PANDORICA - Project Specification

Version: 1.0
Date: 2026-02-09
For: Swarm Kit Implementation
Project Type: Cross-platform markdown notes system with Google Drive sync

---

Executive Summary

PANDORICA is a cross-platform markdown notes application that replaces Obsidian for personal use. It provides a consistent, familiar interface across Mac, Windows, iOS, and iPad, syncing via Google Drive API. The system must be accessible to Claude Code on desktop platforms for programmatic note management.

Core Value Proposition:

- Own your infrastructure (no subscription costs)
- Your files, your storage, your control
- Redirect $10/month from Obsidian subscription to Mike's API budget (~3.3M Haiku tokens/month)
- Consistent UI across all platforms
- Reliable sync without vendor lock-in

Architecture Overview

Technology Stack

Desktop (Mac & Windows):

- Swarm decides: Native Swift (Mac) + Electron (Windows), OR Electron for both
- Decision criteria: Code reuse vs native feel
- Requirement: Visual consistency across platforms matters more than technical purity

Mobile (iOS & iPad):

- Native Swift/SwiftUI
- Can share codebase with Mac if Mac goes native
- Installable via Apple Developer account (no App Store required)

Sync Layer:

- Google Drive API for file storage and sync
- OAuth 2.0 for authentication
- All devices authenticate against same Google account
- No custom server infrastructure required

File Structure

```
Google Drive/
└── PANDORICA/
    ├── daily/
    │   └── YYYY-MM-DD.md
    ├── specs/
    ├── bugs/
    ├── ideas/
    └── archive/
```

Notes:

- Application creates directory structure on first launch
- User can create additional folders as needed
- All files are standard `.md` markdown format
- No proprietary file formats or metadata

Core Features (MVP)

1. Markdown Editing

- Live preview (split pane: editor on left, rendered on right)
- Syntax highlighting in editor pane

- CommonMark standard compliance
- Support for:
  - Headers (H1-H6)
  - Lists (ordered, unordered)
  - Code blocks (with syntax highlighting)
  - Links
  - Images (stored in Google Drive)
  - Tables
  - Blockquotes
  - Bold, italic, strikethrough

2. File Management

- Tree view navigation (collapsible folders)
- Create/rename/delete files and folders
- Move files between folders (drag-and-drop on desktop)
- File search by name
- Quick switcher (CMD/CTRL+P to jump to any file)

3. Search

- Full-text search across all markdown files
- Search results show context (surrounding lines)
- Click result to jump to file and location
- Search while typing (live results)

4. Sync

- Google Drive API integration
- OAuth 2.0 authentication flow
- Conflict resolution: last-write-wins with timestamp
- Offline mode: Changes queue and sync when online

- Sync status indicator (synced, syncing, offline)
- Manual sync trigger button

5. Claude Code Integration (Desktop Only)

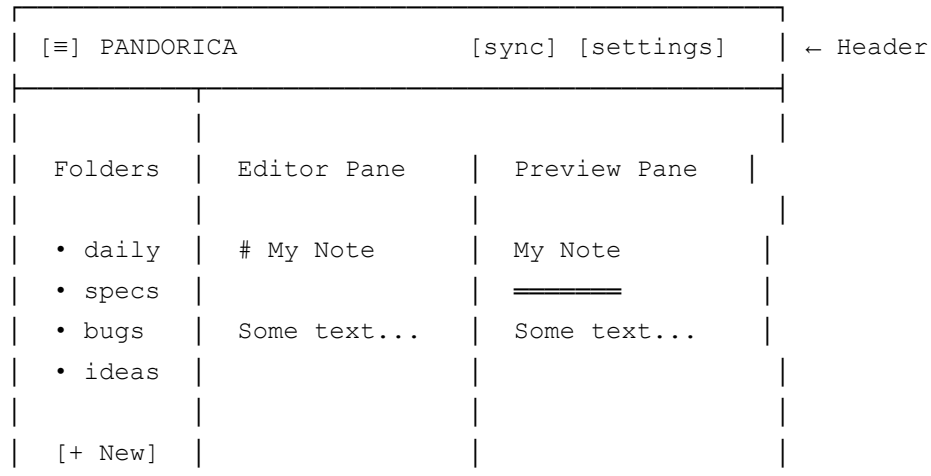
- Markdown files accessible via standard filesystem paths
- Claude Code can read/write files programmatically
- Clear documentation of file locations for each platform
- Example use cases:
  - "Read my notes on Mike's architecture"
  - "Synthesize these three spec documents"
  - "Create a summary and save it to PANDORICA"

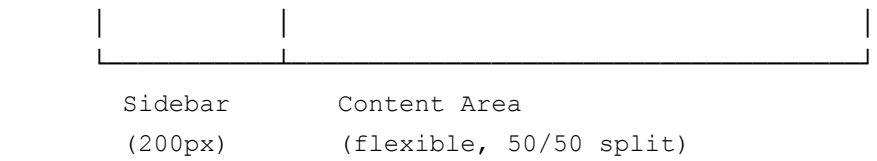
User Interface Requirements

Design Principles

- **Familiar:** Borrow heavily from Apple Notes and Obsidian UI patterns
- **Clean:** Minimal chrome, focus on content
- **Consistent:** Same layout/behavior across Mac and Windows
- **Fast:** Instant search, quick file switching, responsive editing

Layout





Sidebar (Left):

- Folder tree
- New file/folder buttons
- Search box at top
- Collapsible folders

Content Area (Right):

- Split editor/preview for desktop
- Single pane with toggle for mobile
- Tab bar if multiple files open (desktop)

Mobile Adaptations

- Single pane (editor OR preview, toggle button)
- Swipe gesture to reveal folder sidebar
- Toolbar at bottom for formatting
- Share sheet integration for importing/exporting

First-Run Setup Flow

Desktop

1. Launch PANDORICA
2. "Welcome to PANDORICA" screen
3. "Sign in with Google" button
4. OAuth flow in system browser
5. User grants Google Drive permissions
6. App creates `Google Drive/PANDORICA/` structure

7. “Setup complete” → Main interface

Mobile

1. Same flow, optimized for mobile Safari
2. OAuth redirect returns to app
3. Face ID / Touch ID for subsequent launches (stores refresh token securely)

Technical Requirements

Google Drive API

- Use Google Drive API v3
- Required OAuth scopes:
  - `https://www.googleapis.com/auth/drive.file` (access to app-created files)
  - `https://www.googleapis.com/auth/drive.appdata` (for settings storage)
- Store refresh token securely (system keychain)
- Handle token refresh automatically

Sync Strategy

- Watch for file changes locally
- Poll Google Drive for changes every 30 seconds when active
- Push changes immediately on save
- Pull changes on app launch and periodically
- Show sync conflicts to user with merge options

File Format

- UTF-8 encoded `.md` files
- No frontmatter required (optional for metadata)
- Standard markdown - readable in any text editor

Performance

- App launch: < 2 seconds
- File switching: Instant
- Search: Results appear as user types
- Sync: Background operation, non-blocking

Platform-Specific Details

Mac

- Native menu bar integration
- Keyboard shortcuts follow Mac conventions (CMD)
- System Services integration (New Note from Selection)
- Spotlight integration (optional, nice-to-have)

Windows

- System tray icon
- Keyboard shortcuts follow Windows conventions (CTRL)
- File Explorer context menu (optional, nice-to-have)

iOS/iPad

- Share sheet integration (save to PANDORICA)
- Files app integration (view PANDORICA folder)
- Keyboard shortcuts on iPad (CMD key support)
- Split View / Slide Over support (iPad)

Security & Privacy

- All authentication via OAuth 2.0
- No password storage
- Refresh tokens stored in system keychain

- All data stored in user's Google Drive
- No telemetry, no analytics, no tracking
- No data sent anywhere except Google Drive

Future Considerations (Post-MVP)

These are explicitly OUT OF SCOPE for initial build but documented for future reference:

- Tags and tag-based filtering
- Wiki-style links between notes (`[[link]]`)
- Graph view of note connections
- Custom themes / dark mode toggle
- Plugins or extensions
- End-to-end encryption layer
- Alternative sync backends (iCloud, Dropbox, self-hosted)
- Vim keybindings mode
- Export to PDF/HTML

Acceptance Criteria

Must Have (Deployment Blockers)

1. ✓ Can create, edit, delete markdown files
2. ✓ Files sync reliably across Mac, Windows, iOS, iPad
3. ✓ UI is consistent and familiar across desktop platforms
4. ✓ Full-text search works and is fast
5. ✓ Claude Code can access markdown files on desktop
6. ✓ OAuth flow works smoothly
7. ✓ App creates Google Drive folder structure automatically

8. ✓ Offline mode queues changes for sync
9. ✓ No data loss during normal operation
10. ✓ Installable via Apple Developer account (iOS/iPad)

Should Have (Fix Before 1.0)

- Keyboard shortcuts documented
- Conflict resolution UI (if conflicts occur)
- Manual sync trigger
- Settings panel (Google account, folder location)
- About screen with version number

Nice to Have (Can Ship Without)

- Multiple file tabs (desktop)
- Drag-and-drop file reorganization
- Image paste from clipboard
- Recent files list
- Markdown cheat sheet in app

Testing Strategy

Automated Tests

- Unit tests for markdown parser
- Integration tests for Google Drive API calls
- Sync conflict scenarios

Manual Testing Checklist

- Create note on Mac → Appears on Windows
- Create note on iPhone → Appears on Mac
- Edit note on Windows → Changes appear on iPad

- Delete note on iPad → Deleted on all devices
- Offline edit on Mac → Syncs when back online
- Simultaneous edit on two devices → Conflict resolved
- Claude Code can read/write notes on Mac
- Claude Code can read/write notes on Windows
- App survives Google Drive quota exceeded
- App handles network failures gracefully
- OAuth token refresh works
- Large files (>1MB) sync correctly
- Search finds content across all notes

Deployment Package

Deliverables

1. **Mac Application**
   - `.app` bundle (unsigned for now)
   - Installation instructions
   - Claude Code integration documentation
2. **Windows Application**
   - `.exe` installer or portable executable
   - Installation instructions
   - Claude Code integration documentation
3. **iOS/iPad Application**
   - Xcode project
   - Build instructions for Apple Developer account
   - Provisioning profile setup guide
4. **Documentation**

- User guide (how to use PANDORICA)
- Setup guide (Google account, first run)
- Claude Code integration guide
- Architecture documentation for future maintenance

5. **Source Code**
   - Full source with comments
   - Build scripts
   - README with development setup

Budget Justification

Current Cost: Obsidian Sync = $10/month = $120/year

Equivalent Value in Mike's Budget:

- $10/month = ~3.3M Haiku input tokens
- $10/month = ~200K Haiku output tokens
- Annual: ~40M input tokens or ~2.4M output tokens

One-time build cost via Swarm Kit vs **perpetual subscription** = Infrastructure investment that pays for itself.

Additional value:

- Complete control over features
- No vendor lock-in
- Future-proof (own the code)
- Can extend/modify as needed
- Serves as template for other infrastructure tools

Success Metrics

1. **Functional:** All acceptance criteria met

2. **Performance:** Sub-2-second app launch, instant file switching
3. **Reliability:** Zero data loss during 30-day testing period
4. **Usability:** Dino can use it without referring to documentation
5. **Integration:** Claude Code can programmatically manage notes
6. **Budget:** Obsidian subscription cancelled, $10/month redirected to Mike

---

Project Constraints

- **No feature creep:** MVP only, gold medal execution
- **KISS principle:** Simple and boring beats clever and fragile
- **Fail loudly:** Problems surface visibly, no silent failures
- **Your files:** Standard markdown, no lock-in
- **No subscription:** One-time build, perpetual use

Swarm Kit Instructions

Read this entire specification. Ask clarifying questions if anything is ambiguous. Make technology decisions where explicitly delegated (native vs Electron for Mac). Build for gold medal quality - this is infrastructure that will be used daily across four devices. Test thoroughly. Deliver deployment-ready code with clear documentation.

Remember: The goal is to replace Obsidian completely. If there's a feature Dino relies on that isn't in this spec, flag it before building.

End of Specification

Ready for Swarm Kit implementation.