ChatGPT

FounderOS Implementation Plan

Overview

FounderOS is envisioned as a unified operating system for our startup's leadership – a single source of truth in Notion that integrates with our project management (Linear) and communications (Slack/GitHub) tools. In **Step 1**, we completed the architecture blueprint and PR recommendations for FounderOS [1]. Now, in **Step 2**, we will implement this blueprint in Notion by creating the required pages and databases, setting up linked "Product Hub" views, and building a Founder Dashboard. This document outlines each step in detail, with an emphasis on clear structure and best practices.

Our goal is to enable the executive team (acting as the "CEO") to build out the Notion workspace according to the blueprint, with the "Chairman" (project sponsor) overseeing progress. We will preserve the exact database schemas as defined in the architecture blueprint and ensure the system is comprehensive yet easy to navigate.

Step 1: Create the FounderOS Main Page

First, create a top-level Notion page titled **FounderOS**. This page will serve as the home for the entire system – essentially a wiki/dashboard that contains or links to all other components.

- **Page Setup:** In the Notion workspace sidebar, click **New Page** and name it "FounderOS". Set it as a top-level page (for now, not inside any other page). This will be the central hub that everyone can access easily.
- **Paste Architecture Blueprint:** On this page, include the architecture overview (the blueprint from Step 1). You might paste the content of the `agents.md` or the PR summary directly here for reference. This ensures that the team has the full context and list of required databases/fields while building. For example, if the blueprint listed databases like **Products**, **Projects**, **Tasks**, etc., with their fields, paste that list in a section on this page. Consider using a toggle list or code block for clarity if it's in JSON/YAML format.
- **Access & Sharing:** Make sure the **FounderOS** page is shared with the relevant team members (full edit access for the exec team). This will likely reside in the appropriate Teamspace if your workspace uses team-specific sections. Everyone involved should be able to view and edit as needed.

Why this page? It acts as the container for the entire FounderOS. Think of it as the homepage of your internal operating system. From here, you will link out to the various databases and dashboards we create. It's also a good place to put high-level context, goals, or instructions for using the system.

Step 2: Create Core Databases (Schema Implementation)

Next, create the **core databases** in Notion as outlined in the architecture. According to the blueprint, we need to create each database **one-by-one**, using **exact field names and types** as listed. This ensures our

data schema is consistent with the plan. Below are the expected databases and their key fields (adapt these if your blueprint specified differently):

- 🗃 **Products** – A database to list each major product, project, or initiative. Create a new database (Table view) called "Products". Include fields such as:
- **Name** (title): Product name.
- **Description** (text): Short summary of the product or initiative.
- **Product Lead** (relation or person): Who is responsible (could be a Relation to a Team members DB or a Person property).
- **Status** (select): e.g. *Ideation*, *In Development*, *Launched*, *On Hold*, etc.
- **Launch Date/Timeline** (date): Important milestone dates.
- **Related Projects** (relation to Projects DB): This will link each product to all projects under it.
- 🗃 **Projects** – A database for projects (or epics) that roll up under products. Create "Projects" database with fields:
- **Name** (title): Project or initiative name.
- **Description/Goals** (text): What the project aims to achieve.
- **Status** (select or status): *Not Started*, *In Progress*, *Completed*, etc.
- **Owner** (relation or person): Who is driving this project (could link to Team member).
- **Product** (relation to Products DB): The product or program this project falls under.
- **Start Date** and **End Date** (date): Timeline for the project.
- **Related Tasks** (relation to Tasks DB): A backward link to all tasks in this project (Notion will allow linking tasks to projects via a Relation property [2]).
- 🗃 **Tasks** – A database to track to-dos, action items, and issues. Create "Tasks" with fields:
- **Name** (title): Brief task title or user story.
- **Details** (text): Description or requirements of the task.
- **Status** (status or select): e.g. *Backlog*, *In Progress*, *Blocked*, *Done*.
- **Priority** (select or number): e.g. P0, P1, P2… or High/Med/Low.
- **Due Date** (date): Deadline if applicable.
- **Assignee** (person or relation to Team): Who is responsible for this task.
- **Project** (relation to Projects DB): Which project this task belongs to.
- **Product** (relation to Products DB, optional**): If tasks can exist outside a specific project, link directly to Product. Otherwise, the Product can roll-up through the Project relation.

Note: Setting up Relations between these databases is crucial. For example, adding a "Project" relation on the Task table allows each task to link to its corresponding project, and vice versa [2]. This lets you later roll up progress (e.g., count of completed tasks per project). Notion's relation property *"allows you to connect pages from one database to another"* [3], enabling our Tasks ↔ Projects ↔ Products linkage.

- 🗃 **Team** (or **People/Members**) – A database for team members, useful for assigning tasks and denoting project leads. If not already present, create "Team" with fields:
- **Name** (title): Person's name.

- **Role/Title** (text or select): e.g. CEO, CTO, Engineer, Designer, etc.
- **Department/Team** (select): if needed (Engineering, Product, Sales, etc.)
- **Contact Info** (email/phone, if you want to include).
- (You can also simply use a built-in **Person** property on tasks for assignees, but a Team database helps if you want additional info per member and to relate other things to people.)

Database Creation Tips: Use the Notion **slash commands** or menu to create new databases: - To create a full-page database: On the **FounderOS** page, type "/database" and choose "New database – full page". Name it accordingly (e.g., Products). This will create a sub-page that is a database. Alternatively, create a page first then add a database inside it. - Enter the properties (fields) exactly as in the blueprint (pay attention to property types: Text, Number, Select, Multi-select, Relation, Date, Person, etc.). For Selects, pre-create the options listed in the plan (e.g., if Status field has certain categories, add them now). - For Relation fields: when you add a Relation property, Notion will prompt you to pick the other database to link to. Do this for linking Tasks→Projects, Projects→Products, etc. After linking, you can rename the relation field appropriately (e.g., in Tasks call it "Project"). Notion will automatically create a reciprocal relation in the other database – feel free to rename those as well (e.g., in Projects it might auto-add a "Tasks" property showing all related tasks). - Validate that each database's fields match the blueprint exactly. This structure is the backbone of FounderOS, so accuracy is key. You might want to test adding a sample entry in each to ensure the relations and select options work as expected.

Once these core databases are in place, you essentially have the **data layer** of FounderOS built. All other views (product hubs, dashboards, etc.) will be *views or pages* leveraging this data.

Milestone: *After creating and configuring all databases above, notify the Chairman that "Step 2 (Notion databases) is complete — ready to move to Step 3."* This checkpoint ensures the schema is reviewed and approved before we proceed to build on it. Clear communication here is important so that any adjustments to the structure can be made early.

Step 3: Create Product Hubs (Linked Database Views)

With the core databases ready, the next step is to create **Product Hubs** for each product/initiative. A product hub is a page (or set of pages) that gives a 360° view of everything related to a given product – by using **linked views** of the databases we just made. We will utilize Notion's ability to create linked database views to show filtered content relevant to each product.

Approach: We have a couple of options to implement product hubs: 1. **Within the Products Database:** Use the page of each product entry as its hub. Because each entry in the Products database is also a Notion page, we can open a product's page and add inside it any number of linked database views (filtered to that product). This is efficient and keeps everything tied to the product item itself. 2. **Separate Pages:** Alternatively, if a more customized layout is needed, we could create a new page for each product and manually link it to the product entry. However, using the product's own page is simpler and recommended.

We will proceed with option 1 for simplicity.

Steps to set up a Product Hub (for each product):
- **Create a template (optional):** In the **Products** database, create a **Database Template** page called something like "📊 Product Hub Template". This template will contain pre-configured linked views that we

want on every product's page. By doing this, whenever you add a new product, you can apply this template and all the linked views will be there automatically. - **Add linked views:** Within the template (or directly on an existing product page), add linked database views for Projects and Tasks: - On the product page, type "/linked" and select **Create linked database**. Choose the **Projects** database. Now you have a copy (view) of the Projects DB on this page. Rename this section "Projects – *[Product Name]*". Set a filter on this view: filter where the **Product** property == the current product. This will show only projects associated with this product. - Similarly, create another linked database on the product page for **Tasks**. Filter it to tasks where **Product** = current product (or tasks where Project's Product = current product – Notion allows filtered by related relations as well). This displays all tasks for the given product. - You can choose appropriate views for each: for example, Projects might be nicely viewed as a board by Status, or a timeline, whereas Tasks might be a simple list or board. Customize as needed. *Notion allows each linked view to have its own layout, filters, and shown properties without affecting the underlying database* [4]. - If there are other related databases (for example, if we had a **Bugs** or **Design Docs** database linked to Products), add those as well in a similar fashion. - **Verification:** After setting up the linked views and filters, the product page should now dynamically show all content relevant to that product. For instance, opening the page for "Product X" might display a list of all Projects under Product X and all Tasks for Product X, automatically updated. This leverages Notion's relational database power to keep everything in sync – editing a task's status or adding a new project in these views updates the master databases in real-time.

Repeat the above for each existing product in the Products database. If you used a template, you can simply apply the template to each product page to avoid manual repetition. Going forward, any new product can use this template to instantly spin up its hub.

Each **Product Hub** now provides a focused workspace for the team working on that product, while still feeding data to and from the central databases. Teams can even further customize their product-specific views (show or hide certain properties, etc.) without affecting the central data. In Notion, *"when teams set up their own linked views of the centralized tasks database, they can choose to show just the properties [relevant to them] on their main view"* [4] – our setup takes advantage of this, tailoring each hub to the product's needs.

Step 4: Build the Founder Dashboard

With products, projects, and tasks all tracked in Notion, we will create a **Founder Dashboard** – a high-level overview page for leadership (the Founder/CEO and executives) to monitor the most important information at a glance. This dashboard will pull together key views and metrics from our databases.

Create the Dashboard Page: On the main **FounderOS** page (or as a sub-page under it called "Founder Dashboard"), create a new page. This will contain a curated set of sections, such as:

- **Key Project Status Overview:** Insert a linked view of the **Projects** database, filtered to show active or priority projects across all products. For example, a table or board grouped by Status with only "In Progress" or "Blocked" projects could be useful. Include fields like Project Name, Product, Owner, % Tasks Done (you can create a rollup field in Projects to calculate this if desired).
- **Upcoming Deadlines / To-Dos:** Add a linked view of **Tasks**, filtered to tasks due in the next 7 days or high-priority tasks (Priority = P0/P1) that are not done. This acts as an "action list" for leadership attention. You might use a list view sorted by due date, showing Task name, Assignee, Project, and Due Date.

- **High-Level Metrics:** You can use Notion's rollup and formula properties to display some summary metrics. For instance, on the Dashboard you could create a few **Formula or Rollup blocks** (or even use the new Notion **Data visualization** if available) to show things like: total number of active projects, total open tasks, tasks completed this week, etc. These can be achieved by creating a formula property in databases or using a rollup to count related items. If these metrics were set up in the blueprint (e.g., an "Overall Status" or progress bar for projects), include them here.
- **Links/Navigation:** Provide quick links to other parts of the FounderOS. This can be a section with links to each **Product Hub** page (so the founder can jump into Product-specific details easily), as well as links to each core database (if they ever need to see the full tables). You can just paste links to the relevant pages or create a nicely formatted index.
- **Team Updates (Optional):** If desired, dedicate a spot for any recent updates or notes. For example, a linked view of a **Meeting Notes** database filtered to the latest entries, or a placeholder for weekly highlights (this could simply be a text section that the team updates manually).
- **Integration Highlights (Optional):** Given we plan to integrate with Slack/Linear/GitHub, the dashboard could also show any recent important items from those sources. For instance, if Linear issues are synced to the Tasks DB, then the tasks view already covers them. If not, perhaps embed a link or widget. Slack updates could be in a small embedded stream if Notion allows (or just a note: "Check #founder-os Slack for latest discussions"). This section can be fleshed out more once integrations are live, so it may be left blank or minimal in this phase.

Design the layout of the dashboard for clarity. Notion now has **layout customization** features where you can arrange content in columns and toggle full-width. Consider putting two views side-by-side if it makes sense (e.g., Projects status board on left, upcoming tasks list on right). Use headings and dividers to clearly label each section.

The Founder Dashboard should answer at a glance: *"What's the status of our key projects, and what needs attention right now?"* It's the bird's-eye view. By aggregating data from the underlying databases, it ensures the data is always up-to-date. For example, if a team member marks a task as done in the Tasks DB, it will automatically reflect in the dashboard's open tasks list or project progress. This real-time tracking is a major advantage of building on related Notion databases.

Step 5: Team Communication & Next Steps

Building FounderOS is an iterative process. Now that we've set up the structure in Notion (pages, databases, views, dashboard), it's crucial to keep stakeholders informed and plan subsequent integration steps:

- **Progress Check-in:** As noted, once all databases (Step 2) are created and configured, the team should **ping the Chairman** with an update: e.g., in Slack `#founder-os` channel, a message *"Step 2 (Notion databases) complete — ready to move to Step 3 (Product Hubs)."* This confirms the milestone and signals to proceed. Similarly, after finishing the product hubs and dashboard (Steps 3 and 4), update the Chairman that the Notion implementation is complete.
- **Review and Feedback:** The Chairman/executives should review the FounderOS page, the databases, and the dashboard. Ensure the fields and views align with expectations from the Step 1 blueprint. If any adjustments are needed (e.g., an extra field, a different categorization), address them now.
- **Training the Team:** Brief the executive team and any other users on how to use the new system. Since Notion might be new to some, show them how to add new projects or tasks, update statuses,

and use the dashboard. Emphasize that changes in one place reflect everywhere due to the relations (preventing duplicate data entry).

- **Next Steps – Integration:** With the Notion workspace set up as the "CEO's view", we can integrate our other tools to ensure data flows smoothly:
- *Linear:* Since engineering tasks are tracked in Linear, consider using Notion's API or a service (like Zapier or Notion's own integrations) to sync Linear issues into the Tasks database (or vice versa). This way, the founder can see engineering issues on the Notion dashboard without forcing engineers to double-update in Notion. Linear's integration documentation and Agents can help automate this.
- *Slack:* Leverage Slack notifications for important changes. For example, use the Linear→Slack integration to post when critical tasks change status, or Notion's Slack integration to notify the team when the dashboard is updated. We already have a dedicated `#founder-os` Slack channel – that can serve for updates and Q&A around this system.
- *GitHub:* If Step 1 involved a GitHub PR and we expect to track development progress, we might integrate GitHub to Linear (which is commonly done) so PRs link to Linear issues, which then reflect on our Notion tasks via the sync. Ensure the GitHub bot in Slack (as seen in Step 1) continues to post relevant updates for visibility.

These integration tasks likely fall under **Step 3 of the broader plan** (if Notion implementation was Step 2). We will handle them in a separate phase, potentially with assistance from the AI agent (Opus 4.1) that performed well in Step 1. For now, the Notion architecture is in place and can function independently.

- **Maintenance:** Assign an owner (perhaps the Head of Operations or similar) to maintain the FounderOS space. This includes periodically grooming the databases (archive old projects, ensure tasks are updated) and refining the dashboard as needs evolve. Notion makes it easy to adjust — for instance, if we find the dashboard cluttered, we can tweak filters or views without altering the underlying data.

Conclusion

By completing the above steps, we will have a robust **FounderOS** in Notion: all products, projects, and tasks are tracked with the exact schema from our architecture plan, each product has its own hub page summarizing its workstreams, and a top-level dashboard gives leadership an instant overview of the company's execution status. The system is designed to scale and integrate: - The use of Relations and Linked Databases ensures data consistency and avoids silos (e.g., *linking tasks to projects means updates reflect in both contexts* [2]). - Custom views (product hubs, team-specific filters) allow each team to focus on their slice of information without losing the big picture [4] . - The structure lays the groundwork for deeper integration with development and communication tools, uniting our work across Linear, Slack, and GitHub into this single source of truth.

Moving forward, as we connect these integrations and refine the process, FounderOS will become an indispensable asset for running the company. It will enable the CEO (and all executives) to monitor progress, identify bottlenecks, and make informed decisions in real-time, while the Chairman and board can be given access to view high-level dashboards as needed.

Next Action: The team should proceed with creating the Notion page and databases (Steps 1 & 2). Once done, confirm completion to the Chairman and continue with setting up product hubs (Step 3). With diligent execution, we expect to have the core FounderOS implemented in Notion shortly, paving the way for the

subsequent integration phase and ensuring everyone is on the same page – literally and figuratively – about our projects and priorities.

---

[1] alex (real_name: Alex Alaniz; username: alex): *FounderOS Step 1 PR Comparison and Recommendation*
https://bearifiedco.slack.com/archives/C09TLPR53DF/p1763275618277649

[2] [3] Using relation & rollup properties
https://www.notion.com/help/guides/using-relation-and-rollup-properties

[4] Build the perfect workflow with customizable layouts
https://www.notion.com/help/guides/build-the-perfect-workflow-with-customizable-layouts