LADA Sprint 3 NFR & Testing Report

Chosen NFRs

Our team chose to prioritize the following three NFRs:

1. Performance
2. Security
3. Scalability

Performance

We prioritized performance to ensure fast load times and a smooth user experience given our site has a lot going on. A responsive system is critical for user retention and engagement, especially given the interactive nature of our complex travel planning and social features. By choosing to prioritize responsiveness our users can expect near-instant feedback from LADA Land. Optimizing performance directly improves customer satisfaction and keeps up user retention.

Security

Security was chosen as a top priority to protect sensitive user data, including personal details and travel plans. Our system utilized bcrypt with multi-round salting to ensure secure accounts and we enforce HTTPS on our site to protect our users again various attacks such as man-in-the-middle attacks. In today’s landscape, ensuring data protection and privacy is essential for maintaining user trust and complying with industry standards.

Scalability

Given our sites potential for rapid growth scalability was chosen so the system can gracefully handle increases in both the number of users and data volume. We designed our architecture to scale horizontally using cloud resources. In particular we opted to host our site on a VM via the Google Compute Engine. With deployment on Google Cloud Platform, we want to ensure that our application remains responsive under peak loads and can grow alongside our user base without significant rework.

Trade-Offs Made

We decided to accept the following tradeoffs to ensure we could implement our NFRs:

1. Additional Performance Overhead

2. Architectural Complexity

Additional Performance Overhead

Implementing the aforementioned robust security measures, such as HTTPS enforcement and password hashing, introduces additional processing overhead. However, we believe the benefits of protecting user data far outweigh the minor impact on response time. The extra milliseconds of delay are acceptable compared to the risk of data breaches or compromised user trust.

Architectural Complexity

We opted for a simplified architectural design and used frameworks like Next.js for the client and Express.js for our core system to speed up development. This approach sometimes meant foregoing more granular optimizations which we could have made had we chosen a more specialized workflow. However, the trade-off allows us to iterate quickly and focus on core features, of which we had **many**. Additionally, with GCP’s auto-scaling and load balancing capabilities we have an easy way to address scalability later as the user base grows.