ChatGPT

Platform and Technology

Build the app with Flutter to support Android and (eventually) desktop (Linux/Windows) from a single codebase [1]. Flutter's cross-platform model lets you write once and target mobile and desktop (and even web) natively [1]. This simplifies maintenance and ensures consistent behavior across platforms.

UI/UX Design Principles (KISS, Familiarity)

Apply the **KISS (Keep It Simple, Stupid)** principle: avoid clutter, overload of text or colors, and excessive icons [2]. Use standard calendar conventions so users feel immediately at home (for example, calendar grids with weekdays as users expect). Highlight the current date and clearly distinguish weekends or holidays, as Apple's Calendar does [3]. Provide consistent, predictable navigation (e.g. arrows or swipe gestures to change month/year) so the user always knows where to look next [4] [5].

- **Minimalist layout:** Don't overload each screen. Show only what's needed (e.g. month grid with dates, optional side labels) [2].
- **Consistent color-coding:** Use color accents (amber theme) consistently (e.g. for highlighting today, special days, or user events), but avoid too many colors to prevent confusion [2] [6].
- **Easy navigation:** Provide simple controls (back/forward arrows or swipe for months/years). Consider a bottom navigation bar or a subtle year selector. The attached app uses a drawer; you might streamline this by showing year-selector in the header or via long-press, reducing hidden UI.
- **Readable typography:** Use sufficiently large, clear text for dates and labels [7]. Ensure text labels are unambiguous ("ሰኞ" vs. "Monday") and, if using icons (e.g. evangelist symbols), always accompany them with text for clarity.

Visual Design & Theming

Start with the amber theme (amber containers, dark-amber text, white backgrounds) but **verify contrast** to maintain readability. WCAG recommends a minimum contrast ratio of 4.5:1 for normal text [8]. Dark amber on white may be close to this threshold; test or adjust shades accordingly. For dark mode, define a complementary dark palette: typically dark background with light-colored text/icons. Support **Light, Dark, and System** theme modes [9] so that on newer OS versions the app automatically respects the user's preference [9]. For example, Flutter's `MaterialApp` can be configured with `theme` and `darkTheme`, defaulting to `ThemeMode.system` [9].

- **Dark mode support:** Ensure all custom widgets and colors adapt correctly to dark mode. Users expect major UI elements to invert: dark backgrounds with light text/icons [10]. (E.g. your amber containers might become a dark gray or indigo.)
- **Customizable appearance:** Consider offering a setting to choose numeral style (Geez vs. Arabic numerals) and language (Amharic/English) [11], since many Ethiopian calendar users appreciate bilingual support and local numerals [11] [12]. These options make the app feel personalized.

Accessibility Features

Build in accessibility from the start. This boosts usability for everyone and avoids costly fixes later. Key points:

- **Contrast & Font Size:** Maintain high contrast between text and background [8]. Allow users to resize text or respect system font-size settings. Flutter widgets automatically scale, but check that your custom widgets (like evangelist icons) scale properly.
- **Large Tap Targets:** Ensure buttons and date cells are at least ~44×44 pixels or more [8] to be easily tappable. Add sufficient spacing so users with motor impairments don't mis-tap dates or buttons.
- **Semantic Labels:** Provide descriptive labels for interactive elements (e.g. the images for the four Evangelists should have alt-text or semantics describing the evangelist name). This helps screen-readers announce what each element is.
- **Focus Order & Navigation:** Keep a logical, predictable focus order. For example, when opening the year-detail page, focus should start on the top "Evangelist" card and move downward through holidays. Avoid hidden navigation (minimize use of swipes as the only means to navigate).
- **Color Blindness:** Don't rely on color alone to convey meaning (e.g. if amber highlights holidays, also use icons or text).

By following these accessibility guidelines (e.g. high contrast, resizable text [8], clear labels, big touch targets), the app will feel polished and inclusive.

Core Features (Events & Reminders)

Include the essential calendar functions that users expect [13] [6]:

- **Event Scheduling & Creation:** Allow users to tap a date or a "+" button to add an event. A Floating Action Button (FAB) for adding events is a common pattern [14]. Opening a simple dialog (title, optional time and notes) is effective [15]. Store events locally (e.g. in SQLite or Hive). Provide a day or agenda view listing all events for a selected day.
- **Reminders & Notifications:** Let users set reminders/alerts for events. Use a local notifications plugin (e.g. `flutter_local_notifications`) to schedule alarms, since no internet is needed [16]. This ensures offline functionality. Notifications keep users informed of upcoming events without requiring the app to be open.
- **Recurring Events:** Support at least basic recurrence (daily/weekly/monthly/yearly), as many users rely on repeating events (e.g. birthdays or weekly fasts). This can often be added in settings of each event.
- **Multiple Views:** Offer Month, Week, and Day views (or at least Month and an Agenda/Day list). Users appreciate being able to switch between a broad overview and a detailed day schedule.

These features make the app a true calendar (not just a static holiday list) [13] [6]. Keep the UI for creating and editing events straightforward: e.g. a simple form with clear "Save/Cancel" buttons [15].

Ethiopian Calendar Specifics (Dual Calendar & Bahre-Hassab)

Preserve and enhance the unique Ethiopian elements:

- **Dual Ethiopian/Gregorian Display:** Show both calendars side by side. For example, in each month grid cell, display the Ethiopic date as the main number and the Gregorian date in smaller text (or vice versa). The app reviewed by Uptodown emphasizes *"simultaneous… Ethiopian and Gregorian"* display [17]. Make it easy to understand which is which (e.g. label the month/year row with the Ethiopian month name and the Western month name). Allow toggling this dual display if a user prefers one calendar only.
- **Bahre-Hassab Dates (Saints' Feasts and Fasts):** Integrate the calculated dates from Bahre-Hassab as special entries on the calendar. For example, overlay icons or color marks on days like Meskel, Genna, Timket, Nineveh, etc. Tapping a marked day can show the feast name (as in your `DisplayDetails` list). Additionally, include a **Year-Detail screen** (as you have) that summarizes Bahre-Hassab info (Evangelist, year start day, list of feasts) for quick reference. This screen should be easily reachable (e.g. via a "Year Info" button). Existing Ethiopian calendar apps highlight *"detailed Bahire-Hasab insights"* and cultural holidays [12] [17].
- **Cultural Holidays & Saints:** Include built-in data for Ethiopian Orthodox Tewahedo Church (EOTC) saints' days and fasts for each date. The user notes include "EOTC days of saints" – this can be a calendar overlay or a daily detail view. Highlight public holidays (Timkat, Fasika, Enkutatash, etc.) and provide brief info (e.g. "Genna (Ethiopian Christmas)"). This matches features of top Ethiopian calendar apps [12] [17].

In summary, ensure all Bahre-Hassab–derived dates appear both in the overview calendar (as marks or in-day notes) and in a dedicated "Year Details" page like your `DisplayDetails`. Label everything clearly in Amharic and (optionally) English to serve all users [11].

Offline Data & Persistence

Design the app for offline-first operation:

- **Local Database:** Store all events, user-added entries, and calendar metadata in a local database (e.g. SQLite with `sqflite` or NoSQL like Hive). This guarantees instant access without internet latency. As one guide notes, users in low-network areas rely on this [6].
- **Precomputed Calendars:** Preload or compute Ethiopian calendar rules (Bahre-Hassab, leap years, etc.) on-device. Since these calculations are deterministic, the app can generate them for any year on the fly.
- **Sync/Backup (Optional):** Even without online syncing, consider letting users export/import their events (e.g. via an iCalendar/ICS file) as a "backup" feature. This isn't mandatory, but useful.
- **Performance:** Ensure smooth scrolling and quick response. As one UX guide warns, a slow calendar is frustrating [2]. Since everything is local, optimize your database queries and use Flutter's efficient rendering (e.g. `ListView.builder` for lists).

Local operation also means you can schedule all notifications locally (no FCM needed). Using packages like `flutter_local_notifications` or `AwesomeNotifications` will cover most needs.

Additional Features

Focus on one or two high-value extras beyond the core calendar:

- **Reminders (Alerts):** As mentioned, local push notifications for events. This is highly appreciated by users and works offline [18].
- **Widgets:** Provide home-screen widgets (Android) that show the current date (Ethiopian and Gregorian) and upcoming events or holidays. The Uptodown review highlights an *"intuitive home screen widget"* that boosts usability [19]. On Android, you might offer a small widget showing today's date and next event; on Windows/Linux, consider a system tray calendar or desktop widget if feasible.
- **Language Support:** Even if the primary UI is Amharic, include English translations (or vice versa). This is already a popular feature [11]. Having both languages (and optional numeral styles) broadens your audience.
- **Settings & Help:** Keep a simple settings page (dark mode toggle, language toggle, first-day-of-week, etc.) and a help/about page explaining Bahre-Hassab or the Ethiopian calendar basics for curious users.

Don't overload with features – better to polish these well. Extra social or cloud-sync features are *not* needed given the offline requirement. Instead, perfect the core: a seamless, fast calendar with accurate dates and a clean UI.

Summary of UI Flow

- **Main Screen:** Show the current month's grid (Ethiopic months like *Meskerem*, etc.) with each day labeled in Ethiopic and the corresponding Gregorian date on the side. Highlight today. A top app bar or drawer can let the user jump to a different year or open "Year Info."
- **Day Tap:** Tapping a day opens a detail view: list events/reminders for that day and note any saint or holiday.
- **Add Event:** A floating "+" button (FAB) opens a simple dialog to enter an event title and time.
- **Year Info Screen:** Accessible via a button or menu, this scrollable page (similar to your `DisplayDetails`) shows that year's Evangelist, New Year's weekday, and a list of major feasts/fasts with their Gregorian dates. Structure it in sections or a neat list (your current card + list is fine).
- **Navigation:** Provide back/forward arrows (or swipe) to switch months, and clear "Today" shortcut. If using a drawer, ensure its icons and labels are clear (e.g. icons for Settings, Year Info, About).

Each UI element should look and behave intuitively. For inspiration, Apple's Calendar **"does one thing clearly"** (clean, no-frills) [3] and Google Calendar uses familiar layouts and customizable colors [20]. Aim for that level of polish: a calendar that feels familiar, is easy to read, and gets out of the way so users can focus on dates and events.

Sources: UI/UX and feature suggestions are drawn from industry guidelines and examples of top calendar apps [2] [12] [6] [19]. The above plan integrates these best practices while honoring the unique Ethiopian calendar requirements outlined by Bahre-Hassab.

---

[1] [13] [14] [15] A Guide to Crafting a Calendar App with Flutter | by Zeba Academy | Mobile Dev Mag | Medium
https://medium.com/mobiledev/guide-crafting-calendar-app-flutter-b91d730902af

[2] [3] [4] [5] [7] [20] Best practices for Calendar design — FIX-UX | by codevil | Bootcamp | Medium
https://medium.com/design-bootcamp/best-practices-for-calendar-design-fix-ux-dc57b62d9bb7

[6] [18] Flutter Calendar Mobile App Development - Appilian
https://appilian.com/flutter-calendar-mobile-app-development/

[8] Accessibility Guidelines for Mobile Apps | by Crissy Joshua | Oct, 2025 | Medium
https://medium.com/@crissyjoshua/accessibility-guidelines-for-mobile-apps-223ab919b98b

[9] [10] Dark Mode in Flutter. In this article I'll discuss coding for... | by Paul Mutisya | Medium
https://medium.com/@pmutisya/dark-mode-in-flutter-3742062f9f59

[11] [12] Ethiopian Calendar & Holidays App - App Store
https://apps.apple.com/jm/app/ethiopian-calendar-holidays/id1588675742

[16] Scheduling Notifications with Local Notifications in Flutter - Medium
https://medium.com/fludev/scheduling-notifications-with-local-notifications-in-flutter-8bf751b6ac3c

[17] [19] Ethiopian Calendar for Android - Download the APK from Uptodown
https://ethiopian-calendar.en.uptodown.com/android