Project-case CSS Music-streaming

Yardi van Nimwegen
Fontys University of applied sciences
487498@student.fontys.nl

Project Description

This project is a music streaming platform which is inspired by Spotify. The platform is meant for users to be able to:

- Stream music tracks from a music library
- Search for tracks and albums
- Create playlists
- (Optional if enough time) receive personalised recommendations

The primary focus of this project is the **back-end** architecture, to get this project to enterprise grade.

Why this project is enterprise grade software development

A Spotify clone music streaming service is a great complex software because:

- It needs to handle **large-scale traffic** (A million users streaming at the same time)
- It has **complex domain modeling** (Users, playlists, tracks)
- It possibly requires **distributed systems** and might need microservices for high availability and horizontal scaling which might be required for millions of users
- It is **data-intensive** such as the search and optimisations
- It involves **security and compliance** for user data

Non-Functional Requirements

Key non-functional requirements for this music service.

1. **Scalability** - Should be able to reach **100,000** concurrent users, the architecture should allow the application to be able to scale so that when one container cannot handle it anymore the pressure should be divided between multiple containers if required.
2. **Availability** - The service should maintain a **99,99% availability**, since it is a user-facing application which cannot function offline.
3. **Performance** - Should take less that 1 second to get a stream started.
4. **CI/CD Pipeline** - The project must have a CI/CD-pipeline that leverages automation to automate the build,test and deployment process.
5. **Automated testing** - The project must leverage automated tests to ensure that the code always works like intended.
6. **Containerization** - All components of the application should be developed as containers, This is the standard for cloud applications ensures portability to other cloud providers.
7. **Container orchestration** - This project will utilize a container orchestration tool for managing and scaling applications.
8. **Authentication & Authorization** - Implement authentication with a secure standard used in production environments such as OAuth2
9. **Data protection in transit** - All communication must be done securely with **HTTPS**
10. **Data storage diversity** - The correct type of data storage should be used for the type of data that has to be stored.
11. **Security** - Must implement precautions against the OWASP top 10 which are the 10 most common security vulnerabilities

12. **Data consistency** - User data should be consistent no matter what device the user is viewing the application on.
13. **Legal compliance** - Application should follow the GDPR, this includes the requirement for data protection, user consent and the right to be forgotten

Learning outcomes

LO3 - Scalable Architectures

I will prove this learning outcome by applying the non function requirements i have listed above

- Creating an architecture which can **scale** to a large amount of users.
- Keeping **availability** of my services high.
- Making sure the performance of my applications is at an acceptable level for users.

LO4 - Development and Operations (DevOps)

To prove this learning outcome i will apply some of the non functional requirements listed above.

- Creating an **CI/CD-pipeline** that automates building, testing and deploying which is an important part of DevOps.
- Create **automated testing** to make sure all components still work well.

LO5 - Cloud Native

To prove this learning outcome i will apply some of the non functional requirements listed above.

- All parts of the application will be developed as **containers**, the standard for cloud applications
- Use **container orchestration** for managing and scaling the application when needed.

LO6 - Security by Design

To prove this learning outcome i will apply some of the non functional requirements listed above.

- **Security** I will conduct a security risk analysis based on industry best practices which are the OWASP top 10.
- **Authentication** Implement secure authentication using a safe protocol such as OAuth2.
- **Data protection in transit** All communication between client and server must be done over HTTPS.

LO7 - Distributed Data

To prove this learning outcome i will apply some of the non functional requirements listed above.

- **Data consistency** User data must be consistent across all device.
- **Data storage diversity** Data will be stored in different solutions based on what type of data it is.
- **Legal and ethical compliance** The application will comply with the GDPR.