AgentLab — Flowchart Overview

---

1 ▪ High-Level Agent Roles + Primary Tools

Role	Core Responsibilities	LLM Prompt "Persona"
Student CEO (Human)	Sets vision, approves all outputs	n/a
Product Manager Agent	Frame problem, craft user stories, prioritise backlog	"Vision-driven, detail-oriented PM"
Design Agent	Wireframes, UX copy, accessibility checks	"Human-centred product designer"
Development Agent	Generate code scaffolds, API contracts, tests	"Pragmatic full-stack engineer"
AI / Data Agent	Prompt design, ML pipeline, embeddings search	"LLM & data science architect"
DevOps Agent	CI/CD, containerisation,	"Automation-first DevOps lead"

Role	Core Responsibilities	LLM Prompt "Persona"
	environment configs	
Marketing / Growth Agent	Positioning, pitch deck, GTM OKRs	"Story-teller growth hacker"
Content Writer Agent	Blog posts, release notes, reflection doc	"Conversational technical writer"

Tip → Keep prompts modular so you can swap GPT-4o, Claude, or HuggingFace models without rewriting logic.

2 ▪ Interaction Flow (text version)

1. **Student enters idea ➜** routed to **PM Agent**.
2. PM asks clarifying Qs; writes *initial PRD* into memory store.
3. Design Agent consumes PRD ➜ returns Figma wireframe links.
4. Student approves wireframes; PM finalises *MVP user stories*.
5. Development Agent scaffolds repo & OpenAPI spec; pushes to GitHub.
6. AI/Data Agent inserts prompt-wrapper modules & vector DB schema.
7. DevOps Agent spins up preview on Fly.io; URL posted back.
8. Marketing Agent drafts landing-page copy & slide deck.
9. Student reviews; loops back to the relevant agent for revisions.
10. When " MVP accepted" flag set, Content Writer posts recap article.

Communication is mediated by an **Orchestrator Router** that:

- tags messages with role, thread_id, priority
- enforces max 3-turn cycles before requiring student approval.

3 ▪ Memory Management

Layer	Contents	Persistence	Retrieval	Extra Notes
Ephemeral	Last 10 dialogue turns / agent	In-RAM	Concatenate directly into the next LLM prompt	Cheap & fast; cleared when server restarts.
Session Summary	Agent-written bullet digest every ~20 turns	Commit a JSON file to `/sessions/YYYY-MM-DD-HH-MM.json` on each checkpoint	Use GitHub REST API → fetch the latest file by filename pattern	Optionally trigger commit via a GitHub Action or simple `git` CLI call.
Long-Term	**Key-value**: decision-log, design-rationale, pivots	1. Store raw notes in `/archive/…/*.json`. 2. Nightly GitHub Action generates `vector_index.parquet` (or `.sql`) and commits it.	Pull `vector_index.*`, load into local FAISS / pgvector, run cosine-similarity search, then fetch matching	Keeps repo readable while still enabling semantic search.

Layer	Contents	Persistence	Retrieval	Extra Notes
			raw JSON blobs.	
Reflection Log	Weekly “what went well / next” self-evaluation	Markdown files under /reflection/week-NN.md	Download raw file via GitHub CDN	Great for peer review & grading.

Garbage collection.
Add a scheduled GitHub Action to delete or archive vectors & JSON older than 90 days to keep the repo slim.

4 ▪ Tools & Integrations

 LangChain / CrewAI (or Google ADK / AutoAgent) – routes messages between agents, manages tool calls, retries, and step ordering.

 OpenAI GPT-4o / Claude 3 – core LLMs for reasoning, code generation, and content drafting.

 GitHub (private repo) – long-term storage for JSON session snapshots, Markdown reflections, prompt files, and the generated vector index; gives built-in version control and audit history.

 GitHub Actions – automates commits, nightly embedding jobs that build vector_index.parquet, and CI/CD pipelines triggered by the DevOps Agent.

 FAISS (or Chroma) local index – loads the vector file from the repo at app start-up and serves fast cosine-similarity searches without an external vector-DB bill.

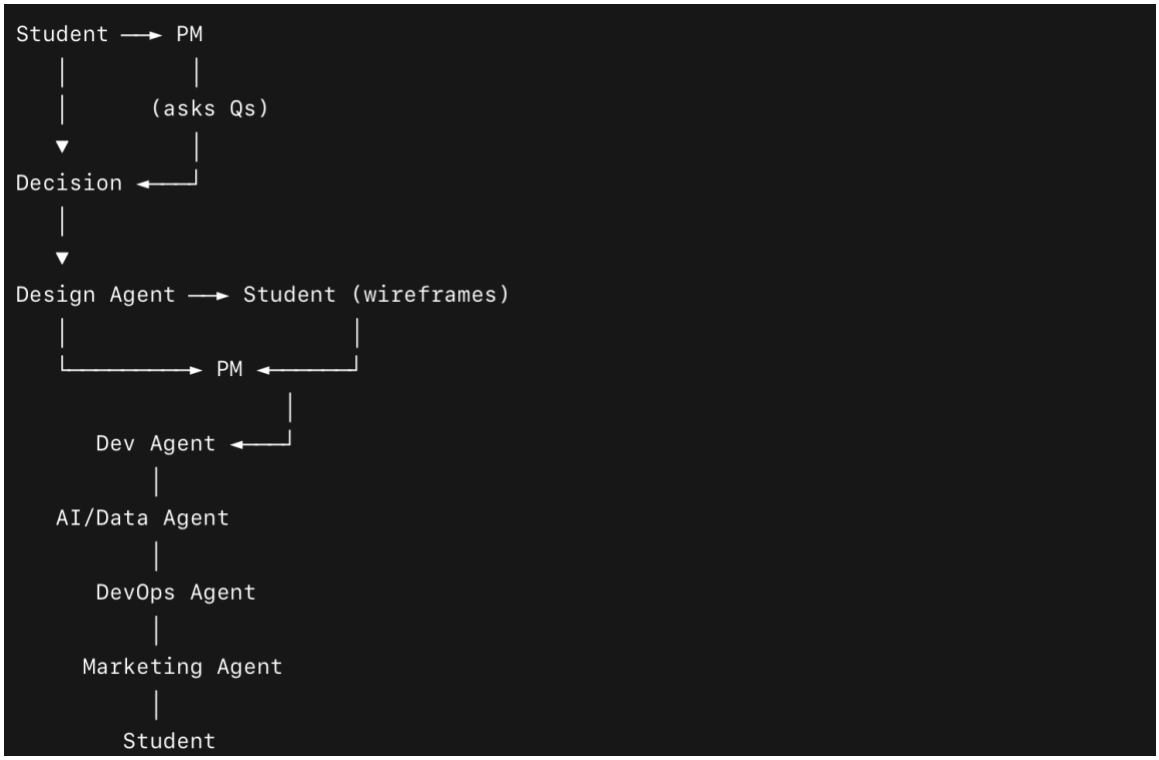
 Next.js + Tailwind – responsive chat UI and canvas pane where students view agent messages, wireframes, and code snippets.

 Figma Plugin API – streams live wireframe thumbnails or links directly into the chat, keeping design iterations visible.

 Fly.io or Render – one-command Docker deploy targets so students can push the whole stack (API + UI) without managing servers.

 Object Storage (optional: Git LFS, GitHub Releases, or S3) – holds large binary assets (e.g., high-res design exports) if they exceed normal repo size limits.

5 ▪ Detailed Interaction Model (state diagram style)



Edges represent message packets with JSON envelope:
{sender, recipient, content, step_id, status}

6 ▪ Example Scenario

Step	Exchange	Key Output
1	Student: “Build campus carpool app.”	Idea registered
2	PM→Student: “Main pain-point?”	Clarified: parking scarcity
3	Design Agent → Student: Wireframe v0.1 (Figma)	Visual validated
4	Dev Agent pushes carpool-app repo	React-Native + Supabase setup
5	AI/Data Agent adds *RideMatchLLM.py*	Uses trip vectors + GPT-4o
6	DevOps Agent deploys to TestFlight	Beta link
7	Marketing Agent posts deck & tagline: “Park less, ride together.”	GTM asset
8	Student toggles MVP_ACCEPTED = true	System locks artefacts

7 ▪ Key Design Tips (expanded)

1. **Single-responsibility prompts** → prevents cross-talk.
2. **Conversation IDs** per feature branch to isolate context.
3. **Automated schema validation** on JSON output from agents.
4. **"Human-in-the-loop" gates** after any destructive operation (DB migrations, prod deploy).
5. **Prompt version control** with semantic diff to track performance drift.
6. **Telemetry hooks**—log token usage, latency, agent success/failure rates for course analytics.

8 ▪ Conclusion

By combining specialised LLM agents, a clear routing protocol, and a persist-and-retrieve memory layer, *AI Expert TeamMate* turns BADM 350 students into CEOs of a guided startup simulation. The framework's modular tools (LangChain, Supabase, GitHub Actions) ensure it can scale from a classroom demo to capstone-level, production-ready projects.