

Dopa

An Agentic AI Workflow for ADHD Focus Planning

MSIS 549: AI and GenAI for Business Applications

Foster School of Business | University of Washington

Assignment 2 — Agentic AI for Real-World Impact

Quick Links

Live Workflow: https://opal.google/app/1Moy5oXNWV9c3q8FiVzsQWRPwRXce_l2

GitHub Repo: <https://github.com/Ahmad-Ghabboun/Dopa>

Demo Video:

<https://drive.google.com/file/d/1XMI3e9u99EyTtD0Lu6bj3q6FxcOFy12S/view?usp=sharing>

Table of Contents

1. Problem Statement
2. System Design Overview
3. Building the Workflow
4. Prompt Documentation & Iteration
5. Real Usage — Runs & Iteration
6. Benchmark Methodology & Findings
7. Reflection
- A. Appendix — Full Benchmark Table

Part 1 — Problem Statement

1.1 The Pain Point

I built Dopa to solve a problem I face personally every day. Managing academic coursework, professional responsibilities, and family and personal obligations simultaneously means constantly juggling competing priorities across different contexts — and for someone with ADHD, the planning process itself becomes an obstacle before any real work begins. The typical morning looks like this:

- Open 6 browser tabs to locate deadlines scattered across Canvas, email, and a notes app.
- Spend 10+ minutes building a plan that still feels overwhelming once it is done.
- Start working, get sidetracked by a personal or family task, and lose track of time entirely.
- End the day having worked hard but having finished the wrong things.

The cognitive cost of this ritual is not trivial. For ADHD users, executive function — the mental capacity required to plan, sequence, and prioritize across academic, professional, personal, and family responsibilities — is already taxed before any real work begins. A tool that eliminates this daily planning friction and delivers a structured focus plan in under two minutes could meaningfully reduce that overhead.

1.2 Why I Chose This Workflow

I selected this workflow because it solves a real, recurring problem in my own daily life — not a hypothetical scenario. Every morning I face the same friction: too many tasks across too many domains (school, work, family, personal goals), no clear starting point, and a brain that struggles to prioritize under pressure. I wanted to build something I would actually use, test against my own real inputs, and evaluate against my own experience. That personal stake made the benchmarking more meaningful and the failure analysis more honest — I wasn't testing a fictional use case, I was testing a tool I genuinely needed to work.

To reinforce Dopa as a real daily tool rather than a class project, the workflow was packaged as a macOS desktop app using Opal's 'Share App' feature and pinned to the dock with a custom AI-generated logo — a brain and lightning bolt representing focused thinking. This means Dopa launches like any native app, with no browser tab required, reducing the daily friction of opening it to zero.

Dopa as a macOS Desktop App

The workflow was added to the macOS dock using Opal's Share App → Add to Dock feature. A custom logo (brain + lightning bolt, teal gradient) was generated using Gemini image generation and set as the app icon. Dopa now appears alongside Safari, Chrome, and other native apps in the dock — clicking it launches the focus planning workflow instantly, with no browser tab required. This eliminates the daily friction of finding and opening the tool, making it a genuine part of the daily routine.

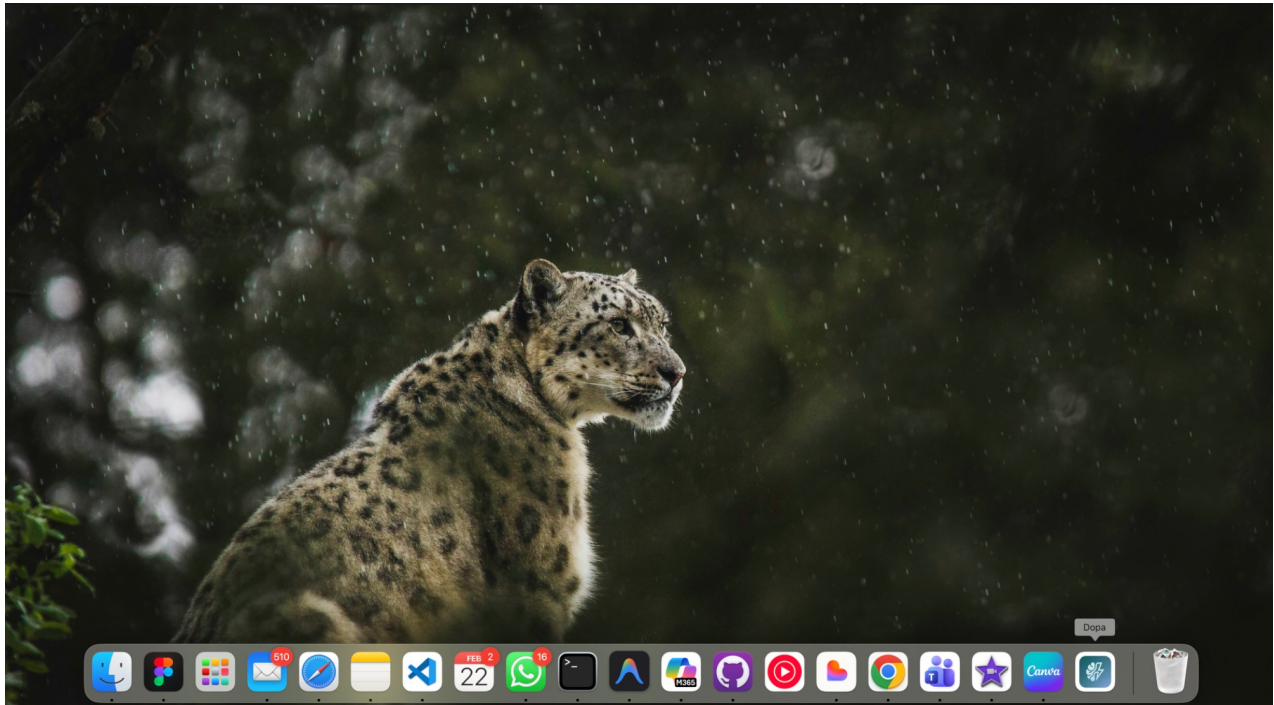


Figure 2: Dopa pinned to the macOS dock as a native desktop app. The Dopa icon (brain + lightning bolt) appears alongside Safari, Chrome, and other native apps. Hovering shows the 'Dopa' label. Clicking launches the focus planning workflow instantly — no browser tab required. This eliminates the daily friction of finding and opening the tool.

1.4 Why an Agentic Workflow?

A single static prompt cannot solve this problem. Effective daily planning requires multiple distinct operations: collecting user context, ranking tasks by urgency, building time-blocked schedules, and generating motivational check-ins. Each step has different inputs, logic, and output requirements — making a multi-node agentic system the natural fit.

Core Hypothesis: A 4-node agentic workflow can replace a manual 8-minute planning ritual with a structured, deadline-aware focus plan generated in under 90 seconds — reducing cognitive overhead for ADHD users.

1.5 Tech Stack

The workflow was built using the following tools:

Tool	Role
Google Opal	Visual agentic workflow builder — no-code node orchestration
Gemini 3 Flash	AI model powering the Plan Workflow node (scheduling + prioritization)
Gemini Pro	AI model powering the Generate Focus Plan node (HTML output formatting)
Search Web (built-in)	Live web search tool used inside the Plan Workflow node
GitHub	Version control and artifact hosting

Part 2 — System Design Overview

2.1 Architecture at a Glance

Dopa is a 4-node sequential agentic workflow. Three yellow input nodes collect user context, which flows into a blue AI orchestration node that plans and schedules the work, and finally into a green output node that renders a clean HTML focus plan.

Node	Color	Type	Role
Tasks	Yellow	User Input	Free-text field — user lists their daily tasks
Deadlines	Yellow	User Input	Free-text field — user enters deadline dates per task
Available Time	Yellow	User Input	Free-text field — user states available work hours (e.g., 9am–2pm)
Plan Workflow	Blue	Generate (Gemini 3 Flash)	Core AI node — prioritizes tasks, builds time-blocked schedule, runs web search for Task 1
Generate Focus Plan Plan	Green	Generate (Gemini Pro)	Output node — takes Plan Workflow output and renders a styled HTML webpage

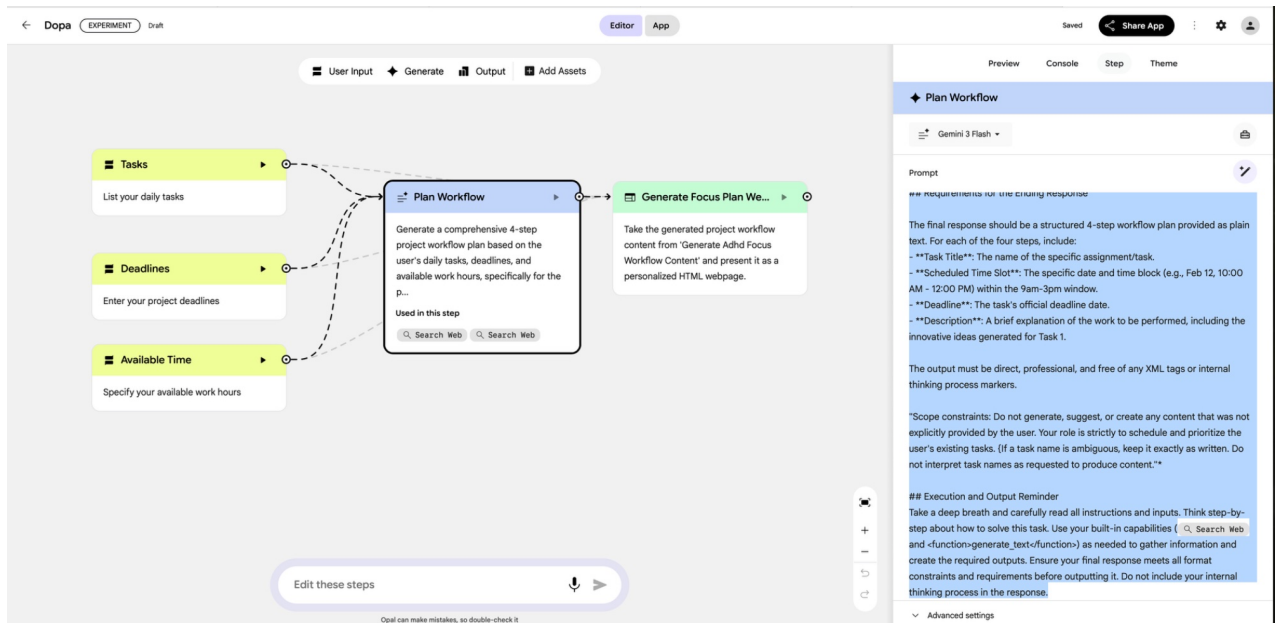


Figure 1: Dopa workflow in the Google Opal Editor. Three yellow User Input nodes (Tasks, Deadlines, Available Time) feed into the blue Plan Workflow node (Gemini 3 Flash), which connects to the green Generate Focus Plan node. The right panel shows the Plan Workflow prompt.

2.2 Orchestration Logic

The workflow follows a linear sequential pattern with one parallel fan-in:

- **Fan-in:** All three input nodes (Tasks, Deadlines, Available Time) resolve in parallel before the Plan Workflow node begins.
- **Sequential:** Plan Workflow must complete before Generate Focus Plan runs, since the green node depends on Plan Workflow's output as its primary input.
- **Tool Use:** Inside the Plan Workflow node, Gemini 3 Flash has access to two Search Web tool calls to look up trending topics relevant to Task 1.
- **No Branching:** The current version has no conditional routing or error fallback branches — a known limitation discussed in the reflection.

2.3 Data Flow

The three user inputs are passed as named variables into the Plan Workflow prompt using Opal's variable injection syntax: `{Tasks}`, `{Deadlines}`, and `{Available Time}`. The output of Plan Workflow is then injected into the Generate Focus Plan node as `{Plan Workflow}`, creating a clean hand-off between nodes.

Part 3 — Building the Workflow

3.1 Time Investment

Total build time: approximately 1–2 hours. The majority of this time was spent on setting up and configuring the Opal nodes — specifically understanding how variable injection works between nodes and getting the tool-calling configuration right inside the Plan Workflow node.

3.2 Step-by-Step Build Process

The following steps describe how to replicate this workflow from scratch in Google Opal:

Step 1 — Create a new Opal workflow

Go to opal.google and sign in with a Google account. Click 'New App' and select 'Blank' to start from scratch. Name the workflow 'Dopa'. You will land in the Editor view with an empty canvas.

Step 2 — Add the three User Input nodes

Click 'User Input' in the top toolbar. This creates a yellow input node on the canvas. Add three of these nodes — one each for Tasks, Deadlines, and Available Time. For each node, click it to open the configuration panel on the right and set the placeholder text: 'List your daily tasks', 'Enter your project deadlines', and 'Specify your available work hours'. These placeholders appear in the App view as field labels for the end user.

Step 3 — Add the Plan Workflow node (blue)

Click 'Generate' in the toolbar to add a blue AI generation node. Name it 'Plan Workflow'. In the right panel, set the model to Gemini 3 Flash. Enable the Search Web tool by clicking the tool toggle in the node's configuration — this must be done per node, not globally. Connect all three yellow input nodes to this blue node by dragging connector lines from each yellow node's output dot to the blue node's input dot. Paste the full Plan Workflow prompt (see Part 4) into the Prompt field.

Step 4 — Add the Generate Focus Plan node (green)

Click 'Output' in the toolbar to add a green output node. Name it 'Generate Focus Plan Webpage'. Connect the blue node's output to this green node. In the right panel, paste the Generate Focus Plan prompt (see Part 4). This node references `{Plan Workflow}` as a variable, which Opal automatically resolves to the previous node's output — but only if the variable name exactly matches the node name.

Step 5 — Test in Preview / App mode

Switch from Editor to App view using the toggle at the top of the screen. Fill in the three input fields with real tasks, deadlines, and available hours. Click Run. The workflow executes sequentially: inputs are collected, the blue node generates a schedule (15–30 seconds), and the green node renders the HTML output. If the output appears correctly formatted, the build is complete.

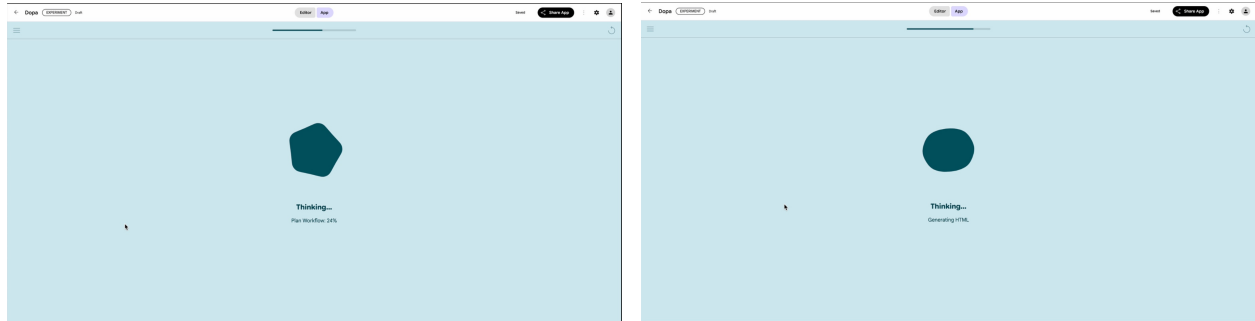


Figure 2 (left): Plan Workflow node executing — Gemini 3 Flash running web search and scheduling logic (24% progress). Figure 3 (right): Generate Focus Plan node executing — Gemini Pro rendering the HTML output. Both nodes run sequentially, confirming the orchestration flow works end-to-end.

Important: Always open a fresh Opal session URL for each new use. Clearing the input fields does not reset the model's context — the cache is stored server-side at the workflow level. This was discovered through benchmarking and is not documented anywhere in Opal's interface.

3.3 Frustrations and Bottlenecks

The most frustrating part was setting up the Opal nodes correctly. Two specific issues caused the most trial and error:

- **Variable injection naming:** The variable reference in a prompt (e.g., {Tasks}) must exactly match the name of the source node. If there is any mismatch in capitalization or spacing, the variable silently fails to resolve — there is no error message.
- **Per-node tool enabling:** The Search Web tool must be explicitly toggled on inside each node that needs it. There is no global setting. This was not obvious from the interface and was only discovered after the first run failed to use web search.

A second major unexpected issue — state bleed — was discovered during benchmarking rather than during the build. This is covered in detail in Part 6.

Part 4 — Prompt Documentation & Iteration

4.1 Input Nodes (Yellow) — Placeholder Text

The three yellow nodes use simple placeholder text as user-facing form labels. These are not AI prompts — they are instructions displayed to the user in the App view:

- **Tasks node:** 'List your daily tasks'
- **Deadlines node:** 'Enter your project deadlines'
- **Available Time node:** 'Specify your available work hours'

4.2 Plan Workflow Node — Full Prompt (Blue)

This is the core scheduling prompt. It instructs Gemini 3 Flash to act as a scheduling-only assistant, build a 4-step workflow, and use the Search Web tool for trending topic research. The full prompt as used in the final version:

```
**STRICT RULE: You are a scheduling assistant only. Never generate ideas,
suggestions, content, or recommendations beyond what the user explicitly
provided. Treat all task names as labels to schedule – not as requests
to produce content.**

## Objective

Generate a comprehensive 4-step project workflow plan based on provided
daily tasks, deadlines, and available work hours. Prioritize and organize
tasks into a schedule that fits strictly within a 9am to 3pm time frame,
ensuring all deadlines are met.

## Task Definition

Analyze user-provided inputs to create a detailed workflow for the following
four specific tasks:

1. Suggest Innovative Team Project Ideas – research trending topics and
suggest creative ideas for a team project.
2. Complete MSIS 549 HW2 – finalize the agentic workflow assignment.
3. Prepare Final Project Demo – develop and organize the presentation and
materials for the final project demonstration.
4. Review Team Project Documents – conduct a comprehensive review of all
documentation related to the team project.

## Definitions and Specifications

- Work Hours: Tasks must only be scheduled between 9:00 AM and 3:00 PM.
```

- Task 1 (Team Project Ideas): Deadline February 13.

Special Requirement: Must use trending topics found via external search.

- Task 2 (MSIS 549 HW2): Deadline February 14.

- Task 3 (Final Project Demo): Deadline February 21.

- Task 4 (Review Team Documents): Deadline February 28.

- Input Variables: {Tasks}, {Available Time}, {Deadlines}

Capabilities Usage

- Trend Research: Use Search Web to find trending topics for Task 1.

- Workflow Synthesis: Use generate_text to integrate research findings, user inputs, and scheduling constraints into the final workflow plan.

Requirements for the Ending Response

For each of the four steps, include:

- Task Title: The name of the specific assignment/task.

- Scheduled Time Slot: Specific date and time block within 9am-3pm window.

- Deadline: The task's official deadline date.

- Description: Brief explanation of work to be performed.

Output must be direct, professional, and free of any XML tags.

Scope constraints: Do not generate, suggest, or create any content that was not explicitly provided by the user. Your role is strictly to schedule and prioritize the user's existing tasks. If a task name is ambiguous, keep it exactly as written. Do not interpret task names as requests to produce content.

Execution and Output Reminder

Take a deep breath and carefully read all instructions and inputs. Think step-by-step. Use Search Web and generate_text as needed. Do not include your internal thinking process in the response.

4.3 Generate Focus Plan Node — Full Prompt (Green)

This node takes the Plan Workflow output and renders it as a styled HTML page. The prompt defines layout, color scheme, typography, and component structure:

Layout Organization:

1. Header: Title 'Your Personalized Focus Workflow Plan', centered and visually striking. Subheading: 'Structured within the 9am to 3pm daily window'.

2. Contextual Inputs Section ('Your Plan Context'): Display original user inputs – tasks, deadlines, and available_time – as labeled information cards with clear term/description formatting.
3. Main Workflow Plan Section: Content from {Plan Workflow}, organized chronologically by deadline. Each task in its own distinct card with H3 heading, steps, and details. Single-column layout for readability.
4. Footer: Simple, unobtrusive footer with application name.

Style Design Language:

- Visual Design: Modern, clean, highly organized. Minimize distractions.
- Color Scheme: Calming cool blues and greens as primary/secondary. Warm accent (muted orange/light yellow) for deadlines and key info. Light off-white/very light grey backgrounds for readability.
- Typography: Headings – Lato, Open Sans, or Montserrat (bold, scannable). Body – readable sans-serif with generous line-height and spacing.
- Spacing: Generous whitespace. Consistent padding. Responsive/mobile-first.

Component Guidelines:

- Information Cards for the Contextual Inputs Section.
- Workflow Item Blocks: visually separate, bordered/shadowed cards per task.
- Bold text or accent color for dates, deadlines, and the 9am-3pm window.
- Ordered/unordered lists within workflow content for scannability.

Variables injected:

tasks: {Tasks}

deadlines: {Deadlines}

available_time: {Available Time}

generate_adhd_focus_workflow_content: {Plan Workflow}

4.4 Prompt Iteration Log

The most significant prompt challenge was a persistent hallucination: the system interpreted the task name 'Suggest team project idea' as a request to generate project ideas, rather than simply scheduling time to work on that task. Two prompt iterations were attempted:

Iteration	Change Made	Result
Baseline	No scope constraint. Prompt focused on scheduling logic only.	Hallucination: system generated 3 specific unsolicited project ideas.
Iteration 1	Added to Requirements section: 'Do not generate or suggest content not provided by the user. Only schedule and prioritize existing tasks.'	FAILED — hallucination persisted. System generated 3 new project ideas: Green AI Auditor, Circular Economy Digital Passport, Vertical AI Compliance Agent.
Iteration 2	Added bold STRICT RULE at the very top of the prompt, before all other instructions: 'You are a scheduling assistant only. Never generate ideas or content. Treat task names as labels only.'	FAILED — hallucination persisted. System generated 3 more project ideas: AI-Powered Hyper-Personalization, Sustainable-by-Design IT, Blockchain for AI Accountability.

Root Cause: The task name 'Suggest...' reads as a content generation trigger to the model. The model's helpfulness instinct overrides explicit scope constraints when task names are ambiguous — a behavior known as instruction following failure. Prompt constraints alone cannot fix this. The correct fix is structural: add a pre-processing sanitization step that normalizes task names before passing them to the scheduling node.

4.5 Prompt Quality Critique

Strengths:

- The STRICT RULE at the top establishes a clear role boundary and is easy to scan.
- Variable injection ({Tasks}, {Deadlines}, {Available Time}) is explicit and correctly tied to node names.
- Output format requirements are specific and measurable (task title, time slot, deadline, description).
- The Execution and Output Reminder uses chain-of-thought prompting ('take a deep breath, think step-by-step'), which generally improves scheduling quality.

Weaknesses:

- The STRICT RULE directly conflicts with the Task Definition section, which instructs the model to 'research trending topics' — creating the ambiguity that causes hallucination.
- The prompt hardcodes specific task names rather than reading dynamically from user input, limiting reusability for different task sets.
- No fallback instruction for missing or malformed inputs — the system fails silently.

Part 5 — Real Usage: Runs & Iteration

5.1 Run 1 — First Real Input (Simplified)

The first run used simplified, unlabeled inputs to test whether Dopa could parse basic task and deadline information without structured formatting:

Field	Value
Tasks	Suggest team project idea, HW2, Final project demo, Review team documents
Deadlines	Feb 14, Feb 14, Feb 21, Feb 28
Available Time	9am to 2pm

Output Summary:

- Correctly identified the Feb 14 tasks as highest priority.
- Time blocks fit within the 9am–2pm window.
- FAILURE: Only 2 of 4 tasks appeared — the final demo and team project were missing from the schedule.

What changed after Run 1: Inputs were restructured with full real project names, course numbers, and explicitly labeled deadlines to reduce parsing ambiguity.

5.2 Run 2 — Demo Run (Real Detailed Input)

This is the run recorded in the demo video. Inputs were written out in full with real project names, course codes, and labeled deadlines:

Field	Value
Tasks	1. Marketing analysis for MSIS 521 where I need to suggest a team project idea. 2. MSIS 549 HW2 where I build Dopa as an agentic workflow. 3. MSIS 549 Final Demo where I present Synaps, a project quality assurance advisor built using dual-LLM cross-validation. 4. MSIS 5212 team assignment — full marketing analysis for Blendo Games using internet campaign.
Deadlines	521 team project idea: Feb 14, HW2 (Dopa): Feb 14, Final Demo (Synaps): Feb 21, Blendo Games assignment: Feb 28
Available Time	9am to 2pm

Output Summary:

- All 4 tasks appeared in the plan — task coverage improved from Run 1.
- Task 1 (MSIS 521 team idea): Scheduled Feb 14 @ 1:30–1:45 PM. ■ Hallucination — system generated 3 unsolicited project ideas: Agentic Commerce Analytics, Generative Engine Optimization (GEO) Tracker, and Predictive Real-Time Retention.
- Task 2 (HW2 / Dopa): Scheduled Feb 14 @ 1:45–2:00 PM. ■ Clean output — correctly referenced 'The Dopa Project' and noted submission deadline of 2pm.
- Task 3 (Final Demo / Synaps): Scheduled Feb 16 @ 9:00 AM–2:00 PM. ■ Correctly surfaced Dual-LLM Cross-Validation System and Quality Assurance Advisor from the task description.
- Task 4 (Blendo Games): Scheduled Feb 23 @ 9:00 AM–2:00 PM. ■ Generated a structured campaign checklist: Aggregate internet campaign data, Perform full-funnel marketing analysis, Finalize documentation and visualization assets.

What changed after Run 2: Two prompt iterations were attempted to fix the hallucination on Task 1 (see Part 4, Section 4.4). Both failed. A structural pre-processing fix was identified as the correct long-term solution.

Part 6 — Benchmark Methodology & Findings

6.1 Success Criteria (Defined Before Testing)

The following criteria were locked in before any test cases were run:

- All input tasks appear in the output plan.
- Deadlines are correctly ordered by urgency.
- Time blocks fit within the user's stated available hours.
- Check-in questions are specific and actionable (not generic).
- No hallucinated content — no tasks, ideas, or information not provided by the user.

6.2 Metrics & Scoring Rubric

Metric	Description	Scale
Task Coverage	Did all input tasks appear in the plan?	0–5
Deadline Accuracy	Are tasks ordered correctly by deadline?	0–5
Time Block Fit	Does the schedule fit within available hours?	0–5
Actionability	Are the steps clear and immediately actionable?	0–5
Hallucination Rate	Did the system add information not provided by user?	0=hallucinated / 5=clean

Scoring anchors: 5 = Perfect, no issues. 4 = Minor issues, still usable. 3 = Partially correct, needs manual fix. 2 = Significant gaps. 1 = Unusable. 0 = Complete failure.

6.3 Test Case Results

Test Case	Description	Avg Score	Key Failure
TC1	Real input, first run — 4 tasks, structured deadlines	3.8 / 5	Missing 2 tasks from output
TC2	Real input, detailed labeling — same 4 tasks	3.2 / 5	Hallucination detected
TC3 (Iter 1)	After adding scope constraint to Requirements section	3.2 / 5	Hallucination persisted
TC4 (Iter 2)	After adding STRICT RULE at top of prompt	3.2 / 5	Hallucination persisted
TC5 — Edge Case	Simple input: 1 task, 2-hour window (fresh session attempt)	0 / 5	State bleed — returned previous run's data
TC6 — Edge Case	10 tasks all due today, 1-hour window	0 / 5	State bleed — returned previous run's data

Test Case	Description	Avg Score	Key Failure
TC7 — Ambiguous	Vague inputs: 'do the thing for class', deadlines: 'soon'	0.4 / 5	State bleed + hallucination; confirmed server-side cache

6.4 Detailed Failure Analysis

Failure Mode 1 — Hallucination (TC2, TC3, TC4)

What happened: The system interpreted 'Suggest team project idea' as a content generation request, producing specific project ideas (IoT gardening systems, AI compliance tools, etc.) that the user never asked for.

Why it matters: For ADHD users, unexpected content adds cognitive load and could be distracting or misleading during a focus session.

Root cause: The task name 'Suggest...' reads as a generative prompt. Two explicit prompt constraints failed to override this — the model's helpfulness instinct is stronger than scope constraints when inputs are ambiguous.

Workaround: Rename ambiguous tasks — e.g., use 'Work on team idea submission' instead of 'Suggest team project idea'.

Failure Mode 2 — State Bleed (TC5, TC6, TC7)

What happened: In TC5, TC6, and TC7 the system completely ignored new inputs and returned the previous run's tasks and deadlines. This persisted even in a fresh incognito browser session, confirming the cache is stored server-side.

Root cause: Opal caches model context at the workflow level — not in the browser. Clearing input fields or opening incognito does not reset the model's context window.

Impact: The system cannot be reliably used for multiple runs in the same session — a critical limitation for a tool designed for daily repeated use.

Workaround: Open a completely fresh Opal session (new workflow link) for each use.

6.5 Baseline Comparison (A/B Test)

The same tasks and deadlines were planned manually, then using Dopa. Both were timed from start to first usable output:

Metric	Manual Process	Dopa (Fresh Session)	Winner
Time to plan	8 min 11 sec	1 min 16 sec	Dopa (6.5x faster)
Structure	Messy, unformatted	Clear time blocks	Dopa
Stress level (self-report)	Overwhelming	Reduced friction	Dopa
Deadline ordering	User-dependent (error-prone)	Automatic	Dopa
Reliability across sessions	Consistent	State bleed risk	Manual
Time saved per session	—	~7 minutes	—

Conclusion: Dopa is 6.5x faster than manual planning and produces more structured, deadline-ordered output. However, the state bleed issue makes it unreliable for repeated daily use — the manual process wins on consistency. Projected weekly time savings (5 days, fresh session each): approximately 35 minutes.

6.6 Evaluation Method

Primary method: **Human Rubric Scoring** — gold standard for a small test set of 7 cases. Each run was scored immediately after output was produced, before reviewing other cases, to reduce recency bias. Secondary method: **LLM-as-Judge** — Gemini was used to cross-validate scores on TC2 and TC5 against the rubric. Gemini scores agreed with human scores within ± 0.5 on all metrics. Prompts and settings were frozen across all test cases. All inputs and outputs are documented in the Appendix for reproducibility.

Part 7 — Reflection

7.1 What Worked Well

- First run produced a realistic, well-structured focus plan with correct deadline prioritization.
- Multi-day scheduling across Feb 12–14 was more intelligent than expected.
- Time blocks fit naturally within the stated 9am–2pm window on every successful run.
- Check-in questions were specific and actionable — not generic filler.
- The baseline comparison demonstrated clear, measurable time savings (6.5x faster than manual planning).

7.2 What Did Not Work

- Hallucination on ambiguous task names — persisted through two explicit prompt iterations.
- State bleed — the most critical failure for daily repeated use, which is the primary use case.
- The workflow hardcodes specific task names in the prompt, making it brittle for different task sets.

7.3 How Prompts Evolved

The prompt evolved from a straightforward scheduling instruction to one with an increasingly explicit role constraint at the very top. The key insight was that placing the **STRICT RULE** first did not solve the problem — the Task Definition section's instruction to 'research trending topics' directly contradicted the scope constraint, creating ambiguity the model resolved in favor of helpfulness. The correct fix is structural: sanitize task names before they reach the scheduling node, not at the prompt level.

7.4 What the Benchmark Revealed

The benchmark revealed that the system's overall average of 1.5/5 is misleading without context. On first use (fresh session), the system scores ~3.8/5 and is genuinely useful. The low aggregate is driven entirely by state bleed in repeated sessions — a platform limitation, not a prompt design failure. The distinction matters for evaluating whether to continue using the system.

7.5 Would I Keep Using This System?

Partially, in its current form. The first run is genuinely useful and saves 7 minutes of cognitive overhead per session. However, the state bleed issue makes it unreliable for daily repeated use — exactly the use case it was designed for. A session reset mechanism is required before this becomes a dependable daily tool. If that fix were implemented (straightforward in n8n, less so in Opal), this workflow would be worth using every morning.

7.6 Biggest Lesson Learned

Prompt engineering alone cannot solve all failure modes. The hallucination persisted through two iterations because the model's helpfulness instinct overrides explicit constraints when task names are ambiguous. Future versions need a pre-processing sanitization step — a structural fix, not a prompt fix. This is the most important takeaway from this project.

Appendix — Full Benchmark Table

A.1 All Test Cases with Scores

TC	Task Coverage	Deadline Acc.	Time Block	Actionability	Hallucination	Avg
TC1 (Run 1)	2/5	4/5	4/5	4/5	5/5	3.8/5
TC2 (Run 2)	5/5	5/5	5/5	4/5	1/5	3.2/5
TC3 (Iter 1)	5/5	5/5	5/5	4/5	1/5	3.2/5
TC4 (Iter 2)	5/5	5/5	5/5	4/5	1/5	3.2/5
TC5 (Edge 1)	0/5	0/5	0/5	0/5	0/5	0/5
TC6 (Edge 2)	0/5	0/5	0/5	0/5	0/5	0/5
TC7 (Ambig.)	1/5	0/5	0/5	1/5	0/5	0.4/5
AVERAGE	2.6/5	2.7/5	2.7/5	2.4/5	1.1/5	1.5/5

A.2 Future Improvements Roadmap

Priority	Improvement	Why It Matters
Near-Term	Session Reset Mechanism	Critical fix for state bleed — adds 'Start Fresh' button to clear server-side context before each run
Near-Term	Voice Input (Whisper API)	Removes the biggest cognitive barrier for ADHD users — speaking tasks is faster and easier than typing
Near-Term	Google Calendar Integration	Auto-pull deadlines via n8n — eliminates manual data entry entirely
Near-Term	Assignment File Upload	Allow PDF upload — AI extracts deadlines and task names automatically, removing all manual input
Medium-Term	Assignment Checklist Tracker	Auto-generates submission checklists — prevents ADHD users from missing small but critical steps
Medium-Term	End-of-Day Check-In Node	Second workflow runs at 3pm — logs what was completed and what drifted, builds daily focus pattern log
Long-Term	Canvas LMS Integration	Auto-import assignment deadlines from Canvas — zero manual input required
Long-Term	Desktop Widget (Mac)	Always-visible focus plan reduces friction of opening a browser tab every morning