Software Requirements Specifications

Individual Project: FestivalConnect

Name: Lucas Jacobs
Class: S-A-RB06
PCN: 490692
Student number: 4607368
Technical teachers: Felipe Ebert, Bartosz Paszkowski
Semester coach: Gerard Elbers

Table of Contents

Glossary

Term	Definition
Response Time	The total time for the web service to respond to the sent request.
Throughput	The amount of data that the system can hold in a given period.
Data breach	An event that results in confidential, private, protected, or sensitive information being exposed to a person not authorized to access it.
DDoS	A DDoS attack will test the limits of a web server, network, and application resources by sending spikes of fake traffic.
Role-based access control (RBAC)	The idea of assigning permissions to users based on their role within the application, for instance, an admin role or festival organizer.
GDPR	General Data Protection Regulation (GDPR), is the standard protection regulation for the EU.
ISO 27001	The international standards for information security.
OS (Operating system)	A collection of software that manages computer hardware resources and provides common services for computer programs.

Introduction

Before making critical decisions, like the decision of the architecture and software frameworks, it will all depend on what the system needs to do and what kind of behavior it needs to have. Based on this we can then further analyze what is needed for FestivalConnect. This document will cover the specific characteristics FestivalConnect will need.

Functional Requirements

The functional requirements specify what the system should do, this section will specify the desired behavior and functionality of FestivalConnect. It will show all requirements the specific features and capabilities the users expect from the system. This overview will help me process to meet these expectations.

ID	Description	Priority (MoSCoW)	Remarks
FR-001	The user shall be able to log in to FestivalConnect with their email and password.	Must	
FR-002	The user shall be able to register as a festival-goer by providing basic information.	Must	
FR-003	The festival organizer shall be able to register as an organizer.	Must	
FR-004	The user shall be able to view all the festival communities that a user can join.	Must	
FR-005	The user shall be able to join communities related to their festival interests	Must	
FR-006	Users must be able to post comments in communities.	Must	
FR-007	The user shall be able to explore festival communities based on when the date, genre, location, or name.	Must	
FR-008	Users must receive notifications for new events comments and community updates.	Could	Get push notifications on the website to view and navigate to.
FR-009	The admin should be able to manage the user's information.	Must	
FR-010	The festival organizer shall be able to create communities	Must	
FR-011	The festival organizer shall be able to adjust their profile information.	Must	
FR-012			

Non-Functional Requirements

A non-functional requirement will describe how a system should behave and what a user will expect from the product. There are a lot of non-functional requirements such as Reliability, Capacity, Serviceability, etc. For my product, we will have a main focus on the following non-functionals that will be kept in mind each sprint:

- Performance
- Scalability
- Security
- Privacy/GDPR

There are more important ones stated in the ISO/IEC 2510. Let's now go over each important non-functional requirement, and list them into requirements.

Performance

This measure is important because when a system is not quickly responding a person will lose their attention. A study has found back from 1993, Jakob Nielsen came to three metrics that are how general human attention works:

- < 0.1 second: Within this time there is no evidence that the application's performance is not instantaneous.
- 1 second: The user will notice a delay.
- 10 seconds: The user will completely lose their attention.

Therefore, performance is really important and should be at all costs, when possible a response time within 1 second. Reaching the 10 seconds, there will be about 55% that leave the website after 3 seconds.
Also when loading a website, time is important because when it loads in 1 second it has three times more chance that the user will stick than when it loads in five seconds. (Nonfunctional Requirements in Software Engineering: Examples, Types, Best Practices, 2023)
Also, the throughput needs to be taken into account, the system needs to handle around one million concurrent users, and this traffic needs to be handled by the system and taken into account when thinking of performance.

Scalability

FestivalConnect needs to be designed to easily scale and to keep in mind an increasing user load. Scalability refers to the capacity that the system can handle growth in terms of data volume and user load. Since the festival period is from April to September, this is a time of peak usage of the application, therefore we need to keep to account that concurrent users can increase to five million concurrent users. Also to keep in mind is the 10-Times Rule, the same goes for time, which can be used for scalability, so we need to handle an amount of concurrent users up to 50 million. The power of ten can be considered to keep in mind the future growth and spikes of demand when big festivals are happening in a short period. (Methods and Rule-Of-Thumbs in The Determination of Minimum Sample Size When Appling Structural)
The number of how many people who can access this platform needs to be ready to expand since the festival industry is a fast-growing industry. When we look at the biggest festivals in the world, that combined is already 10 million plus people.

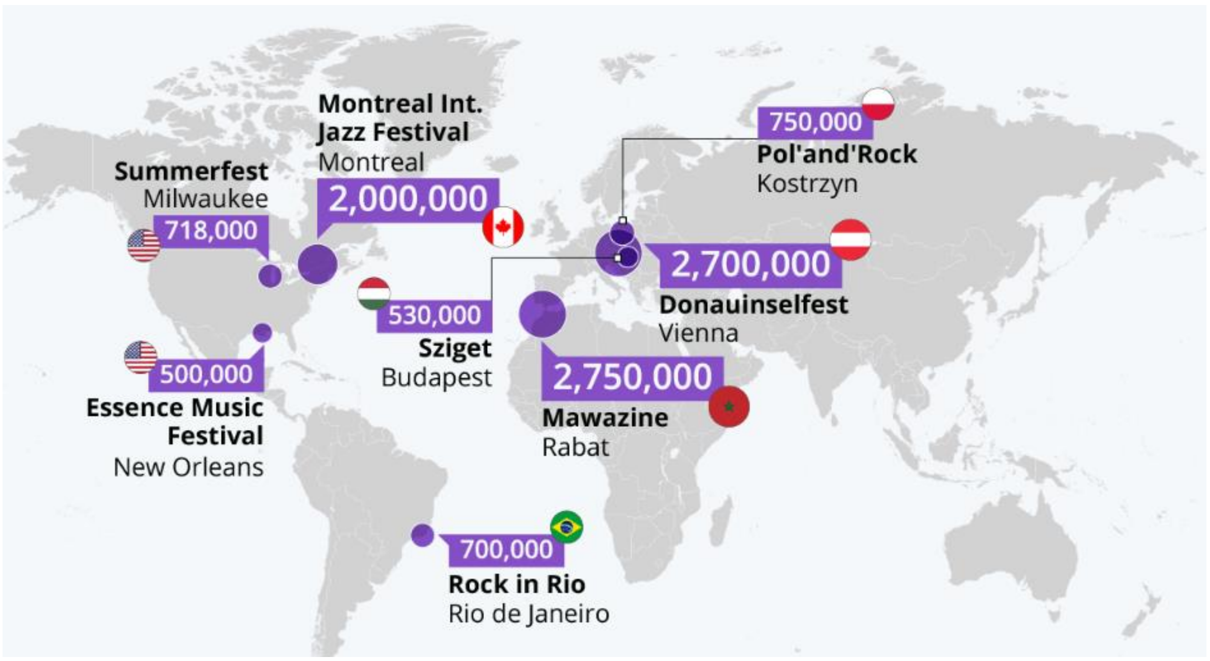

Figure 1 Most popular festivals

Also in the Netherlands, last year 2023 we had a total of 23 million total people visiting a festival. Therefore this whole business is really big and we need to always consider how to grow this platform, in terms of scalability. (Number of festival visits in the Netherlands from 2012 to 2023, n.d.)

Performance meets Scalability metrics

Furthermore, to be more specific about the performance and scalability of FestivalConnect, we can give an average response time and the requests per minute for peak usage, to get a better understanding of what metrics to work with.

Request handled by FestivalConnect

To calculate the average response time, we need a few variables. These are the following:

- n: The number of concurrent users. We will take the peak, which is fifty million
- r: The number of requests per second received by the server
- T_{think}: The average think time per request (in seconds). It is the time between the actions that someone is taking. For example Login > **Think Time** > Search > **Think Time** > Logout. For a normal user, this takes around three to five seconds on average. Therefore for our calculations, this will be 3 seconds. Make sure this is not too big or too small because otherwise, this can impact the server's performance. (perfmatrix, 2019)

To determine the response for peak load, you can use the following formula.

$$T_{\text{response}} = \frac{n}{r} - T_{\text{think}}$$

Now since we want T_{response} to be within one second (see NFR-01), we can calculate the r that needs to be handled by the system. See the following calculation.

$r = n/(T_{\text{response}} + T_{\text{think}})$
$r = 50.000.000/(1 + 3)$
$r = 50.000.000/(4)$
$r = 12.500.000$

So to conclude the number of requests that FestivalConnect needs to handle is aimed at 12.5 million requests a second. Based on this we can make technology decisions that can handle this much without performance degradation. (Establishing Performance Goals)

Tackling Performance

From the conducted research (Jacobs, 2024, Architecture Research), FestivalConnect will use microservices as architecture. In the previous section, we discussed the overall handled load distributed over the system. Now we want to further look at what each service needs to handle, to give a rough sense of how much size of input and outputs, the amount of data to store, and the time available to do calculations, convert data, etc.

When looking at performance, the most important things from a system are:

- **Response time**: People do not want to wait as long. Since the industry is rapidly evolving the standard of having past responses will also go up. People expect that a system is fast.
- **Best Throughput you can get**: Need to look at optimizing how much load can hold the system so that a lot of people can run the tasks at the same time.

These two things are contradictory to each other. With the first goal you want to minimize the load on the system; while the second goal requires you to maximize it. These for you need to find a between-at-level is the optimal load for the system (utilization). This is called the knee, the point to find the maximized with minimal negative impact on response times. (Millsap, 2010)

Another thing to consider is Amdahl's law, which says that performance gains are limited to the proportion of time spent on the improved component. Focus on optimizing high-impact areas first while looking at the costs of the production to fix the issue. Efficiently fix critical business functions before addressing the widespread issues. (Millsap, 2010)

Now that we know more about the importance of performance, let's go over each microservice and estimate the load on them.

- **Authentication Microservice**
  - **Size of Inputs and Outputs:** Inputs are user credentials (email and password) during the login and registration process. Outputs can be authentication tokens (JWT) or session identifiers. It is a small request, around a few kilobytes for user credentials and authentication tokens.
  - **Amount of Data to Store:** The user credentials will be stored securely. This includes hashing and salting the passwords and user data, and storing the user credentials can require a moderate amount of storage space per user, keeping it simple and say 1 MB per user. This means a total of 50 TB is needed for storage.
  - **Time Available for Calculations:** Authentication needs to be performed in an efficient way to minimize the login time, this needs to happen quickly and within milliseconds.
  - **Consequences:** Delays in the authentication process lead to poor user experience and impact potential scalability if authentication becomes a bottleneck.
- **User Management Microservice**
  - **Size of Inputs and Outputs:** User management may involve moderate-sized requests for user data, we assume a request of around 10 KB per request, and the return message is also around 10 KB in size.
  - **Amount of Data to Store:** Needs to store user profiles, which can be around the same size as the authentication microservice.
  - **Time Available for Calculations:** Retrieving and sending data and validating this, needs to happen within milliseconds.
  - **Consequences:** Slow registration or updating can lead to frustration among users.
- **Community Management Microservice**
  - **Size of Inputs and Outputs:** The size of community data (creating, updating) can be around 20 KB per request. With the return information and confirmation messages, can also be around 20 KB in size.
  - **Amount of Data to Store:** A community can vary in ranges from 100 to more than a million. Need to store community data, including member lists, which can increase in

size very fast. So when we take a Metadata Storage of around 2 KB and store a member that has its ID or Username assigned to a list will be 20 bytes. So total space per community can vary from around 4KB to 100 MB (100 vs 5 million users)
    - **Time Available for Calculations:** This can be handled fairly quickly, looking at within milliseconds.
    - **Consequences:** Slow leading community operations can lead to dissatisfaction among the festival organizers, leading potentially to less usage of the application.
- **Post Management Microservice**
    - **Size of Inputs and Outputs:** Posts can vary (text, images, videos, etc), around an average of 1MB per post. So the output will also be around 1MB.
    - **Amount of Data to Store:** So this also comes down to 1MB per post, which means the amount of data to store needs to be constantly scaled, assuming one person creates one post on average, the total will be 50 million MB, or 50 terabytes (TB).
    - **Time Available for Calculations:** It should be handled as efficiently as possible, keeping in mind big videos have a longer time to post and retrieve.
    - **Consequences:** Slow post management will be really bad for the user experience, especially for users who are trying to interact with posts or creating one, this leads to frustration and decreased engagement among users.
- **Notification Microservice**
    - **Size of Inputs and Outputs:** Notification can be content such as text or a link, this can be an average of around 10 KB per notification. Output is the same, also 10 KB.
    - **Amount of Data to Store:** Notifications can be deleted over time, but people can receive them multiple once. Assuming a notification is 10KB, and a person gets on average 5 notifications, this means we need around 250 TB to store.
    - **Time Available for Calculations:** A notification processing and delivering can be quick, around milliseconds.
    - **Consequences:** Delays can lead to users missing important updates, which potentially leads to dissatisfaction.

Keep in mind that there can always be additional microservices or features that are required later on. For example, having a dedicated microservice for managing the GDPR or implementing a two-factor authentication could be beneficial.
What is also is important that a single request will never exceed the processing time for the whole system. Nevertheless, with these estimates, we now have a clear view of what the most carrying microservices are in regards to handling data and storing them as well. But also take into account that these are just estimates and accurate estimations can be performed with testing.

Security and Privacy

Security is also a big thing for FestivalConnect, it needs to prioritize security to protect the data of users and to guarantee the integrity and trust of the system. Therefore the system should be designed against specific threads, such as unauthorized access, data breaches, or DDoS attacks.
Moreover, it will be important to implement an authentication and authorization setup such as role-based access, to control access for instance sensitive data. Additionally, also looks at industry-specific security standards and regulations to keep in mind, such as GDPR, and ISO 27001, which can create benefits to show security. It is not only beneficial but also mandatory since when you violate the rules of GDPR, FestivalConnect can receive a fine that can be as high as 20 million euros or 4% of the revenue that FestivalConnect earned in a year. By using all these principles, we can make sure that FestivalConnect can make a secure application that protects user information and also protects and correctly handles the data of users.
(Nonfunctional Requirements in Software Engineering: Examples, Types, Best Practices, 2023)
Moreover, the FestivalConnect application wants to link with festival organizers, therefore it is also good to look into festival-specific regulations on how to handle their data, and what the festival organizers can use and see. Think about this Festivals can share the data with partners, sponsors, vendors, etc.
Furthermore, to aim further attention to security, OWASP's top 10 is an effective way to keep notice of the risks so that FestivalConnect can focus on which risks are important which will lead to helping them understand, identify, and fix the vulnerabilities in the application. (Mylonas, 2020)

Other Non-Functional Requirements

There are also a lot more non-functional requirements to keep in mind when developing FestivalConnect. See the following that are also important:

- Reliability: The website needs to be consistent and up all the time. The industry standard aims for the five nines (99.999%) availability rate, but most will focus on 99.99% uptime, as will FestivalConnect because of possible downtime for maintenance and overseen issues. (A., 2023)
- Maintainability: The system needs to be easy to maintain. Technologies can be used that will fit the needs of FestivalConnect. So looking at best practices to have good version control, a proper CI/CD, and well-documented code and architecture, by for example setting coding guidelines, or explaining design structures in the system. This is one of the most important ones since we want to have the aim of building a future-oriented application that is easy to maintain by other people as well. Also, it comes with looking crucial at scalability, modularity, and extensibility.
    - Extensibility: The system needs to be built in a way that it can add more features, that can benefit FestivalConnect financially. Think of Ticket systems integration inside the app, Premium features, etc. Keeping this in mind, this will also improve reliability and consistency (Modularity). Also, we will have great pluggability, since we can easily add other components for new features. (10 nonfunctional requirements to consider in your enterprise architecture, 2022)
    - Agility: This project as stated in (Jacobs, 2024, Project Plan) uses an agile way of working. Therefore it is necessary to have a maintainable application so that we easily apply changes and improve the system. This point will also look at the deployability and reconfigurability. (10 nonfunctional requirements to consider in your enterprise architecture, 2022)
- Internationalization: Support of multiple languages is a must. It is becoming increasingly normal for people to travel all over the world to visit there most favorite festivals.
- Usability: The system needs to have an easy-to-use interface. This also needs to be friendly regards smaller devices, for future release on the app store.
- Compatibility: Making sure the website functions are correct across different web browsers and versions.

(Nonfunctional Requirements in Software Engineering: Examples, Types, Best Practices, 2023)

Non-Functional Requirements of FestivalConnect

ID	Description	Priority (MoSCoW)	Remarks
NFR-001	The system should be able to respond within one second when users are having interactions such as browsing events or posting messages in a community.	Must	
NFR-002	The system should be able to handle one million concurrent users, without losing any significant performance loss.	Must	
NFR-003	During peak festival periods, the system should scale up to five million concurrent users, making sure to still have smooth browsing.	Must	Keep in mind the 10-Times rule that can also be applied here, so aim for 50 million.
NFR-004	The system implementation will meet the standard requirements that the FestivalConnect environment has described beforehand.	Must	Think of the rules that are set when implementing requirements.
NFR-005	The system should be able to correctly handle errors and with care so that the system will not crash.	Must	Keep in mind that it needs to prevent data leaks, uptime risks, etc.
NFR-006	The system should be able to comply with the industry standards to secure user data and privacy.	Must	Think of sensitive data protected with security standards such as GDPR and ISO 27001.
NFR-007	The system should maintain an uptime of at least 99.99%.	Must	
NFR-008	The system should have authentication and authorization mechanisms to protect user data and secure access to features	Must	
NFR-009	The system should be able to display and support multiple languages.	Must	This is used for event descriptions, user profiles, and notifications.
NFR-010	The system should be able to handle errors within a certain sprint.	Must	Good Git for version control, CI/CD pipelines for automated testing and deployment.
NFR-011	The system should make sure to have a user-friendly interface for easy navigation.	Must	This while keeping an eye on further expansion on mobile, so also friendly design for this.
NFR-012	The system should implement two-factor authentication (2FA) for improved security for sensitive features or data	Could	
NFR-013	The system should be able to run on different OS and must maintain the same behavior and performance.	Must	Operating on Windows, Mac, Linux
NFR-014	The system should be able to run on different browsers while	Must	Think of Mozilla Firefox, Google Chrome, etc.

	maintaining the behaviors and performance		
NFR-015	The system should implement automated code quality checks as part of the CI/CD pipeline.	Must	Quickly identifying potential issues early on and keeping track of the code quality standards.
NFR-016	The system should use control systems effectively, with proper branching strategies and commit practices.	Must	Keeping track of changes in a maintainable way, easier for an agile method to track changes for certain sprints.
NFR-017	The system should ensure the data integrity of stored information through mechanisms while having checks to prevent data corruption and unauthorized modifications	Must	Making sure the accuracy, consistency, and reliability.
NFR-018	The system's most important features should have minimal navigation.	Must	
NFR-019	The system should respond within 5 seconds when performing high-intensive tasks.		Keep in mind that users can loose patients, try to limit it as much. Uploading a video, advanced filtering, etc.
NFR-020	The system should store data securely and seperated.	Must	

References

10 nonfunctional requirements to consider in your enterprise architecture. (2022, 08 04). Retrieved from redhat: https://www.redhat.com/architect/nonfunctional-requirements-architecture

A., J. (2023, 11 29). *What Is Website Uptime and How to Monitor Website Availability*. Retrieved from hostinger: https://www.hostinger.com/tutorials/how-to-monitor-uptime-and-downtime#:~:text=The%20industry%20standard%20for%20uptime,for%20maintenance%20and%20unforeseen%20issues.

Establishing Performance Goals. (n.d.). Retrieved from oracle: https://docs.oracle.com/cd/E19159-01/819-3680/6n5sqn12j/index.html#abfcc

(n.d.). *Methods and Rule-Of-Thumbs in The Determination of Minimum Sample Size When Appling Structural* . rajpub.

Millsap, C. (2010, 09 01). *Thinking Clearly about Performance*. Retrieved from queue: https://queue.acm.org/detail.cfm?id=1854041

Nonfunctional Requirements in Software Engineering: Examples, Types, Best Practices. (2023 , 12 30). Retrieved from altexsoft: https://www.altexsoft.com/blog/non-functional-requirements/

Number of festival visits in the Netherlands from 2012 to 2023. (n.d.). Retrieved from statista: https://www.statista.com/statistics/673461/number-of-festival-visits-in-the-netherlands/

perfmatrix. (2019, 01 17). *Think Time*. Retrieved from perfmatrix: https://www.perfmatrix.com/think-time/

Jacobs, L. (2024). Project Plan (Unpublished manuscript), FontysICT.
Jacobs, L. (2024). Research Architecture (Unpublished manuscript), FontysICT.