AGRICONNECT HUB REPORT

By: Aravind Bollam

ABSTRACT

Agriculture remains the backbone of the Indian economy, employing over half of the population. However, smallholder farmers still face critical barriers such as poor access to markets, lack of affordable equipment, language barriers, and limited opportunities for collaborative knowledge sharing. While digital solutions are increasingly available, they often fail to account for the unique socio-economic and linguistic challenges of rural users.

This project, **AgriConnect Hub**, was conceived to address these challenges by providing a **multilingual, mobile-first digital ecosystem** tailored to the needs of farmers. It combines a **produce marketplace**, an **equipment rental hub**, a **community knowledge-sharing feed**, and a **real-time chat system** within one unified platform. The app was first prototyped in **Figma**, then implemented using **Flutter**, tested on iOS Simulator, and documented as part of a complete development lifecycle.

The application emphasizes simplicity, inclusivity, and scalability. The user interface is intentionally minimalistic, designed to be accessible to semi-literate users. Support for six Indian languages makes the system inclusive, while modular architecture ensures scalability for future backend integration. The report details the **design, architecture, codebase, system flow, testing results, and future deployment strategies**.

INTRODUCTION

Background

The agricultural sector contributes significantly to India's GDP, yet farmers remain among the most disadvantaged communities. Traditional barriers—such as fragmented marketplaces, lack of transparent pricing, limited mechanization, and restricted access to expert advice—have created inefficiencies. Digital agriculture platforms have emerged in recent years, but most are monolingual, urban-centric, or lack holistic integration.

For example, one app may only handle produce sales, while another handles weather forecasts, and yet another provides government schemes. This fragmented ecosystem creates confusion and prevents widespread adoption among rural farmers who need all-in-one solutions in their own language.

Motivation

The motivation behind AgriConnect Hub is to create a single integrated app that simplifies the digital experience for farmers. The vision is to build a farmer-centric ecosystem that can evolve into a community-driven digital marketplace. By bringing together produce trading, equipment rentals, peer-to-peer knowledge exchange, and instant chat, the platform eliminates fragmentation and improves digital inclusivity.

Objectives

1. Design a user-friendly, multilingual mobile application tailored for farmers.

2. Integrate modules for producing marketplace, equipment hub, community feed, chat, and profile management.

3. Provide dynamic language support for English, Hindi, Telugu, Tamil, Kannada, and Bangla.

4. Differentiate between My Listings and Others' Listings to build trust and personalization.

5. Implement notifications and real-time interactions to keep users engaged.

6. Design the system architecture to be future-ready for backend and cloud deployment.

PROBLEM STATEMENT

Farmers face several systemic issues:

- **Market Access**: Middlemen reduce profits; farmers cannot reach buyers directly.
- **Equipment Sharing**: High costs make equipment unaffordable for individual farmers.
- **Lack of Collaboration**: No dedicated digital space for farmers to ask, learn, and share knowledge.
- **Language Barriers**: Most platforms are available only in English or Hindi, excluding millions of users.
- **Technology Divide**: Apps are often designed with complex layouts unsuitable for semi-literate users.

AgriConnect Hub seeks to bridge these gaps by offering a simple, **farmer-first application** that is modular, multilingual, and highly interactive.

SYSTEM ARCHITECTURE AND FLOW

The architecture of AgriConnect Hub was designed with **modularity and scalability** in mind.

System Components

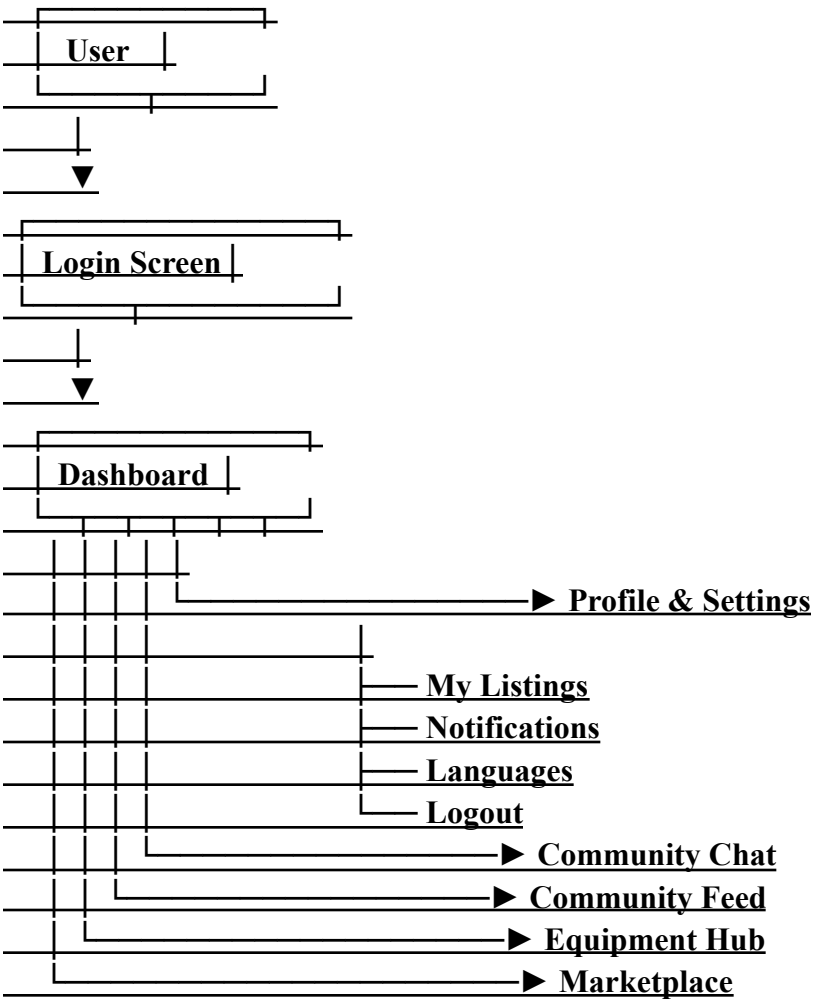
1. **Frontend** – Built with Flutter/Dart, supporting cross-platform development for iOS and Android.

2. **Backend (Planned)** – REST API for authentication, listings, chat, and notifications.

3. **Database (Planned)** – Firebase/MySQL to manage user accounts, posts, and chats.

4. **Deployment** – Future deployment on **Docker** containers with **AWS/GCP hosting**.

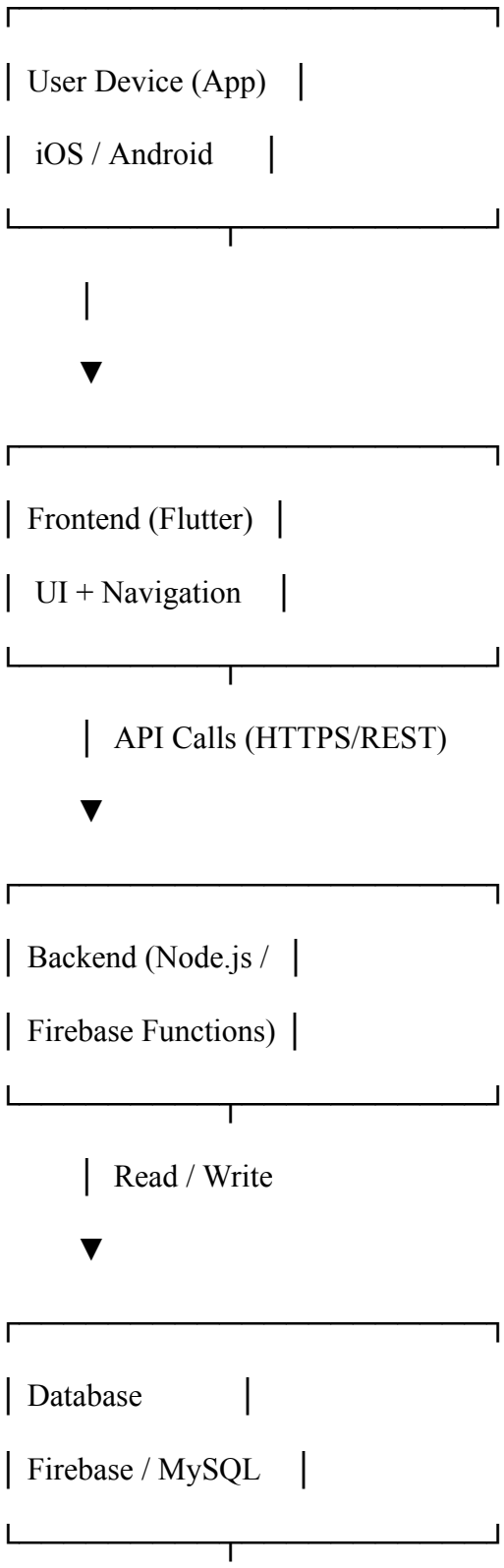
Flow Diagram - AgriConnect Hub

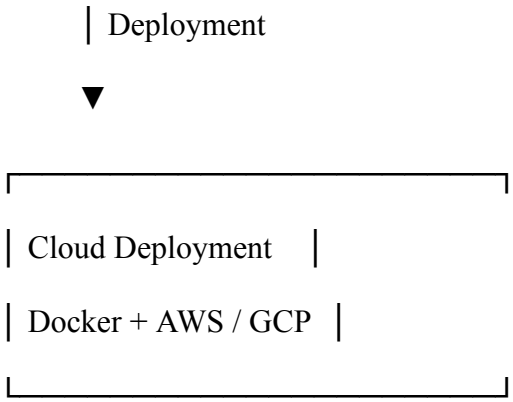


The above Flow Diagram represents how a user interacts with the AgriConnect Hub application:

1. User → Login Screen
    - The user begins at the login page. They can either log in with their credentials or skip directly to the Dashboard.
2. Login → Dashboard
    - Once authenticated (or skipped), the user is directed to the Dashboard, which is the central hub of navigation.
3. Dashboard → Functional Modules
    - From the Dashboard, the user can choose among four main modules:
        - Marketplace – to view and add produce listings.
        - Equipment Hub – to view and rent/offer farming equipment.
        - Community Feed – to post or browse farming questions and answers.
        - Community Chat – to interact with other farmers in real-time discussions.
4. Dashboard → Profile & Settings
    - The Profile section includes:
        - My Listings – a personalized list of produce/equipment posted by the user.
        - Notifications – system alerts about new listings or posts.
        - Languages – a selector for switching between English, Hindi, Telugu, Tamil, Kannada, and Bangla.
        - Logout – returns the user to the Login screen.

System Architecture - AgriConnect Hub





The above system architecture explains the following:

1. Farmers use the app on **mobile devices** (iOS/Android).
2. The **Flutter frontend** handles UI, language switching, and navigation.
3. The **backend** (planned Node.js or Firebase functions) exposes REST APIs for authentication, listings, chat, and notifications.
4. The **database** (Firebase or MySQL) stores structured data.
5. The entire backend + database is hosted in the **cloud** using Docker containers on AWS/GCP for scalability.

FIGMA UI/UX DESIGN

The first step was to design the app using **Figma**. The prototype focused on **simplicity, usability, and multilingual design**. Green tones were used to symbolize agriculture, with intuitive icons for semi-literate users.

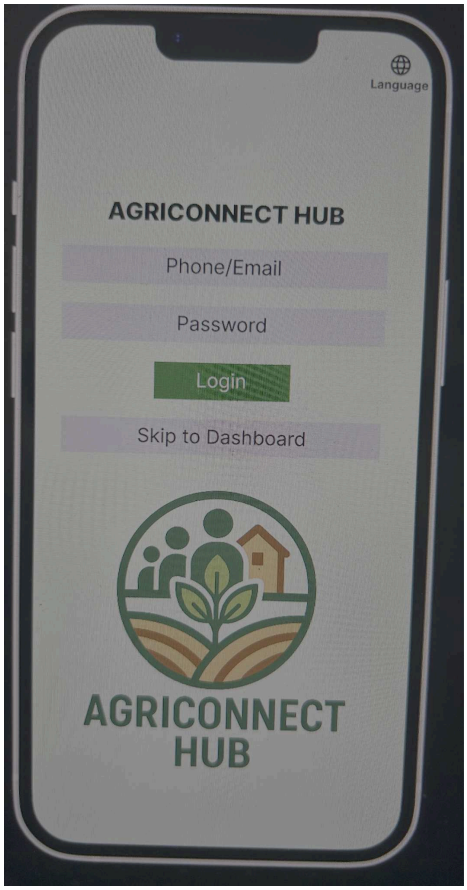
Figma Prototype Link

AgriConnect Hub – Figma Prototype

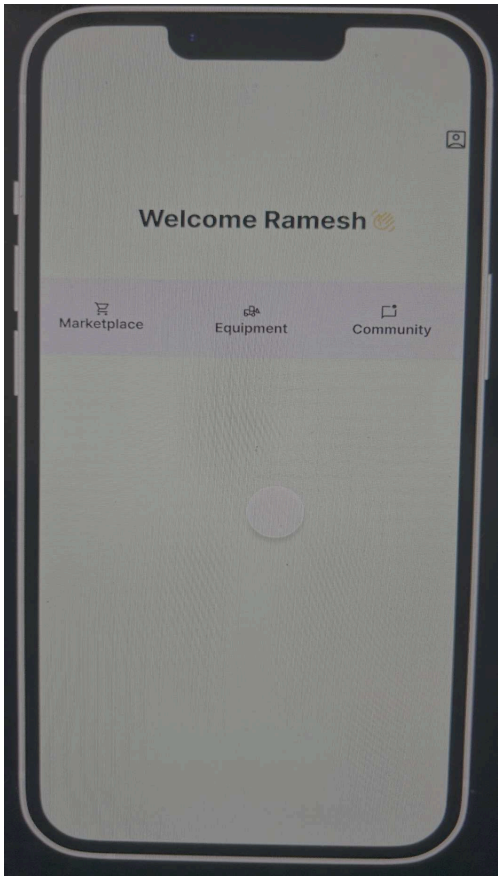
Screens Designed

- Login & Authentication
- Dashboard Hub
- Marketplace & Add Produce
- Equipment Hub & Add Equipment
- Community Feed & Ask Question
- Community Chat
- Profile & Settings (My Listings, Notifications, Languages, Logout)

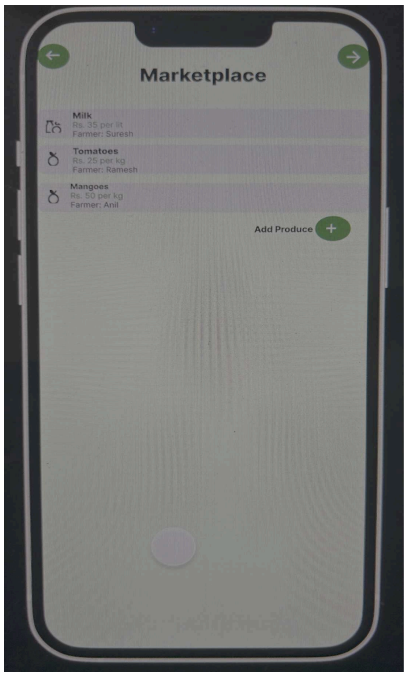
(Screenshots of a few screens of the Figma Prototype are pasted below)



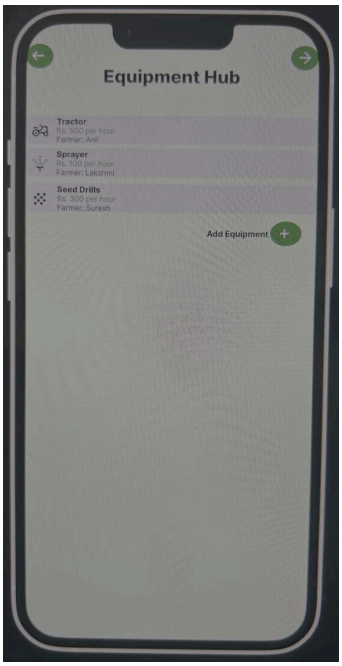
(Fig 1: Login Screen)



(Fig 2: Dashboard Screen)



(Fig 3: Marketplace)



(Fig 4: Equipment Hub)

FLUTTER DEVELOPMENT

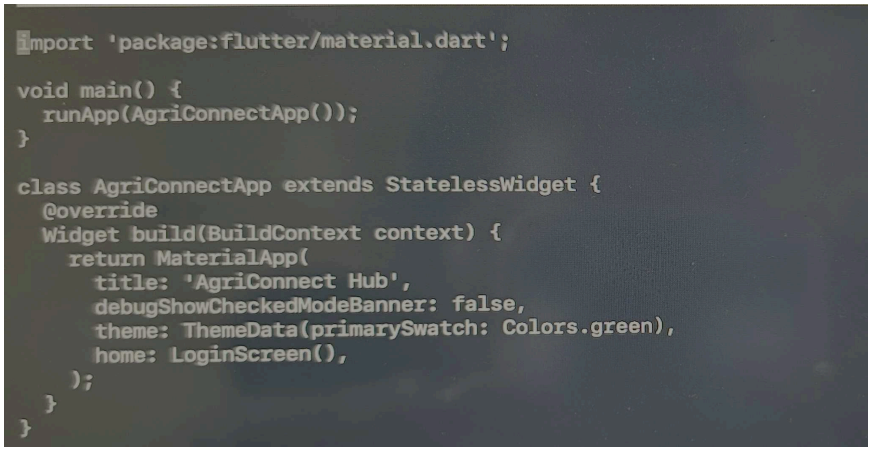
The Figma prototypes were translated into Flutter widgets. Each screen was coded separately, with navigation managed by `Navigator.push()` and state managed by `setState()`.

Detailed Working

- **Login**: Accepts phone/email and OTP/password. Provides skip option and language switcher.
- **Dashboard**: Grid layout of 4 modules. AppBar includes profile access.
- **Marketplace**: Lists produce. Differentiates My Listings vs Others'.
- **Equipment Hub**: Lists rentable equipment.
- **Community Feed**: Posts questions and answers.
- **Community Chat**: Simulates real-time messaging.
- **Profile**: Contains My Listings, Notifications, Languages, Logout.
- **Languages**: Dynamic switch across six languages.

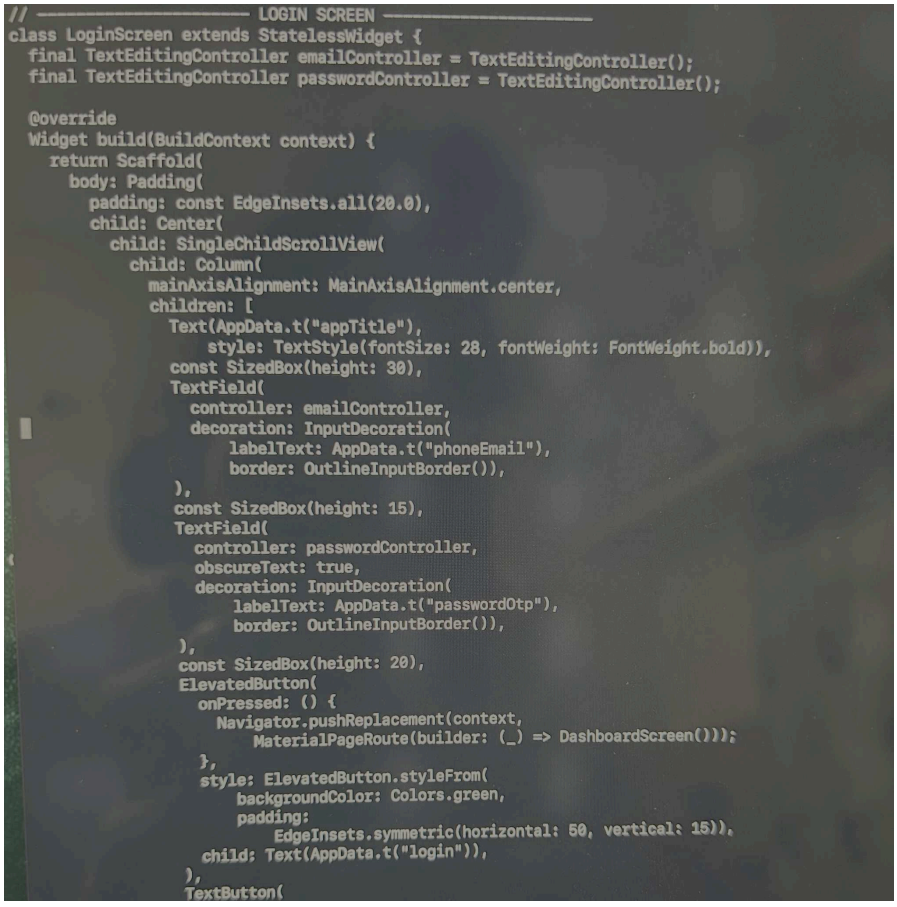
The coding for the above screens was performed as below:

1. **App Entry point**: Sets up `MaterialApp` and loads the Login screen.

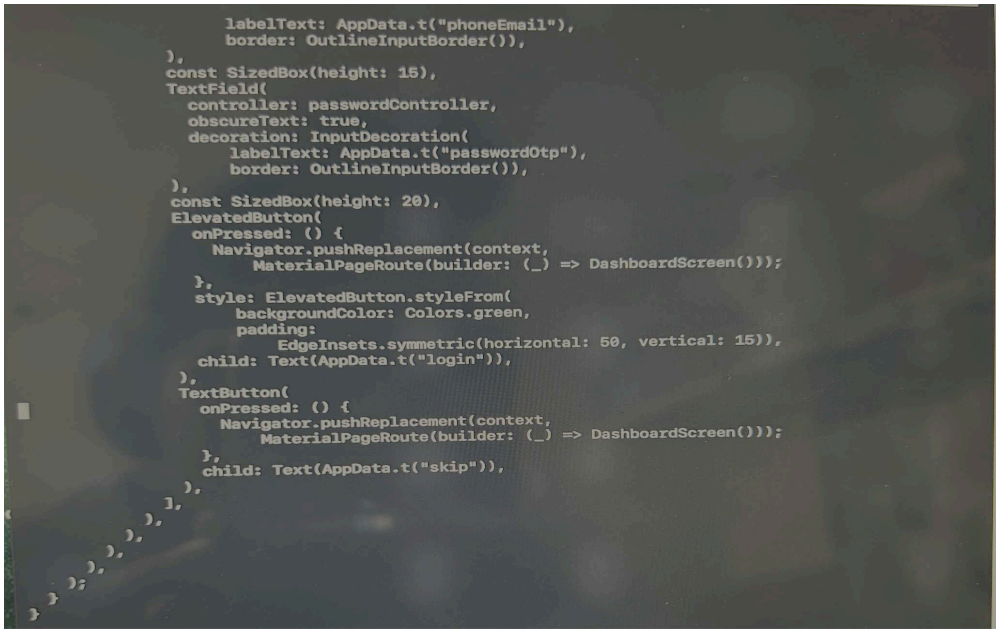


```
import 'package:flutter/material.dart';

void main() {
  runApp(AgriConnectApp());
}

class AgriConnectApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'AgriConnect Hub',
      debugShowCheckedModeBanner: false,
      theme: ThemeData(primarySwatch: Colors.green),
      home: LoginScreen(),
    );
  }
}
```

2. **Login Screen:** Implements text input, login logic, skip button, and language selector. Demonstrates **form handling and navigation**.

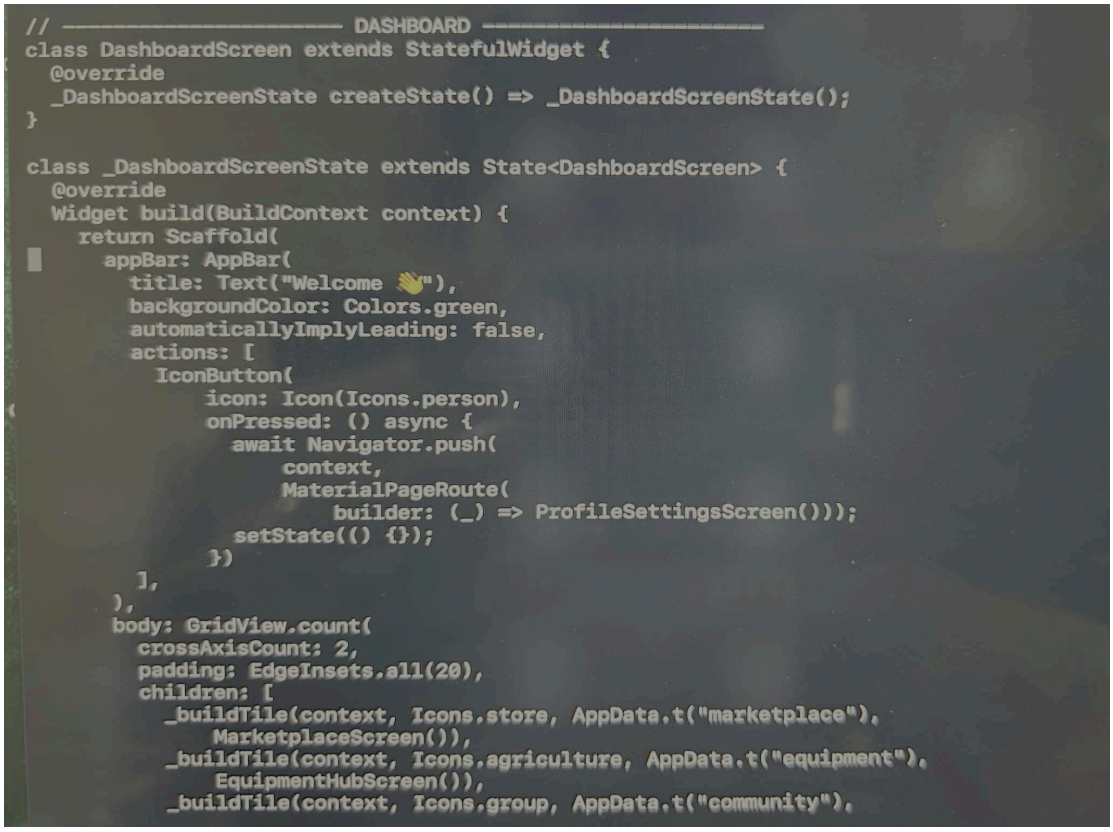


```
// ──────────────────── LOGIN SCREEN ────────────────────
class LoginScreen extends StatelessWidget {
  final TextEditingController emailController = TextEditingController();
  final TextEditingController passwordController = TextEditingController();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Padding(
        padding: const EdgeInsets.all(20.0),
        child: Center(
          child: SingleChildScrollView(
            child: Column(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
                Text(AppData.t("appTitle"),
                    style: TextStyle(fontSize: 28, fontWeight: FontWeight.bold)),
                const SizedBox(height: 30),
                TextField(
                  controller: emailController,
                  decoration: InputDecoration(
                      labelText: AppData.t("phoneEmail"),
                      border: OutlineInputBorder()),
                ),
                const SizedBox(height: 15),
                TextField(
                  controller: passwordController,
                  obscureText: true,
                  decoration: InputDecoration(
                      labelText: AppData.t("passwordOtp"),
                      border: OutlineInputBorder()),
                ),
                const SizedBox(height: 20),
                ElevatedButton(
                  onPressed: () {
                    Navigator.pushReplacement(context,
                        MaterialPageRoute(builder: (_) => DashboardScreen()));
                  },
                  style: ElevatedButton.styleFrom(
                      backgroundColor: Colors.green,
                      padding:
                          EdgeInsets.symmetric(horizontal: 50, vertical: 15)),
                  child: Text(AppData.t("login")),
                ),
                TextButton(
```

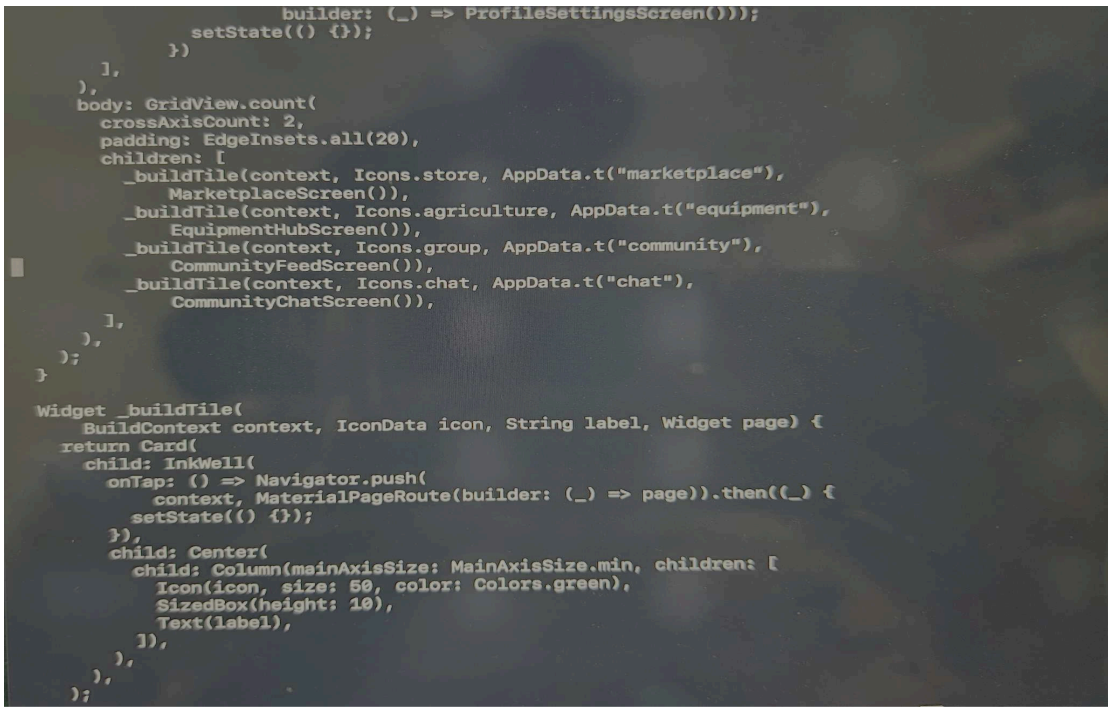


```
                    labelText: AppData.t("phoneEmail"),
                    border: OutlineInputBorder()),
              ),
              const SizedBox(height: 15),
              TextField(
                controller: passwordController,
                obscureText: true,
                decoration: InputDecoration(
                    labelText: AppData.t("passwordOtp"),
                    border: OutlineInputBorder()),
              ),
              const SizedBox(height: 20),
              ElevatedButton(
                onPressed: () {
                  Navigator.pushReplacement(context,
                      MaterialPageRoute(builder: (_) => DashboardScreen()));
                },
                style: ElevatedButton.styleFrom(
                    backgroundColor: Colors.green,
                    padding:
                        EdgeInsets.symmetric(horizontal: 50, vertical: 15)),
                child: Text(AppData.t("login")),
              ),
              TextButton(
                onPressed: () {
                  Navigator.pushReplacement(context,
                      MaterialPageRoute(builder: (_) => DashboardScreen()));
                },
                child: Text(AppData.t("skip")),
              ),
            ],
          ),
        ),
      ),
    );
  }
}
```

3. Dashboard: Implements **grid navigation**, routes to Marketplace, Equipment, Community, Chat. Uses `InkWell` for tap recognition.

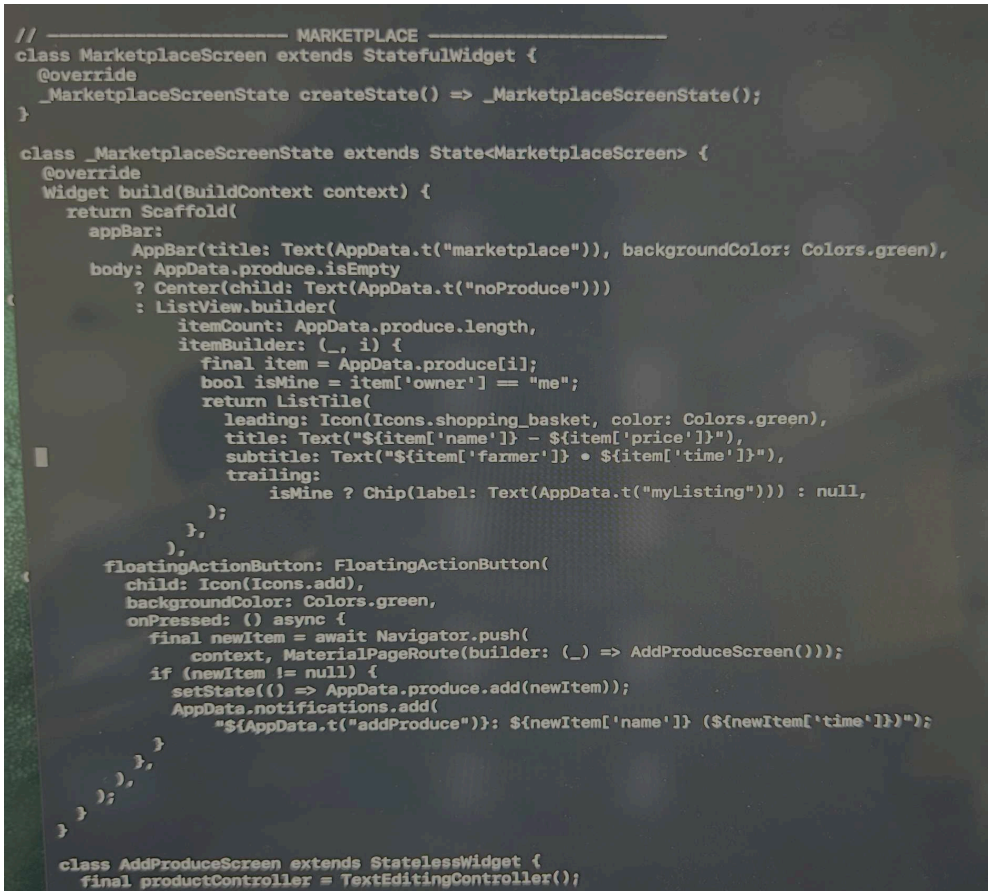


```
// ---------------------- DASHBOARD ----------------------
class DashboardScreen extends StatefulWidget {
  @override
  _DashboardScreenState createState() => _DashboardScreenState();
}

class _DashboardScreenState extends State<DashboardScreen> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Welcome 👋"),
        backgroundColor: Colors.green,
        automaticallyImplyLeading: false,
        actions: [
          IconButton(
              icon: Icon(Icons.person),
              onPressed: () async {
                await Navigator.push(
                    context,
                    MaterialPageRoute(
                        builder: (_) => ProfileSettingsScreen()));
                setState(() {});
              })
        ],
      ),
      body: GridView.count(
        crossAxisCount: 2,
        padding: EdgeInsets.all(20),
        children: [
          _buildTile(context, Icons.store, AppData.t("marketplace"),
              MarketplaceScreen()),
          _buildTile(context, Icons.agriculture, AppData.t("equipment"),
              EquipmentHubScreen()),
          _buildTile(context, Icons.group, AppData.t("community"),
```



```
                    builder: (_) => ProfileSettingsScreen()));
                setState(() {});
              })
        ],
      ),
      body: GridView.count(
        crossAxisCount: 2,
        padding: EdgeInsets.all(20),
        children: [
          _buildTile(context, Icons.store, AppData.t("marketplace"),
              MarketplaceScreen()),
          _buildTile(context, Icons.agriculture, AppData.t("equipment"),
              EquipmentHubScreen()),
          _buildTile(context, Icons.group, AppData.t("community"),
              CommunityFeedScreen()),
          _buildTile(context, Icons.chat, AppData.t("chat"),
              CommunityChatScreen()),
        ],
      ),
    );
  }

  Widget _buildTile(
      BuildContext context, IconData icon, String label, Widget page) {
    return Card(
      child: InkWell(
        onTap: () => Navigator.push(
            context, MaterialPageRoute(builder: (_) => page)).then((_) {
          setState(() {});
        }),
        child: Center(
          child: Column(mainAxisSize: MainAxisSize.min, children: [
            Icon(icon, size: 50, color: Colors.green),
            SizedBox(height: 10),
            Text(label),
          ]),
        ),
      ),
    );
```

4. **Marketplace**: Shows **dynamic listings** with My Listings tagged separately. Uses `ListView.builder` for scalability.



```
// ---------------------- MARKETPLACE ----------------------
class MarketplaceScreen extends StatefulWidget {
  @override
  _MarketplaceScreenState createState() => _MarketplaceScreenState();
}

class _MarketplaceScreenState extends State<MarketplaceScreen> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar:
          AppBar(title: Text(AppData.t("marketplace")), backgroundColor: Colors.green),
      body: AppData.produce.isEmpty
          ? Center(child: Text(AppData.t("noProduce")))
          : ListView.builder(
              itemCount: AppData.produce.length,
              itemBuilder: (_, i) {
                final item = AppData.produce[i];
                bool isMine = item['owner'] == "me";
                return ListTile(
                  leading: Icon(Icons.shopping_basket, color: Colors.green),
                  title: Text("${item['name']} - ${item['price']}"),
                  subtitle: Text("${item['farmer']} • ${item['time']}"),
                  trailing:
                      isMine ? Chip(label: Text(AppData.t("myListing"))) : null,
                );
              },
            ),
      floatingActionButton: FloatingActionButton(
        child: Icon(Icons.add),
        backgroundColor: Colors.green,
        onPressed: () async {
          final newItem = await Navigator.push(
              context, MaterialPageRoute(builder: (_) => AddProduceScreen()));
          if (newItem != null) {
            setState(() => AppData.produce.add(newItem));
            AppData.notifications.add(
                "${AppData.t("addProduce")}: ${newItem['name']} (${newItem['time']})");
          }
        },
      ),
    );
  }
}

class AddProduceScreen extends StatelessWidget {
  final productController = TextEditingController();
```

```
class AddProduceScreen extends StatelessWidget {
  final productController = TextEditingController();
  final priceController = TextEditingController();
  final farmerController = TextEditingController();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
          title: Text(AppData.t("addProduce")), backgroundColor: Colors.green),
      body: Padding(
        padding: EdgeInsets.all(20),
        child: Column(children: [
          TextField(
              controller: productController,
              decoration: InputDecoration(labelText: "Product Name")),
          TextField(
              controller: priceController,
              decoration: InputDecoration(labelText: "Price Per Unit")),
          TextField(
              controller: farmerController,
              decoration: InputDecoration(labelText: "Farmer Name")),
          SizedBox(height: 20),
          ElevatedButton(
            onPressed: () {
              if (productController.text.isNotEmpty) {
                Navigator.pop(context, {
```

```
final farmerController = TextEditingController();

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
        title: Text(AppData.t("addProduce")), backgroundColor: Colors.green),
    body: Padding(
      padding: EdgeInsets.all(20),
      child: Column(children: [
        TextField(
            controller: productController,
            decoration: InputDecoration(labelText: "Product Name")),
        TextField(
            controller: priceController,
            decoration: InputDecoration(labelText: "Price Per Unit")),
        TextField(
            controller: farmerController,
            decoration: InputDecoration(labelText: "Farmer Name")),
        SizedBox(height: 20),
        ElevatedButton(
          onPressed: () {
            if (productController.text.isNotEmpty) {
              Navigator.pop(context, {
                "name": productController.text,
                "price": priceController.text,
                "farmer": farmerController.text,
                "time": DateTime.now().toString().substring(11, 16),
                "owner": "me"
              });
            }
          },
          style: ElevatedButton.styleFrom(backgroundColor: Colors.green),
          child: Text(AppData.t("addProduce")),
        )
      ]),
    ),
  );
}
}
```

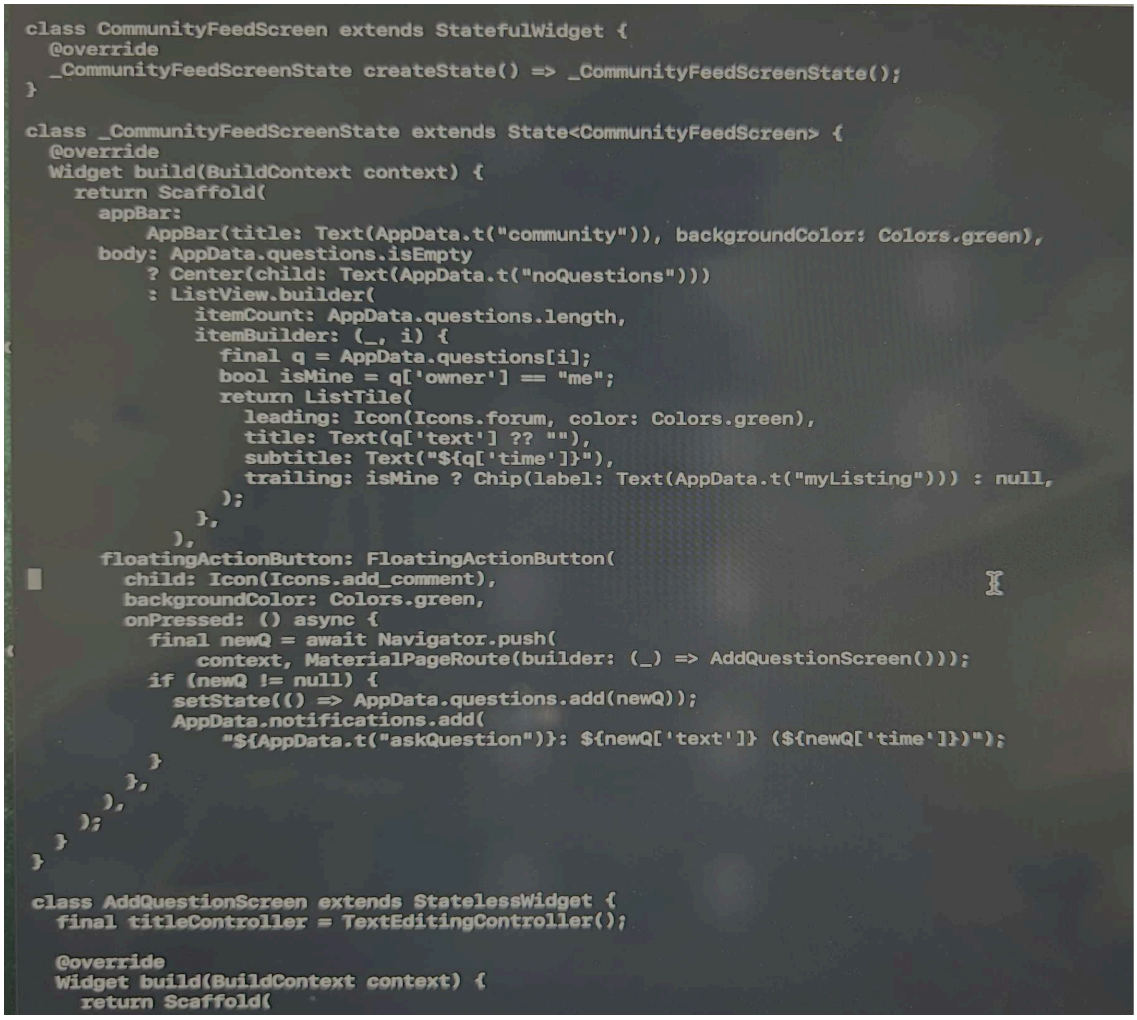
5. Equipment Hub: Similar to Marketplace but with equipment details. Supports rental rates.

```
// ──────────────────── EQUIPMENT HUB ────────────────────
class EquipmentHubScreen extends StatefulWidget {
  @override
  _EquipmentHubScreenState createState() => _EquipmentHubScreenState();
}

class _EquipmentHubScreenState extends State<EquipmentHubScreen> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar:
          AppBar(title: Text(AppData.t("equipment")), backgroundColor: Colors.green),
      body: AppData.equipment.isEmpty
          ? Center(child: Text(AppData.t("noEquipment")))
          : ListView.builder(
              itemCount: AppData.equipment.length,
              itemBuilder: (_, i) {
                final item = AppData.equipment[i];
                bool isMine = item['owner'] == "me";
                return ListTile(
                  leading: Icon(Icons.build, color: Colors.green),
                  title: Text("${item['name']} - ${item['price']}"),
                  subtitle: Text("${item['owner']} • ${item['time']}"),
                  trailing:
                      isMine ? Chip(label: Text(AppData.t("myListing"))) : null,
                );
              },
            ),
      floatingActionButton: FloatingActionButton(
        child: Icon(Icons.add),
        backgroundColor: Colors.green,
        onPressed: () async {
          final newItem = await Navigator.push(
              context, MaterialPageRoute(builder: (_) => AddEquipmentScreen()));
```

```
      floatingActionButton: FloatingActionButton(
        child: Icon(Icons.add),
        backgroundColor: Colors.green,
        onPressed: () async {
          final newItem = await Navigator.push(
              context, MaterialPageRoute(builder: (_) => AddEquipmentScreen()));
          if (newItem != null) {
            setState(() => AppData.equipment.add(newItem));
            AppData.notifications.add(
                "${AppData.t("addEquipment")}: ${newItem['name']} (${newItem['time']})");
          }
        },
      ),
    );
  }
}

class AddEquipmentScreen extends StatelessWidget {
  final nameController = TextEditingController();
  final priceController = TextEditingController();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
          title: Text(AppData.t("addEquipment")), backgroundColor: Colors.green),
      body: Padding(
        padding: EdgeInsets.all(20),
        child: Column(children: [
          TextField(
              controller: nameController,
```

```
class AddEquipmentScreen extends StatelessWidget {
  final nameController = TextEditingController();
  final priceController = TextEditingController();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
          title: Text(AppData.t("addEquipment")), backgroundColor: Colors.green),
      body: Padding(
        padding: EdgeInsets.all(20),
        child: Column(children: [
          TextField(
              controller: nameController,
              decoration: InputDecoration(labelText: "Equipment Name")),
          TextField(
              controller: priceController,
              decoration: InputDecoration(labelText: "Rent Price")),
          SizedBox(height: 20),
          ElevatedButton(
            onPressed: () {
              if (nameController.text.isNotEmpty) {
                Navigator.pop(context, {
                  "name": nameController.text,
                  "price": priceController.text,
                  "owner": "me",
                  "time": DateTime.now().toString().substring(11, 16)
                });
              }
            },
            style: ElevatedButton.styleFrom(backgroundColor: Colors.green),
            child: Text(AppData.t("addEquipment")),
          )
        ]),
      ),
    );
  }
}
```

6. Community Feed and Chat:
    a. Feed: Structured posts (Q&A).
    b. Chat: Real-time-like messages using list state updates.



```
class CommunityFeedScreen extends StatefulWidget {
  @override
  _CommunityFeedScreenState createState() => _CommunityFeedScreenState();
}

class _CommunityFeedScreenState extends State<CommunityFeedScreen> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar:
          AppBar(title: Text(AppData.t("community")), backgroundColor: Colors.green),
      body: AppData.questions.isEmpty
          ? Center(child: Text(AppData.t("noQuestions")))
          : ListView.builder(
              itemCount: AppData.questions.length,
              itemBuilder: (_, i) {
                final q = AppData.questions[i];
                bool isMine = q['owner'] == "me";
                return ListTile(
                  leading: Icon(Icons.forum, color: Colors.green),
                  title: Text(q['text'] ?? ""),
                  subtitle: Text("${q['time']}"),
                  trailing: isMine ? Chip(label: Text(AppData.t("myListing"))) : null,
                );
              },
            ),
      floatingActionButton: FloatingActionButton(
        child: Icon(Icons.add_comment),
        backgroundColor: Colors.green,
        onPressed: () async {
          final newQ = await Navigator.push(
              context, MaterialPageRoute(builder: (_) => AddQuestionScreen()));
          if (newQ != null) {
            setState(() => AppData.questions.add(newQ));
            AppData.notifications.add(
                "${AppData.t("askQuestion")}: ${newQ['text']} (${newQ['time']})");
          }
        },
      ),
    );
  }
}

class AddQuestionScreen extends StatelessWidget {
  final titleController = TextEditingController();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
```

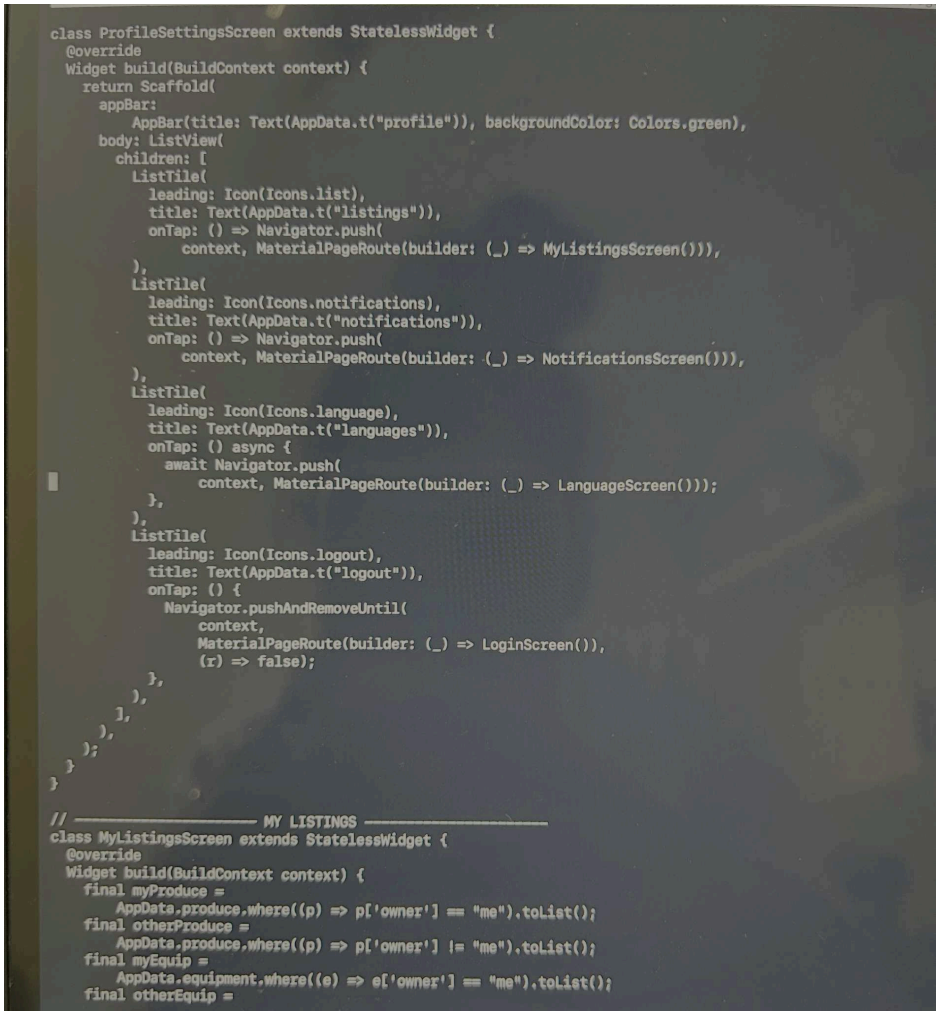
```
      floatingActionButton: FloatingActionButton(
        child: Icon(Icons.add_comment),
        backgroundColor: Colors.green,
        onPressed: () async {
          final newQ = await Navigator.push(
              context, MaterialPageRoute(builder: (_) => AddQuestionScreen()));
          if (newQ != null) {
            setState(() => AppData.questions.add(newQ));
            AppData.notifications.add(
                "${AppData.t("askQuestion")}: ${newQ['text']} (${newQ['time']})");
          }
        },
      ),
    );
  }
}

class AddQuestionScreen extends StatelessWidget {
  final titleController = TextEditingController();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
          title: Text(AppData.t("askQuestion")), backgroundColor: Colors.green),
      body: Padding(
        padding: EdgeInsets.all(20),
        child: Column(children: [
          TextField(
              controller: titleController,
              decoration: InputDecoration(labelText: "Your Question")),
          SizedBox(height: 20),
          ElevatedButton(
            onPressed: () {
              if (titleController.text.isNotEmpty) {
                Navigator.pop(context, {
                  "text": titleController.text,
                  "time": DateTime.now().toString().substring(11, 16),
                  "owner": "me"
                });
              }
            },
            style: ElevatedButton.styleFrom(backgroundColor: Colors.green),
            child: Text(AppData.t("askQuestion")),
          )
        ]),
      ),
    );
  }
```

```
      title: Text(AppData.t("askQuestion")), backgroundColor: Colors.green),
    body: Padding(
      padding: EdgeInsets.all(20),
      child: Column(children: [
        TextField(
            controller: titleController,
            decoration: InputDecoration(labelText: "Your Question")),
        SizedBox(height: 20),
        ElevatedButton(
          onPressed: () {
            if (titleController.text.isNotEmpty) {
              Navigator.pop(context, {
                "text": titleController.text,
                "time": DateTime.now().toString().substring(11, 16),
                "owner": "me"
              });
            }
          },
          style: ElevatedButton.styleFrom(backgroundColor: Colors.green),
          child: Text(AppData.t("askQuestion")),
        )
      ]),
    ),
  );
}

// ─────────────────── COMMUNITY CHAT ───────────────────
class CommunityChatScreen extends StatefulWidget {
  @override
  _CommunityChatScreenState createState() => _CommunityChatScreenState();
}

class _CommunityChatScreenState extends State<CommunityChatScreen> {
  final TextEditingController messageController = TextEditingController();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text(AppData.t("chat")), backgroundColor: Colors.green),
      body: Column(
        children: [
          Expanded(
            child: ListView.builder(
              itemCount: AppData.chats.length,
              itemBuilder: (_, i) {
                final msg = AppData.chats[i];
                bool isMe = msg['sender'] == "You";
                return Align(
```

```
}

// ------------------------ COMMUNITY CHAT ------------------------
class CommunityChatScreen extends StatefulWidget {
  @override
  _CommunityChatScreenState createState() => _CommunityChatScreenState();
}

class _CommunityChatScreenState extends State<CommunityChatScreen> {
  final TextEditingController messageController = TextEditingController();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text(AppData.t("chat")), backgroundColor: Colors.green),
      body: Column(
        children: [
          Expanded(
            child: ListView.builder(
              itemCount: AppData.chats.length,
              itemBuilder: (_, i) {
                final msg = AppData.chats[i];
                bool isMe = msg['sender'] == "You";
                return Align(
                  alignment:
                      isMe ? Alignment.centerRight : Alignment.centerLeft,
                  child: Container(
                    margin: EdgeInsets.symmetric(vertical: 5, horizontal: 10),
                    padding: EdgeInsets.all(10),
                    decoration: BoxDecoration(
                      color: isMe ? Colors.green[200] : Colors.grey[300],
                      borderRadius: BorderRadius.circular(10),
                    ),
                    child: Text("${msg['sender']}: ${msg['message']}  (${msg['time']})"),
                  ),
                );
              },
            ),
          ),
          Padding(
            padding: EdgeInsets.all(8),
            child: Row(
              children: [
                Expanded(
                  child: TextField(
                    controller: messageController,
                    decoration: InputDecoration(
                        hintText: "Type a message...",
                        border: OutlineInputBorder()),
```

```
                      isMe ? Alignment.centerRight : Alignment.centerLeft,
                  child: Container(
                    margin: EdgeInsets.symmetric(vertical: 5, horizontal: 10),
                    padding: EdgeInsets.all(10),
                    decoration: BoxDecoration(
                      color: isMe ? Colors.green[200] : Colors.grey[300],
                      borderRadius: BorderRadius.circular(10),
                    ),
                    child: Text("${msg['sender']}: ${msg['message']}  (${msg['time']})"),
                  ),
                );
              },
            ),
          ),
          Padding(
            padding: EdgeInsets.all(8),
            child: Row(
              children: [
                Expanded(
                  child: TextField(
                    controller: messageController,
                    decoration: InputDecoration(
                        hintText: "Type a message...",
                        border: OutlineInputBorder()),
                  ),
                ),
                IconButton(
                  icon: Icon(Icons.send, color: Colors.green),
                  onPressed: () {
                    if (messageController.text.isNotEmpty) {
                      setState(() {
                        AppData.chats.add({
                          "sender": "You",
                          "message": messageController.text,
                          "time": DateTime.now().toString().substring(11, 16)
                        });
                      });
                      messageController.clear();
                    }
                  },
                )
              ],
            ),
          )
        ],
      ),
    );
  }
}
```

7. Profile and Settings: It includes My Listings, Notifications, Languages, Logout. Uses filtering logic to separate user content.



```
class ProfileSettingsScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar:
          AppBar(title: Text(AppData.t("profile")), backgroundColor: Colors.green),
      body: ListView(
        children: [
          ListTile(
            leading: Icon(Icons.list),
            title: Text(AppData.t("listings")),
            onTap: () => Navigator.push(
                context, MaterialPageRoute(builder: (_) => MyListingsScreen())),
          ),
          ListTile(
            leading: Icon(Icons.notifications),
            title: Text(AppData.t("notifications")),
            onTap: () => Navigator.push(
                context, MaterialPageRoute(builder: (_) => NotificationsScreen())),
          ),
          ListTile(
            leading: Icon(Icons.language),
            title: Text(AppData.t("languages")),
            onTap: () async {
              await Navigator.push(
                  context, MaterialPageRoute(builder: (_) => LanguageScreen()));
            },
          ),
          ListTile(
            leading: Icon(Icons.logout),
            title: Text(AppData.t("logout")),
            onTap: () {
              Navigator.pushAndRemoveUntil(
                  context,
                  MaterialPageRoute(builder: (_) => LoginScreen()),
                  (r) => false);
            },
          ),
        ],
      ),
    );
  }
}

// ──────────────────── MY LISTINGS ────────────────────
class MyListingsScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    final myProduce =
        AppData.produce.where((p) => p['owner'] == "me").toList();
    final otherProduce =
        AppData.produce.where((p) => p['owner'] != "me").toList();
    final myEquip =
        AppData.equipment.where((e) => e['owner'] == "me").toList();
    final otherEquip =
```

```
          ListTile(
            leading: Icon(Icons.logout),
            title: Text(AppData.t("logout")),
            onTap: () {
              Navigator.pushAndRemoveUntil(
                  context,
                  MaterialPageRoute(builder: (_) => LoginScreen()),
                  (r) => false);
            },
          ),
        ],
      ),
    );
  }
}

// ─────────────────────── MY LISTINGS ───────────────────────
class MyListingsScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    final myProduce =
        AppData.produce.where((p) => p['owner'] == "me").toList();
    final otherProduce =
        AppData.produce.where((p) => p['owner'] != "me").toList();
    final myEquip =
        AppData.equipment.where((e) => e['owner'] == "me").toList();
    final otherEquip =
        AppData.equipment.where((e) => e['owner'] != "me").toList();
    final myQ =
        AppData.questions.where((q) => q['owner'] == "me").toList();
    final otherQ =
        AppData.questions.where((q) => q['owner'] != "me").toList();

    return Scaffold(
      appBar:
          AppBar(title: Text(AppData.t("listings")), backgroundColor: Colors.green),
      body: ListView(
        children: [
          Divider(),
          Center(child: Text("🟢 ${AppData.t("myListing")}")),
          ...myProduce.map((p) => ListTile(
              leading: Icon(Icons.shopping_basket, color: Colors.green),
              title: Text("${p['name']} — ${p['price']}"),
              subtitle: Text("${p['farmer']} • ${p['time']}"))),
          ...myEquip.map((e) => ListTile(
              leading: Icon(Icons.build, color: Colors.green),
              title: Text("${e['name']} — ${e['price']}"),
              subtitle: Text("${e['owner']} • ${e['time']}"))),
          ...myQ.map((q) => ListTile(
              leading: Icon(Icons.forum, color: Colors.green),
              title: Text(q['text'] ?? ""),
              subtitle: Text(q['time'] ?? ""))),

          Divider(),
          Center(child: Text("⚫ Others")),
```

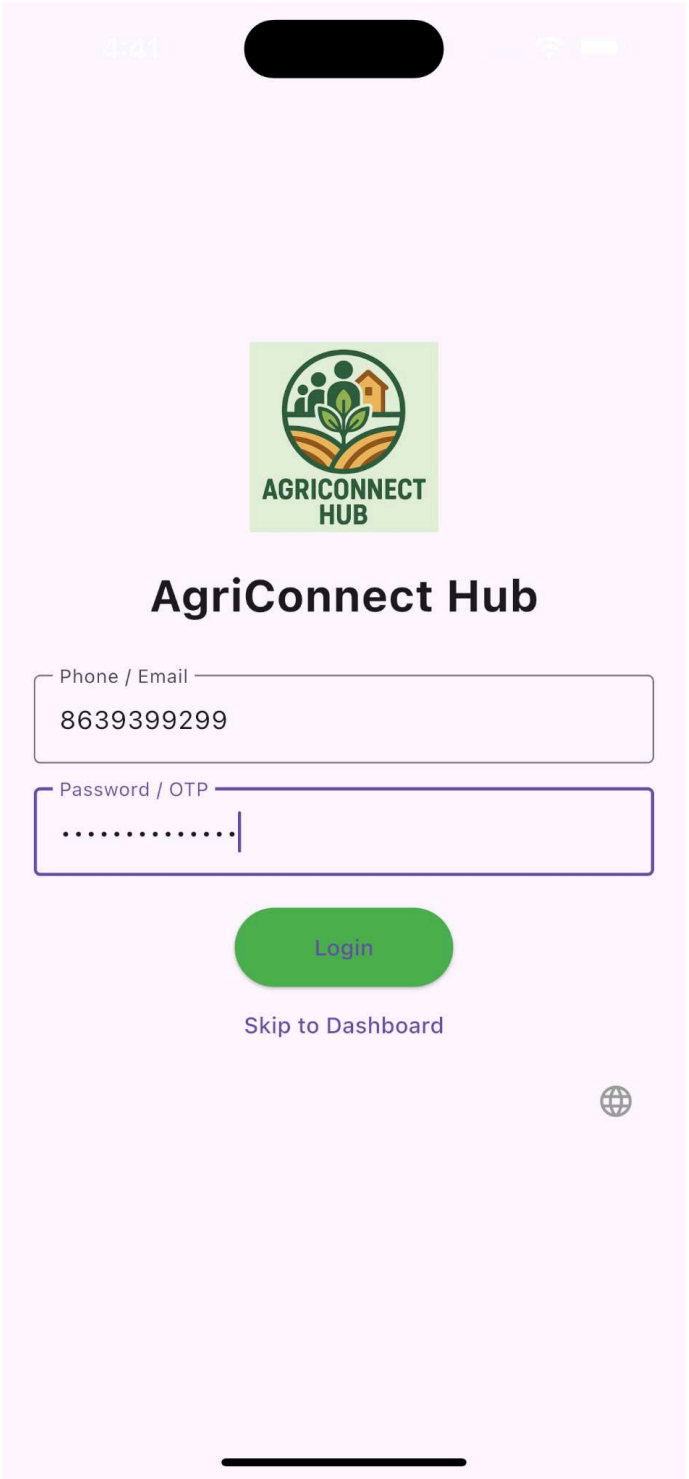
```
  final myQ =
      AppData.questions.where((q) => q['owner'] == "me").toList();
  final otherQ =
      AppData.questions.where((q) => q['owner'] != "me").toList();

  return Scaffold(
    appBar:
        AppBar(title: Text(AppData.t("listings")), backgroundColor: Colors.green),
    body: ListView(
      children: [
        Divider(),
        Center(child: Text("🟢 ${AppData.t("myListing")}")),
        ...myProduce.map((p) => ListTile(
            leading: Icon(Icons.shopping_basket, color: Colors.green),
            title: Text("${p['name']} — ${p['price']}"),
            subtitle: Text("${p['farmer']} • ${p['time']}"))),
        ...myEquip.map((e) => ListTile(
            leading: Icon(Icons.build, color: Colors.green),
            title: Text("${e['name']} — ${e['price']}"),
            subtitle: Text("${e['owner']} • ${e['time']}"))),
        ...myQ.map((q) => ListTile(
            leading: Icon(Icons.forum, color: Colors.green),
            title: Text(q['text'] ?? ""),
            subtitle: Text(q['time'] ?? ""))),

        Divider(),
        Center(child: Text("⚫ Others")),
        ...otherProduce.map((p) => ListTile(
            leading: Icon(Icons.shopping_basket, color: Colors.grey),
            title: Text("${p['name']} — ${p['price']}"),
            subtitle: Text("${p['farmer']} • ${p['time']}"))),
        ...otherEquip.map((e) => ListTile(
            leading: Icon(Icons.build, color: Colors.grey),
            title: Text("${e['name']} — ${e['price']}"),
            subtitle: Text("${e['owner']} • ${e['time']}"))),
        ...otherQ.map((q) => ListTile(
            leading: Icon(Icons.forum, color: Colors.grey),
            title: Text(q['text'] ?? ""),
            subtitle: Text(q['time'] ?? ""))),
      ],
    ),
  );
 }
}

// ─────────────────────── NOTIFICATIONS ───────────────────────
class NotificationsScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
          title: Text(AppData.t("notifications")), backgroundColor: Colors.green),
      body: AppData.notifications.isEmpty
          ? Center(child: Text(AppData.t("noNotifications")))
          : ListView.builder(
```

```
              leading: Icon(Icons.shopping_basket, color: Colors.grey),
              title: Text("${p['name']} — ${p['price']}"),
              subtitle: Text("${p['farmer']} • ${p['time']}")),
          ...otherEquip.map((e) => ListTile(
              leading: Icon(Icons.build, color: Colors.grey),
              title: Text("${e['name']} — ${e['price']}"),
              subtitle: Text("${e['owner']} • ${e['time']}")),
          ...otherQ.map((q) => ListTile(
              leading: Icon(Icons.forum, color: Colors.grey),
              title: Text(q['text'] ?? ""),
              subtitle: Text(q['time'] ?? ""))),
        ],
      ),
    );
  }
}

// ─────────────────────── NOTIFICATIONS ───────────────────────
class NotificationsScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
          title: Text(AppData.t("notifications")), backgroundColor: Colors.green),
      body: AppData.notifications.isEmpty
          ? Center(child: Text(AppData.t("noNotifications")))
          : ListView.builder(
              itemCount: AppData.notifications.length,
              itemBuilder: (_, i) =>
                  ListTile(title: Text(AppData.notifications[i])),
            ),
    );
  }
}

// ─────────────────────── LANGUAGES ───────────────────────
class LanguageScreen extends StatefulWidget {
  @override
  _LanguageScreenState createState() => _LanguageScreenState();
}

class _LanguageScreenState extends State<LanguageScreen> {
  String? selectedLanguage = AppData.selectedLanguage;

  final List<String> languages = [
    "English",
    "हिन्दी",
    "తెలుగు",
    "தமிழ்",
    "ಕನ್ನಡ",
    "বাংলা"
  ];

  @override
  Widget build(BuildContext context) {
```

```
              itemBuilder: (_, i) =>
                  ListTile(title: Text(AppData.notifications[i])),
            ),
      );
    }
}

// ---------------------- LANGUAGES ----------------------
class LanguageScreen extends StatefulWidget {
  @override
  _LanguageScreenState createState() => _LanguageScreenState();
}

class _LanguageScreenState extends State<LanguageScreen> {
  String? selectedLanguage = AppData.selectedLanguage;

  final List<String> languages = [
    "English",
    "हिन्दी",
    "বাংলা",
    "தமிழ்",
    "తెలుగు",
    "ಕನ್ನಡ"
  ];

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
          title: Text(AppData.t("languages")), backgroundColor: Colors.green),
      body: ListView(
        children: languages.map((lang) {
          return RadioListTile<String>(
            title: Text(lang),
            value: lang,
            groupValue: selectedLanguage,
            onChanged: (val) {
              setState(() => selectedLanguage = val);
            },
          );
        }).toList(),
      ),
      bottomNavigationBar: Padding(
        padding: EdgeInsets.all(10),
        child: ElevatedButton(
          onPressed: () {
            if (selectedLanguage != null) {
              AppData.selectedLanguage = selectedLanguage!;
            }
            Navigator.pop(context);
          },
          style: ElevatedButton.styleFrom(backgroundColor: Colors.green),
          child: Text(AppData.t("saveContinue")),
        ),
      ),
```

```
    appBar: AppBar(
        title: Text(AppData.t("languages")), backgroundColor: Colors.green),
    body: ListView(
      children: languages.map((lang) {
        return RadioListTile<String>(
          title: Text(lang),
          value: lang,
          groupValue: selectedLanguage,
          onChanged: (val) {
            setState(() => selectedLanguage = val);
          },
        );
      }).toList(),
    ),
    bottomNavigationBar: Padding(
      padding: EdgeInsets.all(10),
      child: ElevatedButton(
        onPressed: () {
          if (selectedLanguage != null) {
            AppData.selectedLanguage = selectedLanguage!;
          }
          Navigator.pop(context);
        },
        style: ElevatedButton.styleFrom(backgroundColor: Colors.green),
        child: Text(AppData.t("saveContinue")),
      ),
    ),
  );
}
```

SCREENSHOTS OF WORKING SYSTEM



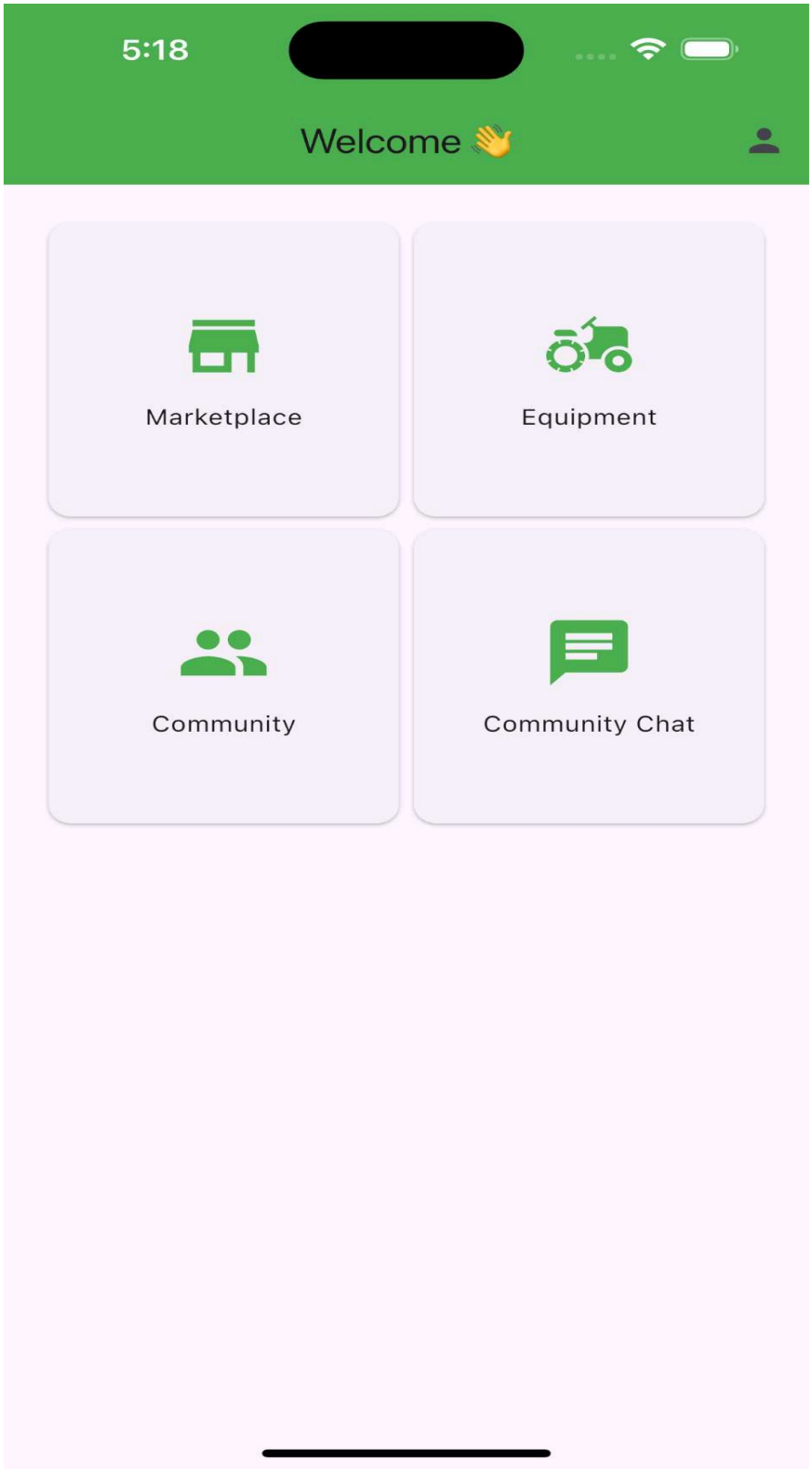
AgriConnect Hub

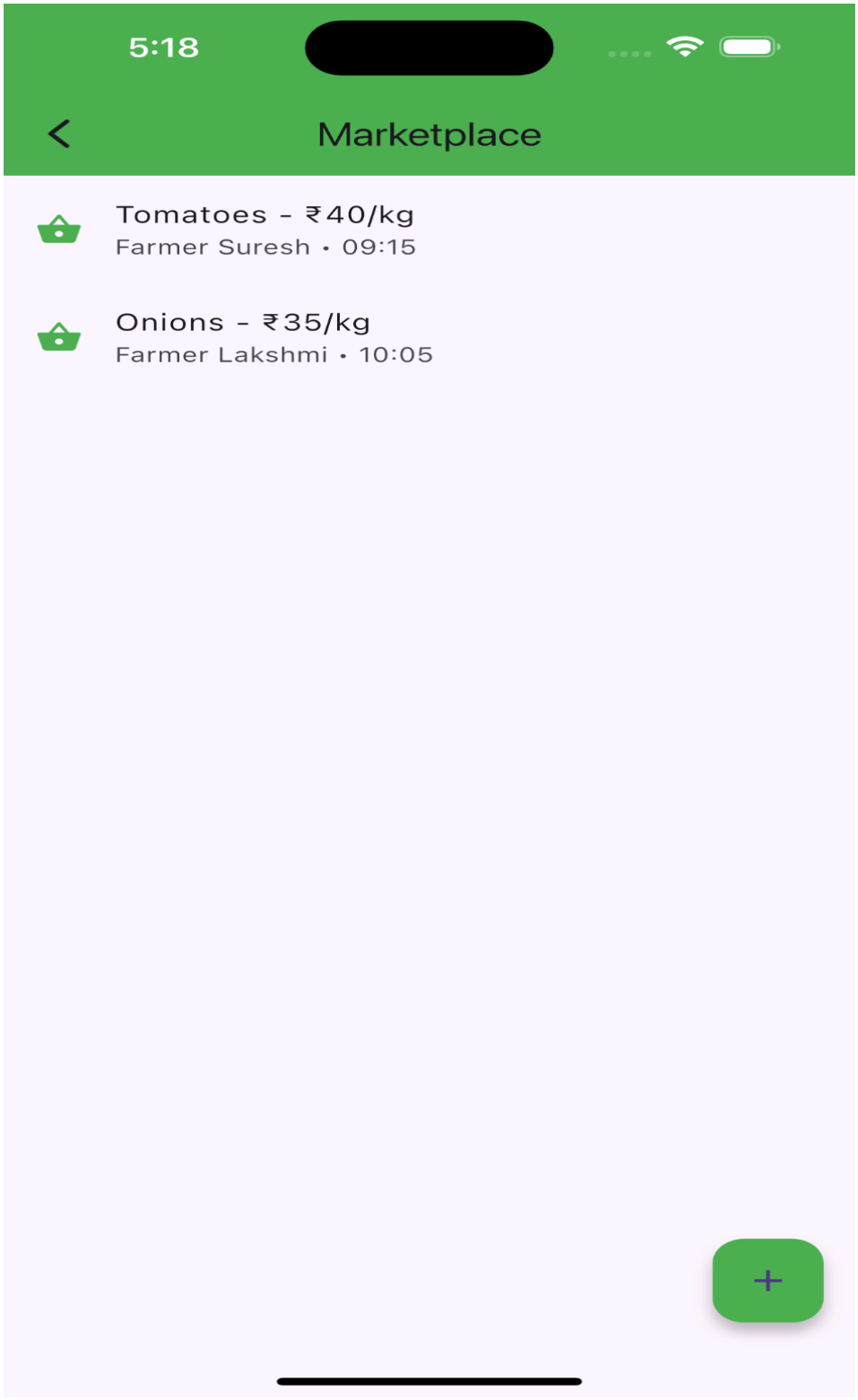
Phone / Email

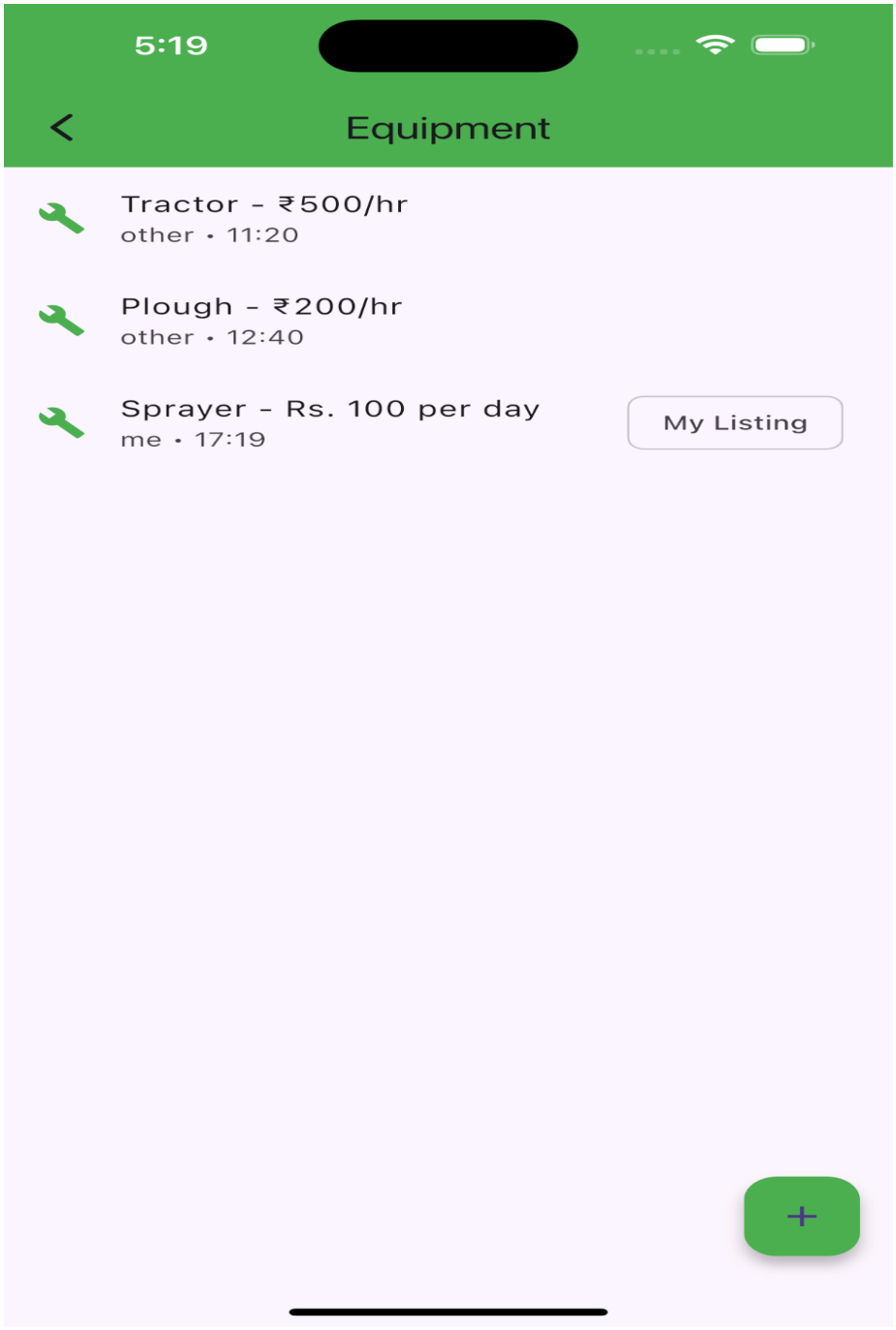
Password / OTP

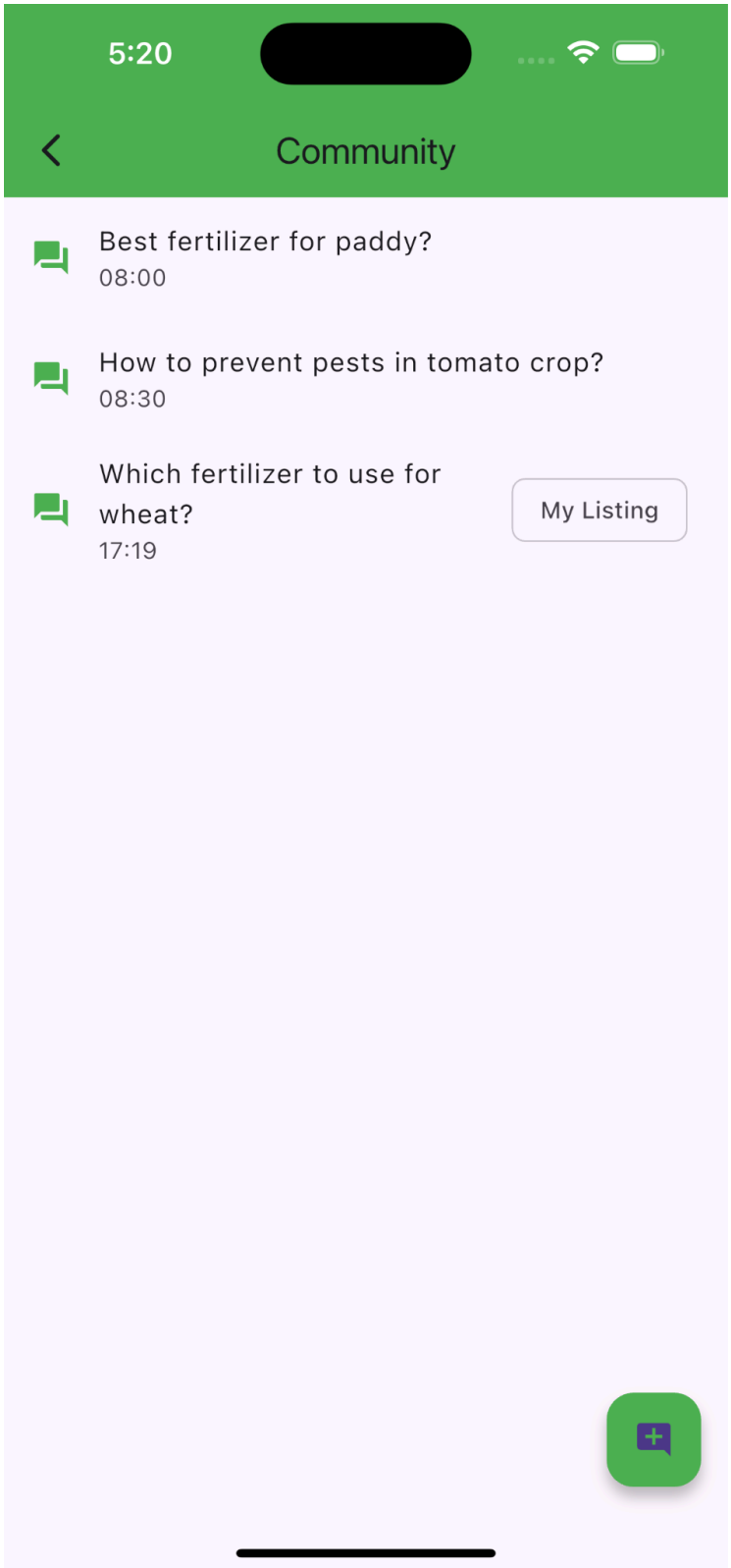
Login

Skip to Dashboard

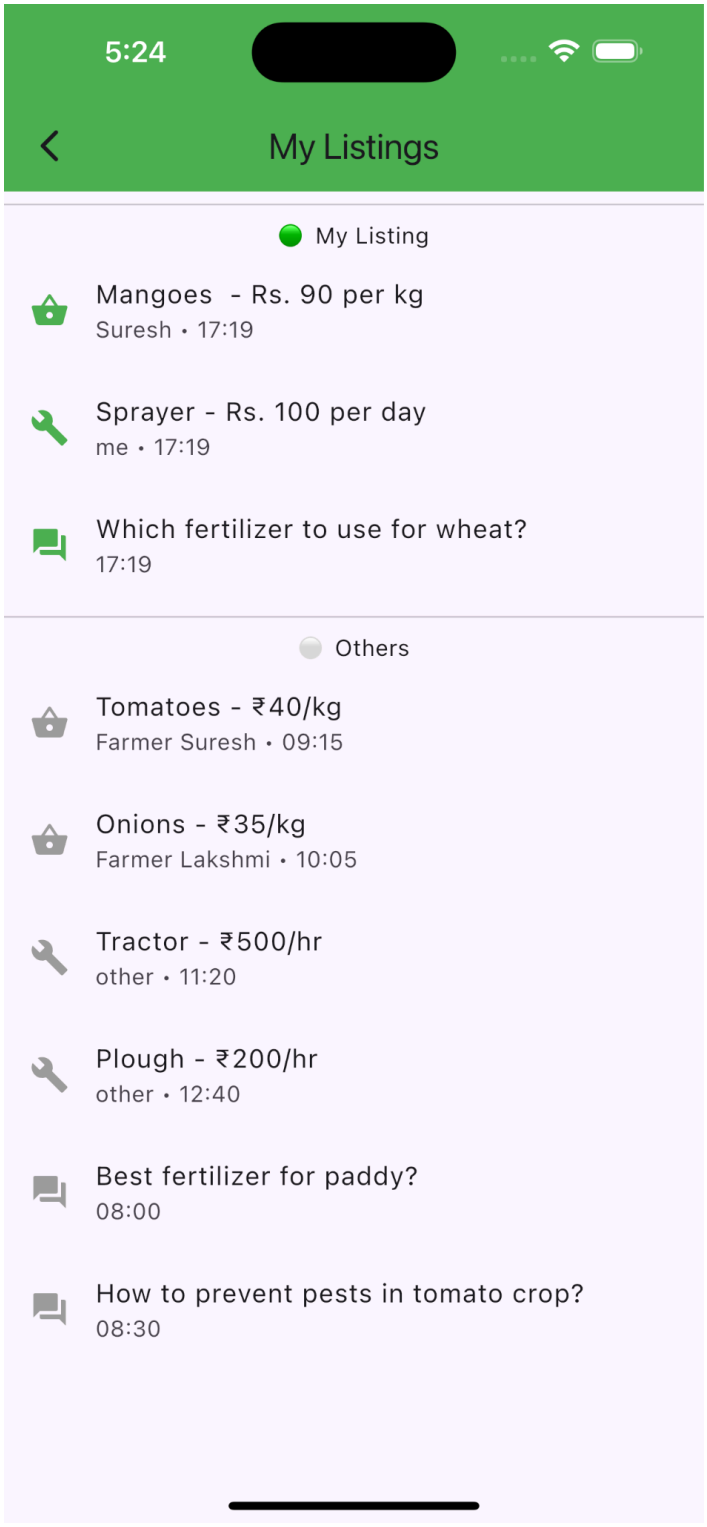

5:18
Welcome 👋
Marketplace
Equipment
Community
Community Chat

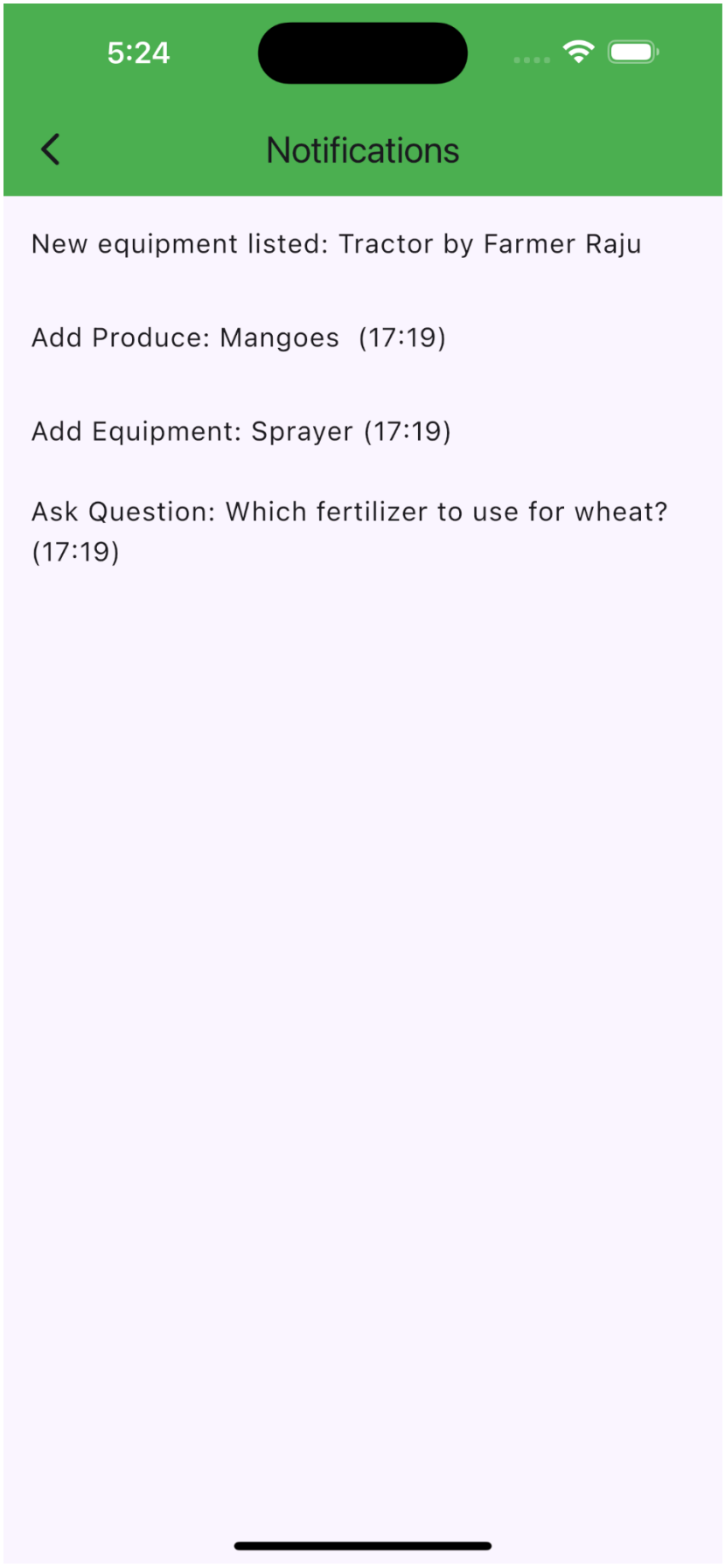

5:18
Marketplace
Tomatoes - ₹40/kg
Farmer Suresh • 09:15
Onions - ₹35/kg
Farmer Lakshmi • 10:05

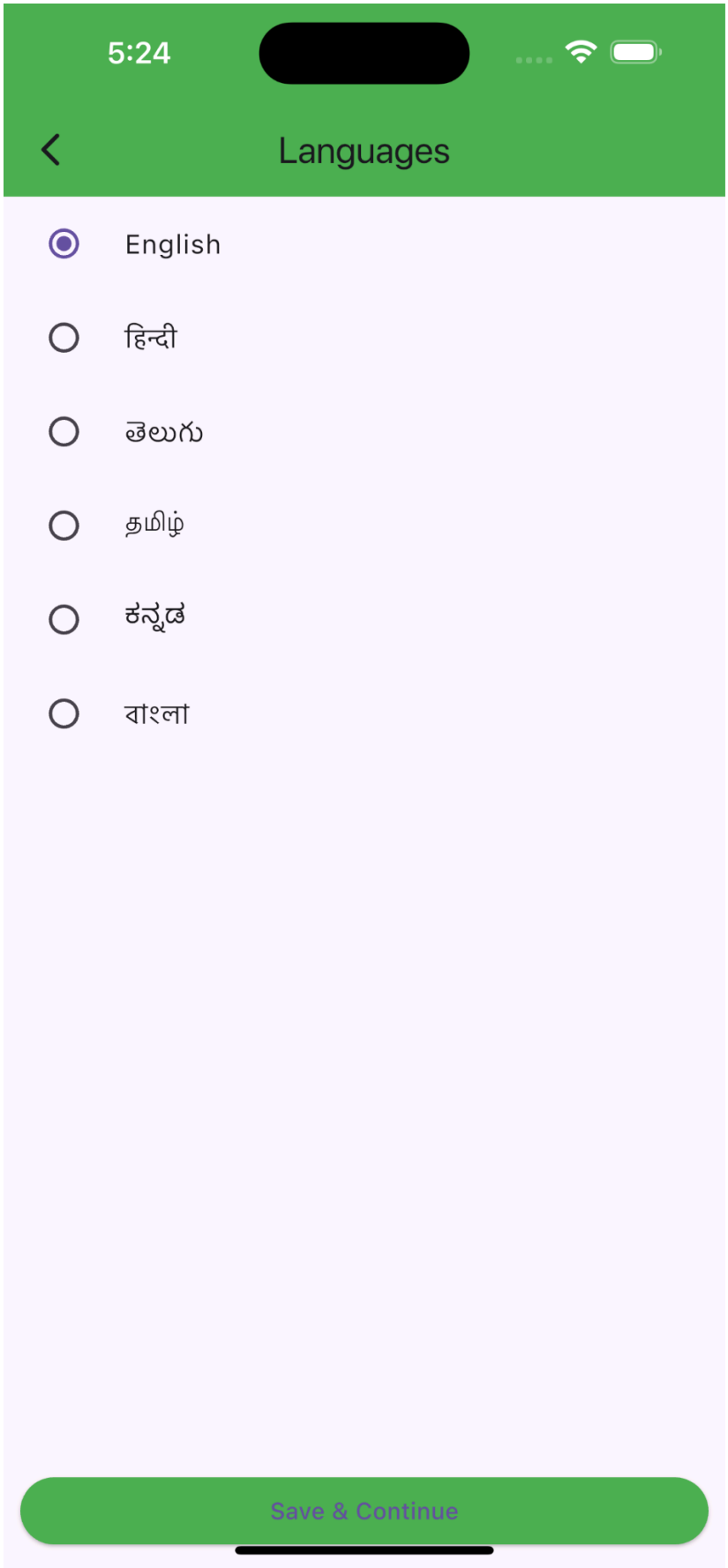

5:19
Equipment
Tractor - ₹500/hr
other • 11:20
Plough - ₹200/hr
other • 12:40
Sprayer - Rs. 100 per day
me • 17:19
My Listing
+


5:20
Community
Best fertilizer for paddy?
08:00
How to prevent pests in tomato crop?
08:30
Which fertilizer to use for wheat?
17:19
My Listing


5:20
Community Chat
Farmer Ramesh: Good morning everyone! (09:00)
Farmer Kavitha: Does anyone have spare seeds? (09:10)
Type a message...


5:24
My Listings
My Listing
Mangoes - Rs. 90 per kg
Suresh • 17:19
Sprayer - Rs. 100 per day
me • 17:19
Which fertilizer to use for wheat?
17:19
Others
Tomatoes - ₹40/kg
Farmer Suresh • 09:15
Onions - ₹35/kg
Farmer Lakshmi • 10:05
Tractor - ₹500/hr
other • 11:20
Plough - ₹200/hr
other • 12:40
Best fertilizer for paddy?
08:00
How to prevent pests in tomato crop?
08:30


5:24
Notifications
New equipment listed: Tractor by Farmer Raju
Add Produce: Mangoes (17:19)
Add Equipment: Sprayer (17:19)
Ask Question: Which fertilizer to use for wheat? (17:19)


5:24
Languages
English
हिन्दी
తెలుగు
தமிழ்
ಕನ್ನಡ
বাংলা
Save & Continue

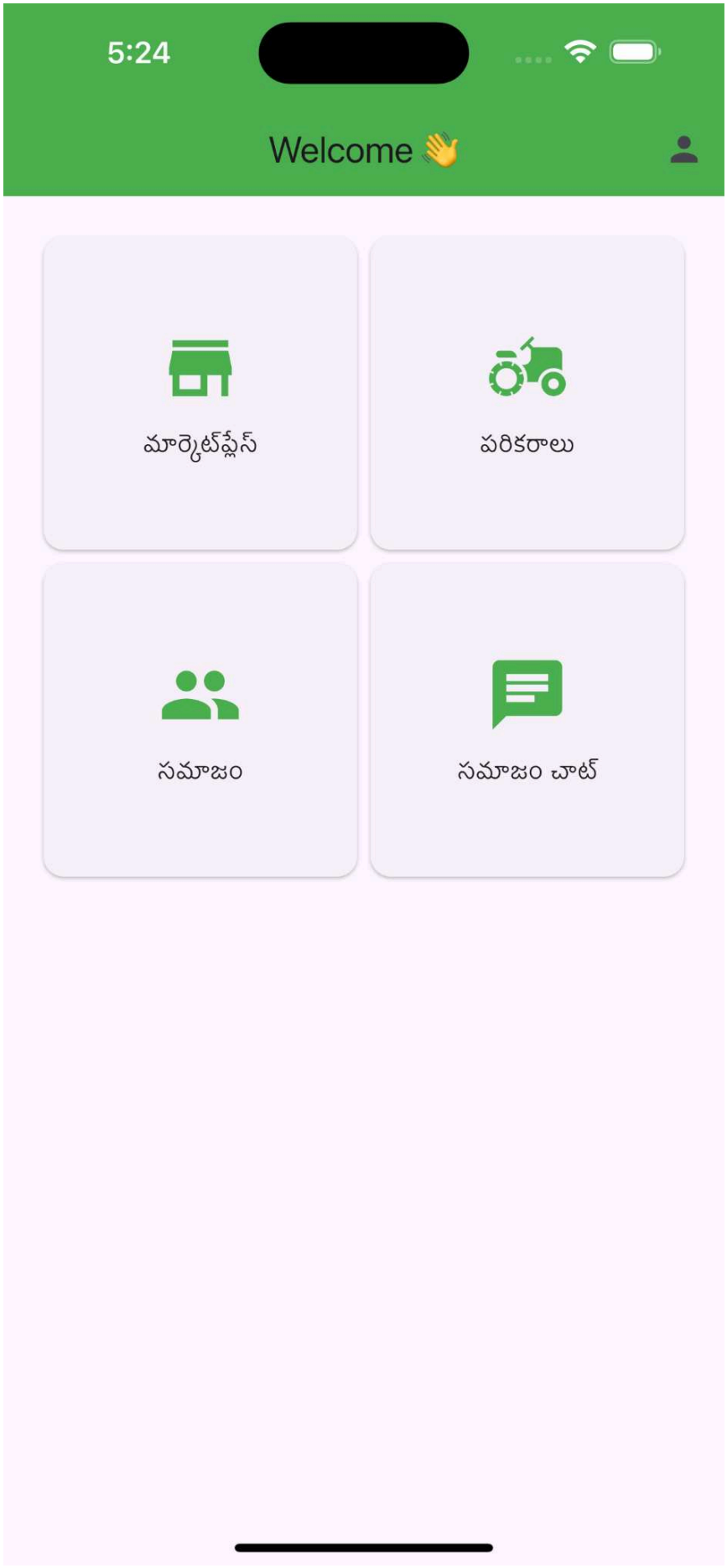
कृषि कनेक्ट हब

फ़ोन / ईमेल

पासवर्ड / ओटीपी

लॉगिन

डैशबोर्ड पर जाएं


5:24
Welcome 👋
మార్కెట్‌ప్లేస్
పరికరాలు
సమాజం
సమాజం చాట్

API DOCUMENTATION (PLANNED)

Endpoint	Method	Description
`/login`	POST	Authenticate user
`/produce`	GET	Fetch produce listings
`/produce/add`	POST	Add produce
`/equipment`	GET	Fetch equipment
`/equipment/add`	POST	Add equipment
`/community`	GET	Fetch Q&A
`/community/add`	POST	Add question
`/chat`	WS	Real-time messages
`/notifications`	GET	Fetch alerts

DEPLOYMENT

Current

- Tested locally using iOS Simulator (iPhone 16 Plus).
- Developed with Flutter SDK on macOS.

Planned

- Firebase backend for real-time chat and listings.
- Docker-based containerization.
- Hosting on AWS/GCP.
- App Store and Play Store release.

CONCLUSION

The **AgriConnect Hub** project demonstrates the end-to-end development of a digital solution for farmers. Beginning with **problem identification**, moving through **UI/UX design in Figma**, and culminating in a working **Flutter prototype**, the project validates the feasibility of a multilingual agricultural app.

Key Achievements:

- Functional modules for **Marketplace, Equipment, Community, Chat, and Profile**.
- Support for **six major Indian languages**.
- Differentiation of **My Listings vs Others' Listings**.
- Implementation of **notifications and interactivity**.

Future Improvements:

- Integrate **Firebase real-time database**.
- Add **secure authentication (OTP/Google login)**.
- Enable **digital payments** for marketplace transactions.
- Deploy backend with **Docker and cloud hosting**.
- Launch on **App Store and Play Store** with push notifications.

Impact:
If scaled, AgriConnect Hub can transform how farmers interact with markets, equipment providers, and communities. It promotes transparency, inclusivity, and collaboration, empowering rural communities with digital tools.