Chatbot for Plant Disease and Pest Information

Adhira Jaijeeth, Sanayya Saima, Anvita V Sajeevan

Computer Science and Technology
Computer Science and Technology
Computer Science and Technology

---***---

Abstract - Chatbots are increasingly being adopted to provide instant query resolution in various domains. This paper discusses the design, implementation, and evaluation of a chatbot application built using Flutter, specifically tailored to address agricultural queries. The chatbot employs a predefined question-answer (FAQ) model and state management through the provider package to deliver a lightweight yet robust user experience. This study also explores the challenges, outcomes, and future scope of such applications in the context of scalability, adaptability, and user engagement.

Key Words: optics, photonics, light, lasers, templates, journals

1.INTRODUCTION

The use of chatbots has proliferated across industries, with applications ranging from customer support to educational tools. Agriculture, a domain where timely and accurate information can significantly impact outcomes, has also begun leveraging chatbot technologies. This paper presents a Flutter-based chatbot designed to address common agricultural questions. The application targets accessibility, simplicity, and scalability while utilizing predefined FAQs to ensure rapid response times.

Objectives:

1. To develop a lightweight chatbot application for agricultural queries.
2. To use Flutter for cross-platform compatibility.
3. To employ provider for state management.
4. To evaluate the effectiveness of the FAQ-based chatbot design.

2. Literature Survey

Chatbot Applications in Agriculture

Sharma and Reddy (2020) highlighted how chatbots bridge knowledge gaps in farming communities. However, they noted that most solutions lack adaptability to dynamic or open-ended queries.

Flutter Framework

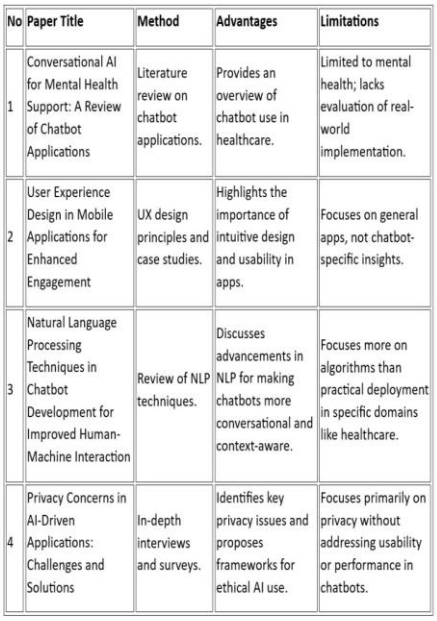
Gupta et al. (2021) emphasized the advantages of Flutter's cross-platform capabilities, particularly in reducing development and maintenance efforts.

State Management with Provider The provider package has been widely recognized for its simplicity and suitability for medium-scale applications (Soni et al., 2020).

Challenges

Studies by Zhou et al. (2021) outlined the limitations of rule-based chatbots, including their inability to handle diverse or unexpected queries without human intervention.

Table



No	Paper Title	Method	Advantages	Limitations
1	Conversational AI for Mental Health Support: A Review of Chatbot Applications	Literature review on chatbot applications.	Provides an overview of chatbot use in healthcare.	Limited to mental health; lacks evaluation of real-world implementation.
2	User Experience Design in Mobile Applications for Enhanced Engagement	UX design principles and case studies.	Highlights the importance of intuitive design and usability in apps.	Focuses on general apps, not chatbot-specific insights.
3	Natural Language Processing Techniques in Chatbot Development for Improved Human-Machine Interaction	Review of NLP techniques.	Discusses advancements in NLP for making chatbots more conversational and context-aware.	Focuses more on algorithms than practical deployment in specific domains like healthcare.
4	Privacy Concerns in AI-Driven Applications: Challenges and Solutions	In-depth interviews and surveys.	Identifies key privacy issues and proposes frameworks for ethical AI use.	Focuses primarily on privacy without addressing usability or performance in chatbots.

3. Methodology

System Design

1.Frontend: Built using Flutter for a responsive and platform-independent UI.
2.Backend: Local predefined FAQs for quick query resolution.
3.State Management: Implemented via provider to manage user authentication and chatbot logic.
4.Routing: Utilized Material App for navigation between login and chatbot screens.

Implementation Workflow

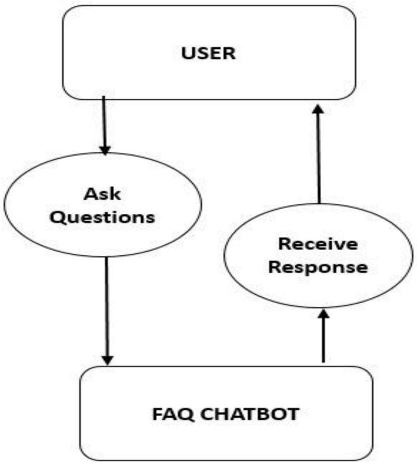
1.User Authentication: Credentials are verified through a local function.

2.Chatbot Interaction:

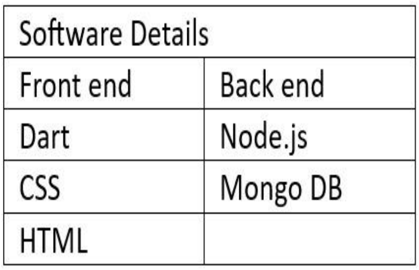
- User inputs are matched against a predefined FAQ map.
- Responses are displayed dynamically.

3. Logout Process: Users are redirected to the login screen, clearing session data.

Architecture



Experimental Details



Software Details	
Front end	Back end
Dart	Node.js
CSS	Mongo DB
HTML	

4. Results and Discussion

Key Outcomes

1.Functional Success: Accurate responses for predefined queries and robust error handling.
2.User Experience: Clean interface and quick response times ensured high usability.
3.Testing: All critical functionalities passed unit and widget tests without issues.

Challenges

1.Static knowledge base limits adaptability to unforeseen queries.
2.Lack of multilingual support restricts accessibility.

Future Enhancements

1.Integrate AI models like OpenAI GPT for dynamic query resolution.
2.Add cloud storage for scalability and analytics.
3.Expand language support to increase inclusivity.

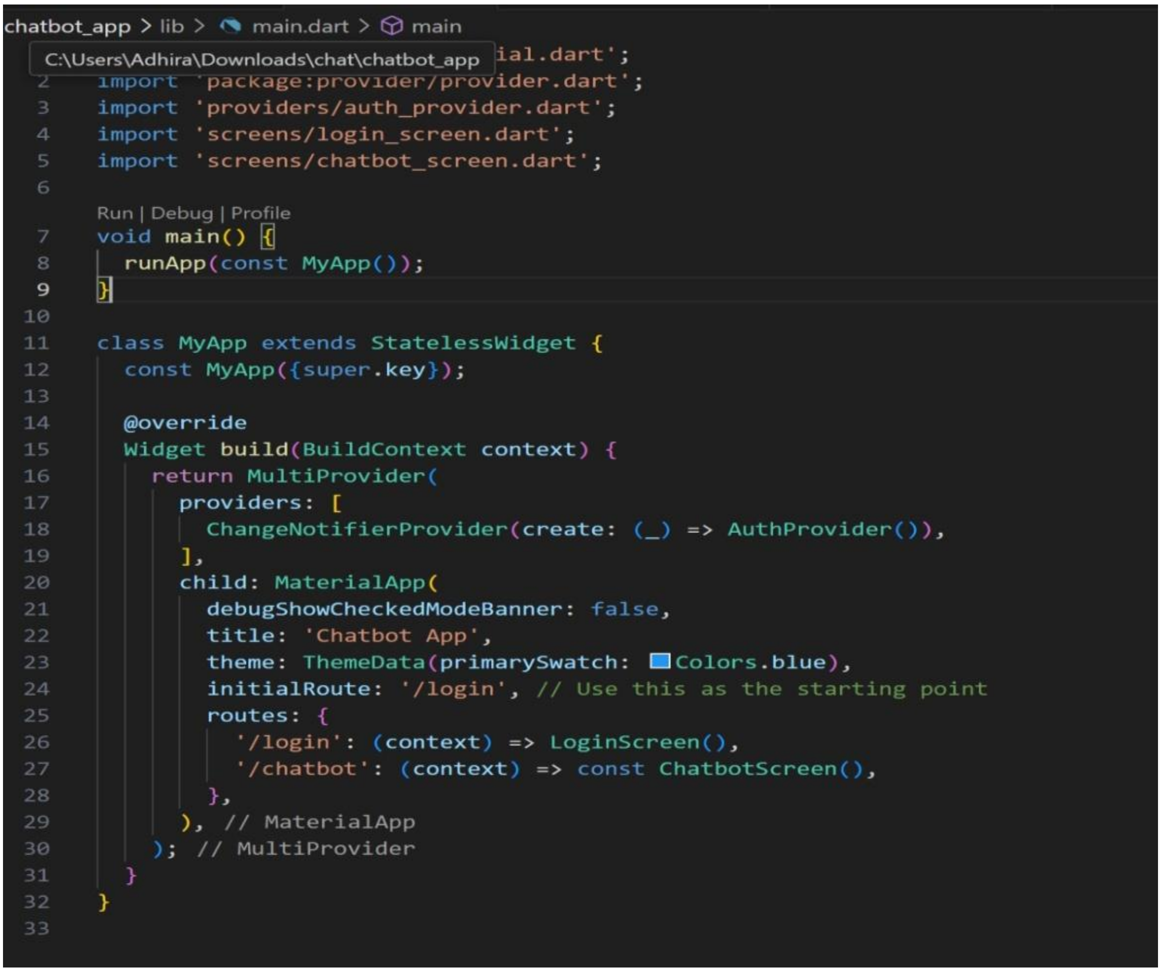
5. Conclusion

This Flutter-based chatbot demonstrates the feasibility of creating a lightweight, responsive application tailored for agricultural queries. While the current implementation effectively addresses predefined questions, future iterations could incorporate dynamic response generation and multilingual support to enhance usability and scalability. This study provides a foundation for further exploration of chatbot applications in agriculture and similar domains.

References

1.Sharma, R., & Reddy, V. (2020). Bridging Agricultural Knowledge Gaps Using Chatbots. Agricultural Informatics Quarterly.
2.Gupta, P., Soni, A., & Mittal, N. (2021). Cross-Platform Mobile Development Using Flutter. Mobile Computing Conference Proceedings.
3.Zhou, L., Gao, J., & Chen, J. (2021). Challenges and Opportunities in Modern Chatbot Design. ACM Computing Surveys.
4.Soni, A., & Patel, K. (2020). Applications of FAQ Chatbots in Education and Beyond. IEEE Transactions on Learning Technologies.

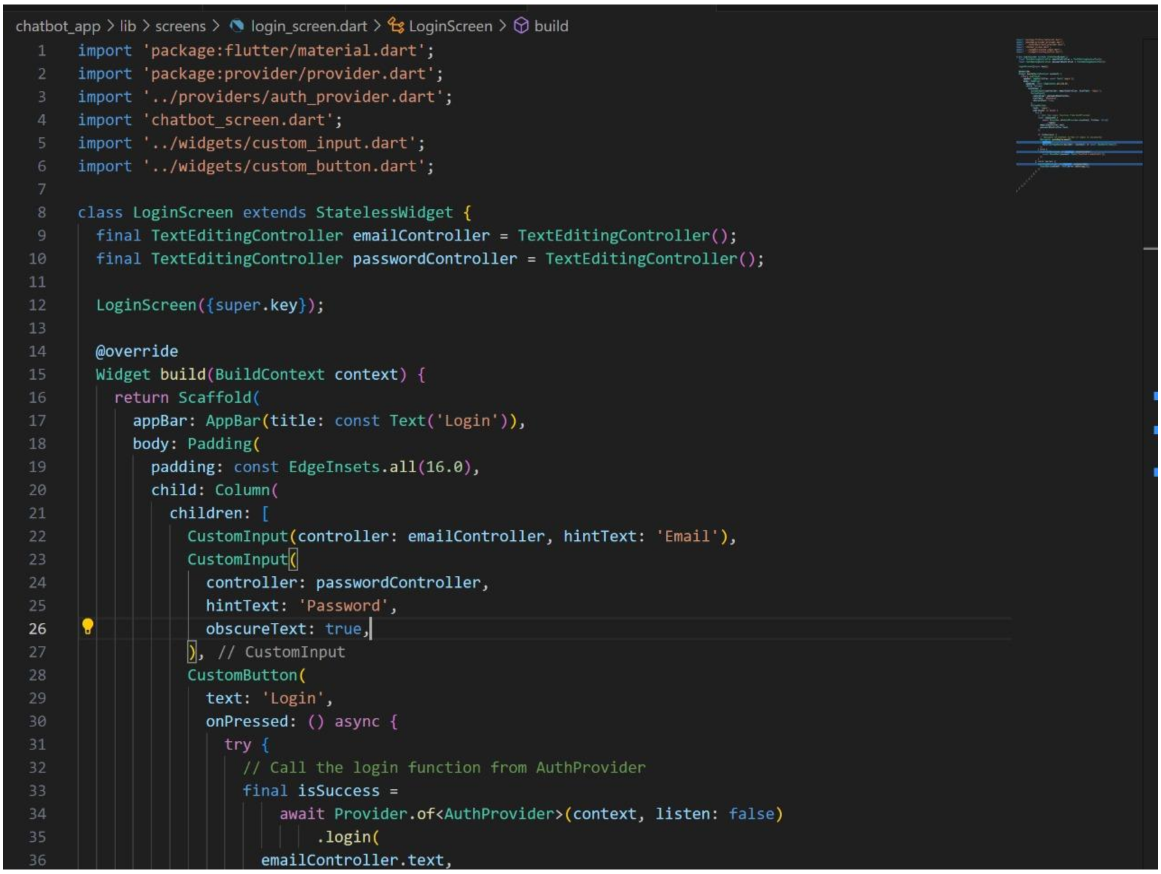
Codes:



chatbot_app > lib > main.dart > main

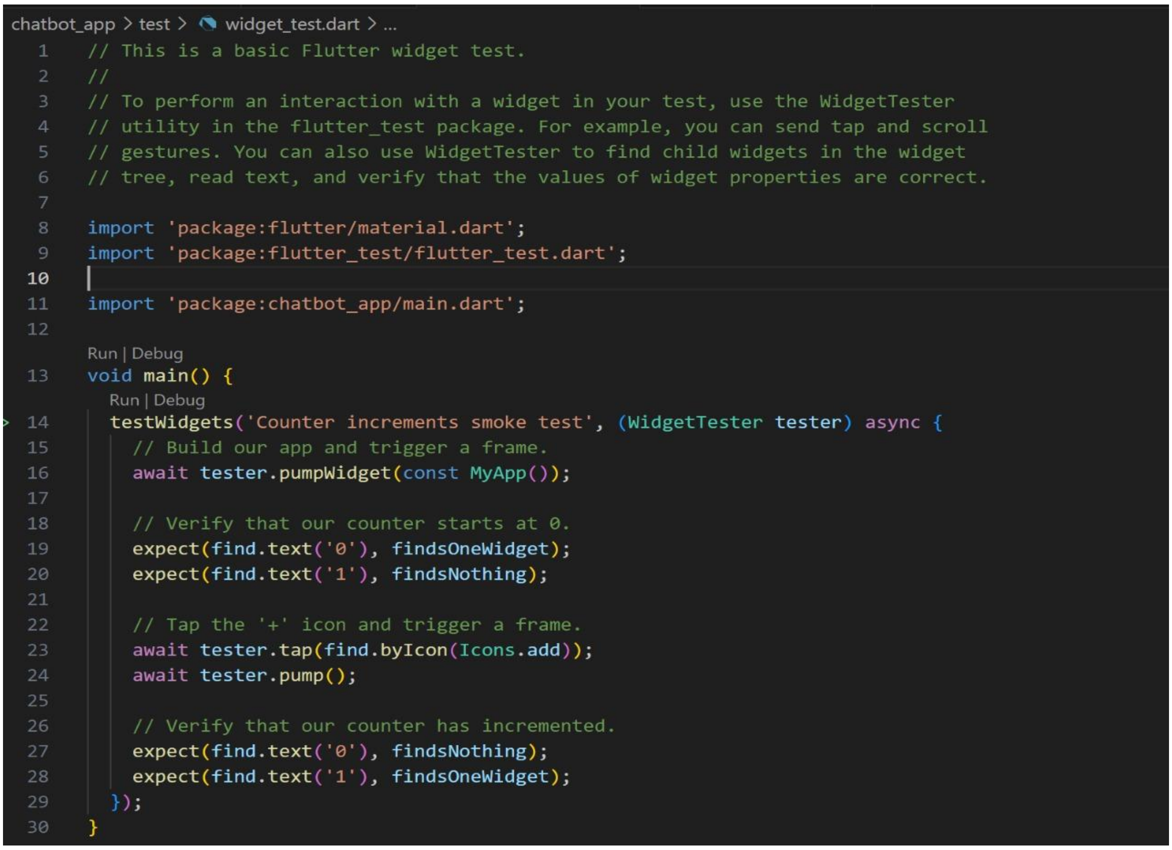
C:\Users\Adhira\Downloads\chat\chatbot_app

```
ial.dart';
import 'package:provider/provider.dart';
import 'providers/auth_provider.dart';
import 'screens/login_screen.dart';
import 'screens/chatbot_screen.dart';

Run | Debug | Profile
void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MultiProvider(
      providers: [
        ChangeNotifierProvider(create: (_) => AuthProvider()),
      ],
      child: MaterialApp(
        debugShowCheckedModeBanner: false,
        title: 'Chatbot App',
        theme: ThemeData(primarySwatch: Colors.blue),
        initialRoute: '/login', // Use this as the starting point
        routes: {
          '/login': (context) => LoginScreen(),
          '/chatbot': (context) => const ChatbotScreen(),
        },
      ), // MaterialApp
    ); // MultiProvider
  }
}
```



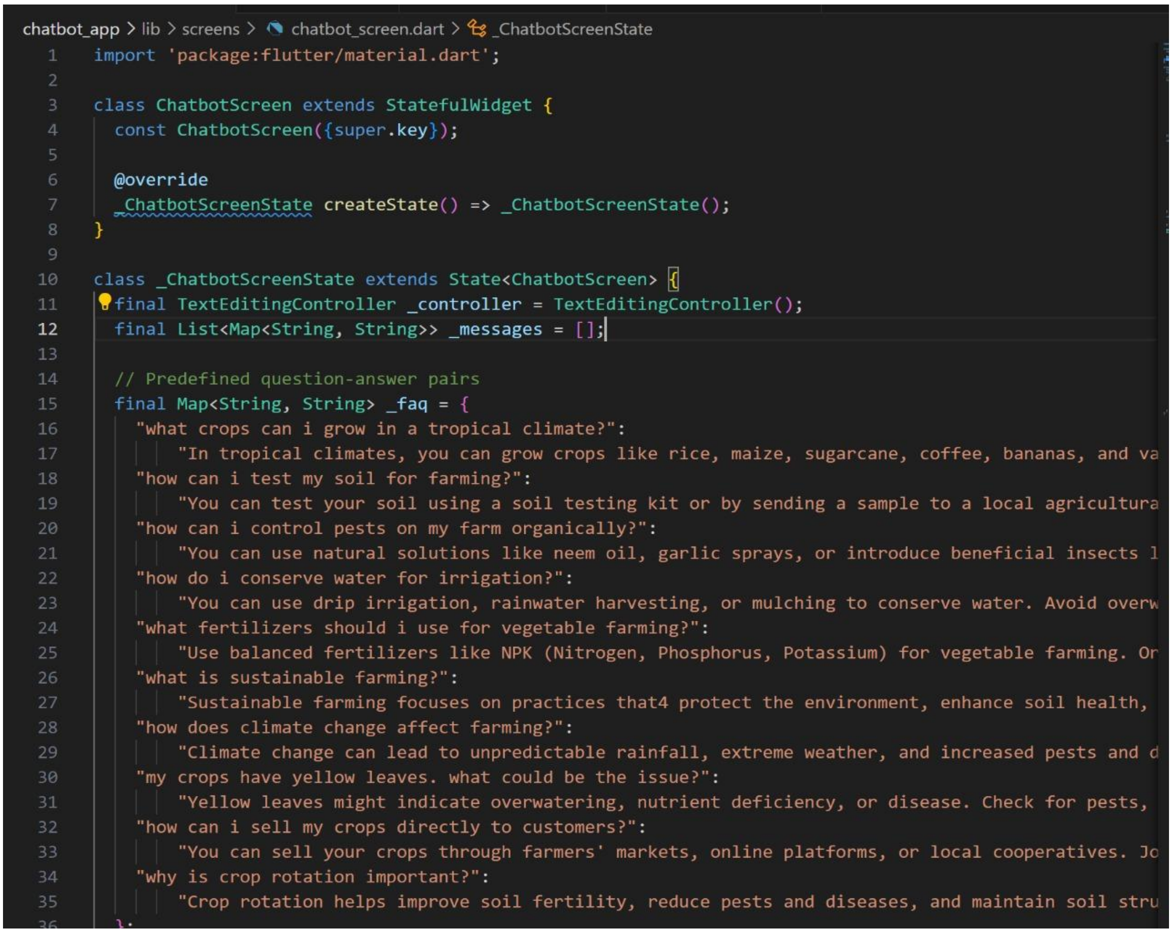
chatbot_app > lib > screens > login_screen.dart > LoginScreen > build

```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import '../providers/auth_provider.dart';
import 'chatbot_screen.dart';
import '../widgets/custom_input.dart';
import '../widgets/custom_button.dart';

class LoginScreen extends StatelessWidget {
  final TextEditingController emailController = TextEditingController();
  final TextEditingController passwordController = TextEditingController();

  LoginScreen({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Login')),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          children: [
            CustomInput(controller: emailController, hintText: 'Email'),
            CustomInput(
              controller: passwordController,
              hintText: 'Password',
              obscureText: true,
            ), // CustomInput
            CustomButton(
              text: 'Login',
              onPressed: () async {
                try {
                  // Call the login function from AuthProvider
                  final isSuccess =
                      await Provider.of<AuthProvider>(context, listen: false)
                          .login(
                    emailController.text,
```



chatbot_app > test > widget_test.dart > ...

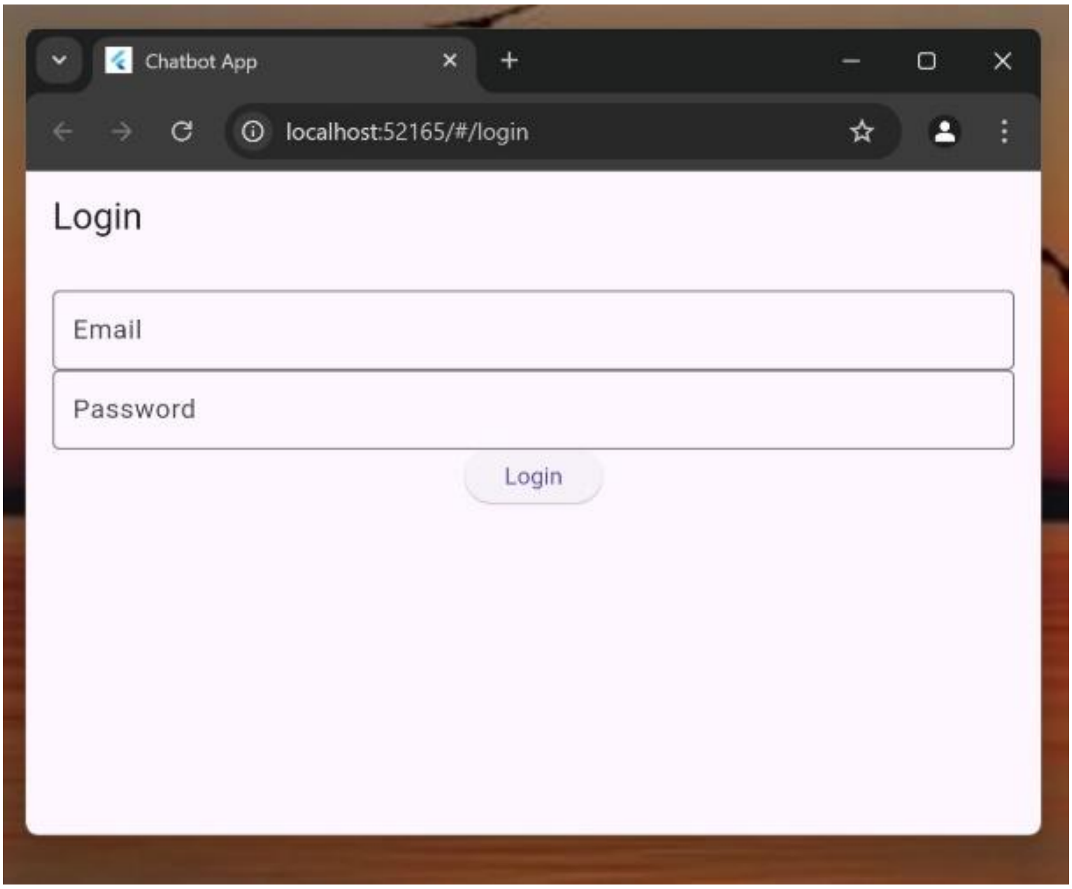
```
// This is a basic Flutter widget test.
//
// To perform an interaction with a widget in your test, use the WidgetTester
// utility in the flutter_test package. For example, you can send tap and scroll
// gestures. You can also use WidgetTester to find child widgets in the widget
// tree, read text, and verify that the values of widget properties are correct.

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

import 'package:chatbot_app/main.dart';

Run | Debug
void main() {
  Run | Debug
  testWidgets('Counter increments smoke test', (WidgetTester tester) async {
    // Build our app and trigger a frame.
    await tester.pumpWidget(const MyApp());

    // Verify that our counter starts at 0.
    expect(find.text('0'), findsOneWidget);
    expect(find.text('1'), findsNothing);

    // Tap the '+' icon and trigger a frame.
    await tester.tap(find.byIcon(Icons.add));
    await tester.pump();

    // Verify that our counter has incremented.
    expect(find.text('0'), findsNothing);
    expect(find.text('1'), findsOneWidget);
  });
}
```

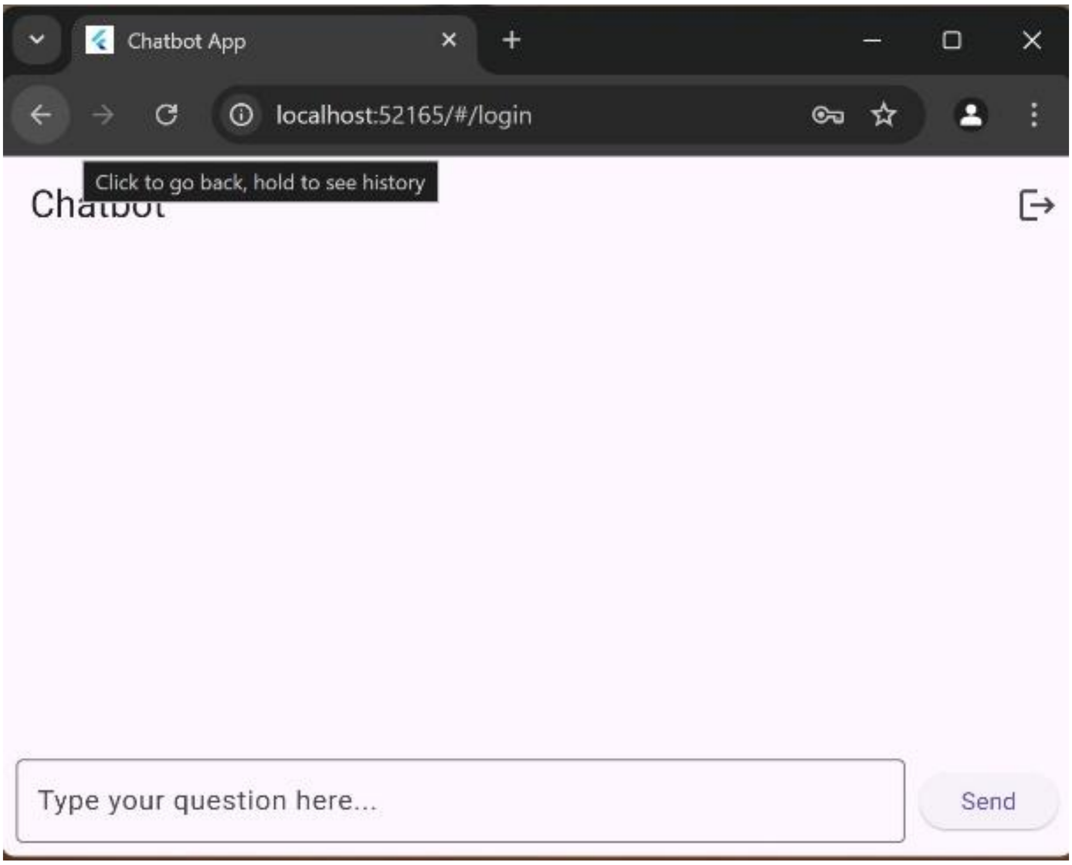


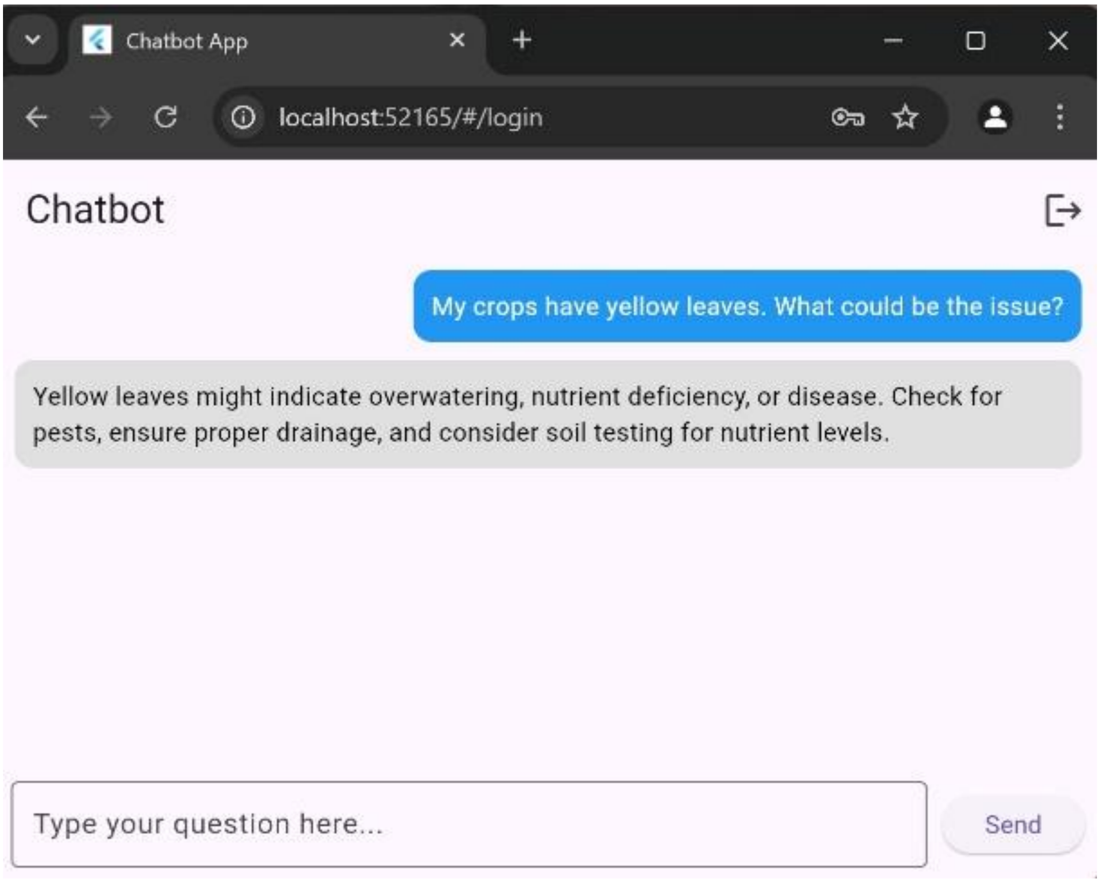
chatbot_app > lib > screens > chatbot_screen.dart > _ChatbotScreenState

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

class ChatbotScreen extends StatefulWidget {
  const ChatbotScreen({super.key});

  @override
  _ChatbotScreenState createState() => _ChatbotScreenState();
}

class _ChatbotScreenState extends State<ChatbotScreen> {
  final TextEditingController _controller = TextEditingController();
  final List<Map<String, String>> _messages = [];

  // Predefined question-answer pairs
  final Map<String, String> _faq = {
    "what crops can i grow in a tropical climate?":
        "In tropical climates, you can grow crops like rice, maize, sugarcane, coffee, bananas, and va
    "how can i test my soil for farming?":
        "You can test your soil using a soil testing kit or by sending a sample to a local agricultura
    "how can i control pests on my farm organically?":
        "You can use natural solutions like neem oil, garlic sprays, or introduce beneficial insects l
    "how do i conserve water for irrigation?":
        "You can use drip irrigation, rainwater harvesting, or mulching to conserve water. Avoid overw
    "what fertilizers should i use for vegetable farming?":
        "Use balanced fertilizers like NPK (Nitrogen, Phosphorus, Potassium) for vegetable farming. Or
    "what is sustainable farming?":
        "Sustainable farming focuses on practices that4 protect the environment, enhance soil health,
    "how does climate change affect farming?":
        "Climate change can lead to unpredictable rainfall, extreme weather, and increased pests and d
    "my crops have yellow leaves. what could be the issue?":
        "Yellow leaves might indicate overwatering, nutrient deficiency, or disease. Check for pests,
    "how can i sell my crops directly to customers?":
        "You can sell your crops through farmers' markets, online platforms, or local cooperatives. Jo
    "why is crop rotation important?":
        "Crop rotation helps improve soil fertility, reduce pests and diseases, and maintain soil stru
  };
```

OUTPUT:


Chatbot App
localhost:52165/#/login
Login
Email
Password
Login


Chatbot App
localhost:52165/#/login
Click to go back, hold to see history
Chatbot
Type your question here...
Send



My crops have yellow leaves. What could be the issue?

Yellow leaves might indicate overwatering, nutrient deficiency, or disease. Check for pests, ensure proper drainage, and consider soil testing for nutrient levels.

Type your question here...