Flutter Timer App – Project Specification Document

📌Overview

This mobile-only Flutter application allows users to manage task-based timers. Users can view, create, and control multiple timers, each tied to a task and project. All data will be handled in-memory (mocked), and the UI will follow a given Figma design system.

🕘Goals

- Create a list of active and completed task timers.
- Allow timer creation via a form.
- View detailed task info and associated timers.
- Implement clean architecture and state management using BLoC.
- Match UI with provided Figma designs.
- Enable screen-level deep linking via route names.

📱 Screens & Features

1. 🕒List of Timers Screen

- **Figma Reference**: List of Timers
- **Functionality**:
- Display a scrollable list of timers.
- Show:
    - Timer name (project/task)
    - Duration (counting up)
    - Play/Pause toggle
    - Favorite status
- + Button in AppBar → Navigates to **Create Timer Screen**.
- **Excluded**:
- Bottom navigation (Recents, Projects, etc.)
- Top tabs (Favorites, Odoo, etc.)

2. Create Timer Screen

- **Figma Reference**: Create Timer
- **Fields**:
- Dropdown: Select Project (mock data)
- Dropdown: Select Task (mock data)
- Text Field: Timer Description
- Checkbox: Mark as Favorite
- **On Submit**:

- Create and start timer
- Return to **List of Timers Screen**

3. **🧾 Task Details Screen**

    - **Figma Reference (Details tab)**: Task Details

    - **Figma Reference (Timesheets tab)**: Timesheets

    - **Tabs**:

    - **Details Tab**:

        - Task title
        - Random/static values for:
        - Deadline
        - Assigned to

    - **Timesheets Tab**:

        - Timer metadata and description
        - Controls:
        - Play / Pause
        - Stop (marks timer as **completed**, not resumable)
        - Timer UI updates should reflect in Timer List screen

    - **Optional**:

    - Display completed timers in expandable sections

State Management

- **Architecture**: Single **App-wide BLoC**
- **Optional**: Use `hydrated_bloc` for data persistence if implemented

📃 Data Handling

- All data is handled in-memory using mock values.
- Projects
- Tasks
- Timers
- No persistent backend required (Bonus points for `hydrated_bloc` usage).

🕔Theme & Styling

- Figma designs must be strictly followed.
- Implement reusable custom widgets where applicable.
- Match:
- Color scheme
- Text styles
- Component paddings and layouts

🩲Deep Linking (Mobile-Only)

- **Platform**: Mobile apps only (iOS, Android)
- **Deep Link Style**: Internal routing only (`myapp://` style NOT required)
- **Route Naming**:

Screen	Route Name
Timer List	`/timers`
Create Timer	`/timers/create`
Task Details (Details)	`/task/:taskId/details`
Task Details (Sheet)	`/task/:taskId/timesheets`

⏱️Timer Behavior

- Timers count **upward** from `00:00`
- **Multiple timers** can run concurrently
- Timers **pause/resume** via Play/Pause
- On **Stop**:
- Timer is marked **complete**
- Cannot be resumed
- Moved to a "Completed" section (UI should reflect this)

🧪Development Notes

- Recommended Flutter SDK: ≥ 3.10.0
- Preferred Packages:
- `flutter_bloc`
- `equatable`
- `hydrated_bloc` (if persistence desired)
- `go_router` or `auto_route` for navigation
- Testing encouraged (unit + widget)