1 Title:

- AI Project Management Agent: From Planning to Execution
- Subtitle:
- Using Preference Fine-Tuning (PFT) + Reinforcement Fine-Tuning (RFT) with Jira, GitHub, Notion, Slack
- Key Message:
- Turn LLMs into senior PM copilots that can plan, track, and communicate across your real tools.

PM work is:

- Fragmented across Jira, GitHub, Notion, Slack
- Full of repetitive status updates and manual planning
- Often inconsistent in quality between teams

LLMs today:

- Can "talk about" projects,
- But aren't reliably grounded in your tickets, repos, docs, and chats.

Opportunity:

An agent that can understand the project, propose realistic plans, and operate tools safely.

3

1. Enterprise-grade planning

- Hierarchical plans, realistic timelines, clear risks
- Tool-native execution
- Create/update Jira issues, GitHub PR summaries, Notion specs, Slack updates
- Reliable & auditable
- Human-in-the-loop approvals
- Policy guardrails and logs
- Composable architecture
- Swap model, tools, or data sources without redoing everything

4 Four main layers:

- Foundation & Alignment Layer
- Base LLM + SFT + PFT (DPO) + safety policies
- Reasoning & Planning Layer
- Project planner, timeline estimator, risk engine, dependency resolver
- Tool & Data Layer
- Jira, GitHub, Notion, Slack connectors
- Vector search over project docs, tickets, specs
- Orchestration & UX Layer
- Agent controller, workflows, approvals
- UI: chat, side-panel in Jira/Notion, Slack app

5

- Start with a strong base LLM (general + coding).

- SFT on:
- PRDs, sprint docs, design docs, runbooks
- PFT / DPO on preference datasets:
- Good vs bad project breakdowns
- Strong vs weak timelines
- Good vs bad risk registers
- High vs low quality stakeholder comms

Outcome:

The model "thinks like a senior PM."

- Offline environment that simulates:

- Jira: create/update tickets, change status, assign
- GitHub: read PR metadata, post summary comments
- Notion: read/write pages
- Slack: draft & send updates (with approval)
- Reward examples:
- +1 if the right Jira field is set
- +1 if the right repo/branch is referenced
- +1 if Slack summary matches actual issue state
- -1 if invalid API calls or inconsistent states

Outcome:

7 - LLM Core (PFT + RFT trained)

- Agent Orchestrator
- Decides when to call tools vs. reason internally
- Memory & Retrieval
- Vector DB over specs, tickets, docs
- Tool Adapters
- JiraClient, GitHubClient, NotionClient, SlackClient
- Policy & Safety Layer
- Access scope, rate limiting, approval policy

1. User asks: "Help me plan Q3 release for Feature X."

- Agent:
- Retrieves relevant docs (Notion/Confluence)
- Fetches Jira epics + open bugs
- Checks GitHub branches/PRs
- Fetches last Slack status updates
- LLM:
- Synthesizes a project plan, timelines, risks
- Proposes Jira changes & Slack message drafts
- Human:
- Reviews & approves changes / messages

9 Input:

"Create a 3-sprint plan to ship Feature X."

Agent:

- Gets existing Jira epics/issues
- Groups into sprints by priority, dependencies, team capacity
- Proposes new tickets for gaps
- Writes a Slack summary for the team

Human approves:

- Agent creates/updates Jira issues
- Posts Slack summary draft to a channel

Input:

- **"List top 10 risks for Feature X and mitigation plans."**
- **Agent:**
- Reads spec, Jira issues, known blockers
- Uses risk-register patterns to propose:
- Risk name, impact, probability, owner, mitigation
- Drafts a Notion "Risk Register" section
- **Human approves and edits as needed.**

- Planning quality

- Human rating of plans (clarity, realism, coverage)
- Tool correctness
- % of successful Jira/GitHub/Notion actions
- Latency & UX
- Time from request to draft plan
- Adoption & satisfaction
- PM/EM satisfaction scores
- Reduction in manual PM work

12 - Scoped permissions per tool

- All actions logged with:
- Who requested
- What the agent proposed
- What was executed
- Sensitive operations require:
- Approval
- Or "draft-only" mode

- Phase 1: Read-only advisor (no writes, only suggestions)

- Phase 2: Draft creator (Jira/Slack drafts, Notion pages)
- Phase 3: Limited auto-execution with guardrails
- Phase 4: Org-wide deployment + custom policies & KPIs

End of Slide Deck