ТЕХНИЧЕСКАЯ СПЕЦИФИКАЦИЯ ПРОЕКТА MINDFLOW (v1.0)

1. ОБЩИЕ СВЕДЕНИЯ

Название: MindFlow

Тип: Кроссплатформенное приложение для ментального здоровья (Wellness / Meditation).

Целевые платформы: iOS, macOS (текущая адаптация), Android (потенциально).

Основная функция: Генерация процедурных аудио-ландшафтов (Soundscapes) и синхронной визуализации для состояний потока, сна и медитации.

1.1. Стек технологий

1) **Frontend & Business Logic:** Flutter (Dart 3.x).
2) **Native Audio Engine:** Swift 5 (AVFoundation).
3) **Bridge:** Flutter Method Channels.
4) **Local Storage:** SharedPreferences (ключ-значение, JSON-сериализация).
5) **Architecture Pattern:** Feature-based Architecture + Repository Pattern + Singleton Services.

2. АРХИТЕКТУРА СИСТЕМЫ

Система построена по **гибридной модели**:

1. **Dart (Logic Layer):** "Мозг" приложения. Рассчитывает математику микширования, управляет состоянием UI, хранит данные и принимает решения (AI).
2. **Swift (Hardware Layer):** "Мышцы" приложения. Отвечает за низкоуровневое воспроизведение звука, наложение эффектов (Reverb, Delay) и работу с аудио-буферами.

2.1. Схема потока данных (Audio Pipeline)

Фрагмент кода

```
[Sensors/Time] -> [AiContextFrame] -> [MindflowAiEngine] -> [AiEngineParams]
       +
[SceneConfig] -> [SoundscapeEngine] -> [AudioParams (Volume, Cutoff, etc.)]
                                                    |
                                              (MethodChannel)
                                                    |
                                                    v
                                          [Native AppDelegate] -> [AVAudioEngine Graph] -> [Speakers]
```

3. ДЕТАЛЬНОЕ ОПИСАНИЕ ПОДСИСТЕМ

3.1. Аудио-Движок (Hybrid Audio Engine)

А. Логический слой (Dart) — `lib/shared/audio/`

1. **`SoundscapeEngine` (Алгоритмическое ядро):**
    1) **Алгоритм "Ветряной порыв" (Wind Gust):** Используется для природных сцен (лес, дождь). Генерирует псевдослучайные колебания громкости и фильтрации, складывая три синусоиды с разной частотой и фазой (`_phase`). Это создает ощущение непредсказуемости природы.
    2) **Scene Mixing Logic:**
        a) *Energy/Run:* Текстура (барабаны) на 95%, Reverb 5%, Cutoff 100% (для четкости атаки).
        b) *Sleep:* Drone 80%, Pad 30%, Reverb 70% (для эффекта "обволакивания").
        c) *Focus:* Pulse 60%, Texture 70% (биты для ритма), Pad 0% (чтобы не усыплять).
2. **`AudioEngine` (Контроллер):** Синглтон, отправляющий команды `loadLayers`, `start`, `stop`, `setVolume`, `updateMix` в нативный слой.

Б. Нативный слой (Swift) — `macos/Runner/AppDelegate.swift`

1. **Граф обработки (`AVAudioEngine`):**
    1) **Sources:** 4 узла `AVAudioPlayerNode` (каналы: Drone, Pad, Texture, Pulse) + 1 узел `sfxNode` (для звуков интерфейса/колокольчиков).
    2) **Mixer:** `AVAudioMixerNode` собирает все каналы.
    3) **Delay:** `AVAudioUnitDelay` (эффект эха).
    4) **Reverb:** `AVAudioUnitReverb` (пресет `.largeHall`).
    5) **Filter:** `AVAudioUnitEQ` (Low Pass Filter 12dB/oct) — используется для "заглушения" звука (эффект "под водой" или затишья).
2. **Механизм воспроизведения (Looping):**
    1) Файлы загружаются в память (`AVAudioPCMBuffer`).
    2) Воспроизведение запускается методом `scheduleBuffer(..., options: .loops)`, что обеспечивает **бесшовное зацикливание** без пауз (sample-accurate looping).
3. **Universal Asset Loader (`findAudioFile`):**
    1) Реализован эвристический алгоритм поиска файла. Проверяет пути в следующем порядке:
        a) Полный путь через `FlutterDartProject.lookupKey`.
        b) `Bundle.main.url` (корневые ресурсы).
        c) Подпапка `flutter_assets` внутри ресурсов.
4. **Self-Healing (Самовосстановление):**
    1) Метод `startEngine()` принудительно пересоздает связи в аудио-графе (`audioEngine.connect`) перед каждым запуском. Это предотвращает фатальную ошибку `player started when in a disconnected state`, возникающую при смене аудио-устройств или перезапуске движка.

3.2. Визуальный Движок (Generative Visuals)

Реализован на `CustomPainter`, отрисовка происходит покадрово (60 FPS).

1. **`GenerativePatternPainter` (Маршрутизатор):**
    1) Принимает `sceneId` и `time` (значение анимации).
    2) Switch-case логика выбирает нужный алгоритм отрисовки.
2. **Алгоритмы визуализации (Painters):**
    1) **`AmbientPainters`:**
        a) *Rain:* Система частиц (линии), падающих сверху вниз. При ударе о "землю" (низ экрана) рисуются расходящиеся круги (ripples).
        b) *Ocean:* Наложение 5 слоев синусоид (`Math.sin`) с разной амплитудой и смещением фазы.
        c) *Fire:* Частицы, движущиеся вверх с уменьшением масштаба и изменением цвета (от красного к желтому).
    2) **`MoveSessionPainters`:**
        a) *Run:* Имитация дороги в перспективе с движущимися боковыми линиями и центральной разметкой.
        b) *Gym:* Пульсирующие концентрические круги (имитация напряжения/расслабления).
    3) **`WorkPainters`:**
        a) *Coding:* Эффект "Matrix Rain" (вертикальные цепочки символов/линий).
        b) *Focus:* Вращающиеся орбиты с частицами (атомная модель).
3. **`LavaLampOverlay` (Интерактивный слой):**
    1) Физическая симуляция "метаболов" (мягких шаров).
    2) Реализовано отталкивание частиц друг от друга и от стен.
    3) Реакция на тач: нажатие создает импульс, расталкивающий шары.

3.3. Искусственный Интеллект (Mindflow AI)

1. **`MindflowAiEngine`:**
    1) **Вход:** `AiContextFrame` (время суток, желаемая интенсивность).
    2) **Циркадный ритм:** Вычисляет коэффициент `circadian` (косинус от времени суток). Утром звук ярче (больше высоких частот), вечером глуше (Low Pass Filter активнее).
    3) **Адаптация:** Учитывает "Learned Biases" (смещения, выученные на основе действий пользователя в `TunePanel`). Если пользователь часто добавляет "Воздух", AI навсегда повышает этот параметр.
2. **`InsightEngine` (Аналитика):**
    1) Анализирует массив `SessionRecord`.
    2) Находит паттерны: "Любимое время суток", "Самое частое настроение", "Текущий стрик".
    3) Генерирует текстовые рекомендации ("Попробуй сессию Фокус, так как сейчас утро").

4. СТРУКТУРА ДАННЫХ И ХРАНЕНИЕ

4.1. Модели Данных (`lib/shared/models/`)

1. **`SessionRecord`:**
    a) Лог завершенной сессии.
    b) Поля: `startedAt`, `durationSeconds`, `moodId`, `completed` (досмотрел до конца или нет), `avgUserAirShift` (настройки эквалайзера).
2. **`UserProfile`:**
    a) Профиль пользователя.
    b) Поля: `name`, `age`, `gender`, `moodId` (последнее состояние), `moodScore`.
3. **`FavoritePreset`:**
    a) Сохраненная комбинация настроения и фона.
    b) Поля: `moodId`, `ambientId` (например, "Фокус" + "Дождь").

4.2. Репозитории (`lib/shared/repo/`)

Все репозитории работают как Синглтоны и используют `ValueNotifier` для реактивного обновления UI.

1) **`SessionRepo`:** Загружает/сохраняет JSON-список истории. Рассчитывает статистику для AI.
2) **`SettingsRepo`:** Хранит глобальные флаги (`haptics`, `defaultDuration`).
3) **`FavoritesRepo`:** CRUD операции для избранного.

5. ПОЛЬЗОВАТЕЛЬСКИЕ СЦЕНАРИИ (FLOWS)

5.1. Сессия (Session Lifecycle)

1. **Инициализация:** Экран `SessionScreen` создает `SessionRunner`.
2. **Загрузка:** `SessionRunner` командует `AudioEngine` загрузить файлы (stems) согласно `SceneConfig`.
3. **Loop:** Таймер (100мс) вызывает `_tick()`:
    1) AI рассчитывает новые параметры.
    2) `SoundscapeEngine` превращает их в `AudioParams` (громкости, фильтры).
    3) Параметры летят в Native слой.
4. **Визуализация:** `AnimationController` обновляет `GenerativePatternPainter`.
5. **Завершение:** При истечении времени данные пишутся в `SessionRepo`, открывается `SessionDoneScreen`.

5.2. Настройка звука (Tuning)

Пользователь открывает `TunePanel` и двигает слайдеры "Воздух" / "Фокус".

1. Это меняет локальные переменные в `SessionRunner`.
2. AI подхватывает изменения мгновенно.
3. При завершении сессии среднее значение этих слайдеров сохраняется в историю для обучения AI.

6. КОНФИГУРАЦИЯ И РЕСУРСЫ

6.1. Ассеты (`assets/audio/stems/`)

Файловая структура плоская, имена файлов жестко привязаны в коде.

1. **Типы файлов:** `.wav` (для бесшовного лупинга).
2. **Роли файлов:**
    a) `silence.wav` — заглушка для пустых слоев.
    b) `ambient_*.wav` — текстуры природы.
    c) `loop_energy_drums.wav` — ритмическая основа.

6.2. Конфиг Сцен (`lib/shared/audio/soundscape_config.dart`)

Словарь `kSoundscapes` маппит ID сцены (например, `forest`) на набор из 4 файлов (Drone, Pad, Texture, Pulse).

7. ИЗВЕСТНЫЕ ОГРАНИЧЕНИЯ И ОСОБЕННОСТИ

1. **macOS Sandbox:** Для работы отладчика и доступа к файлам требуются разрешения `com.apple.security.network.server` и `com.apple.security.network.client` в `.entitlements`.
2. **Asset Path:** На macOS ассеты Flutter упаковываются глубоко в бандл (`App.framework/Resources/flutter_assets`), поэтому используется кастомный алгоритм поиска пути.
3. **Background Audio:** На данный момент режим фонового воспроизведения (при свернутом приложении) реализован базово через настройки `Info.plist`, но требует доработки `AVAudioSession` для iOS.