LMU EVENTFLOW

BY –
VRAJ PATEL
DIVY PATEL
JINIL PATEL
AYUSH PRABHAKAR
JAY PANCHAL

04/30/2024

Purpose of project

- We observed that LMU lacked a unified event management portal (scheduling, resources, approvals).
- The team was passionate about solving real campus problems – we personally experienced fragmented event planning at LMU.
- This project is worthwhile because it automates scheduling, reduces conflicts, and streamlines communication between students, faculty, and campus services.

Project goals

Centralized Scheduling: Create and approve events in one portal (prevent double-booking of rooms/times).

Role-Based Access: Allow students, faculty, and various campus departments (Sodexo catering, ITS, Facilities, etc.) to interact with appropriate features.

Resource Requests: Enable organizers to request support (food, tech, security, parking) as part of event planning.

Real-Time Communication: Provide live chat between event organizers and departments (for updates and confirmations).

Project description

LMU EventFlow is a role-based event management system designed exclusively for the Loyola Marymount University (LMU) community.

It streamlines the process of creating, approving, discovering, and managing events for students, faculty, and administrators while also allowing real-time resource coordination through chat features.

The platform emphasizes security by allowing login only for users with valid LMU emails (@lmu.edu and @lion.lmu.edu) and provides a seamless personalized dashboard experience based on user roles.

Data Flow

Login: User enters @lmu.edu email ⇒ OTP sent via email, after verifying OTP, backend grants access and loads the dashboard.

Event Creation: User fills event title, description, date/time. Data is sent to server and saved in the database. An admin is notified to review/approve.

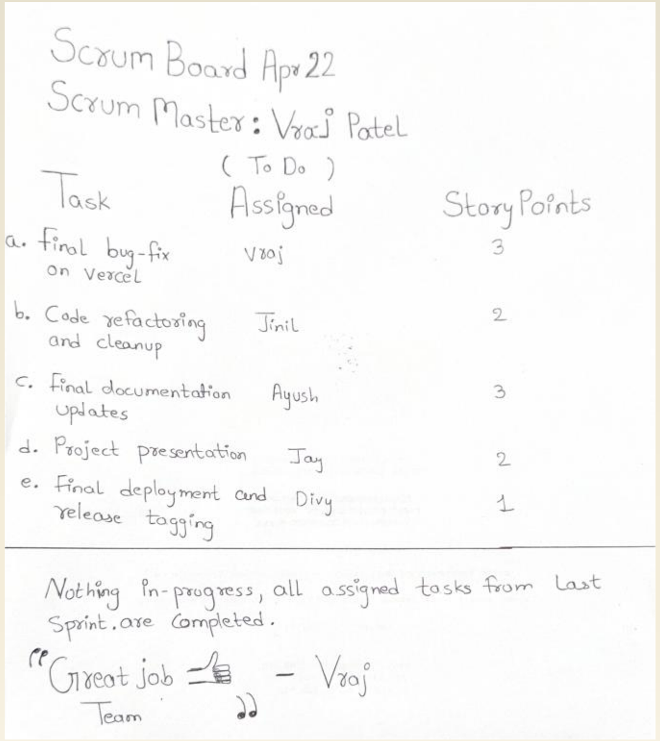
Approval Workflow: Admins see pending event requests (Approve/Delete). Once approved, events appear in the "Upcoming Events" list for all (with RSVP).

Resource Requests: User selects a department (e.g. ITS) and starts a chat. Messages are routed through the system to the appropriate department members.

Room Booking: Users can browse available rooms and time slots (we demo a classroom-booking page) and submit a booking, which is confirmed if free.

Project planning

Tool: We used a Scrum board in simple notebook style later digitalized into excel file



Scrum Board Apr 22
Scrum Master: Vraj Patel

(To Do)

Task	Assigned	Story Points
a. final bug-fix on Vercel	Vraj	3
b. Code refactoring and cleanup	Jinil	2
c. Final documentation Updates	Ayush	3
d. Project presentation	Jay	2
e. Final deployment and release tagging	Divy	1

Nothing in-progress, all assigned tasks from Last Sprint. are Completed.

"Great job Team" - Vraj

Burn-Up Plan for Project

Goal: Complete 90 story points by the end of Sprint 13 (April 26).

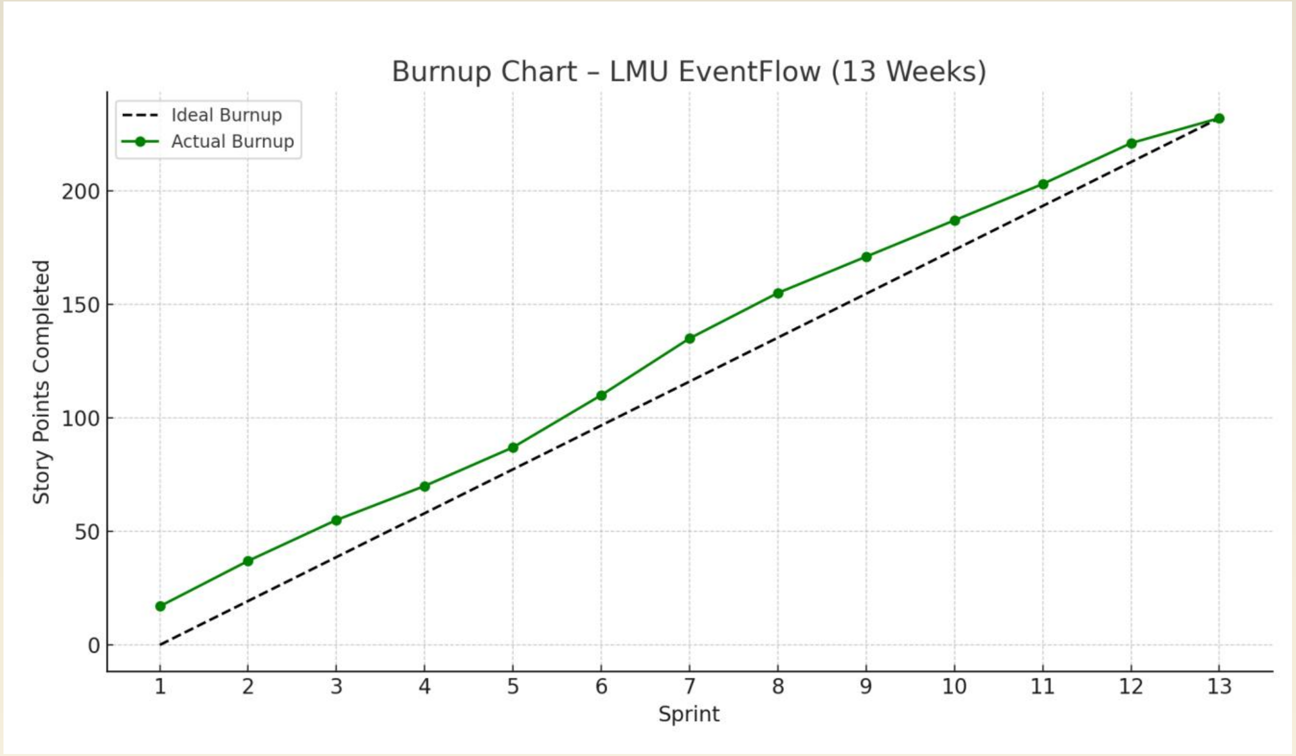
Progress:

The project maintained steady momentum, consistently adding completed story points sprint by sprint.

The actual progress closely follows the ideal trajectory, indicating effective sprint planning and execution.

By Sprint 13, all major features, event flows, and system requirements were completed as planned.

Burn-Up Chart



Burn-Down Plan for Current Iteration

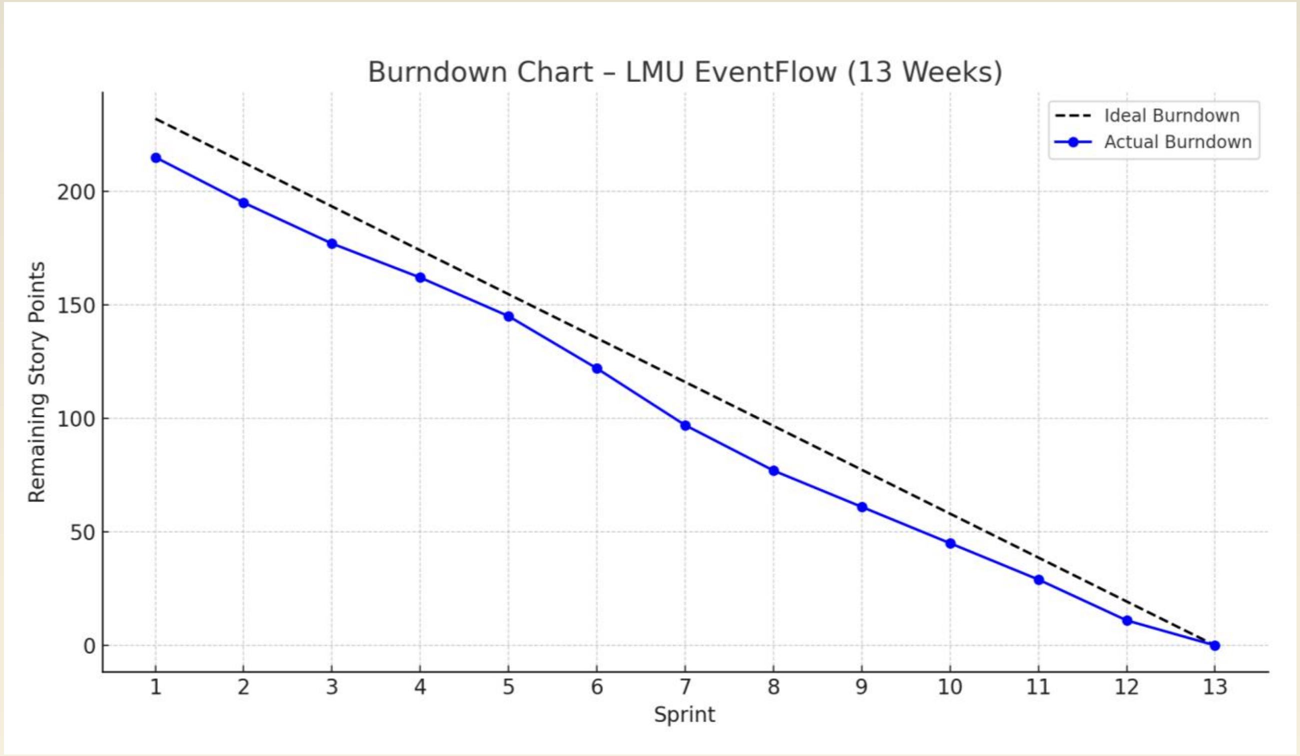
Goal: Reach 0 remaining story points by the end of Sprint 13.

Progress:

Despite minor variations during the mid-sprints (typical in Agile projects), the team consistently worked toward reducing the workload.

Gradual and disciplined progress ensured that by Sprint 13, all pending work was completed, achieving a full burndown to zero before the final delivery.

Burn-Down Chart



Agile Methodology – Scrum

Method Used: We adopted Scrum with 1-week sprints. We held short weekly stand-ups and sprint retrospectives.

Why Scrum: It provided a structured way to manage changing requirements. We could adapt our scope each sprint based on team feedback. Scrum matched our need for frequent demonstrations to get quick feedback.

Effectiveness: Overall, Scrum helped us deliver an MVP incrementally. However, since we were students, sticking rigidly to the sprint schedule was sometimes hard (midterms and holidays interfered).

Agile Methodology - Scrum

Advantages:

Feedback Loops: Regular sprint reviews let us get continuous feedback (so we could refine the UI and features).

Transparency: The board and daily updates kept everyone aware of progress.

Flexibility: We could re-prioritize the backlog mid-project if needed (e.g. adding "Live Preview" feature on feedback).

Drawbacks:

Overhead: Scrum involves meetings (we had to invest time in stand-ups, planning).

Estimation Challenges: As students, we sometimes misestimated tasks (leading to scope creep within sprints).

Rigidity: Rigid sprint deadlines occasionally conflicted with academic schedules.

DEMO

LMU-EVENTFLOW GITHUB

LMU-EVENTFLOW LIVE

Project Challenges

Challenge: Designing role-based navigation and ensuring each user saw only relevant options (e.g., faculty shouldn't see admin-only buttons).

Mitigation: We introduced a Change User Role admin panel. After login, the front-end requests the user's role from the server. Depending on the role, the dashboard shows a different set of menu cards. We carefully structured our React components with conditional rendering. In testing, we created accounts for all roles (Student, Faculty, ITS, Sodexo, Admin) to verify correct access control.

Agile Link: We initially missed this in early sprints. after a sprint review feedback session, we realized role enforcement was weak.

Based on Agile practices, we quickly adapted the next Sprint backlog to prioritize fixing role-based security.

This demonstrated Agile responsiveness — identifying a critical gap early and correcting it before launch through iterative feedback.

Project Challenges

Challenge: Keeping sprint goals realistic and achievable despite evolving feature ideas and mid-sprint discoveries.

Mitigation: We learned to tightly define our Sprint Goals at the beginning of each sprint, focusing only on "Must Have" tasks.
Whenever new ideas came up during a sprint (example: adding a confirmation toast for RSVP), instead of adding them immediately, we logged them into the Product Backlog and considered them in Sprint Planning for future sprints.

Agile Link: During our Sprint Retrospectives, we identified that trying to "squeeze in" extra tasks mid-sprint was harming our sprint velocity.
Applying Agile best practices, we enforced a Sprint Commitment Rule:

"No new scope during active sprint unless critical."
This made our delivery much more predictable and helped us build discipline in backlog management, a core Agile value.

Changes During Development

Change:
We initially planned auto-approval of events, but during early sprint testing, we realized this could allow low-quality or spam events.

What Happened to Force the Change:
Internal team review identified the risk. We decided to add an admin approval step to ensure event quality.

Was It a Substantial Change or Minor Adaptation?
It was a minor adaptation — adding a pending/approved state in the backend and a simple admin approval screen. Took about one sprint.

What Would Have Happened If We Stayed the Course?
The events list could have become unreliable, damaging trust in the system.

Thank You!

(Questions Please...)