# Flutter Riverpod Task Tracker - Comprehensive Guide

## Objective

Build a simple Task Tracker app using Riverpod in Flutter. This project will help you practice and understand all key aspects of Riverpod, including state management, dependency injection, and reactivity.

## Table of Contents

1. Project Overview  
2. Prerequisites  
3. Project Structure  
4. Step-by-Step Implementation  
 - Define Models  
 - Setup Providers  
 - Build UI  
 - Add Filtering  
 - Optional Enhancements  
5. Conclusion

## 1. Project Overview

The Task Tracker app allows users to:  
- Add new tasks.  
- Mark tasks as complete or incomplete.  
- Delete tasks.  
- Filter tasks by status (All, Completed, Pending).  
- Reactively update the UI based on state changes.

## 2. Prerequisites

Ensure you have the following:  
- Flutter SDK installed.  
- Code editor (e.g., VS Code, Android Studio).  
- Basic understanding of Flutter widgets.  
- Installed `flutter\_riverpod` package:  
```bash  
flutter pub add flutter\_riverpod  
```

## 3. Project Structure

Organize your files as follows:  
```  
lib/  
├── main.dart // Entry point  
├── models/ // Task model  
│ └── task.dart  
├── providers/ // Riverpod providers  
│ ├── task\_provider.dart  
│ └── filter\_provider.dart  
├── screens/ // UI screens  
│ └── task\_screen.dart  
├── widgets/ // Reusable widgets  
│ ├── task\_tile.dart  
│ └── filter\_dropdown.dart  
└── services/ // Optional local storage  
 └── local\_storage\_service.dart  
```

## 4. Step-by-Step Implementation

### Step 1: Define the Task Model

Create a `Task` class to represent individual tasks.

```dart  
// lib/models/task.dart  
class Task {  
 final String id;  
 final String title;  
 bool isCompleted;  
  
 Task({required this.id, required this.title, this.isCompleted = false});  
}  
```

### Step 2: Setup Providers

#### Task Provider

Manage the list of tasks using a `StateNotifier`.

```dart  
// lib/providers/task\_provider.dart  
import 'package:flutter\_riverpod/flutter\_riverpod.dart';  
import '../models/task.dart';  
import 'package:uuid/uuid.dart';  
  
class TaskNotifier extends StateNotifier<List<Task>> {  
 TaskNotifier() : super([]);  
  
 void addTask(String title) {  
 state = [...state, Task(id: Uuid().v4(), title: title)];  
 }  
  
 void toggleTask(String id) {  
 state = state.map((task) {  
 if (task.id == id) {  
 return Task(id: task.id, title: task.title, isCompleted: !task.isCompleted);  
 }  
 return task;  
 }).toList();  
 }  
  
 void deleteTask(String id) {  
 state = state.where((task) => task.id != id).toList();  
 }  
}  
  
final taskProvider = StateNotifierProvider<TaskNotifier, List<Task>>((ref) => TaskNotifier());  
```

#### Filter Provider

Handle filtering logic.

```dart  
// lib/providers/filter\_provider.dart  
import 'package:flutter\_riverpod/flutter\_riverpod.dart';  
  
enum TaskFilter { all, completed, pending }  
  
final filterProvider = StateProvider<TaskFilter>((ref) => TaskFilter.all);  
```

### Step 3: Build the UI

...  
(Snipped for brevity but will be filled in similarly to the structure above)