Думкопис — Поточний стан застосунку (UA)

Цей документ — повна українськомовна довідка щодо поточного стану проєкту, архітектури, налаштувань, інтеграцій і кроків для запуску та деплою. Мета — дати максимально детальну картину для роботи, внесків та публікації у приватний репозиторій.

Коротко про застосунок

- Назва пакета (`pubspec.yaml`): `dumkopys`
- Платформи: Android, iOS, Web, Windows, macOS, Linux
- Стан: Riverpod (Notifier + ручні провайдери)
- Запис: `record` + `audioplayers`, локальне збереження аудіо
- Транскрибування: Google Gemini через бекенд-проксі (переважно) або прямий API (фолбек)
- Автентифікація: Firebase Auth (Google Sign-In), убезпечена ініціалізація
- Дані: Firestore для історії (за наявності входу), `SharedPreferences` для локальних налаштувань/історії
- Remote Config: отримує `gemini_api_key` та `gemini_endpoint` із пріоритетами та фолбеком

Версії

- Версія у `pubspec.yaml`: `1.5.5+1`
- У розділі «Про застосунок» у UI відображається: `1.6.9`
- Рекомендація: узгодити джерело правди для версії перед релізом (або синхронізувати UI).

Основні файли

- `lib/main.dart` — старт застосунку, `ProviderScope`, убезпечена ініціалізація Firebase
- `lib/src/app.dart` — `MaterialApp`, тема, шрифти (Montserrat, Pacifico)
- `lib/src/app/auth_gate.dart` — гейт автентифікації: контент або екран входу
- `lib/src/features/recorder/recorder_page.dart` — головна сторінка: запис, історія, налаштування, завантаження файлів
- `lib/src/features/settings/settings_state.dart` — Notifier: тема, тактильний відгук, історія, акаунти, синхронізація з Firestore
- `lib/src/services/firebase_providers.dart` — ручні провайдери `FirebaseAuthService`, `FirestoreService`, stream `authStateChanges`
- `lib/src/services/firestore_service.dart` — обгортка Firestore з ретрай/гардом для CRUD
- `lib/src/services/remote_config_service.dart` — singleton для Remote Config + фолбеки на env та дефолтний endpoint
- `lib/src/services/gemini_service.dart` — вибір ендпоінту, виклики до бекенд-проксі, фолбек на прямий Gemini API

Технологічний стек (з `pubspec.yaml`)

- Core: `flutter`, `google_fonts`
- Стан: `flutter_riverpod`, `riverpod_annotation`, `riverpod_generator`
- Зберігання: `shared_preferences`, `path_provider`
- Аудіо: `record`, `audioplayers`
- UI: `lottie`, `flutter_signin_button`
- Пристрій: `vibration`
- Дані та мережа: `cloud_firestore`, `firebase_core`, `firebase_auth`, `google_sign_in`, `firebase_remote_config`, `http`, `url_launcher`, `share_plus`
- Android foreground: `flutter_foreground_task`
- Dev/test: `build_runner`, `test`, `flutter_test`, `mockito`, `flutter_lints`, `flutter_launcher_icons`

Архітектура

- Riverpod
- `Settings` Notifier керує темою, тактильним відгуком, історією транскрипцій, відомими акаунтами.
- Зберігає стан у `SharedPreferences`; при вході синхронізує історію з Firestore.

- Firebase ініціалізація
- Убезпечена: застосунок працює навіть без коректно налаштованого Firebase.
- Ініціалізація можлива в debug або якщо передано прапорець `FIREBASE_ENABLED=true`.

- Потік транскрибування
- Запис аудіо сегментів локально, візуалізація (еквалайзер).
- Надсилання аудіо на бекенд-проксі (переважний шлях) або на прямий Gemini API (фолбек/дебаг).
- Результати додаються до поточної транскрипції; історія оновлюється та зберігається.

Функціональність

- Запис та відтворення
- Кнопка запису: старт/стоп; візуалізація рівнів.
- Локальне збереження аудіофайлів; експорт/шерінг.

- Транскрибування
- Обробка сегментів; збирання тексту у вью.
- Копіювання, очищення, експорт транскрипції.

- Історія
- Локально: до 25 записів у `SharedPreferences`.
- При вході: синхронізація з Firestore (`users/{uid}/history`), злиття та обрізання до 25.
- Операції: додати, перейменувати, дописати, оновити текст, видалити; Firestore віддзеркалює зміни при вході.

- Акаунти та автентифікація
- Вхід через Google Sign-In (Web/Mobile).

- «Відомі акаунти» (до 5) зберігаються локально; допоміжний UX для входу (login hint на Web).
- Якщо Firebase недоступний або невірно налаштований — UI показує контент без екрану входу.

- Налаштування
- Тема: Світла/Темна/Системна.
- Тактильний відгук: перемикач.
- Швидке копіювання: за наявності в UI.
- Дані: очистити історію (видалення всіх локальних записів та відповідних аудіофайлів).
- Android: інструкції щодо оптимізації батареї для foreground service.

Інтеграції з Firebase

- Auth
- Web: Firebase Auth із Google, підтримка `login_hint`, сценарії заблокованих попапів.
- Mobile: Google Sign-In дістає `idToken`, створюється credential через `GoogleAuthProvider`.

- Firestore
- Колекція: `users/{uid}/history` з полями `id`, `title`, `text`, `createdAt`, `platform`.
- Операції з Firestore не фатальні: локальний стан не ламається при помилках.
- Синхронізація при вході: пуш відсутніх локальних записів, злиття з віддаленими, зріз до 25.

- Бекенд-проксі автентифікація
- Виклики до бекендів (транскрибування) включають заголовок `Authorization: Bearer <Firebase ID token>`, якщо користувач увійшов.

Remote Config та вибір ендпоінтів

- Ключі
- `gemini_api_key`
- `gemini_endpoint`

- Порядок отримання

1) `--dart-define` (середовище) на час запуску
2) Отримання з Firebase Remote Config (з таймаутом та мінімальним інтервалом)
3) Фолбек: дефолтний ендпоінт (наприклад, Cloud Functions/або значення з `vercel.json`/`api/`)

Прапорці та змінні середовища

- `FIREBASE_ENABLED=true` — ініціалізує Firebase поза debug.
- `GEMINI_API_KEY=...` — прямі виклики до Gemini (debug/фолбек).
- `GEMINI_ENDPOINT=...` — перевизначення бекенд-проксі.
- `LOGIN_HINT=...` — хінт для входу на Web.

Платформні примітки

- Web
- Обмеження запису аудіо залежно від браузера; підтримано завантаження файлів.
- Забезпечити авторизований домен у Firebase, коректну конфігурацію та сховище.
- Android
- Foreground service: дотримуватися рекомендацій для енергозбереження.
- iOS/macOS/Windows/Linux
- Налаштувати Firebase згідно з платформними вимогами (bundle IDs, entitlements, `firebase_options.dart`).

Потік даних (узагальнення)

1) Користувач записує аудіо → локально зберігається через `path_provider`.
2) Сегменти обробляються → оновлюється вью транскрипції.
3) Якщо користувач увійшов → історія дзеркалиться у Firestore.
4) Запити до бекенду включають ID токен для авторизації.
5) Локальна історія/налаштування працюють незалежно від доступності Firebase.

Тестування та CI

- Тести: `test/` — парсинг Gemini, рекордер, Settings Notifier, віджет-тести.
- CI: GitHub Actions у `.github/workflows/flutter_ci.yml` та `release_android.yml`.

Запуск та деплой

- Debug
- `flutter run -d chrome` (Web)
- `flutter run -d android` (Android)
- Увімкнути Firebase поза debug
- `flutter run --dart-define=FIREBASE_ENABLED=true`
- Налаштувати транскрибування
- Задати ключі в Remote Config або передати:
- `--dart-define=GEMINI_API_KEY=...`
- `--dart-define=GEMINI_ENDPOINT=...`
- Бекенд
- Перевірити `api/gemini-transcribe.ts` (Vercel) або `functions/src/index.ts` (Cloud Functions) — ендпоінт і авторизація.

Безпека та конфіденційність

- Секрети
- Не комітити API ключі в репозиторій. Використовувати Secrets (GitHub/Vercel/Firebase).
- Обмежувати доступ до Firestore даних правилами — лише власнику `uid`.
- Firestore правила (рекомендаційний шаблон)
- Дозволити читання/запис у `users/{uid}/history` тільки автентифікованому користувачу з тим же `uid`.

- Логи та помилки
- Обробка помилок бекенду та Firestore — нефатальна, локальний UX продовжує працювати.

Відомі обмеження

- Розсинхронізація версії: `pubspec.yaml` vs «Про застосунок» у UI.
- Порядок сортування Firestore за `createdAt` може бути нестабільним; код обробляє це стійко.
- Аудіофайли зберігаються лише локально; Firestore містить лише метадані/текст.
- Продакшн — переважно через бекенд-проксі; прямий Gemini — для debug/фолбеків.

Робота з приватним репозиторієм GitHub

- Чи варто заливати у приватний? Так. Це гарна практика для ранніх версій, з секретами та внутрішніми налаштуваннями.
- Кроки створення приватного репозиторію:
- Створити репозиторій на GitHub з назвою, наприклад, `dumkopys` і позначкою `Private`.
- У локальному проєкті налаштувати remote: `git remote add origin https://github.com/<your_org_or_user>/dumkopys.git`
- Запушити: `git push -u origin main` (або поточну гілку).
- Перевірити `.github/workflows/` — чи потрібні секрети для релізів (Android підпис, тощо). Якщо так — додати у GitHub Secrets.

Перейменування репозиторію

- Пакет уже має назву `dumkopys` у `pubspec.yaml`. Репозиторій можна сміливо назвати `dumkopys` — буде консистентно.
- Для локальної папки:
- Можна перейменувати каталог `balabol` → `dumkopys` (зупинивши IDE/збірку на час операції).
- Якщо remote вже налаштований — оновити URL: `git remote set-url origin https://github.com/<your_org_or_user>/dumkopys.git`.
- Перевірити:
- CI конфіги, шляхи та бейджі у `README.md` (за потреби оновити назву).

Додатково: локалізація UI українською

- Наразі більшість технічної документації англійською. Якщо потрібно, можна:
- Додати Flutter `intl`/генератор локалізації (ARB файли) для українських рядків UI.
- Витягнути всі жорстко закодовані рядки в локалізаційні ресурси.

Подальші кроки

- Узгодити версію між `pubspec.yaml` та UI.
- Перевірити Remote Config (ключі присутні, політика фетчу та кешування ок).

- Перевірити Firestore правила безпеки.
- Налаштувати CI Secrets (за потреби Android Release, тощо).
- Вибрати бекенд-проксі (Vercel/Firebase Functions), зафіксувати endpoint у Remote Config.
- За потреби — додати українську локалізацію UI.

Пов’язані документи

- `docs/Architecture.md` — архітектура
- `docs/Codebook.md` + `docs/Codebook.*.md` — детальні пояснення файлів
- `docs/Setup.md` — початкове налаштування
- `docs/Deployment.md` — деплой
- `docs/Security.md` — безпека
- `BACKEND_SETUP.md` — налаштування бекенду