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

Scalability and Availability

Two key concerns we aimed to resolve in our course-matrix application were its scalability with larger data sets and its availability with increased user counts.

Scalability (how our application dealt with larger and larger data sets), was a critical concern as since the beginning of our project we worked with large datasets containing thousands of entries. Ensuring our application had good scalability was critical in further developing it.

Availability (Our deployment strategy to ensure excellent application uptimes) was a crucial concern as having our application accessible at all times was a key design feature we needed to implement.

Ensuring our Application's Scalability:

To ensure our application's scalability we heavily focused on **optimizing performance** for our application and spreading out our database over **multiple external resources**. With these two strategies we ensured that our application would function well even with large databases and a large user base.

Optimized Performance

- All user requests to the server are completed quickly. During testing we measured our various function's response times with browser dev tools. Using these metrics we worked back and optimized our code until we ensured that all requests took less than 1 second to respond, with the only exception to this being our AI chatbot. Even then we refined our AI chatbot until it took less than 5 seconds to answer any query it was given. With this we ensured our application's scalability as, by having functions that worked quickly, our application would run quickly despite larger data sets.
- All user requests to the server are managed so that no one query is too large. One thing to note about our database is that it stores thousands of courses. We ensured that when a user queried this database not all courses were given at once. Now, no matter how

large we scaled our course database, our queries would still work, ensuring scalability.

Multiple External Resources

- A key tactic in our application was spreading out databases over multiple resources, each one tailor made for their respective stored data. For instance, our user profiles and course entries were stored in supabase, allowing us to better manage large user bases efficiently. We did this with other aspects as well such as using openAI servers to host our AI chatbot. Chat logs and calendars are also stored in their own separate optimized databases to ensure scalability. In doing this we spread out our data over multiple resources each of which were picked for their ability to handle their given scaling data well.

Ensuring our Application's Availability:

One key component for our course-matrix application was ensuring that it was available to all users with as little downtime as possible. To ensure this our project leveraged **Proper Error Handling, and a CI/CD pipeline.** With these two strategies we could ensure despite a large user base our application would run smoothly and that any changes to the application would only briefly make our application unavailable. The following is a detailed explanation on how these strategies were used.

Proper Error Handling

- In our application we extensively utilize many try catch methods and other error handlers to ensure that even when improper data is passed our application will run smoothly. Due to this our application will encounter less run-time crashes and errors leading to it being more consistently available to all users.

CI/CD pipeline

- The CI/CD pipeline ensured that only the latest functional version of our website was running at all times. This increased availability as by always having the latest functional version of our website deployed it meant potential bugs that affected availability would be resolved, and that our application was in its most optimal state, improving performance and therefore availability.

- The CI/CD pipeline ensured that, no matter what, a version of our application was deployed to our virtual machine meaning that our application would be available 24/7.
- The CI/CD pipeline ensured that even if a version of our application had run into issues (e.g. a problematic uncaught bug) we could readily pull a previous functional version of our application and deploy it instead. This meant that even should errors arise our application would still be available to users.
- The CI/CD pipeline also ensured that all deployed versions of our application passed all tests, meaning only functional versions of our applications were deployed. This resolved the issue of what would happen if a non-functional version of our application was deployed unknowingly. This increased availability as it meant that our deployed application would always be functional.