Schedule Diagram – Sprint 2

Group Name: Chroma Notes

Group Members:

Giuseppe D’Orazio

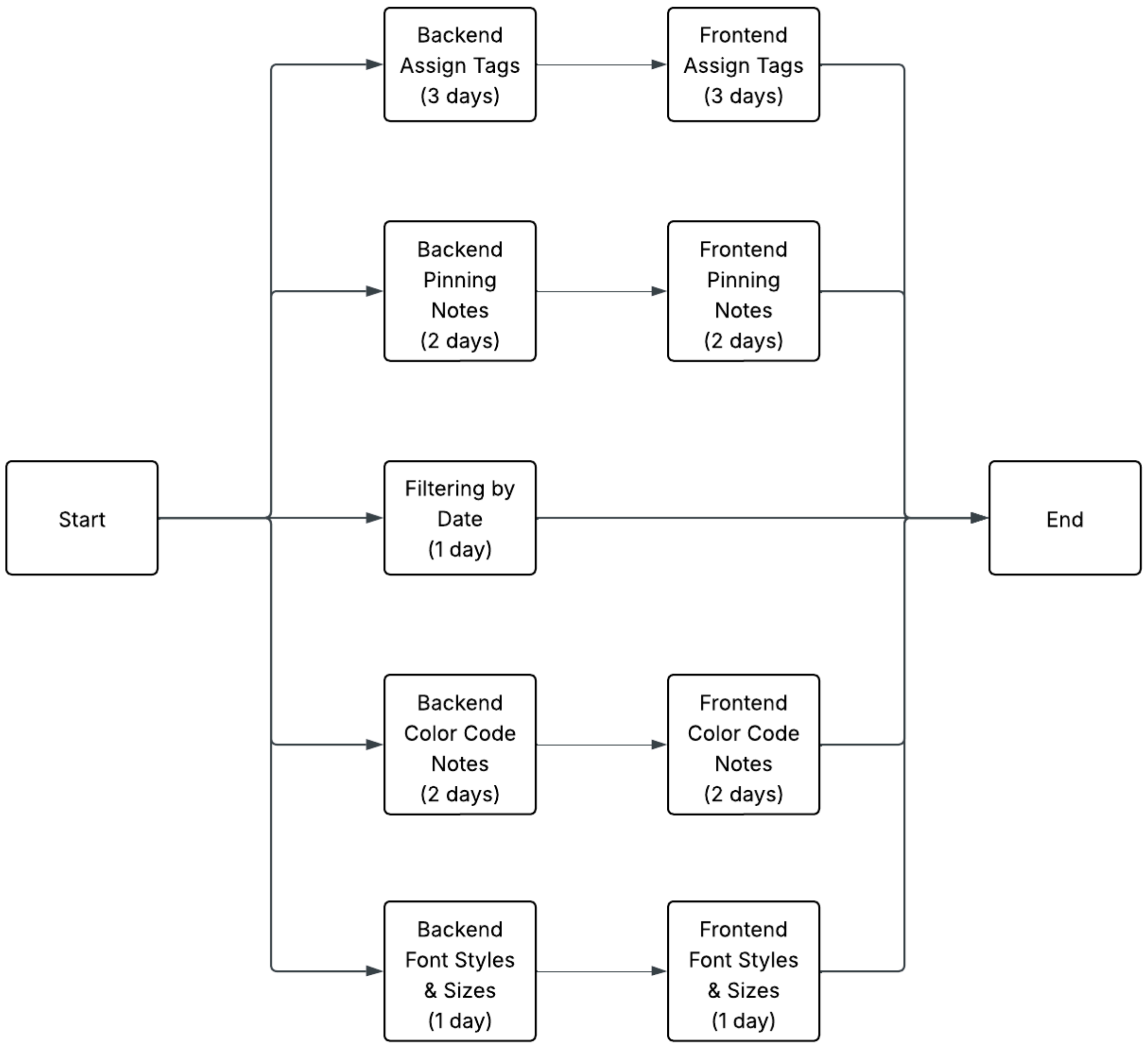
Ravneet Deol

Victoria Bolognese

Araz Manouchian

Date: November 16, 2025

Network Diagram:


Start
Backend Assign Tags (3 days)
Frontend Assign Tags (3 days)
Backend Pinning Notes (2 days)
Frontend Pinning Notes (2 days)
Filtering by Date (1 day)
Backend Color Code Notes (2 days)
Frontend Color Code Notes (2 days)
Backend Font Styles & Sizes (1 day)
Frontend Font Styles & Sizes (1 day)
End

Task Dependences:

The network diagram illustrates all task dependencies for Sprint 2. Several tasks could be done in parallel, but some required the backend to be completed before the frontend could begin.

Backend → Frontend Dependences

1. Pinning Notes
    - Backend Pinning Notes → Frontend Pinning notes
    - 2 days backend, following by 2 days frontend
2. Assign Tags
    - Backend Assign Tags → Frontend Assign Tags
    - 3 days backend, followed by 3 days frontend
3. Color Code Notes
    - Backend Color Code Notes → Frontend Color Code Notes
    - 2 days backend, followed by 2 days frontend
4. Font Styles & Sizes
    - Backend Font Styles & Sizes → Frontend Font Styles & Sizes
    - 1 day backend, followed by 1 day frontend

Independent Task

1. Filtering by Date
    - This task was independent of all others, since it only required existing note timestamps
    - Duration: 1 day

Critical Path Analysis:

The critical path represents the longest sequence of dependent tasks that determines the minimum time needed to complete the sprint's work.

The longest dependency chain is:

1. Backend Assign Tags (3 days)
2. Frontend Assign Tags (3 days)

Total duration of this chain: 6 days

This is longer than all other chains:

- Pinning notes: 2 days + 2 days = 4
- Color Coding Notes: 2 days + 2 days = 4
- Font Styles & Sizes = 1 day + 1 day = 2
- Filtering by Date: 1 day, independent

Conclusion:
The Assign Tags user story forms the critical path
Any delay in either Backend or Frontend would delay the entire sprint, because it is the longest chain of dependent tasks

Ensuring the Sprint Stayed on Schedule:

To keep the sprint on schedule, our team followed several coordination and planning strategies

- Backend-First Prioritization
    - Whenever possible, backend tasks were prioritized so the frontend team could build on stable, completed endpoints
- Adaptive Workflow When Backend Was Delayed
    - In situations where backend work could not begin immediately, the frontend team did not stay idle. They started preparing UI components and layouts ahead of time. Once backend work finished, the frontend developers integrated the final API connections and completed their tasks. This prevented the sprint from falling behind and kept both teams productive
- Workload Balancing
    - Tasks were distributed fairly across team members so no single person was overloaded, and parallel backend tasks could progress simultaneously
- Task Independence
    - Independent tasks such as Filter by Date were intentionally done during lighter workload periods or while waiting for a dependency chain to finish
- Consistent Team Communication
    - We maintained open communication throughout the sprint. Whenever someone finished early or was blocked, they altered the team so tasks could be reprioritized as needed.