ANALYSIS – MICFLIX

FONTYS SEMESTER 6 SOFTWARE PROJECT

Data free icon: Flaticon.com

Date	**04-05-2022**
Version	**0.6**
State	**In progress**
Author	**Mickey Krekels**
Class	**RB04**

VERSION HISTORY

Version	Date	Author(s)	Changes	State
0.1	04-03-2022	Mickey Krekels	Added the main structure of the document.	In progress
0.2	07-04-2022	Mickey Krekels	Added the Deliverables & Non-deliverables chapter	In progress
0.3	08-03-2022	Mickey Krekels	Finished the Deliverables & Non-deliverables chapter	In progress
0.4	08-03-2022	Mickey Krekels	Finished the chapter use cases	In progress
0.5	25-04-2022	Mickey Krekels	Added the chapter security by design	In progress
0.6	04-05-2022	Mickey Krekels	Added the chapter GDPR and data complexities	In progress

PREFACE

This project is made for the Fontys semester-6 personal assignment. The goal is to make a streaming service clone with a strong inspiration from the platform Netflix. This clone will be called Micflix and the final version of the project will be of enterprise quality.

This analysis document serves as the main requirements description of the project. Each of the functional requirements will be coupled to a use case, where the overall steps and problems will be described. The non-functional requirements will be categorized based on their type and for each, there will be a requirement added. At the end of the analysis, there will be a final conclusion.

TABLE OF CONTENT

1 REQUIREMENTS

1.1 DELIVERABLES & NON-DELIVERABLES

These tables show all the deliverables & non-deliverables of this project, all the requirements will be prioritized using the MoSCoW technique (1).

Index	Functional requirements	Must	Should	Could	Won't
FR-1	User is able to register an account	X			
FR-2	User is able to login to his account	X			
FR-3	User can update their user data			X	
FR-4	User can view a selection of movies based on the date added		X		
FR-5	User can view a selection of movies based on last watched	X			
FR-6	User can pause the selected movie		X		
FR-7	User can change the resolution of the selected movie			X	
FR-8	User can download a selected movie				X
FR-9	User gets a notification on their application when a new movie is available.			X	
FR-10	User gets a selection of movie recommendations based on previously watched movies			X	
FR-11	User can filter movies based on name	X			
FR-12	User can filter movies based on genre	X			

To make this project GDPR compliant, the folwing non-fuctionals are listed in the table below.

Index	Non-Functional requirements	Must	Should	Could	Won't
NFR-1	Performance		X		
NFR-2	Scalability	X			
NFR-3	Reusability				X
NFR-4	Maintainability	X			
NFR-5	Reliability	X			
NFR-6	Security	X			
NFR-7	Response time			X	
NFR-8	Availability		X		

1.2 REQUIREMENT USECASES

1.2.1 Login User

Use Case Name:	Login user
Index:	FR-1
Summary:	To view the available videos on the Micflix streaming platform, a user **must** be able to login into the system.
Basic Flow:	1. The user wants to login 2. The system requests the username and password 3. The User provides the username and password 4. The system compares the username and password with the hashed and secured data in the database. 5. The system starts a login session and provided the user with a secured login token
Alternative Flows:	1. if the username is invalid, the user goes back to step 2. 2. if the password is invalid, the user goes back to step 2.
Preconditions:	The user is registered.
Postconditions:	The user is login in on the platform

1.2.2 Register User

Use Case Name:	Register user
Index:	FR-2
Summary:	To login onto the Micflix streaming platform, a user **must** be able to register on the system.
Basic Flow:	1. The user wants to register. 2. The system requests a username, password and mail address. 3. The user enters a username and password. 4. The system checks that the username does not duplicate any existing registered usernames. 5. The system reroutes the user to the login screen
Alternative Flows:	1. If the username duplicates an existing username the system displays a message and the user goes back to step 2.
Preconditions:	None
Postconditions:	The user is now registered on the streaming platform.

1.2.3 Update User data

Use Case Name:	User can update their user data
Index:	FR-3
Summary:	A user wants to update their personal user data(username, password, email)
Basic Flow:	1. User wants to update their account data 2. The system sends a randomized code to the user 3. User copies the randomized code from their mail 4. User enters the randomized code into the system 5. System validates code 6. System sends user to account update page 7. User changes account data 8. System updates users account data
Alternative Flows:	1. If the randomized code is not equal to the one from the mail, user goes back to step 1.
Preconditions:	The user is registered.
Postconditions:	The user account data is updated

1.2.4 Selection of movies based on the date added

Use Case Name:	Selection of movies based on the date added
Index:	FR-4
Summary:	A user must be able to see a selection based on newly added movies
Basic Flow:	1. User wants to see a selection of newly added movies 2. User selects newly added 3. System provides all recently added movies
Alternative Flows:	None
Preconditions:	The user is registered.
Postconditions:	User gets a selection of recently added movies

1.2.5 Selection of movies based on last watched

Use Case Name:	Selection of movies based on last watched
Index:	FR-5
Summary:	A user must be able to continue their previously watched movie, therefor a selection of previously watched movies must be visible for the user.
Basic Flow:	1. User wants to see a selection of previously watched movies 2. User selects previously watched 3. System provides all previously watched movies
Alternative Flows:	1. If there is no record of previously watched movies, user gets rerouted to the main page.
Preconditions:	The user is registered. User must have a record of previously watched movies
Postconditions:	User gets a selection of previously watched movies

1.2.6 Pause video

Use Case Name:	Pause video
Index:	FR-6
Summary:	A user must be able to pause their selected movie
Basic Flow:	1. User wants to pause their selected movie 2. User clicks on the pause button 3. System saves current timestamp of the movie duration
Alternative Flows:	None
Preconditions:	The user is registered. Movie must be selected
Postconditions:	The selected video is paused

1.2.7 Select resolution

Use Case Name:	Select resolution Scenario: user selects the required resolution of the video.
Index:	FR-7
Summary:	A user must be able to change the resolution selected movie
Basic Flow:	1. User wants to change the resolution of the video 2. User clicks the resolution dropdown menu 3. User selects their preferred resolution 4. System saves current timestamp of the movie duration 5. System retrieves the data selected resolution version of the movie 1. System reloads movie from the point of the timestamp
Alternative Flows:	None
Preconditions:	The user is registered. Movie must be selected
Postconditions:	The resolution is changed and the movie is continued from the timestamp

1.2.8 Download a selected movie

Use Case Name:	Download a selected movie
Index:	FR-8
Summary:	A user wants to download a movie for watching offline.
Basic Flow:	1. User wants to download the selected movie 2. User clicks the download button 3. System download the selected movie on the selected resolution 4. System adds the download movies to the downloaded selection
Alternative Flows:	1. If the internet is not available, the System cancels the download and the user is redirected to step 1 with a message no internet available.
Preconditions:	The user is registered. The user must have internet access
Postconditions:	The movie is downloaded on their local device

1.2.9 Notification when a new movie is available.

Use Case Name:	Notification when a new movie is available.
Index:	FR-9
Summary:	When a new movie is available, the system lets the user know with a notification.
Basic Flow:	1. System sends notification that there is a new movie available 2. User clicks on the notification 3. System directs the user to newly added movie
Alternative Flows:	None
Preconditions:	The user is registered.
Postconditions:	User is informed or redirected about the newly added movie

1.2.10 Movie recommendations based on previously watched movies

Use Case Name:	Movie recommendations based on previously watched movies
Index:	FR-10
Summary:	When a user has watched movies with a type of genre, a selection Is made available with recommendations for movies with that genre type.
Basic Flow:	1. User views a set of movies with a similar genre 2. System provides a set of recommendations in a selection view
Alternative Flows:	None
Preconditions:	The user is registered. The user viewed a set of movies with a similar genre
Postconditions:	User has a selection view added with recommendations

1.2.11 Filter movies based on genre

Use Case Name:	Filter movies based on genre
Index:	FR-11
Summary:	A user must be able to filter movies based on genre types
Basic Flow:	1. User wants to see a selection of movies with a genre type 2. User types the genre in the search bar 2. System provides all movies with the genre type
Alternative Flows:	1. If the genre is not recognized by the system, user gets redirected to step 2 2. If the user enters nothing and hits enter the system redirected to step 2
Preconditions:	The user is registered.
Postconditions:	User is shown a selection of movies with the selected genre type

1.2.12 Filter movies based on name

Use Case Name:	Filter movies based on name
Index:	FR-12
Summary:	A user must be able to filter movies based on name
Basic Flow:	1. User wants to see a selection of movies with a specific name 2. User types the name in the search bar 3. System provides all movies with the specific name
Alternative Flows:	1. If the name is not recognized by the system, user gets redirected to step 2 2. If the user enters nothing and hits enter the system redirected to step 2
Preconditions:	The user is registered.
Postconditions:	User is shown a selection of movies with the specific name

2 SECURITY BY DESIGN

The phrase "Security-by-Design" refers to an approach in which security is a priority across the whole software development lifecycle. The security-by-design technique suggests that security-related activities be considered at every stage of software development.

2.1 OWASP

The OWASP Foundation is a non-profit organization dedicated to improving software security. The Foundation is the source for developers and technologists to safeguard the web through community based open-source software projects.

In this chapter of the document, the OWASP top 10 security issues are going to be explored for potential pitfalls within the Micflix project. The context for each of these risks will be explained with the information provided by the “OWASP Top Ten” webpage (2) .

2.1.1 Broken Access Control

Access control enforces policy such that users cannot act outside of their intended permissions. Failures frequently result in unauthorized information disclosure, alteration, or loss of all data, as well as the execution of a business function outside of the user's capabilities.

2.1.1.1 Is the project vulnerable?

The project is currently vulnerable on an **Average** level. Security options such as Cross-Origin Resource Sharing (CORS) and Keycloak as a session identifier are already implemented. This is thus far a big improvement on the previous vulnerability of the project, but rate-limiting APIs, log access control failures and disabling web server directory listing are still improvements that can be made.

2.1.2 Cryptographic Failures

The use of a hard-coded password, Broken Crypto Algorithms, and Insufficient Entropy can lead to sensitive data exposure. This means that passwords, credit card numbers, health records, personal information, and business secrets, for example, require extra security, particularly if the data is subject to privacy laws, such as the EU's GDPR.

2.1.2.1 Is the project vulnerable?

No, for this part of the security Keycloak is used to authenticate, and it generated keys cryptographically randomly. All of the services within the Micflix project use this functionality to only send data when a user is correctly authenticated. Therefore the security measurement of cryptographic failures is on a **Good** level.

2.1.3 Injection

The security problem injection occurs when the following issues are present. Namely, the user-supplied data is not validated or sanitized, dynamic queries or non-parameterized calls without context awareness, destructive data is used within object-relational mapping search parameters and malicious data is directly used or concatenated by the application.

2.1.3.1 Is the project vulnerable?

No, for the issue injection the security grading is on an **Excellent** level. The databases for the services of this project are of the type MongoDB, which means that the query is a NoSQL language. Therefore SQL injections are not likely to be more difficult to happen. The Golang API also uses a parameterized interface and object-relational mapping which makes it extremely difficult to inject SQL queries.

2.1.4 Insecure Design

Secure design is a culture and approach that examines risks regularly and guarantees that code is well-designed and tested to avoid known attack methods.

2.1.4.1 Is the project vulnerable?

Yes, currently the code/system testing in the Micflix project is minimal, therefore this does not guarantee that the critical flows are resistant to the threat model. There is also no limit to the resource consumption for the services, these issues could bring security problems in the future and therefore need to be addressed as soon as possible. In conclusion, the security grading of insecure design is at a **Poor** level.

2.1.5 Security Misconfiguration

Instead of faulty code, the term "security misconfiguration" is a catchall word that incorporates the most common vulnerabilities created by the application's configuration settings. These wrongly set config options can lead to security risks if these are not addressed quickly.

2.1.5.1 Is the project vulnerable?

As a CI/CD pipeline, all of the most important config settings are stored as secrets. This ensures that the configuration of these services are not mistakenly altered. The second important issue is error handling, but because nature of Golang the change of sending overly informative error messages to the user is minimal. Because of these steps, the current security grading on security misconfiguration is set at a **Good** level.

2.1.6 Vulnerable and Outdated Components

Every day, new cyber vulnerabilities and threats emerge, putting users at risk. The majority of these vulnerabilities are caused by software dependencies, such as the utilization of libraries and frameworks that are susceptible to not updated software patches.

2.1.6.1 Is the project vulnerable?

With the use of docker files, all of these software patches are up to date. But this could also be a problem when certain libraries are not working anymore, luckily with the additions of the CI/CD pipeline this problem can be quickly spotted and adjusted. Therefore the current security grading on Vulnerable and Outdated Componentsis set at a **Good** level.

2.1.7 Identification and Authentication Failures

The ability to uniquely identify a user of a system or an application executing on the system is defined as identification. Authentication refers to the capacity to establish that a user or program is who they claim to be. This security issue specifies the possible failures in this field.

2.1.7.1 Is the project vulnerable?

No, the prevention against these issues are built into Keycloak. For example, implementing weak password checks, random session IDs and limiting failed login attempts are features that can be enabled within the Keycloak service. In conclusion, the security grading of insecure design is at a **Good** level.

2.1.8 Software and Data Integrity Failures

Code and infrastructure that do not defend against integrity violations are referred to as software and data integrity failures. A program that uses plugins, libraries, or modules from untrusted sources, repositories, or content delivery networks is an example of this (CDNs).

2.1.8.1 Is the project vulnerable?

No, the CI/CD pipeline has a proper configuration and access control. This ensures the integrity of the code flowing through the build and deploys processes. But there can be some improvements made, such as adding the OWASP Dependency-Check to the project. Because of this, the current vulnerability is graded on the **Average** level.

2.1.9 Security Logging and Monitoring Failures

The Logging and Monitoring of projects help to assist in the detection, escalation, and response to active breaches. Breach detection is not possible without logging and monitoring.

2.1.9.1 Is the project vulnerable?

Yes, currently the only logging that is used is with the external application called Grafana. This helps to spot containers that are using up too much CPU or memory. The addition of more specialized tools for logging is certainly needed. In conclusion, the security grading of security logging and monitoring failures is graded at the **Poor** level.

2.1.10 Server-Side Request Forgery

This issue can occur when a web application is fetching a remote asset without confirming the sender URL. This allows an attacker to force the application to send a fabricated request to an unforeseen destination, even when a firewall is used.

2.1.10.1 Is the project vulnerable?

No, the docker version of the project already used Ngnix proxy manager. The option for disabling HTTP redirections already was an option with this service. It also Enforced “deny by default” firewall policies to block all but the intended traffic. Because of these steps, the current security grading on server-side request forgery is set at a **Good** level.

Owasp vulnerability table

In this table, the project's overall OWAP vulnerability risks will be categorized by the following labels: **Excellent security**, **Good security**, **Average security**, **Poor security**, and **Bad security**.

Index	OWASP RISKS	PROJECT VULNERABILITIES
2.1.1	Broken Access Control	Average security
2.1.2	Cryptographic Failures	Good security
2.1.3	Injection	Excellent security
2.1.4	Insecure Design	Poor security
2.1.5	Security Misconfiguration	Good security
2.1.6	Vulnerable and Outdated Components	Good security
2.1.7	Identification and Authentication Failures	Good security
2.1.8	Software and Data Integrity Failures	Average security
2.1.9	Security Logging and Monitoring Failures	Poor security
2.1.10	Server-Side Request Forgery	Good security

3 GDPR AND DATA COMPLEXITIES

3.1 GDPR

The GDPR (3) is an abbreviation for General Data Protection Regulation, it is a European Union legislation that governs how corporations and organizations gather and use personal data. It was enacted by the European Parliament and the Council of the European Union.

For this semester the minimum requirements for making your own project GDPR-proof are the following essentials (This information is from the FHICT Canvas course (4)). Therefore I will answer each of them in the following subchapters.

3.1.1 Is the personal data identified and processed in a fair way?

The only data that is saved besides the registered information, is the user profiles with assigned tags. These tags are only used for generating recommendations based on the genre of movies previously watched.

3.1.2 Is the personal data well managed and protected from data breaches?

Yes, the data is safeguarded behind a Keycloak service. Where data is hashed and only reachable with the right authorization Token.

3.1.3 Can the personal data be updated but also deleted from your system?

Yes, the user has the option to change their personal data on the home screen of the Micflix application. This Includes changing email, password and even their 5 different profile names.

3.1.4 Is the personal data saved such that it is clear where it is stored?

Yes, the data Important data such as passwords and email addresses are stored within the database behind the Keycloak-service. The profile data with the saved tags (For movie recommendations) are stored in the User-service database (this data is only linked with one GUID for privacy protection in case of a breach).

3.2 CAP THEOREM

The three letters in CAP stand for three desired features of distributed systems with replicated data: consistency among replicated copies, system availability for reading and write operations, and partition tolerance.

The CAP theorem implies that in a distributed system with data replication, all three desirable features: consistency, availability, and partition tolerance cannot be guaranteed at the same time (see Figure 1 for a visual overview).

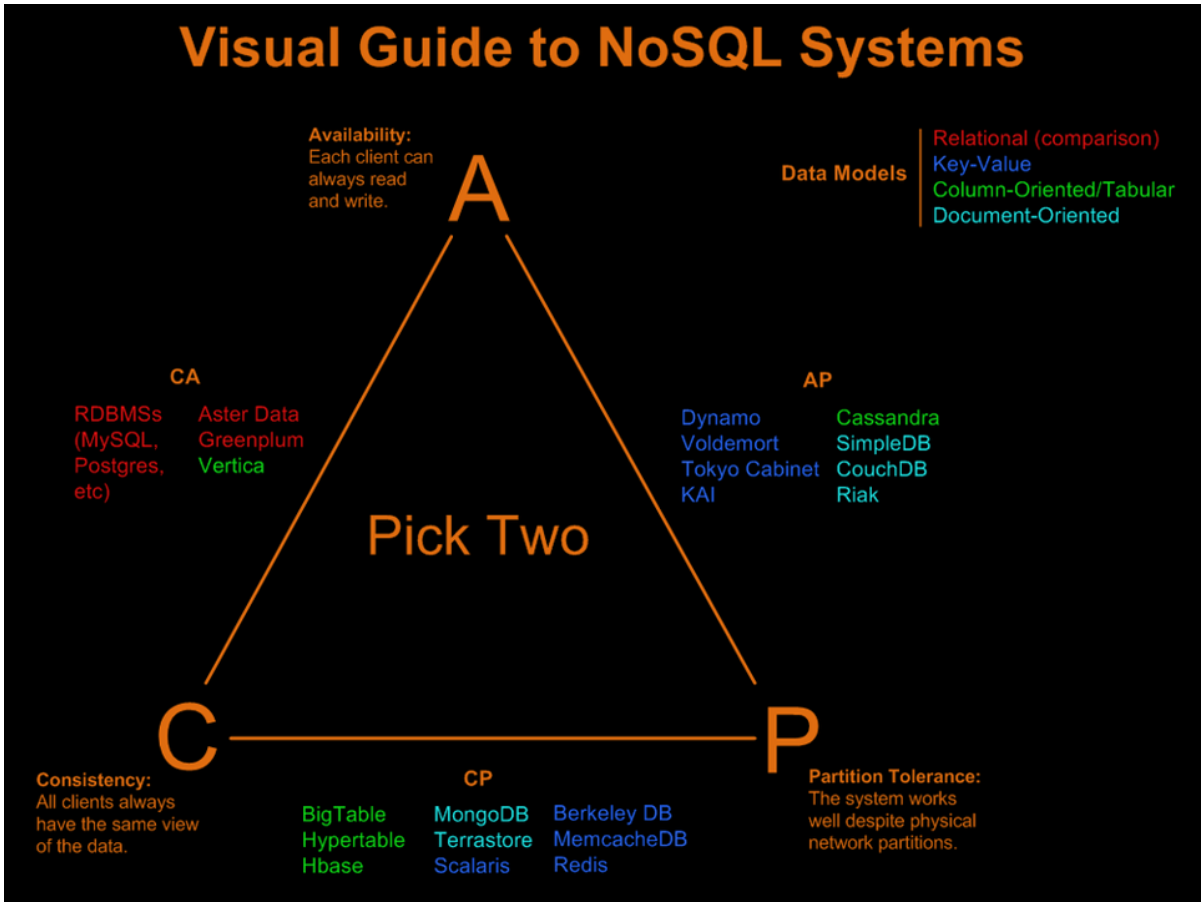

Figure 1 Fontys S-ESE6-CMK Module GDPR and Data Complexities- Theoretical Background on Canvas

3.2.1 Matching Data Store

In this chapter, the type of matching Data Store is going to be chosen based on the steps provided by the page “GDPR and Data Complexities- How to Apply?” (5) on Canvas.

3.2.1.1 Describe all the properties of and requirements for persistent data in your system

The Micflix project is essentially a clone of the popular streaming platform Netflix, which means that a lot of movie data is persistently stored within the database structure. An easily scalable database is a must-have for this project. A document-oriented database is perfect for this task, the first one that comes to mind is MongoDB.The article Database Scaling (6) states that *MongoDB is built from the ground up to scale massively and has high availability*, therefore this would be a great choice for a database.

3.2.1.2 Identify privacy-sensitive data in your system and plan functionalities to comply with GDPR

Enable to use the Micflix streaming platform, a user must be registered in the system. This means that the password and personal information must be guarded for possible data leaks. The system also keeps track of the genre types that the profile of the person watches a lot. This is only to recommend movies that the user may want to watch based on their view history.

So to summarize the privacy-sensitive data includes:

1. User login information (Mail, Username, Password)
2. User genre viewed history (Genre, Watched-count, Timestamp)

3.2.1.3 Find patterns to improve your architecture with distributed data without losing scalability.

From the information provided by the CAP theorem table (see figure 1), the data requirements are leaning more toward CP (consistency and partition tolerance).

The feature **consistency** states that the user always has the same view of the data. In project Micflix this is also the case, the only thing that changes to the view is when a new movie is added to the system.

The feature **partition tolerance** states that the system works well physical despite network partitions. On the program, Netflix streaming packages are often delayed when there are network problems. It compensates for this problem by switching the resolution data from an available node. Hence since this project is a clone, the system should be able to operate in the saim mainer when database nodes are delayed or not available.

3.2.1.4 Use the CAP theorem to evaluate candidate data stores against your data requirements

In 3.2.1.3 we found out that the pattern CP was most fitting for the project. In the table below the following databases will be listed with their pros and cons assigned.

Database types	Data models	Pros	Cons
MongoDB	**Document-Oriented**	**Scalable, High speed, flexible**	**No join, High memory**
Terrastore	**Document-Oriented**	**Scalable, flexible**	**Limited documentation**
Scalaris	**Key-Value**	**Scalable**	**Limited documentation**
Berkeley DB	**Key-Value**	**Scalable, High-performance**	**Difficult Implementation**
MemcacheDB	**Key-Value**	**High-performance, Stable**	**No large object support (>1MB)**
Redis	**Key-Value**	**Cache speed**	**Single-threaded, Scaling**
BigTable	**Column-Oriented**	**Scalable, Data durability**	**Compatibility**
Hypertable	**Column-Oriented**	**High Performance**	**Limited documentation**
Hbase	**Column-Oriented**	**Scalable, Large data**	**Latencies, authentication**

4 CONCLUSION

5 BIBLIOGRAFIE

1. **Vliet, Hans van.** MoSCoW-methode. *wikipedia.* [Online] 2008. https://nl.wikipedia.org/wiki/MoSCoW-methode.

2. **OWASP Top Ten. *owasp.* [Online] 2022. https://owasp.org/www-project-top-ten/.**

3. Data protection in the EU. *europa.* [Online] https://ec.europa.eu/info/law/law-topic/data-protection/data-protection-eu_en.

4. GDPR and Data Complexities- How to Apply? *fhict.instructure.* [Online] Fontys. https://fhict.instructure.com/courses/12090/pages/gdpr-and-data-complexities-how-to-apply?module_item_id=751979.

5. GDPR and Data Complexities- How to Apply? *fhict.instructure.* [Online] fhict. https://fhict.instructure.com/courses/12090/pages/gdpr-and-data-complexities-how-to-apply?module_item_id=751979.

6. Database Scaling. *mongodb.* [Online] https://www.mongodb.com/databases/scaling.