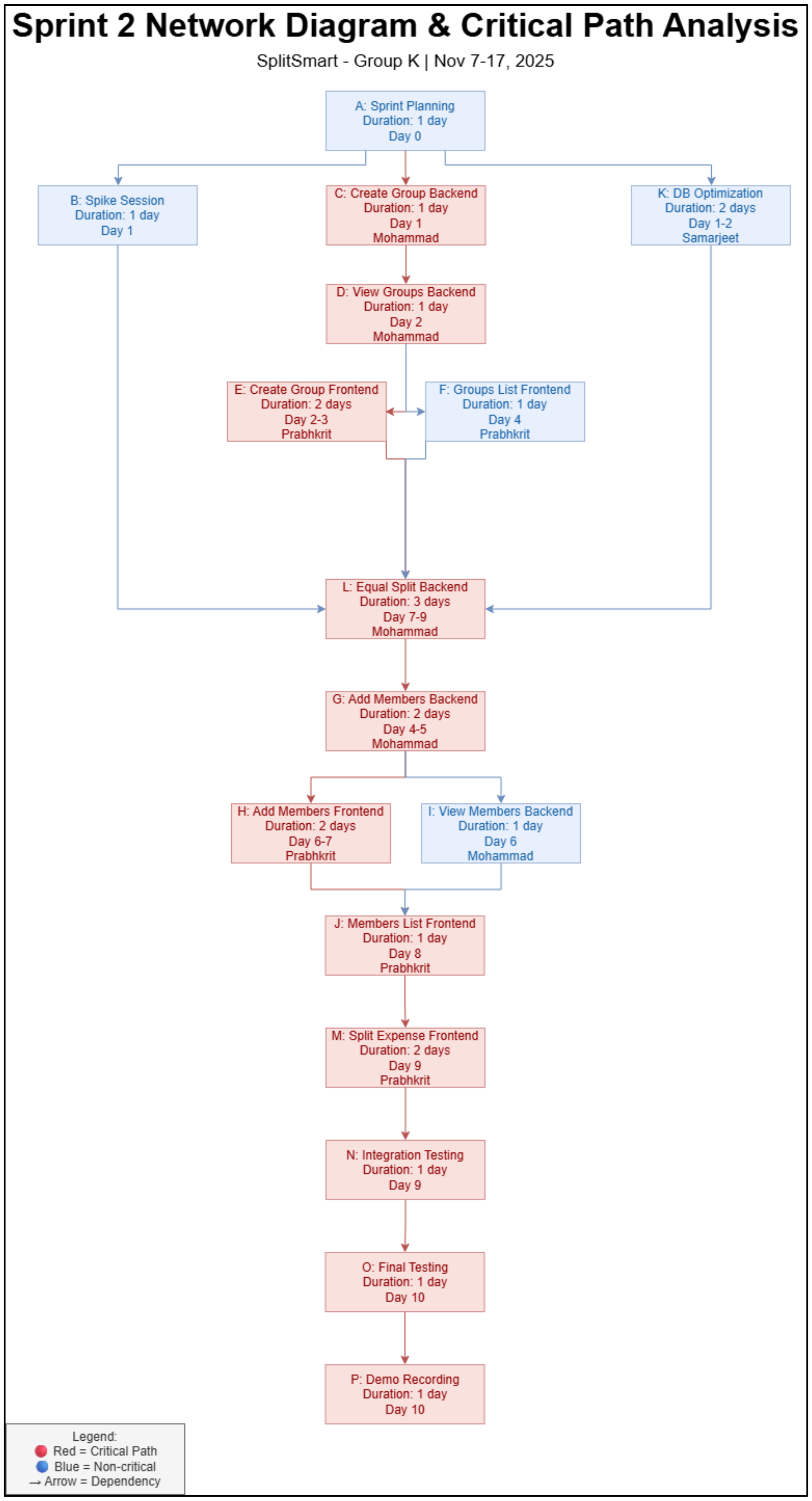
# Sprint 2 Schedule


Sprint 2 Network Diagram & Critical Path Analysis
SplitSmart - Group K | Nov 7-17, 2025
A: Sprint Planning Duration: 1 day Day 0
B: Spike Session Duration: 1 day Day 1
C: Create Group Backend Duration: 1 day Day 1 Mohammad
K: DB Optimization Duration: 2 days Day 1-2 Samarjeet
D: View Groups Backend Duration: 1 day Day 2 Mohammad
E: Create Group Frontend Duration: 2 days Day 2-3 Prabhkrit
F: Groups List Frontend Duration: 1 day Day 4 Prabhkrit
L: Equal Split Backend Duration: 3 days Day 7-9 Mohammad
G: Add Members Backend Duration: 2 days Day 4-5 Mohammad
H: Add Members Frontend Duration: 2 days Day 6-7 Prabhkrit
I: View Members Backend Duration: 1 day Day 6 Mohammad
J: Members List Frontend Duration: 1 day Day 8 Prabhkrit
M: Split Expense Frontend Duration: 2 days Day 9 Prabhkrit
N: Integration Testing Duration: 1 day Day 9
O: Final Testing Duration: 1 day Day 10
P: Demo Recording Duration: 1 day Day 10
Legend:
🔴 Red = Critical Path
🔵 Blue = Non-critical
→ Arrow = Dependency

*Note: L isn't in the critical path*

## Sprint 2 Critical Path Analysis

Critical Path:

A → C → D → E → G → H → J → M → N → O → P

- Total Duration: 10 working days
- Tasks on Critical Path: 11 tasks
- Non-Critical Tasks: 5 tasks (B, F, I, K, parallel work)
- Sprint Status: Completed

WHAT KEPT SPRINT ON SCHEDULE:

1. Proper Task Sequencing
   - Backend APIs completed before frontend UI work began
   - Clear dependencies identified during sprint planning
   - No frontend blockers waiting for backend completion
   - Task breakdown aligned with team specializations
2. Parallel Work Optimization
   - Samarjeet handled DB optimization (Days 1-2) parallel to feature work
   - Frontend and backend teams worked independently when possible
   - Spike session (Task B) completed parallel to critical path
   - Non-critical tasks (F, I) didn't delay critical path
3. Early Risk Mitigation
   - Spike session on equal splitting algorithm completed Day 1
   - Complexity reduced before implementation started
   - Weekly sync meeting (Day 7) caught potential issues early
   - Daily standups identified and resolved blockers immediately
4. Strong Team Coordination
   - Clear ownership: Mohammad (Backend), Prabhkrit (Frontend), Samarjeet (DB/DevOps)
   - All team members completed assigned tasks on schedule
   - Communication via daily standups prevented delays
   - PR review process didn't create bottlenecks
5. Strategic Testing Approach
   - Integration testing started Day 9 (not last minute)
   - Final testing on Day 10 provided cushion for demo prep
   - Bug fixes handled during final testing day
   - Demo recorded successfully with all features working

## RISK MANAGEMENT:

Risks Identified During Planning:

- Critical path had zero slack time (any delay = sprint delay)
- Equal splitting algorithm complexity unknown initially
- Frontend entirely dependent on backend API completion
- Time pressure before Nov 17 deadline

Mitigations Successfully Applied:

- ✓ Spike session Day 1 reduced algorithm complexity from 8 SP to manageable
- ✓ Backend tasks prioritized to unblock frontend work early
- ✓ Daily standups caught blockers before they caused schedule delays
- ✓ DB optimization done in parallel by Samarjeet (no critical path impact)
- ✓ Weekly sync meeting Day 7 provided mid-sprint checkpoint

## DEPENDENCIES & SEQUENCING:

Critical Dependencies (Must be Sequential):

- Task C (Create Group Backend) → Task D (View Groups Backend)
- Task D → Task E (Create Group Frontend)
- Task G (Add Members Backend) → Task H (Add Members Frontend)
- Task H → Task J (View Members Frontend)
- Task L (Equal Split Backend) → Task M (Split Expense Frontend)

Parallel Work Opportunities:

- Tasks B, C, K all started Day 1 (3 parallel tracks)
- Tasks E and F ran parallel (both frontend, different features)
- Tasks H and I ran parallel (frontend and backend, different features)
- Task L merged inputs from B and K (efficient convergence)

## LESSONS LEARNED:

1. Critical Path Management
   - Zero slack worked but required perfect execution
   - Proper backend → frontend sequencing prevented blockers
   - Weekly syncs essential for mid-sprint course correction
   - Daily standups caught issues before they became delays
2. Task Estimation Accuracy
   - 20% buffer added to estimates (lesson from Sprint 1) worked well
   - Equal splitting (8 SP) took full 3 days as estimated
   - Carried-forward tasks ([2.1], [2.2]) completed faster (80% done already)
   - Average velocity: 12 SP/week exceeded Sprint 1's 6.5 SP/week
3. Parallel Work Optimization
   - DB optimization parallel to features saved critical time
   - Samarjeet's work (K) didn't delay critical path
   - Could have started integration testing Day 7 vs Day 9 (add buffer)
   - Non-critical tasks provided natural flexibility

4. For Sprint 3 Planning
   - Add 1-day buffer to critical path where possible
   - Start integration testing by Day 7 (not Day 9)
   - Continue spike sessions for complex features
   - Maintain daily standups for early issue detection
   - Keep backend → frontend sequencing pattern

**SPRINT 2 OUTCOME:**

- ✓ All 5 core stories completed (24 Story Points)
- ✓ Zero delays on critical path
- ✓ Demo recorded on time with working features
- ✓ 100% sprint success rate (5 of 5 stories done)
- ✓ Velocity increased 85% from Sprint 1 (6.5 → 12 SP/week)
- ✓ Test coverage: 82% backend, 75% frontend
- ✓ Zero critical bugs at demo time

Sprint 2 was completed successfully with excellent schedule adherence. The critical path analysis conducted during planning helped identify dependencies early, and parallel work optimization kept the sprint on track. The team demonstrated strong coordination and execution.