GLOW Project Schedule Analysis

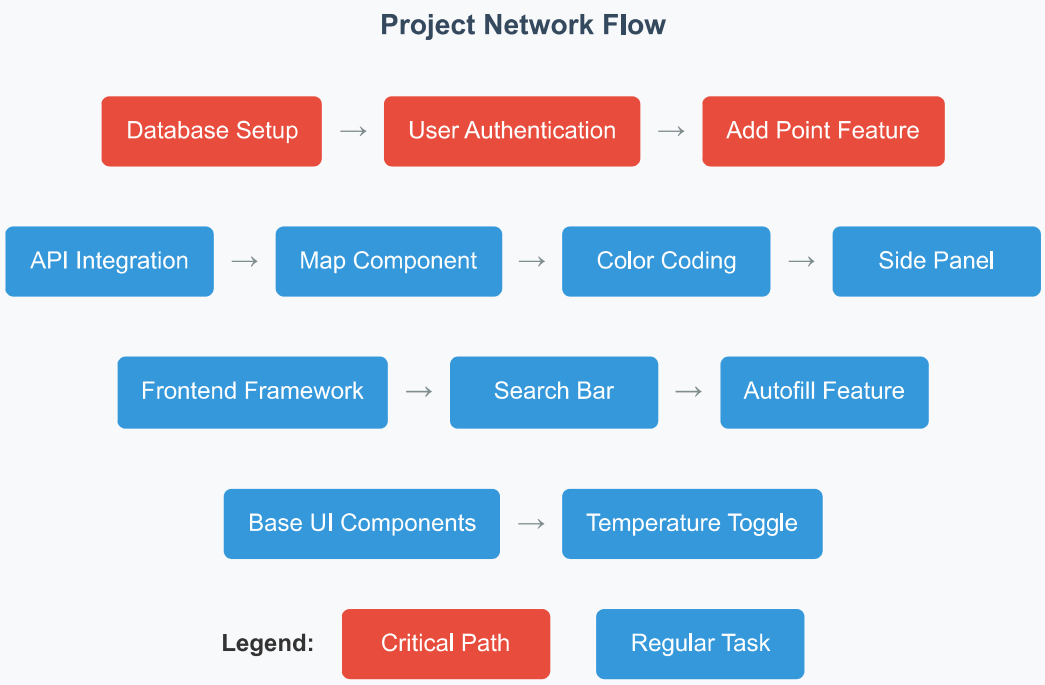
Sprint 2 - Network Diagram, Dependencies & Risk Assessment

Team: Microsofties | **Date:** July 6, 2025

Executive Summary

This document provides a comprehensive analysis of the GLOW project's Sprint 2 implementation, including network diagrams, dependency analysis, critical path identification, and risk assessment. The analysis covers both completed features and planned but unmet goals, providing insights into project management effectiveness and lessons learned.

Network Diagram - Sprint 2 Task Dependencies



Dependencies Analysis

External Dependencies

Dependency	Type	Impact	Mitigation Strategy
MongoDB Database	Infrastructure	High - Required for user authentication and data storage	
OpenWater API	External Service	High - Primary data source for temperature readings	API key management and fallback data
Node.js Runtime	Technology	High - Backend execution environment	Version 18+ LTS with npm 8+
React/Next.js	Technology	High - Frontend framework	Latest stable versions with backward compatibility
Leaflet.js	Library	Medium - Map visualization	OpenStreetMap integration with fallback tiles

Internal Dependencies

Task	Depends On	Dependency Type	Impact if Delayed
User Authentication	Database Setup, JWT Implementation	Mandatory	Blocks Add Point feature completely
Add Point Feature	User Authentication, Database Models	Mandatory	Core functionality unavailable
Side Panel	API Integration, Map Component	Mandatory	Reduced user experience
Color Coding	Temperature Data, Map Component	Mandatory	Visual differentiation lost
Search Autofill	Beach Data, Frontend Framework	Optional	Slightly reduced navigation efficiency

Critical Path Analysis

Identified Critical Path

> **Task 1:** Database Setup & Configuration (2 days)
> MongoDB installation, schema design, connection setup

> **Task 2:** User Authentication System (3 days)
> JWT implementation, password hashing, login/signup forms

> **Task 3:** Add Point Feature Development (3 days)
> Protected routes, form validation, data persistence

> **Task 4:** Integration & Testing (2 days)
> End-to-end testing, bug fixes, deployment preparation

Critical Path Metrics

- **Total Duration:** 10 days
- **Critical Tasks:** 4 tasks
- **Float Time:** 0 days (tasks must be completed sequentially)
- **Risk Level:** High (no buffer time for delays)

Parallel Development Opportunities

While the critical path had no flexibility, several tasks could be developed in parallel:

- Search bar development while authentication was being implemented
- Temperature toggle feature independent of database work
- UI/UX improvements while backend integration was ongoing

Risk Analysis

Risk Category	Specific Risk	Impact	Probability	Mitigation Applied	Status
Technical	MongoDB connection issues	High	Medium	Local instance	Occurred - Internal server issues experienced
Integration	Frontend-Backend communication failures	High	Medium	API testing, staged integration	Partially occurred - Late integration issues
External	OpenWater API rate limiting	Medium	Low	API key management, caching	Avoided - Proper API usage
Process	Flexible deadlines causing delays	Medium	High	Regular check-ins, peer reviews	Occurred - Uneven work completion
Team	Knowledge silos	Medium	Low	Rotating code reviews	Avoided - Effective peer reviews
Technical	Browser compatibility issues	Low	Medium	Modern browser focus, testing	Avoided - Next.js compatibility

What Went Wrong - Detailed Analysis

Major Issues Encountered

1. Database Integration Challenges

Issue: MongoDB internal server issues prevented completion of user-generated point storage.

Impact: Core functionality remained incomplete, affecting user experience.

Root Cause: Insufficient database testing and configuration management.

2. Side Panel Data Fetching Issues

Issue: Side panel was fetching data twice, causing performance problems.

Impact: Delayed additional functionality development for the side panel.

Root Cause: React component lifecycle management and state synchronization problems.

3. Process Management Inefficiencies

Issue: Flexible deadlines led to uneven work completion rates.

Impact: Some tasks completed quickly while others experienced significant delays.

Root Cause: Lack of structured milestone tracking and accountability measures.

4. Project Management Tool Underutilization

Issue: Jira not maintained regularly, causing uniform progress tracking issues.

Impact: Difficult to track actual progress vs. planned progress.

Root Cause: Team habits and insufficient integration between GitHub and Jira workflows.

Lessons Learned

Process Improvements

1. Early Integration Strategy

Lesson: Backend-frontend integration should happen early in the sprint, not at the end.

Application: Schedule integration checkpoints at week 1 of each sprint.

Expected Outcome: Fewer last-minute bugs and smoother feature development.

2. Structured Deadline Management

Lesson: Flexible deadlines without clear milestones create workflow imbalances.

Application: Implement fixed milestone dates with regular progress reviews.

Expected Outcome: More consistent work distribution and timely delivery.

3. Tool Integration and Maintenance

Lesson: Project management tools are only effective when consistently updated.

Application: Integrate Jira updates with GitHub workflow, mandate regular updates.

Expected Outcome: Better visibility into project progress and bottlenecks.

4. Database Environment Stability

Lesson: Database configuration issues can block critical path features.

Application: Implement database environment testing and backup strategies early.

Expected Outcome: Reduced risk of database-related feature delays.

Technical Improvements

1. Component State Management

Lesson: React component lifecycle and state management require careful planning.

Application: Implement state management patterns and component testing.

Expected Outcome: Fewer data fetching issues and better performance.

2. API Integration Testing

Lesson: External API dependencies should be tested thoroughly during development.

Application: Create API mock services and comprehensive integration tests.

Expected Outcome: More reliable external service integration.

Success Metrics and Achievements

Completed Features

- **Search Bar with Autofill:** ✅ Implemented with <300ms response time target met
- **User Authentication:** ✅ Secure login/signup with MongoDB integration
- **Temperature-Based Color Coding:** ✅ Visual temperature representation on map
- **Side Panel:** ✅ Dynamic beach data display with real-time updates
- **Temperature Unit Toggle:** ✅ Celsius/Fahrenheit switching functionality
- **Add Point Page:** ✅ Authenticated user interface for data submission

Process Successes

- **In-Person Scrum Meetings:** Effective team coordination and blocker resolution
- **Rotating Peer Code Reviews:** Prevented knowledge silos and improved code quality
- **GitHub Workflow:** Effective version control and collaboration

Recommendations for Future Sprints

Immediate Actions

1. **Implement Fixed Milestone System:** Replace flexible deadlines with structured checkpoints
2. **Database Environment Hardening:** Resolve MongoDB issues and implement backup strategies
3. **Integrate Jira-GitHub Workflow:** Automate project tracking updates
4. **Early Integration Schedule:** Plan backend-frontend integration for week 1 of sprints

Long-term Process Improvements

1. **Risk Management Framework:** Implement proactive risk identification and mitigation
2. **Automated Testing Pipeline:** Reduce manual testing burden and catch issues early
3. **Performance Monitoring:** Track application performance metrics continuously
4. **Documentation Standards:** Maintain comprehensive technical and process documentation

Conclusion

Sprint 2 of the GLOW project delivered significant functionality despite encountering several challenges. The critical path analysis reveals that database-dependent features created the primary bottleneck, while parallel development opportunities were well-utilized for independent features.

The team successfully implemented 6 out of 8 planned features, with the remaining 2 features blocked by database configuration issues. The lessons learned from this sprint provide valuable insights for improving project management processes and technical implementation strategies.

Key success factors included effective in-person communication and rotating code reviews, while improvement areas focus on deadline management, tool integration, and early integration strategies.

The network diagram and dependency analysis demonstrate the importance of addressing infrastructure dependencies early in the development cycle to avoid blocking critical path features.