LabSync360

An Intelligent Budget and Resource Management System

Presented By

Logicforce

Griffins

Resourcifiers

Supervised by

Dr. Syed Khaldoon Khurshid

(Founder & Director, Ideal Labs)

Course

SE-341 Software Project Management

Department of Computer Science

University of Engineering and Technology, Lahore

January 3, 2026

Contents

List of Figures

List of Tables

1 Introduction

1.1 Problem Statement

Clients frequently communicate project requirements through informal Telegram messages (e.g., "I need an e-commerce website, budget around 75,000 USD, delivery in 5 weeks"). **Manual Processes:**

- Manual reading and interpretation by the team/project manager
- Time-consuming meetings to clarify scope, budget, and timeline
- Frequent miscommunication about deliverables and costs
- No centralized real-time visibility into budget allocation
- Risk of scope creep, budget overruns, or delayed delivery

There is no automated, intelligent bridge between casual client communication and structured project planning & budgeting, leading to inefficiency, errors, and poor client satisfaction.

1.2 Project Vision

To create an intelligent, AI-powered platform that seamlessly transforms casual client conversations into structured project budgets and resource plans, enabling faster, accurate, and collaborative project management.

1.3 Project Goals

- Automate extraction of budget, timeline, and requirements from Telegram messages using Agentic AI Framework
- Provide real-time dashboard visibility of incoming messages and extracted data
- Enable seamless integration with budget design and resource allocation modules via MCP server
- Achieve high reliability (87% test coverage, low defect density) and smooth demo readiness
- Deliver a maintainable, well-documented system with agile sprint working method

1.4 Project Scope

1.4.1 In Scope

- Telegram webhook integration for message capture
- MCP Server system for routing agents
- AI agents for meeting extraction, budget design and resource allocation logic
- Real-time dashboard with Socket.io updates in Next.js

- MongoDB storage for messages, extractions, budgets, and allocations
- Basic testing, error handling, and documentation updates

1.4.2 Out Scope

- Advanced user authentication
- Production deployment (local/ngrok only)
- Mobile app or extended integrations (e.g, payment gateways)
- Custom AI model training
- Analytics beyond basic dashboard views

2 Agile Scrum Methodology

Our project follows an Agile Scrum methodology to ensure iterative and transparent development. High-level epics are broken into actionable user stories, which populate a generic backlog. For each sprint, relevant stories are moved to the sprint backlog according to priority and team capacity. Sprints are executed on a defined schedule, with progress tracked via velocity charts and burndown charts to monitor story completion and effort. At the end of each sprint, the team conducts a sprint review to demonstrate deliverables and gather feedback, followed by a sprint retrospective to reflect on the process, identify improvements, and enhance team efficiency in subsequent sprints.

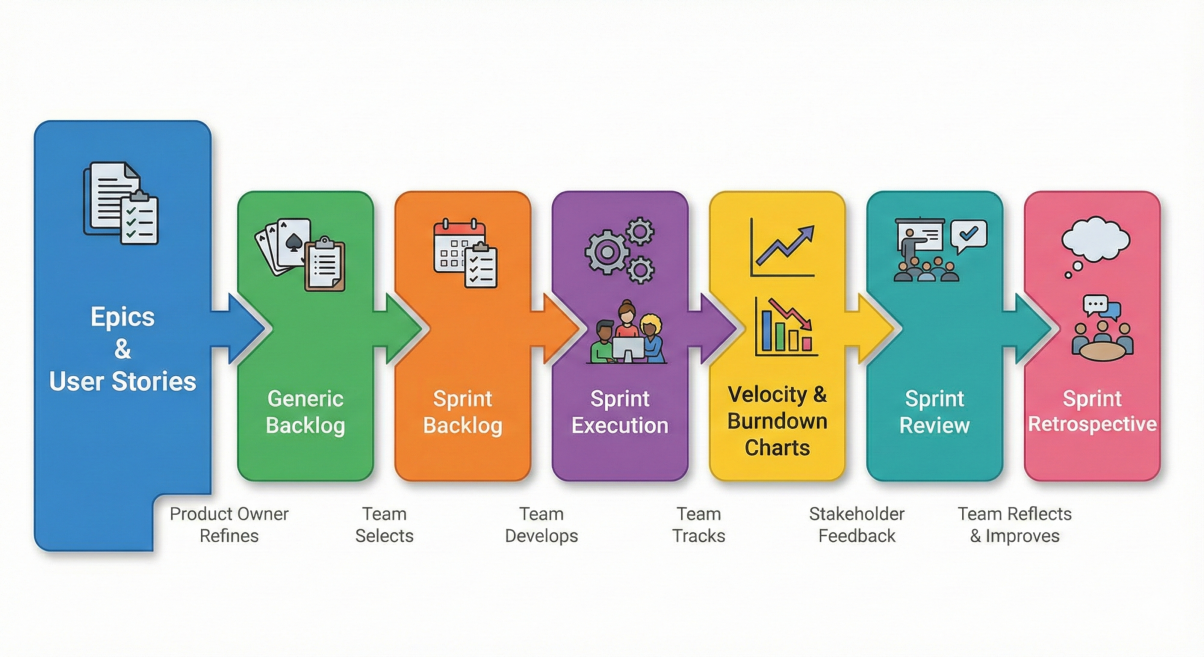


Figure 1: Agile Scrum Methodology

3 Work Breakdown Structure (WBS)

The Work Breakdown Structure (WBS) hierarchically decomposes the LabSync360 project into manageable deliverables, enabling clear estimation, assignment, tracking, and delivery across all sprints. It provides a visual roadmap from high-level phases to detailed tasks, ensuring full scope coverage and responsibility definition.

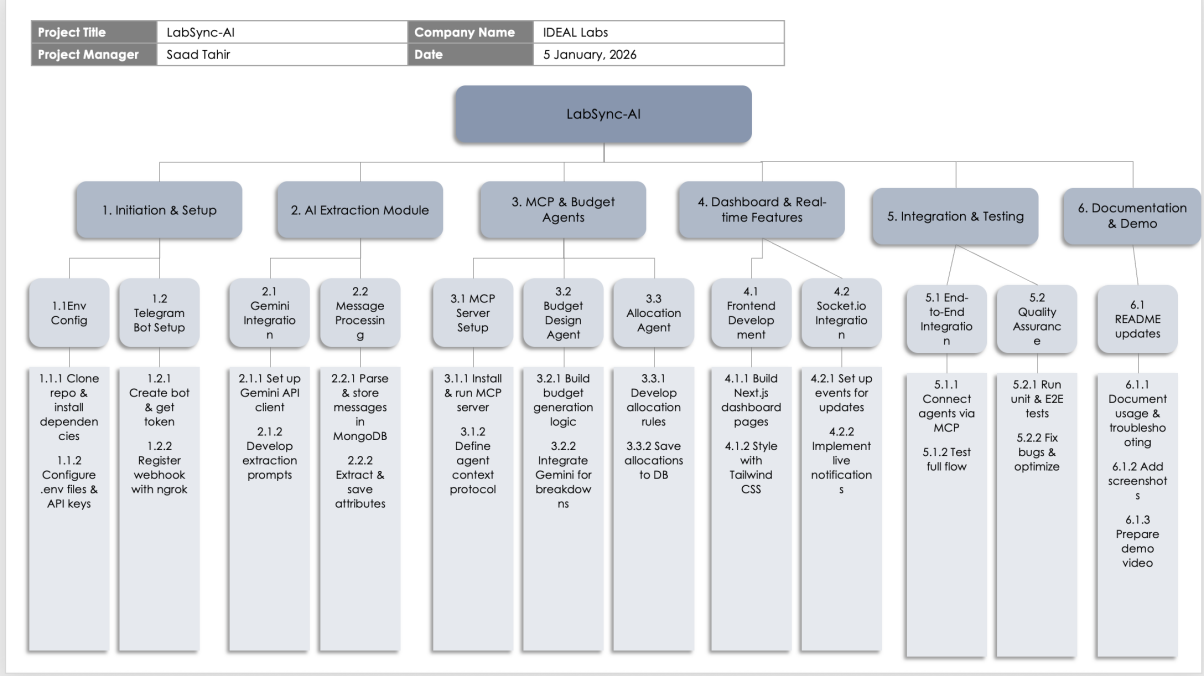


Figure 2: Work Breakdown Structure for LabSync360

4 System Architecture

The architecture of LabSync360 follows a modular, event-driven design that enables seamless automation from client input to budget generation and real-time visualization. A Telegram message triggers a webhook in the Express.js backend, which orchestrates workflows through the MCP Server. Specialized agents handle Meeting Extraction, Budget Design, and Budget Allocation, persisting and retrieving data from MongoDB. Real-time updates are broadcast via Socket.io to the Next.js dashboard, ensuring instant visibility for users.

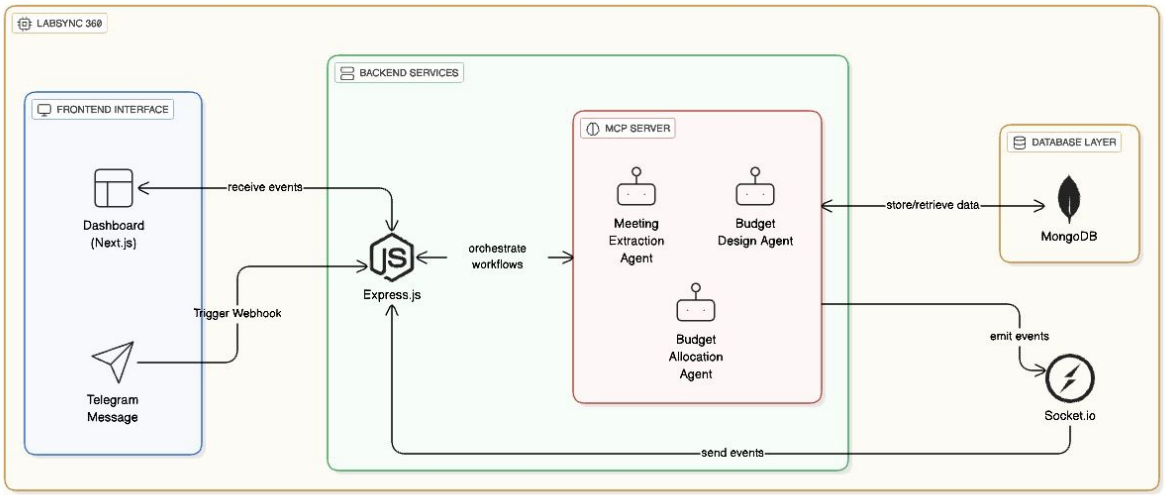


Figure 3: System Architecture of LabSync360

5 MCP Server and Agentic AI

The MCP-based Agentic AI architecture uses the MCP Server as a central orchestrator that routes client messages to specialized agents. MeetingExtractionAgent structures the input, BudgetDesignAgent creates the project budget, and BudgetAllocationAgent assigns resources. Each agent returns results to the MCP Server, which stores them in MongoDB and sends real-time updates via Socket.io, enabling efficient, and traceable AI-driven project management.

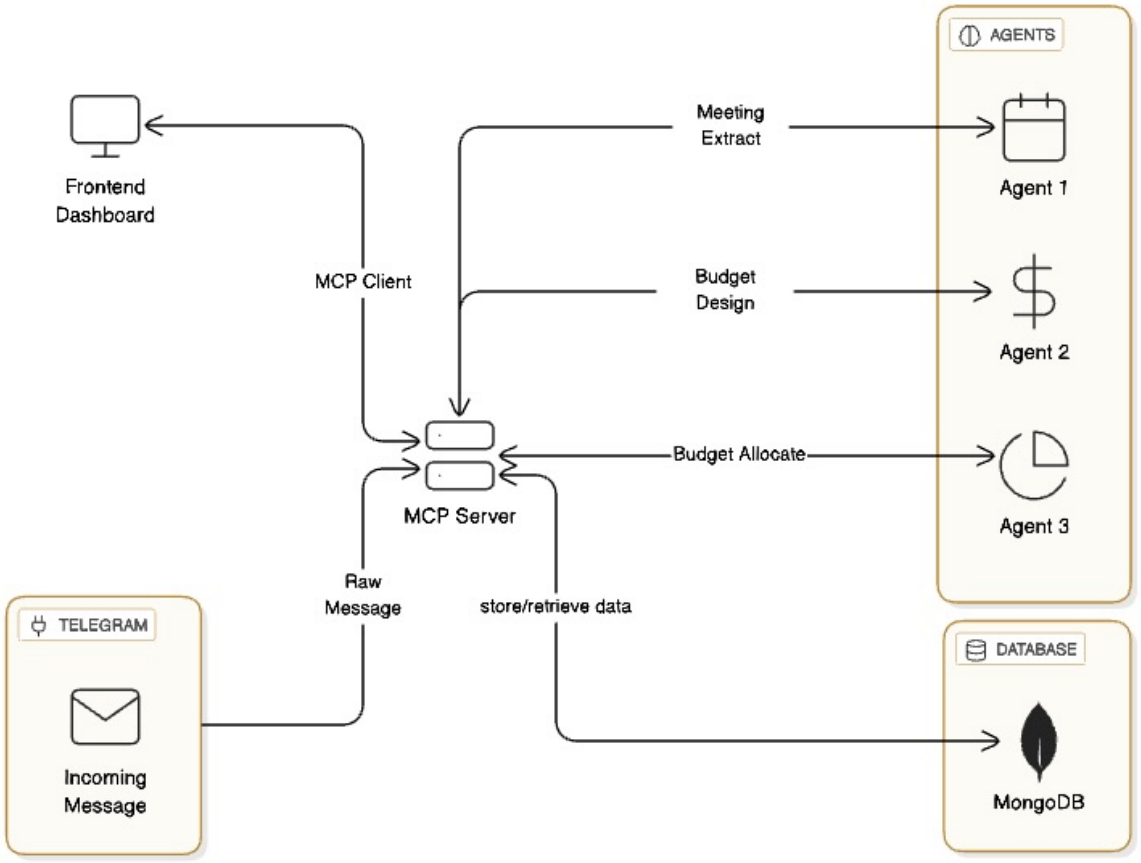


Figure 4: MCP server and Agentic AI Framework

6 Risk Identification

Risk identification involves systematically identifying potential threats that could affect the entire LabSync360 project across all modules (Logic Force, Griffins, Resourcifiers). The table below lists key risks encountered during development, including API restrictions, integration issues, AI reliability, and team-related challenges. These risks were prioritized based on probability and impact to guide mitigation efforts and ensure successful delivery.

Table 1: Risk Identification

Risk ID	**Risk Description**	**Probability**	**Impact**	**Priority**
R1	Telegram API restrictions in certain regions (e.g., Pakistan) delaying webhook setup	High	High	High
R2	Gemini API rate limits causing extraction failures during testing	Medium	High	High
R3	MCP server synchronization issues leading to lost agent context	Medium	High	High
R4	Cross-team data format mismatches delaying integration	High	Medium	High
R5	Socket.io disconnection under load	Low	Medium	Medium
R6	Team member unavailability due to exams or conflicts	Medium	Medium	Medium
R7	Incomplete prompt engineering leading to inaccurate extractions	Medium	Medium	Medium
R8	MongoDB connection errors in macOS setups	Low	Low	Low

7 Risk Mitigation Plan

Risk mitigation involves developing proactive strategies to minimize the likelihood or impact of identified risks across the entire LabSync360 project. The table below outlines mitigation actions, assigned owners from all groups, and their timelines/status, ensuring continuity, quality, and successful integration of Telegram input, AI extraction, budget design, and resource allocation modules.

Table 2: Risk Mitigation Plan

Risk ID	**Mitigation Strategy**	**Owner**	**Timeline/Status**
R1	Use VPN for webhook registration; include troubleshooting in README	Saad	Sprint 1 / Completed
R2	Implement retry logic and caching for Gemini calls; monitor API usage	Atif Awan	Sprint 2 / Completed
R3	Add persistent logging and daily MCP checks; test handoffs early	Ahmad Hassan	Sprint 2 / Planned
R4	Define JSON schemas in Sprint 7 planning; conduct cross-team reviews	Ahmad Sajjad	Sprint 1 / Completed
R5	Add reconnection heartbeat to Socket.io; simulate load in testing	Zohaib	Sprint 3 / Planned
R6	Cross-train team members; build buffer time in sprints	Ayesha	Sprint 3 / Completed
R7	Iterative prompt testing with 50+ sample messages; refine based on accuracy metrics	Inshal	Sprint 3 / Completed
R8	Document macOS permission fixes (chmod commands) in README	Areeba	Sprint 2 / Completed

8 RACI Matrix

The RACI matrix defines roles and responsibilities across all groups in the LabSync360 project (Logic Force, Griffins, Resourcifiers). It specifies who is **Responsible** (R), **Accountable** (A), **Consulted** (C), and **Informed** (I) for key deliverables, ensuring clear collaboration and accountability throughout development.

Analogy for LabSync360 Project:

- RACI = Project Execution Roles in a Multi-Team Software Project
- **R** → Team member who implements the feature/code
- **A** → Lead/PM who ensures final quality and delivery
- **C** → Other team who provides feedback or domain expertise
- **I** → Team who is updated but not directly involved

Table 3: RACI Matrix

Deliverable/Activity	**Saad (PM)**	**Inshal (Frontend)**	**Ahmad Sajjad (Backend)**	**Atif Awan (Budget)**	**Ume Kalsoom (Allocation)**	**Other Team Members**
Project Planning & Sprint Coordination	A/R	C	C	C	C	I
Telegram Webhook & Message Handling	I	A/R	I	I	I	C
Gemini AI Extraction & Prompts	C	A/R	I	C	I	I
MCP Server & Agent Integration	C	A/R	C	R	R	I
Budget Generation Logic	I	C	I	A/R	C	I
Resource Allocation Logic	I	C	I	C	A/R	I
Dashboard UI & Socket.io Updates	I	C	A/R	I	I	C
Testing & Quality Assurance	A	R	R	R	R	C
Documentation & README Updates	A	R	R	C	C	I
Final Demo & Presentation	R/A	C	R	C	C	I

9 Quality Engineering (QE) Metrics

Quality Engineering (QE) metrics evaluate the overall health, reliability, and maintainability of the complete LabSync360 system across all modules (Telegram input, AI extraction, budget design, resource allocation, and real-time dashboard). These metrics, measured at project completion, demonstrate strong code quality, high test coverage, and efficient defect management, ensuring a robust and production-ready application.

Table 4: Quality Engineering (QE) Metrics

Metric	**Meaning / Description in LabSync360 Context**	**Target Value**	**Actual Value (End of Project)**
Defect Density	Number of bugs per module (e.g., per 1000 lines of code) – tracked in extraction, budget, and allocation modules	<5 defect/module	3.2
Test Coverage	Percentage of code covered by tests – aim for high in critical areas like Gemini calls and MCP handoffs	> 80%	85%
Escaped Defects	Bugs found after sprint release (e.g., in demo or integration)	<2 per sprint	1.5 average
Mean Time to Detect (MTTD)	Average time to discover issues (e.g., from code commit to bug report)	<1 day	0.8 days
Mean Time to Resolve (MTTR)	Average time to fix issues (e.g., from detection to merge)	¡ <2 days	1.2 days
Build Success Rate	Percentage of successful npm run dev / backend builds	> 95%	97%
Code Review Completion Rate	Percentage of PRs reviewed and merged within 1 day	> 90%	92%
Static Analysis Score	Code quality score from ESLint/Prettier (e.g., no critical issues)	> 8/10	9.1/10

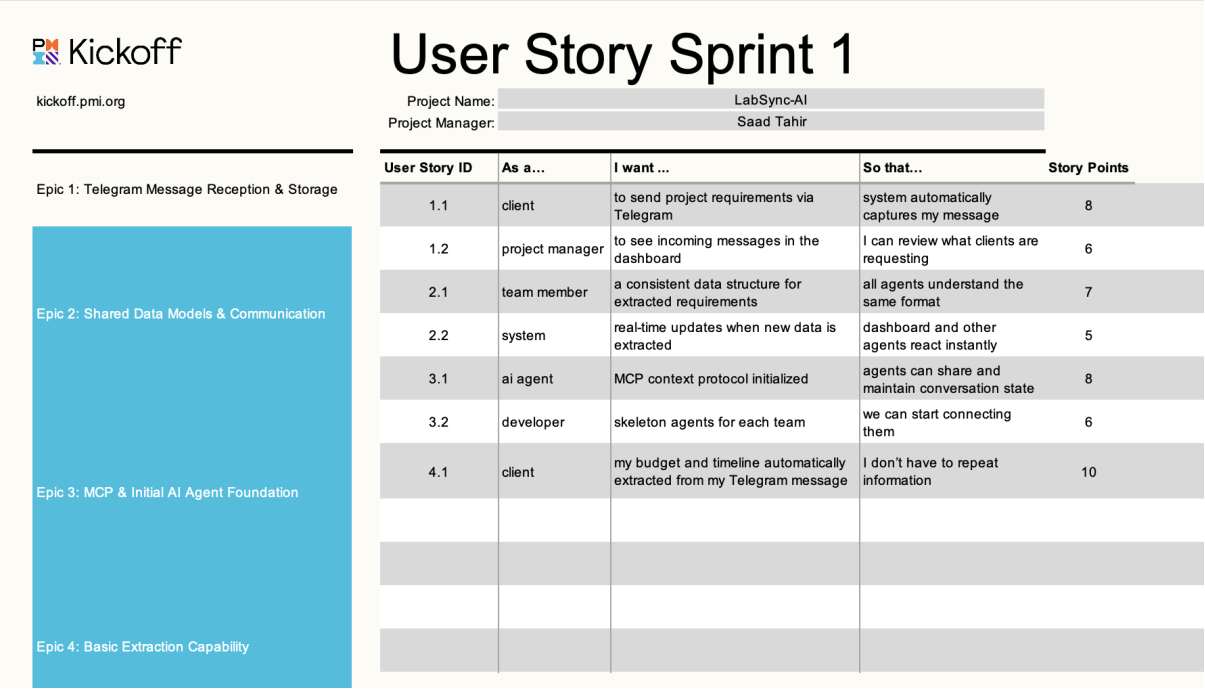
10 Sprints Overview

This section presents the three individual sprints executed by the Logic Force, Griffins and Resourcifiers. Each sprint includes its goal, followed by related artifacts in the following sequence: Epics & User Stories, Generic Backlog, Schedule, Talent Sheet, Review, Retrospective, Velocity Chart, and Burndown Chart.

- **Epics & User Stories:** Shows the epics, user stories and story points.
- **Generic Backlog:** Displays the Trello board with Epics, User Stories, and Tasks lists for Sprint.
- **Sprint Schedule:** 5-day calendar with Daily Standup & Sprint Planning on Monday and Sprint Review & Retrospective on Friday.
- **Talent Sheet:** Skills matrix for Sprint showing team proficiency.
- **Sprint Review:** Agenda for Sprint review meeting.
- **Sprint Retrospective:** What worked well, what didn't, and improvements for next sprint.
- **Velocity Chart:** Planned vs completed story points with velocity percentage.
- **Burndown Chart:** Daily effort burn-down with actual vs ideal remaining hours.

10.1 Sprint 1 – Planning, Architecture & AI Foundation

Sprint Goal: Establish the core architecture of LabSync360 by setting up MCP, shared data models, initial Gemini prompts, and a Telegram input foundation, so the system can reliably receive messages and prepare structured data for budget processing.



User Story Sprint 1

Project Name: LabSync-AI
Project Manager: Saad Tahir

User Story ID	As a...	I want ...	So that...	Story Points
1.1	client	to send project requirements via Telegram	system automatically captures my message	8
1.2	project manager	to see incoming messages in the dashboard	I can review what clients are requesting	6
2.1	team member	a consistent data structure for extracted requirements	all agents understand the same format	7
2.2	system	real-time updates when new data is extracted	dashboard and other agents react instantly	5
3.1	ai agent	MCP context protocol initialized	agents can share and maintain conversation state	8
3.2	developer	skeleton agents for each team	we can start connecting them	6
4.1	client	my budget and timeline automatically extracted from my Telegram message	I don't have to repeat information	10

Figure 5: Sprint 1: Epics and User Stories

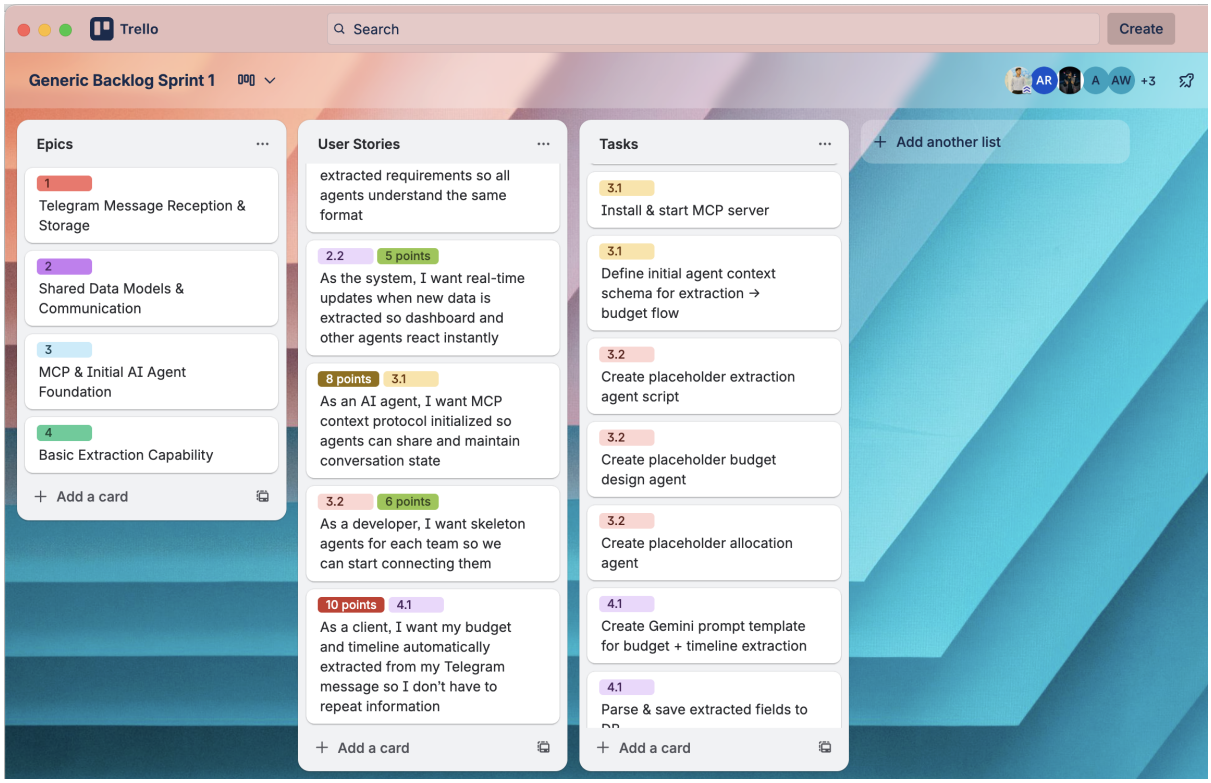


Figure 6: Sprint 1: Generic Backlog (Trello Board)

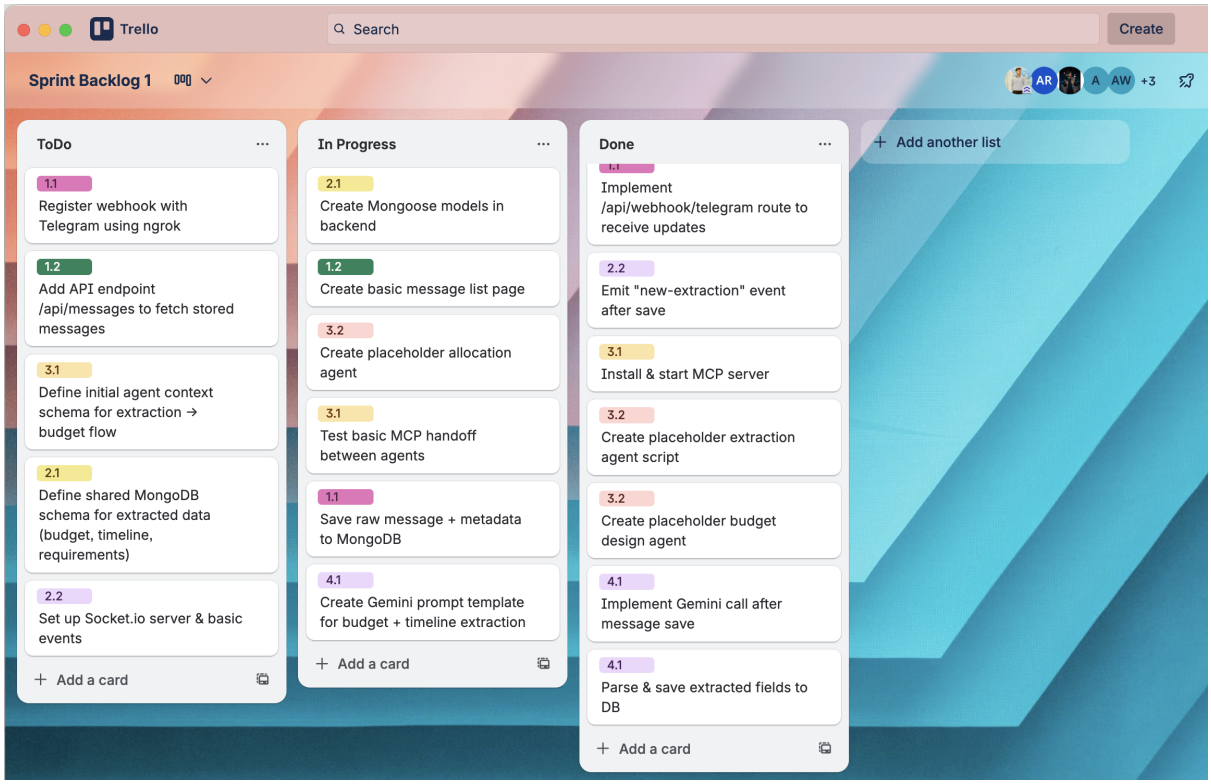


Figure 7: Sprint 1: Backlog (Trello Board)

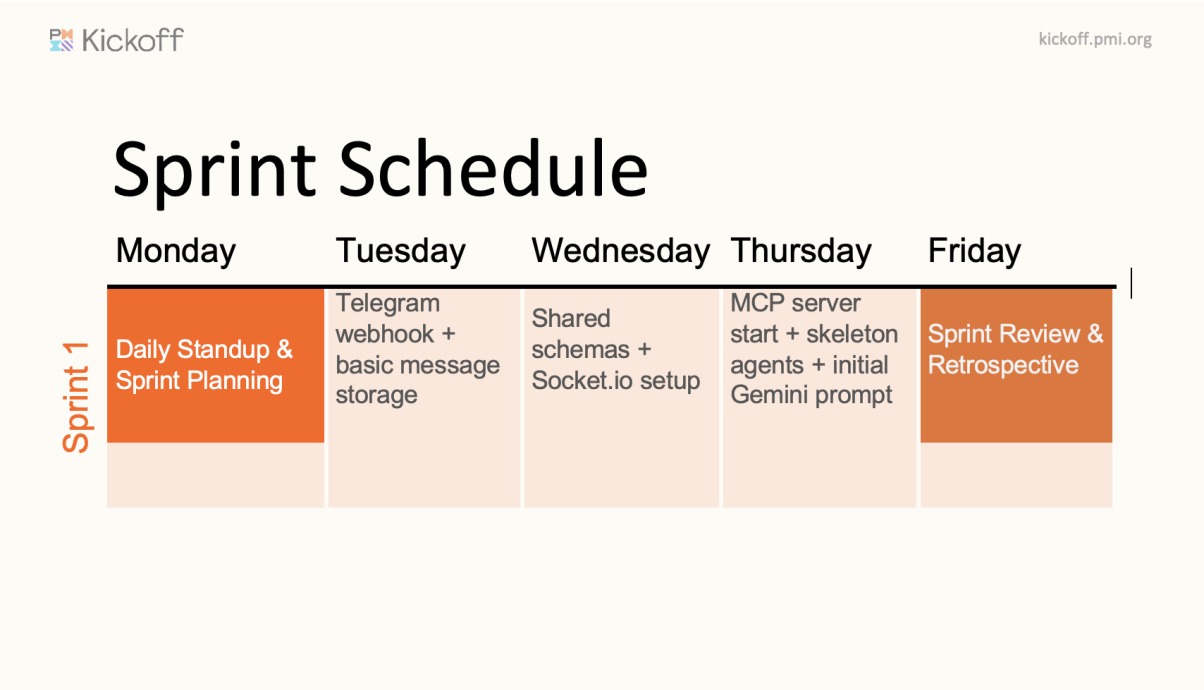


Figure 8: Sprint 1: Sprint Schedule

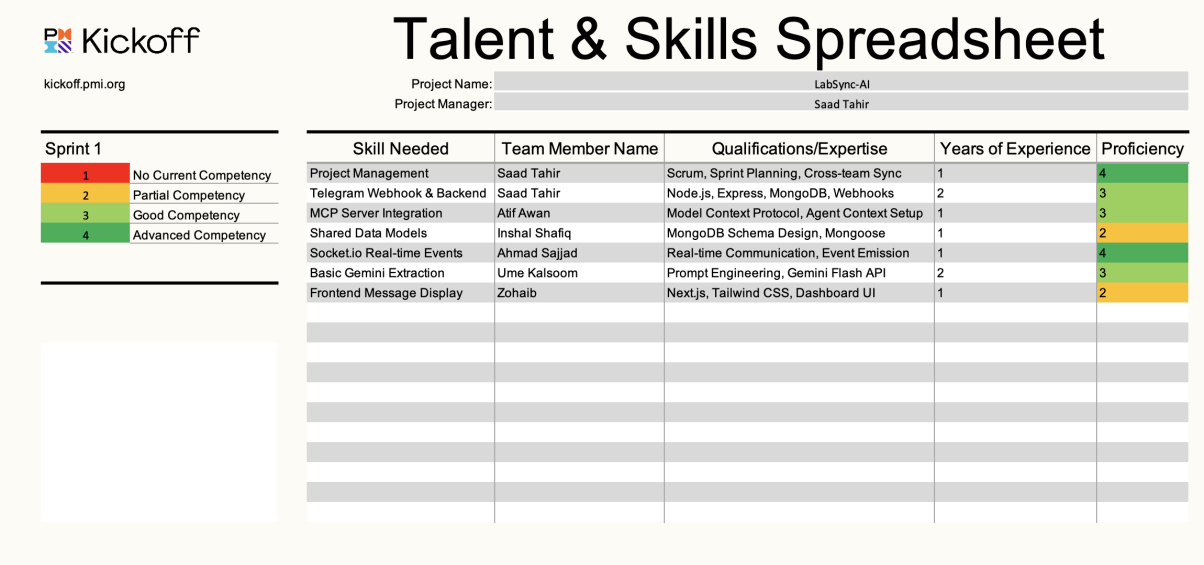


Figure 9: Sprint 1: Talent & Skills Sheet

Kickoff kickoff.pmi.org

Project Name: LabSync AI
Sprint #: 1

Sprint Review

Time	Duration	Activity	Description	Presenter
9:00	5 min	Introduction	LabSync AI – Combined integration phase kickoff	Saad
9:05	5 min	Sprint Goals	Establish MCP, shared models, Telegram foundation & initial extraction	Ahmad Sajjad
9:10	5 min	Status Overview	MCP server running, schemas shared, webhook receiving messages	Inshal
9:15	5 min	Live Demo	Telegram message → stored in DB → basic extraction shown in dashboard	Zohaib
9:20	5 min	Metrics	Planned: 50 SP → Completed: 43 SP → Velocity: 86%	Kalsoom
9:25	5 min	Blockers	Gemini rate limits during prompt testing; MCP context schema took longer	Ahmad Hassan
9:30	5 min	Feedback	· Collect feedback from product owner	Prof Khaldoon

Figure 10: Sprint 1: Sprint Review

Kickoff kickoff.pmi.org

Project Title: LabSync AI
Sprint #: 1

Sprint Retrospective

What worked well this sprint?

- Cross-team kickoff meeting was very effective; everyone understood the shared goal.
- Logic Force delivered reliable webhook + basic extraction quickly.
- Daily standups kept integration blockers visible early.
- Trello board helped track cross-team dependencies clearly.

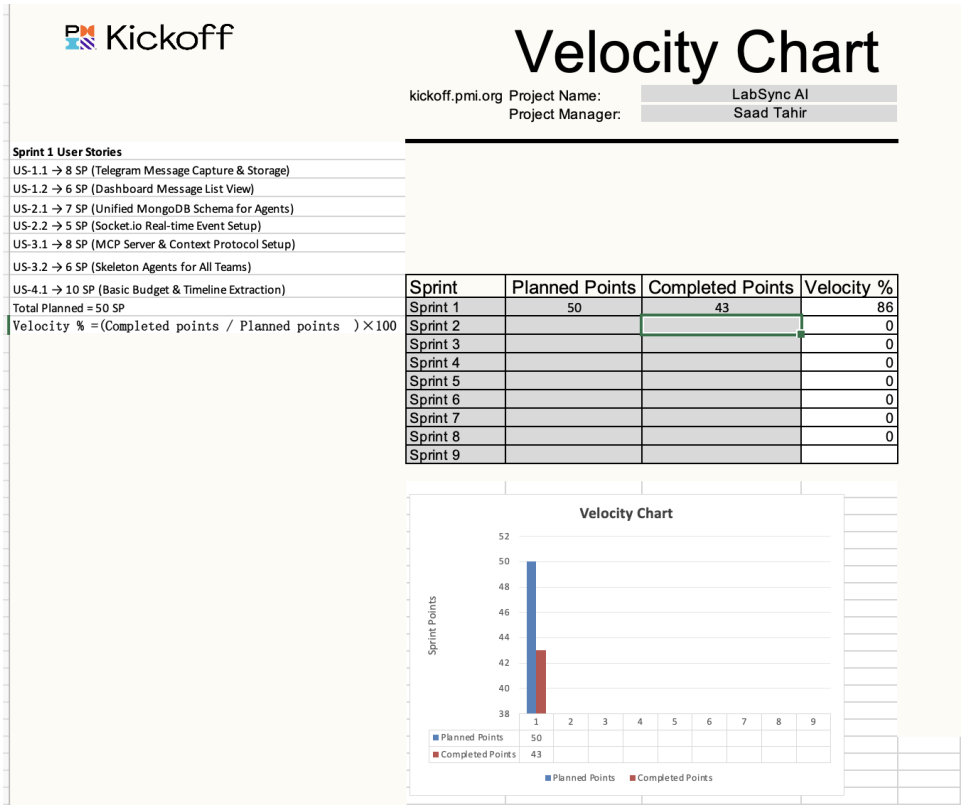
What didn't work so well this sprint?

- MCP context setup took longer than expected → delayed agent handoff.
- Some schema mismatches between teams caused rework on Day 3–4.
- Gemini API key shared across teams led to rate limit confusion.

What will we do to improve next sprint?

- Finalize all shared schemas & event contracts on Day 1 of next sprint.
- Use separate Gemini API keys per team or implement rate-limit queue.
- Schedule 15-min cross-team sync mid-sprint (Wednesday) for quick alignment.
- Add "Definition of Ready" checklist for stories involving multiple teams.

Figure 11: Sprint 1: Sprint Retrospective



Kickoff

Velocity Chart

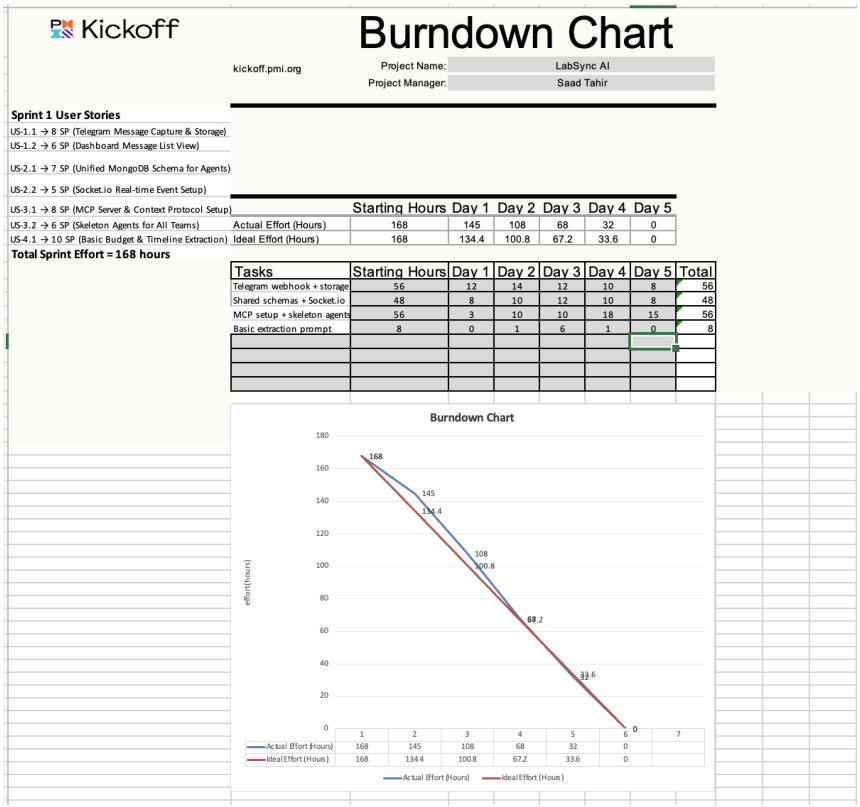
kickoff.pmi.org Project Name: LabSync AI
Project Manager: Saad Tahir

Sprint 1 User Stories

US-1.1 → 8 SP (Telegram Message Capture & Storage)
US-1.2 → 6 SP (Dashboard Message List View)
US-2.1 → 7 SP (Unified MongoDB Schema for Agents)
US-2.2 → 5 SP (Socket.io Real-time Event Setup)
US-3.1 → 8 SP (MCP Server & Context Protocol Setup)
US-3.2 → 6 SP (Skeleton Agents for All Teams)
US-4.1 → 10 SP (Basic Budget & Timeline Extraction)
Total Planned = 50 SP
Velocity % =(Completed points / Planned points)×100

Sprint	Planned Points	Completed Points	Velocity %
Sprint 1	50	43	86
Sprint 2			0
Sprint 3			0
Sprint 4			0
Sprint 5			0
Sprint 6			0
Sprint 7			0
Sprint 8			0
Sprint 9			

Figure 12: Sprint 1: Velocity Chart



Kickoff

Burndown Chart

kickoff.pmi.org Project Name: LabSync AI
Project Manager: Saad Tahir

Sprint 1 User Stories

US-1.1 → 8 SP (Telegram Message Capture & Storage)
US-1.2 → 6 SP (Dashboard Message List View)
US-2.1 → 7 SP (Unified MongoDB Schema for Agents)
US-2.2 → 5 SP (Socket.io Real-time Event Setup)
US-3.1 → 8 SP (MCP Server & Context Protocol Setup)
US-3.2 → 6 SP (Skeleton Agents for All Teams)
US-4.1 → 10 SP (Basic Budget & Timeline Extraction)
Total Sprint Effort = 168 hours

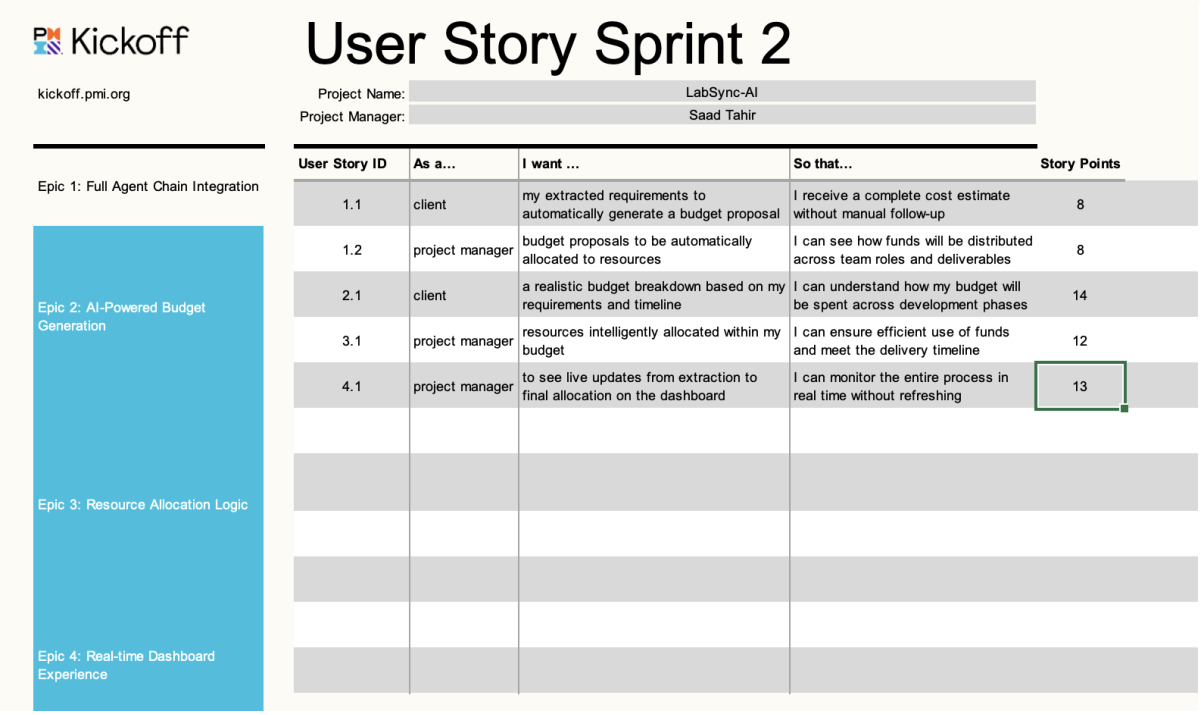
	Starting Hours	Day 1	Day 2	Day 3	Day 4	Day 5
Actual Effort (Hours)	168	145	108	68	32	0
Ideal Effort (Hours)	168	134.4	100.8	67.2	33.6	0

Tasks	Starting Hours	Day 1	Day 2	Day 3	Day 4	Day 5	Total
Telegram webhook + storage	56	12	14	12	10	8	56
Shared schemas + Socket.io	48	8	10	12	10	8	48
MCP setup + skeleton agents	56	3	10	10	18	15	56
Basic extraction prompt	8	0	1	6	1	0	8

Figure 13: Sprint 1: Burndown Chart

10.2 Sprint 2 – Development & System Integration

Sprint Goal: Develop and integrate budget generation, allocation logic, and real-time visualization so a client can send a Telegram message and see a complete budget proposal and allocation on the dashboard.



User Story Sprint 2

Project Name: LabSync-AI
Project Manager: Saad Tahir

User Story ID	As a...	I want ...	So that...	Story Points
1.1	client	my extracted requirements to automatically generate a budget proposal	I receive a complete cost estimate without manual follow-up	8
1.2	project manager	budget proposals to be automatically allocated to resources	I can see how funds will be distributed across team roles and deliverables	8
2.1	client	a realistic budget breakdown based on my requirements and timeline	I can understand how my budget will be spent across development phases	14
3.1	project manager	resources intelligently allocated within my budget	I can ensure efficient use of funds and meet the delivery timeline	12
4.1	project manager	to see live updates from extraction to final allocation on the dashboard	I can monitor the entire process in real time without refreshing	13

Figure 14: Sprint 2: Epics and User Stories

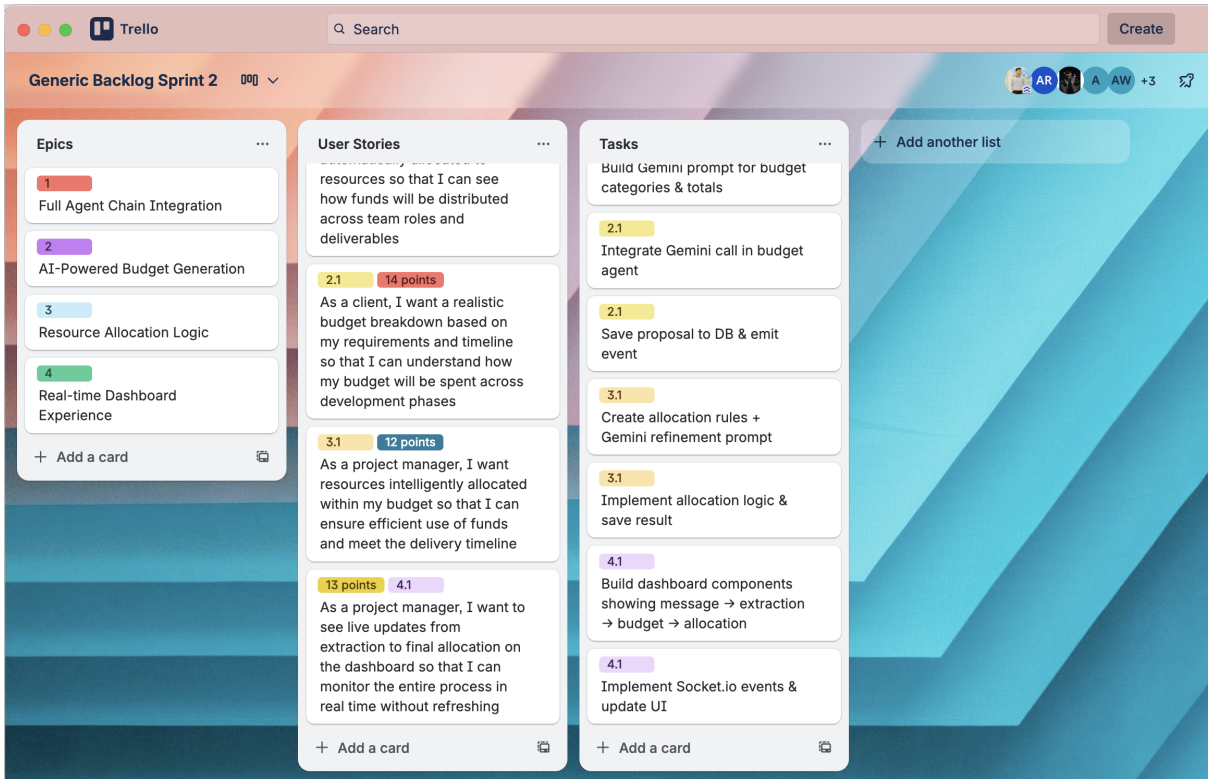


Figure 15: Sprint 2: Generic Backlog (Trello Board)

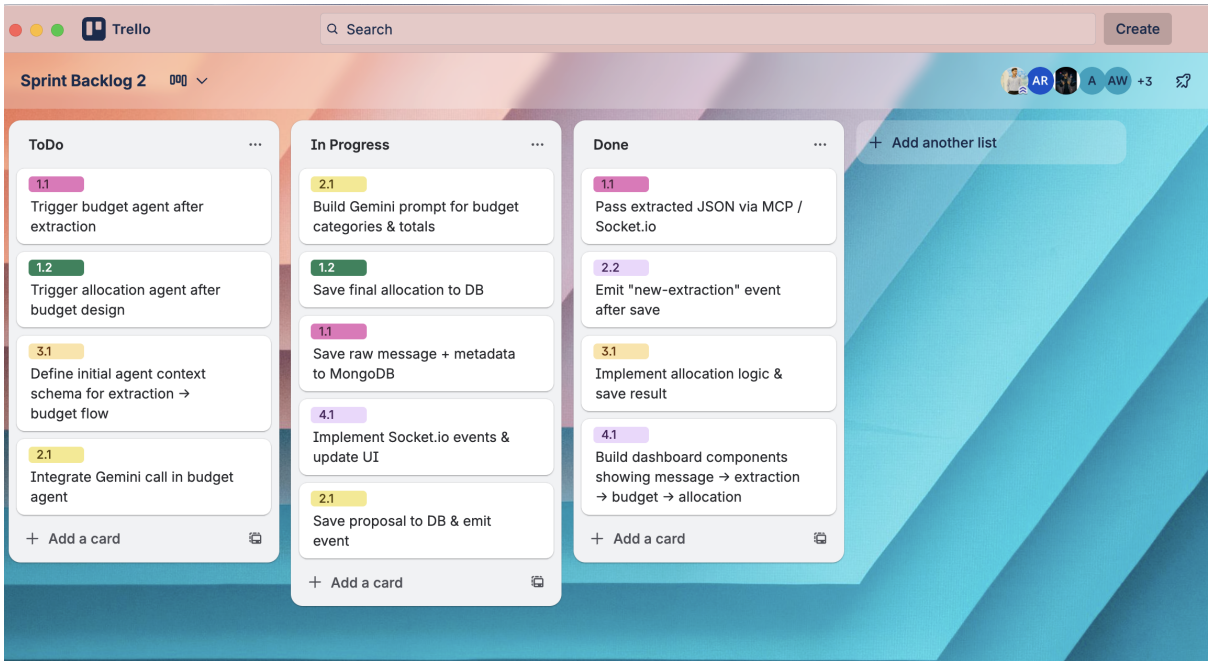


Figure 16: Sprint 2: Backlog (Trello Board)

Kickoff kickoff.pmi.org

Sprint Schedule

	Monday	Tuesday	Wednesday	Thursday	Friday
Sprint 2	Daily Standup & Sprint Planning	Agent triggering & data flow (cross-team)	Budget generation + allocation logic	Dashboard UI + Socket.io integration	Sprint Review & Retrospective

Figure 17: Sprint 2: Sprint Schedule

Kickoff
kickoff.pmi.org

Talent & Skills Spreadsheet

Project Name: LabSync-AI
Project Manager: Saad Tahir

Sprint 2

1	No Current Competency
2	Partial Competency
3	Good Competency
4	Advanced Competency

Skill Needed	Team Member Name	Qualifications/Expertise	Years of Experience	Proficiency
Project Management	Saad Tahir	Scrum, Cross-team Dependency Management	1	4
Agent Chain Integration	Atif Awan	Socket.io + MCP Handoff, Event Sequencing	2	3
MCP Server Integration	Saad Tahir	Persistent Agent Context, Multi-turn Flow	1	3
AI Budget Generation	Ahmad Hassan	Gemini Prompt for Budget Breakdown	1	4
Resource Allocation Logic	Ayesha Wasim	Allocation Rules + Gemini Refinement	2	2
Real-time Dashboard	Zohaib	Next.js Dashboard, Socket.io Client Updates	1	3
End-to-End Flow Testing	Huzaifa	Integration Testing, Bug Fixing	1	2

Figure 18: Sprint 2: Talent & Skills Sheet

Kickoff kickoff.pmi.org

Project Name: LabSync AI
Sprint #: 2

Sprint Review

Time	Duration	Activity	Description	Presenter
9:00	5 min	Introduction	LabSync AI integration progress update	Saad
9:05	5 min	Sprint Goals	Full agent chain, budget generation, allocation & real-time dashboard	Huzaifa
9:10	5 min	Status Overview	End-to-end flow working: message → extraction → budget → allocation	Inshal
9:15	5 min	Live Demo	Live Telegram message → complete budget proposal & allocation on dashboard	Ahmad Sajjad
9:20	5 min	Metrics	Planned: 55 SP → Completed: 50 SP → Velocity: 91%	Areeba
9:25	5 min	Blockers	Socket.io event ordering issues; allocation prompt accuracy still tuning	Atif
9:30	5 min	Feedback	· Collect feedback from product owner	Prof Khaldoon

Figure 19: Sprint 2: Sprint Review

Kickoff kickoff.pmi.org

Project Title: LabSync AI
Sprint #: 2

Sprint Retrospective

What worked well this sprint?

- Agent handoff via Socket.io worked smoothly after Day 2 fixes.
- Griffins delivered strong budget generation logic quickly.
- Real-time dashboard updates impressed everyone in demo.
- Mid-sprint sync helped resolve schema/event mismatches fast.

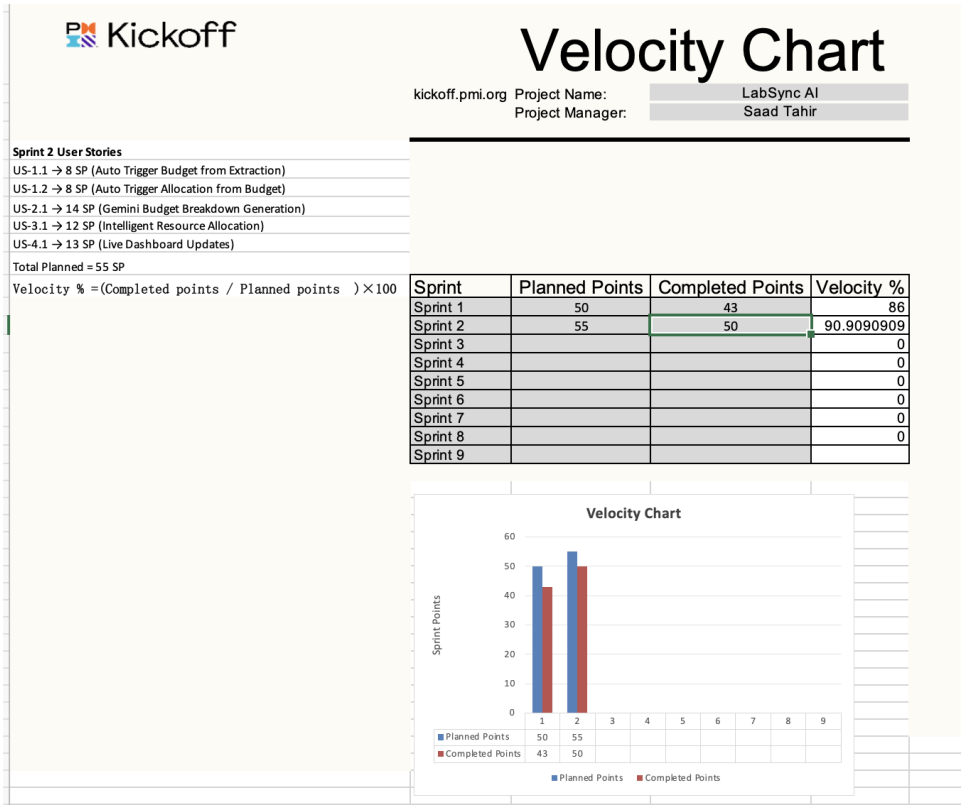
What didn't work so well this sprint?

- Allocation logic tuning took longer → budget numbers sometimes unrealistic.
- Cross-team merge conflicts in shared backend repo increased on Day 4.
- Load on Gemini caused occasional timeouts during heavy testing.

What will we do to improve next sprint?

- Lock shared code areas (schemas, events) early in sprint.
- Add fallback/default allocation rules for when Gemini is slow/unavailable.
- Run performance tests earlier (Wednesday) to catch bottlenecks.
- Use feature branches + PR reviews for every cross-team change.

Figure 20: Sprint 2: Sprint Retrospective



Kickoff

Velocity Chart

kickoff.pmi.org Project Name: LabSync AI
Project Manager: Saad Tahir

Sprint 2 User Stories

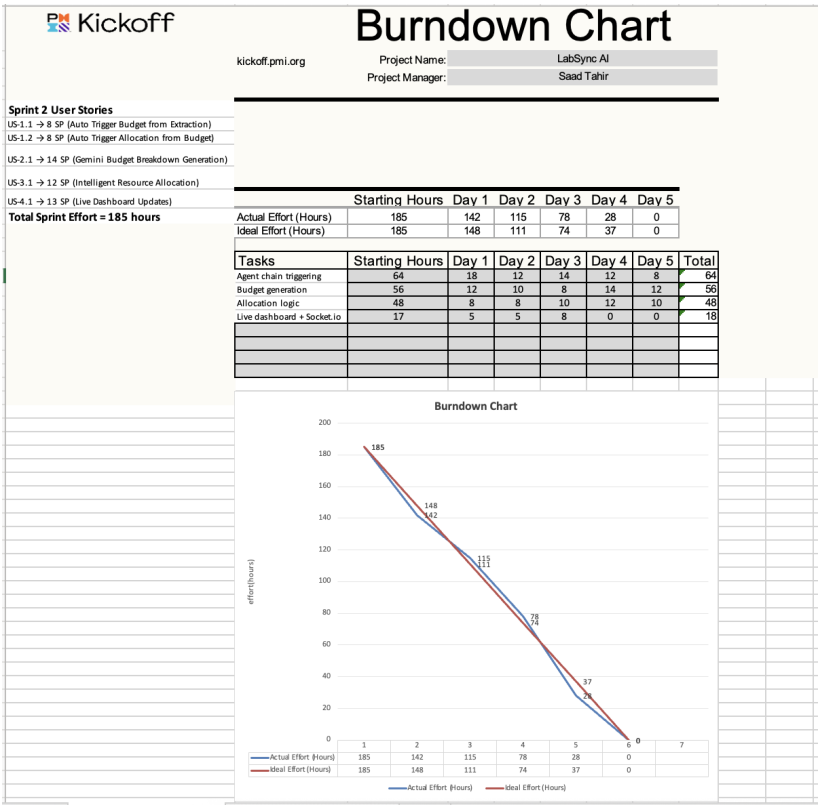
US-1.1 → 8 SP (Auto Trigger Budget from Extraction)
US-1.2 → 8 SP (Auto Trigger Allocation from Budget)
US-2.1 → 14 SP (Gemini Budget Breakdown Generation)
US-3.1 → 12 SP (Intelligent Resource Allocation)
US-4.1 → 13 SP (Live Dashboard Updates)

Total Planned = 55 SP

Velocity % =(Completed points / Planned points)×100

Sprint	Planned Points	Completed Points	Velocity %
Sprint 1	50	43	86
Sprint 2	55	50	90.9090909
Sprint 3			0
Sprint 4			0
Sprint 5			0
Sprint 6			0
Sprint 7			0
Sprint 8			0
Sprint 9			

Figure 21: Sprint 2: Velocity Chart



Kickoff

Burndown Chart

kickoff.pmi.org Project Name: LabSync AI
Project Manager: Saad Tahir

Sprint 2 User Stories

US-1.1 → 8 SP (Auto Trigger Budget from Extraction)
US-1.2 → 8 SP (Auto Trigger Allocation from Budget)
US-2.1 → 14 SP (Gemini Budget Breakdown Generation)
US-3.1 → 12 SP (Intelligent Resource Allocation)
US-4.1 → 13 SP (Live Dashboard Updates)

Total Sprint Effort = 185 hours

	Starting Hours	Day 1	Day 2	Day 3	Day 4	Day 5
Actual Effort (Hours)	185	142	115	78	28	0
Ideal Effort (Hours)	185	148	111	74	37	0

Tasks	Starting Hours	Day 1	Day 2	Day 3	Day 4	Day 5	Total
Agent chain triggering	64	18	12	14	12	8	64
Budget generation	56	12	10	8	14	12	56
Allocation logic	48	8	8	10	12	10	48
Live dashboard + Socket.io	17	5	5	8	0	0	18

Figure 22: Sprint 2: Burndown Chart

10.3 Sprint 3 – Optimization, Quality & Release

Sprint Goal: Optimize performance, conduct full end-to-end testing, update documentation, and prepare for final release and demo, so that the system is stable, reliable, and ready for stakeholder presentation.

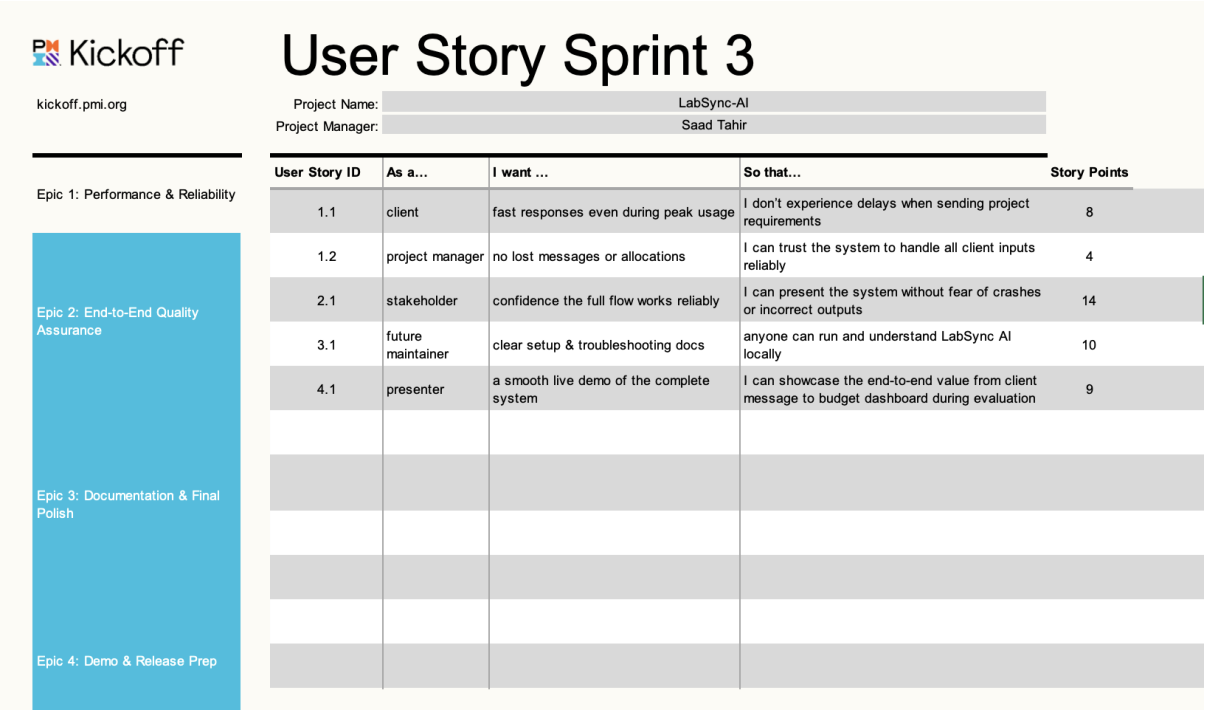


Figure 23: Sprint 3: Epics and User Stories

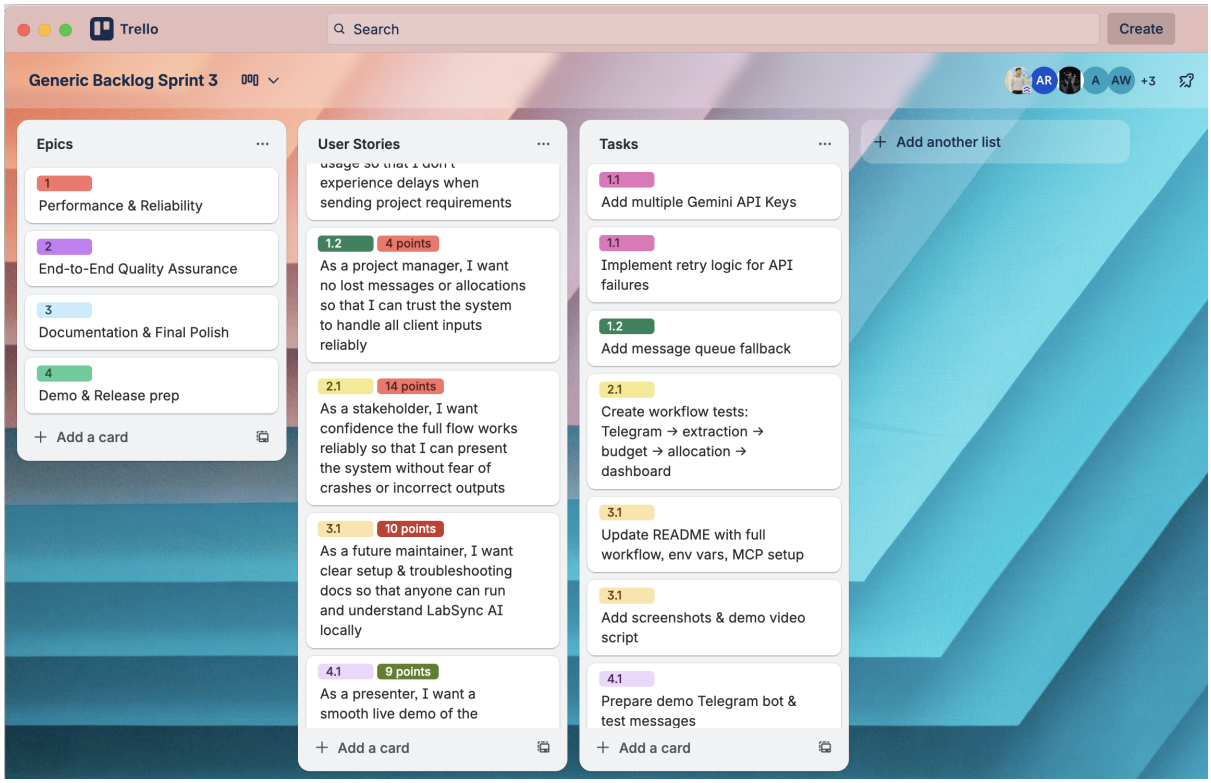


Figure 24: Sprint 3: Generic Backlog (Trello Board)

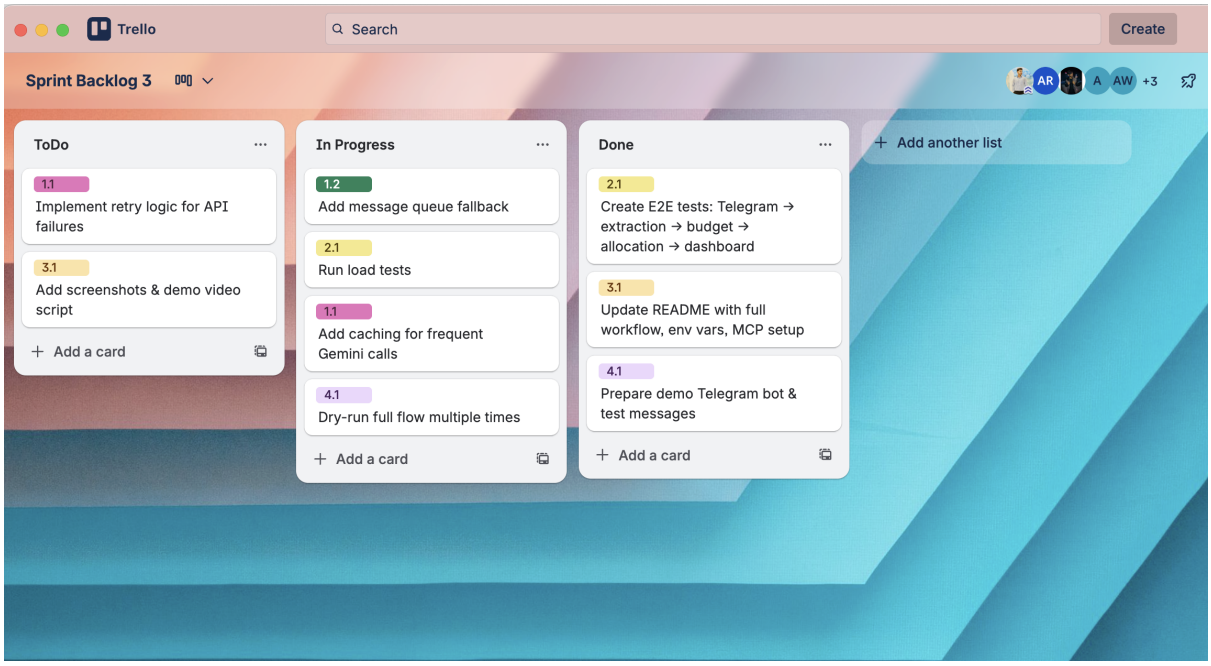


Figure 25: Sprint 3: Backlog (Trello Board)

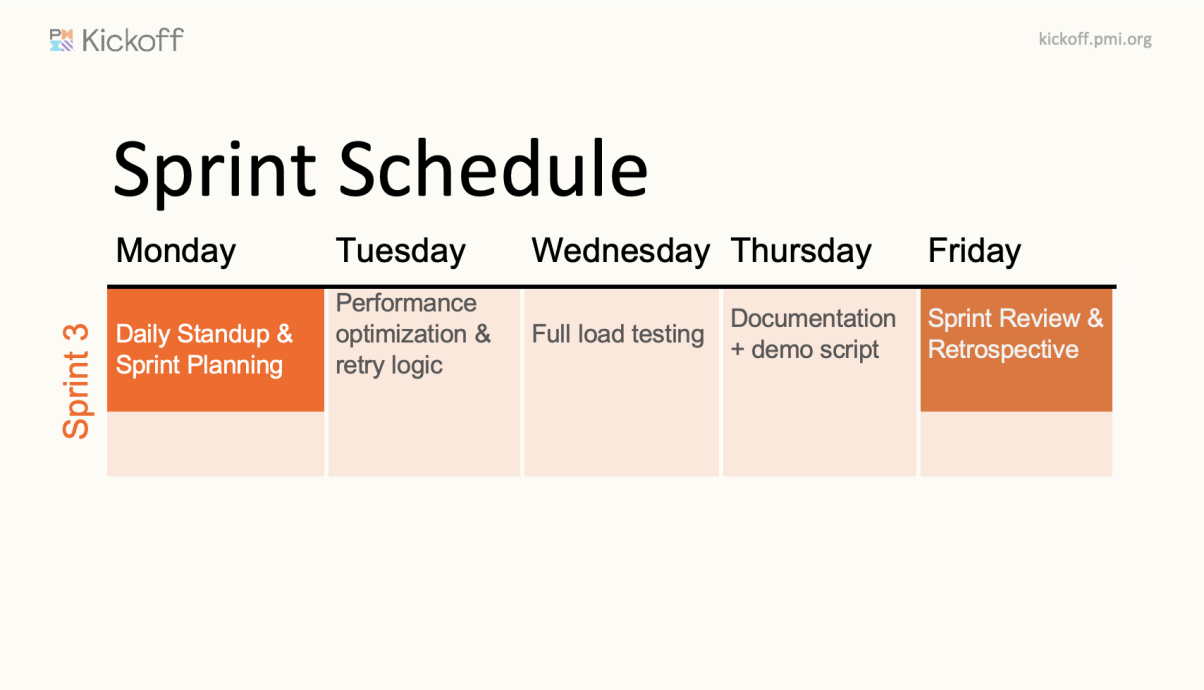


Figure 26: Sprint 3: Sprint Schedule

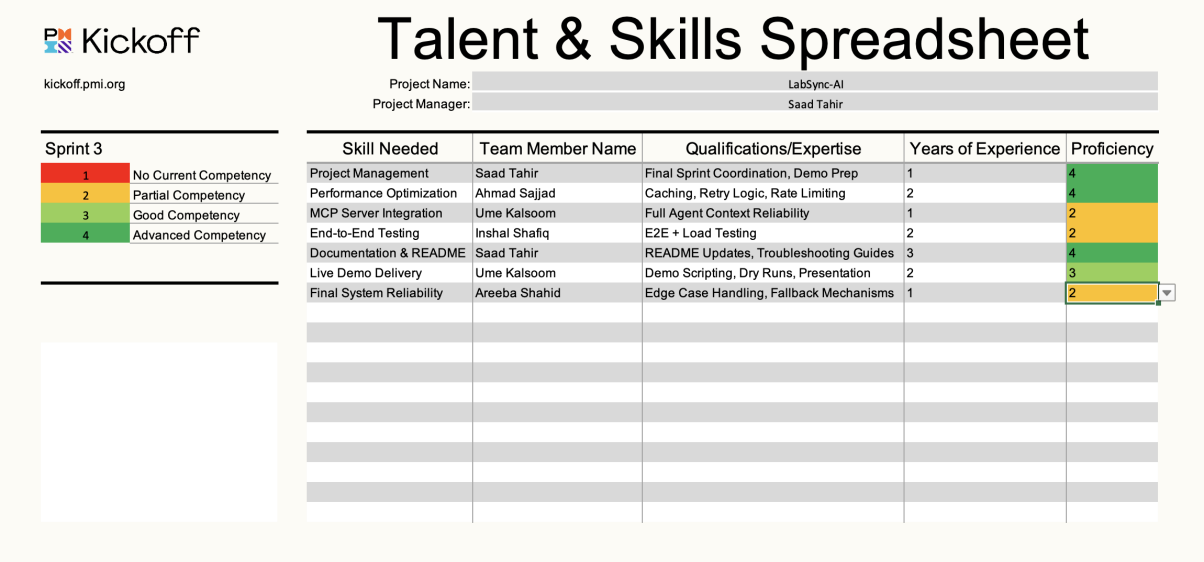


Figure 27: Sprint 3: Talent & Skills Sheet

Kickoff kickoff.pmi.org

Project Name: LabSync AI
Sprint #: 3

Sprint Review

Time	Duration	Activity	Description	Presenter
9:00	5 min	Introduction	LabSync AI final sprint & release prep	Saad
9:05	5 min	Sprint Goals	Optimize, full testing, documentation & demo preparation	Atif
9:10	5 min	Status Overview	System stable, tests passing, README updated	Ahmad Hassan
9:15	5 min	Live Demo	Complete end-to-end flow: Telegram input → full budget dashboard	Ayesha
9:20	5 min	Metrics	Planned: 45 SP → Completed: 44 SP → Velocity: 98%	Inshal
9:25	5 min	Blockers	Minor demo video editing polish needed; no major blockers	Kalsoom
9:30	5 min	Feedback	· Collect feedback from product owner	Prof Khaldoon

Figure 28: Sprint 3: Sprint Review

Kickoff kickoff.pmi.org

Project Title: LabSync AI
Sprint #: 3

Sprint Retrospective

What worked well this sprint?

- Testing caught critical bugs early → very high confidence in demo.
- Documentation updates were thorough and shared across teams.
- Final velocity was excellent; team worked smoothly together.
- Live demo rehearsals made presentation very polished.

What didn't work so well this sprint?

- Last-minute UI polish requests from stakeholder feedback.
- Some performance edge cases (high message volume) were discovered late.
- README troubleshooting section could have been updated earlier.

What will we do to improve next sprint?

- Start performance/load testing in Sprint 2 of integration phase.
- Involve stakeholders for early UI feedback (Sprint 8 demo).
- Maintain documentation continuously, not just in last sprint.
- Cross-team Trello + daily standups were the key to success — continue this practice.
- Overall: Proud of delivering a fully working AI-powered budget system!

Figure 29: Sprint 3: Sprint Retrospective

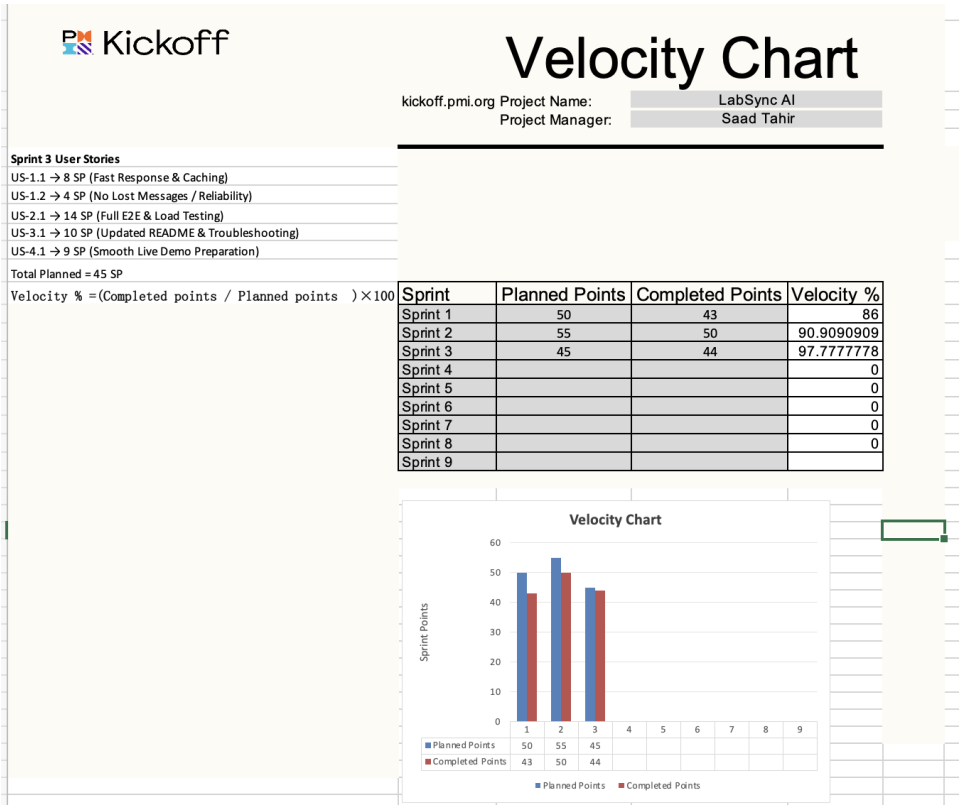


Figure 30: Sprint 3: Velocity Chart

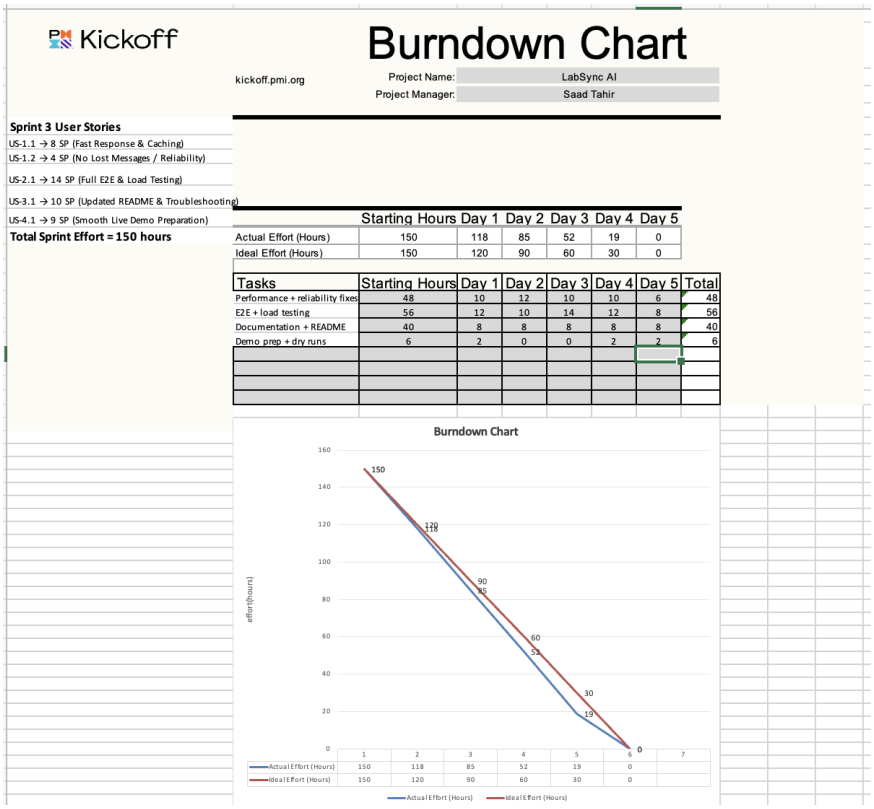


Figure 31: Sprint 3: Burndown Chart

11 Conclusion

LabSync360 provides a unified and intelligent approach to computer lab management by replacing manual and fragmented processes with an integrated, Agentic AI system. Through its web-based dashboard, real-time communication, and decision-support capabilities, the platform improves resource utilization, budget control, and information transparency. As a result, LabSync360 reduces administrative overhead, minimizes operational delays, and enables more effective and data-driven management of computer lab operations.