Course Matrix/term-group-project-c01w25-project-course-matrix

NFRs Prioritized in our Application

Performance:

Performance was a critical focus throughout the development of our application. Given that our system integrates with both a large relational database and vector database containing thousands of entries, ensuring optimal performance was essential. Without efficient performance, queries, test and core functionalities - especially those related to chatbot interactions would suffer from significant delay thus failing to tackle one of the main problems course matrix set out to resolve which was time consuming scheduling process. To address this, our team continuously optimized database queries, streamlined backend logic and leveraged caching mechanisms to maintain responsiveness and efficiency. In addition we also incorporate multiple test cases (unit and integration) and bug fixes that ensure core features work as expected.

Usability:

From the planning step, our team aimed to address a major usability issue present in the existing system: the cumbersome process of creating timetables that involved multiple websites for querying course information and adding course/personal events. To improve the user experience, we prioritized usability as a key NFR, ensuring that our application offered an intuitive and seamless interface. A user-friendly design was essential not only for practical use but also for demonstrating the value of our application to our evaluators. By making usability a primary concern, we ensured that all features were accessible, easy to navigate and effectively addressed the original pain points in timetable creation for UTSC students. Our team leveraged our experience as UTSC students to enhance the design and UI, making it more intuitive and ensuring that users could easily navigate the product with minimal instruction in various use cases.

Interoperability:

Interoperability played a crucial role in our application's architecture. Throughout development, our system integrated various third-party software solutions via well-documented REST APIs. This approach allowed us to maximize efficiency by leveraging existing technologies rather than developing redundant features from scratch. Key

integrations included OpenAI, Schedule-X, Pinecone, etc. By adopting this strategy we ensured that our application could seamlessly interact with external tools, enhancing both functionality and development speed.

Tradeoffs Made for our NFRs

Development Times:

A significant trade-off in prioritizing performance and usability was the additional development time required. Ensuring high performance demanded continuous code refinement and optimization, increasing the time spent on testing and debugging. However, we deemed this an acceptable trade-off, as poor performance would have led to longer execution times, ultimately slowing down testing and development cycles.

Similarly, our focus on usability required extensive pre-planning and interactive design improvements to ensure an intuitive user experience. While this increased development time, it also enhanced user engagement and satisfaction, making our application more effective in addressing its core objective - simplifying timetable creation. This investment in usability also improved the application's overall presentation, making features more demonstrable and earning higher evaluation scores

Flexibility:

Our commitment to interoperability came at the cost of flexibility. Relying on third-party software meant that we had to work within the constraints of those tools, limiting customization options. For instance, by integrating a third-party calendar UI component, we sacrificed control over monthly, weekly and daily view customized configuration layout. However, this trade-off was justified by the significant efficiency gains by saving time populating a dynamic calendar that users can drag and drop events. Rather than reinventing standard functionalities, we leveraged extensively tested and updated libraries, which not only accelerated development but also ensured robust and well-maintained code quality.