SUMMER INTERNSHIP

on

(NAME OF TECHNOLOGY)

Submitted by

Prerna Anand
12218114
Bachelor of Computer Application

School of Computer Application
Lovely Professional University, Phagwara

(5 June – 15 July 2024)

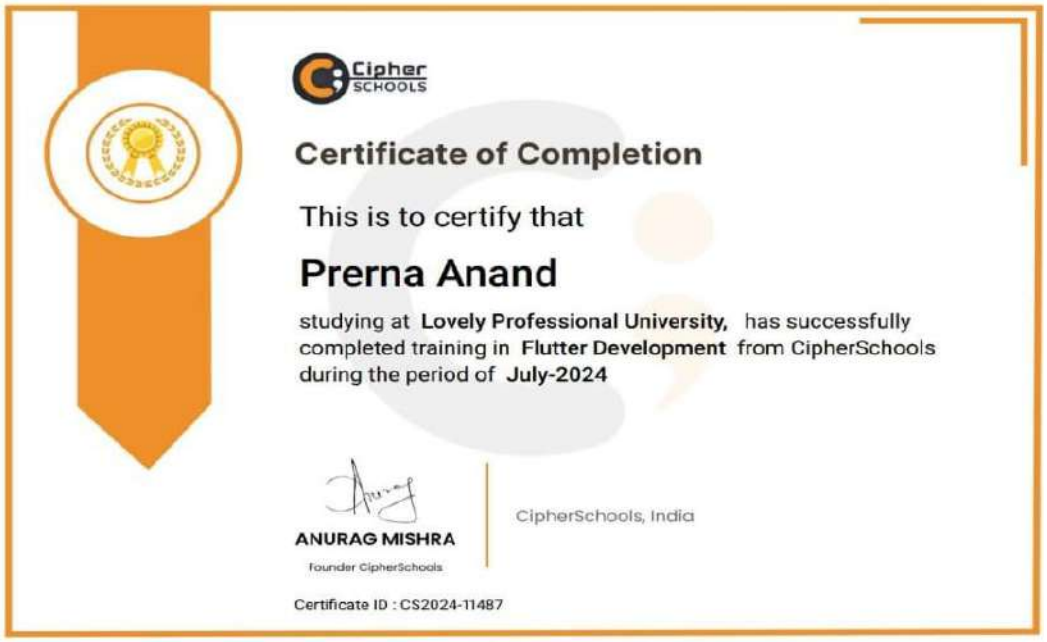
Acknowledgment

The internship opportunity I had with CipherSchool for Flutter Development was a great chance for learning and professional development. Therefore, I consider myself as a very lucky individual as I was provided with an opportunity to be a part of it. I am also grateful for having a chance to learn from professionals who led me through this internship period.

I express my deepest thanks to Training and Placement Coordinator, School of Computer Application, Lovely Professional University for allowing me to grab this opportunity. I choose this moment to acknowledge his contribution gratefully by giving necessary advices and guidance to make my internship a good learning experience.

Prerna Anand
12218114

Internship Certificate



Cipher SCHOOLS

Certificate of Completion

This is to certify that

Prerna Anand

studying at **Lovely Professional University,** has successfully completed training in **Flutter Development** from CipherSchools during the period of **July-2024**

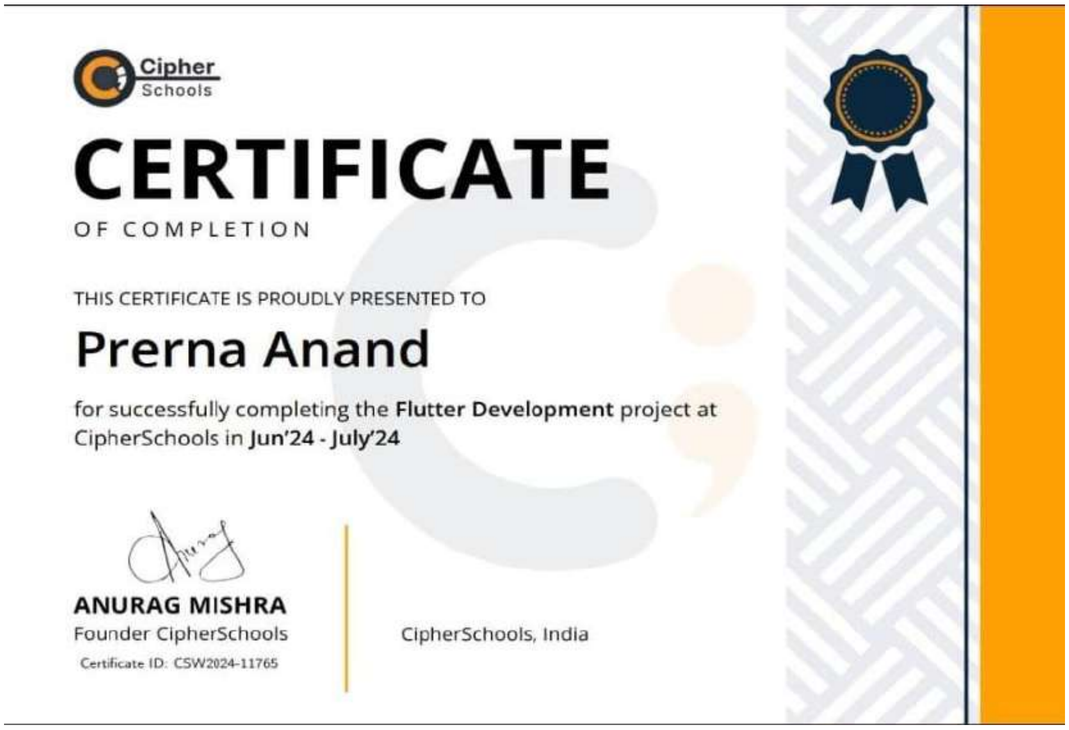
[illegible]

ANURAG MISHRA

Founder CipherSchools

CipherSchools, India

Certificate ID : CS2024-11487



Cipher Schools

CERTIFICATE

OF COMPLETION

THIS CERTIFICATE IS PROUDLY PRESENTED TO

Prerna Anand

for successfully completing the **Flutter Development** project at CipherSchools in **Jun'24 - July'24**

[illegible]

ANURAG MISHRA

Founder CipherSchools

Certificate ID: CSW2024-11765

CipherSchools, India

Table of contents

Sr. No.	Description	Page No.
1	Introduction of the Course	
2	Technical Learnings from the course	
3	Introduction of Mini Project or Technical Assignments	
4	Details of Mini Project	
4.1	Interfaces Designed (If any)	
4.2	Code snippets (If any)	
5	Grade sheet of assignments/ marks card from the MOOC	
6	Bibliography or References	

1. Introduction of the Course

This course was aimed at providing comprehensive knowledge and practical skills in Flutter development. Throughout the course, we delved into the fundamental concepts of Flutter, explored widget creation, and worked on developing interactive and responsive mobile applications. The course was designed to help students gain hands-on experience through a series of technical assignments and projects.

Course Overview:

1. **Objective and Goals:**
    - **Objective:** The primary goal of this course was to impart a deep understanding of Flutter, an open-source UI software development toolkit created by Google. It aimed to enable students to build natively compiled applications for mobile, web, and desktop from a single codebase.
    - **Goals:** Students were expected to grasp fundamental Flutter concepts, learn how to create and manage widgets, and develop interactive, aesthetically pleasing, and responsive applications.
2. **Course Content:**
    - **Introduction to Flutter:**
        - **Overview of Flutter:** An introduction to Flutter, its architecture, and its advantages over other mobile development frameworks.
        - **Installation and Setup:** Instructions on setting up the Flutter environment, including installation of the Flutter SDK, and configuring IDEs such as Android Studio or Visual Studio Code.
    - **Core Concepts:**
        - **Dart Programming Language:** Basic to advanced concepts of Dart, the programming language used with Flutter, including syntax, data structures, and asynchronous programming.
        - **Flutter Widgets:** Exploration of Flutter's rich set of widgets, including Material and Cupertino widgets. Students learned how to use and customize these widgets to build user interfaces.
    - **State Management:**
        - **Managing State:** Techniques for managing state in Flutter applications, including Provider, Riverpod, and other state management solutions.
        - **Stateful vs. Stateless Widgets:** Understanding the difference between stateful and stateless widgets and their use cases.
    - **User Interface Design:**
        - **Layout and Design:** Creating responsive and adaptive layouts using Flutter's layout widgets such as Rows, Columns, Containers, and Flex.
        - **Themes and Styles:** Applying themes and styles to maintain consistency across the app and enhance visual appeal.
    - **Advanced Features:**
        - **Navigation and Routing:** Implementing navigation and routing to handle multiple screens and pass data between them.
        - **Animations:** Adding animations to improve user experience and make the app more engaging.
    - **Backend Integration:**
        - **APIs and Data Handling:** Integrating APIs to fetch and manipulate data, and

        using packages for networking and database operations.
        - **Local Storage:** Techniques for local storage and persistence of data using shared preferences or SQLite.
        - **Firebase Integration:** Learned to set up and integrate Firebase, a powerful backend-as-a-service platform, with their Flutter applications. This included configuring Firebase Authentication for secure login and signup functionalities.

    - **Project Work:**
        - **Technical Assignments:** A series of assignments designed to apply the concepts learned in practical scenarios. These included tasks like building a to-do list app, implementing custom widgets, and managing state.
        - **Mini Projects:** Hands-on projects where students created fully functional applications, demonstrating their ability to integrate and apply various Flutter components.

3. **Learning Outcomes:**

    - **Practical Skills:** Students developed practical skills in building Flutter applications, from setting up the development environment to deploying their apps on different platforms.
    - **Problem-Solving Abilities:** By working on various assignments and projects, students enhanced their problem-solving skills and learned how to tackle real-world development challenges.
    - **Application Development:** The course aimed to ensure that students could independently design, develop, and deploy Flutter applications that are both functional and aesthetically pleasing.
    - **Firebase Integration:** Students gained experience in integrating Firebase with Flutter applications, enhancing their understanding of backend services and user authentication mechanisms.

4. **Teaching Methodology:**

    - **Lectures and Tutorials:** Interactive lectures and tutorials were conducted to explain theoretical concepts and demonstrate practical implementations.
    - **Hands-On Practice:** Emphasis was placed on hands-on practice through coding exercises and real-world projects.
    - **Feedback and Evaluation:** Regular feedback was provided to students to help them improve their skills and understand their progress.

2. Technical Learnings from the Course

Firstly, I am deeply grateful to CipherSchools for providing an excellent platform and resources that facilitated my learning and development. The comprehensive training and insights gained during this summer program were instrumental in advancing my skills.
I would also like to express my sincere appreciation to Shubham Mishra Sir for his mentorship and valuable feedback. His expertise and encouragement were pivotal in refining this project, inspiring me to enhance my understanding and capabilities.
My gratitude extends to Lovely Professional University (LPU) for granting me the opportunity to engage in this summer training program. The supportive environment and resources provided by LPU were crucial in achieving the successful completion of this project.
Throughout the course, I have gained a comprehensive set of technical skills that are essential for Flutter development and beyond. Here’s an in-depth look at the key areas of learning:

Flutter Framework
I developed a thorough understanding of the Flutter framework, which serves as the foundation for building cross-platform mobile applications. This included learning about Flutter's architecture, which relies on a reactive and declarative approach to build user interfaces. I explored key concepts such as Widgets, the widget tree, and the rendering process, which allowed me to create highly interactive and responsive applications.

Dart Programming Language
Proficiency in Dart, the programming language used with Flutter, was a major focus of the course. I enhanced my skills in Dart, including its syntax, data types, functions, and object-oriented principles. This included mastering Dart's asynchronous programming features with Future and Stream, which are essential for handling asynchronous operations in Flutter applications.

State Management
State management is a critical aspect of Flutter development, and I learned various techniques to handle application state effectively. This included:

- **Provider:** A simple and effective way to manage state and dependencies in Flutter applications.
- **setState:** A basic method for managing local state within a widget, which is useful for simpler applications.

Each state management technique was explored in depth, providing me with the tools to choose the most appropriate solution based on application requirements.

User Interface Design
The course covered essential UI/UX design principles necessary for creating engaging and user-friendly interfaces. I learned how to use Flutter's rich set of widgets to build custom components, apply themes, and design layouts that are both functional and aesthetically pleasing. This involved understanding layout widgets like Column, Row, and Stack, as well as advanced concepts such as animations and responsive design.

API Integration
Integrating RESTful APIs was a significant part of the learning process. I acquired skills in making network requests, parsing JSON data, and handling responses to display dynamic and real-time content within Flutter applications. This knowledge is crucial for building applications that interact with web services and external data sources.

 API Keys Management: I learned how to generate and securely manage API keys, which are essential for authenticating and authorizing API requests. This involved setting up API keys for various services and ensuring they are used correctly in the application.
 Mini Project - Map API Integration: As part of a mini project, I integrated a map API from pub.dev. This project involved using a map service API to display interactive maps within the app, adding markers, and handling user interactions. The integration demonstrated how to work with third-party APIs and utilize their functionalities effectively in a Flutter application.

Debugging and Testing
Improving debugging and testing practices was another key learning outcome. I became adept at using Flutter DevTools to diagnose and resolve issues, monitor performance, and ensure the stability of applications. Additionally, I learned about unit testing, widget testing, and integration testing to validate the functionality and reliability of my code.

Firebase Integration
A noteworthy aspect of the course was learning how to integrate Firebase into Flutter applications. Firebase provides a suite of tools and services that enhance app functionality and backend operations. I gained experience in:

- **Firebase Authentication:** Implementing user authentication using Firebase, including features like email/password sign-up, social login, and user management.

Overall, the course provided me with a solid foundation in Flutter development, equipping me with the skills to create sophisticated, high-performance mobile applications.

3. Introduction of Mini Project or Technical Assignments

The mini project and technical assignments were pivotal components of the Flutter development course, serving as key elements for translating theoretical knowledge into practical skills. These components were meticulously designed to offer students real-world experience in mobile application development, thereby enhancing their proficiency in creating functional and interactive applications.

Mini Project Overview

The mini project was centered around the development of a To-Do List application using Flutter, a popular framework for building cross-platform mobile applications. This project was strategically selected to provide a comprehensive learning experience by incorporating various aspects of Flutter development into a single cohesive application.

Project Purpose: The To-Do List application was chosen because it represents a common type of mobile app that includes essential features used across many applications. By developing this project, students were able to:

- **Consolidate Learning:** Integrate multiple concepts learned during the course, such as widget usage, state management, and navigation.
- **Practical Application:** Experience the development lifecycle of a complete application, from initial design to final deployment.
- **Skill Development:** Apply theoretical knowledge to solve practical challenges and build a fully functional app.

Project Objectives:

1. **Understand Flutter Basics:**
    - Gain a solid grasp of Flutter’s foundational concepts, including widget creation, state management, and navigation principles.
    - Apply these concepts to create a working To-Do List application that reflects core functionalities.
2. **Implement Core Features:**
    - **Task Creation:** Allow users to add new tasks to their list.
    - **Task Editing:** Enable users to modify existing tasks.
    - **Task Deletion:** Provide functionality to remove tasks from the list.
    - **Task Completion:** Implement a mechanism for users to mark tasks as complete or incomplete.
3. **Design User Interface:**
    - Focus on crafting a user-friendly and intuitive interface that enhances user experience.
    - Employ design principles to ensure the application is visually appealing and easy to navigate.
4. **Integrate Firebase:**
    - Set up Firebase to manage user authentication, including functionalities for user login and signup.
    - Ensure secure and efficient management of user data through Firebase’s cloud-based services.
5. **Enhance Practical Skills:**
    - Develop problem-solving skills by addressing challenges encountered during development.
    - Implement best practices in app design, coding, and testing to create a robust application.

Technical Assignments Overview
In addition to the mini project, the course incorporated various technical assignments, each focusing on specific aspects of Flutter development. These assignments were designed to provide targeted practice and deepen students' understanding of different facets of mobile app development.

Assignment Highlights:

1. **Widget Creation and Layout:**
    - Design and implement various Flutter widgets to create visually appealing and functional user interfaces.
    - Explore layout options and customization to build responsive and adaptive layouts that work across different screen sizes.
2. **State Management:**
    - Engage in assignments that focused on different state management techniques, such as:
        - **Provider:** Learn to manage application state and dependencies in a straightforward manner.
        - **Riverpod:** Explore an advanced state management solution that offers enhanced flexibility and safety.
        - **BLoC (Business Logic Component):** Understand how to separate business logic from UI code for better maintainability.
3. **API Integration:**
    - Integrate external APIs into Flutter applications to fetch and display data dynamically.
    - Work with API keys and handle responses to enable real-time data updates and interactions.
4. **Database Integration:**
    - Set up local databases using SQLite or integrate cloud-based databases like Firebase Firestore.
    - Manage app data persistently and perform operations such as data retrieval, storage, and synchronization.
5. **User Authentication:**
    - Implement user authentication features, including:
        - **Login and Registration:** Enable users to create accounts and log in securely.
        - **Password Recovery:** Provide functionality for users to recover or reset their passwords.

Learning Outcomes:

- **Comprehensive Knowledge:**
    - Develop a thorough understanding of Flutter's core concepts, including widget lifecycle, asynchronous programming, and responsive design.
- **Practical Experience:**
    - Gain hands-on experience by applying theoretical knowledge to real-world projects, enhancing skills in app development.
- **Problem-Solving Skills:**
    - Improve critical thinking and problem-solving abilities by tackling various development challenges.
- **Professional Development:**
    - Build a portfolio with practical examples of work, demonstrating proficiency to potential employers and preparing for future development roles.

The mini project and technical assignments provided a structured yet flexible framework for exploring Flutter development. They offered valuable opportunities to apply learning in a practical context, fostering both technical proficiency and creative problem-solving abilities.

4. Details of Mini Project (To-Do List App)

The To-Do List app is a practical application designed to help users manage their daily tasks efficiently. It offers a range of features to simplify task management and enhance productivity. Below is a detailed overview of the app's functionality and the interfaces designed:

4.1 Interfaces Designed

1. **Sign-Up Page:**
    - **Purpose:** The Sign-Up Page facilitates user registration by allowing new users to create an account.
    - **Features:**
        - **User-Friendly Form:** The page includes a form where users can enter their username, email, password, and confirm their password.
        - **Validation:** Each input field has validation rules to ensure data integrity:
            - **Username:** Required field, must not be empty.
            - **Email:** Required field, must be in a valid email format.
            - **Password:** Required field, must be at least 8 characters long, contain both uppercase and lowercase letters, and include at least one number.
            - **Confirm Password:** Must match the password entered.
        - **Password Visibility:** Options to toggle password visibility for both password and confirm password fields to enhance user convenience.
        - **Loading Indicator:** A progress indicator is shown while the registration process is ongoing.
        - **Navigation:** Includes a link to the Login Page for users who already have an account.

2. **Login Page:**
    - **Purpose:** The Login Page allows existing users to access their accounts by entering their credentials.
    - **Features:**
        - **Login Form:** Users can enter their email and password to log in.
        - **Validation:** Checks for empty fields and ensures that the credentials match those stored in the database.
        - **Password Visibility:** Option to toggle password visibility to ensure users can easily view their input.
        - **Forgot Password Link:** A link to recover a forgotten password, guiding users through the password recovery process.
        - **Navigation:** Provides a link to the Sign-Up Page for new users to create an account.

3. **Splash Screen:**
    - **Purpose:** The Splash Screen is the initial screen displayed when the app is launched, providing a visual introduction to the app.
    - **Features:**
        - **Branding:** Displays the app's logo or name to create a professional and engaging first impression.
        - **Animation:** Includes a simple animation or transition effect to make the loading experience more dynamic and visually appealing.
        - **Duration:** Remains visible for a few seconds while the app initializes and loads necessary resources.

4. **Home Page:**
    - **Purpose:** The Home Page serves as the main interface where users can manage their to-do items.
    - **Features:**
        - **Task List:** Displays a list of all tasks, with options to view, add, edit, or delete items.
        - **Add Task Button:** A button to add new tasks, with a form to input task details.
        - **Edit Task:** Allows users to modify existing tasks, including changing the task title, description, and due date.
        - **Delete Task:** Option to remove tasks from the list.
        - **Task Completion:** Users can mark tasks as complete or incomplete, with visual indicators to differentiate between completed and pending tasks.
        - **Sorting and Filtering:** Options to sort tasks by date, priority, or category, and to filter tasks based on their completion status or due date.
        - **Task Completion Tracking**: The interface allows users to mark tasks as complete or incomplete. Completed tasks may be moved to a separate section or visually distinguished from pending tasks, helping users see their progress.
5. **Task Management:**
    - **Purpose:** Provides an in-depth interface for managing individual tasks.
    - **Features:**
        - **Task Details:** Displays detailed information about each task, including title, description, due date, and priority.
        - **Completion Status:** Users can toggle the completion status of a task with a checkbox or switch.
        - **Priority Levels:** Allows users to categorize tasks based on priority levels (e.g., high, medium, low), with visual indicators for each priority.
        - **Edit and Delete Options:** Users can edit or delete tasks from this detailed view.

6. **Splash Screen:**

- **Purpose:** The Splash Screen serves as the initial screen displayed when the app is launched. It provides users with a visually engaging introduction to the app and helps in setting the tone for the user experience.
- **Features:**
    - **Branding:** The screen prominently features the app's logo or name, establishing brand identity and creating a cohesive first impression.
    - **Animation and Transition:** Includes a simple animation or transition effect, such as a fade-in or zoom effect, to make the loading process more dynamic and engaging. This helps keep users entertained while the app initializes.
    - **Duration:** The Splash Screen is displayed for a brief period (typically 2-3 seconds) to allow time for the app to load and prepare the main interface. It ensures a smooth transition to the Home Page or Login Page.
    - **Background Design:** The background may feature a gradient or image that complements the app's theme, enhancing visual appeal without distracting from the primary branding elements.
    - **Loading Indicators:** Optional progress indicators or animations can be included to signal to users that the app is loading, providing a seamless user experience during startup.

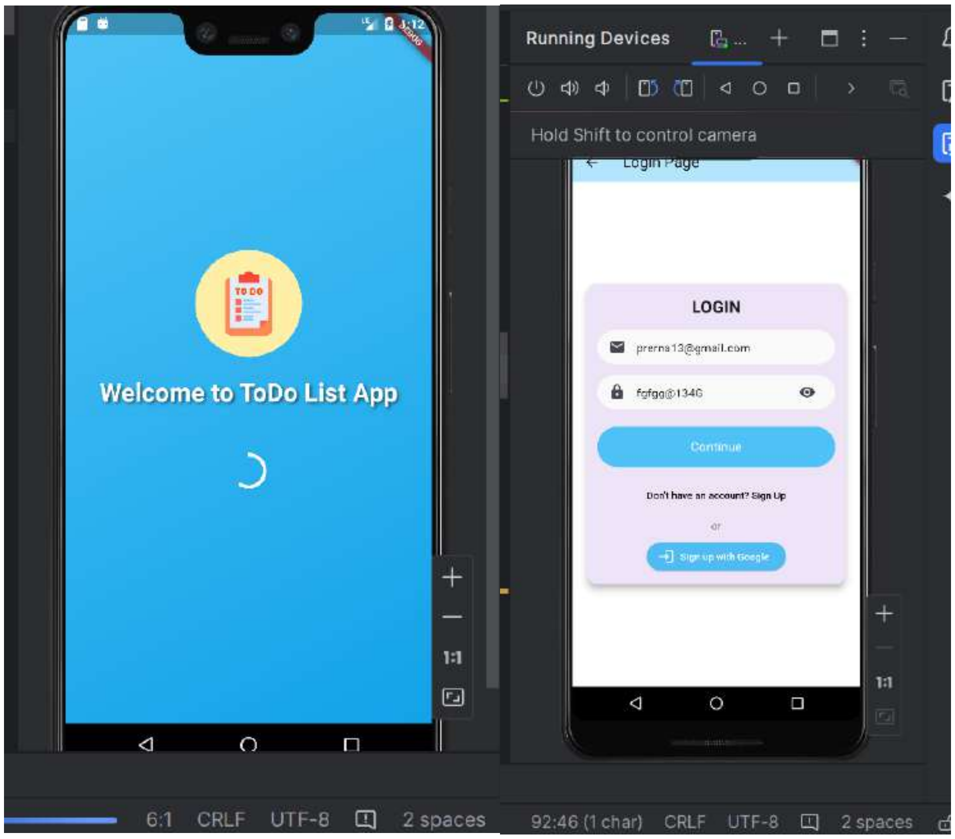
7. **Login Page:**

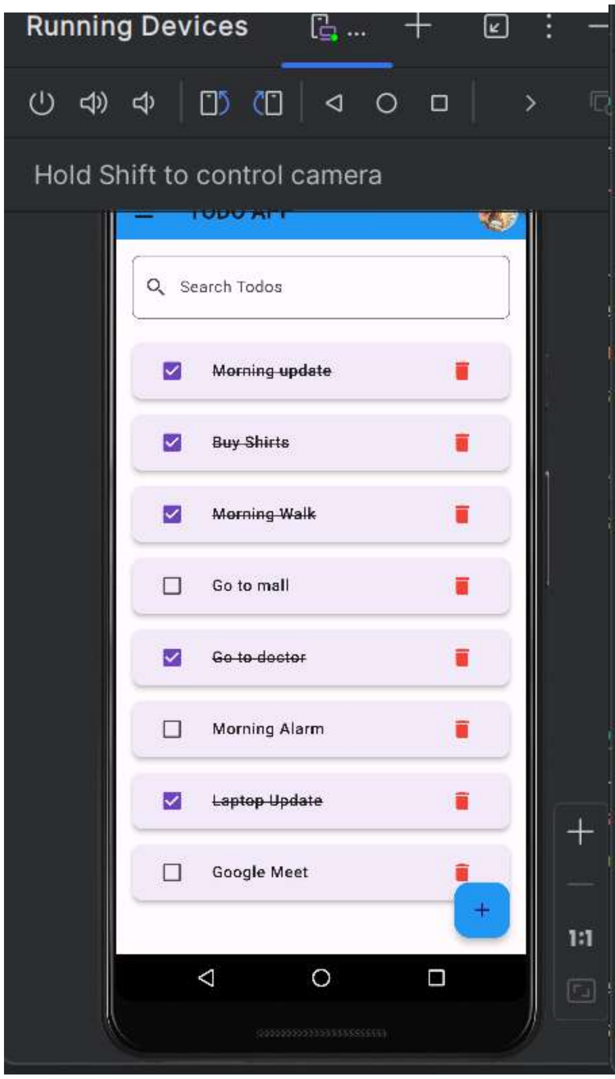
- **Purpose:** The Login Page is designed for users to access their existing accounts by entering their credentials. It ensures secure authentication and provides a gateway to the main features of the app.
- **Features:**
    - **Login Form:** A clear and straightforward form where users can enter their email and password to log in.

        - **Email Field:** Requires a valid email address to identify the user. Includes validation to ensure the email format is correct.
        - **Password Field:** Requires the user’s password for authentication. Includes an option to toggle password visibility to help users enter their credentials accurately.
    - **Validation:**
        - **Email:** Checks that the email field is not empty and follows a valid email format.
        - **Password:** Ensures the password field is not empty and matches the password stored in the database.
        - **Error Messages:** Displays informative error messages for incorrect credentials or validation failures to guide users in correcting their inputs.
    - **Forgot Password Link:** Provides a link for users who have forgotten their password, leading them to a password recovery process to regain access to their accounts.
    - **Sign-Up Link:** Includes a link to the Sign-Up Page for new users, allowing them to create an account if they don’t already have one.
    - **Login Button:** A prominent button that submits the login form. It is typically styled to stand out and encourage user interaction.
    - **Loading Indicator:** Displays a progress indicator while the login process is ongoing, ensuring users are aware that their request is being processed.
    - **Navigation:** Includes navigation options to transition seamlessly to other parts of the app, such as the Sign-Up Page or Home Page, based on user actions.

Overall, the To-Do List app emphasizes user experience with a clean, intuitive interface that makes task management straightforward and efficient. Each interface is designed to be user-friendly, ensuring that users can easily navigate through the app and perform tasks with minimal effort.

4.2 Code Snippets:




Running Devices

Hold Shift to control camera

TODO APP

Search Todos

~~Morning update~~

~~Buy Shirts~~

~~Morning Walk~~

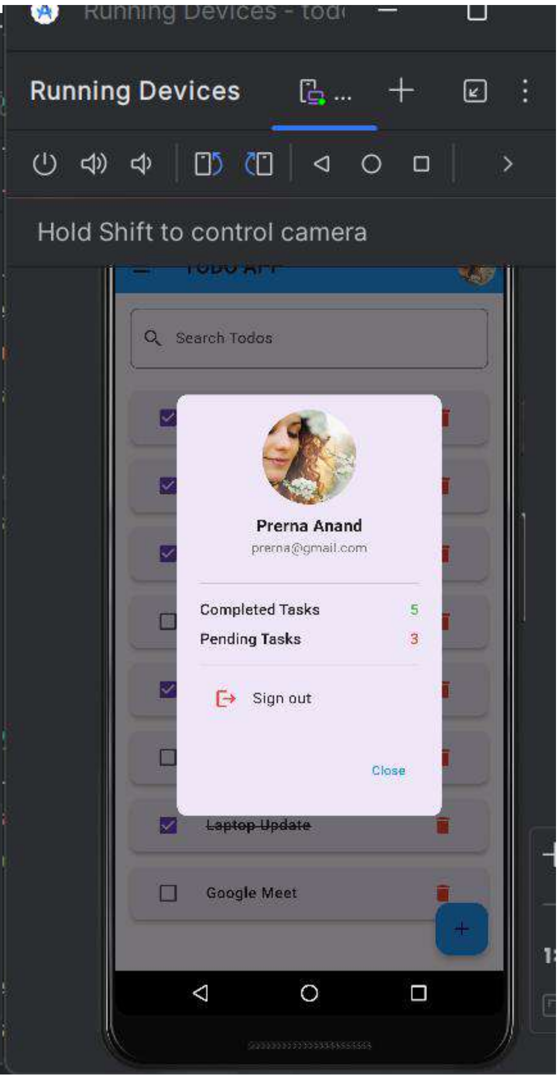
Go to mall

~~Go to doctor~~

Morning Alarm

~~Laptop Update~~

Google Meet


Running Devices - todo

Running Devices

Hold Shift to control camera

TODO APP

Search Todos

Prerna Anand
prerna@gmail.com

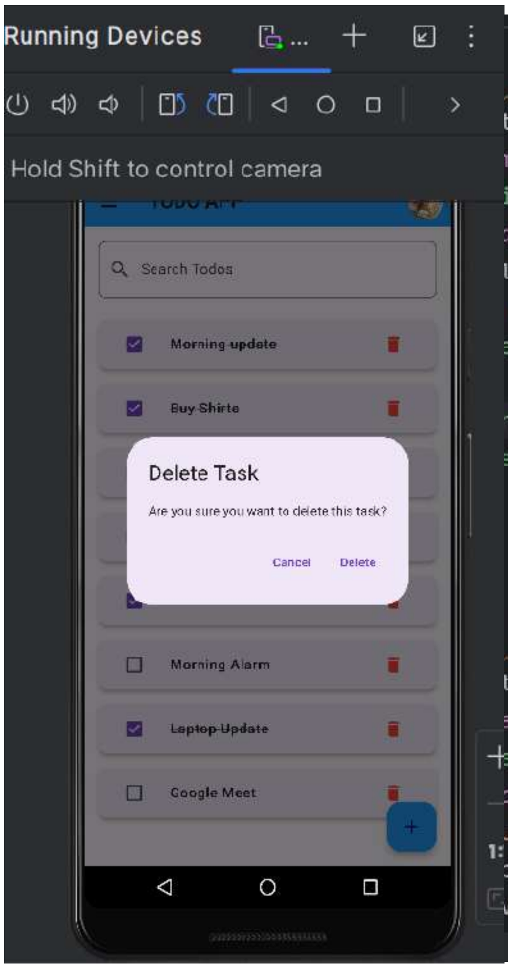
Completed Tasks	5
Pending Tasks	3

Sign out

Close

~~Laptop Update~~

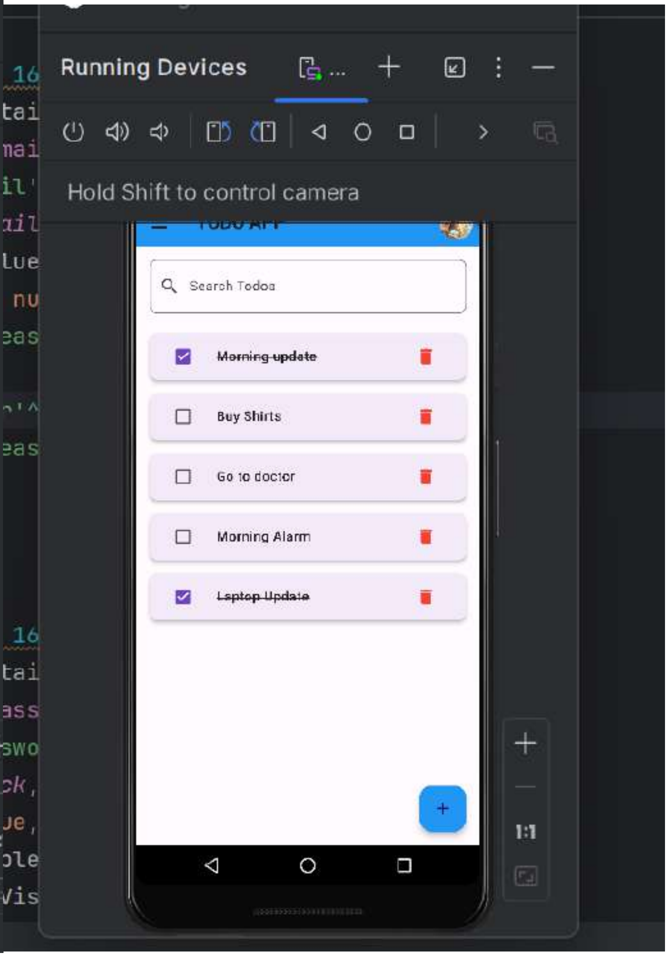
Google Meet



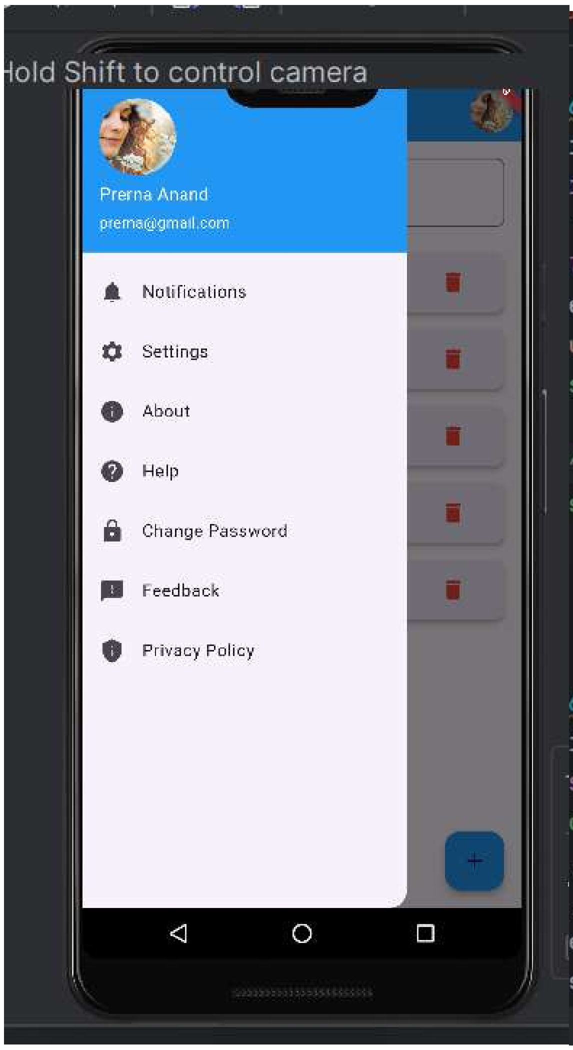
Delete Task

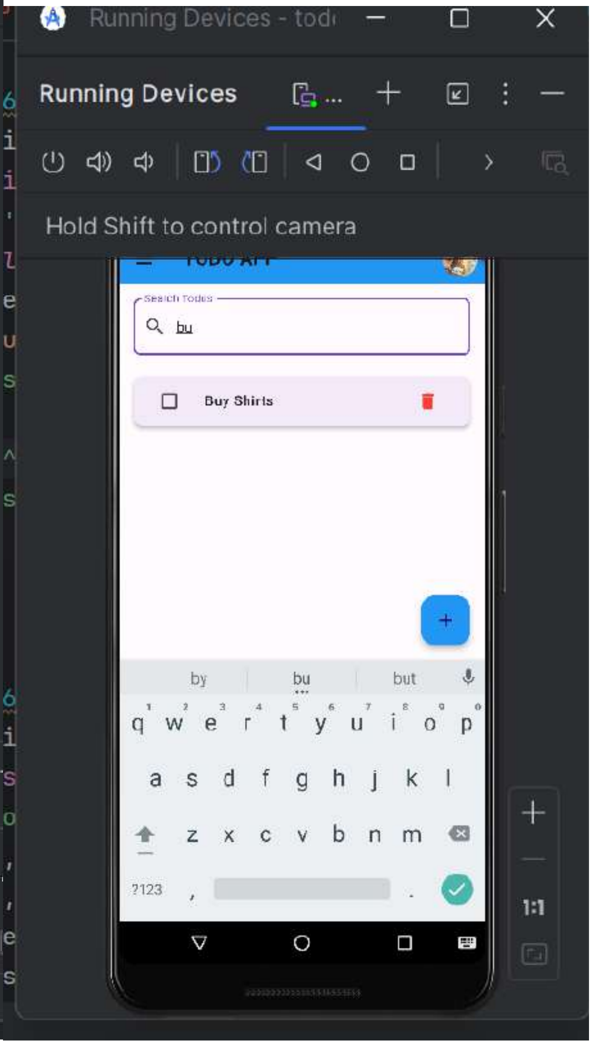
Are you sure you want to delete this task?

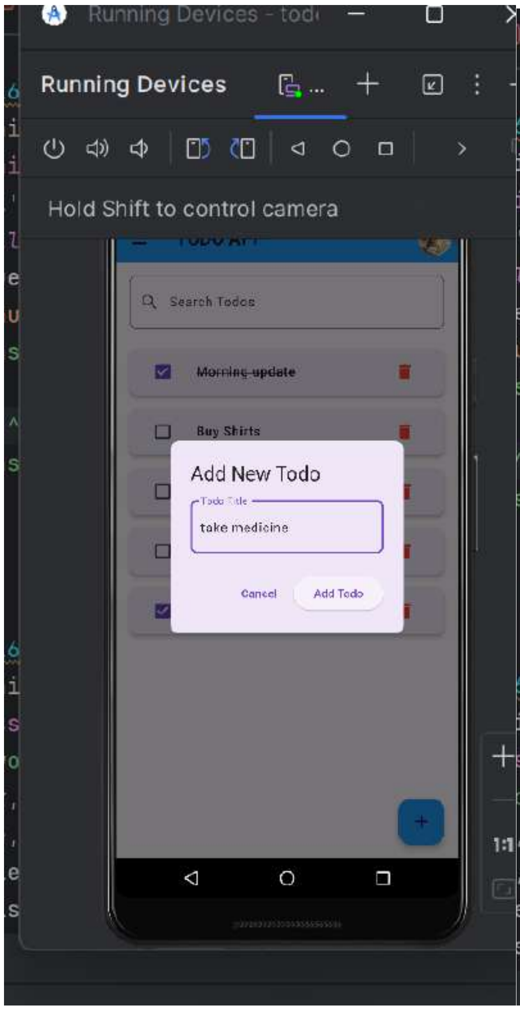
- [x] ~~Morning update~~
- [x] ~~Buy Shirts~~
- [ ] Morning Alarm
- [x] ~~Laptop Update~~
- [ ] Google Meet

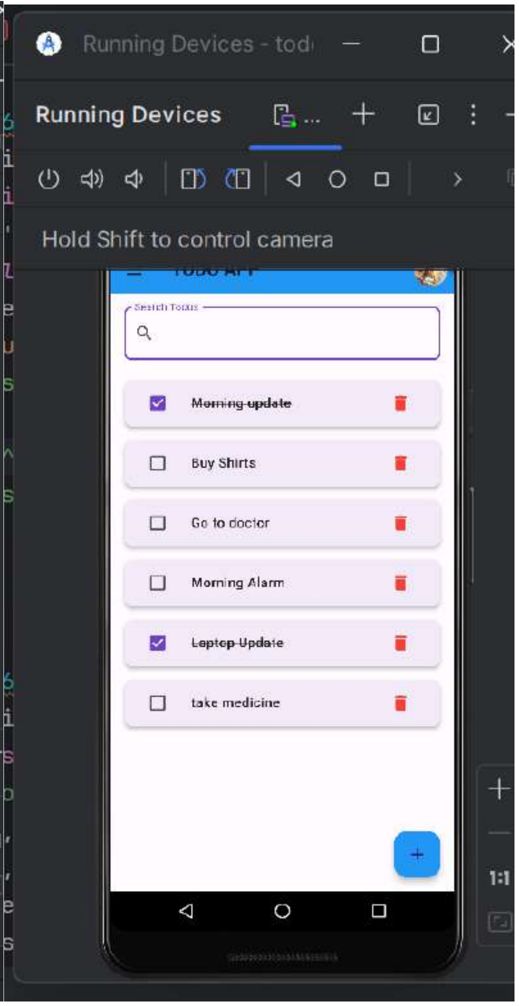


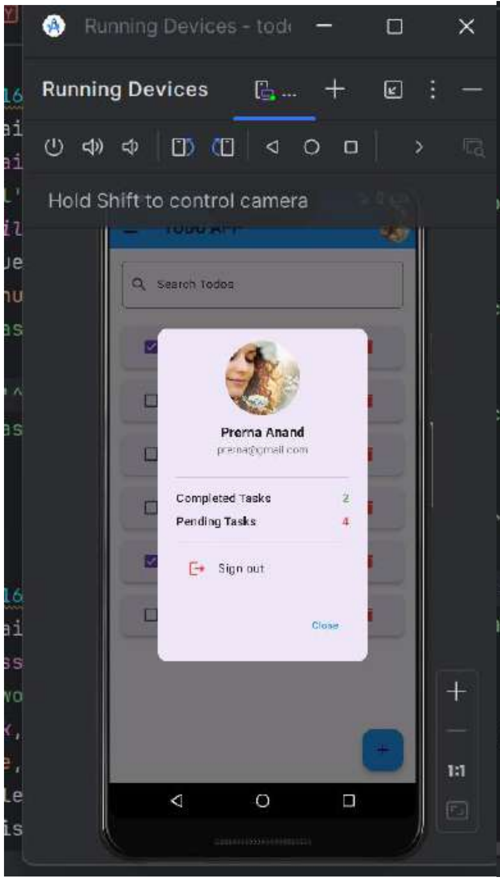
- [x] ~~Morning update~~
- [ ] Buy Shirts
- [ ] Go to doctor
- [ ] Morning Alarm
- [x] ~~Laptop Update~~


Hold Shift to control camera
Prerna Anand
prerna@gmail.com
Notifications
Settings
About
Help
Change Password
Feedback
Privacy Policy


Running Devices - todo
Running Devices
Hold Shift to control camera
Search Todos
bu
Buy Shirts
by
bu
but

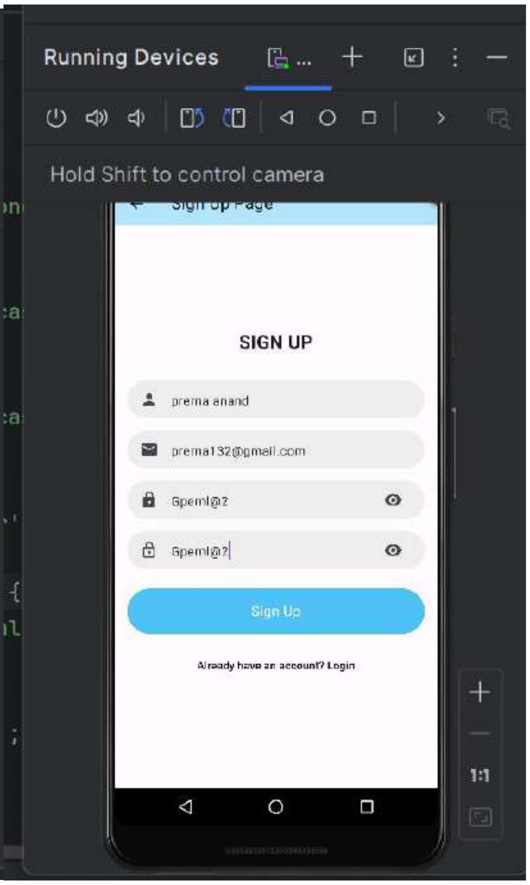

Running Devices - todo
Running Devices
Hold Shift to control camera
TODO APP
Search Todos
Morning update
Buy Shirts
Add New Todo
Todo Title
take medicine
Cancel
Add Todo


Running Devices - todo
Running Devices
Hold Shift to control camera
TODO APP
Search Todos
~~Morning update~~
Buy Shirts
Go to doctor
Morning Alarm
~~Laptop Update~~
take medicine



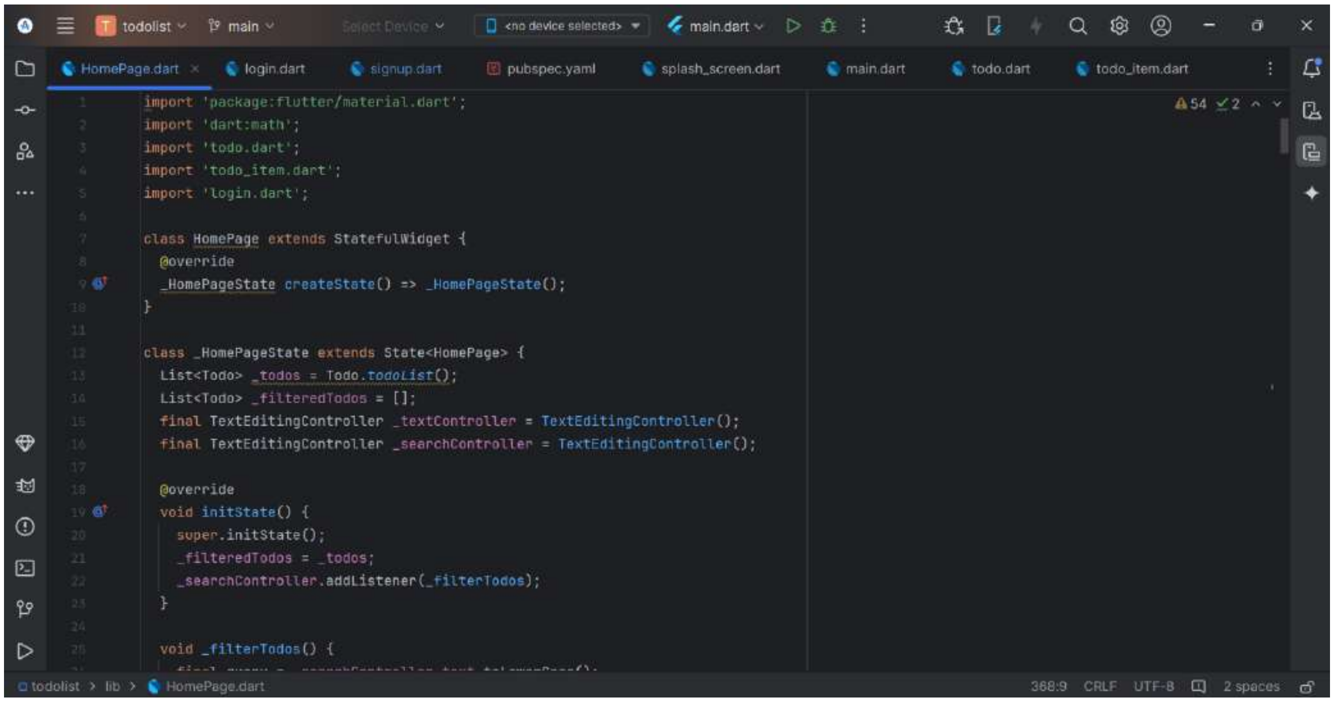
Prema Anand
prema@gmail.com

Completed Tasks 2
Pending Tasks 4

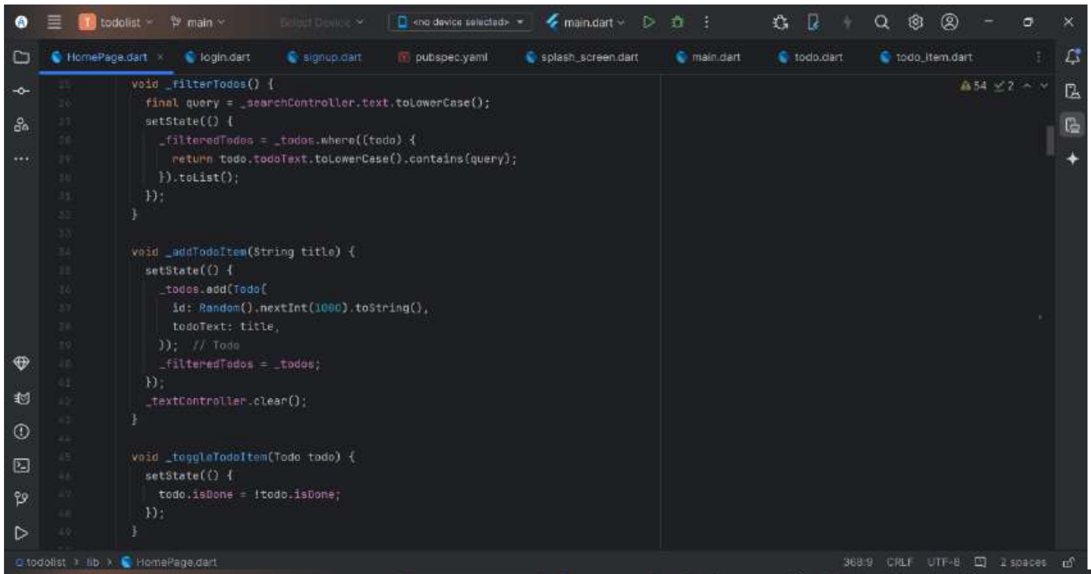


prema anand
prema132@gmail.com
6peml@2
6peml@2

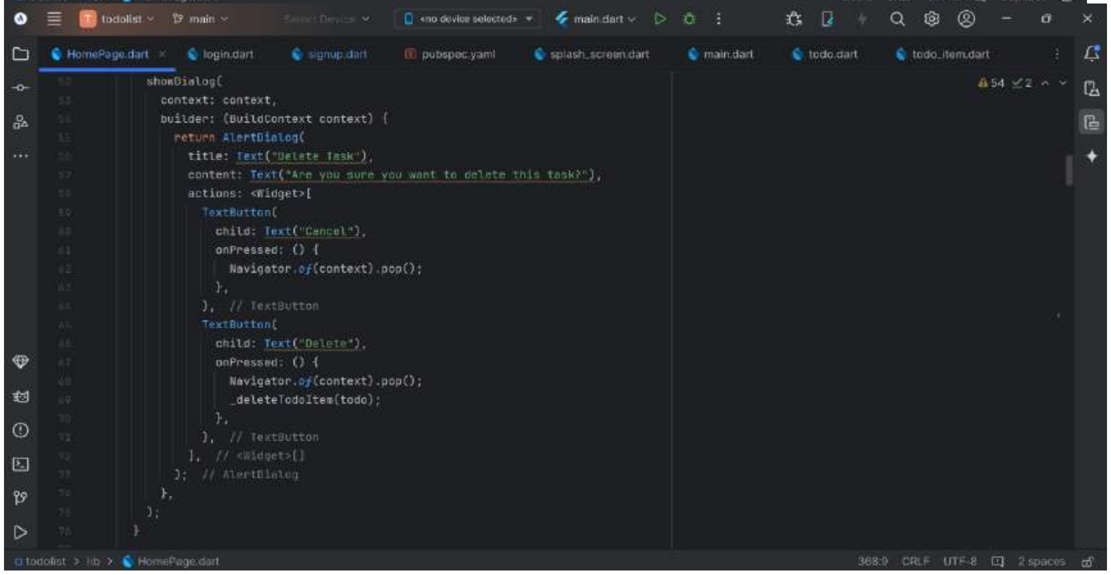
HOME PAGE:



```
import 'package:flutter/material.dart';
import 'dart:math';
import 'todo.dart';
import 'todo_item.dart';
import 'login.dart';

class HomePage extends StatefulWidget {
  @override
  _HomePageState createState() => _HomePageState();
}

class _HomePageState extends State<HomePage> {
  List<Todo> _todos = Todo.todoList();
  List<Todo> _filteredTodos = [];
  final TextEditingController _textController = TextEditingController();
  final TextEditingController _searchController = TextEditingController();

  @override
  void initState() {
    super.initState();
    _filteredTodos = _todos;
    _searchController.addListener(_filterTodos);
  }

  void _filterTodos() {
```



```
void _filterTodos() {
  final query = _searchController.text.toLowerCase();
  setState(() {
    _filteredTodos = _todos.where((todo) {
      return todo.todoText.toLowerCase().contains(query);
    }).toList();
  });
}

void _addTodoItem(String title) {
  setState(() {
    _todos.add(Todo(
      id: Random().nextInt(1000).toString(),
      todoText: title,
    )); // Todo
    _filteredTodos = _todos;
  });
  _textController.clear();
}

void _toggleTodoItem(Todo todo) {
  setState(() {
    todo.isDone = !todo.isDone;
  });
}
```



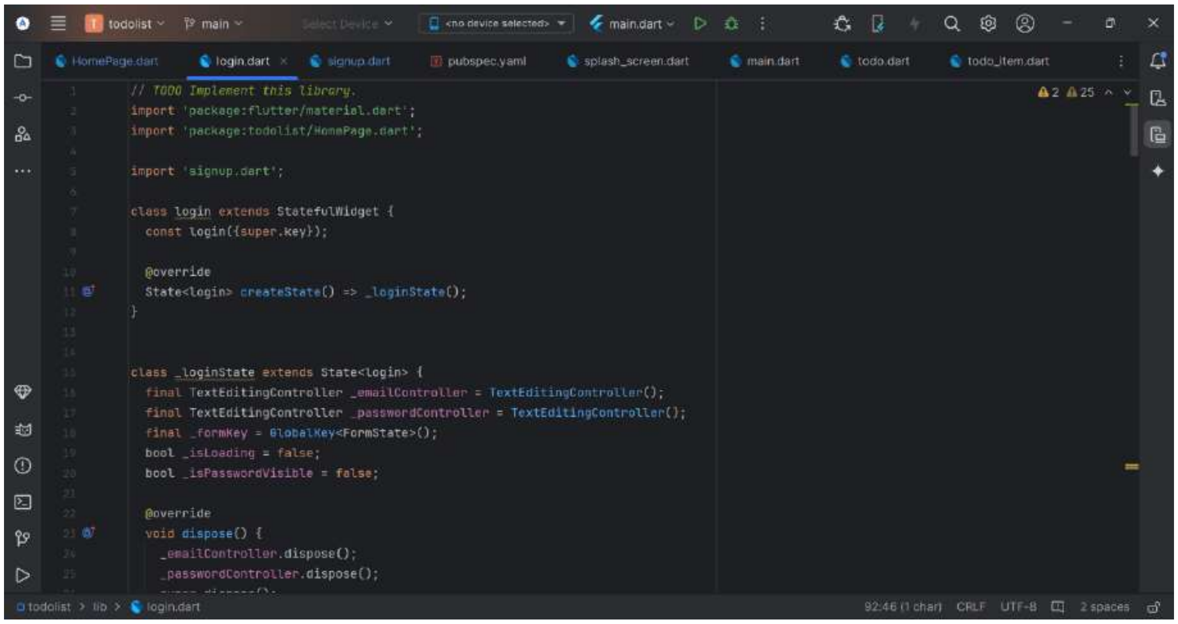
```
    showDialog(
      context: context,
      builder: (BuildContext context) {
        return AlertDialog(
          title: Text("Delete Task"),
          content: Text("Are you sure you want to delete this task?"),
          actions: <Widget>[
            TextButton(
              child: Text("Cancel"),
              onPressed: () {
                Navigator.of(context).pop();
              },
            ), // TextButton
            TextButton(
              child: Text("Delete"),
              onPressed: () {
                Navigator.of(context).pop();
                _deleteTodoItem(todo);
              },
            ), // TextButton
          ], // <Widget>[]
        ); // AlertDialog
      },
    );
  }
```

```
import 'package:flutter/material.dart';
import 'dart:math';
import 'todo.dart';
import 'todo_item.dart';
import 'login.dart';

class HomePage extends StatefulWidget {
  @override
  _HomePageState createState() => _HomePageState();
}

class _HomePageState extends State<HomePage> {
  List<Todo> _todos = Todo.todoList();
  List<Todo> _filteredTodos = [];
  final TextEditingController _textController = TextEditingController();
  final TextEditingController _searchController = TextEditingController();

  @override
  void initState() {
    super.initState();
    _filteredTodos = _todos;
    _searchController.addListener(_filterTodos);
  }
```

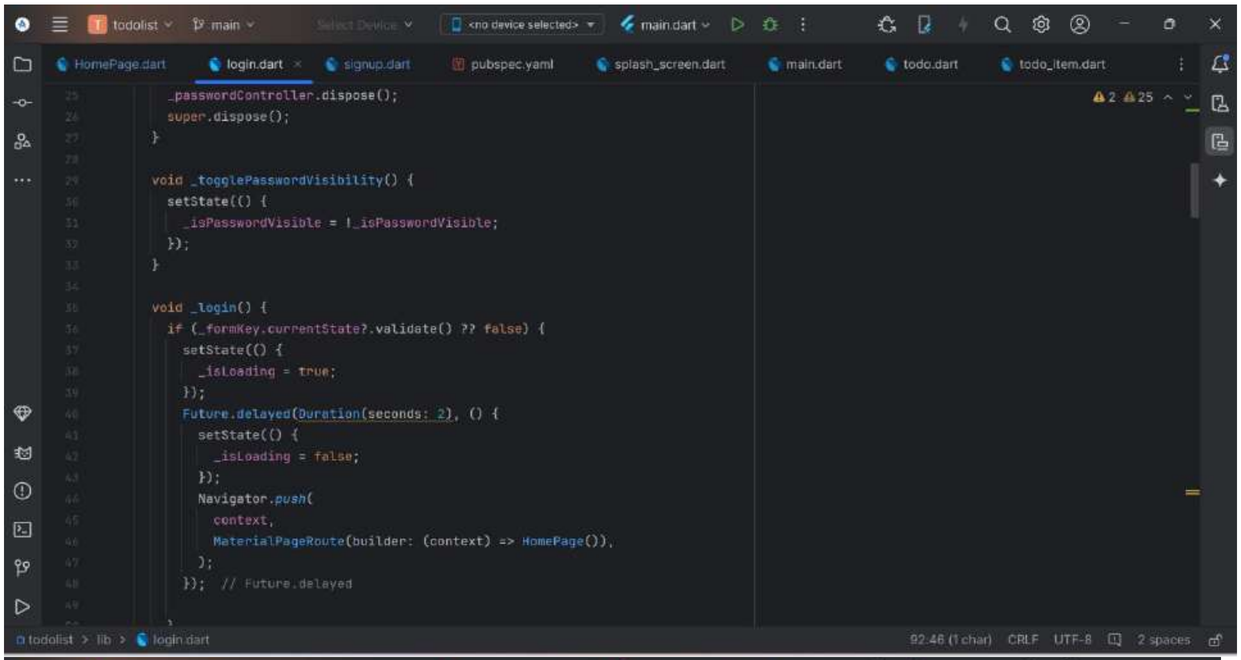
```
void _filterTodos() {
  final query = _searchController.text.toLowerCase();
  setState(() {
    _filteredTodos = _todos.where((todo) {
      return todo.todoText.toLowerCase().contains(query);
    }).toList();
  });
}

void _addTodoItem(String title) {
  setState(() {
    _todos.add(Todo(
      id: Random().nextInt(1000).toString(),
      todoText: title,
    ));
    _filteredTodos = _todos;
  });
  _textController.clear();
}

void _toggleTodoItem(Todo todo) {
  setState(() {
    todo.isDone = !todo.isDone;
  });
}

void _confirmDeleteTodoItem(BuildContext context, Todo todo) {
  showDialog(
    context: context,
    builder: (BuildContext context) {
      return AlertDialog(
        title: Text("Delete Task"),
        content: Text("Are you sure you want to delete this task?"),
        actions: <Widget>[
          TextButton(
            child: Text("Cancel"),
            onPressed: () {
              Navigator.of(context).pop();
            },
          ),
          TextButton(
            child: Text("Delete"),
            onPressed: () {
              Navigator.of(context).pop();
              _deleteTodoItem(todo);
            },
          ),
        ],
      );
    },
  );
}

void _deleteTodoItem(Todo todo) {
  setState(() {
    _todos.remove(todo);
    _filteredTodos = _todos;
  });
}

void _showProfileOptions(BuildContext context) {
```

```
String userEmail = 'prerna@gmail.com';
String userName = 'Prerna Anand';
int completedTasks = _todos.where((todo) => todo.isDone).length;
int pendingTasks = _todos.length - completedTasks;

showDialog(
  context: context,
  builder: (BuildContext context) {
    return AlertDialog(
      shape: RoundedRectangleBorder(
        borderRadius: BorderRadius.all(Radius.circular(10.0)),
      ),
      content: Column(
        mainAxisSize: MainAxisSize.min,
        children: <Widget>[
          ClipOval(
            child: Image.network(
              "https://cdn.pixabay.com/photo/2018/03/13/17/46/please-3223194_1280.jpg",
              width: 100,
              height: 100,
              fit: BoxFit.cover,
            ),
          ),
          SizedBox(height: 10),
          Text(
            userName,
            style: TextStyle(fontSize: 18, fontWeight: FontWeight.w600),
          ),
          Text(
            userEmail,
            style: TextStyle(color: Colors.grey[600], fontSize: 14),
          ),
          SizedBox(height: 20),
          Divider(thickness: 1),
          Padding(
            padding: const EdgeInsets.symmetric(vertical: 8.0),
            child: Column(
              children: [
                Row(
                  mainAxisAlignment: MainAxisAlignment.spaceBetween,
                  children: [
                    Text(
                      'Completed Tasks',
                      style: TextStyle(fontSize: 16, fontWeight: FontWeight.w500),
                    ),
                    Text(
                      '$completedTasks',
                      style: TextStyle(fontSize: 16, fontWeight: FontWeight.w500, color: Colors.green),
                    ),
                  ],
                ),
                SizedBox(height: 8),
                Row(
                  mainAxisAlignment: MainAxisAlignment.spaceBetween,
                  children: [
                    Text(
                      'Pending Tasks',
                      style: TextStyle(fontSize: 16, fontWeight: FontWeight.w500),
                    ),
                    Text(
                      '$pendingTasks',
                      style: TextStyle(fontSize: 16, fontWeight: FontWeight.w500, color: Colors.red),
                      ),
                    ],
                  ),
                ],
              ),
            ),
            Divider(thickness: 1),
            ListTile(
              leading: Icon(Icons.logout, color: Colors.red),
              title: Text('Sign out'),
              onTap: () {
                Navigator.of(context).pop();
                Navigator.of(context).pushReplacement(
                  MaterialPageRoute(builder: (context) => login()),
                );
              },
            ),
          ],
        ),
        actions: <Widget>[
          TextButton(
            child: Text(
              "Close",
              style: TextStyle(color: Colors.blue),
            ),
            onPressed: () {
              Navigator.of(context).pop();
            },
          ),
        ],
      );
    },
  );
}

void _onMenuOptionSelected(String value) {

  switch (value) {
    case 'Settings':
      break;
    case 'About':
      break;
    case 'Help':
      break;
    case 'Change Password':
      break;
    case 'Feedback':
      break;
  }
}

void _showAddTodoDialog(BuildContext context) {
  showDialog(
    context: context,
    builder: (BuildContext context) {
      return AlertDialog(
        shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(10.0),
        ),
        title: Text('Add New Todo'),
        content: TextField(
          controller: _textController,
          decoration: InputDecoration(
            labelText: 'Todo Title',
            border: OutlineInputBorder(
              borderRadius: BorderRadius.circular(8.0),
            ),
          ),
        ),
        actions: <Widget>[
          TextButton(
            child: Text('Cancel'),
            onPressed: () {
              Navigator.of(context).pop();
            },
          ),
          ElevatedButton(
            onPressed: () {
              if (_textController.text.isNotEmpty) {
                _addTodoItem(_textController.text);
                Navigator.of(context).pop();
              }
            },
            child: Text('Add Todo'),
          ),
        ],
      );
    },
  );
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      backgroundColor: Colors.blue,
      leading: Builder(
        builder: (context) => IconButton(
          icon: Icon(Icons.menu),
          onPressed: () => Scaffold.of(context).openDrawer(),
        ),
      ),
      title: Text('TODO APP', style: TextStyle(fontWeight: FontWeight.bold)),
      actions: <Widget>[
        Padding(
          padding: const EdgeInsets.all(8.0),
          child: GestureDetector(
            onTap: () => _showProfileOptions(context),
            child: ClipOval(
              child: Image.network(
                "https://cdn.pixabay.com/photo/2018/03/13/17/46/please-3223194_1280.jpg",
                width: 40,
                height: 40,
                fit: BoxFit.cover,
              ),
            ),
          ),
        ),
      ],
    ),
    drawer: Drawer(
      child: ListView(
        padding: EdgeInsets.zero,
        children: <Widget>[
    UserAccountsDrawerHeader(
      accountName: Text("Prerna Anand"),
      accountEmail: Text("prerna@gmail.com"),
      currentAccountPicture: ClipOval(
        child: Image.network(
          "https://cdn.pixabay.com/photo/2018/03/13/17/46/please-3223194_1280.jpg",
          width: 40,
          height: 40,
          fit: BoxFit.cover,
        ),
      ),
      decoration: BoxDecoration(
        color: Colors.blue,
      ),
    ),
    ListTile(
      leading: Icon(Icons.notifications),
      title: Text('Notifications'),
      onTap: () {},
    ),
    ListTile(
      leading: Icon(Icons.settings),
      title: Text('Settings'),
      onTap: () {
        _onMenuOptionSelected('Settings');
      },
    ),
    ListTile(
      leading: Icon(Icons.info),
      title: Text('About'),
      onTap: () {
        _onMenuOptionSelected('About');
      },
    ),
    ListTile(
      leading: Icon(Icons.help),
      title: Text('Help'),
      onTap: () {
        _onMenuOptionSelected('Help');
      },
    ),
    ListTile(
      leading: Icon(Icons.lock),
      title: Text('Change Password'),
      onTap: () {
        _onMenuOptionSelected('Change Password');
      },
    ),
    ListTile(
      leading: Icon(Icons.feedback),
      title: Text('Feedback'),
      onTap: () {
        _onMenuOptionSelected('Feedback');
      },
    ),
    ListTile(
      leading: Icon(Icons.privacy_tip),
      title: Text('Privacy Policy'),
      onTap: () {},
    ),
  ],
),
      ),
      body: Column(
        children: <Widget>[
          Padding(
            padding: const EdgeInsets.all(16.0),
            child: TextField(
              controller: _searchController,
              decoration: InputDecoration(
                labelText: 'Search Todos',
                border: OutlineInputBorder(
                  borderRadius: BorderRadius.circular(8.0),
                ),
                prefixIcon: Icon(Icons.search),
              ),
            ),
          ),
          Expanded(
            child: ListView.builder(
              itemCount: _filteredTodos.length,
              itemBuilder: (context, index) {
                return TodoItem(
                  todo: _filteredTodos[index],
                  onToggle: _toggleTodoItem,
                  onDelete: (todo) => _confirmDeleteTodoItem(context, todo),
                );
              },
            ),
          ),
        ],
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: () => _showAddTodoDialog(context),
        child: Icon(Icons.add),
        backgroundColor: Colors.blue,
      ),
    );
  }
}
```

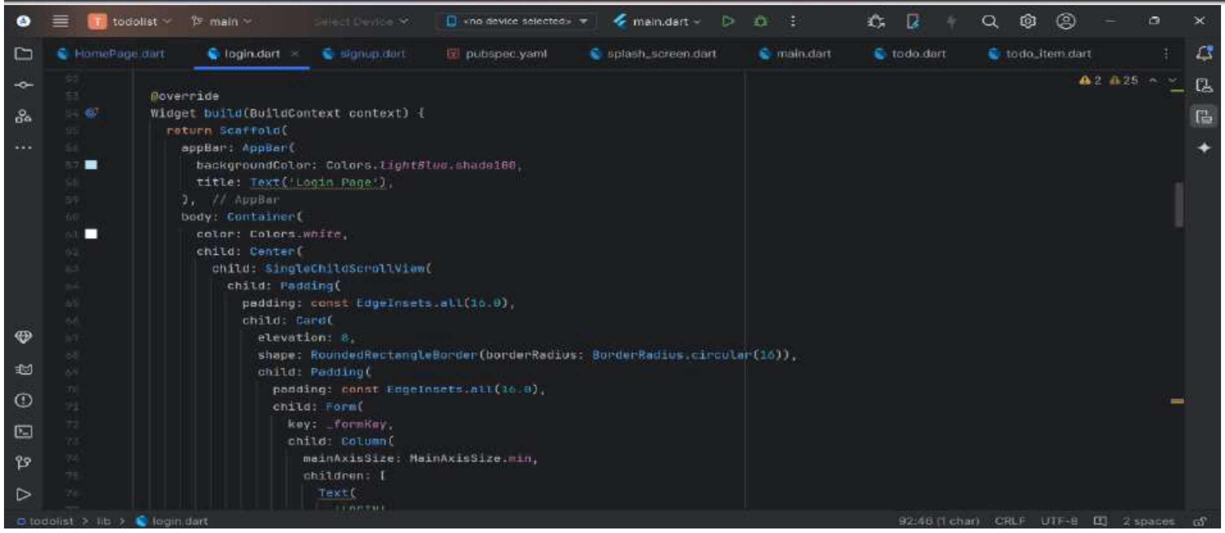
LOGIN PAGE:



```
// TODO Implement this library.
import 'package:flutter/material.dart';
import 'package:todolist/HomePage.dart';

import 'signup.dart';

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

  @override
  State<login> createState() => _loginState();
}


class _loginState extends State<login> {
  final TextEditingController _emailController = TextEditingController();
  final TextEditingController _passwordController = TextEditingController();
  final _formKey = GlobalKey<FormState>();
  bool _isLoading = false;
  bool _isPasswordVisible = false;

  @override
  void dispose() {
    _emailController.dispose();
    _passwordController.dispose();
```



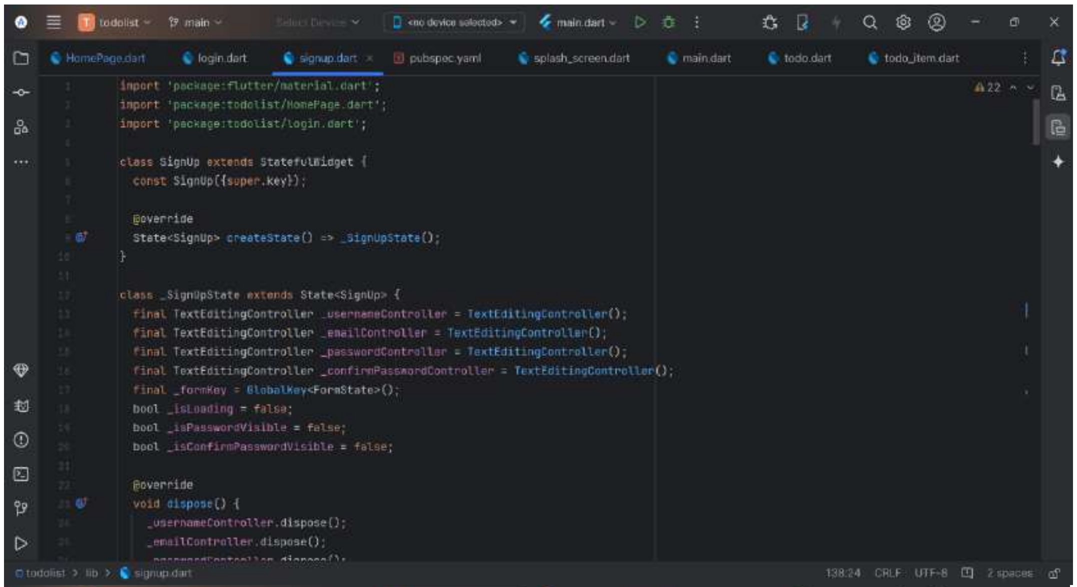
```
    _passwordController.dispose();
    super.dispose();
  }

  void _togglePasswordVisibility() {
    setState(() {
      _isPasswordVisible = !_isPasswordVisible;
    });
  }

  void _login() {
    if (_formKey.currentState?.validate() ?? false) {
      setState(() {
        _isLoading = true;
      });
      Future.delayed(Duration(seconds: 2), () {
        setState(() {
          _isLoading = false;
        });
        Navigator.push(
          context,
          MaterialPageRoute(builder: (context) => HomePage()),
        );
      });  // Future.delayed
```



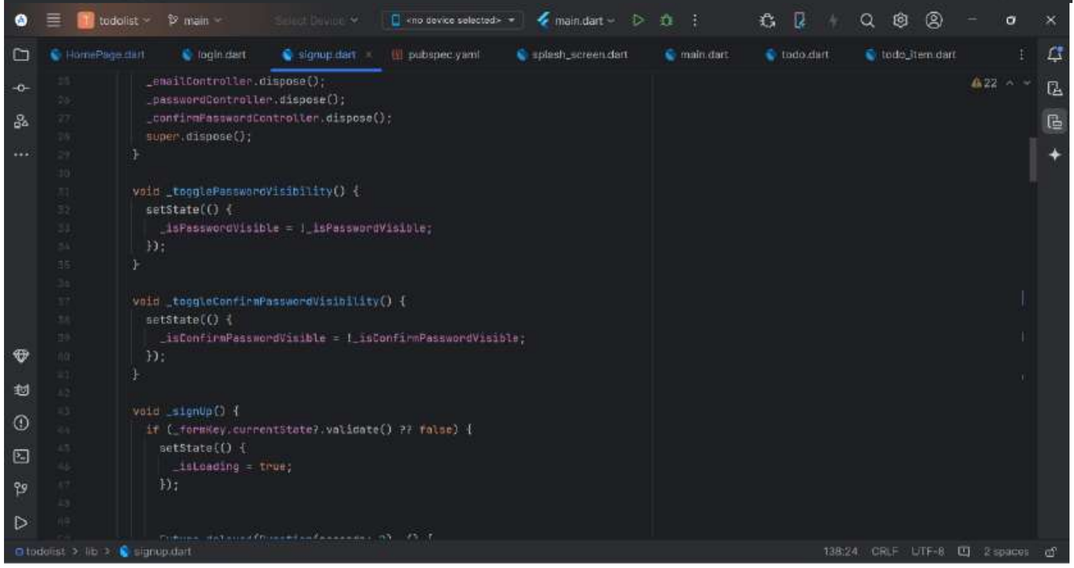
```
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor: Colors.lightBlue.shade100,
        title: Text('Login Page'),
      ),  // AppBar
      body: Container(
        color: Colors.white,
        child: Center(
          child: SingleChildScrollView(
            child: Padding(
              padding: const EdgeInsets.all(16.0),
              child: Card(
                elevation: 8,
                shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(16)),
                child: Padding(
                  padding: const EdgeInsets.all(16.0),
                  child: Form(
                    key: _formKey,
                    child: Column(
                      mainAxisSize: MainAxisSize.min,
                      children: [
                        Text(
```

```
// TODO Implement this library.
import 'package:flutter/material.dart';
import 'package:todolist/HomePage.dart';
import 'signup.dart';

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

  @override
  State<login> createState() => _loginState();
}


class _loginState extends State<login> {
  final TextEditingController _emailController = TextEditingController();
  final TextEditingController _passwordController = TextEditingController();
  final _formKey = GlobalKey<FormState>();
  bool _isLoading = false;
  bool _isPasswordVisible = false;

  @override
  void dispose() {
    _emailController.dispose();
    _passwordController.dispose();
    super.dispose();
  }

  void _togglePasswordVisibility() {
    setState(() {
      _isPasswordVisible = !_isPasswordVisible;
    });
  }

  void _login() {
    if (_formKey.currentState?.validate() ?? false) {
      setState(() {
        _isLoading = true;
      });
      Future.delayed(Duration(seconds: 2), () {
        setState(() {
          _isLoading = false;
        });
        Navigator.push(
          context,
          MaterialPageRoute(builder: (context) => HomePage()),
        );
      });

    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor: Colors.lightBlue.shade100,
        title: Text('Login Page'),
      ),
      body: Container(
        color: Colors.white,
        child: Center(
          child: SingleChildScrollView(
            child: Padding(
              padding: const EdgeInsets.all(16.0),
              child: Card(
                elevation: 8,
                shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(16)),
                child: Padding(
                  padding: const EdgeInsets.all(16.0),
                  child: Form(
                    key: _formKey,
                    child: Column(
                      mainAxisSize: MainAxisSize.min,
                      children: [
                        Text(
                          'LOGIN',
                          style: TextStyle(
                            fontSize: 24,
                            fontWeight: FontWeight.bold,
                          ),
                        ),
                        SizedBox(height: 16),
                        _buildRoundedContainer(
                          controller: _emailController,
                          hintText: 'Email',
    icon: Icons.email,
    validator: (value) {
      if (value == null || value.isEmpty) {
        return 'Please enter your email';
      }
      if (!RegExp(r'^[^@]+@[^@]+\.[^@]+').hasMatch(value)) {
        return 'Please enter a valid email';
      }
      return null;
    },
  ),
  SizedBox(height: 16),
  _buildRoundedContainer(
    controller: _passwordController,
    hintText: 'Password',
    icon: Icons.lock,
    isPassword: true,
    isPasswordVisible: _isPasswordVisible,
    togglePasswordVisibility: _togglePasswordVisibility,
    validator: (value) {
      if (value == null || value.isEmpty) {
        return 'Please enter your password';
      }
      if (!RegExp(r'[A-Z]').hasMatch(value)) {
        return 'Password must contain at least one uppercase letter';
      }
      if (!RegExp(r'[0-9]').hasMatch(value)) {
        return 'Password must contain at least one number';
      }
      if (!RegExp(r'[!@#$%^&*(),.?":{}|<>]').hasMatch(value)) {
        return 'Password must contain at least one special character';
      }
      return null;
    },
  ),
  SizedBox(height: 25),
  _isLoading
      ? CircularProgressIndicator()
      : GestureDetector(
    onTap: _login,
    child: Container(
      padding: EdgeInsets.symmetric(vertical: 16),
      decoration: BoxDecoration(
        color: Colors.lightBlue.shade300,
        borderRadius: BorderRadius.circular(32),
      ),
      child: Center(
        child: Text(
          'Continue',
          style: TextStyle(
            color: Colors.white,
            fontSize: 18,
          ),
        ),
      ),
    ),
  ),
  SizedBox(height: 16),
  TextButton(
    onPressed: () {
      Navigator.push(
        context,
                      MaterialPageRoute(builder: (context) => SignUp()),
                    );
                  },
                  child: Text(
                    "Don't have an account? Sign Up",
                    style: TextStyle(color: Colors.black),
                  ),
                ),
                SizedBox(height: 8),
                Text(
                  'or',
                  style: TextStyle(
                    fontSize: 16,
                    color: Colors.black54,
                  ),
                ),
                SizedBox(height: 8),
                ElevatedButton.icon(
                  icon: Icon(Icons.login),
                  label: Text('Sign up with Google'),
                  onPressed: () {
                    // Implement your Google sign-up logic here
                  },
                  style: ElevatedButton.styleFrom(
                    primary: Colors.lightBlue.shade300,
                    onPrimary: Colors.white,
                    shape: RoundedRectangleBorder(
                      borderRadius: BorderRadius.circular(32),
                    ),
                  ),
                ),
              ],
            ),
          ),
        ),
      ),
    ),
  ),
  ),
  ),
  );
  }

  Widget _buildRoundedContainer({
    required TextEditingController controller,
    required String hintText,
    required IconData icon,
    bool isPassword = false,
    bool isPasswordVisible = false,
    void Function()? togglePasswordVisibility,
    String? Function(String?)? validator,
  }) {
    return Container(
      padding: EdgeInsets.symmetric(horizontal: 16),
      decoration: BoxDecoration(
        color: Colors.grey.shade50,
        borderRadius: BorderRadius.circular(32),
      ),
      child: TextFormField(
        controller: controller,
        decoration: InputDecoration(
          icon: Icon(icon),
          hintText: hintText,
          border: InputBorder.none,
          suffixIcon: isPassword
              ? IconButton(
            icon: Icon(
              isPasswordVisible ? Icons.visibility : Icons.visibility_off,
            ),
            onPressed: togglePasswordVisibility,
          )
              : null,
        ),
        obscureText: isPassword && !isPasswordVisible,
        validator: validator,
      ),
    );
  }
}
```

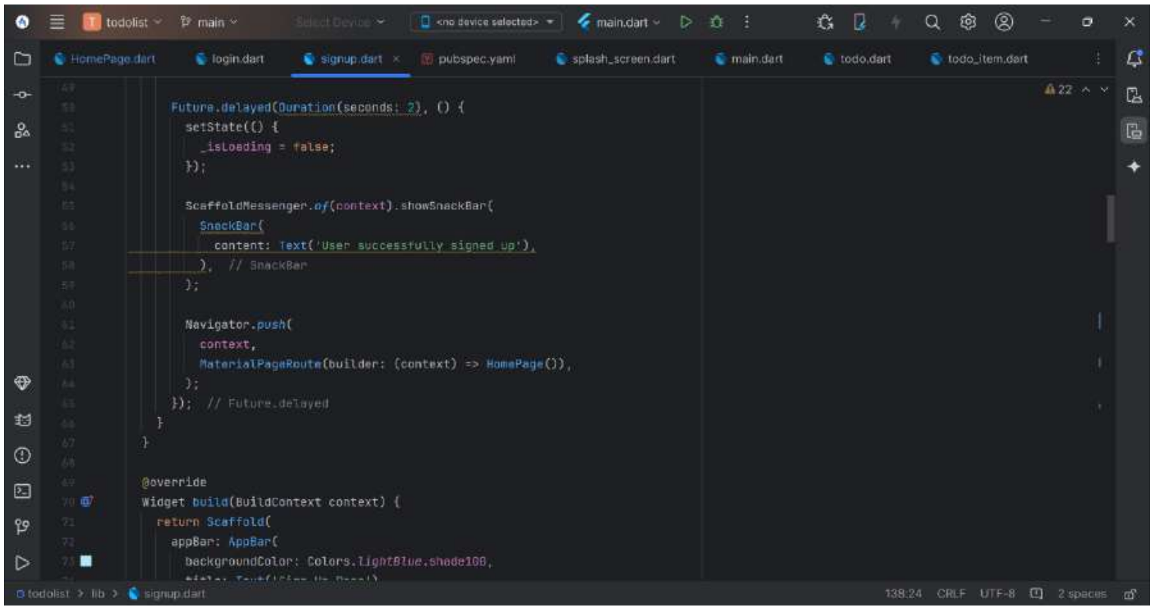
SIGN UP PAGE:



```
import 'package:flutter/material.dart';
import 'package:todolist/HomePage.dart';
import 'package:todolist/login.dart';

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

  @override
  State<SignUp> createState() => _SignUpState();
}

class _SignUpState extends State<SignUp> {
  final TextEditingController _usernameController = TextEditingController();
  final TextEditingController _emailController = TextEditingController();
  final TextEditingController _passwordController = TextEditingController();
  final TextEditingController _confirmPasswordController = TextEditingController();
  final _formKey = GlobalKey<FormState>();
  bool _isLoading = false;
  bool _isPasswordVisible = false;
  bool _isConfirmPasswordVisible = false;

  @override
  void dispose() {
    _usernameController.dispose();
    _emailController.dispose();
```



```
    _emailController.dispose();
    _passwordController.dispose();
    _confirmPasswordController.dispose();
    super.dispose();
  }

  void _togglePasswordVisibility() {
    setState(() {
      _isPasswordVisible = !_isPasswordVisible;
    });
  }

  void _toggleConfirmPasswordVisibility() {
    setState(() {
      _isConfirmPasswordVisible = !_isConfirmPasswordVisible;
    });
  }

  void _signUp() {
    if (_formKey.currentState?.validate() ?? false) {
      setState(() {
        _isLoading = true;
      });
```

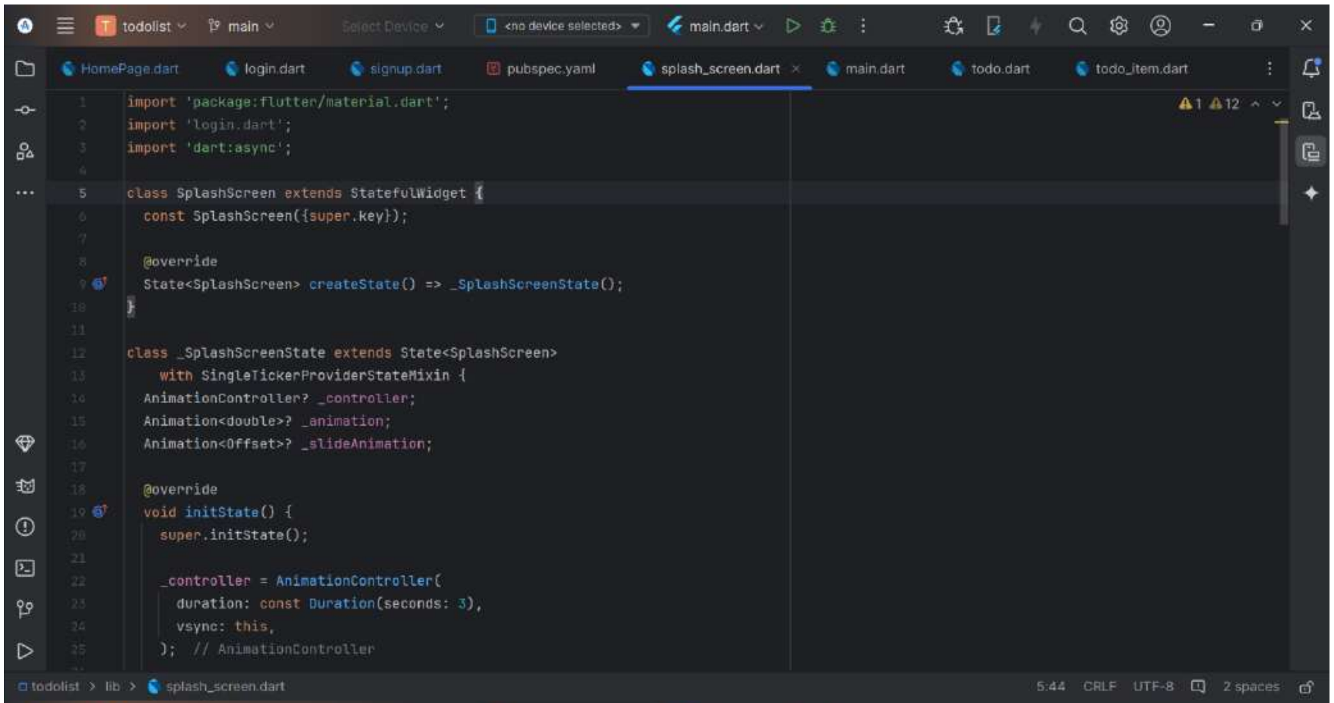


```
      Future.delayed(Duration(seconds: 2), () {
        setState(() {
          _isLoading = false;
        });

        ScaffoldMessenger.of(context).showSnackBar(
          SnackBar(
            content: Text('User successfully signed up'),
          ), // SnackBar
        );

        Navigator.push(
          context,
          MaterialPageRoute(builder: (context) => HomePage()),
        );
      }); // Future.delayed
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor: Colors.lightBlue.shade100,
```

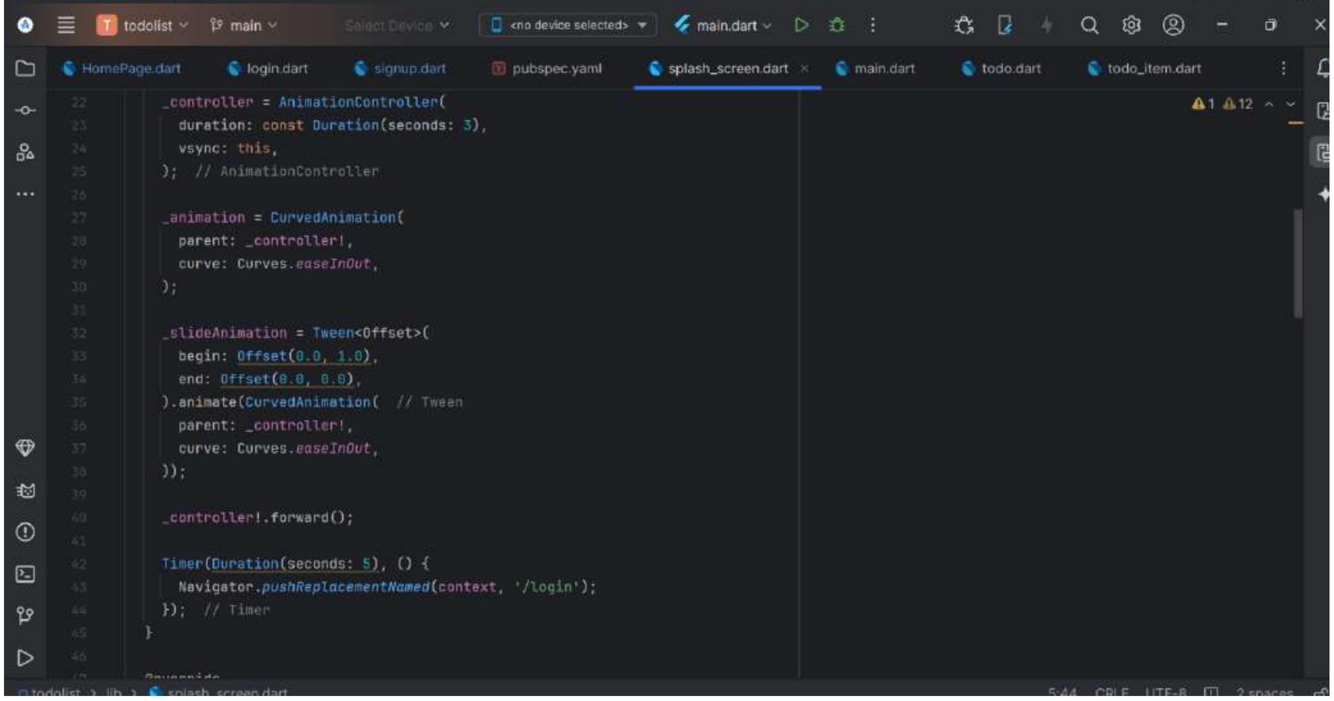
```
import 'package:flutter/material.dart';
import 'package:todolist/HomePage.dart';
import 'package:todolist/login.dart';

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

  @override
  State<SignUp> createState() => _SignUpState();
}

class _SignUpState extends State<SignUp> {
  final TextEditingController _usernameController = TextEditingController();
  final TextEditingController _emailController = TextEditingController();
  final TextEditingController _passwordController = TextEditingController();
  final TextEditingController _confirmPasswordController = TextEditingController();
  final _formKey = GlobalKey<FormState>();
  bool _isLoading = false;
  bool _isPasswordVisible = false;
  bool _isConfirmPasswordVisible = false;

  @override
  void dispose() {
    _usernameController.dispose();
    _emailController.dispose();
    _passwordController.dispose();
    _confirmPasswordController.dispose();
    super.dispose();
  }

  void _togglePasswordVisibility() {
    setState(() {
      _isPasswordVisible = !_isPasswordVisible;
    });
  }

  void _toggleConfirmPasswordVisibility() {
    setState(() {
      _isConfirmPasswordVisible = !_isConfirmPasswordVisible;
    });
  }
```

```
void _signUp() {
  if (_formKey.currentState?.validate() ?? false) {
    setState(() {
      _isLoading = true;
    });


    Future.delayed(Duration(seconds: 2), () {
      setState(() {
        _isLoading = false;
      });

      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(
          content: Text('User successfully signed up'),
        ),
      );

      Navigator.push(
        context,
        MaterialPageRoute(builder: (context) => HomePage()),
      );
    });
  }
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      backgroundColor: Colors.lightBlue.shade100,
      title: Text('Sign Up Page'),
    ),
    body: Padding(
      padding: EdgeInsets.all(16.0),
      child: Center(
        child: SingleChildScrollView(
          child: Form(
            key: _formKey,
            child: Column(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
                Text(
                  'SIGN UP',
                  style: TextStyle(
                    fontSize: 24,
                    fontWeight: FontWeight.bold,
                  ),
                ),
                SizedBox(height: 32),
                _buildRoundedContainer(
                  controller: _usernameController,
                  hintText: 'Username',
                  icon: Icons.person,
                  validator: (value) {
                    if (value == null || value.isEmpty) {
                      return 'Please enter your username';
                    }
                    return null;
                  },
                ),
                SizedBox(height: 16),
          _buildRoundedContainer(
            controller: _emailController,
            hintText: 'Email',
            icon: Icons.email,
            validator: (value) {
    if (value == null || value.isEmpty) {
    return 'Please enter your email';
    }
  if (!RegExp(r'^[^@]+@[^@]+\.[^@]+').hasMatch(value)) {
  return 'Please enter a valid email';
    }


  return null;
            },
          ),
          SizedBox(height: 16),
          _buildRoundedContainer(
            controller: _passwordController,
            hintText: 'Password',
            icon: Icons.lock,
            isPassword: true,
            isPasswordVisible: _isPasswordVisible,
            togglePasswordVisibility: _togglePasswordVisibility,
            validator: (value) {
              if (value == null || value.isEmpty) {
                return 'Please enter your password';
              }
              if (value.length < 8) {
                return 'Password must be at least 8 characters long';
              }
              if (!RegExp(r'^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)[a-zA-Z\d]{8,}$').hasMatch(value)) {
                return 'Password must include at least one uppercase letter, one lowercase letter, and one number';
              }
              return null;
            },
          ),
          SizedBox(height: 16),
          _buildRoundedContainer(
            controller: _confirmPasswordController,
            hintText: 'Confirm Password',
            icon: Icons.lock_outline,
            isPassword: true,
            isPasswordVisible: _isConfirmPasswordVisible,
            togglePasswordVisibility: _toggleConfirmPasswordVisibility,
            validator: (value) {
              if (value == null || value.isEmpty) {
                return 'Please confirm your password';
              }
              if (value != _passwordController.text) {
                return 'Passwords do not match';
              }

              return null;
            },
          ),
          SizedBox(height: 25),
          _isLoading
              ? CircularProgressIndicator()
              : GestureDetector(
            onTap: _signUp,
            child: Container(
                    padding: EdgeInsets.symmetric(vertical: 16),
                    decoration: BoxDecoration(
                      color: Colors.lightBlue.shade300,
                      borderRadius: BorderRadius.circular(32),
                    ),
                    child: Center(
                      child: Text(
                        'Sign Up',
                        style: TextStyle(
                          color: Colors.white,
                          fontSize: 18,
                        ),
                      ),
                    ),
                  ),
                ),
                SizedBox(height: 16),
                TextButton(
                  onPressed: () {
                    Navigator.push(
                      context,
                      MaterialPageRoute(builder: (context) => login()),
                    );
                  },
                  child: Text(
                    "Already have an account? Login",
                    style: TextStyle(color: Colors.black),
                  ),
                ),
              ],
            ),
          ),
        ),
      ),
    ),
  );
}

Widget _buildRoundedContainer({
  required TextEditingController controller,
  required String hintText,
  required IconData icon,
  bool isPassword = false,
  bool isPasswordVisible = false,
  void Function()? togglePasswordVisibility,
  String? Function(String?)? validator,
}) {
  return Container(
    padding: EdgeInsets.symmetric(horizontal: 16),
    decoration: BoxDecoration(
      color: Colors.grey.shade200,
      borderRadius: BorderRadius.circular(32),
    ),
    child: TextFormField(
      controller: controller,
      decoration: InputDecoration(
        icon: Icon(icon),
        hintText: hintText,
        border: InputBorder.none,
        suffixIcon: isPassword
            ? IconButton(
          icon: Icon(
              isPasswordVisible ? Icons.visibility : Icons.visibility_off,
            ),
            onPressed: togglePasswordVisibility,
          )
              : null,
        ),
        obscureText: isPassword && !isPasswordVisible,
        validator: validator,
      ),
    );
  }
}
```

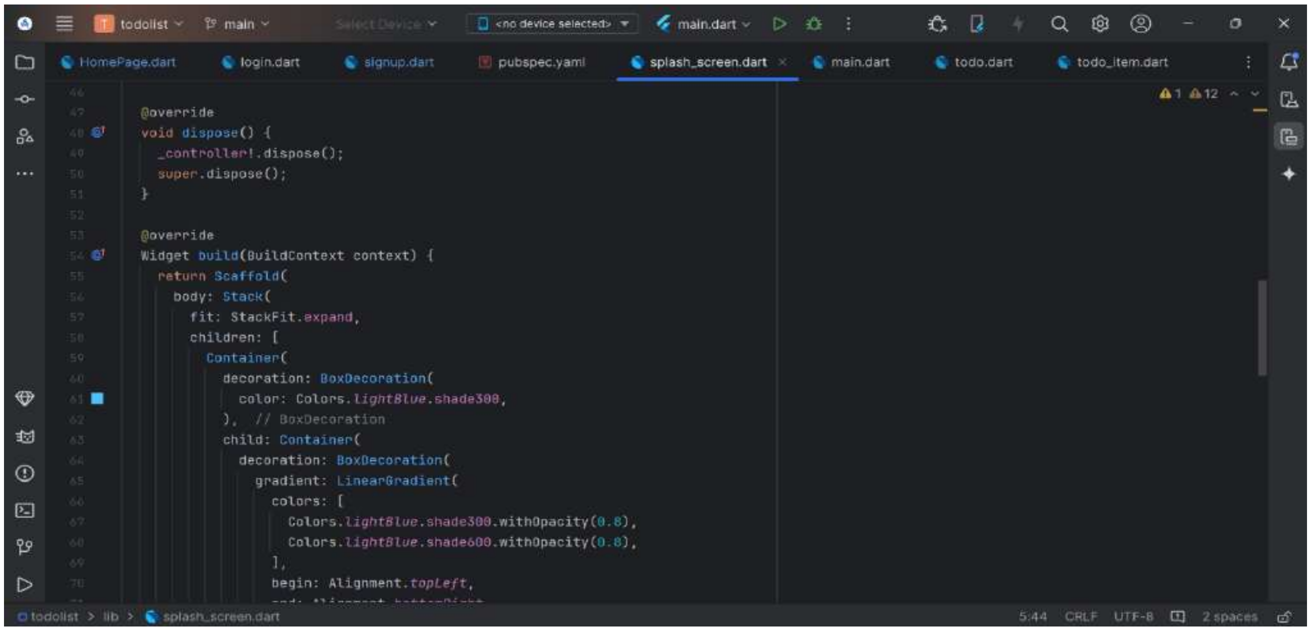
SPLASH SCREEN:



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

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

  @override
  State<SplashScreen> createState() => _SplashScreenState();
}

class _SplashScreenState extends State<SplashScreen>
    with SingleTickerProviderStateMixin {
  AnimationController? _controller;
  Animation<double>? _animation;
  Animation<Offset>? _slideAnimation;

  @override
  void initState() {
    super.initState();

    _controller = AnimationController(
      duration: const Duration(seconds: 3),
      vsync: this,
    );  // AnimationController
```



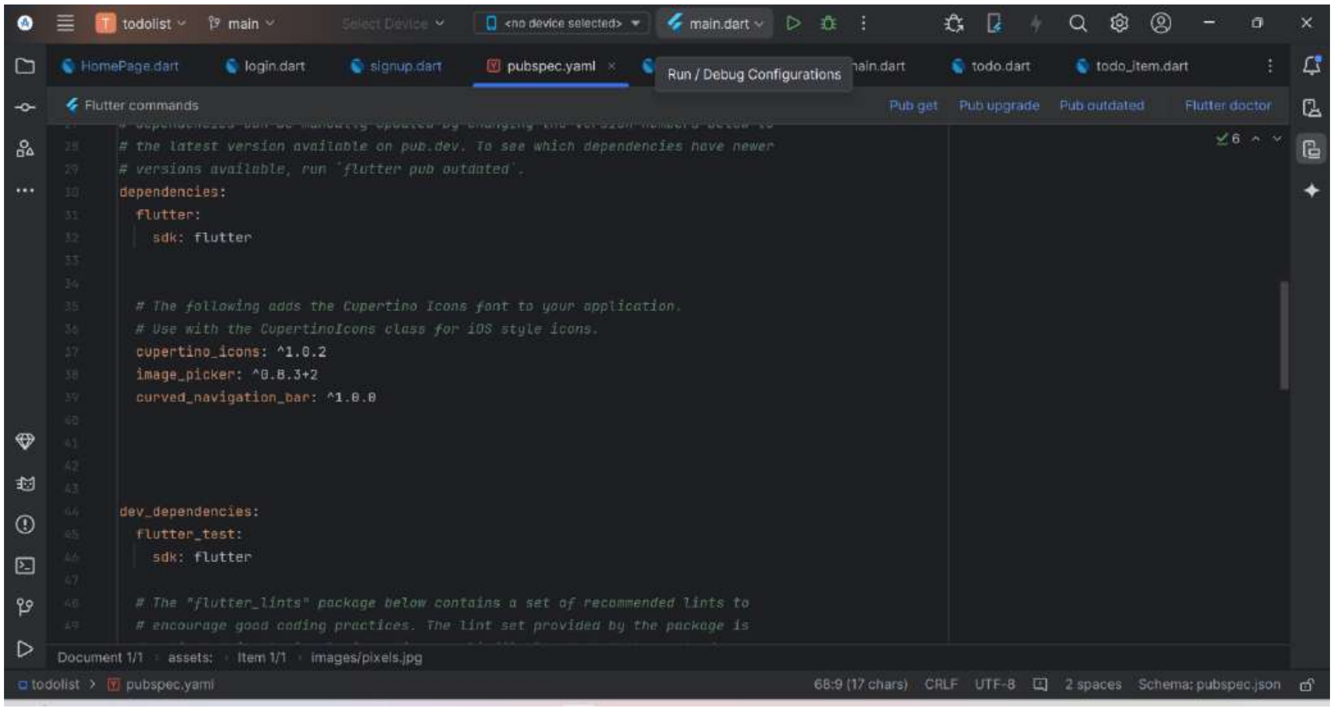
```dart
    _controller = AnimationController(
      duration: const Duration(seconds: 3),
      vsync: this,
    );  // AnimationController

    _animation = CurvedAnimation(
      parent: _controller!,
      curve: Curves.easeInOut,
    );

    _slideAnimation = Tween<Offset>(
      begin: Offset(0.0, 1.0),
      end: Offset(0.0, 0.0),
    ).animate(CurvedAnimation(  // Tween
      parent: _controller!,
      curve: Curves.easeInOut,
    ));

    _controller!.forward();

    Timer(Duration(seconds: 5), () {
      Navigator.pushReplacementNamed(context, '/login');
    });  // Timer
  }
```



```
  @override
  void dispose() {
    _controller!.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Stack(
        fit: StackFit.expand,
        children: [
          Container(
            decoration: BoxDecoration(
              color: Colors.lightBlue.shade300,
            ), // BoxDecoration
            child: Container(
              decoration: BoxDecoration(
                gradient: LinearGradient(
                  colors: [
                    Colors.lightBlue.shade300.withOpacity(0.8),
                    Colors.lightBlue.shade600.withOpacity(0.8),
                  ],
                  begin: Alignment.topLeft,
```

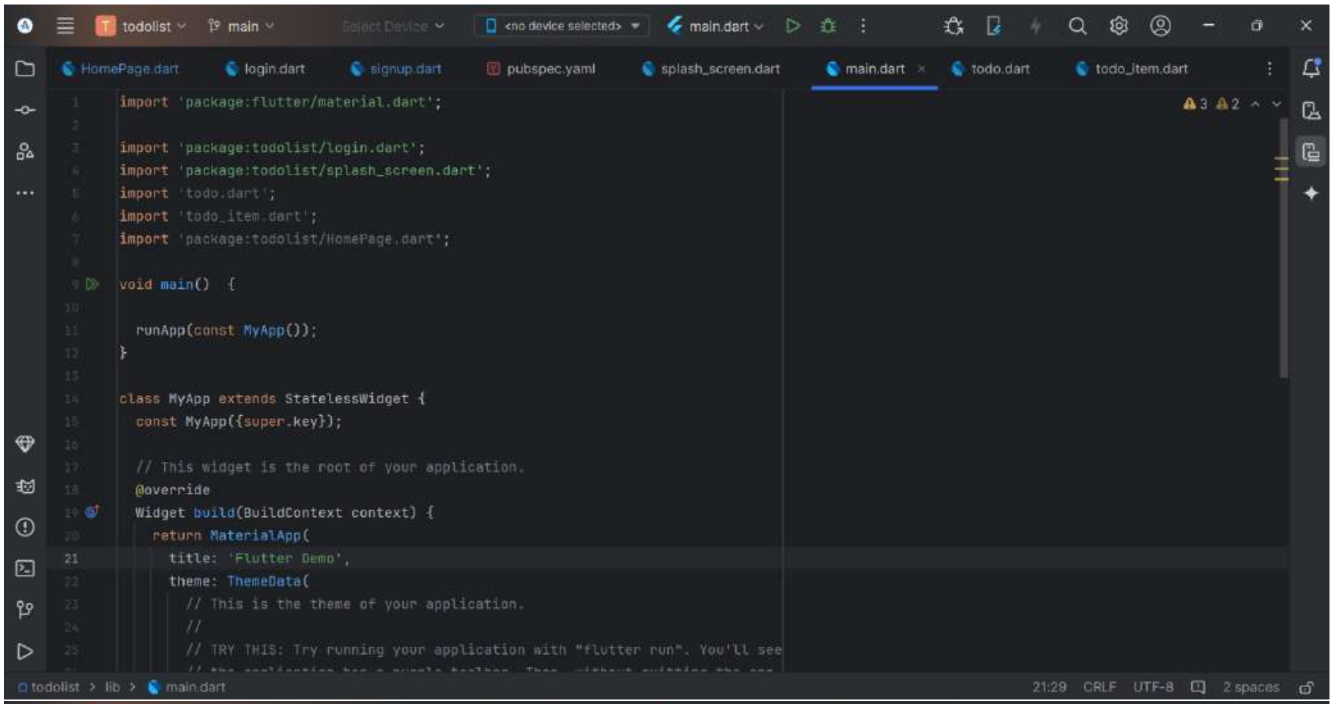
```
import 'package:flutter/material.dart';
import 'login.dart';
import 'dart:async';

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

  @override
  State<SplashScreen> createState() => _SplashScreenState();
}

class _SplashScreenState extends State<SplashScreen>
    with SingleTickerProviderStateMixin {
  AnimationController? _controller;
  Animation<double>? _animation;
  Animation<Offset>? _slideAnimation;

  @override
  void initState() {
    super.initState();

    _controller = AnimationController(
      duration: const Duration(seconds: 3),
      vsync: this,
    );

    _animation = CurvedAnimation(
      parent: _controller!,
      curve: Curves.easeInOut,
    );

    _slideAnimation = Tween<Offset>(
      begin: Offset(0.0, 1.0),
      end: Offset(0.0, 0.0),
    ).animate(CurvedAnimation(
      parent: _controller!,
      curve: Curves.easeInOut,
    ));

    _controller!.forward();
    Timer(Duration(seconds: 5), () {
      Navigator.pushReplacementNamed(context, '/login');
    });
  }

  @override
  void dispose() {
    _controller!.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Stack(
        fit: StackFit.expand,
        children: [
          Container(
            decoration: BoxDecoration(
              color: Colors.lightBlue.shade300,
            ),
            child: Container(
              decoration: BoxDecoration(
                gradient: LinearGradient(
                  colors: [
                    Colors.lightBlue.shade300.withOpacity(0.8),
                    Colors.lightBlue.shade600.withOpacity(0.8),
                  ],
                  begin: Alignment.topLeft,
                  end: Alignment.bottomRight,
                ),
              ),
            ),
          ),
          Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              ScaleTransition(
                scale: _animation!,
                child: CircleAvatar(
                  radius: 60,
                  backgroundColor: Colors.transparent,
                  child: ClipOval(
                    child: Image.network(
                      'https://static.vecteezy.com/system/resources/previews/016/516/991/non_2x/to-do-list-icon-free-vector.jpg',
                      fit: BoxFit.cover,
                      width: 200,
                      height: 200,
                    ),
                  ),
                ),
              ),
              SizedBox(height: 20),
              SlideTransition(
                position: _slideAnimation!,
                child: Text(
                  'Welcome to ToDo List App',
                  style: TextStyle(
                    color: Colors.white,
                    fontSize: 28,
                    fontWeight: FontWeight.bold,
                    shadows: [
                      Shadow(
                        blurRadius: 10.0,
                        color: Colors.black54,
                        offset: Offset(2.0, 2.0),
                      ),
                    ],
                  ),
                ),
              ),
              SizedBox(height: 50),
              CircularProgressIndicator(
                valueColor: AlwaysStoppedAnimation<Color>(Colors.white),
              ),
            ],
          ),
        ],
      ),
    );
   }
  }
```

PUBSEC.YAML:

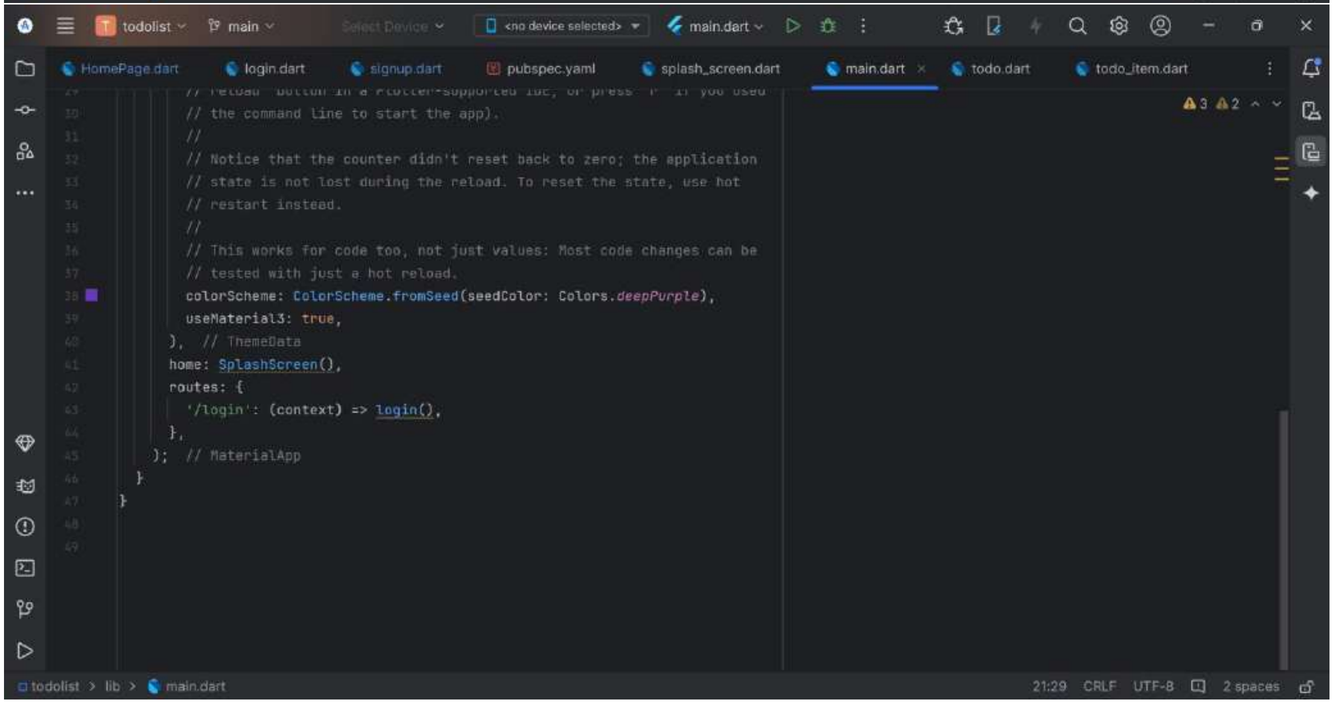


```
# the latest version available on pub.dev. To see which dependencies have newer
# versions available, run `flutter pub outdated`.
dependencies:
  flutter:
    sdk: flutter


  # The following adds the Cupertino Icons font to your application.
  # Use with the CupertinoIcons class for iOS style icons.
  cupertino_icons: ^1.0.2
  image_picker: ^0.8.3+2
  curved_navigation_bar: ^1.0.0




dev_dependencies:
  flutter_test:
    sdk: flutter

  # The "flutter_lints" package below contains a set of recommended lints to
  # encourage good coding practices. The lint set provided by the package is
```

MAIN.DART:

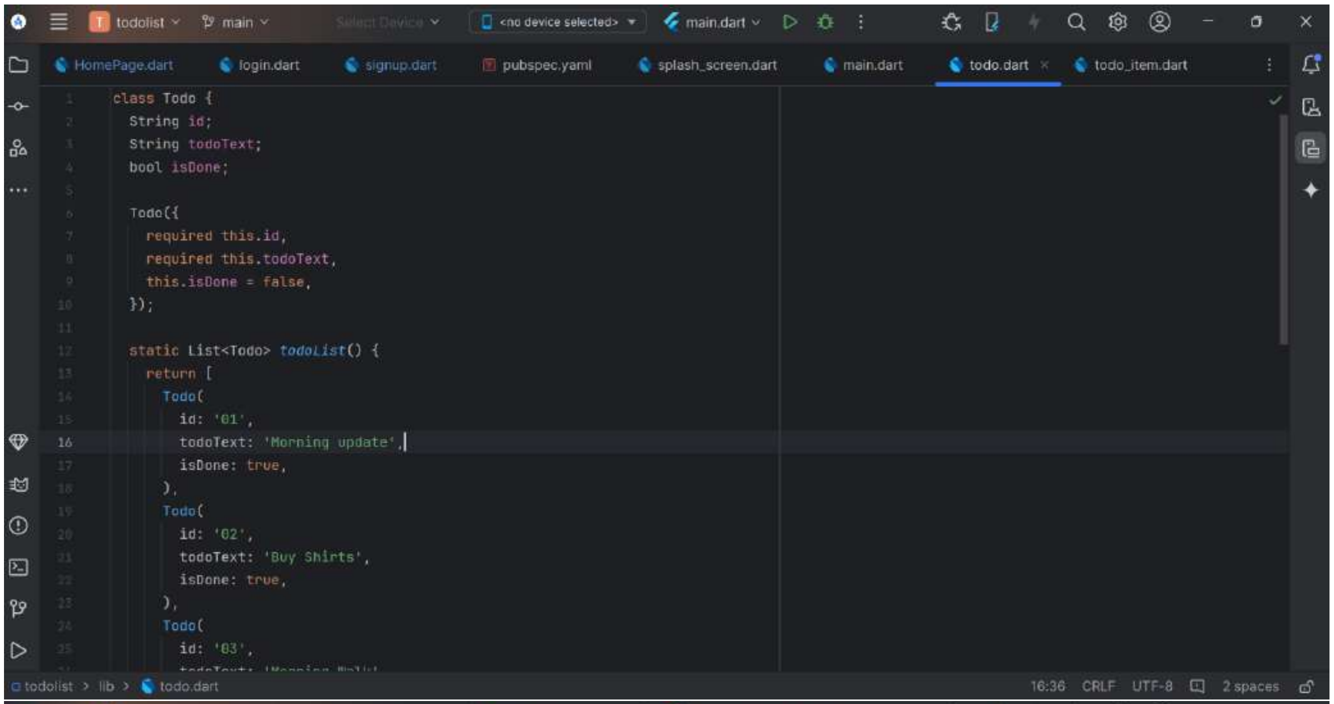


```
import 'package:flutter/material.dart';

import 'package:todolist/login.dart';
import 'package:todolist/splash_screen.dart';
import 'todo.dart';
import 'todo_item.dart';
import 'package:todolist/HomePage.dart';

void main()  {

  runApp(const MyApp());
}

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

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        // This is the theme of your application.
        //
        // TRY THIS: Try running your application with "flutter run". You'll see
        // the application has a purple toolbar. Then, without quitting the app
```

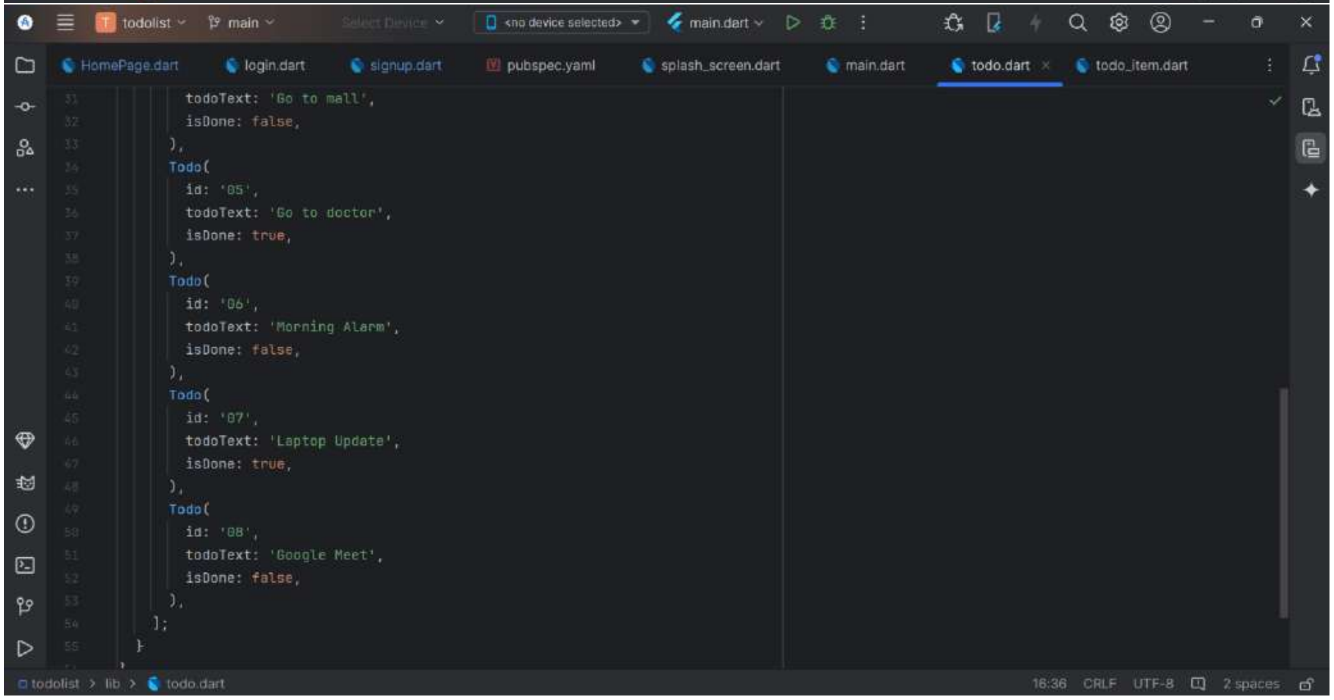


```
        // reload" button in a Flutter-supported IDE, or press "r" if you used
        // the command line to start the app).
        //
        // Notice that the counter didn't reset back to zero; the application
        // state is not lost during the reload. To reset the state, use hot
        // restart instead.
        //
        // This works for code too, not just values: Most code changes can be
        // tested with just a hot reload.
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,
      ), // ThemeData
      home: SplashScreen(),
      routes: {
        '/login': (context) => login(),
      },
    ); // MaterialApp
  }
}
```

TODO.DART:

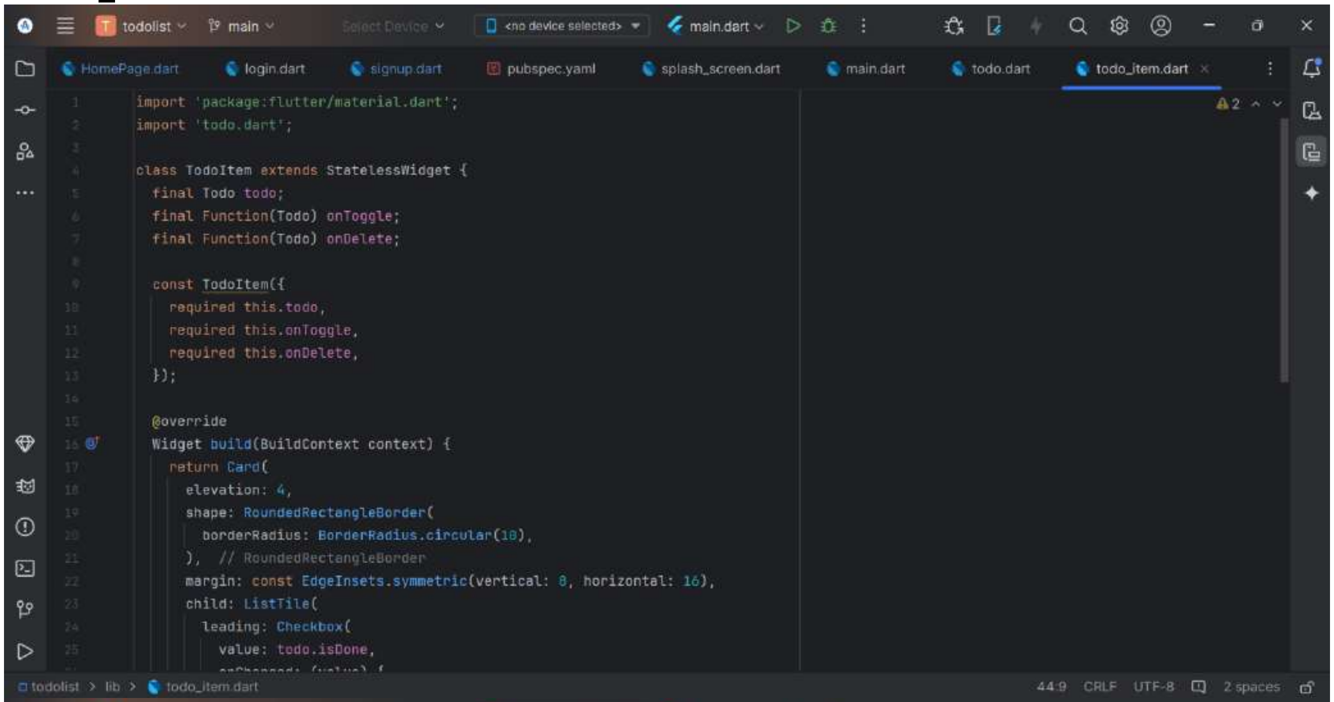


```
class Todo {
  String id;
  String todoText;
  bool isDone;

  Todo({
    required this.id,
    required this.todoText,
    this.isDone = false,
  });

  static List<Todo> todoList() {
    return [
      Todo(
        id: '01',
        todoText: 'Morning update',
        isDone: true,
      ),
      Todo(
        id: '02',
        todoText: 'Buy Shirts',
        isDone: true,
      ),
      Todo(
        id: '03',
```

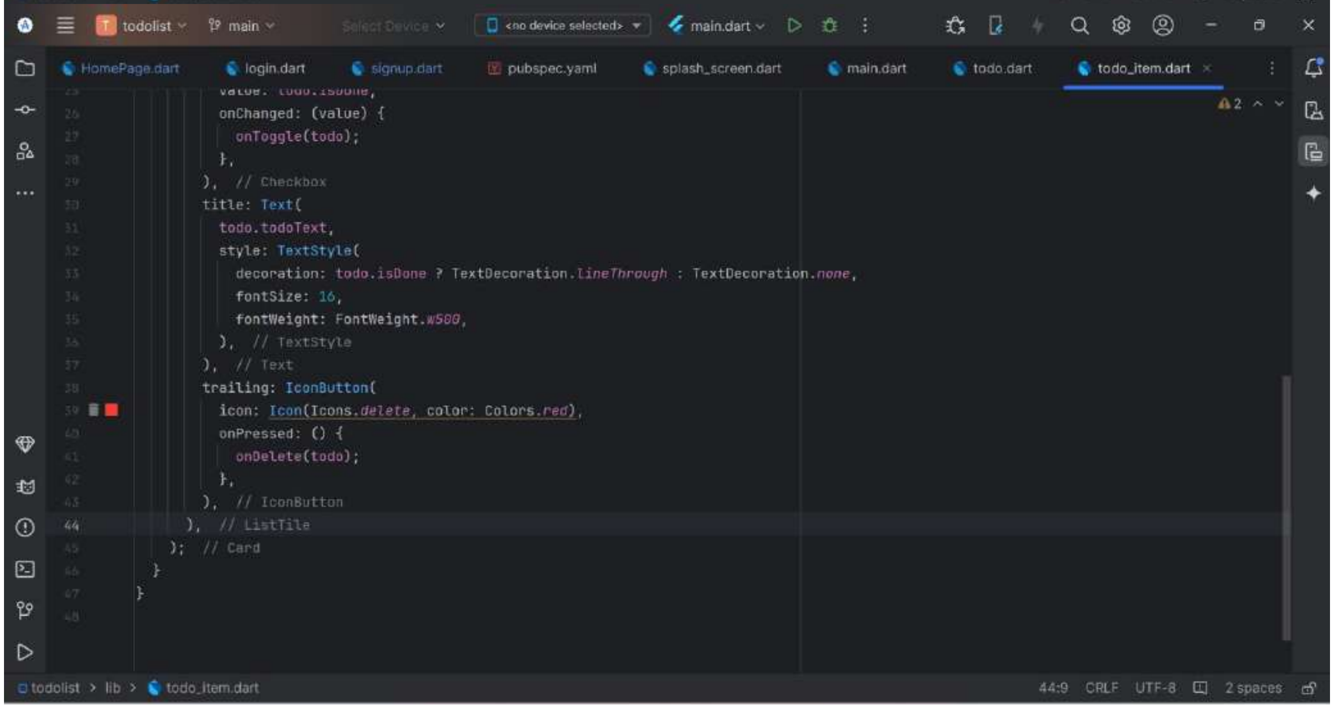


```
        todoText: 'Go to mall',
        isDone: false,
      ),
      Todo(
        id: '05',
        todoText: 'Go to doctor',
        isDone: true,
      ),
      Todo(
        id: '06',
        todoText: 'Morning Alarm',
        isDone: false,
      ),
      Todo(
        id: '07',
        todoText: 'Laptop Update',
        isDone: true,
      ),
      Todo(
        id: '08',
        todoText: 'Google Meet',
        isDone: false,
      ),
    ];
  }
```

TODO_ITEM.DART:



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

class TodoItem extends StatelessWidget {
  final Todo todo;
  final Function(Todo) onToggle;
  final Function(Todo) onDelete;

  const TodoItem({
    required this.todo,
    required this.onToggle,
    required this.onDelete,
  });

  @override
  Widget build(BuildContext context) {
    return Card(
      elevation: 4,
      shape: RoundedRectangleBorder(
        borderRadius: BorderRadius.circular(10),
      ), // RoundedRectangleBorder
      margin: const EdgeInsets.symmetric(vertical: 8, horizontal: 16),
      child: ListTile(
        leading: Checkbox(
          value: todo.isDone,
```



```dart
          onChanged: (value) {
            onToggle(todo);
          },
        ), // Checkbox
        title: Text(
          todo.todoText,
          style: TextStyle(
            decoration: todo.isDone ? TextDecoration.lineThrough : TextDecoration.none,
            fontSize: 16,
            fontWeight: FontWeight.w500,
          ), // TextStyle
        ), // Text
        trailing: IconButton(
          icon: Icon(Icons.delete, color: Colors.red),
          onPressed: () {
            onDelete(todo);
          },
        ), // IconButton
      ), // ListTile
    ); // Card
  }
}
```

6. Bibliography or References

The following bibliography provides a comprehensive list of sources, tools, and resources that were instrumental in the development of the To-Do List app. This section acknowledges the contributions from various authors, platforms, and educational resources that supported the creation and completion of the project.

1. Flutter Documentation

- Flutter. (n.d.). *Flutter documentation*. Retrieved from https://flutter.dev/docs
  - The official Flutter documentation was extensively used for understanding Flutter's widgets, state management, and navigation principles.

2. Dart Programming Language

- Dart. (n.d.). *Dart language*. Retrieved from https://dart.dev/guides
  - The Dart documentation provided insights into the programming language used for Flutter development, including syntax, functions, and object-oriented concepts.

3. Firebase Documentation

- Firebase. (n.d.). *Firebase documentation*. Retrieved from https://firebase.google.com/docs
  - Firebase documentation was essential for integrating user authentication and other backend services into the To-Do List app.

4. API Integration

- Pub.dev. (n.d.). *Pub.dev - The Dart package repository*. Retrieved from https://pub.dev
  - Resources from Pub.dev were used for integrating APIs, including map APIs and other third-party services.

5. Flutter Packages

- Flutter Packages. (n.d.). *Flutter packages*. Retrieved from https://pub.dev/packages
  - The Flutter package repository provided access to various libraries and tools that were incorporated into the project.

6. Tutorials and Guides

- YouTube. (2023). *Flutter Tutorials*. Retrieved from https://www.youtube.com
  - Various YouTube tutorials were consulted for practical demonstrations and troubleshooting during the development process.

7. Forums and Community Discussions

- Stack Overflow. (n.d.). *Stack Overflow - Questions and Answers*. Retrieved from https://stackoverflow.com
  - Community discussions and solutions on Stack Overflow were instrumental in addressing specific issues and gaining insights from experienced developers.

8. Course Material

- CipherSchools. (2024). *Flutter Development Course*. Retrieved from https://cipherschools.com
  - The course material provided foundational knowledge and practical assignments that guided the development of the To-Do List app.

9. University Resources

- Lovely Professional University (LPU). (2024). *Summer Training Program*. Retrieved from https://www.lpu.in
  - The resources and support provided by LPU were crucial for the successful completion of the project.

This bibliography acknowledges the various sources and resources that contributed to the successful development and completion of the To-Do List app project.

****