**TaskFlow AI: Project Plan & Architecture**

This document outlines the complete technical and product blueprint for **TaskFlow AI**, a next-generation productivity assistant built with Flutter.

**Core Technology Stack:**

* **Framework:** Flutter
* **Backend & Database:** Firebase (Authentication, Firestore, Cloud Functions)
* **State Management:** Riverpod
* **AI Services:** Google Cloud AI (Gemini, Vision, Translate)

**1. Clean Project Architecture**

A clean architecture is essential for a scalable and maintainable app. We will follow a feature-first, layered approach.

**Project Folder Structure:**

lib/

├── main.dart

│

├── core/ # Core services, models, and utilities

│ ├── constants/ # App-wide constants (colors, strings)

│ ├── error/ # Failure/Exception handling classes

│ ├── providers/ # Global Riverpod providers (e.g., Firebase)

│ └── services/ # Abstract service definitions

│

├── features/ # Each feature gets its own folder

│ ├── auth/ # Authentication feature

│ │ ├── application/ # State logic (Riverpod providers)

│ │ ├── domain/ # Business models and repository contracts

│ │ ├── infrastructure/# Data sources (e.g., Firebase Auth repo)

│ │ └── presentation/ # UI (screens, widgets)

│ │

│ ├── tasks/ # Task management feature

│ │ ├── application/

│ │ ├── domain/

│ │ ├── infrastructure/

│ │ └── presentation/

│ │

│ └── ai\_tools/ # All AI features grouped here

│ ├── application/

│ ├── domain/

│ ├── infrastructure/

│ └── presentation/

│

└── app.dart # Main MaterialApp widget and theme setup

**State Management with Riverpod**

We will leverage different Riverpod providers for specific jobs:

* Provider: For providing simple, unchanging dependencies like a Repository.
  + *Example:* Provider<AuthRepository>((ref) => FirebaseAuthRepository())
* StateNotifierProvider: For managing complex state that can change, like user input in a form.
  + *Example:* StateNotifierProvider<TaskFormNotifier, TaskFormState>
* FutureProvider: For handling one-time asynchronous operations, like fetching a user's profile.
  + *Example:* FutureProvider<UserProfile>((ref) => ref.watch(userRepoProvider).fetchProfile())
* StreamProvider: Perfect for real-time data from Firebase.
  + *Example:* StreamProvider<List<Task>>((ref) => ref.watch(taskRepoProvider).watchTasks())

**2. Screen Flow & Navigation**

This defines the user's journey through the app. We will have **8 core screens**.

**Navigation Flow:**

Splash Screen -> Auth Gate

* **If NOT logged in:** -> Login Screen -> Sign Up Screen (optional)
* **If LOGGED in:** -> Home (Dashboard) Screen

**From Home (Dashboard) Screen, user can navigate to:**

* Task List Screen -> Add/Edit Task Screen
* AI Tools Hub Screen -> (Specific AI Tool Screens)
* Profile & Settings Screen

**Screen-by-Screen Breakdown:**

1. **Splash Screen**
   * **Purpose:** Initial loading screen. Shows the app logo. Checks for user's authentication status in the background.
2. **Login Screen**
   * **Purpose:** Allow users to sign in.
   * **Functionality:** Email/password fields, "Sign In" button, "Sign in with Google" button, and a link to the Sign Up screen.
3. **Sign Up Screen**
   * **Purpose:** New user registration.
   * **Functionality:** Name, email, password fields, "Sign Up" button, and a link back to the Login screen.
4. **Home (Dashboard) Screen**
   * **Purpose:** The main landing page after login. Provides a high-level, intelligent overview.
   * **Functionality:**
     + Welcome message (e.g., "Good morning, [User Name]!").
     + **Proactive AI Dashboard:**
       - Summary of today's tasks ("You have 3 tasks due today").
       - **AI Insight Card:** A card showing personalized insights generated by a nightly Cloud Function (e.g., "You completed 20% more tasks this week!").
       - **AI Suggestion Card:** A card with a contextual suggestion (e.g., "You often work on 'Project X' in the morning. Start a task?").
     + **Gamification Widget:**
       - Displays user's current point total and daily streak.
     + Quick-access buttons to "View All Tasks" and "AI Tools."
     + Bottom navigation bar for core app sections.
5. **Task List Screen**
   * **Purpose:** Display and manage all user tasks.
   * **Functionality:**
     + List of tasks, filterable by project, priority, or date.
     + Each task item shows title, deadline, and priority. Tapping a task opens the Add/Edit screen.
     + Checkbox to mark tasks as complete.
     + **Conversational Task Creation:** A prominent microphone icon. On tap, it opens a voice input interface. The user's speech ("Add a task to buy groceries tomorrow at 5 pm...") is sent to the Gemini API, which pre-fills the Add/Edit Task Screen for user confirmation.
6. **Add/Edit Task Screen**
   * **Purpose:** Create a new task or modify an existing one.
   * **Functionality:**
     + Fields for task title, description, deadline, and priority.
     + Option to assign the task to a project.
     + "Save Task" button. If editing, also a "Delete Task" button.
7. **AI Tools Hub Screen**
   * **Purpose:** A central place to access all the advanced AI features.
   * **Functionality:** A grid or list of buttons for:
     + "Scan Image to Task" (OCR)
     + "Ask AI a Question" (Problem Solver)
     + "Translate Text"
     + "Generate Image"
     + "Conversational AI" (Voice/Text Commands)
8. **Profile & Settings Screen**
   * **Purpose:** Manage user account and app preferences.
   * **Functionality:**
     + Display user's name and email.
     + Theme selection (Light/Dark mode).
     + **Notifications & Integrations:**
       - Toggle for enabling/disabling **Smart Notifications**.
       - Button to connect/disconnect Google Calendar to enhance notification intelligence.
     + "Logout" button.

### Screen-by-Screen Breakdown:

(Screen descriptions remain as previously defined)

1. **Splash Screen**
2. **Login Screen**
3. **Sign Up Screen**
4. **Home (Dashboard) Screen**
5. **Task List Screen**
6. **Add/Edit Task Screen**
7. **AI Tools Hub Screen**
8. **Profile & Settings Screen**

## 3. Detailed Feature Breakdown

This section provides a detailed look into each functionality.

### Core Productivity Features

1. **User Authentication (Auth)**
   * **Functionality:** Secure user sign-up, login, and session management.
   * **Implementation:** Use **Firebase Authentication** with email/password and Google Sign-In providers. When a user signs up, a corresponding user document will be created in **Cloud Firestore** using their unique uid.
   * **Riverpod:** A StreamProvider will listen to authStateChanges() to determine if a user is logged in, automatically navigating between auth screens and the main app.
2. **Task Management**
   * **Functionality:** Users can create, read, update, and delete tasks. Tasks can have a title, description, deadline, priority level, and be assigned to a project.
   * **Implementation:** All tasks will be stored in a 'tasks' sub-collection within the user's document in **Cloud Firestore**.
   * **Riverpod:** A StreamProvider will listen to the tasks sub-collection in real-time, so the UI on the Task List Screen updates instantly with any changes. A StateNotifierProvider will manage the state of the form on the Add/Edit Task Screen.
3. **Project Categorization**
   * **Functionality:** Users can create project categories (e.g., "Work," "Personal") and assign tasks to them.
   * **Implementation:** Projects will be stored as a separate sub-collection in the user's Firestore document. The UI will allow filtering the task list by these projects.

### Advanced AI & Cloud Features

1. **Conversational Task Creation**
   * **Functionality:** Users can create tasks using natural language voice commands.
   * **Implementation:**
     1. The microphone icon on the Task List Screen will use the speech\_to\_text package to capture the user's voice.
     2. The transcribed text (e.g., "Remind me to call John tomorrow at 10 am") is sent to the **Gemini API**.
     3. A carefully crafted prompt will ask Gemini to extract entities: title, deadline, time, priority.
     4. The app receives the structured JSON from Gemini and pre-fills the fields on the Add/Edit Task Screen for the user to confirm.
2. **Proactive AI Dashboard**
   * **Functionality:** The dashboard will display personalized insights and suggestions.
   * **Implementation:** A **Firebase Cloud Function** will be triggered on a nightly schedule (pub/sub). This function will read the user's task data from the past week, analyze it (e.g., count completed tasks, find most productive day), and save the insight into the user's document in Firestore. The Home (Dashboard) Screen will then simply read and display this pre-computed insight.
3. **Smart Notifications & Google Calendar Integration**
   * **Functionality:** Deliver timely, context-aware reminders.
   * **Implementation:**
     1. Users will authorize access to their Google Calendar using the official Google APIs for Flutter.
     2. When a task is due, a **Cloud Function** (triggered by a task's deadline) will first check the user's Google Calendar.
     3. If the user is busy, the notification is snoozed. If they are free, a push notification is sent via **Firebase Cloud Messaging (FCM)**.
4. **Gamification System**
   * **Functionality:** Reward users with points and streaks for completing tasks.
   * **Implementation:** A **Cloud Function** will be triggered whenever a task's isCompleted status is set to true. This function will calculate points based on priority and timeliness and update the points and streak fields in the user's Firestore document.
5. **AI Problem Solver ("Ask AI a Question")**
   * **Functionality:** Users can ask general knowledge questions or get help with problems by typing or uploading an image.
   * **Implementation:**
     1. If the user uploads an image, **Google Cloud Vision AI** will perform OCR to extract text.
     2. The extracted text (or typed text) is sent to the **Gemini API** with a prompt like "You are a helpful assistant. Please answer the following question: [user's question]".
     3. The response is streamed back to the UI.