Project SAttend: Task Dependencies and Schedule Analysis

Project Overview

Project SAttend is an attendance system designed to simplify the process of recording attendance in educational environments. This document analyzes the implementation of the Teaching Assistant login functionality, which allows TAs to access the same system as instructors to assist with attendance reporting.

Task Identification and Dependencies

Based on the analysis of the GitHub repository, we identified the following key tasks required to implement the TA login functionality:

Task ID	Task Name	Prerequisites	Duration (days)
T1	System Analysis & Requirements Definition	None	2
T2	Data Model Design	T1	2
T3	Backend API Design	T1	1
T4	Database Schema Design	T2	1
T5	User Authentication Implementation	T2, T3	3
T6	Course Access Control Implementation	T2, T3	2
T7	API Implementation	T5, T6	3
T8	Frontend Login UI Design	T3	1
T9	Frontend Login UI Implementation	T8	2
T10	Frontend Course Management UI	T7, T9	3
T11	Integration Testing	T7, T10	2
T12	Deployment & Documentation	T11	1

Task Details

1. **System Analysis & Requirements Definition**
   - Analyze existing system architecture
   - Define specific requirements for TA login functionality
   - Identify necessary changes to support multiple user roles
2. **Data Model Design**
   - Design user model with role support (Instructor, TA)
   - Define access control and permission models
   - Design relationship between users and courses
3. **Backend API Design**

- Define authentication API endpoints
- Design course access APIs
- Plan user management endpoints

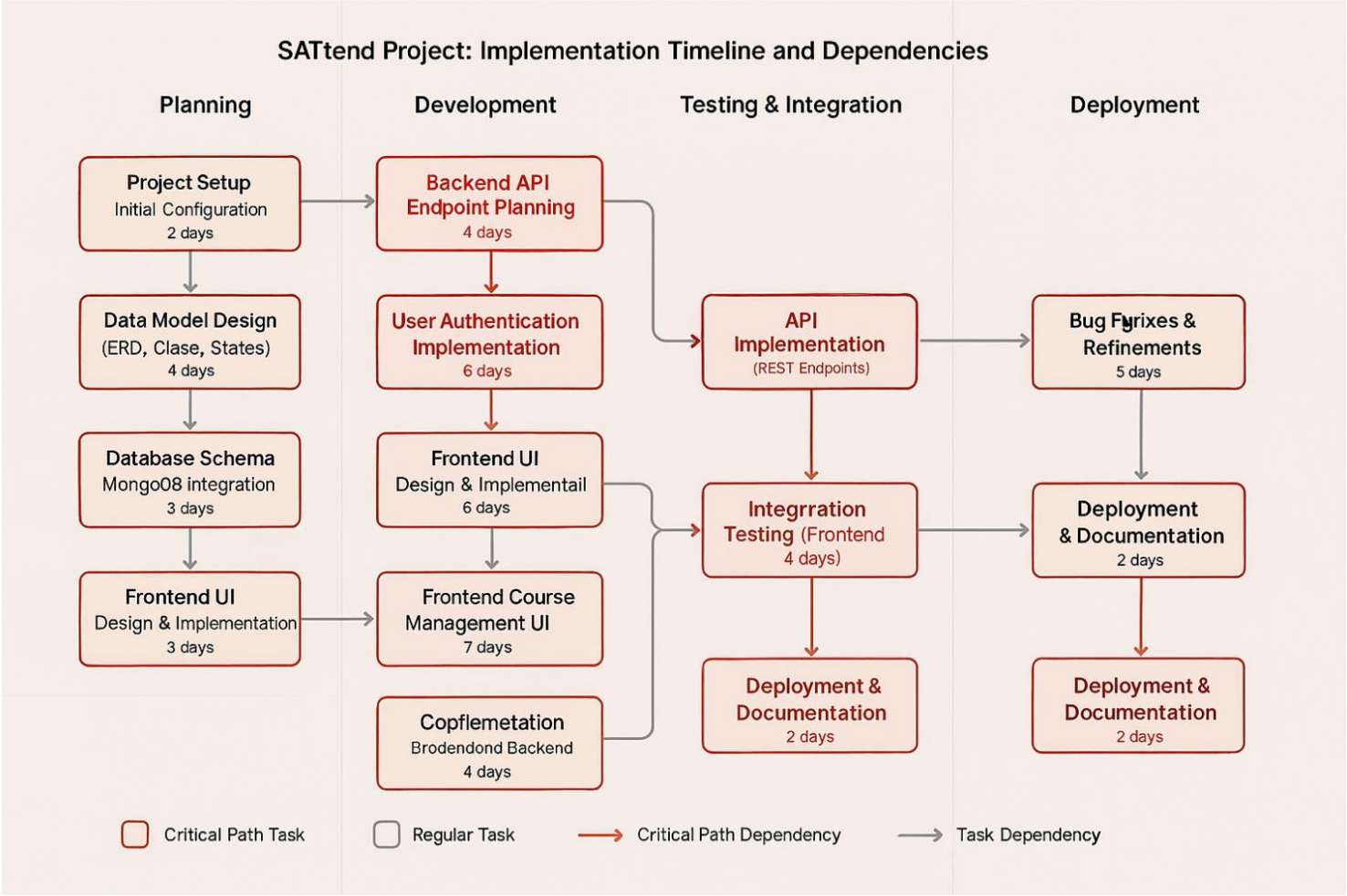
4. **Database Schema Design**
   - Design MongoDB schema for user objects
   - Create schema for user-course relationships
   - Define indexes and access patterns
5. **User Authentication Implementation**
   - Implement user login and validation
   - Create password security mechanisms
   - Implement session management
6. **Course Access Control Implementation**
   - Implement permission checking logic
   - Create course assignment functionality
   - Implement access controls for attendance records
7. **API Implementation**
   - Implement RESTful API endpoints
   - Configure CORS support
   - Implement error handling
8. **Frontend Login UI Design**
   - Design login interface wireframes
   - Create visual design assets
   - Design role-specific UI elements
9. **Frontend Login UI Implementation**
   - Implement React login component
   - Create authentication state management
   - Implement form validation
10. **Frontend Course Management UI**
    - Implement course listing interface
    - Create attendance management UI
    - Add role-specific UI elements and permissions
11. **Integration Testing**
    - Test frontend-backend integration

- Verify role-based access controls
- Test authentication flows

12. **Deployment & Documentation**
    - Deploy updated system
    - Document new features
    - Create user guides for TAs and instructors

Network Diagram

Below is a network diagram based on the task dependencies described above, using a modern Activity- on-Node (AON) format with full task names:



The diagram clearly shows all project tasks with their full names and durations. Critical path tasks are highlighted in red, while standard dependencies are shown in gray. Arrows indicate the flow and dependencies between tasks.

Critical Path Analysis

Based on the network diagram and task duration estimates, we calculated the following paths:

1. T1 → T2 → T4 → T5 → T7 → T11 → T12 = 2+2+1+3+3+2+1 = 14 days
2. T1 → T2 → T5 → T7 → T11 → T12 = 2+2+3+3+2+1 = 13 days
3. T1 → T2 → T6 → T7 → T11 → T12 = 2+2+2+3+2+1 = 12 days
4. T1 → T3 → T5 → T7 → T11 → T12 = 2+1+3+3+2+1 = 12 days

5. T1 → T3 → T6 → T7 → T11 → T12 = 2+1+2+3+2+1 = 11 days
6. T1 → T3 → T8 → T9 → T10 → T11 → T12 = 2+1+1+2+3+2+1 = 12 days

Through this analysis, we determined that the critical path is:

T1 → T2 → T4 → T5 → T7 → T11 → T12

This path has a total duration of 14 days. Any delay in tasks along this path will directly impact the overall project timeline.

Sprint Schedule Management Strategies

To ensure the sprint proceeded according to plan, we implemented the following measures:

1. Critical Path Focus

- Allocated more resources to tasks on the critical path
- Conducted daily progress reviews for critical path tasks
- Established early warning mechanisms for critical path tasks
- Created buffer time for high-risk tasks on the critical path (particularly T5: User Authentication)

2. Parallel Development Strategy

- Initiated parallel frontend and backend development immediately after design completion
- Backend team implemented user authentication and course management functionalities concurrently
- Frontend team began interface implementation using mock data once API design was completed
- Used feature branches to allow independent progress on separate components

3. Risk Management

- Identified technical risks: potential difficulties with MongoDB integration
- Identified resource risks: team's limited familiarity with Kotlin
- Developed contingency plans: prepared alternative approaches and additional learning resources
- Created time buffers for high-risk tasks

4. Agile Practices Application

- Synchronized progress and obstacles through daily stand-ups
- Used Kanban board to track task status
- Conducted regular code reviews to ensure quality
- Focused on feature-based releases rather than time-based iteration goals
- Maintained clear documentation of APIs to facilitate parallel work

Sprint Execution Analysis

Plan Execution Status

During actual implementation, we encountered several key issues that affected the sprint timeline:

1. **Technical Integration Challenges**: MongoDB integration proved more difficult than anticipated, particularly when combined with Kotlin coroutines. The data access layer required significant refactoring to work correctly.
2. **Environment Configuration Delays**: Maven configuration issues led to difficulties in building and running tests. Inconsistent development environments across team members caused additional delays during integration.

3. **Cross-Platform Compatibility**: Inconsistent behavior occurred across different development environments (Windows/Mac), particularly with file path handling in the build process.

Plan Deviations

Referring to the network diagram, we experienced deviations from the plan in the following areas:

1. **T5 (User Authentication Implementation)**:
   - Originally planned for 3 days, actually took 5 days, resulting in a 2-day delay
   - Primary reason: Difficulties with integrating Spark framework and Jackson JSON library
   - Impact: Directly delayed the critical path
2. **T7 (API Implementation)**:
   - Originally planned for 3 days, actually took 4 days, resulting in a 1-day delay
   - Primary reason: CORS configuration and error handling logic were more complex than expected
   - Impact: Further extended the critical path delay
3. **T11 (Integration Testing)**:
   - Originally planned for 2 days, actually took 3 days, resulting in a 1-day delay
   - Primary reason: Unstable build environment due to Maven configuration issues
   - Impact: Added the final extension to the critical path delay

Overall, the sprint was delayed by 4 days, extending from the originally planned 14 days to 18 days, primarily due to delays along the critical path.

What Went Wrong

1. **Underestimation of Technical Complexity**:
   - The integration of modern technologies (Kotlin, MongoDB, React) proved more complex than anticipated
   - The team's varying levels of experience with these technologies wasn't adequately accounted for in estimates
2. **Environment Configuration Issues**:
   - Insufficient attention was given to standardizing development environments
   - Build configuration was not fully tested across all platforms before development began
3. **Inadequate Risk Buffers**:
   - Time estimates for critical path tasks were too optimistic
   - Insufficient buffer time was allocated for high-risk tasks

Lessons Learned

Through the execution of this sprint, we learned several important lessons:

1. **Technology Selection Requires More Thorough Evaluation**
   - More detailed feasibility studies should be conducted before adopting new technologies
   - Sufficient time should be allocated for learning and experimentation
   - Technical spikes should be used to validate complex integrations before committing to implementation approach
2. **Infrastructure First Approach**
   - Build and deployment environment stability should be prioritized
   - Standardized development environment configurations should be provided to the team
   - Docker could be used to ensure consistent environments across all team members
3. **Risk Buffers Are Essential**
   - Reasonable time buffers should be added to tasks on the critical path
   - PERT estimating should be considered for tasks with high uncertainty
   - Regular risk reassessment should be performed throughout the sprint
4. **Enhanced Knowledge Sharing**
   - Team knowledge base should be established to document common issues and solutions
   - Pair programming should be encouraged, especially when dealing with complex or novel technologies
   - Regular technical sharing sessions should be held to level up the entire team

Future Improvement Plan

Based on the experience from this sprint, we plan to implement the following improvements in future projects:

1. **Establish Technology Pre-Research Phase**
   - Add a dedicated technology exploration phase before formal development begins
   - Create Proof of Concept (POC) implementations to validate the feasibility of key technologies
   - Document key learnings from technical explorations
2. **Improve Estimation Methods**
   - Introduce more precise task breakdown
   - Use historical data to aid estimation
   - Consider experience level differences among team members
   - Add explicit risk buffers to high-uncertainty tasks
3. **Strengthen Automation**
   - Improve CI/CD processes to catch integration issues earlier
   - Increase automated test coverage

- Simplify environment setup steps with containerization
- Implement automated static code analysis

4. **Enhance Monitoring Mechanisms**
   - Implement daily progress metrics tracking
   - Conduct more frequent checkpoints for critical path tasks
   - Establish early warning systems to promptly identify and address deviations
   - Create visual dashboards for sprint progress monitoring

Through these improvements, we believe we can achieve better plan execution and delivery quality in future sprints.

Conclusion

The implementation of the TA login functionality for Project SAttend demonstrated both challenges and opportunities for improvement in our development process. While we successfully delivered the functionality, the 4-day delay highlights the importance of realistic estimation, risk management, and proper technical preparation.

The critical path analysis provided valuable insights into the interdependencies of our tasks and helped identify where our process broke down. By applying the lessons learned from this sprint, we are confident that future sprints will be executed more efficiently with fewer delays.

The teaching assistant login feature has added significant value to the SAttend system, enabling more efficient attendance recording and management through expanded access for teaching staff.