

iOS app

What the app does

🕒 5 min read · Updated 13 October 2026 · Verified against [693a8f4](#) on 2026-09-27

Gurbani Soul screen by screen: the five tabs, widgets and Live Activity, Siri and deep links, settings, what the public build leaves out, and its accessibility.

<https://docs.gurbanisoul.com/ios/what-the-app-does/>

A tour of the app for an engineer who has not opened it, traced to the code in `gurbani-soul-ios` at the pinned commit. Every screen reads through `CorpusActor`, the on-device equivalent of the API (app architecture), and every verse it shows is verbatim, cited by Ang.

The tabs

The tab bar appears only after the launch integrity check passes; until then the app shows a “Verifying” view, or a failure view that presents no scripture. The app opens on **Nitnem**.

Tab	What it is for	Main screens
Nitnem	The daily prayers: what to read now, today’s progress, and the morning, evening, night, popular and ceremony sets.	Nitnem home, bani reader, My Nitnem (reorder, hide or add banis), reminders, the reading journey (a month record of days read, which you can clear)
Reader	The Granth Ang by Ang, 1 to 1430, resuming where you left off.	Ang pages you swipe through, Jump to Ang (type, scrub, step or pick a raag), the Hukam
Search	Search and quotation checking.	Modes Auto, Gurmukhi, Roman, English (when the build has it), First letters, Theme and Verify; Verify accepts an optional Ang (“@712”)
Explore	Study views over the whole Granth.	Index (major compositions, raags, banis and sections, voices), Themes, Lineage (the contributors, compare two voices), Insights (contributors, raags, theme network, resonance, flow), Constellation, Vaars (the salok and pauri structure), the Raag Clock
More	Settings and the app’s own facts.	Saved verses, settings, About (the integrity details), privacy policy, diagnostics

Every verse row carries the same actions, from its context menu and as VoiceOver actions: **Copy** (the Gurmukhi only), **Share** (the Gurmukhi and “— Sri Guru Granth Sahib Ji, Ang N”), **Share as card** (an image), **Save**, and **Explore related**, which opens the trail of related verses shown in bands of relatedness rather than raw scores.

The Raag Clock

The Explore tab's clock shows which pahar (watch of the day) it is and the raags the traditions assign to it. It runs in **Fixed** mode (eight three-hour watches from 06:00; the midnight watch has no raags, and says so) or **Solar** mode, the traditional reckoning: four equal watches of daylight and four of night at your location, stretching with the season. Location is asked for only when you choose “use my location” (when-in-use only), and only the coordinates rounded to two decimal places are kept; you can also type them, and “Forget location” (from 1.4.0) removes them. A sheet shows where timing traditions disagree. In the Reader, a timing chip on each Ang opens the clock at that raag.

Widgets and the Live Activity

Widget	Sizes	Shows	Refreshes	Tap opens
Raag now	small, medium, large, and the three Lock Screen sizes	the current pahar and its raags, following the Fixed or Solar setting	every minute for three hours, then every fifteen minutes and at each pahar boundary	the Raag Clock
Hukam verse	medium, large	the opening verse of a Hukam, its transliteration and Ang	after 00:05 each day; the verse itself is drawn afresh at each app launch	that composition
Nitnem	small, medium, and the three Lock Screen sizes	the next bani and today's progress	at fixed points through the day	that bani, or the Nitnem tab

A **Live Activity** (opt-in in settings) shows the bani being read and a percentage on the Lock Screen and in the Dynamic Island; it never shows a verse. The widgets read only what the app writes to the App Group ([privacy and on-device data](#)).

Siri, Shortcuts, Spotlight and links

- **App Intents:** draw a Hukam, what raag it is now, open a bani (ten choices, Japji Sahib by default), search Gurbani, and open an Ang. The first four have Siri phrases; opening an Ang is available in Shortcuts only.
- **Spotlight** indexes saved verses only; tapping one opens its composition.
- **The sggs:// scheme** (handled by Router.handle): ang/N?line=, shabad/N?line=, bani/<key>?variant=, composition/<key>, theme/<slug>, search?q=, hukam, nitnem, and clock or clock/<raag>. Widgets, intents and reminder notifications all open the app this way.

gurbani-soul-ios/ios/App/Sources/Navigation/Router.swift · handle · lines 194–298

```

func handle(_ url: URL, container: AppContainer) {
    guard url.scheme == "sggs" else { return }
    let host = url.host ?? ""
    let value = url.pathComponents.dropFirst().first ?? "" // path only – the query
is excluded
    let line: Int? = URLComponents(url: url, resolvingAgainstBaseURL: false)?
        .queryItems?.first(where: { $0.name == "line"
    })?.value.flatMap(Int.init).flatMap { $0 > 0 ? $0 : nil }
    let variant: String? = URLComponents(url: url, resolvingAgainstBaseURL: false)?
        .queryItems?.first(where: { $0.name == "variant" })?.value
    switch host {
    case "ang":
        if let n = Int(value), GranthAngs.all.contains(n) {
            // A deep link / intent to an Ang must land ON the Reader, not behind a
shabad/hukam
            // sheet that happened to be open (the reader would otherwise be hidden
until dismissed).
            container.dismissSheetForNavigation()
            openAng(n, lineId: line)
        }
    case "clock":
        openClock(raag: value.isEmpty ? nil : value)
    case "nitnem":
        selectedTab = .nitnem
        nitnemPath = NavigationPath()
    case "bani":
        // A variant is an explicit request for ONE form of a bani (e.g. the Vaar as
printed
        // vs its kirtan interleave), which is a composition read, not the day's Nitnem.
        if let v = variant, !v.isEmpty {
            openComposition(key: value, variant: v)
        } else {
            openBani(key: value)
        }
    case "vaars":
        container.dismissSheetForNavigation()
        openVaar(nil)
    case "vaar":
        // sggs://vaar/5 · sggs://vaar/5/read · sggs://vaar/5?pauri=12 · sggs://vaar/5?
line=20900
        guard let id = Int(value), Router.vaarIds.contains(id) else { break }
        let pauri = URLComponents(url: url, resolvingAgainstBaseURL: false)?
            .queryItems?.first(where: { $0.name == "pauri" })?.value.flatMap(Int.init)
        container.dismissSheetForNavigation()
        openVaar(id, read: url.pathComponents.dropFirst().dropFirst().first == "read",
pauri: pauri, line: line)
    case "composition":
        openComposition(key: value, variant: variant ?? "")
    case "raag":
        // Only a value that names a raag may close an open sheet; anything else changes
nothing.
        if Router.raagName(value) != nil {
            container.dismissSheetForNavigation()
            openRaag(value)
        }
    case "themes":
        container.dismissSheetForNavigation()
        selectedTab = .explore

```

```

        explorePath = NavigationPath([Route.themes])
    case "theme":
        // sggs://theme/<slug>[?term=<Gurmukhi word>&rahao=1]: the theme page, or its
verse list
        // when narrowed. A malformed slug is ignored, never queried.
        // a well-formed name that is not one of the themes (a Siri phrase like "mercy")
opens
        // the index rather than a not-found page
        if let slug = ConstellationLink.slug(value),
            container.meta.map({ $0.concepts.contains { $0.concept == slug } }) == false
        {
            container.dismissSheetForNavigation()
            selectedTab = .explore
            explorePath = NavigationPath([Route.themes])
        } else if let slug = ConstellationLink.slug(value) {
            let items = URLComponents(url: url, resolvingAgainstBaseURL:
false)?.queryItems ?? []
            let term = items.first(where: { $0.name == "term" })?.value
                .map { $0.trimmingCharacters(in: .whitespaces) }
                .flatMap { $0.isEmpty || $0.count > 40 ? nil : $0 }
            let rahao = items.first(where: { $0.name == "rahao" })?.value == "1"
            container.dismissSheetForNavigation() // never land behind an open
shabad sheet
            selectedTab = .explore
            if term != nil || rahao {
                let f = ThemeFilter(term: term, lens: ConstellationLens(rahaoOnly:
rahao))
                explorePath = NavigationPath([Route.themes, Route.theme(slug),
Route.themeVerses(slug, f)])
            } else {
                explorePath = NavigationPath([Route.themes, Route.theme(slug)])
            }
        }
    case "shabad":
        if let c = Int(value), (1...Int(Int32.max)).contains(c) {
            container.present(.shabad(compId: c, focusLineId: line))
        }
    case "hukam":
        // sggs://hukam draws afresh; sggs://hukam/today opens the Hukam of the day (the
widget's verse)
        container.present(value == "today" ? .hukamOfTheDay : .hukam)
    case "constellation":
        let items = URLComponents(url: url, resolvingAgainstBaseURL: false)?.queryItems
?? []
        func q(_ k: String) -> String? { items.first(where: { $0.name == k
})?.value.flatMap(ConstellationLink.slug) }
        container.dismissSheetForNavigation()
        openConstellation(ConstellationLink(theme: q("theme") ?? "naam", with:
q("with"),
                                alone: items.first(where: { $0.name ==
"alone" })?.value == "1"))
    case "voice":
        if Router.isValidVoiceSlug(value) {
            container.dismissSheetForNavigation()
            selectedTab = .explore
            explorePath = NavigationPath([Route.lineage, Route.voice(value)])
        }
    case "search":
        let q = URLComponents(url: url, resolvingAgainstBaseURL: false)?
.queryItems?.first(where: { $0.name == "q" })?.value ?? ""

```

```

        if !q.isEmpty {
            pendingSearchQuery = q
            selectedTab = .search
        }
        default: break
    }
}

```

Read from the source at build time — [view these lines on GitHub at a386422](#) .

Source: `ios/App/Sources/Navigation/Router.swift` in `gurbani-soul-ios`, read at the pinned commit.

Settings

Setting	Default	Notes
Traditional saroop	on	A display form of the Gurmukhi; copy, share and VoiceOver use the verbatim text
Transliteration	on	
English translation	on	Only in a build that bundles English
Gurmukhi size	24	18 to 32
Appearance	system	system, light or dark
Accent	Soul Gold	five palettes (brand)
Rehras form	SGPC	or Taksal
Timing chip in the Reader	on	
Sehaj focus	off	hides the Reader's page controls while reading
Line spacing, page tone	0.4, paper	tone is paper, warm or night
Auto-scroll pace	steady	not offered with Reduce Motion or assistive technology
Live Activity	off	
Reminders	off	one per band, at 06:00, 18:00 and 21:30 by default; local notifications only

What the public build leaves out

There are no feature flags in the code. The app looks at which tables its bundled database has and hides what is missing. The **public** profile, the one on the App Store, has no English layer, so it has no English search mode, no English toggle and no translation credit. Timing and the banis are gated the same way. See [the database pair](#) for the two profiles and the licence gate that decides which may ship.

Accessibility

- VoiceOver reads the **verbatim** Gurmukhi tagged as Punjabi, never the saroop display form; transliteration is hidden from VoiceOver, and English is announced as “English translation”.
- Gurmukhi scales with Dynamic Type. Two places cap it: the Reader’s title and the Hukam widget.
- Reduce Motion is honoured in the reader, the clock and the controls.
- The Raag Clock’s dial is decorative for VoiceOver; the list of pahars is the accessible path.
- The interface is in English only; there are no localisations yet.

Links

1. <https://github.com/Algorythmos-AI/gurbani-soul-ios/blob/a386422e90c2c681f9ecc4f7167b4deb6a35fb92/ios/App/Sources/Navigation/Router.swift#L194-L298>

Sri Guru Granth Sahib Ji — Knowledge Base wiki. Generated from commit c3d9360. Updated 13 October 2026. Verified against 693a8f4 on 2026-09-27.
This copy may be out of date; the page is at <https://docs.gurbanisoul.com/ios/what-the-app-does/>