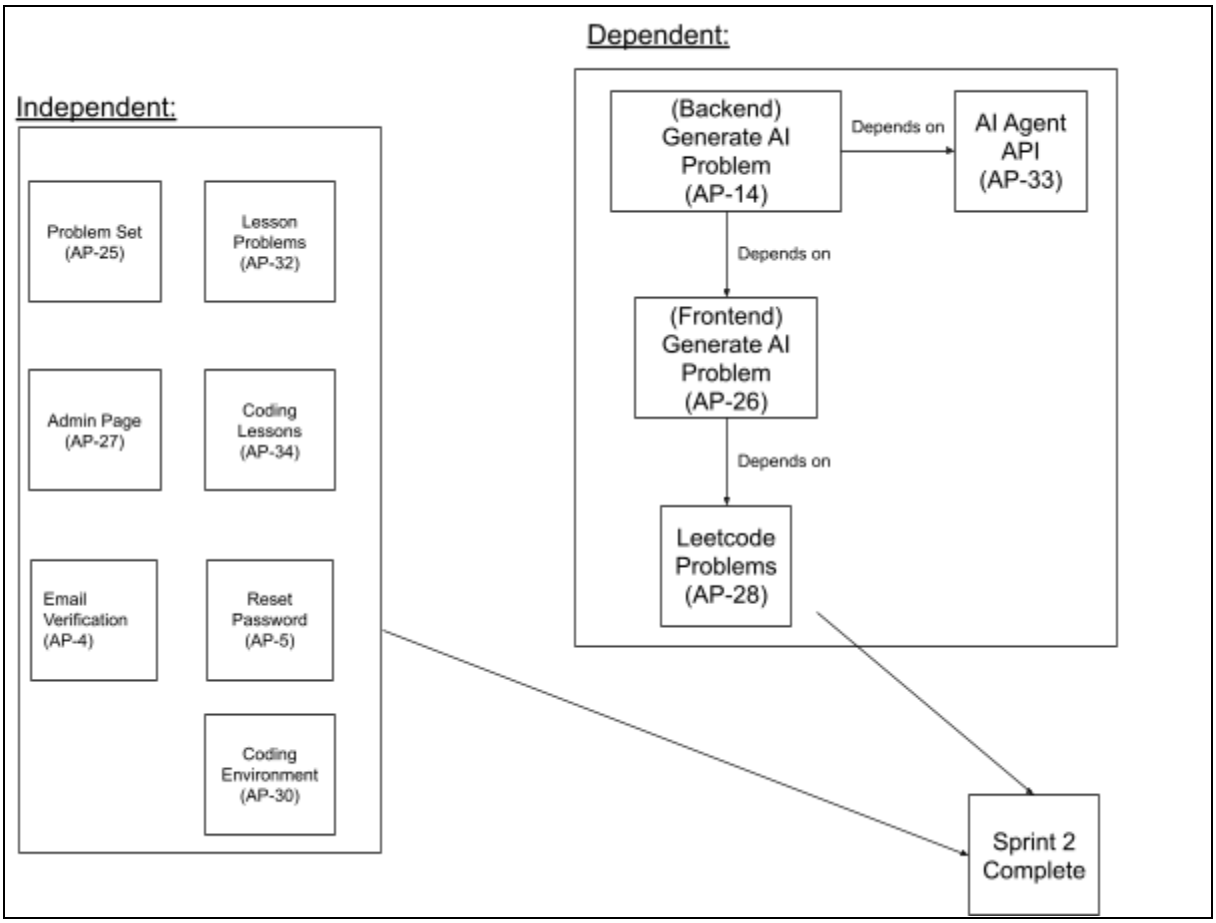


During Sprint 2, we identified tasks with dependencies and mapped them in a network diagram to determine the **critical path**. Tasks that relied on database schemas or specific API endpoints were prioritized early to avoid bottlenecks later in the sprint.

## Strategies to Keep the Sprint on Schedule

To ensure that work progressed smoothly and that dependencies did not cause delays, we implemented the following strategies:

- **Daily Standups**: Regular check-ins helped identify blockers early and keep everyone aligned.
- **Pair Programming for Critical Tasks**: This approach improved efficiency and ensured complex issues were tackled collaboratively.
- **Early Planning for Database-Dependent Tasks**: At the start of the sprint, we scheduled meetings to discuss and finalize database schema requirements, ensuring that dependent tasks could proceed without delays.

## Challenge: Why We Could Not Complete AP-34

One significant challenge was **AP-34**, which we were unable to complete due to unforeseen complexity. The reasons were:

- The API endpoint required handling **more cases than initially expected**, increasing development time.
- Complex **validation logic** was necessary to ensure progress tracking worked correctly.

## Lessons Learned

This experience highlighted the importance of:

- **Better Story Point Estimation**: We underestimated the complexity of FIN-53, which led to delays. Moving forward, we will allocate more time for analyzing tasks before assigning story points.
- **Frequent Check-ins to Mitigate Delays**: We held meetings to identify blockers early and reassign work as needed to keep the sprint moving forward.