What's Next: Error Logging and Monitoring

Now that your application is deployed and running in production, there's one final piece to consider: observability.

Even the best code will eventually throw errors. What matters is how quickly you know about them and how easily you can trace them.

1. Centralized Logging

By default, Spring Boot uses **Logback** for logging, and most of your logs are written to the console. That's fine for local development—but in production, you want a more structured way to:

- Store logs across deployments
- Search through errors and warnings
- Filter by log levels (INFO, WARN, ERROR, etc.)
- Track down bugs reported by users

Check out this great tutorial:
https://www.baeldung.com/spring-boot-logging

2. Error Monitoring with Sentry

Sentry is a production-grade tool for real-time error monitoring and alerting.
With Sentry, you can:

- Automatically log and track unhandled exceptions
- View full stack traces tied to user sessions
- Get email or Slack alerts when something breaks in production
- Annotate errors with custom tags, logs, or user info

It integrates well with Spring Boot using their official SDK and gives you a clear view into what's failing—and why.

Check out their Spring Boot integration guide here:
https://docs.sentry.io/platforms/java/guides/spring-boot/