Notion iOS Reverse Engineering Report

Name: Hussein Hussein
Student ID: 58301
Semester: 6th
Course: Team Project Task 1
GitHub Repository: https://github.com/Leo-devv/Notion-IOS-Reverse

Introduction

This report documents the reverse engineering and design system recovery of the Notion iOS mobile application. The goal of this task was to analyze a real app in depth, understand its structure, and document it in a way that would allow an AI coding assistant like Copilot or Cursor to recreate it based on my observations. Notion was chosen because it aligns closely with the kind of app I aim to design for my team project an interface based on nested content, flexible editing, and collaborative use.

App Concept and Purpose

Notion is an all-in-one workspace that blends note-taking, task management, and database functions into one seamless app. Users can create pages with customizable blocks, nest pages inside each other, and collaborate in real time with others. The interface is clean and minimalist but allows a high level of customization.

Primary use cases include:

- Personal note-taking and task planning
- Collaborative documentation in teams
- Internal wikis and project tracking

Target Users and Motivations

The app targets a wide spectrum of users:

- Students who want to organize their academic notes and assignments
- Freelancers and professionals looking for a lightweight project management tool
- Teams that want to centralize their documentation and collaborate in real time

Users choose Notion for its flexibility, clean design, and the ability to create everything from simple checklists to structured databases.

Functional Scope

The app includes the following main modules and screens:

- Login / Onboarding
- Workspace and sidebar navigation
- Rich-text block-based editor
- Search interface
- Notifications and mentions (Inbox)
- Page sharing and permissions
- Templates gallery
- Settings screen

Each screen has its own function but integrates smoothly into a unified workflow where users create, manage, and share content.

Core Screens (must-have):

- Login

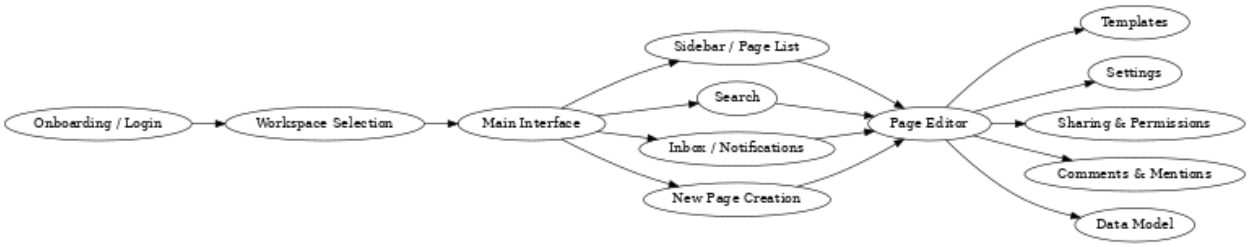
- Sidebar
- Page editor
- Search
- Sharing

Extended Screens (nice to have):

- Templates
- Comments and mentions
- Offline mode

Navigation and Structure

The navigation structure is a combination of a bottom-tab layout and a sidebar hierarchy. After login, the user selects or lands in a workspace. From there, the sidebar allows access to nested pages, while the bottom tab bar offers quick access to the editor, search, inbox, and creation tools.



Peer Feedback and Revision

After preparing the initial summary and navigation diagram, I shared the content with a peer. Their suggestion was to clearly define which screens belong in the MVP version of the app and which ones could be added later. I updated the report accordingly, creating

a clear split between core and extended features. This helped clarify scope and prioritize implementation order.

AI Prompts for Core Features

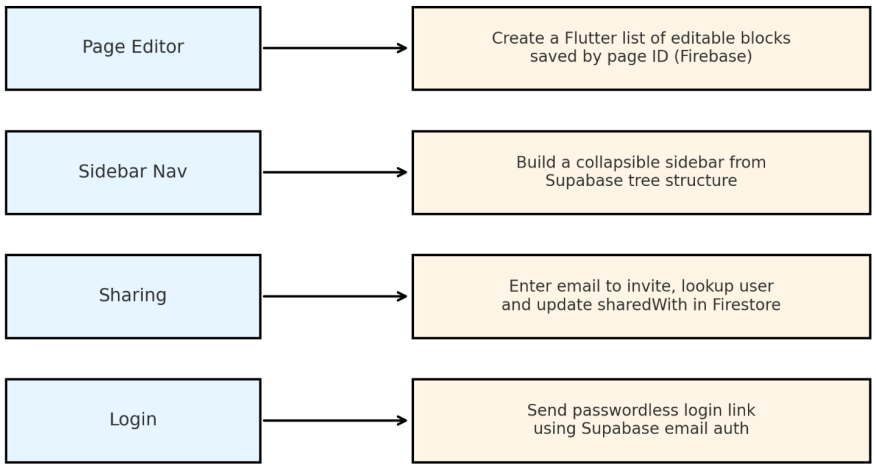
These are some example prompts I would give to GitHub Copilot to help me code core parts of the app:

1. Page Editor (Flutter + Firebase)
"Create a screen that displays a vertical list of editable text blocks. Each block should be a text field, and hitting Enter adds a new block below. Save to Firestore by page ID."

2. Sidebar Navigation (React Native + Supabase)
"Build a collapsible sidebar that displays nested pages using parent_id relationships. Clicking on a page loads its content."

3. Sharing Pages (Flutter + Firebase)
"Implement a screen to invite users to a page. Enter an email, look it up in Firestore, and add the user to the sharedWith list with view or edit access."

📎 Full file: `prompts/ai-prompts.md`
📎 Visual overview of prompt-to-feature mapping:



Implementation Effort and Planning

Each core screen was evaluated for implementation complexity:

Feature / Screen	Effort	Blockers or Dependencies	MVP?
Login & Authentication	Medium	Needs backend auth setup	Yes
Page Editor	High	Custom block logic, text input management	Yes
Sidebar Navigation	Medium	Tree structure for nested pages	Yes
Search	Medium	Indexed queries or filters	Yes
Sharing Pages	Medium	User lookup and permissions logic	Yes
Comments & Mentions	Medium	Comment UI and tag syncing	Later
Templates	Low	Mostly static or locally stored	Later
Offline Mode	High	Requires caching and sync engine	Later

📎 Visual file: `diagrams/complexity_estimation_table.png`

Screenshots

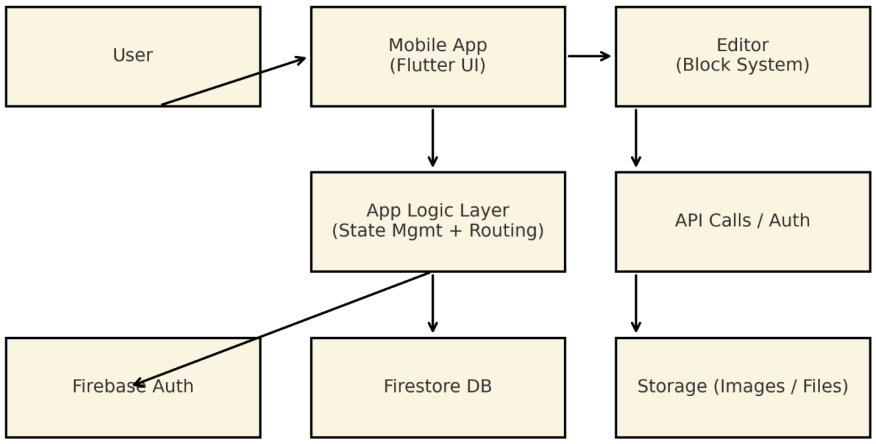
I took screenshots from the Notion iOS app to support the structure and visuals described:

- `login.png` – Login screen
- `sidebar.jpeg` – Sidebar and workspace browser
- `editor.jpeg` – Page editor interface
- `search.jpeg` – Search page
- `notes.png` – Notes / nested page view

📎 Folder: `/screenshots/`

System View

A basic architecture diagram was created to illustrate how the app interacts with backend services:



Reflection

This task helped me understand how to look at a real-world app not just as a user, but as a system. I learned to break it down into screens, flows, logic, and UI patterns. Writing prompts for an AI tool also made me think more clearly about what I expect from each part of the interface. It was especially useful to prioritize features for MVP and anticipate blockers before starting development. Peer feedback also helped focus the scope.

Final Notes

All files and supporting documents are hosted in the GitHub repository linked at the top. The structure and documentation here are designed to be used by both humans and AI tools as a blueprint for recreating the Notion iOS experience.

Thank you for reviewing this work.