1. Overview

GlucoVibe is a cross-platform Flutter mobile app to help people with diabetes manage glucose, hydration, medication adherence, meals, and activity. The experience is futuristic and minimal — holographic/neon aesthetic on a dark background — with an accessible reduced-motion fallback.

2. Goals

- Provide simple, reliable logging for glucose, water, medication, meals, and activity.
- Present a clear dashboard with present state and quick actions.
- Encourage adherence with lightweight notifications and progress indicators.
- Be visually modern and accessible (reduced motion, readable contrast, large tap targets).
- Integrate with CI/test pipelines (TestSprite) and support future persistence and device integrations.

3. Primary Users & Personas

- Primary: Adults with diabetes who want daily logging and trend snapshots.
- Secondary: Caregivers and clinicians reviewing summary reports.
- Tech: Mobile-first users on Android/iOS; tablet/desktop support for clinicians.

4. Core Features

- Landing hero: animated, neon holographic visual (fallback PNG for reduced motion).
- Dashboard: quick status (latest glucose, hydration total, meds adherence), quick actions (Record glucose, Log water, Log meal, Log activity).
- Detailed record dialogs/screens:
    - Glucose: value, sample type (capillary/venous/plasma), context (fasting/pre/post/other), timestamp, notes.
    - Hydration: custom units (ml/oz/cup) and presets.
    - Meal tracker: name, calories (kcal), sugar (g), timestamp; daily totals.
    - Activity tracker: type, intensity, duration, estimated calories burned (MET-based).
- Profile screen: editable personal info (name, age, weight, diabetes type).
- Basic in-memory services with clear interfaces for hydration/activity/glucose; prepared for persistence (hive/sqflite/shared_preferences).
- Accessibility: reduced motion support and static fallback (assets/images/landing_fallback.png).

5. Functional Requirements

- FR1: App must launch to LandingScreen and navigate to Dashboard.
- FR2: Dashboard quick actions open respective detailed UI.
- FR3: Glucose entries saved to GlucoseService and visible on Dashboard.
- FR4: Hydration accepts ml/oz/cup and stores internally in ml.
- FR5: Meal entries aggregate daily totals.
- FR6: Activity logs estimate calories and aggregate totals.
- FR7: Profile edits persist to chosen storage (future implementation).
- FR8: All forms validate required fields and show confirmations (SnackBar).

6. Non-Functional Requirements

- NFR1: Responsive UI across phone/tablet/desktop (hero and components scale).
- NFR2: Lightweight rendering — particle counts and heavy visuals scale with device.
- NFR3: Reduced motion mode respects OS accessibility setting (MediaQuery.disableAnimations).
- NFR4: Secure handling of personal data; no plaintext secrets in repo.
- NFR5: Test coverage: unit tests for services, widget tests for landing/dashboard interactions.

7. UX / Visual Design

- Dark base: #050814
- Neon palette: Teal #00D1C1, Indigo #4B6FFF, Violet #8A6CFF
- Glassmorphism: frosted translucent panels with subtle inner glow and soft shadows.
- Typography: sleek geometric sans (use system font or variable font). App title glows softly.
- CTA: glowing “Get Started” with subtle pulsing (stop in reduced motion).
- Edge glow for logo: glow follows PNG alpha edges (not a circular halo).
- All icons and labels on dark surfaces use high contrast (labels white, icons light purple).

8. Assets & File Locations

- Put logo and fallback in: assets/images/
    - health_icon.png — primary logo (transparent PNG).
    - landing_fallback.png — static fallback (512×512).
- Declare in pubspec.yaml:
- Provide high-DPI variants in 2.0x/3.0x if available.

9. Data Model (in-memory sketches)

- GlucoseReading { id, valueMgDl (double), sampleType, context, timestamp, notes }
- HydrationEvent { id, ml (int), timestamp }
- MealEntry { id, name, kcal (int), sugarG (double), timestamp }
- ActivityEntry { id, type, intensity, durationMin, weightKg, caloriesEstimated, timestamp }
- Profile { name, dob, weightKg, diabetesType, preferredUnits }

10. Services / APIs (local-first)

- lib/services/glucose_service.dart — CRUD for GlucoseReading
- lib/services/hydration_service.dart — logDrinkMl/logDrinkUnit, todayTotalMl(), todayTotalInUnit()
- lib/services/activity_service.dart — logActivity, todayActivities, totals
- Future: sync adapters for backend/EHR (OAuth 2.0 or FHIR) — design interfaces now to swap implementations.

11. Security & Privacy

- No hardcoded API keys in repo. Use environment/secret storage (CI secrets, secure local config).
- Store PHI locally encrypted for persistent storage (recommend: hive + flutter_secure_storage or SQLCipher).
- Provide clear privacy notice and allow export/delete of stored data.
- Use least-privilege for permissions (no extraneous device permissions).

12. Testing Strategy

- Unit tests for services (glucose, hydration, activity calculations).
- Widget tests for LandingScreen, DashboardScreen, ProfileScreen basic flows.
- Integration tests for end-to-end logging flows.
- CI integration: GitHub Actions workflow (.github/workflows/testsprite.yml) runs `flutter test --coverage` and can invoke TestSprite CLI if configured.
- Local helper: tool/run_tests_and_testsprite.ps1 to run tests and call TestSprite CLI.

13. CI / TestSprite Integration

- Workflow runs tests and collects coverage.
- DO NOT store API keys in repo. Use GitHub Secrets:
  - TESTSPRITE_CMD — CLI command (e.g., testsprite upload …)
  - TESTSPRITE_API_KEY — API key for the service
- VS Code local TestSprite config in settings.json must not include live keys — remove or replace prior to committing.

14. Accessibility

- Respect reduce motion (MediaQuery.disableAnimations).
- Provide semantic labels for major interactive elements.
- Ensure contrast and large tap targets (min 44×44 dp).
- Provide text alternatives for images and fallback PNG.

15. Performance & Optimization

- Scale and clamp particle counts and blur radii based on device pixel ratio and canvas size.
- Use ImageFiltered sparingly; prefer pre-rasterized assets for heavy glows on low-end devices.
- Profile on lower-end Android devices; provide "low motion / low effect" quality toggle.

16. Internationalization & Localization

- Prepare for i18n: wrap user-facing strings with Intl/plurals support.
- Dates/times should follow device locale or user preference.

17. Roadmap / Next Milestones

- M1 (Now): Polish landing + dashboard aesthetics (completed).
- M2: Add persistent storage (hive/shared_preferences) for readings & preferences.
- M3: Add charts for glucose trends (fl_chart).
- M4: Export/import CSV and basic reporting for clinicians.
- M5: Optional integrations: Bluetooth glucose meter ingestion, FHIR backend sync.
- M6: Accessibility audit & UX testing.

18. Acceptance Criteria

- App launches to landing and navigates to dashboard.
- Glucose/hydration/meal/activity log flows create entries and show confirmations.
- Reduced motion toggles to static fallback and no animated heavy effects.
- Assets load and display; no hardcoded secrets remain in repo.

- CI runs tests and (if configured) invokes TestSprite command using secrets.

19. Notes / Developer Guidance

- Keep secrets out of settings.json and any committed configs. Replace with reference to environment variables.
- Edge-glow logo: use stacked blurred, color-filtered copies of the PNG to follow alpha edges.
- Export animations to Rive/Lottie for richer, lighter runtime animations if needed; add runtime loader (rive/lottie packages) later.