Product Requirements Document (PRD) for AccessAdmin AI

1. Problem/Pain Point

People with disabilities (e.g., visual, hearing, cognitive, or learning impairments) often face significant barriers in administrative and office-based roles, which are among the most accessible entry-level jobs for this group. Key pain points include:

- **Accessibility gaps in core tasks**: Difficulty with real-time meeting participation (e.g., following discussions without captions or audio descriptions), note-taking, scheduling, and document handling due to reliance on single-mode tools (e.g., speech-only or text-only).
- **Limited access to modern AI tools**: Many disabled individuals struggle with AI prompting (e.g., ChatGPT) because interfaces aren't multimodal or adaptive, excluding them from productivity boosts like automated summaries or task automation.
- **Employability challenges**: In Singapore, where employment rates for persons with disabilities hover around 30-40% (with admin roles being a key focus via programs like the Open Door Programme), these barriers lead to reduced independence, higher error rates, and missed opportunities for career advancement.
- **Media handling inefficiencies**: Admin work involves consuming videos/webinars, but standard formats lack context (e.g., visual descriptions in videos for the visually impaired), increasing isolation and task completion time.
- **Broader impact**: Fragmented tools lead to fatigue, reduced confidence, and employer hesitation, perpetuating social exclusion despite tech advancements.

The solution aims to create a unified, adaptive AI companion that levels the playing field, making admin work as efficient for disabled users as for others, while fostering skill-building through accessible AI.

2. Target Audience

- **Primary Users**: Individuals with disabilities seeking or employed in admin/office roles (ages 18-65, entry-level to mid-career).
    - Visually impaired: Need audio/haptic aids for navigation and content.
    - Hearing impaired: Rely on visual/text/sign language outputs.
    - Cognitively/developmentally disabled: Require simplified, step-by-step guidance.
    - Those with dyslexia/learning disabilities: Benefit from visual/audio alternatives to text.
    - Focus on Singapore residents, where urban tech access is high, but disability employment gaps persist (e.g., via SG Enable initiatives).

- **Secondary Users**: Employers, HR teams, or job coaches who integrate the app into supported employment programs.
- **User Personas**:
    - "Alex": Visually impaired admin assistant struggling with email summaries and meeting notes.
    - "Jamie": Hearing-impaired receptionist needing real-time captions for calls/meetings.
    - "Taylor": Person with dyslexia in data entry, overwhelmed by text-heavy tasks.
- **Market Size**: In Singapore, ~3% of the population (170,000+) has disabilities; globally, 1.3 billion people with disabilities, with admin jobs as a common pathway.

3. Features

Features are prioritized for an MVP in a 2-day hackathon context: High for core functionality, Medium for enhancements, Low for nice-to-haves. Feasibility considers integration ease (e.g., Google Meet has limited real-time API access—post-meeting artifacts via REST API for paid Workspace users; real-time often requires third-party bots or device-based capture. Use Google Calendar API for scheduling, which is free and straightforward. On-device AI preferred for privacy/low latency).

High Priority (MVP Essentials: Must-have for demo, feasible with existing APIs)

- **Real-time Meeting Assistance**:
    - Capture audio via device mic (feasible: Use on-device libraries like Flutter's audio recorder).
    - Provide live transcription/summaries (feasible: Integrate open-source Whisper model or Google Cloud Speech-to-Text API; avoid direct Google Meet real-time integration—use post-meeting fetch if Workspace account, or bot services like Recall.ai for advanced, but keep simple for hackathon).
    - Multimodal outputs: Voice readout, visual captions, simplified icons (feasible: TTS via Flutter TTS package; haptics via Vibration plugin).
- **Smart Scheduling & Task Management**:
    - Voice/text/gesture commands for calendar events (feasible: Google Calendar API integration—easy OAuth setup, free tier).
    - Adaptive prompts: Simplified templates for cognitive users (feasible: Custom UI logic in app).
- **Email & Document Handling**:
    - Summarization and dictation (feasible: Gmail API for fetch/send; OCR via ML Kit for docs).

Medium Priority (Post-MVP: Add if time allows, moderate feasibility)

- **Accessible AI Prompting (Prompt Hub)**:
    - Multimodal interface for AI queries (e.g., voice-to-prompt Gemini API).
    - Guided templates for personal tasks or no-code app ideas (feasible: Gemini or OpenAI API calls; low-cost, quick integration).

- **Basic Media Accessibility Converter**:
    - YouTube/video to enriched podcast (feasible: YouTube Data API for extraction; add AI descriptions via Gemini—API limits apply, but free for low volume).

Low Priority (Future Iterations: Stretch goals, lower feasibility for hackathon)

- **Advanced Media Conversions**:
    - Full audio-to-text or text-to-audio with visuals (feasible but complex: Requires more API chaining; potential latency issues).
- **Emotional Check-Ins & Social Scripts**:
    - Mood detection and suggestions (feasible: Sentiment analysis via NLP libraries, but adds UI complexity).
- **App Creation Prototyping**:
    - Generate basic no-code apps from prompts (feasible: Integrate with tools like Bubble API, but high integration effort).

4. Tech Stack

Chosen for cross-platform (mobile/desktop), accessibility focus, and hackathon speed: Emphasize open-source/free tiers for feasibility.

- **Frontend**: Flutter (Dart) – Excellent for multimodal UI (voice, gestures, haptics); built-in accessibility (screen reader support); rapid prototyping.
- **Backend**: Firebase (for auth, real-time database, cloud functions) – Easy setup, no server management; handles user data securely.
- **AI/ML**:
    - Transcription: OpenAI Whisper (local/offline via Flutter plugin) or Google Cloud Speech-to-Text (API key needed, free tier limits).
    - Prompting/Summarization: Google Gemini API (free/low-cost, multimodal support) or Hugging Face models for offline.
    - OCR/Object Detection: Google ML Kit (free, on-device).
- **Integrations**:
    - Google Workspace: Calendar API (free), Gmail API (free), Meet API (post-meeting transcripts; real-time via third-party like Meeting BaaS if needed—$0.35-$0.50/hour, but avoid for MVP).
    - Other: YouTube Data API (free), Flutter plugins for TTS (flutter_tts), Vibration, Gesture Recognition (via camera_awesome or similar).
- **Deployment**: Firebase Hosting for web version; Android/iOS builds via Flutter.
- **Testing**: Flutter's built-in unit tests; accessibility audits with tools like WAVE.

5. Suggested Project Structure

Standard Flutter structure for maintainability; assume a monorepo for hackathon simplicity.

text

```
accessadmin_ai/
├── android/              # Android-specific configs
├── ios/                  # iOS-specific configs
├── lib/                  # Main Dart code
│   ├── main.dart         # App entry point
│   ├── models/           # Data models (e.g., UserProfile, MeetingNote)
│   │   └── user_profile.dart
│   ├── services/         # API integrations and business logic
│   │   ├── ai_service.dart # Gemini/Whisper calls
│   │   ├── calendar_service.dart # Google Calendar API
│   │   ├── transcription_service.dart # Audio capture & transcription
│   │   └── auth_service.dart # Firebase Auth
│   ├── ui/               # Screens and widgets
│   │   ├── screens/      # Main app pages
│   │   │   ├── home_screen.dart # Dashboard
│   │   │   ├── meeting_assist_screen.dart # Real-time meeting UI
│   │   │   └── prompt_hub_screen.dart # AI prompting interface
│   │   └── widgets/      # Reusable components (e.g., adaptive_button.dart for multimodal inputs)
│   └── utils/            # Helpers (e.g., accessibility_utils.dart for mode detection)
├── assets/               # Icons, images for visual aids
├── test/                 # Unit tests (e.g., ai_service_test.dart)
├── pubspec.yaml          # Dependencies (flutter_tts, firebase_core, google_ml_kit, etc.)
└── README.md             # Project overview
```

- **Version Control**: Git with branches (main, feature/meeting-assist).
- **Build Tools**: Flutter CLI for builds; Firebase CLI for deployment.

6. Database Schema

Use Firebase Firestore (NoSQL) for scalability and real-time sync; fallback to SQLite for offline MVP.

- **Collections**:
  - **users** (Document ID: userID from Firebase Auth)
    - Fields:
      - disability_profile: String (e.g., "visual", "hearing", "cognitive") – For adaptive UI.
      - preferences: Map (e.g., {output_mode: "voice", haptic_enabled: true}) – Custom settings.
      - created_at: Timestamp.
  - **meeting_notes** (Subcollection under users or standalone with userID index)
    - Fields:
      - meeting_id: String (unique hash).
      - transcript: String (full text).
      - summary: String (AI-generated).
      - action_items: Array of Maps (e.g., [{task: "Follow up email", due: Timestamp}]).
      - timestamp: Timestamp.
  - **prompt_history** (Subcollection under users)

        - Fields:
            - prompt: String (user input).
            - response: String (AI output).
            - mode: String (e.g., "voice").
            - timestamp: Timestamp.
- **Security**: Firestore rules to restrict reads/writes to authenticated users.
- **Offline Support**: Firebase's offline persistence for mobile use.
- **Schema Evolution**: Start simple; add indexes for queries (e.g., by timestamp).