Non-Functional Requirements Priority Summary

GLOW - Great Lakes Observations of Water Temperatures

Team Name: *Microsofties*
Course: *CSCC01 - Introduction to Software Engineering*
Sprint: *3*
Date: *July 20, 2025*

Top 3 Prioritized NFRs

1. Performance - Response Time Requirements

Category: Performance
Description: The system maintains response times of ≤2 seconds for page loads through Next.js optimization, ≤500ms for API endpoints with Express.js, and ≤1 second for map rendering using the lightweight Leaflet library.

Why This NFR Was Prioritized:
Our team prioritized performance as the top NFR because GLOW is designed for field use where users may have limited internet connectivity and patience. Fast response times are critical for:

- **Field Data Collection:** Researchers and volunteers collecting data in remote locations need immediate feedback
- **Mobile User Experience:** Mobile devices with slower processors and network connections require optimized performance
- **User Retention:** Studies show that users abandon applications with response times > 3 seconds
- **Real-time Data Visualization:** Temperature data visualization on maps must be responsive for effective analysis

2. Usability - Mobile-First Responsive Design

Category: Usability/Accessibility
Description: Mobile-first responsive design optimized for devices at 568px width and progressively adapting to smaller screens (down to 320px), with WCAG 2.1 AA compliance and touch-optimized interactions.

Why This NFR Was Prioritized:
Mobile-first design was prioritized because:

- **Primary Use Case:** Data collection happens in the field using mobile devices (smartphones, tablets)
- **Accessibility Requirements:** Academic project must meet accessibility standards for inclusive use
- **User Demographics:** Environmental researchers and citizen scientists often work in field conditions with mobile devices
- **Touch Optimization:** Minimum 44px touch targets ensure usability in challenging field conditions (cold weather, gloves)
- **Progressive Enhancement:** Ensures core functionality works on all devices, enhancing on larger screens

Security - Authentication and Data Protection

Category: Security
Description: JWT token-based authentication, HTTPS encryption, BCrypt password hashing with salt factor of 12, and comprehensive input validation through express-validator.

Why This NFR Was Prioritized:
Security was prioritized because:

- **User Trust:** Citizen science projects require trust in data handling and user privacy protection
- **Academic Standards:** University projects must demonstrate understanding of security best practices

- **Data Integrity:** Temperature data must be protected from tampering to maintain scientific validity
- **User Account Protection:** User credentials and personal information must be secure
- **API Security:** Rate limiting and input validation prevent malicious attacks that could compromise the system

Trade-offs Made to Prioritize These NFRs

1. Advanced Features vs. Core Performance

Trade-off: We chose to limit advanced analytics features (complex data visualization, predictive modeling, advanced filtering) in favor of optimizing core performance for basic map interaction and data upload.

Why We're Okay With This Trade-off:

- **MVP Approach:** For Sprint 3, establishing a solid foundation with excellent performance is more valuable than feature richness
- **User Feedback Priority:** Fast, reliable core functionality generates better user feedback than slow feature-rich applications
- **Future Scalability:** A performant foundation makes it easier to add features incrementally in future sprints
- **Development Resources:** Limited sprint time is better invested in optimization than feature expansion

Benefits Received:

- Consistently fast user experience across all devices
- Better user adoption and retention rates
- Solid foundation for future feature development
- Reduced technical debt from performance issues

2. Desktop-First UI vs. Mobile-First Design

Trade-off: We chose mobile-first responsive design over desktop-optimized interfaces, which means some desktop users may have a less information-dense experience.

Why We're Okay With This Trade-off:

- **Primary User Context:** The majority of data collection happens in field conditions using mobile devices
- **Progressive Enhancement:** Mobile-first design naturally enhances to desktop, but desktop-first often breaks on mobile
- **Accessibility Benefits:** Mobile-first design patterns inherently improve accessibility for all users
- **Future-Proofing:** Mobile usage continues to grow, making mobile-first a strategic long-term decision

Benefits Received:

- Excellent mobile user experience for field data collection
- Better accessibility compliance across all device types
- Consistent user interface behavior across device sizes
- Easier maintenance with a single responsive codebase
- Higher user satisfaction among primary user demographic (field researchers)
- Flexible layouts that adapt to screens from 568px down to smaller devices

Impact Assessment

Performance Benefits Achieved:

- Express.js backend with optimized response handling for API endpoints
- Leaflet integration providing lightweight map rendering capabilities
- Next.js frontend with optimized component rendering and code splitting
- Smooth map interactions demonstrated on various mobile devices

Usability Benefits Achieved:

- Consistent experience across device types through responsive CSS implementation
- Accessibility considerations incorporated in component design
- Touch-optimized interface elements with appropriate sizing (min. 44px)
- CSS media queries targeting 568px breakpoint with responsive adaptation to smaller screens
- Flexible UI implementation with percentage-based layouts accommodating various screen sizes

Security Benefits Achieved:

- JWT token authentication successfully implemented with proper error handling
- BCrypt password hashing with salt factor of 12 protecting user credentials
- Data transmission secured through HTTPS protocols
- Comprehensive input validation implemented across controllers using express-validator

Trade-off Validation:

The trade-offs made have resulted in a focused application that demonstrates strong core functionality. Code analysis confirms that the implementation aligns with the non-functional requirements, with particular strengths in security implementation, responsive design, and a solid performance foundation that can be built upon in future iterations.