CS Group 8 TinyBridge Process Update #2

Update

TinyBridge - A Lightweight URL Shortening Service designed for Efficiency and Branding.

1. UI design improvement and Figma prototyping —- Completed

Following the UI design challenges identified in Update #1, the team has adopted Figma as the primary prototyping platform. By referencing established design systems and open-source UI patterns, we have significantly improved interface consistency and user interaction flows. The new design better aligns with modern user-centered design principles and addresses previous usability concerns.

2. White paper completion —- Completed

The TinyBridge white paper has been finalized, providing comprehensive documentation of the project's purpose, target users, technical architecture, and cost estimation. This document serves as a reference for both internal development and external evaluation, clearly articulating the system's value proposition and implementation strategy.

3. Backend project initialization —- Completed

The backend development environment has been successfully initialized. The project structure, dependency management, and basic configurations are in place. The team has established coding standards and selected the core technology stack, laying the foundation for upcoming feature implementation.

Challenges

1. Database architecture design complexity

While the white paper outlines the overall database strategy, translating it into a practical and scalable implementation presents challenges. Key concerns include optimizing table structures for high-frequency read/write operations, designing efficient indexing strategies for short URL lookups, and ensuring data consistency across distributed components.

2. Team coordination and time management

As the project enters the intensive development phase, coordinating parallel workstreams (backend, frontend, testing) has become more demanding. Balancing progress across multiple tasks while maintaining code quality and meeting deadlines requires careful planning and communication.

Solutions

1. Adopt incremental database design with performance benchmarking

Start with a minimal viable database schema that supports core functionalities (URL shortening and redirection). Use performance profiling tools to identify bottlenecks early, then iteratively refine the design based on actual load testing results. Consider adopting proven patterns such as database sharding or caching layers if scalability issues emerge.

2. Implement structured project management practices

Establish clear task assignments, regular stand-up meetings, and milestone tracking to improve team coordination. Use project management tools (GitHub Projects) to visualize progress and dependencies. Set realistic deadlines with buffer time to accommodate unforeseen technical challenges.

Next Steps

1. **Begin core feature implementation**: Prioritize development of short link generation, user authentication, and redirection logic. Ensure each module is independently testable and well-documented.
2. **Set up testing and deployment infrastructure**: Write unit tests for critical functions, prepare deployment environments, and establish basic automated testing workflows to ensure code quality.
3. **Prepare demonstration and presentation materials**: Create demo scenarios, prepare slides, and practice presentations to effectively communicate the project's value and technical achievements during final evaluation.