {  final name = snapshot.data?['name'] ?? 'User';  final profileImage = snapshot.data?['profile\_image'];  ImageProvider imageProvider;  if (profileImage != null && profileImage.isNotEmpty && File(profileImage).existsSync()) {  imageProvider = FileImage(File(profileImage));  } else {  imageProvider = const AssetImage('assets/content/profile.jpg');  }  return Column(  crossAxisAlignment: CrossAxisAlignment.start,  children: [  CircleAvatar(  radius: 36, // increased size for profile image  backgroundImage: imageProvider,  backgroundColor: Colors.white,  // Add subtle shadow around avatar for depth  child: Container(  decoration: BoxDecoration(  shape: BoxShape.circle,  boxShadow: [  BoxShadow(  color: Colors.black26,  blurRadius: 8,  offset: Offset(0, 2),  ),  ],  ),  ),  ),  const SizedBox(height: 16), // increased spacing  Text(  name,  style: const TextStyle(  color: Colors.white,  fontSize: 20,  fontWeight: FontWeight.bold,  letterSpacing: 0.3,  ),  ),  ],  );  },  ),  ),  ListTile(  leading: const Icon(Icons.dashboard),  title: const Text('Dashboard'),  onTap: () {  Navigator.pushReplacement(  context,  MaterialPageRoute(builder: (context) => const DashboardPage()),  );  },  ),  ListTile(  leading: const Icon(Icons.fastfood),  title: const Text('Food'),  onTap: () {  Navigator.pushReplacement(  context,  MaterialPageRoute(builder: (context) => const Dietplan()),  );  },  ),  ListTile(  leading: const Icon(Icons.person),  title: const Text('Profile'),  onTap: () {  Navigator.pushReplacement(  context,  MaterialPageRoute(builder: (context) => const STATUS()),  );  },  ),  ListTile(  leading: const Icon(Icons.logout),  title: const Text('Sign out'),  onTap: () {  Navigator.pushReplacement(  context,  MaterialPageRoute(builder: (context) => const LoginPage()),  );  },  ),  ],  ),  );  }  // Fetch both name and profile image from SharedPreferences  Future<Map<String, String>> \_getUserInfo() async {  final prefs = await SharedPreferences.getInstance();  return {  'name': prefs.getString('name') ?? 'User',  'profile\_image': prefs.getString('profile\_image') ?? '',  };  } |

4.**alarmworkout.dart:**

|  |
| --- |
| import 'package:flutter/material.dart';  import 'dart:async';  import 'package:audioplayers/audioplayers.dart';  class AlarmWorkoutScreen extends StatefulWidget {  const AlarmWorkoutScreen({super.key});  @override  State<AlarmWorkoutScreen> createState() => \_AlarmWorkoutScreenState();  }  class \_AlarmWorkoutScreenState extends State<AlarmWorkoutScreen> {  TimeOfDay? \_selectedTime;  Timer? \_timer;  final AudioPlayer \_audioPlayer = AudioPlayer();  bool \_isAlarmPlaying = false;    final List<String> \_tones = ['alarm.mp3', 'beep.mp3', 'ringtone.mp3'];  String \_selectedTone = 'alarm.mp3';    // Initial and final time dropdown inputs  int? \_initialHour = 7;  int? \_initialMinute = 0;  String \_initialAmPm = 'AM';  int? \_finalHour = 9;  int? \_finalMinute = 0;  String \_finalAmPm = 'AM';  int \_convertTo24Hour(int hour, String ampm) {  if (ampm == 'AM') {  return hour == 12 ? 0 : hour;  } else {  return hour == 12 ? 12 : hour + 12;  }}  Future<void> \_setAlarmFromInputs() async {  final initial24Hour = \_convertTo24Hour(\_initialHour ?? 7, \_initialAmPm);  final initialTimeOfDay = TimeOfDay(hour: initial24Hour, minute: \_initialMinute ?? 0);  final now = DateTime.now();  DateTime alarmDateTime = DateTime(  now.year,  now.month,  now.day,  initialTimeOfDay.hour,  initialTimeOfDay.minute,);  if (alarmDateTime.isBefore(now)) {  alarmDateTime = alarmDateTime.add(const Duration(days: 1)); }  final difference = alarmDateTime.difference(now);  setState(() {  \_selectedTime = initialTimeOfDay;  \_timer?.cancel();  \_timer = Timer(difference, \_triggerAlarm);  });  \_showSnackBar("Alarm set for ${\_selectedTime!.format(context)}");  }  Future<void> \_pickTime() async {  final TimeOfDay? picked = await showTimePicker(  context: context,  initialTime: TimeOfDay.now(), );  if (picked != null) {  final now = DateTime.now();  DateTime alarmTime = DateTime(  now.year,  now.month,  now.day,  picked.hour,  picked.minute, );  if (alarmTime.isBefore(now)) {  alarmTime = alarmTime.add(const Duration(days: 1)); }  final difference = alarmTime.difference(now);  setState(() {  \_selectedTime = picked;  \_timer?.cancel();  \_timer = Timer(difference, \_triggerAlarm);  });  \_showSnackBar("Alarm set for ${picked.format(context)}");}  }  Future<void> \_triggerAlarm() async {  setState(() {  \_isAlarmPlaying = true;  });  \_showSnackBar("⏰ Alarm ringing!");  try {  await \_audioPlayer.setReleaseMode(ReleaseMode.loop);  await \_audioPlayer.setVolume(1.0);  await \_audioPlayer.play(AssetSource('sounds/$\_selectedTone'));  } catch (e) {  debugPrint("Error playing alarm sound: $e");  \_showSnackBar("Error playing alarm sound.");  setState(() {  \_isAlarmPlaying = false; }); } }  Future<void> \_stopAlarm() async {  await \_audioPlayer.stop();  setState(() {  \_isAlarmPlaying = false;  });  \_showSnackBar("Alarm stopped.");  }  Future<void> \_playTestSound() async {  try {  await \_audioPlayer.setReleaseMode(ReleaseMode.stop);  await \_audioPlayer.setVolume(1.0);  await \_audioPlayer.play(AssetSource('sounds/$\_selectedTone'));  \_showSnackBar("Playing test sound...");  } catch (e) {  debugPrint("Error playing sound: $e");  \_showSnackBar("Failed to play sound."); } }  void \_showSnackBar(String msg) {  if (!mounted) return;  ScaffoldMessenger.of(context).showSnackBar(  SnackBar(content: Text(msg), duration: const Duration(seconds: 2)),  );  }  @override  void dispose() {  \_timer?.cancel();  \_audioPlayer.dispose();  super.dispose(); }  List<DropdownMenuItem<int>> \_buildHourItems() {  return List.generate(12, (index) {  final hour = index + 1;  return DropdownMenuItem(value: hour, child: Text(hour.toString()));  });  }  List<DropdownMenuItem<int>> \_buildMinuteItems() {  return List.generate(60, (index) {  final text = index.toString().padLeft(2, '0');  return DropdownMenuItem(value: index, child: Text(text));  });  }  List<DropdownMenuItem<String>> \_buildAmPmItems() {  return ['AM', 'PM'].map((period) {  return DropdownMenuItem(value: period, child: Text(period));  }).toList();  }  @override  Widget build(BuildContext context) {  final timeText = \_selectedTime == null  ? "No alarm set"  : "Alarm set for: ${\_selectedTime!.format(context)}";  return Scaffold(  backgroundColor: Colors.white,  appBar: AppBar(  title: const Text("Workout Alarm"),  backgroundColor: Colors.white,  centerTitle: true,  ),  body: Center(  child: Padding(  padding: const EdgeInsets.symmetric(horizontal: 24.0),  child: SingleChildScrollView(  child: Column(  mainAxisAlignment: MainAxisAlignment.center,  children: [  const Icon(Icons.alarm, size: 100, color: Colors.red),  const SizedBox(height: 20),  Text(timeText, style: const TextStyle(fontSize: 22)),  const SizedBox(height: 30),  // Initial Time Selector  const Text('Initial Time (optional):', style: TextStyle(fontSize: 16)),  const SizedBox(height: 8),  Row(  mainAxisAlignment: MainAxisAlignment.center,  children: [  DropdownButton<int>(  value: \_initialHour,  items: \_buildHourItems(),  onChanged: (val) {  setState(() {  \_initialHour = val;  });  },  ),  const Text(' : '),  DropdownButton<int>(  value: \_initialMinute,  items: \_buildMinuteItems(),  onChanged: (val) {  setState(() {  \_initialMinute = val;  });  },  ),  const SizedBox(width: 10),  DropdownButton<String>(  value: \_initialAmPm,  items: \_buildAmPmItems(),  onChanged: (val) {  if (val != null) {  setState(() {  \_initialAmPm = val;  });  }  },  ),  ],  ),  const SizedBox(height: 20),  // Final Time Selector  const Text('Final Time (optional):', style: TextStyle(fontSize: 16)),  const SizedBox(height: 8),  Row(  mainAxisAlignment: MainAxisAlignment.center,  children: [  DropdownButton<int>(  value: \_finalHour,  items: \_buildHourItems(),  onChanged: (val) {  setState(() {  \_finalHour = val;  });  },  ),  const Text(' : '),  DropdownButton<int>(  value: \_finalMinute,  items: \_buildMinuteItems(),  onChanged: (val) {  setState(() {  \_finalMinute = val;  });  },  ),  const SizedBox(width: 10),  DropdownButton<String>(  value: \_finalAmPm,  items: \_buildAmPmItems(),  onChanged: (val) {  if (val != null) {  setState(() {  \_finalAmPm = val;  });  }  },  ),  ],  ),  const SizedBox(height: 30),  ElevatedButton.icon(  icon: const Icon(Icons.alarm\_add),  label: const Text("Set Alarm From Input"),  onPressed: \_setAlarmFromInputs,  style: ElevatedButton.styleFrom(  minimumSize: const Size(double.infinity, 50),  ),  ),  const SizedBox(height: 20),  ElevatedButton.icon(  icon: const Icon(Icons.access\_time),  label: const Text("Pick Alarm Time"),  onPressed: \_pickTime,  style: ElevatedButton.styleFrom(  minimumSize: const Size(double.infinity, 50), ), ),  const SizedBox(height: 20),  ElevatedButton.icon(  icon: const Icon(Icons.play\_arrow),  label: const Text("Test Sound"),  onPressed: \_playTestSound,  style: ElevatedButton.styleFrom(  minimumSize: const Size(double.infinity, 50), ),),  const SizedBox(height: 20),  ElevatedButton.icon(  icon: const Icon(Icons.stop\_circle\_outlined),  label: const Text("Stop Sound"),  onPressed: \_stopAlarm,  style: ElevatedButton.styleFrom(  backgroundColor: Colors.red,  foregroundColor: Colors.white,  minimumSize: const Size(double.infinity, 50),  ),  ),  const SizedBox(height: 30),  const Text("Select Alarm Tone:", style: TextStyle(fontSize: 16)),  const SizedBox(height: 10),  DropdownButton<String>(  value: \_selectedTone,  isExpanded: true,  items: \_tones.map((tone) {  return DropdownMenuItem(  value: tone,  child: Text(tone.replaceAll('.mp3', '').toUpperCase()),  );  }).toList(),  onChanged: (value) {  if (value != null) {  setState(() {  \_selectedTone = value;  });  }  },  ),  ],  ),  ),  ),  ),  );  }  } |

**5.dietplan.dart:**

|  |
| --- |
| // ignore\_for\_file: deprecated\_member\_use  import 'package:flutter/material.dart';  import 'dashboard.dart'; // make sure this contains buildAppDrawer()  class Dietplan extends StatelessWidget {  const Dietplan({super.key});  @override  Widget build(BuildContext context) {  final ScrollController scrollController = ScrollController();    final theme = Theme.of(context);  final headerColor = Colors.green.shade700;  final rowColor1 = Colors.grey.shade100;  // ignore: prefer\_const\_declarations  final rowColor2 = Colors.white;    return MaterialApp(  title: 'Diet Plan',  debugShowCheckedModeBanner: false,  theme: ThemeData(  colorSchemeSeed: Colors.green,  useMaterial3: true,  ),  home: Scaffold(  backgroundColor: Colors.grey.shade50,  appBar: AppBar(  title: const Text(  'Weekly Diet Plan',  style: TextStyle(fontWeight: FontWeight.w600, letterSpacing: 0.5),  ),  elevation: 2,  backgroundColor: theme.colorScheme.primaryContainer,  foregroundColor: theme.colorScheme.onPrimaryContainer,  centerTitle: true,  ),  drawer: buildAppDrawer(context),  body: RawScrollbar(  controller: scrollController,  // ignore: deprecated\_member\_use  thumbColor: theme.colorScheme.primary.withOpacity(0.6),  radius: const Radius.circular(8),  thickness: 6,  trackVisibility: true,  scrollbarOrientation: ScrollbarOrientation.bottom,  interactive: true,  child: SingleChildScrollView(  controller: scrollController,  scrollDirection: Axis.horizontal,  child: Padding(  padding: const EdgeInsets.all(16.0),  child: Container(  decoration: BoxDecoration(  color: Colors.white,  borderRadius: BorderRadius.circular(12),  boxShadow: const [  BoxShadow(  color: Colors.black12,  blurRadius: 10,  offset: Offset(0, 5),  ),  ],  ),  child: DataTable(  columnSpacing: 24,  // ignore: deprecated\_member\_use  headingRowColor: MaterialStateProperty.all(headerColor.withOpacity(0.15)),  headingTextStyle: TextStyle(  color: headerColor,  fontWeight: FontWeight.bold,  fontSize: 16,  ),  // ignore: deprecated\_member\_use  dataRowColor: MaterialStateProperty.resolveWith<Color?>(  (Set<MaterialState> states) {  if (states.contains(MaterialState.selected)) {  return theme.colorScheme.primary.withOpacity(0.1);  }  return null;  },  ),  border: TableBorder.symmetric(  inside: BorderSide(width: 0.5, color: Colors.grey.shade300),  outside: BorderSide(width: 1, color: Colors.grey.shade300),  ),  columns: const [  DataColumn(label: Text('Time')),  DataColumn(label: Text('Mon')),  DataColumn(label: Text('Tue')),  DataColumn(label: Text('Wed')),  DataColumn(label: Text('Thu')),  DataColumn(label: Text('Fri')),  DataColumn(label: Text('Sat')),  DataColumn(label: Text('Sun')),  ],  rows: List.generate(  \_dietData.length,  (index) {  final row = \_dietData[index];  return DataRow(  // ignore: deprecated\_member\_use  color: MaterialStateProperty.all(  index.isEven ? rowColor1 : rowColor2),  cells: [  DataCell(Text(row['time']!, style: const TextStyle(fontWeight: FontWeight.w600))),  DataCell(Text(row['mon']!)),  DataCell(Text(row['tue']!)),  DataCell(Text(row['wed']!)),  DataCell(Text(row['thu']!)),  DataCell(Text(row['fri']!)),  DataCell(Text(row['sat']!)),  DataCell(Text(row['sun']!)),  ],  );  },  ),  ),  ),  ),  ),  ),  ),  );  }  }  final List<Map<String, String>> \_dietData = [  {  'time': 'Morning',  'mon': 'Oatmeal + Eggs',  'tue': 'Smoothie + Toast',  'wed': 'Oats + Milk',  'thu': 'Boiled Eggs + Tea',  'fri': 'Poha + Fruits',  'sat': 'Idli + Sambar',  'sun': 'Cornflakes + Milk',  },  {  'time': 'Mid-Morning',  'mon': 'Fruit Salad',  'tue': 'Green Tea + Almonds',  'wed': 'Banana',  'thu': 'Cucumber Salad',  'fri': 'Nuts Mix',  'sat': 'Apple',  'sun': 'Lassi',  },  {  'time': 'Lunch',  'mon': 'Brown Rice + Veg Curry',  'tue': 'Grilled Chicken + Rice',  'wed': 'Dal + Chapati',  'thu': 'Tofu + Rice',  'fri': 'Chickpea Salad',  'sat': 'Paneer Wrap',  'sun': 'Veg Pulao',  },  {  'time': 'Evening Snack',  'mon': 'Sprouts',  'tue': 'Green Tea + Crackers',  'wed': 'Fruits',  'thu': 'Murmura',  'fri': 'Nuts',  'sat': 'Bhel Puri',  'sun': 'Buttermilk',  },  {  'time': 'Dinner',  'mon': 'Lentil Soup + Salad',  'tue': 'Grilled Fish + Veggies',  'wed': 'Chapati + Veg Curry',  'thu': 'Rice + Lentils',  'fri': 'Khichdi',  'sat': 'Soup + Bread',  'sun': 'Vegetable Stew',  },  {  'time': 'Night',  'mon': 'Warm Milk',  'tue': 'Chamomile Tea',  'wed': 'Milk + Honey',  'thu': 'Herbal Tea',  'fri': 'Warm Milk',  'sat': 'Green Tea',  'sun': 'Almond Milk',  },  ]; |

6.Exercise\_plan.dart:

|  |
| --- |
| // ignore\_for\_file: library\_private\_types\_in\_public\_api  import 'package:flutter/material.dart';  import 'package:youtube\_player\_flutter/youtube\_player\_flutter.dart';  // Import your drawer function from dashboard.dart or just copy it here  import 'dashboard.dart'; // Make sure this import path is correct  class ExerciseScreen extends StatefulWidget {  const ExerciseScreen({super.key});  @override  \_ExerciseScreenState createState() => \_ExerciseScreenState();  }  class \_ExerciseScreenState extends State<ExerciseScreen> {  final List<String> videoUrls = [ 'https://youtu.be/U3HlEF\_E9fo?si=5wd8JD6GgZyw-dCn',  'https://youtu.be/i9sTjhN4Z3M?si=VrKPGIF\_Hxw26u1T',  'https://youtu.be/wrwwXE\_x-pQ?si=GB1JY1UR6XnKZ19U',  'https://youtube.com/shorts/7vVMxbFPLhA?si=Vp5CZaXAnRq3vpZA',  'https://youtu.be/uLVt6u15L98?si=wEBy1f1n1diF66ve', 'https://youtu.be/sQKT1bHRrg4?si=gup8dmwTVllHdWyZ',];  final List<String> exerciseTitles = [  'Squats',  'Push-Ups',  'Lunges',  'Plank',  'Jumping Jacks',  'Burpees',];  @override  Widget build(BuildContext context) {  return Scaffold(  backgroundColor: Colors.white,  appBar: AppBar(  title: const Text('Exercise Plan Videos'),  // Do NOT set automaticallyImplyLeading to false here  // Flutter will show the hamburger menu automatically because drawer is set  ),  drawer: buildAppDrawer(context), // Add the drawer here  body: ListView.builder(  itemCount: videoUrls.length,  itemBuilder: (context, index) {  return Padding(  padding: const EdgeInsets.all(8.0),  child: Container(  padding: const EdgeInsets.all(12),  decoration: BoxDecoration(  color: Colors.grey[200],  borderRadius: BorderRadius.circular(12),  ),  child: Column(  crossAxisAlignment: CrossAxisAlignment.start,  children: [  Text(  exerciseTitles[index],  style: const TextStyle(  fontSize: 18,  fontWeight: FontWeight.bold,  ),  ),  const SizedBox(height: 10),  YoutubePlayer(  controller: YoutubePlayerController(  initialVideoId:  YoutubePlayer.convertUrlToId(videoUrls[index])!,  flags: const YoutubePlayerFlags(  autoPlay: false, mute: false),  ),  showVideoProgressIndicator: true,  ),  const SizedBox(height: 20),  ],), ), );}, ),); }} |

7.footsteptime.dart:

|  |
| --- |
| import 'dart:async';  import 'package:flutter/material.dart';  class FootstepTimer extends StatefulWidget {  const FootstepTimer({super.key});  @override  State<FootstepTimer> createState() => \_FootstepTimerState();  }  class \_FootstepTimerState extends State<FootstepTimer> {  final Stopwatch \_stopwatch = Stopwatch();  Timer? \_timer;  String \_displayTime = "00";  // Track hovered buttons  final Map<String, bool> \_isHovering = {  "Start": false,  "Stop": false,  "Reset": false,  };  void \_startStopwatch() {  if (!\_stopwatch.isRunning) {  \_stopwatch.start();  \_timer = Timer.periodic(const Duration(seconds: 1), (timer) {  setState(() {  final elapsed = \_stopwatch.elapsed;  \_displayTime = elapsed.inSeconds.toString().padLeft(2, '0'); }); });  }}  void \_stopStopwatch() {  if (\_stopwatch.isRunning) {  \_stopwatch.stop();  \_timer?.cancel();  }  }  void \_resetStopwatch() {  \_stopwatch.reset();  \_stopwatch.stop();  \_timer?.cancel();  setState(() {  \_displayTime = "00";  });  }  @override  void dispose() {  \_stopwatch.stop();  \_timer?.cancel();  super.dispose();  }  Color \_getButtonColor(String label) {  switch (label) {  case "Start":  return Colors.green.shade100; // light green  case "Stop":  return Colors.red.shade100; // light red  case "Reset":  return Colors.grey.shade300; // light grey  default:  return Colors.grey.shade200;  }  }  Color \_getHoverColor(String label) {  switch (label) {  case "Start":  return Colors.green.shade300;  case "Stop":  return Colors.red.shade300;  case "Reset":  return Colors.grey.shade500;  default:  return Colors.grey.shade400;  }  }  @override  Widget build(BuildContext context) {  return Scaffold(  backgroundColor: Colors.white,  appBar: AppBar(  title: const Text('Footstep Timer', style: TextStyle(color: Colors.black)),  backgroundColor: Colors.grey[200],  centerTitle: true,  iconTheme: const IconThemeData(color: Colors.black),  elevation: 1,  ),  body: SingleChildScrollView(  padding: const EdgeInsets.symmetric(horizontal: 24.0, vertical: 30),  child: Column(  mainAxisAlignment: MainAxisAlignment.center,  children: [  // Smaller footstep circle  Container(  width: 120,  height: 120,  decoration: BoxDecoration(  color: Colors.grey.shade100, // very light grey for subtle contrast  shape: BoxShape.circle,  ),  child: Center(  child: Image.network(  'https://cdn3.iconfinder.com/data/icons/basic-mobile-part-2/512/footprints-512.png',  width: 60,  height: 60,  fit: BoxFit.contain,  ),  ),  ),  const SizedBox(height: 30),  Text(  \_displayTime,  style: const TextStyle(fontSize: 40, fontWeight: FontWeight.bold),  ),  const SizedBox(height: 30),  Row(  mainAxisAlignment: MainAxisAlignment.spaceEvenly,  children: ["Start", "Stop", "Reset"].map((label) {  return MouseRegion(  onEnter: (\_) {  setState(() {  \_isHovering[label] = true;  });  },  onExit: (\_) {  setState(() {  \_isHovering[label] = false;  });  },  child: AnimatedContainer(  duration: const Duration(milliseconds: 200),  curve: Curves.easeInOut,  decoration: BoxDecoration(  color: \_isHovering[label]!  ? \_getHoverColor(label)  : \_getButtonColor(label),  borderRadius: BorderRadius.circular(6),  ),  child: TextButton(  onPressed: () {  if (label == "Start") \_startStopwatch();  if (label == "Stop") \_stopStopwatch();  if (label == "Reset") \_resetStopwatch();  },  style: TextButton.styleFrom(  padding: const EdgeInsets.symmetric(horizontal: 14, vertical: 8),  foregroundColor: Colors.black87,  textStyle: const TextStyle(fontSize: 14),  ),  child: Text(label),  ),  ),  );  }).toList(),  ),  ],  ),  ),  );  }  } |

8.healthmonitor.dart:

|  |
| --- |
| *// ignore\_for\_file: prefer\_const\_declarations*  import 'dart:math';  import 'package:flutter/material.dart';  import 'dashboard.dart'; // Optional: Replace this with your actual dashboard.dart if needed  void main() {  runApp(const HealthMonitorscreen());}  class HealthMonitorscreen extends StatelessWidget {  const HealthMonitorscreen({super.key});  @override  Widget build(BuildContext context) {  return MaterialApp(  title: 'Health Monitor - BMI Calculator',  debugShowCheckedModeBanner: false,  theme: ThemeData(  primarySwatch: Colors.blue,  scaffoldBackgroundColor: Colors.grey[100],),  home: const BMICalculatorScreen(), ); }}  class BMICalculatorScreen extends StatefulWidget {  // ignore: use\_super\_parameters  const BMICalculatorScreen({Key? key}) : super(key: key);  @override  State<BMICalculatorScreen> createState() => \_BMICalculatorScreenState();}  class \_BMICalculatorScreenState extends State<BMICalculatorScreen>  with SingleTickerProviderStateMixin {  final TextEditingController \_weightController = TextEditingController();  final TextEditingController \_heightCmController = TextEditingController();  final TextEditingController \_heightFeetController = TextEditingController();  final TextEditingController \_heightInchController = TextEditingController();  String \_weightUnit = 'kg';  String \_heightUnit = 'cm';  double? \_bmiResult;  String \_bmiCategory = '';  String \_diseasePrediction = '';  late AnimationController \_animationController;  late Animation<double> \_rotationAnimation;  final List<String> \_emojis = ['😟', '😊', '😐', '⚠️'];  double \_angleForCategory(int index) {  return (2 \* pi / \_emojis.length) \* index;}  void \_calculateBMI() {  final weight = double.tryParse(\_weightController.text);  double? heightCm;  if (weight == null || weight <= 0) {  ScaffoldMessenger.of(context).showSnackBar(  const SnackBar(content: Text('Enter valid weight')),);  return;}  if (\_heightUnit == 'cm') {  heightCm = double.tryParse(\_heightCmController.text);  if (heightCm == null || heightCm <= 0) {  ScaffoldMessenger.of(context).showSnackBar(  const SnackBar(content: Text('Enter valid height in cm')), );  return; }  } else {  final feet = double.tryParse(\_heightFeetController.text);  final inches = double.tryParse(\_heightInchController.text) ?? 0;  if (feet == null || feet <= 0) {  ScaffoldMessenger.of(context).showSnackBar(  const SnackBar(content: Text('Enter valid height in feet')),  );  return;  }  heightCm = (feet \* 30.48) + (inches \* 2.54);  }  double weightKg = \_weightUnit == 'lbs' ? weight \* 0.453592 : weight;  final heightM = heightCm / 100;  final bmi = weightKg / (heightM \* heightM);  String category;  String prediction;  int categoryIndex;  if (bmi < 18.5) {  category = 'Underweight';  prediction = 'Risk: Nutritional deficiencies, osteoporosis.';  categoryIndex = 0;  } else if (bmi < 24.9) {  category = 'Normal weight';  prediction = 'Great! Keep maintaining your healthy lifestyle.';  categoryIndex = 1;  } else if (bmi < 29.9) {  category = 'Overweight';  prediction = 'Risk: High blood pressure, diabetes, heart disease.';  categoryIndex = 2;  } else {  category = 'Obesity';  prediction = 'Risk: Severe heart disease, diabetes, stroke.';  categoryIndex = 3;  }  setState(() {  \_bmiResult = bmi;  \_bmiCategory = category;  \_diseasePrediction = prediction;  });  \_animateNeedle(categoryIndex);  }  void \_animateNeedle(int categoryIndex) {  final targetAngle = \_angleForCategory(categoryIndex);  \_rotationAnimation = Tween<double>(  begin: \_rotationAnimation.value,  end: targetAngle,  ).animate(  CurvedAnimation(parent: \_animationController, curve: Curves.easeInOut),  );  \_animationController  ..reset()  ..forward();  }  @override  void initState() {  super.initState();  \_animationController = AnimationController(  duration: const Duration(milliseconds: 700),  vsync: this,  );  \_rotationAnimation = Tween<double>(begin: 0, end: 0).animate(  CurvedAnimation(parent: \_animationController, curve: Curves.easeInOut),  );  }  @override  void dispose() {  \_weightController.dispose();  \_heightCmController.dispose();  \_heightFeetController.dispose();  \_heightInchController.dispose();  \_animationController.dispose();  super.dispose();  }  @override  Widget build(BuildContext context) {  const compassSize = 250.0;  return Scaffold(  appBar: AppBar(title: const Text('BMI Calculator')),  drawer: buildAppDrawer(context), // If you have one  body: SingleChildScrollView(  padding: const EdgeInsets.all(20),  child: Card(  elevation: 4,  shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(16)),  child: Padding(  padding: const EdgeInsets.all(20),  child: Column(  children: [  Row(  children: [  Expanded(  child: TextField(  controller: \_weightController,  keyboardType: const TextInputType.numberWithOptions(decimal: true),  decoration: const InputDecoration(  labelText: 'Weight',  border: OutlineInputBorder(), ),), ),  const SizedBox(width: 10),  DropdownButton<String>(  value: \_weightUnit,  items: ['kg', 'lbs']  .map((unit) => DropdownMenuItem(value: unit, child: Text(unit)))  .toList(),  onChanged: (val) => setState(() => \_weightUnit = val!),  ),  ],  ),  const SizedBox(height: 16),  Row(  children: [  const Text("Height in: "),  const SizedBox(width: 8),  DropdownButton<String>(  value: \_heightUnit,  items: ['cm', 'ft/in']  .map((unit) => DropdownMenuItem(value: unit, child: Text(unit)))  .toList(),  onChanged: (val) => setState(() => \_heightUnit = val!),  ),  ],  ),  const SizedBox(height: 12),  if (\_heightUnit == 'cm')  TextField(  controller: \_heightCmController,  keyboardType: const TextInputType.numberWithOptions(decimal: true),  decoration: const InputDecoration(  labelText: 'Height (cm)',  border: OutlineInputBorder(),  ),  )  else  Row(  children: [  Expanded(  child: TextField(  controller: \_heightFeetController,  keyboardType: const TextInputType.numberWithOptions(decimal: true),  decoration: const InputDecoration(  labelText: 'Feet',  border: OutlineInputBorder(),),), ),  const SizedBox(width: 10),  Expanded(  child: TextField(  controller: \_heightInchController,  keyboardType: const TextInputType.numberWithOptions(decimal: true),  decoration: const InputDecoration(  labelText: 'Inches',  border: OutlineInputBorder(),),), ), ],),  const SizedBox(height: 20),  ElevatedButton(  onPressed: \_calculateBMI,  style: ElevatedButton.styleFrom(  minimumSize: const Size(double.infinity, 48),  ),  child: const Text('Calculate BMI'),  ),  const SizedBox(height: 24),  if (\_bmiResult != null)  Column(  children: [  Text(  'Your BMI: ${\_bmiResult!.toStringAsFixed(2)} ($\_bmiCategory)',  style: const TextStyle(fontSize: 22, fontWeight: FontWeight.bold),  ),  const SizedBox(height: 10),  Text(  \_diseasePrediction,  style: const TextStyle(fontSize: 16, color: Colors.black87),  textAlign: TextAlign.center,  ),  const SizedBox(height: 30),  Divider(thickness: 1, color: Colors.grey[400]),  const SizedBox(height: 20),  SizedBox(  width: compassSize,  height: compassSize,  child: Stack(  alignment: Alignment.center,  children: [  Container(  width: compassSize,  height: compassSize,  decoration: BoxDecoration(  shape: BoxShape.circle,  color: Colors.blue.shade50,  border: Border.all(color: Colors.blue.shade700, width: 3), ),),  ...List.generate(\_emojis.length, (index) {  final angle = \_angleForCategory(index);  final radius = compassSize / 2 - 30;  final dx = radius \* cos(angle);  final dy = radius \* sin(angle);  return Positioned(  left: (compassSize / 2) + dx - 12,  top: (compassSize / 2) + dy - 12,  child: Text(\_emojis[index], style: const TextStyle(fontSize: 24)),  );  }),  AnimatedBuilder(  animation: \_rotationAnimation,  builder: (context, child) {  return Transform.rotate(  angle: \_rotationAnimation.value,  child: child, ); },  child: Container(  width: 4,  height: compassSize / 2 - 30,  decoration: BoxDecoration(  color: Colors.redAccent,  borderRadius: BorderRadius.circular(2),),  alignment: Alignment.topCenter,  child: const Text('🧭', style: TextStyle(fontSize: 24)),),  ),  Container(  width: 16,  height: 16,  decoration: const BoxDecoration(  shape: BoxShape.circle,  color: Colors.redAccent,), ),  ],  ),), ], ), ],),),),), );}} |

8.homescreen.dart:

|  |
| --- |
| import 'package:flutter/material.dart';  import 'exercise\_plan.dart';  import 'specificdietplan.dart';  import 'alarmworkout.dart';  import 'healthmonitor.dart';  class HomeScreen extends StatelessWidget {  const HomeScreen({super.key});  Widget \_buildCard(BuildContext context, String imagePath, String label, Widget screen) {  return StatefulBuilder(  builder: (context, setState) {  bool isHovered = false;  return MouseRegion(  onEnter: (\_) => setState(() => isHovered = true),  onExit: (\_) => setState(() => isHovered = false),  child: AnimatedContainer(  duration: const Duration(milliseconds: 200),  margin: const EdgeInsets.symmetric(vertical: 10),  decoration: BoxDecoration(  color: isHovered ? Colors.white70 : Colors.white,  borderRadius: BorderRadius.circular(12),  boxShadow: [  BoxShadow(  color: isHovered ? Colors.blue.withOpacity(0.3) : Colors.grey.withOpacity(0.2),  blurRadius: isHovered ? 10 : 4,  offset: const Offset(0, 4),  ),  ],  ),  child: Material(  color: Colors.transparent,  borderRadius: BorderRadius.circular(12),  child: InkWell(  borderRadius: BorderRadius.circular(12),  onTap: () {  Navigator.push(context, MaterialPageRoute(builder: (\_) => screen));  },  child: Column(  children: [  ClipRRect(  borderRadius: const BorderRadius.vertical(top: Radius.circular(12)),  child: Image.asset(  imagePath,  width: double.infinity,  height: 80,  fit: BoxFit.cover,  ),  ),  Padding(  padding: const EdgeInsets.all(10),  child: Text(  label,  style: const TextStyle(  fontSize: 16,  fontWeight: FontWeight.w600,  color: Colors.black87, ), ), ), ], ), ), ),), );}, );}  @override  Widget build(BuildContext context) {  return Container(  color: Colors.grey[100],  child: SafeArea(  child: SingleChildScrollView(  padding: const EdgeInsets.symmetric(horizontal: 20, vertical: 18),  child: Column(  crossAxisAlignment: CrossAxisAlignment.center,  children: <Widget>[  const Icon(Icons.fitness\_center, size: 90, color: Colors.blueAccent),  const SizedBox(height: 18),  const Text(  'Welcome to FitMe360!',  style: TextStyle(fontSize: 24, fontWeight: FontWeight.bold, color: Colors.black87),  ),  const SizedBox(height: 10),  const Text(  'Track your fitness, diet, and progress with ease.',  textAlign: TextAlign.center,  style: TextStyle(fontSize: 15, color: Colors.black87),  ),  const SizedBox(height: 25),  // Feature Cards  \_buildCard(context, 'assets/content/image1.jpg', 'Exercise Plan', const ExerciseScreen()),  \_buildCard(context, 'assets/content/image2.jpg', 'Diet Plan', const SpecificDietPlan()),  \_buildCard(context, 'assets/content/image3.jpg', 'Workout Alarm', const AlarmWorkoutScreen()),  \_buildCard(context, 'assets/content/image4.jpg', 'Health Monitor', const HealthMonitorscreen()),  ],  ),  ),  ),  );  }  } |

10.login.dart:

|  |
| --- |
| import 'package:flutter/material.dart';  import 'package:shared\_preferences/shared\_preferences.dart';  import 'database\_helper.dart';  import 'dashboard.dart';  import 'signup\_page.dart';  class LoginPage extends StatefulWidget {  const LoginPage({super.key});  @override  \_LoginPageState createState() => \_LoginPageState();  }  class \_LoginPageState extends State<LoginPage> {  final TextEditingController emailController = TextEditingController();  final TextEditingController passwordController = TextEditingController();  final GlobalKey<FormState> \_formKey = GlobalKey<FormState>();  void \_login() async {  if (\_formKey.currentState!.validate()) {  String email = emailController.text.trim();  String password = passwordController.text.trim();  final user = await DatabaseHelper().getUser(email, password);  if (user != null) {  final prefs = await SharedPreferences.getInstance();  await prefs.setString('email', email);  await prefs.setString('password', password);  if (user['name'] != null) {  await prefs.setString('name', user['name']);  }  if (user['profile\_image'] != null) {  await prefs.setString('profile\_image', user['profile\_image']);  }  Navigator.pushReplacement(  context,  MaterialPageRoute(builder: (context) => const DashboardPage()),  );  } else {  ScaffoldMessenger.of(context).showSnackBar(  const SnackBar(content: Text('Invalid email or password')),  );  }  }  }  void \_navigateToSignup() {  Navigator.push(  context,  MaterialPageRoute(builder: (context) => const SignUpPage()),  );  }  InputDecoration \_inputDecoration(String label) {  return InputDecoration(  labelText: label,  filled: true,  fillColor: Colors.white,  contentPadding: const EdgeInsets.symmetric(horizontal: 20, vertical: 16),  enabledBorder: OutlineInputBorder(  borderRadius: BorderRadius.circular(12),  borderSide: BorderSide(color: Colors.grey.shade300, width: 1.5),  ),  focusedBorder: OutlineInputBorder(  borderRadius: BorderRadius.circular(12),  borderSide: const BorderSide(color: Colors.blue, width: 2),  ),  errorBorder: OutlineInputBorder(  borderRadius: BorderRadius.circular(12),  borderSide: const BorderSide(color: Colors.red, width: 2),  ),  focusedErrorBorder: OutlineInputBorder(  borderRadius: BorderRadius.circular(12),  borderSide: const BorderSide(color: Colors.redAccent, width: 2),  ),  // Shadow inside input field  // Flutter InputDecoration doesn't support shadow directly,  // so we wrap TextFormField with Material for elevation/shadow  );  }  @override  Widget build(BuildContext context) {  return Scaffold(  backgroundColor: Colors.grey[100],  appBar: AppBar(  title: const Text('Login'),  backgroundColor: Colors.white,  foregroundColor: Colors.black,  elevation: 0,  ),  body: Scrollbar(  thumbVisibility: true,  child: SingleChildScrollView(  padding: const EdgeInsets.all(24),  child: Form(  key: \_formKey,  child: Column(  children: [  const SizedBox(height: 50),  Text(  'Welcome Back!',  style: TextStyle(  fontSize: 28,  fontWeight: FontWeight.bold,  color: Colors.blue.shade700,  ),  ),  const SizedBox(height: 12),  Text(  'Log in to your account',  style: TextStyle(  fontSize: 16,  color: Colors.grey.shade600,  ),  ),  const SizedBox(height: 40),    // Email Field with shadow & rounded corners  Material(  elevation: 3,  shadowColor: Colors.black26,  borderRadius: BorderRadius.circular(12),  child: TextFormField(  controller: emailController,  decoration: \_inputDecoration('Email'),  validator: (value) =>  value == null || value.isEmpty ? 'Enter Email' : null,  keyboardType: TextInputType.emailAddress,  ),  ),  const SizedBox(height: 20),  // Password Field with shadow & rounded corners  Material(  elevation: 3,  shadowColor: Colors.black26,  borderRadius: BorderRadius.circular(12),  child: TextFormField(  controller: passwordController,  decoration: \_inputDecoration('Password'),  obscureText: true,  validator: (value) =>  value == null || value.isEmpty ? 'Enter Password' : null,  ),  ),  const SizedBox(height: 30),  // Login Button with gradient & rounded corners  SizedBox(  width: double.infinity,  height: 50,  child: ElevatedButton(  onPressed: \_login,  style: ElevatedButton.styleFrom(  elevation: 5,  shape: RoundedRectangleBorder(  borderRadius: BorderRadius.circular(14)),  padding: EdgeInsets.zero,  ),  child: Ink(  decoration: BoxDecoration(  gradient: const LinearGradient(  colors: [Color(0xFF42A5F5), Color(0xFF1976D2)],  begin: Alignment.topLeft,  end: Alignment.bottomRight,  ),  borderRadius: BorderRadius.circular(14),  ),  child: Container(  alignment: Alignment.center,  child: const Text(  'Login',  style: TextStyle(  fontSize: 18, fontWeight: FontWeight.bold),  ),  ),  ),  ),  ),  const SizedBox(height: 20),  TextButton(  onPressed: \_navigateToSignup,  child: Text(  "Don't have an account? Sign up",  style: TextStyle(color: Colors.blue.shade700),  ),  ),  ],  ),  ),  ),  ),  );  }  } |

**11.profilestatus.dart:**

|  |
| --- |
| // ignore\_for\_file: library\_private\_types\_in\_public\_api  import 'dart:io';  import 'package:flutter/material.dart';  import 'package:image\_picker/image\_picker.dart';  import 'package:shared\_preferences/shared\_preferences.dart';  import 'database\_helper.dart';  import 'footsteptimer.dart';  import 'rememberlist.dart';  import 'login.dart';  import 'dietplan.dart';  import 'dashboard.dart';  class STATUS extends StatefulWidget {  const STATUS({super.key});  @override  \_STATUSState createState() => \_STATUSState();  }  class \_STATUSState extends State<STATUS> {  Map<String, dynamic>? userData;  bool isLoading = true;  File? \_image;  int \_currentIndex = 0;  final List<Widget> \_pages = const [  DashboardPage(),  FootstepTimer(),  RememberList(),  ];  @override  void initState() {  super.initState();  loadUser();  loadImage();  }  Future<void> loadUser() async {  final prefs = await SharedPreferences.getInstance();  final email = prefs.getString('email');  final password = prefs.getString('password');  if (email != null && password != null) {  final user = await DatabaseHelper().getUser(email, password);  if (user != null) {  await prefs.setString('name', user['name'] ?? '');  setState(() {  userData = user;  isLoading = false;  });  } else {  setState(() {  userData = null;  isLoading = false;  });  }  } else {  setState(() {  userData = null;  isLoading = false;  });  }  }  Future<void> loadImage() async {  final prefs = await SharedPreferences.getInstance();  final path = prefs.getString('profile\_image');  if (path != null) {  final file = File(path);  if (await file.exists()) {  setState(() {  \_image = file;  });  }}}  Future<void> pickImage() async {  final picker = ImagePicker();  final pickedFile = await picker.pickImage(source: ImageSource.gallery);  if (pickedFile != null) {  final prefs = await SharedPreferences.getInstance();  await prefs.setString('profile\_image', pickedFile.path);  setState(() {  \_image = File(pickedFile.path);}); }}  @override  Widget build(BuildContext context) {  return Scaffold(  backgroundColor: Colors.white,  appBar: AppBar(title: const Text('Profile')),  drawer: Drawer(  child: ListView(  padding: EdgeInsets.zero,  children: [  const DrawerHeader(  decoration: BoxDecoration(color: Colors.blue),  child: Text(  'Menu',  style: TextStyle(color: Colors.white, fontSize: 24),  ),  ),  \_buildDrawerItem(Icons.dashboard, 'Dashboard', const DashboardPage()),  \_buildDrawerItem(Icons.fastfood, 'Food', const Dietplan()),  ListTile(  leading: const Icon(Icons.person),  title: const Text('Profile'),  onTap: () => Navigator.pop(context),  ),  ListTile(  leading: const Icon(Icons.logout),  title: const Text('Sign out'),  onTap: () {  Navigator.pushReplacement(  context,  MaterialPageRoute(builder: (context) => const LoginPage()),  );  },  ),  ],  ),  ),  body: Center(  child: isLoading  ? const CircularProgressIndicator()  : userData == null  ? const Text('User not found')  : Container(  padding: const EdgeInsets.all(8),  color: Colors.white70,  height: 600,  width: 300,  child: SingleChildScrollView(  child: Column(  mainAxisAlignment: MainAxisAlignment.start,  children: [  const SizedBox(height: 20),  GestureDetector(  onTap: pickImage,  child: CircleAvatar(  radius: 50,  backgroundImage: \_image != null  ? FileImage(\_image!)  : const AssetImage('assets/content/image1.jpg') as ImageProvider,  child: \_image == null  ? const Icon(Icons.add\_a\_photo, color: Colors.white70)  : null,  ),  ),  const SizedBox(height: 10),  \_buildUserInfo('Username', userData!['name']),  \_buildUserInfo('Father Name', userData!['fathername']),  \_buildUserInfo('Gender', userData!['gender']),  \_buildUserInfo('Weight', userData!['weight']),  \_buildUserInfo('Height', userData!['height']),  \_buildUserInfo('Age', userData!['age']),  \_buildUserInfo('DOB', userData!['dob']),  \_buildUserInfo('Email', userData!['email']), ], ), ),),),  bottomNavigationBar: BottomNavigationBar(  currentIndex: \_currentIndex,  selectedItemColor: Colors.blue,  unselectedItemColor: Colors.grey,  onTap: (index) {  setState(() {  \_currentIndex = index;  });  Navigator.pushReplacement(  context,  MaterialPageRoute(builder: (context) => \_pages[index]),  );  },  items: const [  BottomNavigationBarItem(  icon: Icon(Icons.home),  label: 'Home',  ),  BottomNavigationBarItem(  icon: Icon(Icons.directions\_run),  label: 'Running',  ),  BottomNavigationBarItem(  icon: Icon(Icons.note),  label: 'Notepad',  ),  ],  ),  );  }  Widget \_buildUserInfo(String label, dynamic value) {  return Padding(  padding: const EdgeInsets.symmetric(vertical: 4),  child: Text('$label: ${value ?? "N/A"}'),  );  }  ListTile \_buildDrawerItem(IconData icon, String title, Widget destination) {  return ListTile(  leading: Icon(icon),  title: Text(title),  onTap: () {  Navigator.pushReplacement(  context,  MaterialPageRoute(builder: (context) => destination),  );  },  );  }  } |

12.rememberlist.dart:

|  |
| --- |
| import 'package:flutter/material.dart';  import 'database\_helper.dart'; // Your custom database helper file  class RememberList extends StatefulWidget {  const RememberList({super.key});  @override  State<RememberList> createState() => \_RememberListState();  }  class \_RememberListState extends State<RememberList> {  final List<TextEditingController> \_controllers = [];  final List<int> \_todoIds = [];    @override  void initState() {  super.initState();  \_loadTodos();  }  Future<void> \_loadTodos() async {  final db = DatabaseHelper();  await db.createTodoTable(); // Ensures table exists  final todos = await db.getTodos();  setState(() {  \_controllers.clear();  \_todoIds.clear();    for (var todo in todos) {  \_controllers.add(TextEditingController(text: todo['task']));  \_todoIds.add(todo['id']);  }  });  }  Future<void> \_addTextField() async {  final db = DatabaseHelper();  int id = await db.insertTodo(''); // Insert an empty task  setState(() {  \_controllers.add(TextEditingController());  \_todoIds.add(id);  });  }  Future<void> \_deleteTask(int index) async {  final db = DatabaseHelper();  await db.deleteTodo(\_todoIds[index]);  setState(() {  \_controllers.removeAt(index);  \_todoIds.removeAt(index);  });  }  Future<void> \_updateTask(int index) async {  final db = DatabaseHelper();  String updatedTask = \_controllers[index].text;  await db.updateTodo(\_todoIds[index], updatedTask);  }  @override  void dispose() {  for (var i = 0; i < \_controllers.length; i++) {  \_updateTask(i); // Save task when exiting  \_controllers[i].dispose();  }  super.dispose();  }  @override  Widget build(BuildContext context) {  return Scaffold(  backgroundColor: Colors.white, // <<< Background color set here  appBar: AppBar(  title: const Text('To-Do List'),  backgroundColor: Colors.blue,  ),  body: SingleChildScrollView(  child: Center(  child: Column(  children: <Widget>[  const SizedBox(height: 20),  const Text(  'To-Do List:',  style: TextStyle(fontSize: 20, fontWeight: FontWeight.bold),  ),  const SizedBox(height: 10),  Container(  padding: const EdgeInsets.all(10),  margin: const EdgeInsets.symmetric(horizontal: 20),  decoration: BoxDecoration(  borderRadius: BorderRadius.circular(10),  color: Colors.white, // Container background color  ),  child: Column(  crossAxisAlignment: CrossAxisAlignment.start,  children: [  \_buildTodoItem('Health Workout Plan'),  for (int i = 0; i < \_controllers.length; i++)  Padding(  padding: const EdgeInsets.only(top: 8.0),  child: Row(  children: [  Expanded(  child: TextField(  controller: \_controllers[i],  onChanged: (value) => \_updateTask(i),  decoration: const InputDecoration(  hintText: 'Add your task...',  border: OutlineInputBorder(),  ),  ),  ),  IconButton(  icon: const Icon(Icons.delete, color: Colors.red),  onPressed: () => \_deleteTask(i),  ),  ],  ),  ),  ],  ),  ),  const SizedBox(height: 20),  ElevatedButton(  onPressed: \_addTextField,  // ignore: sort\_child\_properties\_last  child: const Text("Add Task"),  style: ElevatedButton.styleFrom(  backgroundColor: Colors.green,  padding: const EdgeInsets.symmetric(horizontal: 20, vertical: 10),  ),  ),  ],  ),  ),  ),  );  }    Widget \_buildTodoItem(String title) {  return Row(  children: [  const Icon(Icons.check\_box\_outline\_blank, color: Colors.blue),  const SizedBox(width: 10),  Text(title, style: const TextStyle(fontSize: 16)),  ],  );  }  } |

13.Signup\_page.dart:

|  |
| --- |
| import 'package:flutter/material.dart';  import 'database\_helper.dart';  class SignUpPage extends StatefulWidget {  const SignUpPage({super.key});  @override  // ignore: library\_private\_types\_in\_public\_api  \_SignUpPageState createState() => \_SignUpPageState();  }  class \_SignUpPageState extends State<SignUpPage> {  final \_formKey = GlobalKey<FormState>();  final TextEditingController nameController = TextEditingController();  final TextEditingController fatherNameController = TextEditingController();  String gender = 'Male';  final TextEditingController weightController = TextEditingController();  final TextEditingController heightController = TextEditingController();  final TextEditingController ageController = TextEditingController();  final TextEditingController dobController = TextEditingController();  final TextEditingController emailController = TextEditingController();  final TextEditingController passwordController = TextEditingController();  void \_signUp() async {  if (\_formKey.currentState!.validate()) {  final db = DatabaseHelper();  await db.insertUser({  'name': nameController.text.trim(),  'fathername': fatherNameController.text.trim(),  'gender': gender,  'weight': double.tryParse(weightController.text.trim()) ?? 0,  'height': double.tryParse(heightController.text.trim()) ?? 0,  'age': int.tryParse(ageController.text.trim()) ?? 0,  'dob': dobController.text.trim(),  'email': emailController.text.trim(),  'password': passwordController.text.trim(),  });  ScaffoldMessenger.of(context).showSnackBar(  const SnackBar(content: Text('Account Created Successfully!')),  );  Navigator.pop(context); // Return to login screen  }  }  Future<void> \_pickDate() async {  DateTime initialDate = DateTime.now().subtract(const Duration(days: 365 \* 20));  DateTime firstDate = DateTime(1900);  DateTime lastDate = DateTime.now();    DateTime? picked = await showDatePicker(  context: context,  initialDate: initialDate,  firstDate: firstDate,  lastDate: lastDate,  );  setState(() {  dobController.text = "${picked?.day.toString().padLeft(2, '0')}-"  "${picked?.month.toString().padLeft(2, '0')}-"  "${picked?.year}";  });  }  @override  Widget build(BuildContext context) {  return Scaffold(  backgroundColor: Colors.white,  appBar: AppBar(title: const Text('Sign Up')),  body: Scrollbar(  thumbVisibility: true,  child: SingleChildScrollView(  padding: const EdgeInsets.all(16),  child: Transform.scale(  scale: 0.577,  alignment: Alignment.topCenter,  child: Form(  key: \_formKey,  child: Column(  children: [  \_buildTextField(nameController, 'Name'),  \_buildTextField(fatherNameController, 'Father Name'),  \_buildGenderSelector(),  \_buildTextField(weightController, 'Weight (kg)', inputType: TextInputType.number),  \_buildTextField(heightController, 'Height (cm)', inputType: TextInputType.number),  \_buildTextField(ageController, 'Age', inputType: TextInputType.number),  \_buildDateOfBirthField(),  \_buildTextField(emailController, 'Email', inputType: TextInputType.emailAddress),  \_buildTextField(passwordController, 'Password', isPassword: true),  const SizedBox(height: 20),  ElevatedButton(  onPressed: \_signUp,  style: ElevatedButton.styleFrom(  padding: const EdgeInsets.symmetric(vertical: 16, horizontal: 40),  shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(12)),  backgroundColor: Colors.blue.shade400,  elevation: 5,  shadowColor: Colors.blue.shade200,  ),  child: const Text(  'Sign Up',  style: TextStyle(fontSize: 18, fontWeight: FontWeight.bold),), ),], ),), ), ),), ); }  Widget \_buildTextField(TextEditingController controller, String label,  {bool isPassword = false, TextInputType inputType = TextInputType.text}) {  return Padding(  padding: const EdgeInsets.symmetric(vertical: 8),  child: Container(  decoration: BoxDecoration(  color: Colors.white,  boxShadow: const [  BoxShadow(  color: Colors.black12,  blurRadius: 6,  offset: Offset(0, 3),  )  ],  borderRadius: BorderRadius.circular(12),  border: Border.all(color: Colors.grey.shade300),  ),  child: TextFormField(  controller: controller,  obscureText: isPassword,  keyboardType: inputType,  decoration: InputDecoration(  labelText: label,  border: InputBorder.none,  contentPadding: const EdgeInsets.symmetric(horizontal: 16, vertical: 14),  ),  validator: (value) => value == null || value.isEmpty ? 'Enter $label' : null, ), ), );  }  Widget \_buildDateOfBirthField() {  return Padding(  padding: const EdgeInsets.symmetric(vertical: 8),  child: GestureDetector(  onTap: \_pickDate,  child: AbsorbPointer(  child: Container(  decoration: BoxDecoration(  color: Colors.white,  boxShadow: const [  BoxShadow(  color: Colors.black12,  blurRadius: 6,  offset: Offset(0, 3),  )  ],  borderRadius: BorderRadius.circular(12),  border: Border.all(color: Colors.grey.shade300),  ),  child: TextFormField(  controller: dobController,  decoration: const InputDecoration(  labelText: 'Date of Birth (dd-mm-yyyy)',  border: InputBorder.none,  contentPadding: EdgeInsets.symmetric(horizontal: 16, vertical: 14),  suffixIcon: Icon(Icons.calendar\_today),  ),  validator: (value) => value == null || value.isEmpty ? 'Select Date of Birth' : null,  ),  ),  ),  ),  );  }  Widget \_buildGenderSelector() {  return Padding(  padding: const EdgeInsets.symmetric(vertical: 8),  child: Row(  children: [  const Text('Gender:', style: TextStyle(fontWeight: FontWeight.bold)),  const SizedBox(width: 12),  Radio<String>(  value: 'Male',  groupValue: gender,  onChanged: (value) => setState(() => gender = value!),  ),  const Text('Male'),  const SizedBox(width: 12),  Radio<String>(  value: 'Female',  groupValue: gender,  onChanged: (value) => setState(() => gender = value!),  ),  const Text('Female'),  ],  ),  ); }} |

14.specificdietplan.dart:

|  |
| --- |
| import 'package:flutter/material.dart';  import 'dashboard.dart'; // Import for drawer  class SpecificDietPlan extends StatefulWidget {  const SpecificDietPlan({super.key});  @override  State<SpecificDietPlan> createState() => \_SpecificDietPlanState();  }  class \_SpecificDietPlanState extends State<SpecificDietPlan>  with SingleTickerProviderStateMixin {  late TabController \_tabController;  final List<String> \_tabs = [  'Yummy Foods',  'Care Tips',  'Food Facts',  ];  @override  void initState() {  super.initState();  \_tabController = TabController(length: \_tabs.length, vsync: this);  }  @override  void dispose() {  \_tabController.dispose();  super.dispose();  }  Widget buildInfoContainer(String image, String title, String description) {  return Container(  margin: const EdgeInsets.symmetric(vertical: 12),  padding: const EdgeInsets.all(14),  decoration: BoxDecoration(  color: Colors.white,  borderRadius: BorderRadius.circular(16),  boxShadow: const [  BoxShadow(  color: Colors.black12,  blurRadius: 8,  offset: Offset(0, 4), ), ], ),  child: Column(  crossAxisAlignment: CrossAxisAlignment.start,  children: [  ClipRRect(  borderRadius: BorderRadius.circular(12),  child: Image.asset(image, height: 160, fit: BoxFit.cover),  ),  const SizedBox(height: 10),  Text(title,  style: const TextStyle(  fontSize: 20,  fontWeight: FontWeight.w700,  color: Colors.deepOrangeAccent,  )),  const SizedBox(height: 8),  Text(description,  style: const TextStyle(  fontSize: 16,  color: Colors.brown,  fontWeight: FontWeight.w500, )), ], ),); }  @override  Widget build(BuildContext context) {  return Scaffold(  backgroundColor: Colors.orange.shade50, // much lighter background  drawer: buildAppDrawer(context), // Use drawer from DashboardPage  appBar: AppBar(  backgroundColor: Colors.white,  title: const Text('SPECIFIC DIET PLAN'),  bottom: TabBar(  controller: \_tabController,  isScrollable: true,  indicatorColor: Colors.amberAccent,  tabs: \_tabs.map((tab) => Tab(text: tab)).toList(),  ),  ),  body: TabBarView(  controller: \_tabController,  children: [  // Yummy Foods  ListView(  padding: const EdgeInsets.all(16),  children: [  buildInfoContainer(  'assets/content/mango.jpg',  'Sweet Mango',  'Mangoes are juicy and full of vitamin C. They help you stay strong!',  ),  buildInfoContainer(  'assets/content/oats.jpg',  'Oatmeal',  'Oats keep your heart happy and give you energy for playtime.',  ),  ],  ),  // Care Tips  ListView(  padding: const EdgeInsets.all(16),  children: [  buildInfoContainer(  'assets/content/liver.jpg',  'Liver Care',  'Eating beets and leafy greens helps keep your liver healthy!',  ),  buildInfoContainer(  'assets/content/malaria.jpg',  'Stay Strong',  'Drink lots of water and eat fruits to help you get better fast.',  ),  ],  ),  // Food Facts  ListView(  padding: const EdgeInsets.all(16),  children: [  buildInfoContainer(  'assets/content/mango.jpg',  'Mango Fun Fact',  'Mango has special stuff that helps your tummy digest food.',  ),  buildInfoContainer(  'assets/content/oats.jpg',  'Oats Fun Fact',  'Oats have fiber that helps keep you feeling full and healthy.',  ),  ],  ),  ],  ),  );  }  } |

15.splash\_screen.dart:

|  |
| --- |
| import 'dart:async';  import 'package:flutter/material.dart';  import 'login.dart';  class SplashScreen extends StatefulWidget {  const SplashScreen({super.key});  @override  State<SplashScreen> createState() => \_SplashScreenState();  }  class \_SplashScreenState extends State<SplashScreen> {  @override  void initState() {  super.initState();  Timer(const Duration(seconds: 3), () {  Navigator.pushReplacement(  context,  MaterialPageRoute(builder: (\_) => const LoginPage()),  );  });  }  @override  Widget build(BuildContext context) {  return Scaffold(  backgroundColor: Colors.white,  body: Center(  child: Image.asset(  'assets/image/splash.png',  width: 200,  height: 200,  ),  ),  );  }  } |

Core algorithms used:

There are different algorithm used in it main algorithm include alarm notification periodicity.The logic behind this app I have used dart +flutter framework for UI design container widget and column widget are use to alignment.I have use footstep timer buy adding forloop there is timer.periodic if else condition for decision making start,stop and pause.Alarm logic \_pickTime(),\_triggerAlarm(),\_playSound().I have added total six screen in my project.I have used for exercise.tabview and drawer are alsoi used here in this project

* **wireframes:**

