Experiment	8
Aim	Implement user authentication and registration using database with flutter
Objective	● Create Registration page ● Create database ● Store, fetch, delete values from database. ● User Authentication from database
Name	**Prabhat Anand Tiwari**
UCID	**2024510066**
Class	**FYMCA**
Batch	**C**
Date of Submission	**17/04/2025**

Technology used	**Flutter**
Task	Create a basic registration page and store data n database. Fetch and also delete few entries from database. You can use either firebase database or postgresql.
Code with proper label	**main.dart**

```dart
import
'package:firebase_core/firebase_core.dart';
// import 'package:flutter/foundation.dart';
import 'package:flutter/material.dart';
import
'package:prac8/screens/splash_screen.dart';
import
'package:prac8/services/theme_service.dart';
import 'package:provider/provider.dart';

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp();
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);
  @override
  Widget build(BuildContext context) {
    return ChangeNotifierProvider(
      create: (context) => ThemeService(),
      builder: (context, _) {
        final themeService =
Provider.of<ThemeService>(context);
        return MaterialApp(
          title: 'Todo App',
          debugShowCheckedModeBanner: false,
          themeMode:
              themeService.isDarkModeOn ?
ThemeMode.dark : ThemeMode.light,
          theme: ThemeData(
            primarySwatch: Colors.teal,
            brightness: Brightness.light,
            appBarTheme: const AppBarTheme(
              elevation: 0,
            ),
          ),
          darkTheme: ThemeData(
            primarySwatch: Colors.teal,
            brightness: Brightness.dark,
            appBarTheme: const AppBarTheme(
              elevation: 0,
            ),
          ),
          home: const SplashScreen(),
        );
      },
    );
  }
```

```
}
```

todo widget

```
import 'package:flutter/material.dart';
import
'package:prac8/models/todo_model.dart';

class TodoItem extends StatelessWidget {
  final TodoModel todo;
  final VoidCallback onTap;
  final VoidCallback onToggleComplete;
  final VoidCallback onDelete;

  const TodoItem({
    Key? key,
    required this.todo,
    required this.onTap,
    required this.onToggleComplete,
    required this.onDelete,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Card(
      elevation: 2,
      margin: const
EdgeInsets.symmetric(vertical: 8),
      child: ListTile(
        contentPadding: const
EdgeInsets.symmetric(horizontal: 16,
vertical: 8),
        leading: Checkbox(
          value: todo.isCompleted,
          onChanged: (_) =>
onToggleComplete(),
        ),
        title: Text(
          todo.title,
          style: TextStyle(
            fontWeight: FontWeight.bold,
            decoration: todo.isCompleted ?
TextDecoration.lineThrough : null,
          ),
        ),
        subtitle: Column(
          crossAxisAlignment:
CrossAxisAlignment.start,
          children: [
            Text(
              todo.description.length > 50
                  ?
'${todo.description.substring(0, 50)}...'
                  : todo.description,
              style: TextStyle(
                decoration:
                    todo.isCompleted ?
TextDecoration.lineThrough : null,
              ),
            ),
            const SizedBox(height: 4),
            Row(
              children: [
                if (todo.dueDate != null)
...[
                  const
Icon(Icons.calendar_today, size: 12),
                  const SizedBox(width: 4),
                  Text(

'${todo.dueDate!.day}/${todo.dueDate!.month}
/${todo.dueDate!.year}',
                    style: const
TextStyle(fontSize: 12),
                  ),
                  const SizedBox(width: 8),
                ],
                const Icon(Icons.flag, size:
12),
                const SizedBox(width: 4),
                Text(
                  todo.priority ?? 'Medium',
                  style: const
TextStyle(fontSize: 12),
                ),
              ],
            ),
          ],
        ),
        trailing: IconButton(
          icon: const Icon(Icons.delete),
          onPressed: () {
            showDialog(
              context: context,
              builder: (context) =>
AlertDialog(
                title: const Text('Delete
Todo'),
                content:
                    const Text('Are you sure
you want to delete this todo?'),
                actions: [
                  TextButton(
                    onPressed: () =>
Navigator.of(context).pop(),
                    child: const
Text('Cancel'),
                  ),
                  TextButton(
                    onPressed: () {

Navigator.of(context).pop();
                      onDelete();
                    },
                    child: const
Text('Delete'),
                  ),
                ],
              ),
            );
          },
        ),
        onTap: onTap,
      ),
    );
  }
}
```

add todo page

```
import 'package:flutter/material.dart';
import
'package:prac8/models/todo_model.dart';
import
```

```
'package:prac8/services/auth_service.dart';
import
'package:prac8/services/database_service.dar
t';

class AddTodoPage extends StatefulWidget {
  final TodoModel? todo;

  const AddTodoPage({Key? key, this.todo}) :
super(key: key);

  @override
  State<AddTodoPage> createState() =>
_AddTodoPageState();
}

class _AddTodoPageState extends
State<AddTodoPage> {
  final _formKey = GlobalKey<FormState>();
  final _titleController =
TextEditingController();
  final _descriptionController =
TextEditingController();
  final AuthService _authService =
AuthService();
  final DatabaseService _databaseService =
DatabaseService();
  DateTime? _dueDate;
  String _priority = 'Medium';
  bool _isLoading = false;

  @override
  void initState() {
    super.initState();
    if (widget.todo != null) {
      _titleController.text =
widget.todo!.title;
      _descriptionController.text =
widget.todo!.description;
      _dueDate = widget.todo!.dueDate;
      _priority = widget.todo!.priority ??
'Medium';
    }
  }

  @override
  void dispose() {
    _titleController.dispose();
    _descriptionController.dispose();
    super.dispose();
  }

  Future<void> _selectDueDate() async {
    final DateTime? picked = await
showDatePicker(
      context: context,
      initialDate: _dueDate ??
DateTime.now(),
      firstDate: DateTime.now(),
      lastDate: DateTime(2100),
    );
    if (picked != null && picked !=
_dueDate) {
      setState(() {
        _dueDate = picked;
      });
    }
  }

  void _saveTodo() async {
    if (_formKey.currentState!.validate()) {
      setState(() {
        _isLoading = true;
      });

      final user =
_authService.getCurrentUser();
      if (user == null) {

ScaffoldMessenger.of(context).showSnackBar(
          const SnackBar(content: Text('User
not logged in')),
        );
        setState(() {
          _isLoading = false;
        });
        return;
      }

      try {
        final todo = TodoModel(
          id: widget.todo?.id,
          userId: user.uid,
          title:
_titleController.text.trim(),
          description:
_descriptionController.text.trim(),
          isCompleted:
widget.todo?.isCompleted ?? false,
          createdAt: widget.todo?.createdAt
?? DateTime.now(),
          dueDate: _dueDate,
          priority: _priority,
        );

        if (widget.todo == null) {
          await
_databaseService.addTodo(todo);
        } else {
          await
_databaseService.updateTodo(widget.todo!.id!
, todo);
        }

        if (mounted) {
          Navigator.of(context).pop();
        }
      } catch (e) {

ScaffoldMessenger.of(context).showSnackBar(
          SnackBar(content: Text('Error:
${e.toString()}')),
        );
      } finally {
        if (mounted) {
          setState(() {
            _isLoading = false;
          });
        }
      }
    }
  }
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(widget.todo == null ?
'Add Todo' : 'Edit Todo'),
      ),
      body: SingleChildScrollView(
        padding: const EdgeInsets.all(20.0),
        child: Form(
          key: _formKey,
          child: Column(
            crossAxisAlignment:
CrossAxisAlignment.stretch,
            children: [
              TextFormField(
                controller:
_titleController,
                decoration: const
InputDecoration(
                  labelText: 'Title',
                  border:
OutlineInputBorder(),
                ),
                validator: (value) {
                  if (value == null ||
value.isEmpty) {
                    return 'Please enter a
title';
                  }
                  return null;
                },
              ),
              const SizedBox(height: 20),
              TextFormField(
                controller:
_descriptionController,
                decoration: const
InputDecoration(
                  labelText: 'Description',
                  border:
OutlineInputBorder(),
                ),
                maxLines: 5,
                validator: (value) {
                  if (value == null ||
value.isEmpty) {
                    return 'Please enter a
description';
                  }
                  return null;
                },
              ),
              const SizedBox(height: 20),
              Row(
                children: [
                  Expanded(
                    child: InkWell(
                      onTap: _selectDueDate,
                      child: InputDecorator(
                        decoration: const
InputDecoration(
                          labelText: 'Due
Date',
                          border:
```

```
OutlineInputBorder(),
                        ),
                        child: Text(
                          _dueDate == null
                              ? 'Select Due
Date'
                              :
'${_dueDate!.day}/${_dueDate!.month}/${_dueD
ate!.year}',
                        ),
                      ),
                    ),
                  ),
                  const SizedBox(width: 10),
                  IconButton(
                    icon: const
Icon(Icons.calendar_today),
                    onPressed:
_selectDueDate,
                  ),
                ],
              ),
              const SizedBox(height: 20),

DropdownButtonFormField<String>(
                decoration: const
InputDecoration(
                  labelText: 'Priority',
                  border:
OutlineInputBorder(),
                ),
                value: _priority,
                items: ['Low', 'Medium',
'High'].map((String priority) {
                  return
DropdownMenuItem<String>(
                    value: priority,
                    child: Text(priority),
                  );
                }).toList(),
                onChanged: (String?
newValue) {
                  if (newValue != null) {
                    setState(() {
                      _priority = newValue;
                    });
                  }
                },
              ),
              const SizedBox(height: 30),
              ElevatedButton(
                onPressed: _isLoading ? null
: _saveTodo,
                style:
ElevatedButton.styleFrom(
                  padding: const
EdgeInsets.symmetric(vertical: 15),
                ),
                child: _isLoading
                    ? const
CircularProgressIndicator(color:
Colors.white)
                    : Text(widget.todo ==
null ? 'Add Todo' : 'Update Todo'),
              ),
            ],
          ),
        ),
      ),
    );
  }
}
```

home page

```
import
'package:cloud_firestore/cloud_firestore.dar
t';
import 'package:flutter/material.dart';
import 'package:prac8/auth/login_page.dart';
import
'package:prac8/models/todo_model.dart';
import
'package:prac8/screens/add_todo_page.dart';
import
'package:prac8/screens/todo_detail_page.dart
';
import
'package:prac8/services/auth_service.dart';
import
'package:prac8/services/database_service.dar
t';
import
'package:prac8/services/theme_service.dart';
import
'package:prac8/widgets/todo_item.dart';
import 'package:provider/provider.dart';

class HomePage extends StatefulWidget {
  const HomePage({Key? key}) : super(key:
```

```
key);

  @override
  State<HomePage> createState() =>
_HomePageState();
}

class _HomePageState extends State<HomePage>
{
  final AuthService _authService =
AuthService();
  final DatabaseService _databaseService =
DatabaseService();
  final TextEditingController
_searchController = TextEditingController();
  bool _isSearching = false;
  String _searchQuery = '';

  @override
  void dispose() {
    _searchController.dispose();
    super.dispose();
  }

  void _signOut() async {
    await _authService.signOut();
    Navigator.of(context).pushReplacement(
      MaterialPageRoute(builder: (_) =>
const LoginPage()),
    );
  }

  void _navigateToAddTodo() {
    Navigator.of(context).push(
      MaterialPageRoute(builder: (_) =>
const AddTodoPage()),
    );
  }

  void _toggleSearch() {
    setState(() {
      _isSearching = !_isSearching;
      if (!_isSearching) {
        _searchController.clear();
        _searchQuery = '';
      }
    });
  }

  @override
  Widget build(BuildContext context) {
    final user =
_authService.getCurrentUser();
    final themeService =
Provider.of<ThemeService>(context);

    return Scaffold(
      appBar: AppBar(
        title: _isSearching
            ? TextField(
                controller:
_searchController,
                decoration: const
InputDecoration(
                  hintText: 'Search
todos...',
                  border: InputBorder.none,
                ),
                style: const
TextStyle(color: Colors.white),
                onChanged: (value) {
                  setState(() {
                    _searchQuery = value;
                  });
                },
              )
            : const Text('My Todos'),
        actions: [
          IconButton(
            icon: Icon(_isSearching ?
Icons.close : Icons.search),
            onPressed: _toggleSearch,
          ),
          IconButton(
            icon: Icon(
              themeService.isDarkModeOn ?
Icons.light_mode : Icons.dark_mode,
            ),
            onPressed: () {
              themeService.toggleTheme();
            },
          ),
          IconButton(
            icon: const Icon(Icons.logout),
            onPressed: _signOut,
          ),
        ],
      ),
      body: user == null
```

```
          ? const Center(child: Text('No
user logged in'))
          : StreamBuilder<QuerySnapshot>(
              stream: _searchQuery.isEmpty
                  ?
_databaseService.getTodos(user.uid)
                  :
_databaseService.searchTodos(user.uid,
_searchQuery),
              builder: (context, snapshot) {
                if (snapshot.connectionState
== ConnectionState.waiting) {
                  return const Center(child:
CircularProgressIndicator());
                }

                if (snapshot.hasError) {
                  return Center(child:
Text('Error: ${snapshot.error}'));
                }

                if (!snapshot.hasData ||
snapshot.data!.docs.isEmpty) {
                  return Center(
                    child: Column(
                      mainAxisAlignment:
MainAxisAlignment.center,
                      children: [
                        const Icon(

Icons.note_alt_outlined,
                          size: 80,
                          color:
```

```
Colors.grey,
                        ),
                        const
SizedBox(height: 20),
                        Text(

_searchQuery.isEmpty
                              ? 'No todos
yet. Add one!'
                              : 'No todos
found for "$_searchQuery"',
                          style: const
TextStyle(
                            fontSize: 18,
                            color:
Colors.grey,
                          ),
                        ),
                      ],
                    ),
                  );
                }

                final todos =
snapshot.data!.docs.map((doc) {
                  return TodoModel.fromMap(
                    doc.data() as
Map<String, dynamic>,
                    doc.id,
                  );
                }).toList();

                return ListView.builder(
                  padding: const
EdgeInsets.all(8),
                  itemCount: todos.length,
                  itemBuilder: (context,
index) {
                    final todo =
todos[index];
                    return TodoItem(
                      todo: todo,
                      onTap: () {

Navigator.of(context).push(
                          MaterialPageRoute(
                            builder: (_) =>
TodoDetailPage(todo: todo),
                          ),
                        );
                      },
                      onToggleComplete: ()
async {
                        await
_databaseService.updateTodo(
                          todo.id!,

todo.copyWith(isCompleted:
!todo.isCompleted),
                        );
                      },
                      onDelete: () async {
                        await
_databaseService.deleteTodo(todo.id!);
                      },
                    );
```

```
                      },
                    );
                  },
                ),
      floatingActionButton:
FloatingActionButton(
        onPressed: _navigateToAddTodo,
        child: const Icon(Icons.add),
      ),
    );
  }
}
```

auth service

```
import
'package:firebase_auth/firebase_auth.dart';

class AuthService {
  final FirebaseAuth _auth =
FirebaseAuth.instance;

  // Check if user is logged in
  bool isUserLoggedIn() {
    return _auth.currentUser != null;
  }

  // Get current user
  User? getCurrentUser() {
    return _auth.currentUser;
  }

  // Register with email and password
  Future<User?>
```

```
registerWithEmailAndPassword(
      String email, String password) async {
    try {
      UserCredential result = await
_auth.createUserWithEmailAndPassword(
        email: email,
        password: password,
      );
      return result.user;
    } catch (e) {
      print(e.toString());
      return null;
    }
  }

  // Sign in with email and password
  Future<User?> signInWithEmailAndPassword(
      String email, String password) async {
    try {
      UserCredential result = await
_auth.signInWithEmailAndPassword(
        email: email,
        password: password,
      );
      return result.user;
    } catch (e) {
      print(e.toString());
      return null;
    }
  }

  // Sign out
  Future<void> signOut() async {
    try {
      return await _auth.signOut();
    } catch (e) {
      print(e.toString());
    }
  }
}
```

database service

```
import
'package:cloud_firestore/cloud_firestore.dar
t';
import
'package:prac8/models/todo_model.dart';

class DatabaseService {
  final FirebaseFirestore _firestore =
FirebaseFirestore.instance;

  // Create user profile
  Future<void> createUserProfile(String uid,
String name, String email) async {
    await
_firestore.collection('users').doc(uid).set(
{
      'name': name,
      'email': email,
      'createdAt': Timestamp.now(),
    });
  }

  // Get user profile
  Future<DocumentSnapshot>
```

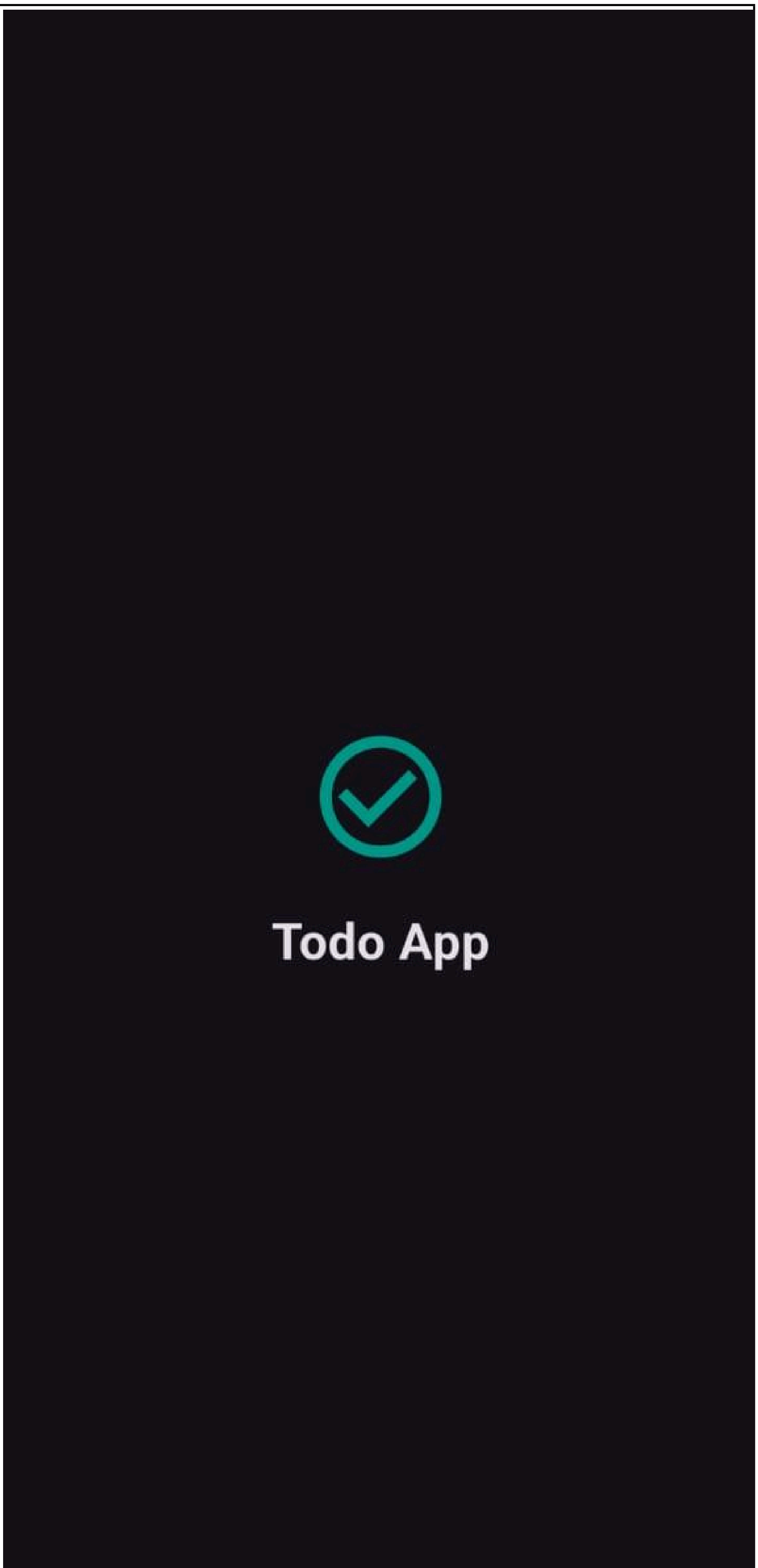
```
getUserProfile(String uid) async {
    return await
_firestore.collection('users').doc(uid).get(
);
  }

  // Add a new todo
  Future<void> addTodo(TodoModel todo) async
{
    await
_firestore.collection('todos').add(todo.toMa
p());
  }

  // Update a todo
  Future<void> updateTodo(String id,
TodoModel todo) async {
    await
_firestore.collection('todos').doc(id).updat
e(todo.toMap());
  }

  // Delete a todo
  Future<void> deleteTodo(String id) async {
    await
_firestore.collection('todos').doc(id).delet
e();
  }

  // Get todos for a specific user
  Stream<QuerySnapshot> getTodos(String uid)
{
    return _firestore
```

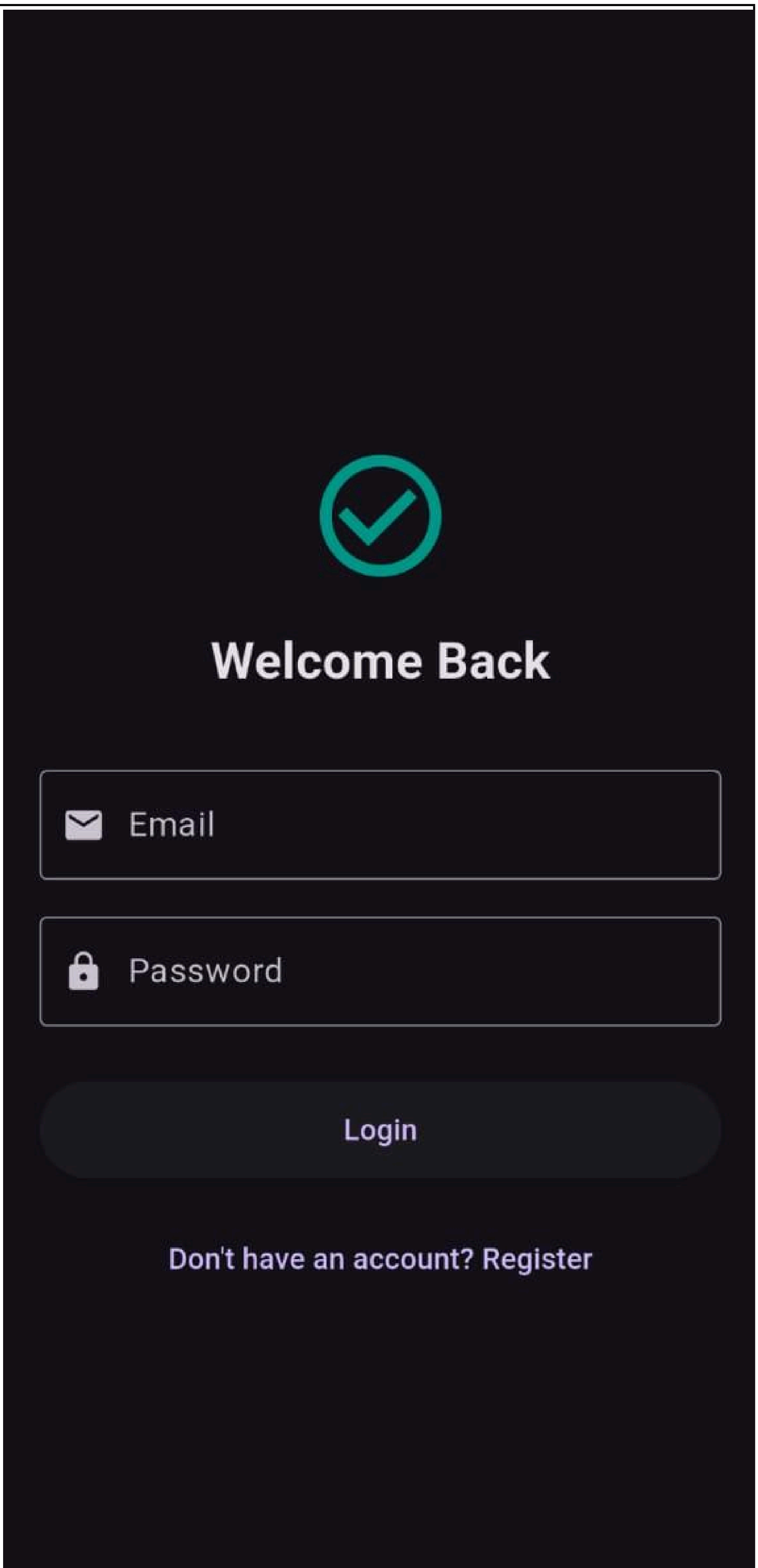
```
        .collection('todos')
        .where('userId', isEqualTo: uid)
        .orderBy('createdAt', descending:
true)
        .snapshots();
  }

  // Search todos
  Stream<QuerySnapshot> searchTodos(String
uid, String query) {
    return _firestore
        .collection('todos')
        .where('userId', isEqualTo: uid)
        .where('title',
isGreaterThanOrEqualTo: query)
        .where('title', isLessThanOrEqualTo:
query + '\uf8ff')
        .snapshots();
  }
}
```

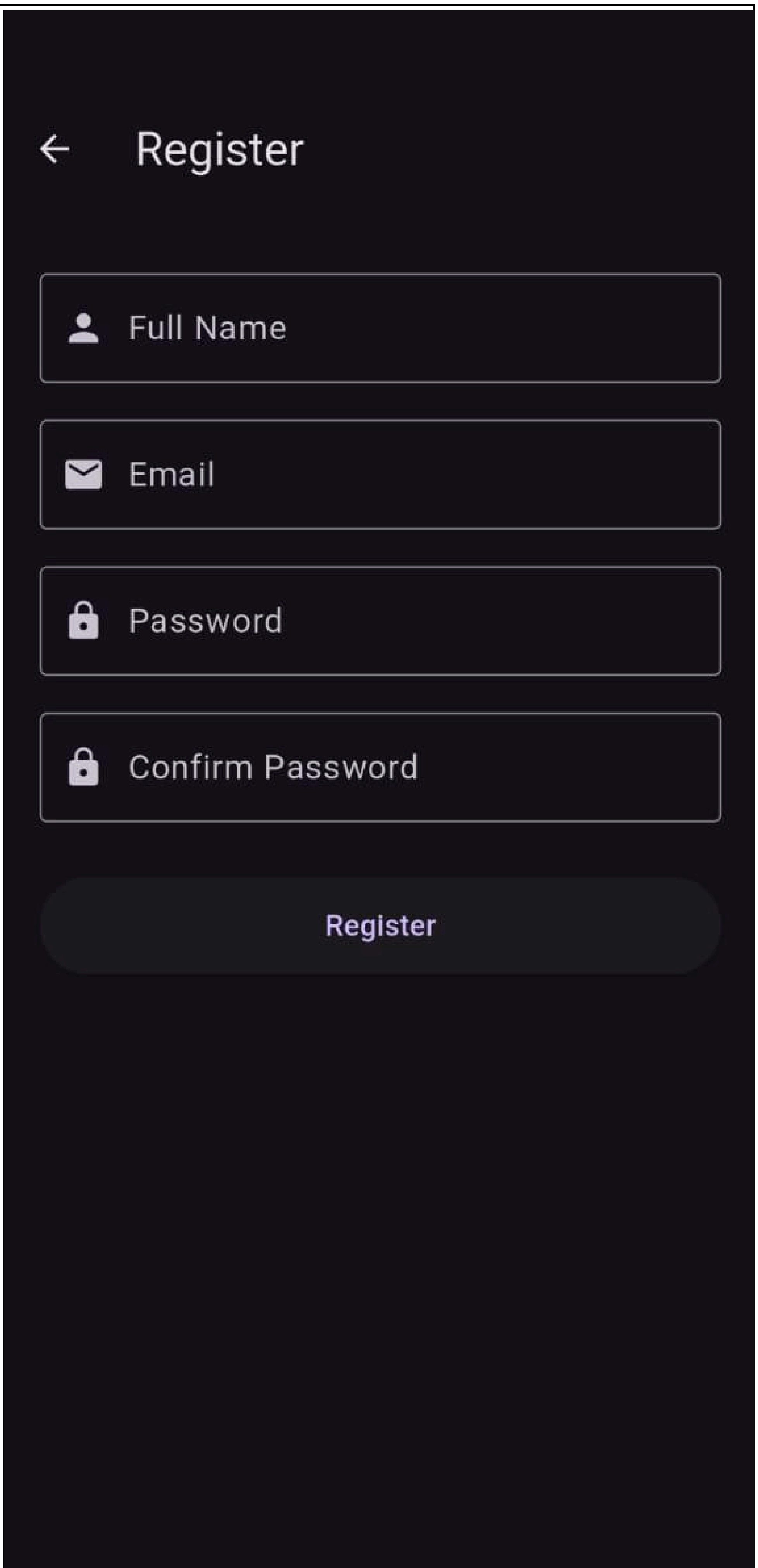
theme service

```
import 'package:flutter/material.dart';
import
'package:shared_preferences/shared_preferenc
es.dart';

class ThemeService extends ChangeNotifier {
  final String _themeKey = 'isDarkMode';
  bool _isDarkModeOn = false;
  bool get isDarkModeOn => _isDarkModeOn;

  ThemeService() {
    _loadThemeFromPrefs();
  }

  void _loadThemeFromPrefs() async {
    final prefs = await
SharedPreferences.getInstance();
    _isDarkModeOn = prefs.getBool(_themeKey)
?? false;
    notifyListeners();
  }

  void toggleTheme() async {
    _isDarkModeOn = !_isDarkModeOn;
    final prefs = await
SharedPreferences.getInstance();
    await prefs.setBool(_themeKey,
_isDarkModeOn);
    notifyListeners();
  }
}
```

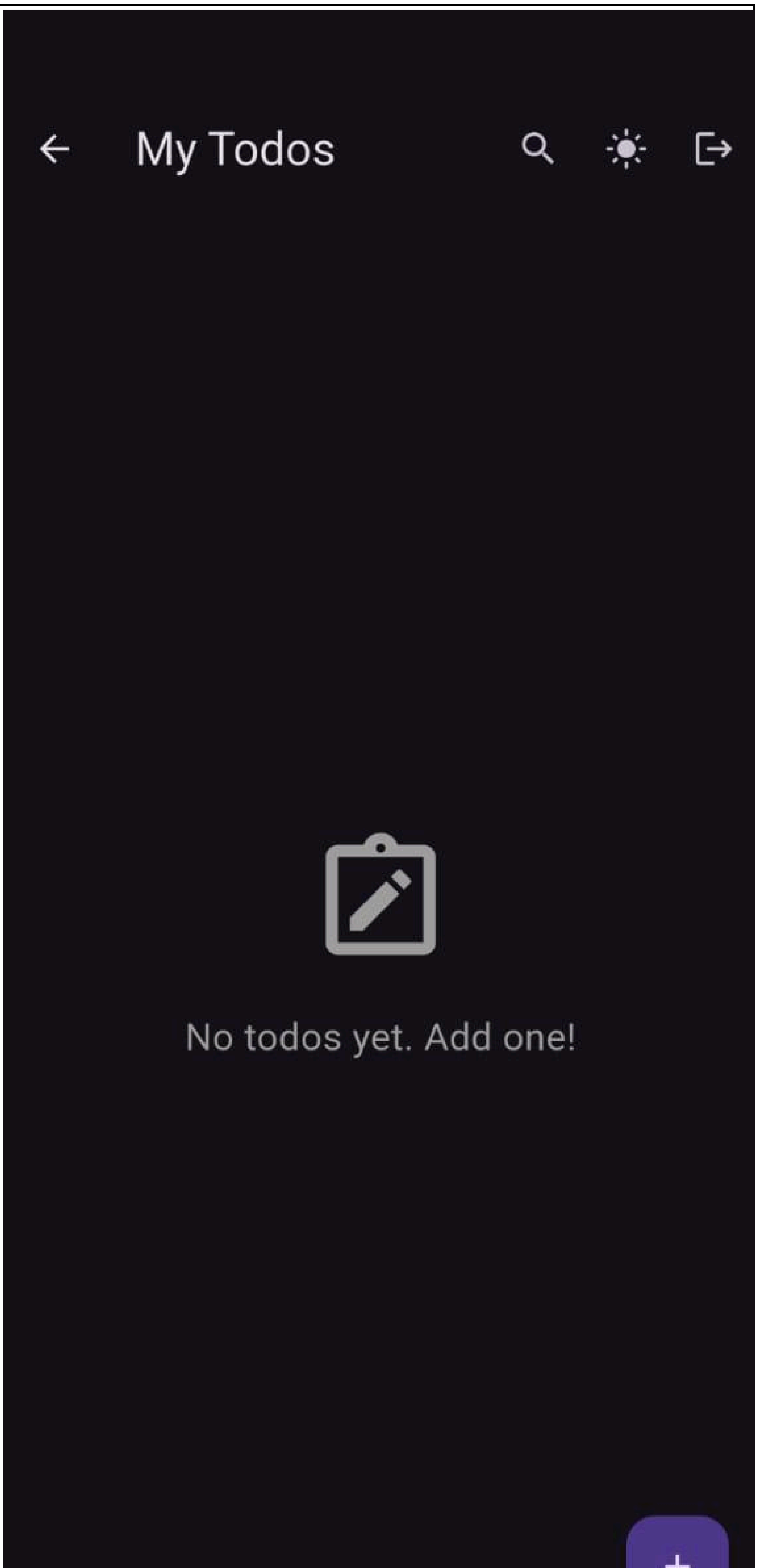
Screenshots	

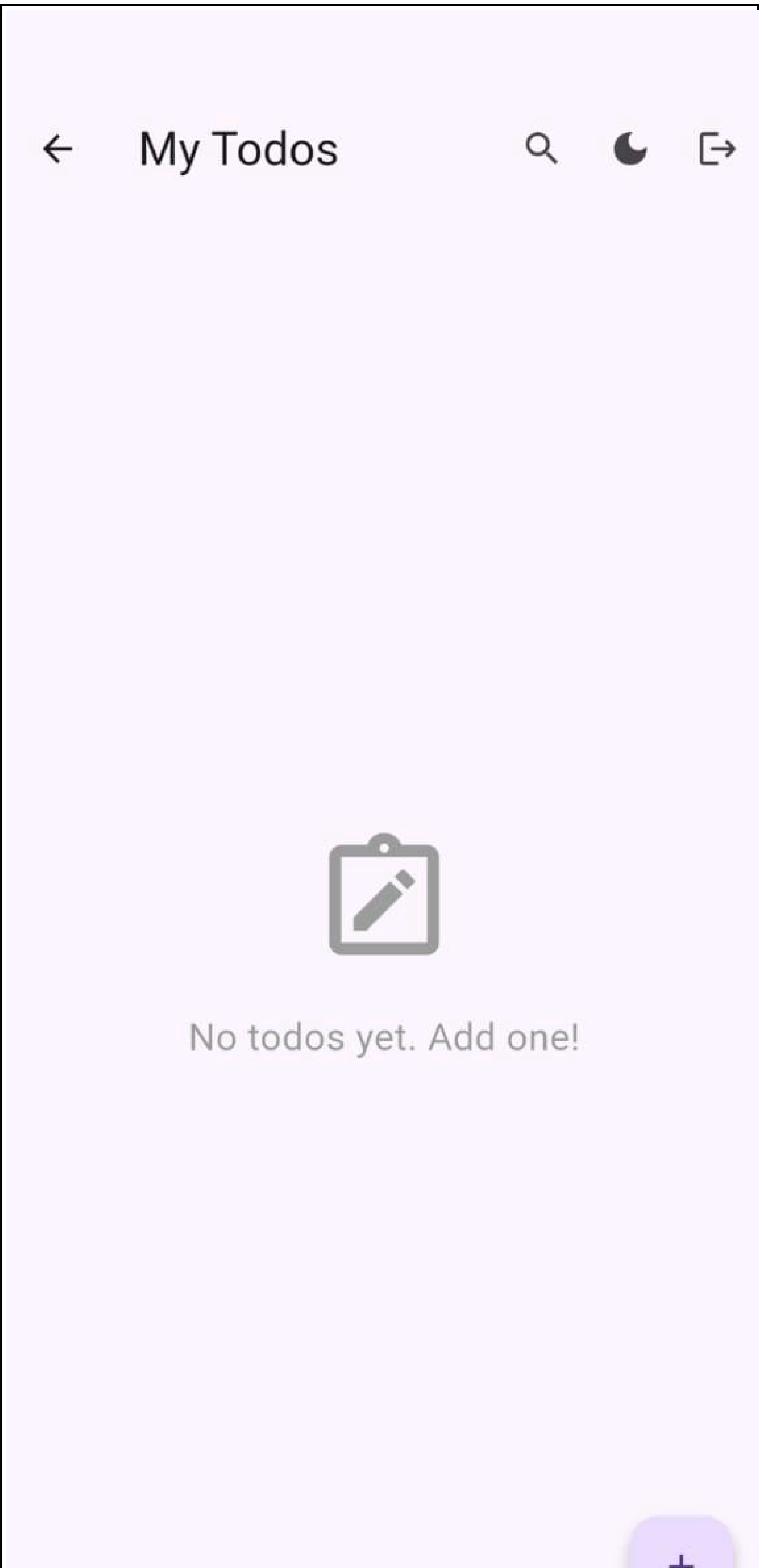
	 Todo App

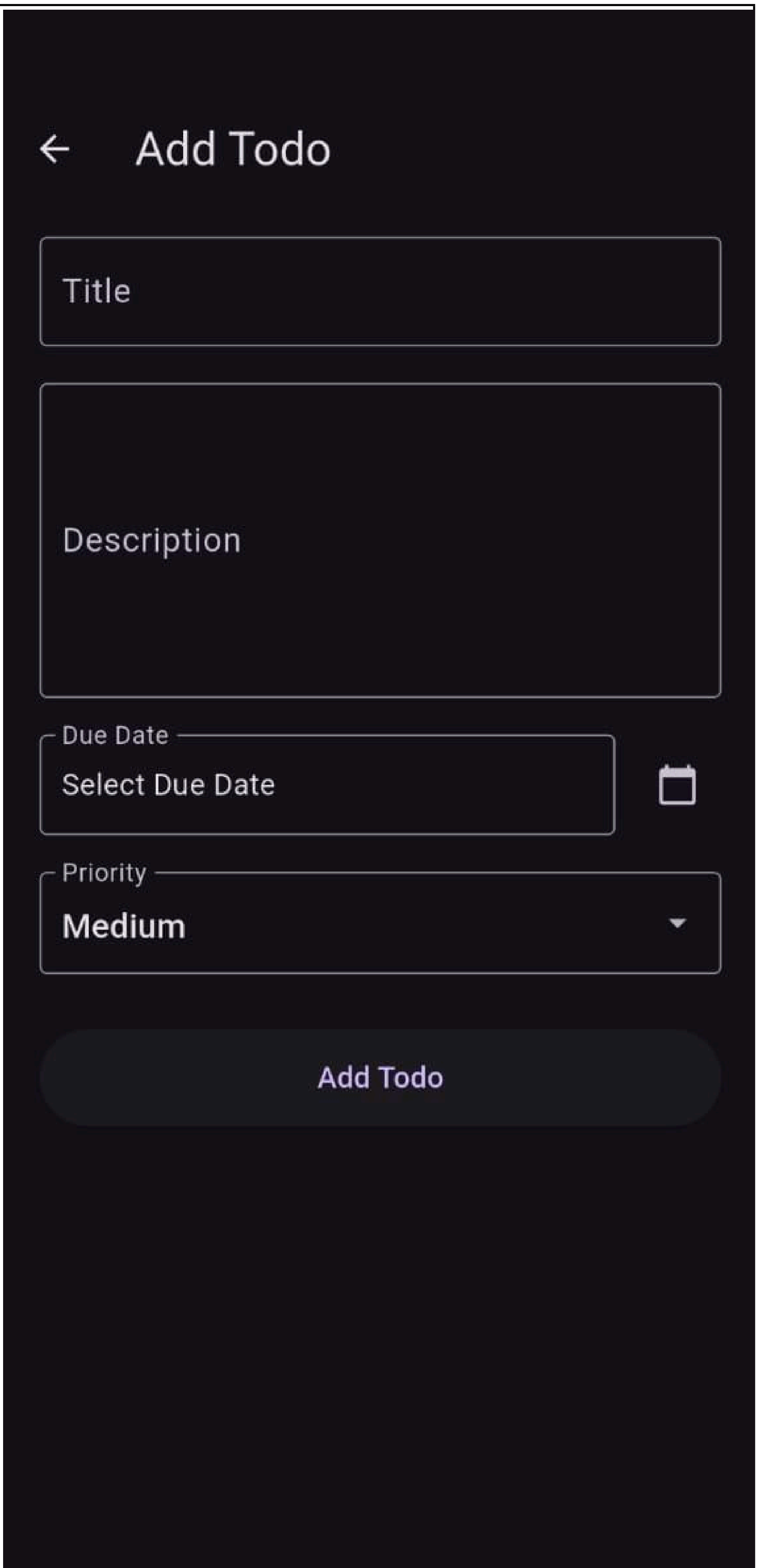
	 Welcome Back

Register
Full Name
Email
Password
Confirm Password
Register

	 No todos yet. Add one!

	 My Todos No todos yet. Add one!

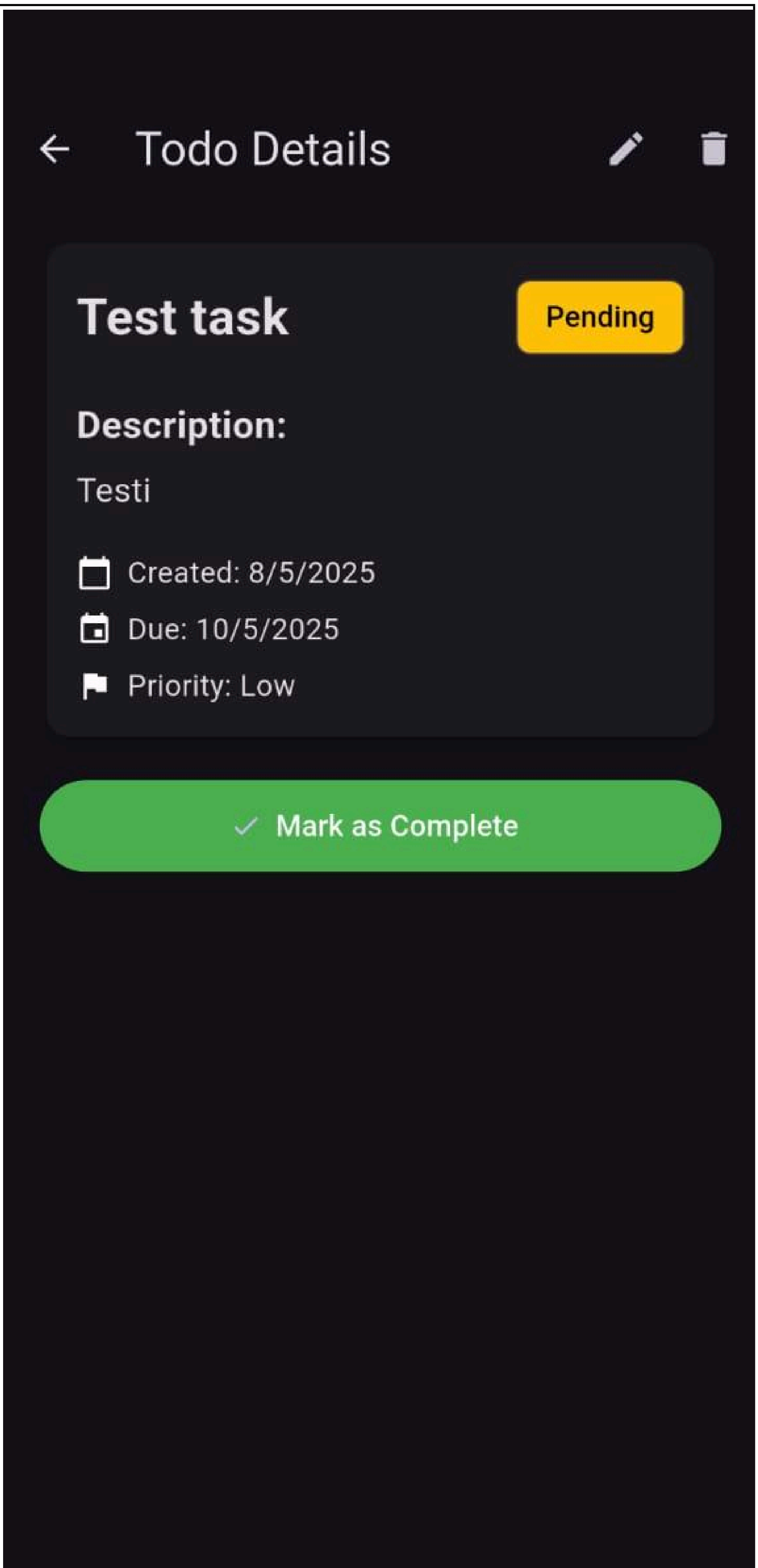
Add Todo

Title

Description

Due Date: Select Due Date

Priority: Medium

Todo Details

Test task

Pending

Description:

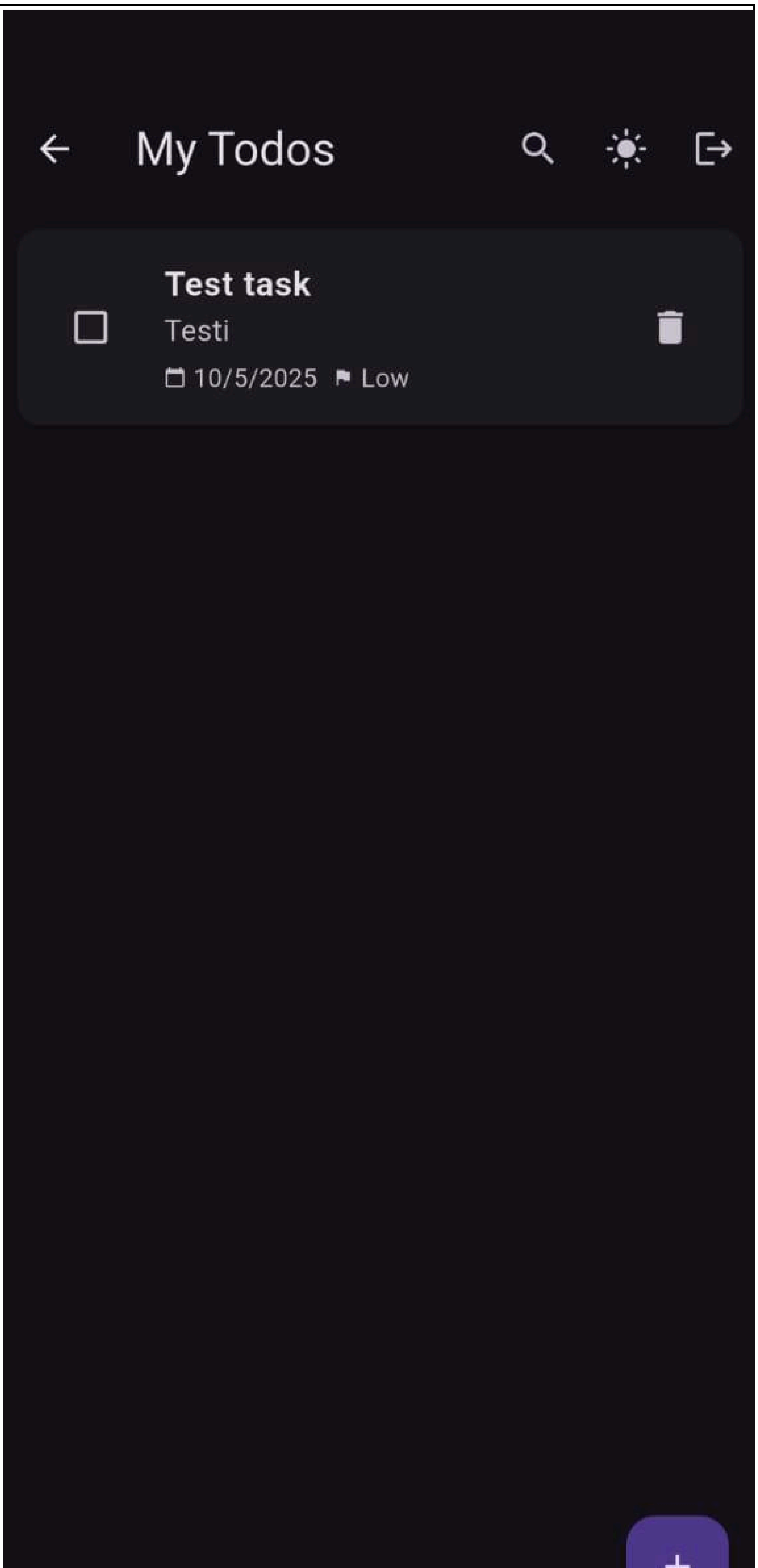
Testi

Created: 8/5/2025

Due: 10/5/2025

Priority: Low

Mark as Complete

	 My Todos Test task Testi 10/5/2025 Low

Question and Answers	Answer the following Questions: 1. Write the connection steps, step-by-step for database connectivity Add Firebase SDK to Your Flutter App Initialize Firebase in Your App apply plugin: 'com.google.gms.google-services' add dependency and services in build.gradle add firebase auth dependency
Conclusion	In this mini Flutter project, a functional To-Do application was successfully developed and integrated with Firebase services. The app includes: Email & Password Authentication for secure user login and registration Firebase Realtime Database integration for storing and managing user-specific to-do tasks This project demonstrates a foundational implementation of cloud-based backend services in Flutter. It showcases how to build a real-world application with essential features like authentication, persistent storage, and real-time data updates.