# MediaHub

## Software Engineering Project

**Project Team:** Adrian Locher
Benjamin Kern
Dominik Ehrle
Jason Benz
Marco Agostini

**Project Advisor:** Thomas Kälin

Version: 01.00
Date: 10.06.2022

Computer Science
University of Applied Sciences of Eastern Switzerland

# Contents

# Glossary

| Name | Description |
|---|---|
| Contact | People which are added to the contact list and therefore connected to the user |
| Media | All type of media content like movies etc. |
| Frontend | Module containing the website and other user interface elements |
| Backend | Module containing data, the business logic and the persistence binding |
| Profile page | Site with all user and contact data |
| Rate | Process to rate a piece of media |
| Add media to profile | Process of adding a piece of media to a users profile, expressing interest |
| Rating | Specific given value for ”rated media” |

Table 0.1.: Glossary

# Part I.

# Management Summary

# Management Summary

## The Problem

The problem we wanted to solve is the lack of a social media site that focus on the users media interests instead of the user per se. Facebook, Twitter, etc. connect users based on the social graph (who knows whom). This leads to very small bubbles and echo chambers the user lives in. The task at hand was therefore to create a social media site where users can go beyond their usual social range, meeting new people and exchange with others sharing the same interests in different media (e.g. movies, books, etc.).

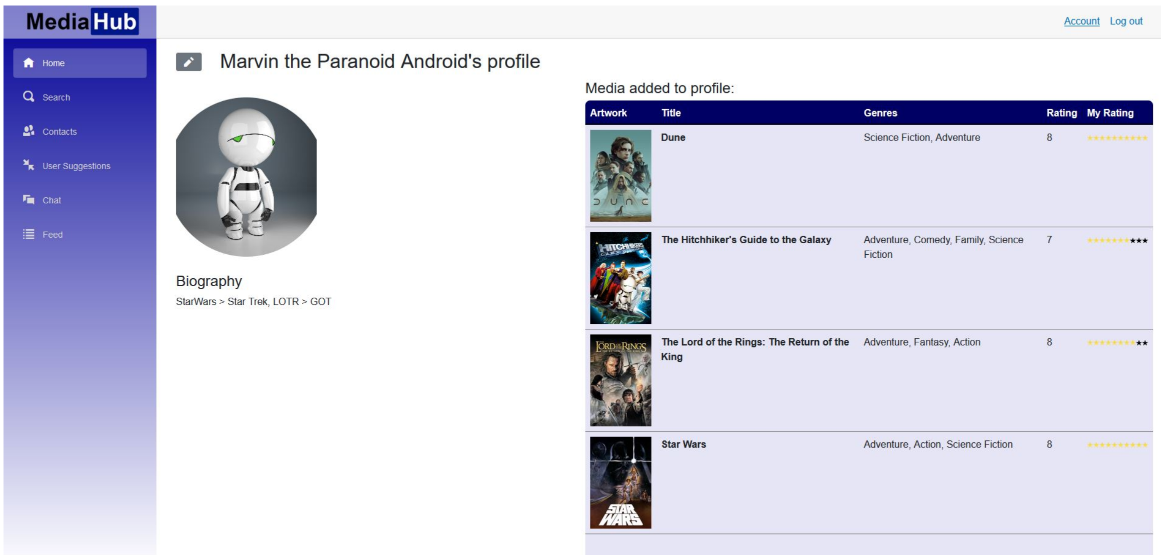
## The Solution

The solution to this problem is a social media site that connects users solely based on their interest. Who is connected to whom does not matter; The only aspect that is considered when suggesting users to each other is their media preferences. Users cannot even search for other users to add. This is very much intentional, as there are other platforms for simply communicating with people one knows. The platform also provides all the surrounding features that are necessary for a media centric platform: A search to discover new media, the ability to like, rate and comment media, administration of ones contacts and new user suggestions as well as a chat to communicate.

## Evaluation

The solution is fitting to solve the problem. While the software could be greatly expanded (e.g. adding group chats, adding media news, supporting media other than movies, etc.) we achieved a solution that solves the problem to a sufficient degree.

The details of the final result can be found in chapter 6.



Marvin the Paranoid Android's profile

Biography

StarWars > Star Trek, LOTR > GOT

Media added to profile:

| Artwork | Title | Genres | Rating | My Rating |
|---|---|---|---|---|
| | Dune | Science Fiction, Adventure | 8 | ★★★★★★★★★★ |
| | The Hitchhiker's Guide to the Galaxy | Adventure, Comedy, Family, Science Fiction | 7 | ★★★★★★★★★★ |
| | The Lord of the Rings: The Return of the King | Adventure, Fantasy, Action | 8 | ★★★★★★★★★★ |
| | Star Wars | Adventure, Action, Science Fiction | 8 | ★★★★★★★★★★ |

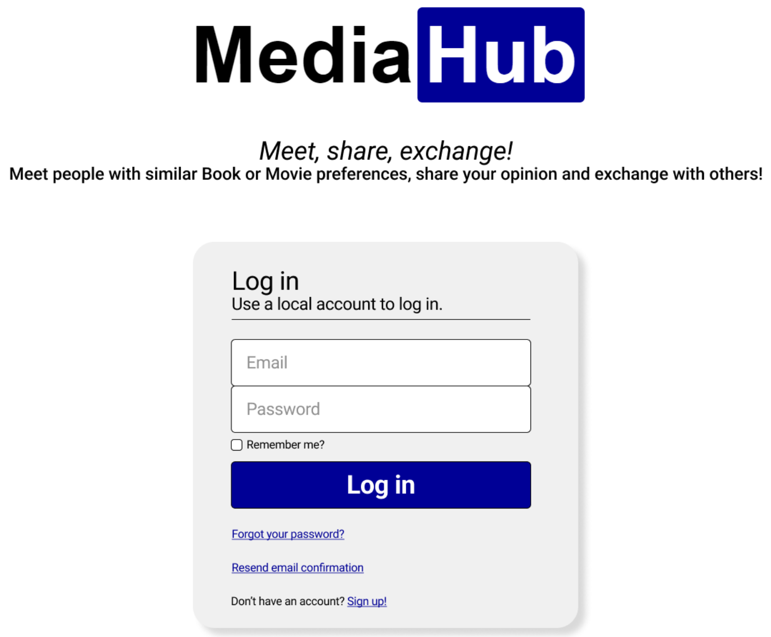
Figure 0.1.: Home-Screen / Profile overview

# Part II.

# Product Documentation

# 1. Vision

The vision of this project is to create a modern platform to connect people based on their interest on different entertainment media (e.g. movies). The users can create a profile and get suggestions for different people who have similar media preferences and might be open for discussions. Ideally, a chat functionality provides this aforementioned discourse opportunity to the users.



MediaHub

*Meet, share, exchange!*

Meet people with similar Book or Movie preferences, share your opinion and exchange with others!

Log in

Use a local account to log in.

Email

Password

Remember me?

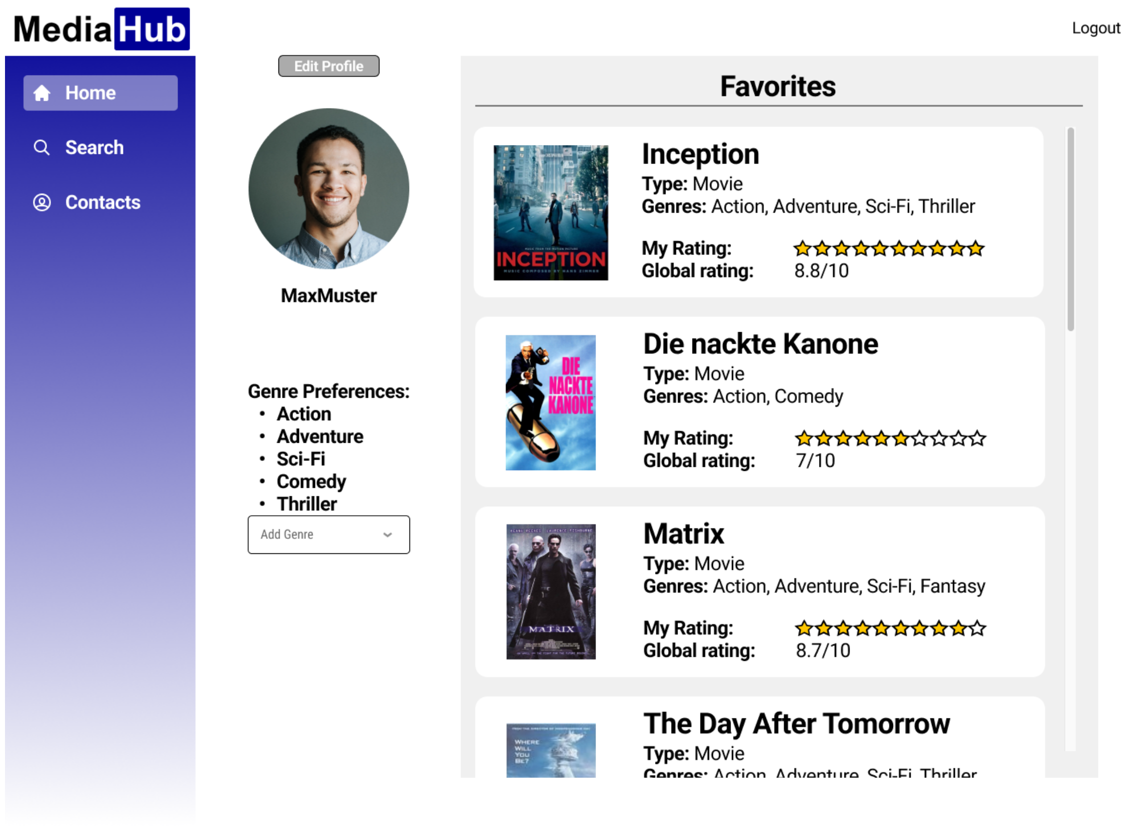
Log in

Forgot your password?

Resend email confirmation

Don't have an account? Sign up!

Figure 1.1.: Mockup Login Page



MediaHub

Logout

Home

Search

Contacts

Edit Profile

MaxMuster

Genre Preferences:
- Action
- Adventure
- Sci-Fi
- Comedy
- Thriller

Add Genre

Favorites

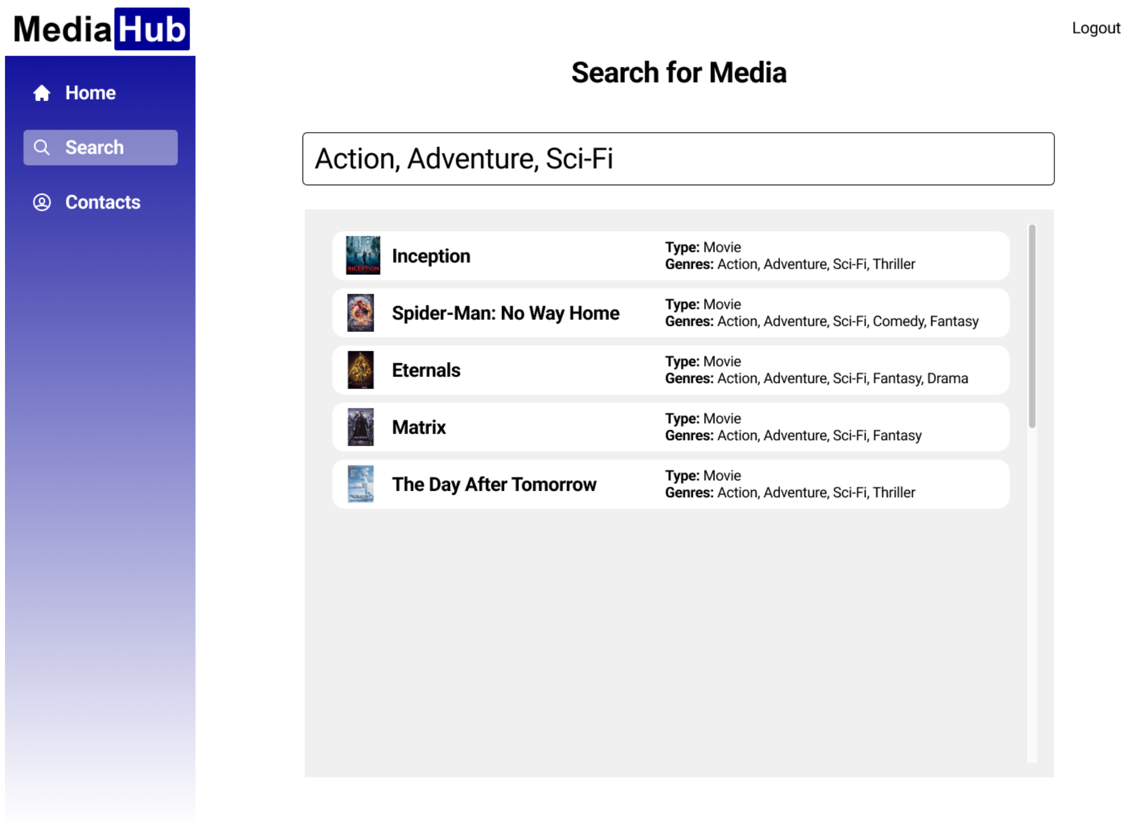
Inception
**Type:** Movie
**Genres:** Action, Adventure, Sci-Fi, Thriller
**My Rating:** ★★★★★★★★★★
**Global rating:** 8.8/10

Die nackte Kanone
**Type:** Movie
**Genres:** Action, Comedy
**My Rating:** ★★★★★★☆☆☆☆
**Global rating:** 7/10

Matrix
**Type:** Movie
**Genres:** Action, Adventure, Sci-Fi, Fantasy
**My Rating:** ★★★★★★★★★☆
**Global rating:** 8.7/10

The Day After Tomorrow
**Type:** Movie
**Genres:** Action, Adventure, Sci-Fi, Thriller

Figure 1.2.: Mockup Profile Page

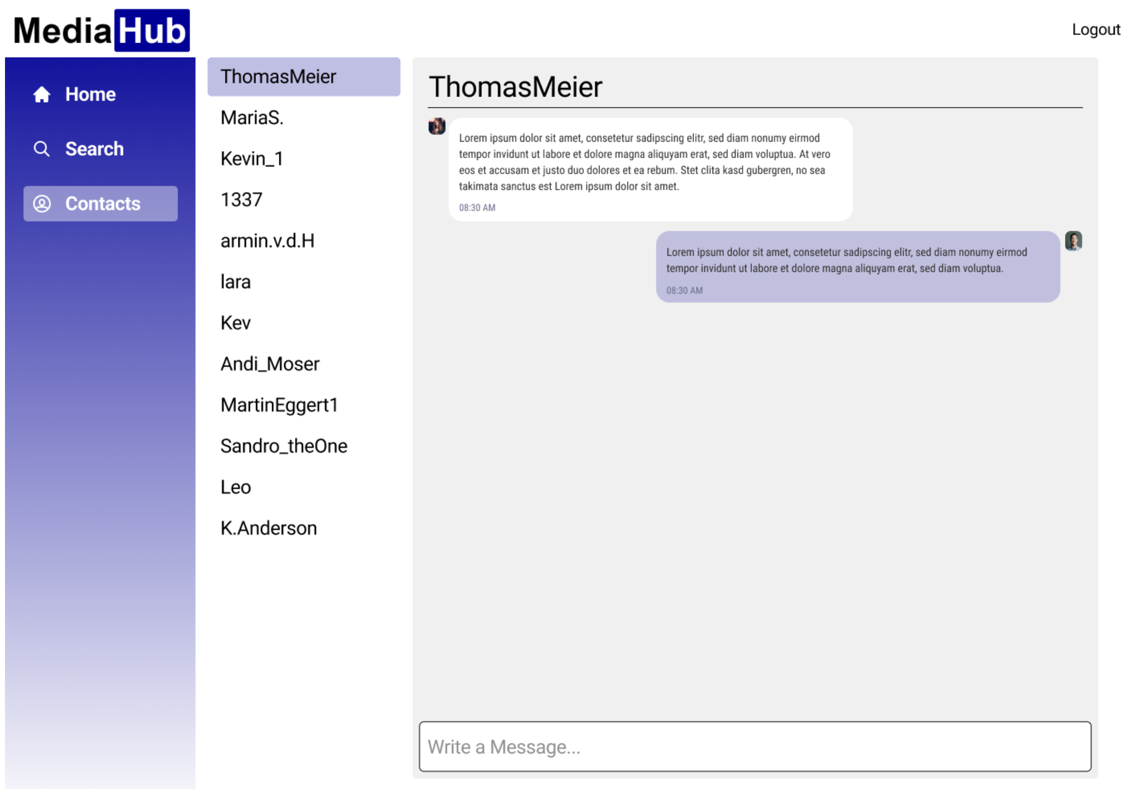


Search for Media

Action, Adventure, Sci-Fi

| | |
|---|---|
| Inception | **Type:** Movie<br>**Genres:** Action, Adventure, Sci-Fi, Thriller |
| Spider-Man: No Way Home | **Type:** Movie<br>**Genres:** Action, Adventure, Sci-Fi, Comedy, Fantasy |
| Eternals | **Type:** Movie<br>**Genres:** Action, Adventure, Sci-Fi, Fantasy, Drama |
| Matrix | **Type:** Movie<br>**Genres:** Action, Adventure, Sci-Fi, Fantasy |
| The Day After Tomorrow | **Type:** Movie<br>**Genres:** Action, Adventure, Sci-Fi, Thriller |

Figure 1.3.: Mockup Search



ThomasMeier

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.
08:30 AM

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.
08:30 AM

Figure 1.4.: Mockup Chat

# 2. Requirements

## 2.1. Functional Requirements

All functional requirements are grouped into epics. The 'must' functional requirements form the MVP line (gray highlighted). The rough priorization is done by categories 'must', 'should' and 'could'. The detailed priorization and the time estimations are done directly on Gitlab and may change during the development process.

### 2.1.1. Refinement Process

In the refinement meeting the team selects the next few requirements (tasks on Gitlab).
In groups of two or three the task will be refined which includes some research and discussing. Afterwards the people will present the refined task to the whole team. If someone has an input or does not agree with the result, the discussion will be continued, otherwise the requirement is ready to be developed.

### 2.1.2. Access Control

| No | Level | Requirement |
|---|---|---|
| #23 | Must | Users create accounts with profiles |
| #23 | Must | The system sends verification emails to the users upon registration |
| #23 | Must | Users login to their accounts with their email-address and password |
| #23 | Must | Users delete their account and with it their profile |
| #23 | Should | Users reset or change their password |
| #23 | Should | Users edit their profile |
| #37 | Could | Users use two-factor authentication for login |

Table 2.1.: Access Control Requirements

### 2.1.3. Media Management

| No | Level | Requirement |
|---|---|---|
| #63 | Must | Users see a media overview |
| #38, #40 | Must | Users rate media, that they've seen and by that, add them to their profile |
| #39 | Must | Users search for media by name |
| #41 | Should | Users comment media |
| #42 | Could | Users manually capture media and they are visible for all other users afterwards as well |
| #28 | Could | Users subscribe to genres |
| #76 | Could | Users can watch trailers of movies |

Table 2.2.: Media Management Requirements

### 2.1.4. Feed

| No | Level | Requirement |
|---|---|---|
| #44 | Should | Users see all activities of their friends on a feed page |
| #45 | Could | Users see media news (from newspaper pages, social medias etc.) |
| #46 | Could | Users see all notifications |

Table 2.3.: Feed Requirements

### 2.1.5. Chat functionality

| No | Level | Requirement |
|---|---|---|
| #47 | Must | Users can chat with each other |
| #48 | Could | Users send images |
| #49 | Could | Users send media references from the application |
| #50 | Could | Users create chat groups |
| #51 | Could | Users get a read confirmation |
| #52 | Could | Users delete their local chat history |
| #53 | Could | Users delete sent message on all related devices |
| #54 | Could | Users forward messages |
| #55 | Could | Users send broadcasts |

Table 2.4.: Chat Functionality Requirements

### 2.1.6. List of friends

| No | Level | Requirement |
|---|---|---|
| #56 | Must | Users get suggestions of userprofiles, of users with similar media preferences |
| #57 | Should | Users manage their friends |
| #58 | Could | Users block annoying users |
| #59 | Could | Users search other users by their username |

Table 2.5.: Friend List Requirements

### 2.1.7. Personal profile

| No | Level | Requirement |
|---|---|---|
| #61 | Must | Users have a personal public profile |
| #60 | Must | Users see the rated movies in an overview on their profile page |
| #61 | Should | Users set a profile picture |
| #61 | Should | Users set a public visible short description / biography |
| #64 | Could | Users choose between different levels of privacy for their accounts |
| #62 | Could | Users see some statistic about their favorite genres etc. |

Table 2.6.: Personal Profile Requirements

### 2.1.8. Infrastructure

This epic is all about setting up the infrastructure and no classical epic with a increment for the customer. Most of the other epics depend on this one.

| No | Level | Requirement |
|---|---|---|
| #15 | Must | A developer sets up the database |
| #7 | Must | A developer sets up and configure the server |
| #2, #6 | Must | A developer sets up CI / CD |
| #4, #5 | Must | A developer sets up the project repository |
| #68 | Should | A developer sets up container for the infrastructure |

Table 2.7.: Functional Requirements for the Infrastructure

### 2.1.9. Planning

All product (MediaHub) specific planning requirements are tracked in this epic. This is everything which is required or of interest for the customer such as requirement engineering, architecture draft etc.

| No | Level | Requirement |
|---|---|---|
| #3 | Must | The team hold a kickoff meeting |
| #75 | Must | A team member creates a rough project plan |
| #8 | Must | A team member captures all milestones |
| #12 | Must | A team member identifies and classifies all possible risks |
| #14 | Must | A team member creates a long term plan |
| #17 | Must | A team member creates a detailed short term plan |
| #9 | Must | A team member identifies all functional requirements |
| #18 | Must | A team member identifies all non-functional requirements |
| #25 | Must | A team member defines all important workflows |
| #19 | Must | A team member creates a domain model |
| #21 | Must | A team member creates a frontend mockup |
| #22, #73 | Must | A team member creates an architecture concept |

Table 2.8.: Functional Requirements for the planning

## 2.2. Non-functional Requirements

The following non-functional requirements (NFRs) are specified for our product. The verification methods are defined in subsection 2.2.1 Verification Methods, the test-results can be seen in subsection 2.2.2 NFR-Tests.

| ID | Category | Priority | Requirement |
|---|---|---|---|
| NFR-1 | Time-behavior | Medium | The system reacts to interactions in appropriate time |
| NFR-2 | Capacity | Medium | The system architecture allows for a reasonable amount of users |
| NFR-3 | Capacity | High | The development system can handle at least 5 simultaneous users |
| NFR-4 | Usability | Medium | Logging into the system is achievable in a reasonable time |
| NFR-5 | Usability | Medium | Searching a movie and adding it to the profile is achievable in a resonable time |
| NFR-6 | Non-repudiation | High | Text-messages are non-reputable |
| NFR-7 | Authenticity | High | Authentication is the only action unauthenticated users can perform on the system |
| NFR-8 | Modularity | High | The business-logic and data-model of the system are independent of the UI-Technology, database and APIs |
| NFR-9 | Reusability | High | The business-logic and data-model are reusable for other interfacing plattforms on or of the www |
| NFR-10 | Testability | High | The business-logic, data-model and the UI are testable independently |
| NFR-11 | Installability | Medium | The server runs on Linux- as well as Windows-based operating systems |

Table 2.9.: Non-Functional Requirements

### 2.2.1. Verification Methods

| NFR-ID | Verification method |
|---|---|
| NFR-1 | Automated testing using Cypress and usertests |
| NFR-2 | Manual verification of components |
| NFR-3 | Stress testing and manual verification |
| NFR-4 | Automated testing using Cypress |
| NFR-5 | Automated testing using Cypress |
| NFR-6 | Manual verification of the data model |
| NFR-7 | Manual verification of the authentication mechanism |
| NFR-8 | Manual verification of the architecture |
| NFR-9 | Manual verification of the architecture |
| NFR-10 | Manual verification by defining tests |
| NFR-11 | Test by installation of the system on the specified platforms |

Table 2.10.: NFR Verification Methods

| NFR-ID | Trigger | Reaction | Measures |
|---|---|---|---|
| NFR-1 | Users want a system that feels responsive | The system is designed with performance in mind | The system reacts to interactions in less than 5 seconds |
| NFR-2 | The development team does not know the amount of users the system will have | The team designs the system in a scalable way | Review of the code and components does not show hard limitations of scalability |
| NFR-3 | The system should handle at least enough users to be demonstrable | components and resources are allocated accordingly | The development system is able to handle average load of at least 5 simultaneous users |
| NFR-4 | Users lose interest in a system that has bad usability | The login-UI is implemented as simple and fast to use as possible | Login can be done in less than 15 seconds |
| NFR-5 | Users want a system to be fast to use | The UI is implemented as simple and fast to use as possible | Searching for a well known movie and adding it to the profile takes less than 15 seconds |
| NFR-6 | Users might post/send illegal contents | The chat system is designed in a non-reputable way | Text-Messages are always stored with data, that uniquely identifies a user |
| NFR-7 | Anonymous users might post/send illegal contents | The system is made accessible only through authentication | Access to the system is prohibited through an authentication mechanism |
| NFR-8 | The UI-Technology and database/APIs might lose support by their maintainers | The named components are included in a replaceable manner | Dependency injection and similar patterns are used to completely decouple the components from each other |
| NFR-9 | Requirements of the platform might change | The named components are included in a replaceable manner | There are no dependencies from Non-UI Components to UI Components |
| NFR-10 | The test environment must be fixable | The named components are made mockable | Dependency injection and similar patterns are used to completely decouple the components from each other |
| NFR-11 | Future operators of the system might depend on this | The system is designed platform independent | No platform-dependent components are used, the system is tested on each of the platforms |

Table 2.11.: NFR Triggers, Reactions and Measures

### 2.2.2. NFR-Tests

#### Alpha Release

| NFR-ID | Passed | Notes |
|---|---|---|
| NFR-1 | Yes | Manually tested by user tests |
| NFR-2 | Yes | |
| NFR-3 | Yes | Manually tested by user tests |
| NFR-4 | Yes | Tests implemented in Cypress (See folder nfr-testing) |
| NFR-5 | Yes | Tests implemented in Cypress (See folder nfr-testing) |
| NFR-6 | Yes | |
| NFR-7 | Yes | |
| NFR-8 | Yes | |
| NFR-9 | Yes | |
| NFR-10 | Yes | |
| NFR-11 | Yes | |

Table 2.12.: Alpha Release NFR Tests

#### Beta Release

| NFR-ID | Passed | Notes |
|---|---|---|
| NFR-1 | Yes | Manually tested. Login takes by far the longest time to respond. |
| NFR-2 | Yes | |
| NFR-3 | Yes | Manually tested by developer tests |
| NFR-4 | Yes | Tests implemented in Cypress (See folder nfr-testing) |
| NFR-5 | Yes | Tests implemented in Cypress (See folder nfr-testing) |
| NFR-6 | Yes | |
| NFR-7 | Yes | |
| NFR-8 | Yes | |
| NFR-9 | Yes | |
| NFR-10 | Yes | |
| NFR-11 | Yes | |

Table 2.13.: Beta Release NFR Tests

## 2.3. Design Decision

We have used ASP .NET Core because it is a technology with which the whole team is familiar. Reasones for the chosen technologies are statet in the Table 2.14.

| **Technology** | **Name** | **Statement** |
|---|---|---|
| Frontend, Backend | Blazor Server | Considering the available technologies, we used Blazor Server because it works best together with ASP .NET Core. |
| E-Mail Service | SendGrid | We used the Twilio SendGrid service because of his good reputation and also the already existent C# API. |
| Database | Microsoft SQL Server | We used the current Microsoft SQL Server 2019 as our backend database. This technology was chosen because we already use Microsoft technologies for the application and wanted to have an ecosystem from one provider. |
| OR Mapping | EF Core | The EF Core tool was introduced to most of us during the lecture in the previous semester. Therefore we decided to use our knowledge in this project. |
| Hosting Plattform | Docker | Most of us have already worked with Docker and some of us also with Kubernetes. Therefore we decided to use Docker for a simple and easy deployment of our application. |
| Movie API | TMDB | We have chosen to use TMDB over the very popular IMDB API, because it offers more capabilities without a paid subscription. |

Table 2.14.: Design Decisions

# 3. Domain Analysis

## 3.1. Overview

The domain model only incorporates the must requirements i.e. the problem domain of the MVP. Unfortunately, domain models cannot capture all processes and relations. The following important but invisible must requirements/aspects cannot or should not (considering the level of abstraction) be shown using a domain model:

- A user receives an email upon registration
- Users can delete their account
- Users can search for media by name
- Users see their rated movies in an overview on their profile page
- If the user similarity of two users exceeds the threshold, the users are recommended to each other.

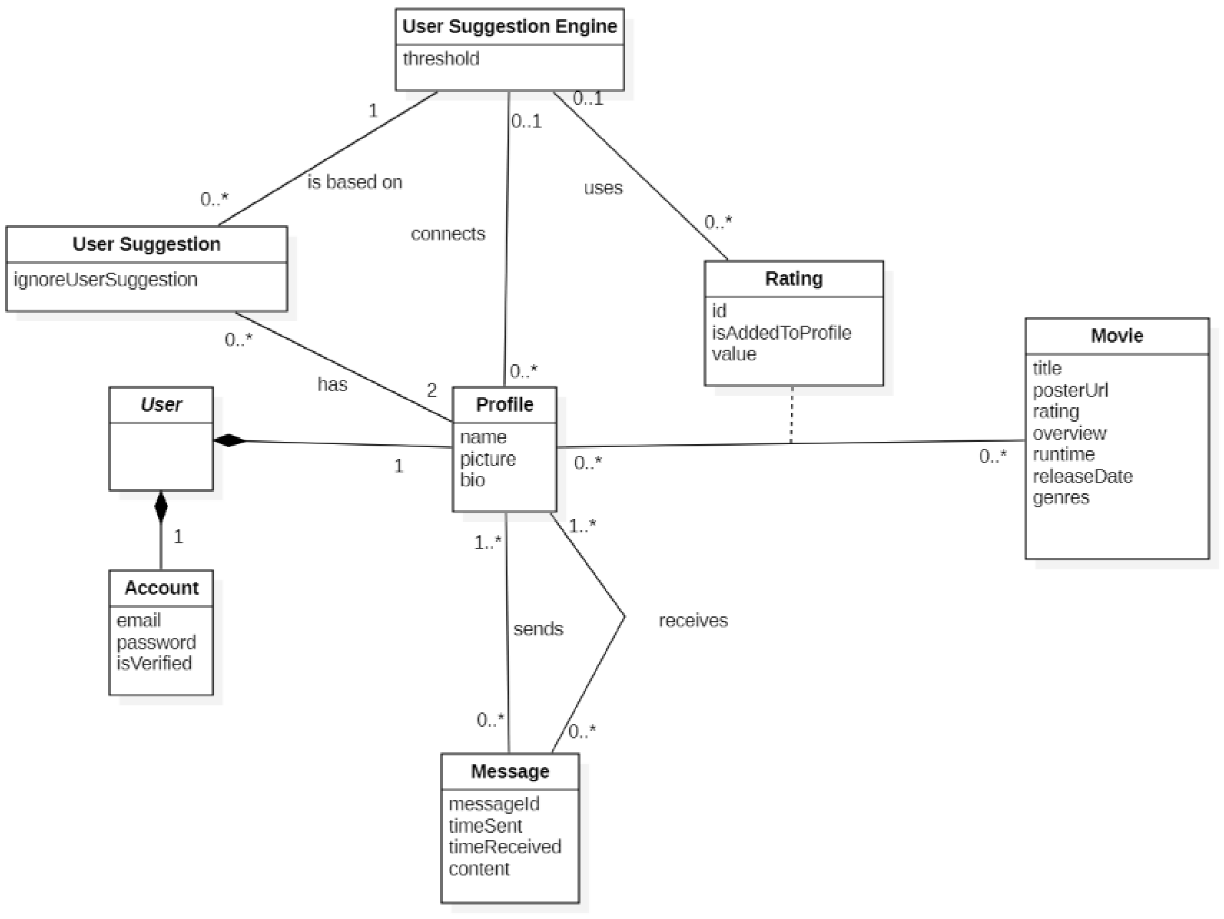


Figure 3.1.: Domain Model of the MediaHub-MVP

## 3.2. User Suggestions

The heart of the application is the social interaction possibility. Compared to other online media collections, the aspect of social interaction with other users having similar interests is the core value added by MediaHub. One of the key components for that is our user suggestions algorithm. There are many ways to realize this algorithm. Therefore in this chapter only a proof of concept is described, which includes the rule set (which user should be suggested) and possible technical ways to realize it.

### 3.2.1. Definition of Rules

The MVP specifies that a user can express his excitement about specific media in two ways:

1. Add media to profile (in the following chapters 'add' is used as shortform)
2. Rate media (scale from 1 to 10)

In the future (optional requirement), a user may also be able to follow a genre, which then would also be included in the calculation.

#### Rule set

The following rule set is the theoretical structure. It may be that optimizations and adjustments are being made during development or in production. Examples for the application ot the rule set can be seen in Table 3.1.
Two users should be suggested to eachother, when they have at least five matches. A match can be:

- Case 1: Both added the same media item
- Case 2: Both rated the same media item at least 7/10
- Case 3: One added a media item and the other rated the same media item at least 7/10

| **Media item** | **Alice** | **Bob** | **Match** | **Description** |
|---|---|---|---|---|
| RED | Added | Added | Yes | Case 1 |
| Die Hard | Rated 10/10 | Rated 9/10 | Yes | Case 2 |
| Winnetou | Added | Rated 8/10 | Yes | Case 3 |
| The Expendables | Rated 9/10 | Added | Yes | Case 3 |
| Star Wars | Rated 6/10 | Rated 10/10 | No | Only Bob rated $\geq$ 7/10 |
| Baywatch | Added | | No | Bob didn't interact with the media item |
| The Fast and the Furious | | Rated 9/10 | No | Alice didn't interact with the media item |

Table 3.1.: Examples of a match

### 3.2.2. Technical opportunities

Prerequisites are the existence of the add and rate media functionalities and a contact list as well as the according database table(s).
The user suggestion mechanism requires its own table, where every user suggestion can be inserted. If a suggestion gets accepted by both users and ends up in a user contact, the user suggestion entry must be deleted or flagged. Before implementation, it is necessary to check if the tables (e.g. user profiles, contacts, etc.) already exist due to another task or if they need to be created.
Figure 3.2 shows an overview of the user suggestion process.

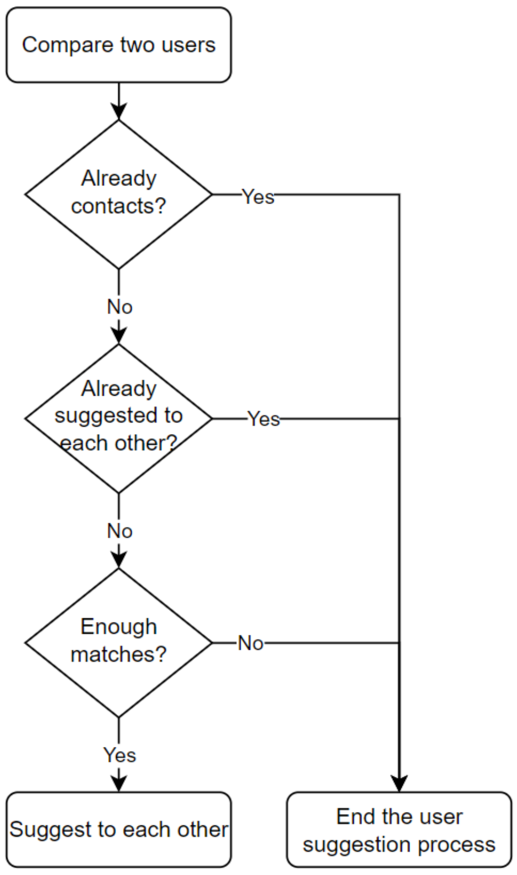


Figure 3.2.: User suggestion process overview

### Option 1 - Daily suggestions

One way to realize it is a cron job (”command run on notice”) on a daily basis. E.g. every day at midnight the job starts and all matches between all users are evaluated. An advantage of this option is, that the job is separated from the product frontend. Therefore the sytem workload of the frontend won’t be that high. But the big disadvantage is the required runtime. Due to the fact that all users are compared with each other, the cron job will run for a long time as soon as there are some daily active users. A lot of optimization and scaling will be needed to handle the duration and the generated database traffic.

### Option 2 - Instant suggestions

Another option is to instantly run the suggestion mechanism each time a user rates or adds a media item. It may be that one of the mechanism runs several times a day for the same user. However, many small jobs are spread over the whole day instead of one big one at night. Not all users have to be compared with each other, but only those who add or rate new media. Another benefit is that users thus receive new suggestions much faster.

### Decisions

Option 2 seems to be much more reliable and it has more advantages, which is why this option is preferred. A possible optimization would be to start the process only a few minutes (e.g. 2 min) after the last add or rating. So if a user adds or rates several media items directly one after the other, the mechanism has to be performed only once.

# 4. Architecture

## 4.1. Context, Containers and Components

This section contains the software architecture of this project according to the C4 documentation standard. The 4th 'C' (Code) is contained in chapter 3 Domain Analysis, in the form of a domain model.

### 4.1.1. Context

Figure 4.1 shows the external systems our product interfaces with.
TMDB and SendGrid were chosen to comply with NFR-2 and NFR-3. While TMDB already complies right now, SendGrid needs an upgrade to a premium version of the service in a productive system.

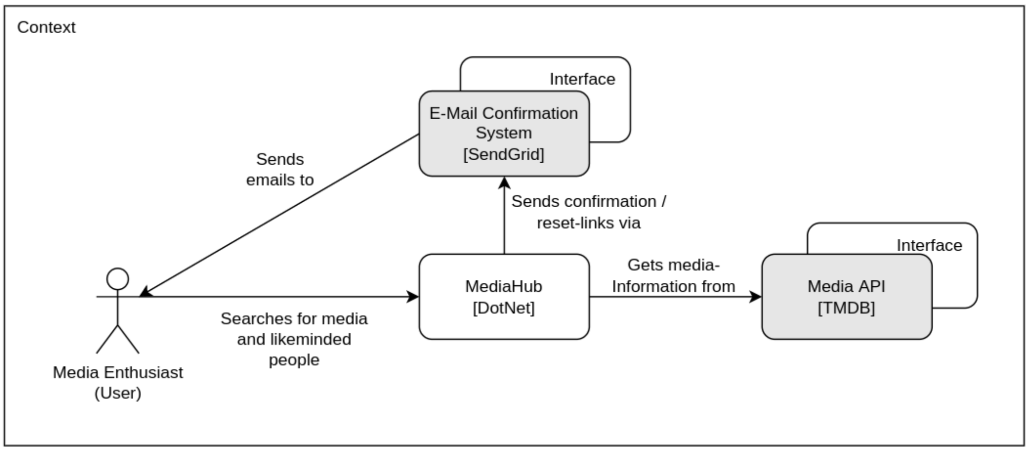

Context
Interface
E-Mail Confirmation System [SendGrid]
Sends emails to
Sends confirmation / reset-links via
Interface
Gets media-Information from
MediaHub [DotNet]
Media API [TMDB]
Searches for media and likeminded people
Media Enthusiast (User)

Figure 4.1.: Context

### 4.1.2. Container

The container (dotted line) contains the complete system.

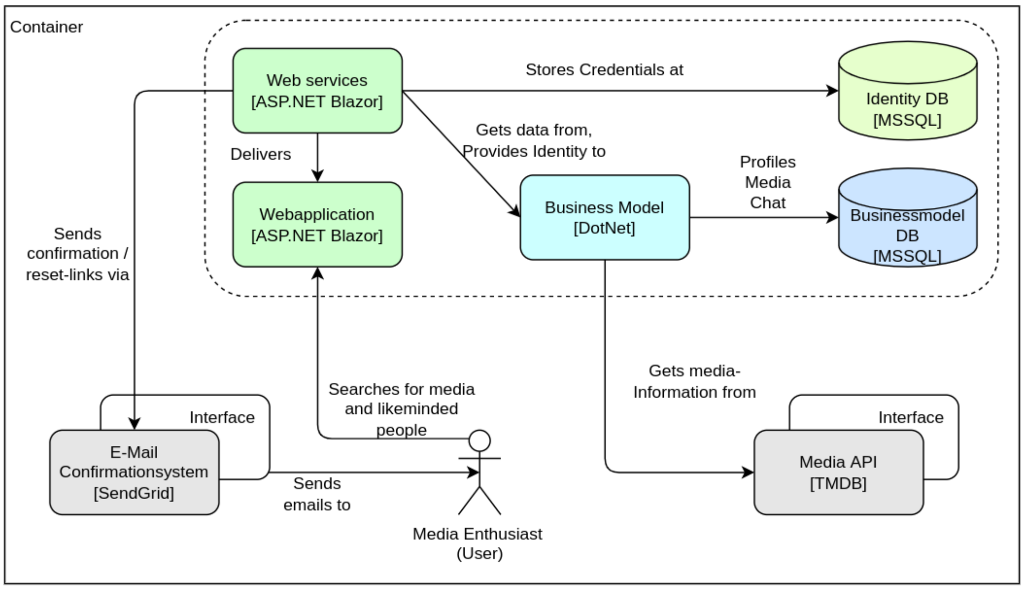

Container
Web services [ASP.NET Blazor]
Stores Credentials at
Identity DB [MSSQL]
Gets data from, Provides Identity to
Delivers
Profiles Media Chat
Webapplication [ASP.NET Blazor]
Business Model [DotNet]
Businessmodel DB [MSSQL]
Sends confirmation / reset-links via
Gets media-Information from
Searches for media and likeminded people
Interface
E-Mail Confirmationsystem [SendGrid]
Sends emails to
Interface
Media API [TMDB]
Media Enthusiast (User)

Figure 4.2.: Containers

### 4.1.3. Components

There are two main components in our system. The frontend, consisting of all web services (UI), the business model containing all business data and logic and the persistency layer that interfaces with the databases and APIs.

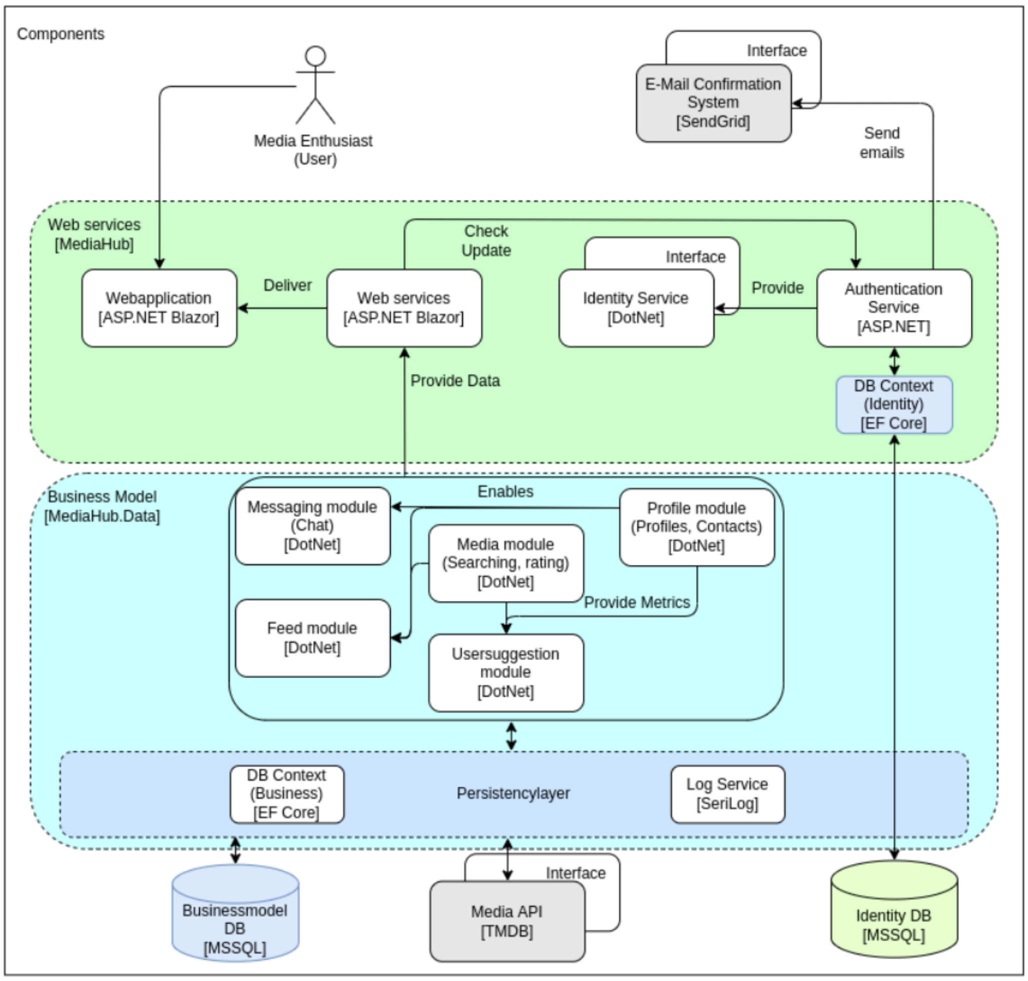

Components
Media Enthusiast (User)
Interface
E-Mail Confirmation System [SendGrid]
Send emails
Web services [MediaHub]
Check Update
Interface
Webapplication [ASP.NET Blazor]
Deliver
Web services [ASP.NET Blazor]
Identity Service [DotNet]
Provide
Authentication Service [ASP.NET]
Provide Data
DB Context (Identity) [EF Core]
Business Model [MediaHub.Data]
Enables
Messaging module (Chat) [DotNet]
Profile module (Profiles, Contacts) [DotNet]
Media module (Searching, rating) [DotNet]
Provide Metrics
Feed module [DotNet]
Usersuggestion module [DotNet]
DB Context (Business) [EF Core]
Persistencylayer
Log Service [SeriLog]
Interface
Businessmodel DB [MSSQL]
Media API [TMDB]
Identity DB [MSSQL]

Figure 4.3.: Components

## 4.2. Layers and project structure

The architecture of the system in its layers is shown in Figure 4.4.
We've chosen a *clean* approach, where each layer in the circle can depend on everything going inwards, while outwards dependencies are forbidden. Therefore all solid arrows only go from outer layers to inner layers. Dashed arrows represent dependencies that have been abstracted by dependency inversion. The outermost layer ("Configuration / DI") is the DI-Configuration layer, which injects dependencies into dependent modules.

Clean architecture was chosen in combination with an MVVM ("Model-View-ViewModel") approach for splitting backend from frontend, to comply with NFR-8, NFR-9 and NFR-10.

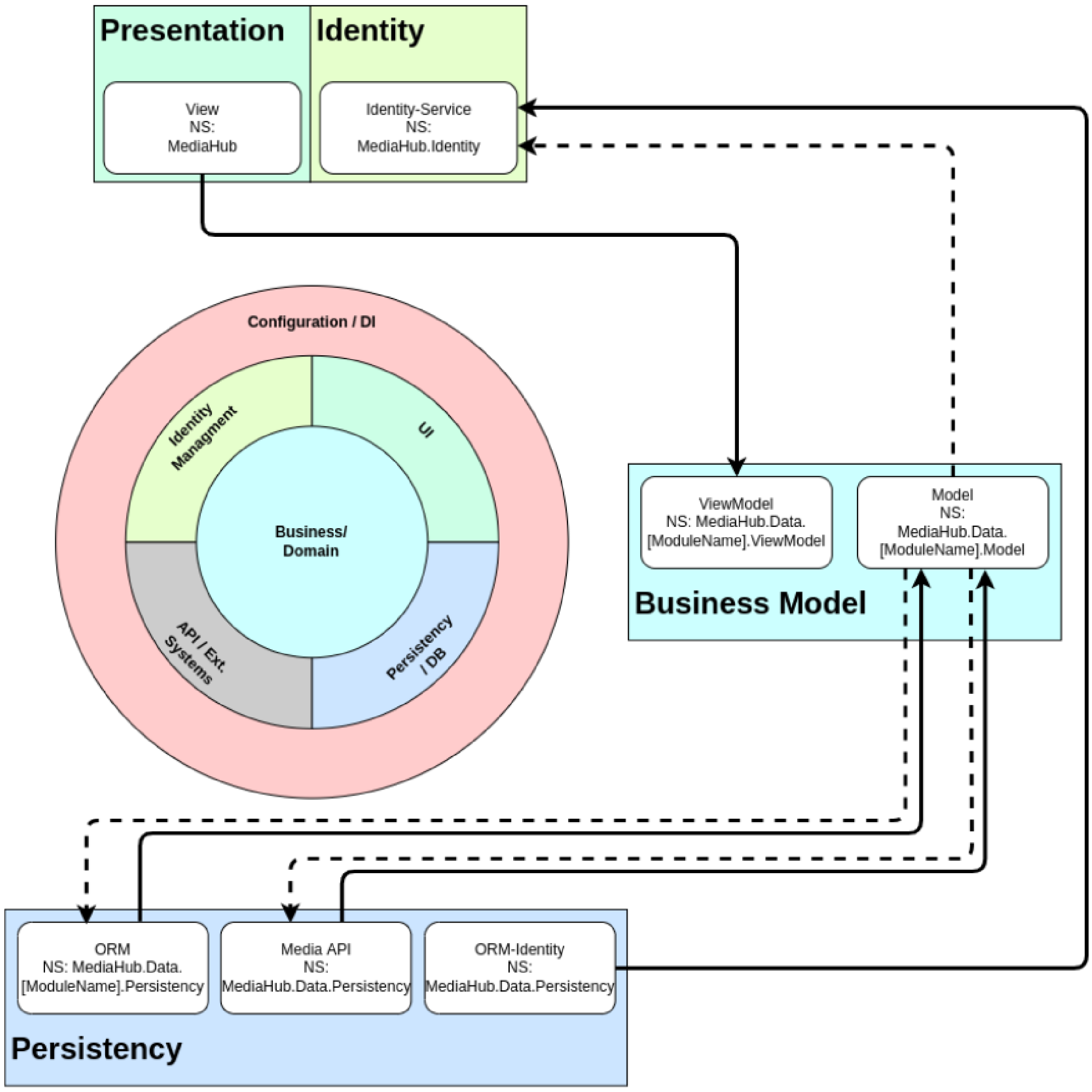


Figure 4.4.: Architecture

## 4.3. Sequences

The Figure 4.5 shows sequences of commands throughout the system. The "General" sequence is meant as a template how sequences usually run through the system. Following the "General" sequence are different kinds of sequences, each portraying how the communication works regarding their feature. Note that this diagram is not complete. The features "feed" and "add contacts" are omitted. Their sequences work in a similar scheme as the sequences displayed in this diagram.
Dashed lines indicate actions that might be omitted, if data is already loaded.

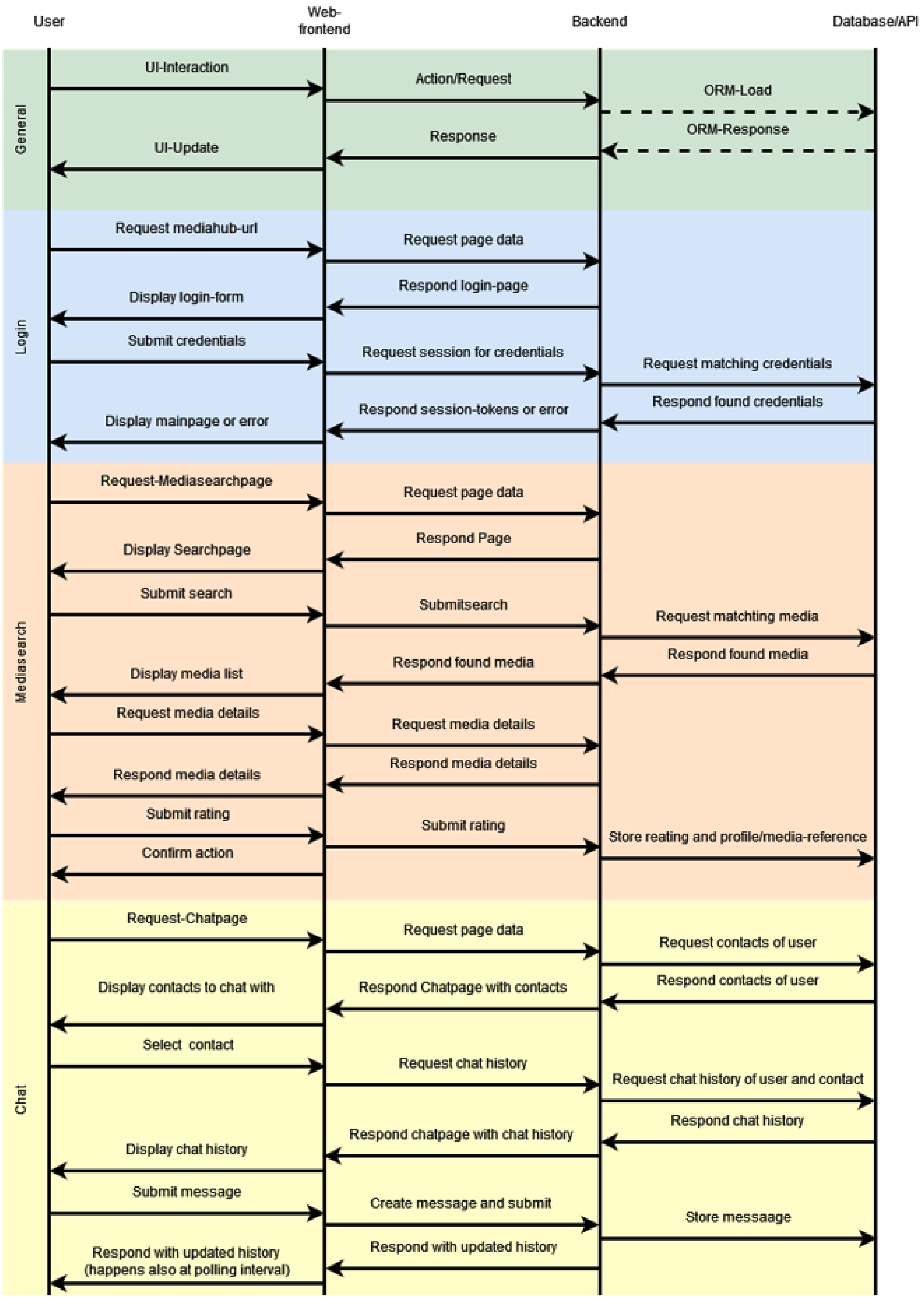


Figure 4.5.: Sequence Diagramm

## 4.4. Dependencies

The Figure 4.6 shows dependencies between different components of the system from an implementations point of view.

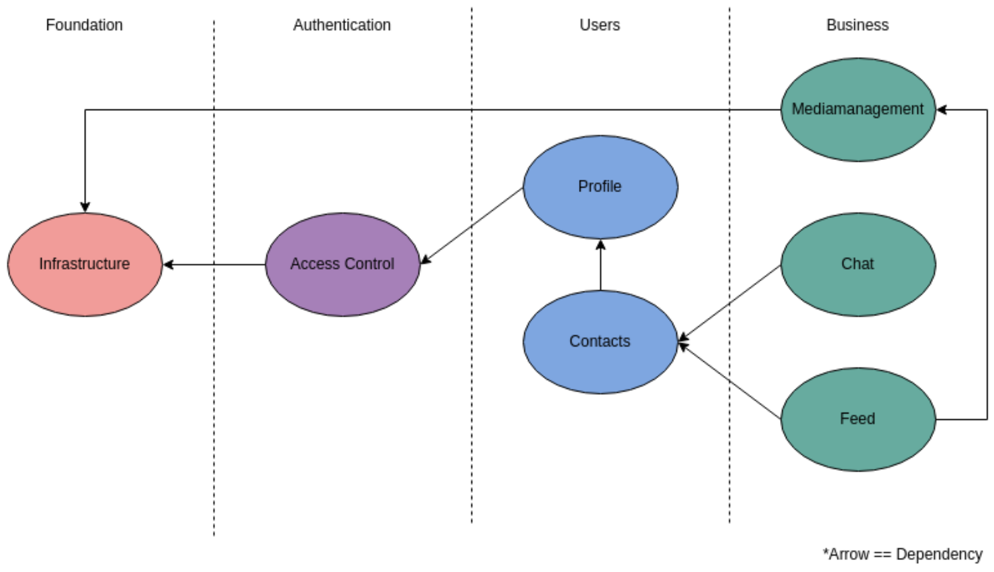


Figure 4.6.: Dependencies of components

## 4.5. User-Interaction

The Figure 4.7 graphically describes the flow of user interactions concerning the MVP. It abstracts the chain of actions that can be performed and what will be done before a certain action is reached. The page-flow is chosen to comply with NFR-7.

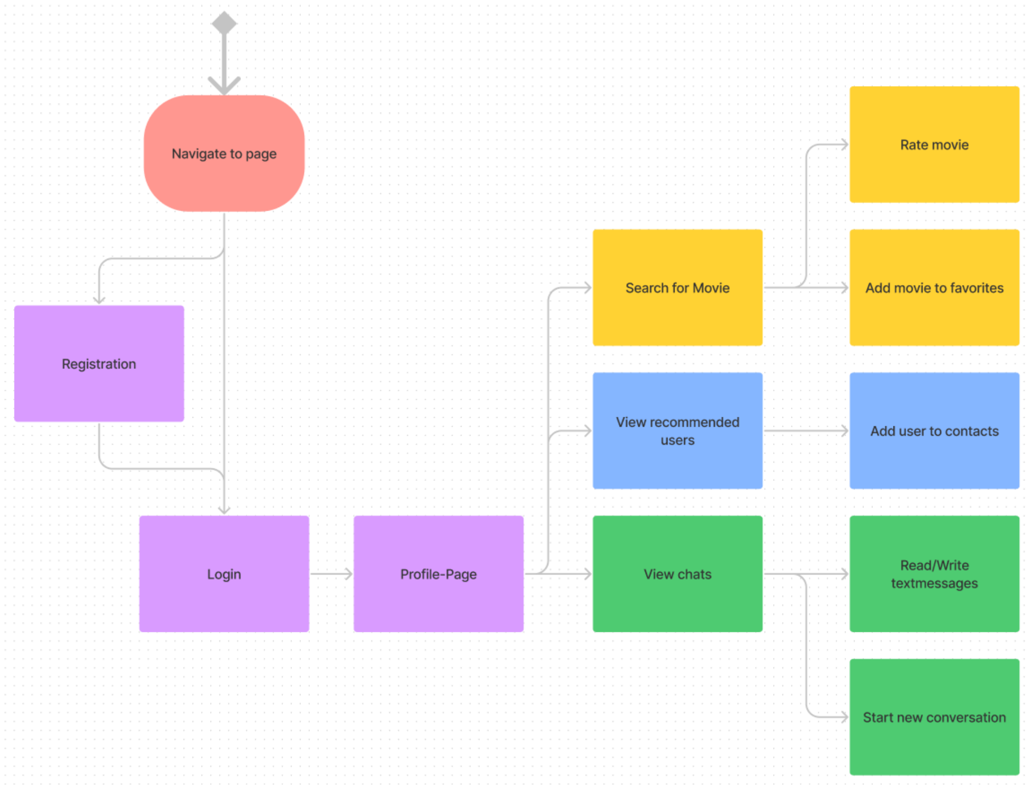


Figure 4.7.: User-Interaction Diagram

## 4.6. Deployment

This section describes the deployment as well as the container architecture.

### 4.6.1. Deployment Process

The deployment handling is done via Gitlab. There are different stages in the process to ensure consistent code quality and deployment onto the development server.

1. Unit tests are triggered as soon as the developer merges a branch into the main branch or push into any other branch.
2. On Gitlab a docker image is build and saved to the repository to ensure the build doesn't fail.
3. An automatic code review in our SonarQube instance is executed.
4. The developer can manually trigger an additional pipeline job to deploy the newest application onto the deployment server.

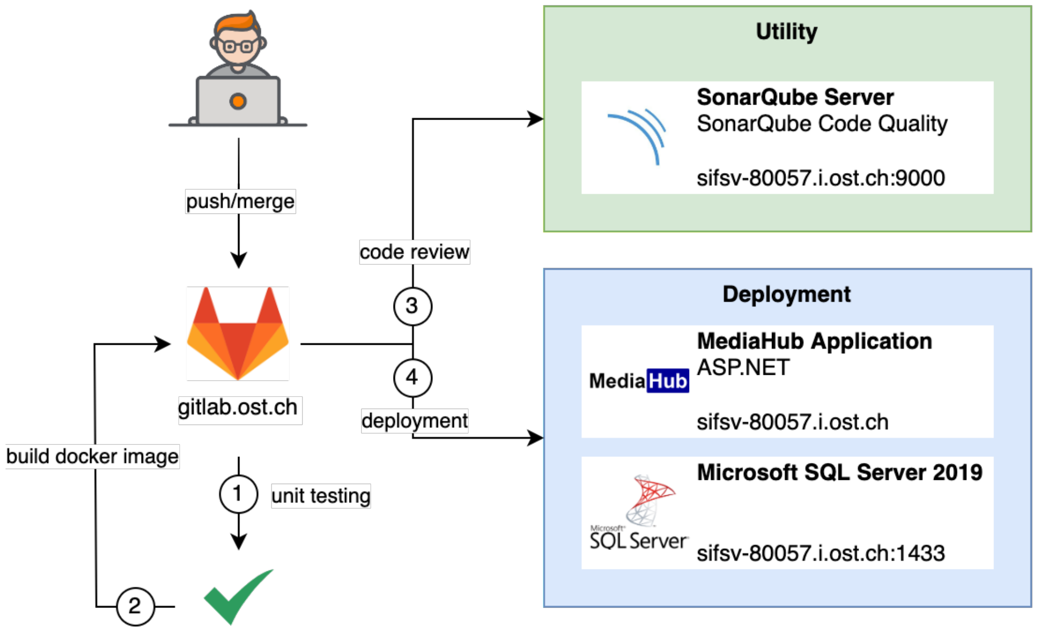


Figure 4.8.: Gitlab Deployment Strategy

### 4.6.2. Deployment Architecture

Our application is containerized and therefore deployed via Docker. We automatically can start a deployment of the application with 'docker compose'. The deployment consist of a Microsoft SQL Server 2019 and our MediaHub application. The deployment sever does expose the HTTP port (TCP: 80) in order to make our application reachable from the network. The application container does communicate over the docker network with the database server.

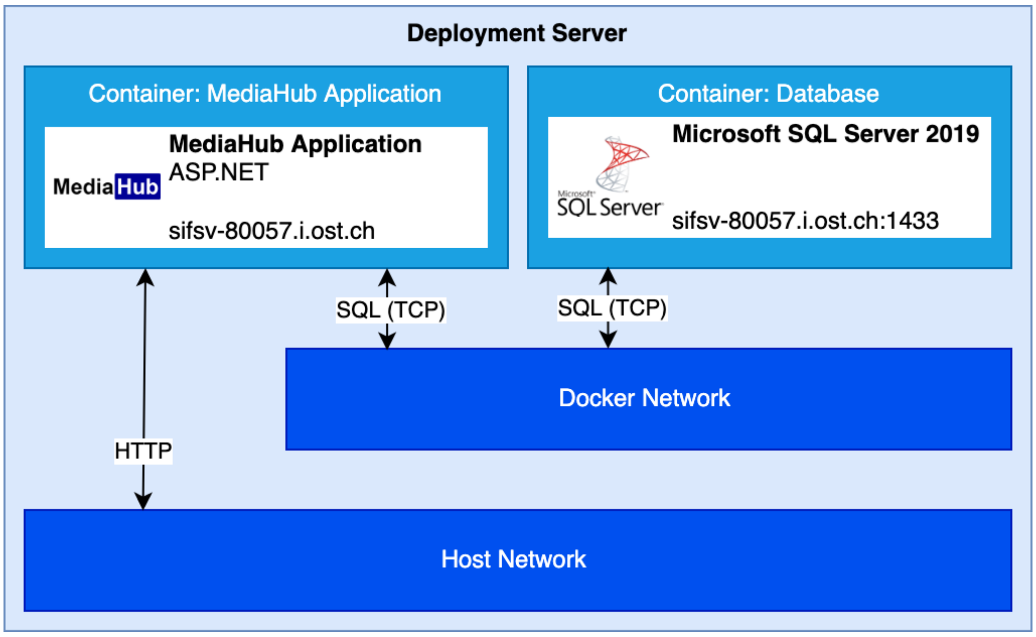


Figure 4.9.: Deployment of Container

# 5. Quality Measures

## 5.1. NFR Quality Measures

The NFRs can be found in section 2.2 Non-functional Requirements (2 Requirements).
Upon implementing a component of our product, the NFRs are checked for dependencies to components and if there are dependencies the tests for the components are implemented accordingly. NFR verification methods in this document are declared as general as possible and as precise as needed. The exact scope of the verification is decided when the component is implemented.

Usability tests for NFRs (5.4.3 Usability Testing) are manually executed whenever a feature is implemented. The tests are also executed right before any release (Branch: ”release”).

## 5.2. Quality Assessment Tools

To assess the quality of our code, we use Sonarqube to track the quality in a central way in our code repository. We also use Sonarlint to get real-time suggestions while writing code.

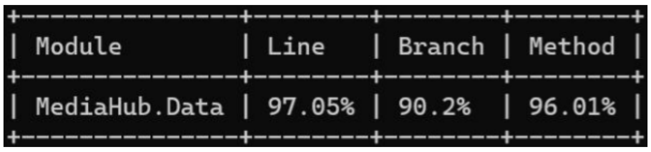
### 5.2.1. Quality Metrics

Independent of our NFRs, three code-metrics are chosen for tracking the quality of our product.

- **Maintainability** → Tracked by Sonarqube
- **Test-Coverage** → Tracked in the form of branch coverage by coverlet (80%)
- **Coupling** → Tracked in the form of module coupling (DB-Persistency, UI, Data model, ...) by review of the architecture

### 5.2.2. End of Project Analysis

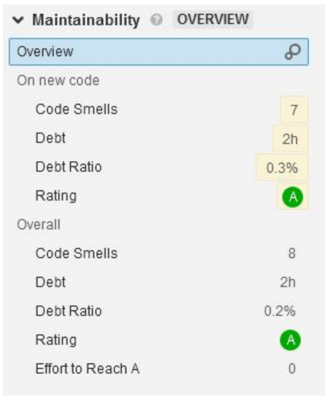
At the end of the project, a branch-coverage of 90.2% was reached.



```
+---------------+--------+--------+--------+
| Module        | Line   | Branch | Method |
+---------------+--------+--------+--------+
| MediaHub.Data | 97.05% | 90.2%  | 96.01% |
+---------------+--------+--------+--------+
```

Figure 5.1.: Test-Coverage report

Sonarqube reports a maintainability rating of ’A’. Features implemented by our team have 0 hours of code debt. The 2 hours reported by Sonarqube stem from the Microsoft authentication module we used.



Maintainability OVERVIEW
Overview

| | |
|---|---|
| On new code | |
| Code Smells | 7 |
| Debt | 2h |
| Debt Ratio | 0.3% |
| Rating | A |
| Overall | |
| Code Smells | 8 |
| Debt | 2h |
| Debt Ratio | 0.2% |
| Rating | A |
| Effort to Reach A | 0 |

Figure 5.2.: Sonarqube maintainability report

## 5.3. Collaboration workflow

The following diagrams show the workflow that is used to collaboratively work on the implementation of this project. Since Git is used as our version control system, the diagrams show the process in a Git-perspective.

The Figure 5.3 shows the branching strategy. The main-branch is the one which gets actively deployed to our server during production.
Feature-branches are used to implement features and task-branches to implement the individual "parts" of a feature. The release-branch contains commits from frozen states of the system (alpha, beta, etc.).

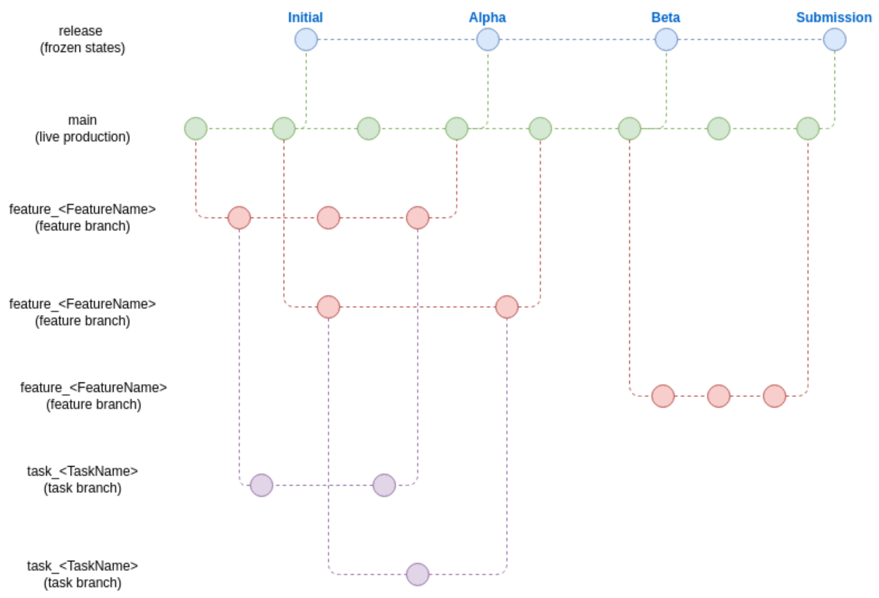


Figure 5.3.: Branching Strategy

The Figure 5.4 shows our workflow within the specified branches. A developer, tagged with a branch-color, is in some way affiliated with the branch of this color. After merging, the branches are deleted. Commits to the release-branch are done after the current state of the product got verified by every member of the team.

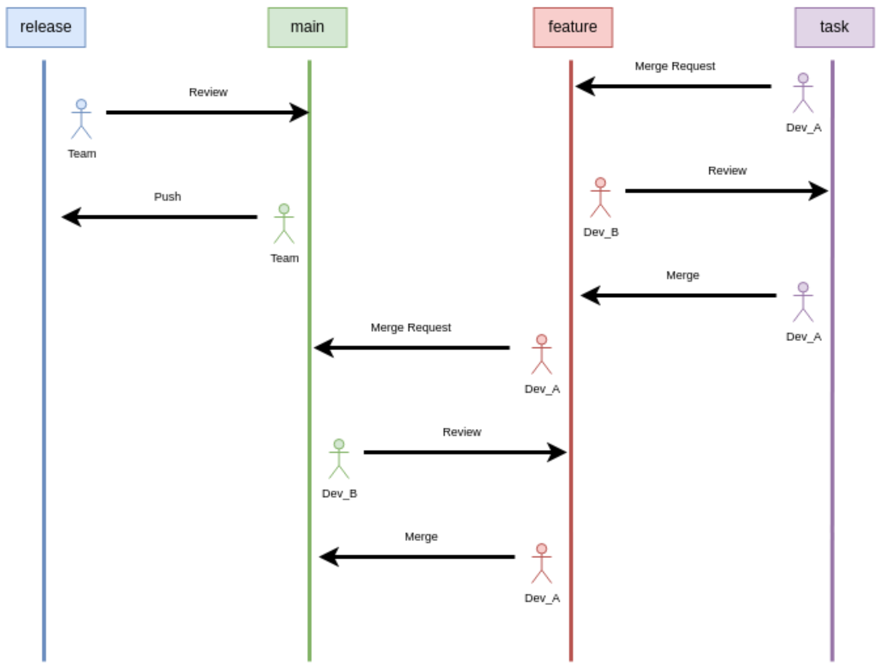


Figure 5.4.: Git Workflow

## 5.4. Test concept

### 5.4.1. Unit Testing

Unit tests are declared in the project (and namespace) "MediaHub.Test". xUnit is used as a test framework. Every feature that is subject to unit tests has its own class of unit tests. Since the unit tests only test the effective methodes itself and not use the databases, each feature which performs database-actions (read / write / update / delete) has a "mock" which mockes the database interactions. All unit tests are run upon committing to the repository. Branch coverage by unit tests is an important metric of this project, as it is defined in section 5.2 Quality Assessment Tools.

### 5.4.2. Integration Testing

Integration tests are also declared inside the "MediaHub.Test"-project an namespace. In comapre to the unit tests, the integration tests effectively use and perform actions on the databases to test, if all databse-actions performed by a feature are persistent and correct.

### 5.4.3. Usability Testing

Usability tests are defined in section 2.2 Non-functional Requirements (chapter 2 Requirements). The section 5.1 describes, when they are run. The test results for each release (alpha, beta, etc.) can be seen in subsection 2.2.2 NFR-Tests.

### 5.4.4. System / E2E Testing

System and End-To-End tests are defined with cypress. They are run manually by the team at every release. The tests can be found in the "nfr-tests"-folder inside the code-repository.

### 5.4.5. User tests

One set of user tests was performed in the alpha state of our product and did result in the following suggestions for improvement:

- Registration form is too complicated → no information about password requirements
- After email confirmation, there's no link to go back to the login page
- While adding a profile picture, the website freezes
- There's no information about how big the profile image may be
- There's no indication which contact I'm writing to in the chat
- messages are displayed only after page refreshing
- Chat-input field notifies it would be required like a normal HTML form
- The use of the user suggestion is not clear
- Why a password cannot exceed 100 characters
- Why the password isn't hashed immediately
- Contact-list items do not advertise that they're clickable
- After sending a message, it is nowhere shown that it actually got sent
- Trying to save more than 255 characters to the bio throws an exception and the page freezes

User tests of the beta-release did not yield any new problems.

### 5.4.6. Developer tests

The following are tests conducted by the team of developers at the beta state on 20.05.2022.

**Test agenda:**

1. Redeploy the application in a clean state (empty database)
2. All developers register and log in simultaneously
3. All developers set their name and a profile picture smaller than 2 MB simultaneously
4. All developers search for the movie ”Dune (2021)”
5. All developers rate ”Dune (2021)” with a rating greater than 7
6. All developers add the first 5 parts of the ”Harry Potter”-movies to their profiles simultaneously in order
7. All developers remove the first part of ”Harry Potter” from their profile simultaneously
8. All developers refresh their page and check if the rating is still correct
9. All developers add the movie ”Dune (2021)” to their profile
10. All developers refresh their page and check if the movie is still added to their profile
11. All developers add each other to their contacts
12. All developers check if other developers actions are visible on their feed
13. All developers chat with each other randomly
14. All developers block each others profiles
15. All developers check if their feed is empty
16. All developers delete their profile simultaneously

**Results:**

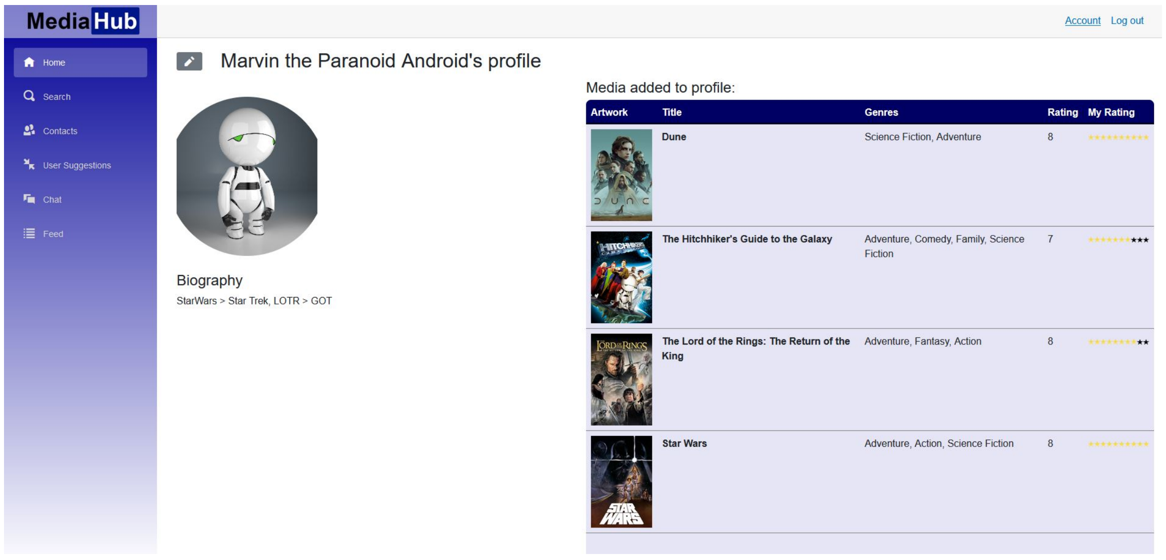
- Not all contacts and contact-requests are visible after simultaneously adding users as contacts
- Chat does not refresh itself continuously
- Users cannot block each other

These problems have been fixed until 28.05.2022.

# 6. Final Product

This chapter describes the final product, more precisely the individual pages of our application.

## 6.1. Home



Marvin the Paranoid Android's profile

Biography

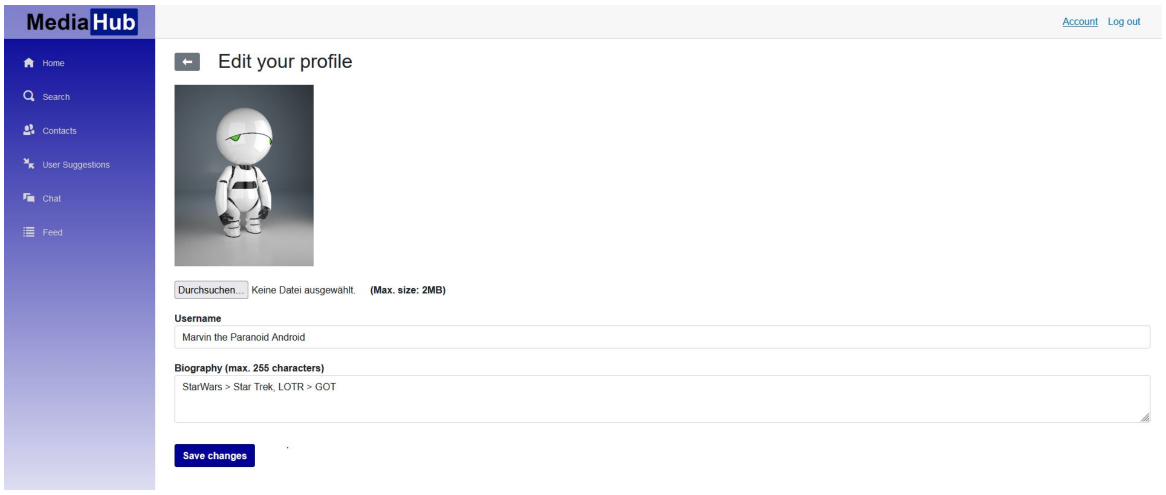
StarWars > Star Trek, LOTR > GOT

Media added to profile:

| Artwork | Title | Genres | Rating | My Rating |
|---|---|---|---|---|
| | Dune | Science Fiction, Adventure | 8 | ★★★★★★★★★★ |
| | The Hitchhiker's Guide to the Galaxy | Adventure, Comedy, Family, Science Fiction | 7 | ★★★★★★★☆☆☆ |
| | The Lord of the Rings: The Return of the King | Adventure, Fantasy, Action | 8 | ★★★★★★★★☆☆ |
| | Star Wars | Adventure, Action, Science Fiction | 8 | ★★★★★★★★★★ |

Figure 6.1.: Home screen / Profile overview

The 'Home' page of the website consists of an overview of the users profile. The left column includes the username, a button to edit the profile next to it, a profile picture and a biography. The right column consists of a table with all movies the user chose to add to his/her profile.



Edit your profile

Keine Datei ausgewählt. **(Max. size: 2MB)**

**Username**

Marvin the Paranoid Android

**Biography (max. 255 characters)**

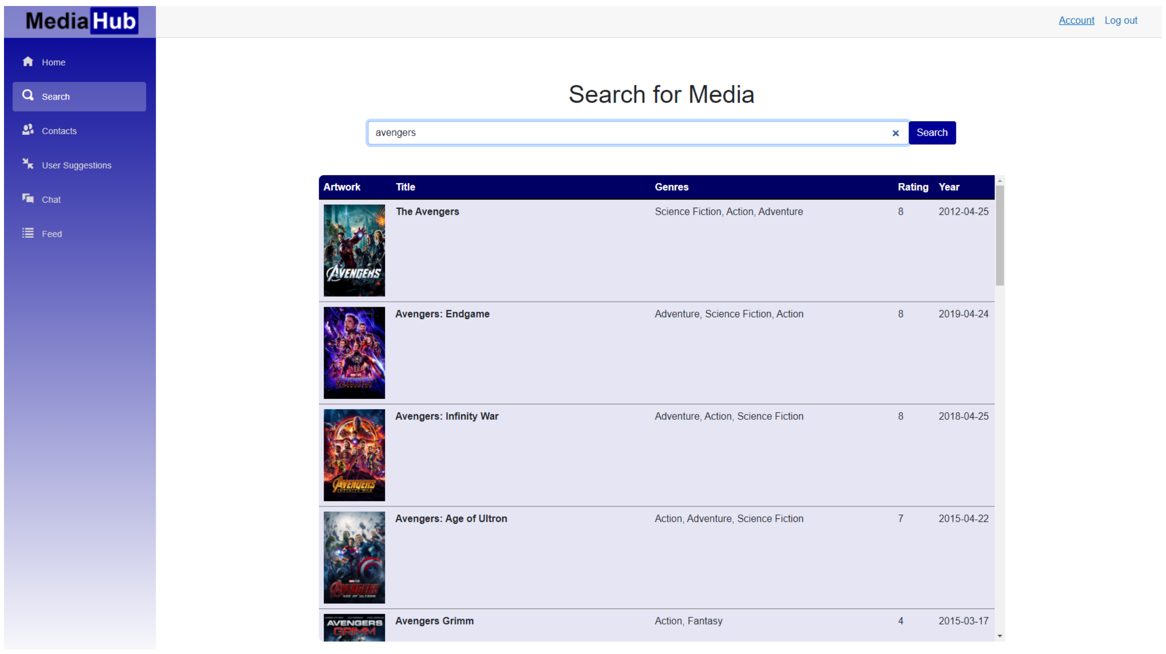
StarWars > Star Trek, LOTR > GOT

Figure 6.2.: Home screen / Edit profile

On the "edit profile" page the user has the option to change his/her profile picture, username and biography.

## 6.2. Search

To search for a movie, simply writing a name and the api searches for corresponding titles in its database.



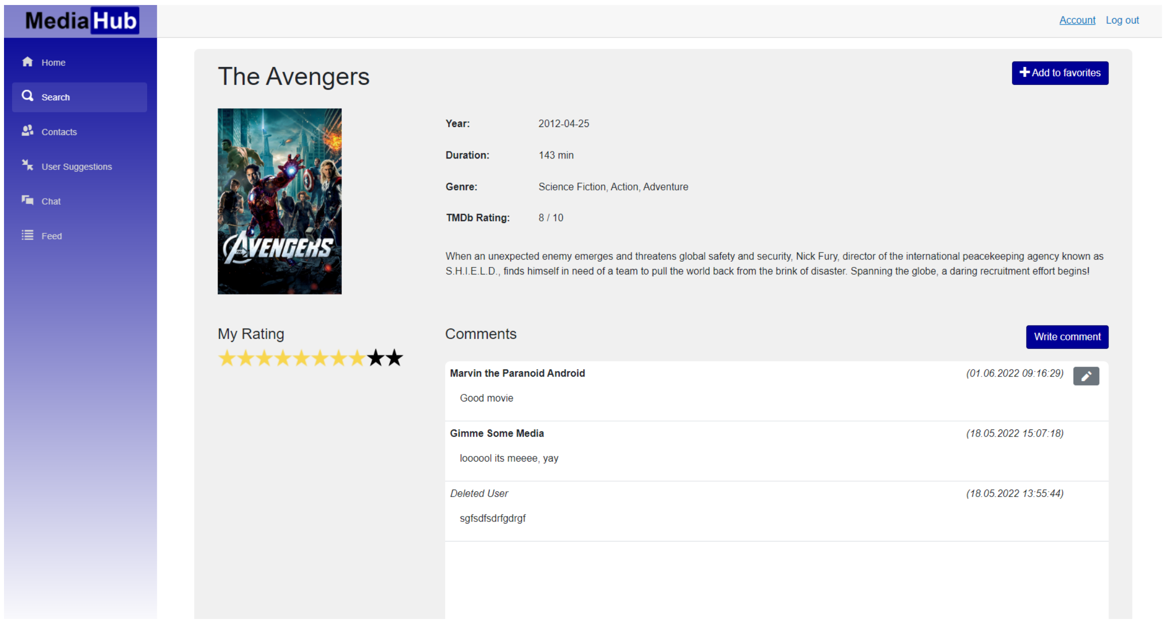
Search for Media

avengers

| Artwork | Title | Genres | Rating | Year |
|---|---|---|---|---|
| | The Avengers | Science Fiction, Action, Adventure | 8 | 2012-04-25 |
| | Avengers: Endgame | Adventure, Science Fiction, Action | 8 | 2019-04-24 |
| | Avengers: Infinity War | Adventure, Action, Science Fiction | 8 | 2018-04-25 |
| | Avengers: Age of Ultron | Action, Adventure, Science Fiction | 7 | 2015-04-22 |
| | Avengers Grimm | Action, Fantasy | 4 | 2015-03-17 |

Figure 6.3.: Search movie

To view a movies' information in detail, the movie detail page can be opened by clicking on the movies' artwork picture. Besides some details, there is also the possibility to add the movie to ones profile (button on the top right corner), rate the movie with stars from one to ten or to leave a comment in the comment section.



The Avengers

| | |
|---|---|
| Year: | 2012-04-25 |
| Duration: | 143 min |
| Genre: | Science Fiction, Action, Adventure |
| TMDb Rating: | 8 / 10 |

When an unexpected enemy emerges and threatens global safety and security, Nick Fury, director of the international peacekeeping agency known as S.H.I.E.L.D., finds himself in need of a team to pull the world back from the brink of disaster. Spanning the globe, a daring recruitment effort begins!

My Rating

★★★★★★★★★★

Comments

**Marvin the Paranoid Android** (01.06.2022 09:16:29)
Good movie

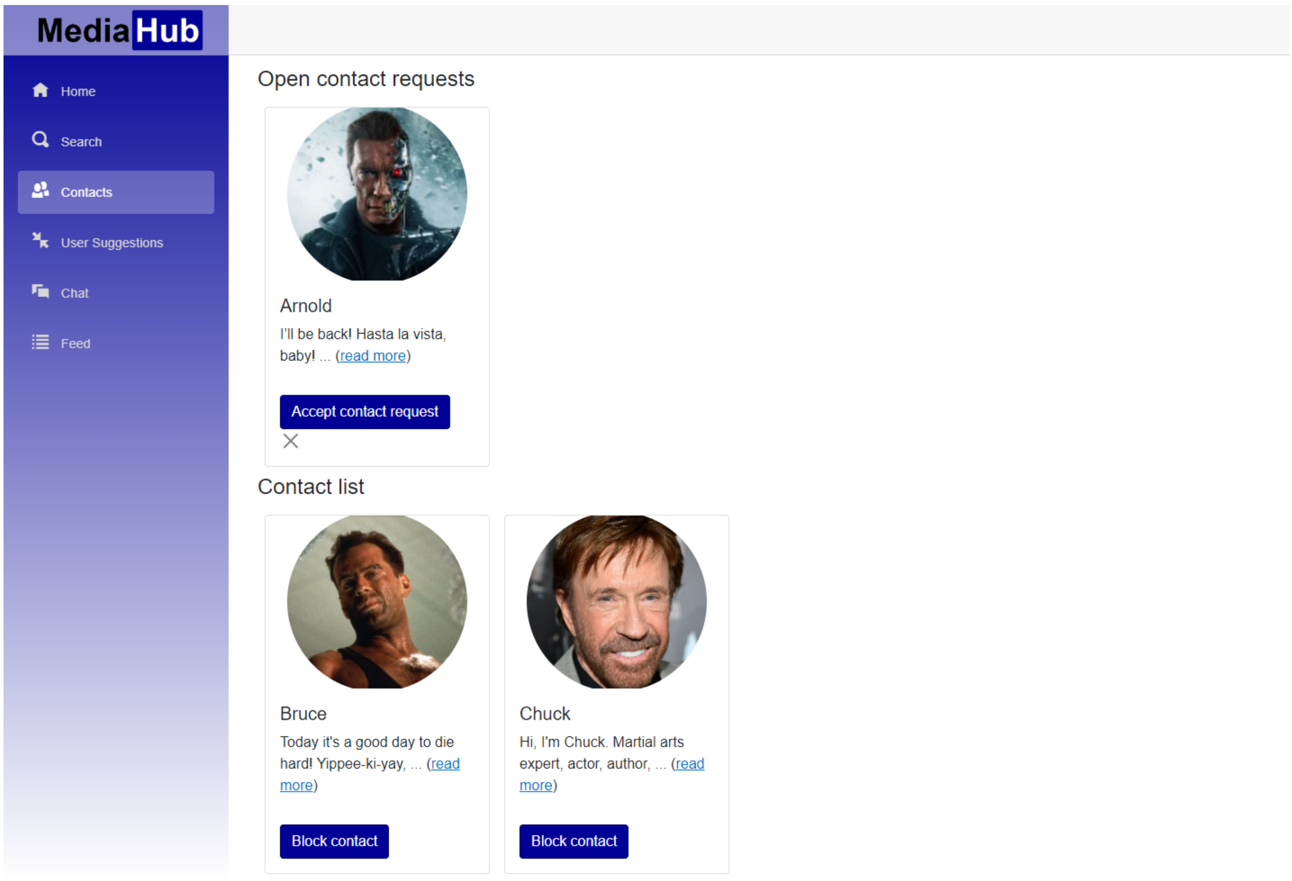
**Gimme Some Media** (18.05.2022 15:07:18)
loooool its meeee, yay

*Deleted User* (18.05.2022 13:55:44)
sgfsdfsdrfgdrgf

Figure 6.4.: Detail view of movie

## 6.3. Contacts

The contacts and contact requests can be managed on this page. When a contact request is sent, it appears on the contact page of the other user. He then can accept or decline the request. If a request is accepted, it will be shown at the contact list of both users. Otherwise, when the request is declined it disappears. Users can also be blocked through the contact list. There is currently no option to unblock contacts or to display a list of all blocked contacts.



Open contact requests

Arnold
I'll be back! Hasta la vista, baby! ... (read more)

Contact list

Bruce
Today it's a good day to die hard! Yippee-ki-yay, ... (read more)

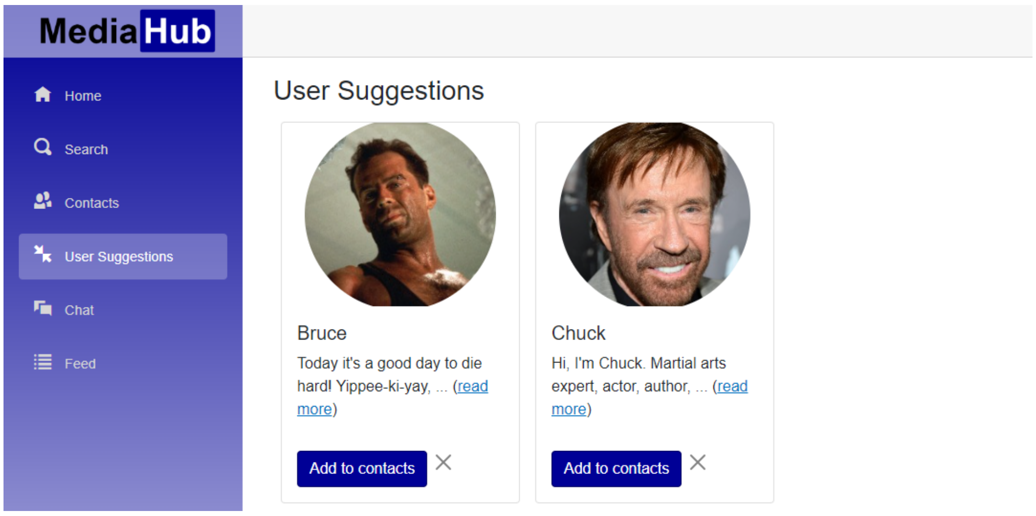
Chuck
Hi, I'm Chuck. Martial arts expert, actor, author, ... (read more)

Figure 6.5.: Contacts page

## 6.4. User Suggestions

All possibly intressting users found (according to the process in section 3.2) are displayed on the user suggestions page. Always both users concerned get this suggestion. A user can do the following interactions:

- Read more: Go to the suggested user profile
- Add to contacts: Send a contact request. The other user has to confirm the request before the two people are contacts
- X: Ignore the suggestion



User Suggestions

Bruce

Today it's a good day to die hard! Yippee-ki-yay, ... (read more)

Chuck

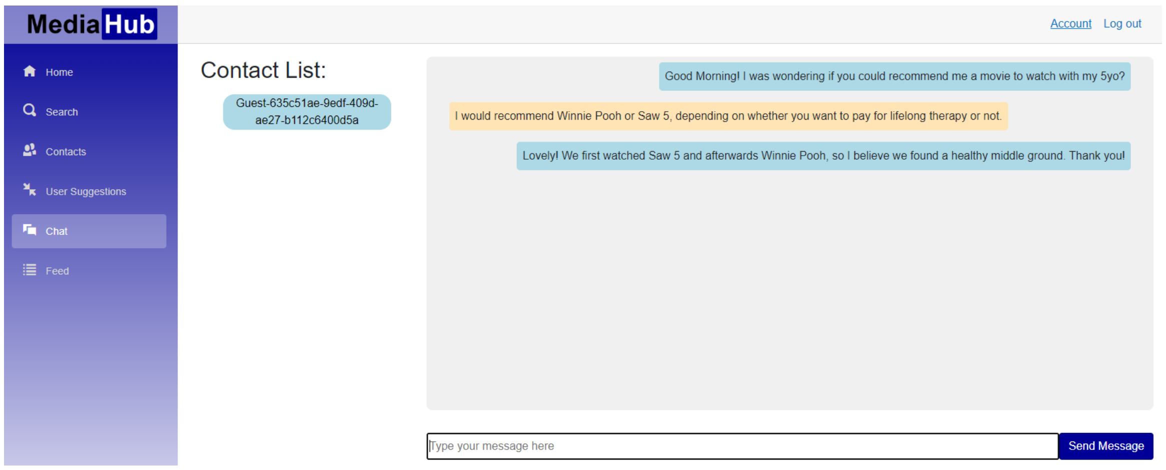
Hi, I'm Chuck. Martial arts expert, actor, author, ... (read more)

Figure 6.6.: User Suggestions page

## 6.5. Chat

On the chat page the user can chat with other users. A user can only contact his own contacts, not all users. If a user does not have any contacts, the contact list will be empty and a placeholder will inform the user that there are currently no contacts.

Implementationwise new messages are fetched by polling the database every 2 seconds or whenever the user sends a new message.
The window keeps scrolling to the bottom whenever new messages appear so the user sees them.



Contact List:

Guest-635c51ae-9edf-409d-ae27-b112c6400d5a

Good Morning! I was wondering if you could recommend me a movie to watch with my 5yo?

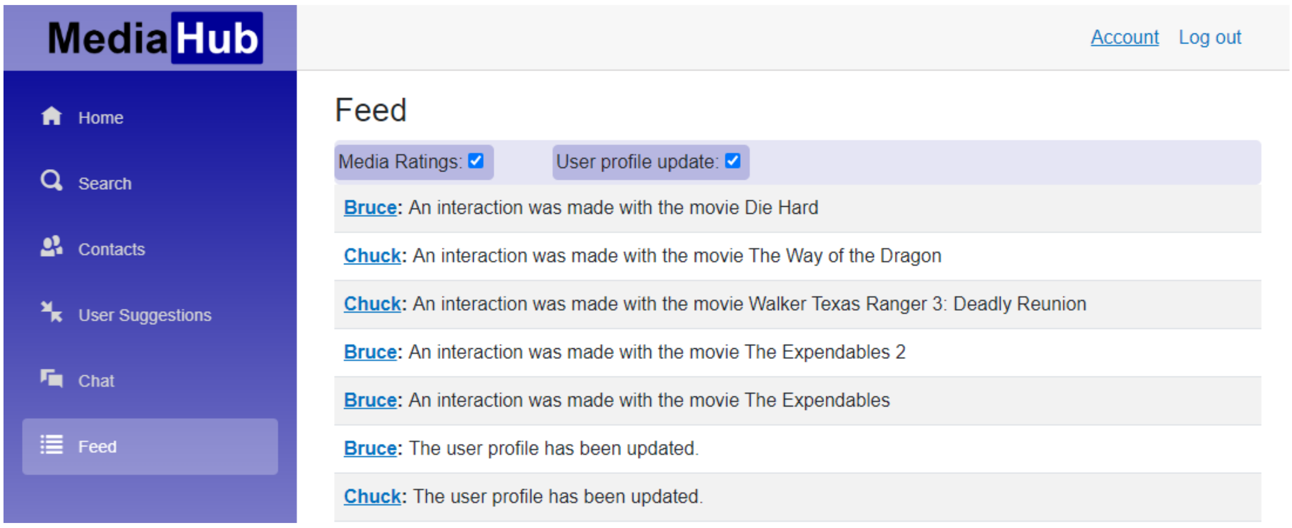
I would recommend Winnie Pooh or Saw 5, depending on whether you want to pay for lifelong therapy or not.

Lovely! We first watched Saw 5 and afterwards Winnie Pooh, so I believe we found a healthy middle ground. Thank you!

Figure 6.7.: Chat page

## 6.6. Feed

The activity of a users contacts are shown on the feed page. This includes the user profile updates and interactions with media items. With the filter, individual categories can be shown and hidden. By clicking on the user name, you will be redirected to the corresponding user profile.



Feed

Media Ratings: ✓ User profile update: ✓

Bruce: An interaction was made with the movie Die Hard
Chuck: An interaction was made with the movie The Way of the Dragon
Chuck: An interaction was made with the movie Walker Texas Ranger 3: Deadly Reunion
Bruce: An interaction was made with the movie The Expendables 2
Bruce: An interaction was made with the movie The Expendables
Bruce: The user profile has been updated.
Chuck: The user profile has been updated.

Figure 6.8.: Feed page

In the future the feed could be expanded as follows

- More details: Display which parts of the user profile were updated (e.g. profile picture) and display what kind of interaction was made with the media item (e.g. added to profile)
- More categories: Show new contacts as well on the feed page
- Graphical improvement: Additionally display the profile picture
- Item linking: Add a link to the movie which the user interacted with

# Part III.

# Project Documentation

# 7. Initial Project Proposal

**Project name:** MediaHub

### Team Members

1. Adrian Locher (adrian.locher@ost.ch)
2. Benjamin Kern (benjamin.kern@ost.ch)
3. Dominik Ehrle (dominik.ehrle@ost.ch)
4. Jason Benz (jason.benz@ost.ch)
5. Marco Agostini (marco.agostini@ost.ch)

### Availabilities

| Time slot | Mon | Tue | Wed | Thu | Fri |
|---|---|---|---|---|---|
| 08h00-09h00 | XO | - | - | - | - |
| 09h00-10h00 | XO | - | - | - | - |
| 10h00-11h00 | XO | - | - | - | - |
| 11h00-12h00 | XO | - | - | - | - |
| 12h00-13h00 | - | - | - | - | - |
| 13h00-14h00 | - | - | - | - | - |
| 14h00-15h00 | - | - | - | - | - |
| 15h00-16h00 | - | - | - | (XO) | - |
| 16h00-17h00 | - | - | - | - | - |
| 17h00-18h00 | - | - | - | - | - |
| 18h00-19h00 | - | - | - | - | - |

### Project Idea

The goal of this project is to create a modern platform to connect people based on their interest on different entertainment media (e.g. movies). The users can create a profile and get suggestions for different people who have similar media preferences and might be open for discussions. Ideally, a chat functionality provides this aforementioned discourse opportunity to the users.

Possible future extensions are the integration of additional media types or additional social media like features (e.g. liking, sharing, review, forming chat groups, etc.).

### Proposed Realisation

The product will be implemented as a web app using .NET technologies (ASP.NET core). A database, a C# backend and a frontend will be provided. The further technical details (architecture, database selection, database mapper, etc.) will be determined in the planning phase of the project.

# 8. Project Plan

## 8.1. Processes, Meetings and Roles

### 8.1.1. Processes

- Scrum+ (Scrum + RUP)

### 8.1.2. Recurring Meetings

| Meeting type | Occurrence | Time | Duration | Start Date |
|---|---|---|---|---|
| Weekly sync | Every odd semesterweek on Monday | 09.00 am | 15 min | 21.02.2022 |
| Sprint review | Every even semesterweek on Monday | 08.30 am | 30 min | 07.03.2022 |
| Sprint retrospective | Every even semesterweek on Monday | 09.00 am | 60 min | 07.03.2022 |
| Advisor review | Every even semesterweek on Monday | 10.10 am | 45 min | 07.03.2022 |
| Sprint planning | Every even semesterweek on Monday | 11.00 am | 60 min | 21.02.2022 |

### 8.1.3. Review Meetings

| No. | Title | Date |
|---|---|---|
| R1 | Project plan | 07.03.2022 |
| R2 | Requirements | 21.03.2022 |
| R3 | End of elaboration & arch prototype | 04.04.2022 |
| R4 | Quality | 25.04.2022 |
| R5 | Architecture | 16.05.2022 |
| R6 | Project presentation | 30.05.2022 |

### 8.1.4. Technical Roles

- Architect / Integration manager: Adrian Locher
- DevOps supervisor: Marco Agostini
- Front-End supervisor: Dominik Ehrle
- Back-End supervisor: Adrian Locher & Jason Benz
- Data-Base supervisor: Benjamin Kern
- Security supervisor: Marco Agostini
- Testing supervisor: Per division

### 8.1.5. Administrative Roles

- Scrum master / project manager (overall overview): Jason Benz
- Product owner (backlog prioritization): Benjamin Kern
- Minute taker: Round robin

## 8.2. Phases, Iterations and Milestones

### 8.2.1. Milestones

| No. | Milestone | Goal | Due date |
|---|---|---|---|
| M0 | Project kickoff | The project is started successfully with a first meeting to define approaches, tools etc. | 21.02.2022 |
| M1 | Project Plan | The project plan is completed. Small adjustments in the future are still possible. | 13.03.2022 |
| M2 | Requirements | All known requirements are captured in the form of epics in the Gitlab backlog. They are categorized in 'Functional Requirements' (must / should / can) and 'Non-functional Requirements'. | 20.03.2022 |
| M3 | End of elaboration & arch prototype | The elaboration phase is completed and an architecture prototype (high level overview of the system architecture) exists. | 03.04.2022 |
| M4 | Architecture | The architecture is defined and in a stable condition. | 24.04.2022 |
| M5 | Alpha | An alpha version - architecture and basics functions (must requirements) are provided - of the MediaHub is released. | 01.05.2022 |
| M6 | Beta | A beta version - most functions are provided in a stable version - of the MediaHub is released. | 19.05.2022 |
| M7 | Final submission is released | The final MediaHub version with the full range of functions (according to plan) is released. | 01.06.2022 |

### 8.2.2. Phases

| Phase | Associated milestones | Duration |
|---|---|---|
| Inception | M0 | 1 week |
| Elaboration | M1, M2, M3 | 5 weeks |
| Construction | M4, M5, M6 | 7 weeks |
| Transition | M7 | 2 weeks |

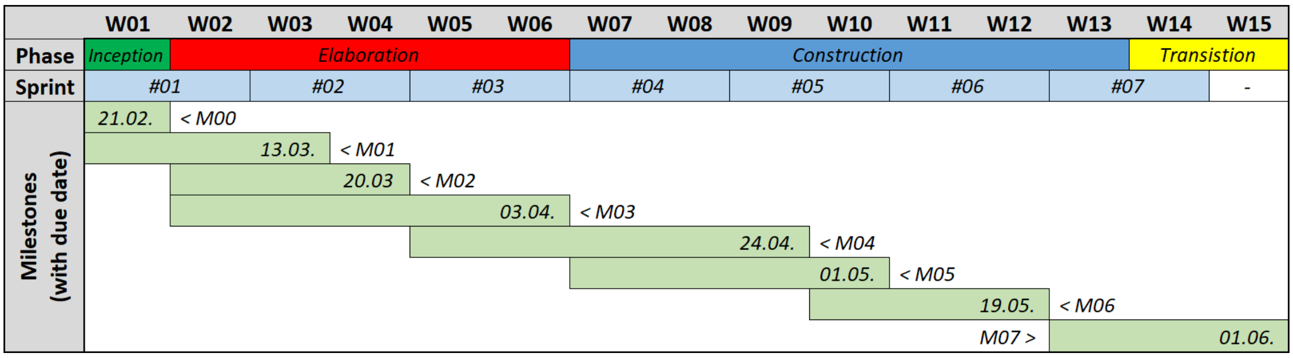
### 8.2.3. Iterations

Each sprint takes 2 weeks.
Sprint 01 is not a real sprint, since the tools, procedures etc. are introduced and established. It is listed as a Sprint because there is an output (project plan). Since the team has never worked together before and there are only 7 sprints, the project does not use reference stories and relative time estimation. This means that the work items are estimated in hours and days. Furthermore the whole project is timeboxed by 120 hours per person. Therefore, it makes even more sense to estimate in time rather than story points.

#### Sprint time table:

- **Sprint 01**: 21.02. - 07.03.2022
- **Sprint 02**: 07.03. - 21.03.2022
- **Sprint 03**: 21.03. - 04.04.2022
- **Sprint 04**: 04.04. - 18.04.2022
- **Sprint 05**: 18.04. - 02.05.2022
- **Sprint 06**: 02.05. - 16.05.2022
- **Sprint 07**: 16.05. - 30.05.2022

### 8.2.4. Big picture with rough estimates



| | W01 | W02 | W03 | W04 | W05 | W06 | W07 | W08 | W09 | W10 | W11 | W12 | W13 | W14 | W15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Phase | Inception | Elaboration | | | | | Construction | | | | | | | Transistion | |
| Sprint | #01 | | #02 | | #03 | | #04 | | #05 | | #06 | | #07 | | - |
| Milestones (with due date) | 21.02. | < M00 | | | | | | | | | | | | | |
| | | | 13.03. | < M01 | | | | | | | | | | | |
| | | | | 20.03 | < M02 | | | | | | | | | | |
| | | | | | | 03.04. | < M03 | | | | | | | | |
| | | | | | | | | | 24.04. | < M04 | | | | | |
| | | | | | | | | | | 01.05. | < M05 | | | | |
| | | | | | | | | | | | | 19.05. | < M06 | | |
| | | | | | | | | | | | | M07 > | | | 01.06. |

Figure 8.1.: Big Picture of Project Plan

### 8.2.5. Long Term Plan Refinement

**Overview:**

- **Capacity**: 392 h
- **Performed work (sprint 1-3)**: 168 h
- **Finalize MVP**: 110 h
- **Other requirements**: 176 h

#### Capacity

The team is completing 7 sprints which last two weeks each. Without the sprint and advisor meeting overhead, 59 - 65 hours of project-work are available per sprint. The effective development capacity is 56 h per sprint. A small reserve is included to cover unexpected events (sick team member, infrastructure failure, etc.).
Therefore, the total project development capacity is 392 hours.

#### Performed Work

In retrospect of the first 2.5 sprints, the capacity estimation of 56 hours of development capacity per sprint was very realistic. Thus, we will continue to use this approximation and set the performed work (also with the ongoing sprint at the time of this analysis) to 3x 56 h. This results in 168 hours of work on the tasks.

#### MVP

At the time of this analysis, sprint 03 is in progress. From sprint 4 onwards, approximately 110 h are required for the finalization of the MVP (all 'must' requirements). Since certain dependencies exist, it may be that this time specification will change slightly. This means 2 more sprints are required to finish the MVP.
Therefore, milestone M5 (alpha version with all must requirements) at the end of sprint 05 is a realistic goal. The long term plan doesn't need to be changed.

#### Other Requirements

After completing the MVP, 114 hours of capacity are still available. All 'should' and 'could' requirements will need an estimated effort of 176 hours. However, this estimate is rather inaccurate, since according to the agile approach many of these tasks have to be refined at a given time. Therefore it will be possible to complete around 60% of all non-'must' requirements.
The product owner defined a prioritization on Gitlab and communicated it to the team. On one hand, there are more important features like some additional chat functionalities and two factor authentication. On the other hand, there are some requirements with a very low priority such as sending broadcasts.
Which additional features are going to be implemented will be defined in the ongoing agile process. The customer can expect approximately 80 - 100 hours of performed work on 'should' and 'could' requirements according to the prioritization list, which is not final and will be refined further.
Milestone M6 (beta version) will include about 56 hours of performed work on non-'must' requirements. The last sprint will be used to test and improve the software.

### 8.2.6. Short Term Plan / Work Items

A detail plan can be found on Gitlab. All last and current sprints (sprint 03) are tracked there and some tasks for the next sprint are prepared / refined.

## 8.3. Risk Management

### 8.3.1. Risks

| No. | Risk | Mitigation |
|---|---|---|
| R01 | Illness of Teammember | Each member assigns a deputy for his field of supervision |
| R02 | Permanent loss of a teammember through sickness or termination of studies | Each member assigns a deputy for his field of supervision. Trust-agreement, that a terminating member will finish his work on the Project |
| R03 | Dispute between teammebers | Voting by the members not involved or - if all involved - by majority |
| R04 | Miscommunication in the seperation of duties (Per task) | Our tools, meetings and meeting protocols |
| R05 | Change of workstyle to homeoffice | Unnecessary to mitigate |
| R06 | Temporary loss of the code- and document repositories | Accepted |
| R07 | Permament loss of code and documents | Mirroring of all data to an external NAS appliance |
| R08 | Temporary loss of host-server of the application | Accepted |
| R09 | Loss of data and configuration of the host-server | Building-plan and test-data are held redundant |
| R10 | Loss of workstation of a teammember | Every teammember has either a second workstation or a concept of aquiring one the same day |
| R11 | Wrong assessment of complexity of the task | Additional reviews with an external stakeholder |
| R12 | Unfeasability of the task with the chosen technologies | Clarifications ahead of start |
| R13 | External APIs change unexpectedly | Abstraction of all APIs for fast switching |
| R14 | Non-comply with deadlines | Time tracking and observation of burndown chart |
| R15 | Non-finalization of project | Use of agile methodologies with Scrum (Each increment adds to a working product) |

### 8.3.2. Risk cost analysis

| No. | Measure costs | Max loss | Probability | Weighted loss | Priority | Residual |
|---|---|---|---|---|---|---|
| R01 | 0.5h | 4h | 30% | 1.2h | Low | 10% |
| R02 | 0.5h | 60h | 1% | 6h | Medium | 0.5% |
| R03 | 0.5h | 20h | 5% * | 1h | Low | 5h |
| R04 | 0h | 5h | 80% * | 4h | Medium | 20% |
| R07 | 0h | 600h | 1% | 60h | High | 0.1% |
| R09 | 0.5h | 10h | 3% | 0.3h | Low | 2h |
| R10 | 0.5h | 10h | 30% | 0.3h | Low | 2h |
| R11 | 1h | 30h | 5% | 1.5h | Low | 2% |
| R12 | 1h | 300h | 10% | 30h | Hight | 1% |
| R13 | 3h | 40h | 20% | 8h | Medium | 10h |
| R14 | 2h | 20h | 15% | 3h | Low | 10% |
| R15 | 0h | 600h | 5% | 30h | High | 0.001% |

∗ Probability per work item, while the default is the probability regarding the whole project.

### 8.3.3. Risk Matrix

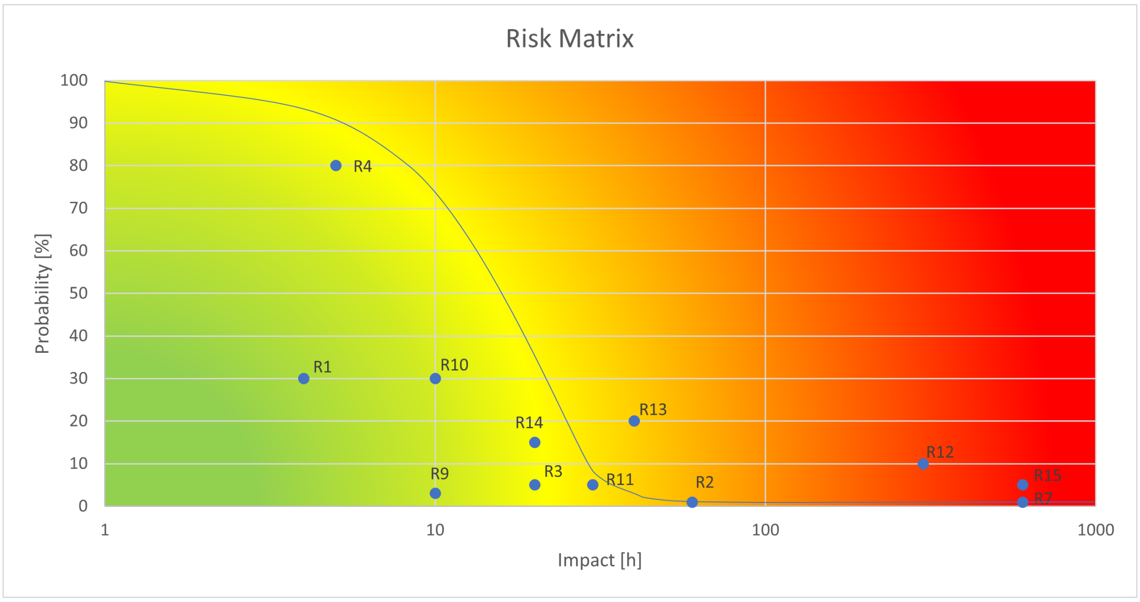


Figure 8.2.: Risk Matrix before Mitigations

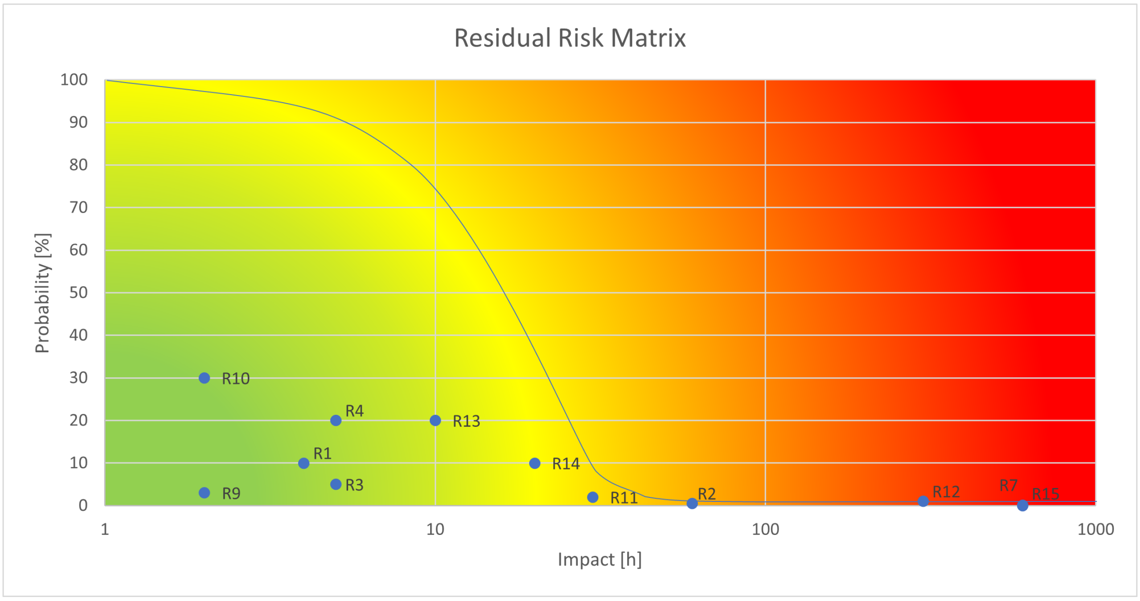


Figure 8.3.: Risk Matrix after Mitigations

### 8.3.4. Consequences of Risk Assessment

The evaluated risks impact the architecture of the platform (i.e. chosen levels of abstractions) where applicable, which subsequently affects the estimates of those work items (more abstraction = more work). Other risks can only be accepted and cannot be incorporated into long-term planning, such as sickness of teammates.

### 8.3.5. Occured Risks

In sprint 02 the risk R01 - Illness of Teammember occured. Jason was very sick and not able to work for one week. This meant a loss of time for the project of about 8 hours. His important tasks were partially done by other teammembers and by himself in the second week of the sprint. Due to the fact that some tasks took less time than planned, this risk occurence had no impact on the overall project.

## 8.4. Planning Tools

- Gitlab is used for everything: Time tracking, issue tracking and milestones
- Milestones are tracked in the code repository
- Everything is handled in english, except for direct communication between the teammates
- Style guide for code: Microsoft C# Coding conventions

# 9. Time Tracking Report

This project is time tracked on Gitlab. The duration of the project is 14.5 Weeks. In order to achieve the specified workload, each team member is required to work 8 - 8.5 hours per week on the project. Theoretically, that would be a total of 80 - 86 hours per sprint. However, there is a certain overhead due to sprint and reviews with the project advisor. Therefore, our planned and recorded time will be 59 - 65 hours per sprint.

**Time overhead per sprint:**

| Interval | Subject | Time/meeting | Time/sprint |
|---|---|---:|---:|
| Weekly | Weekly sync | 00:15h | 02:30h |
| Weekly | Issues capturing and refinement | 00:45h | 03:45h |
| 2nd week | Sprint review | 00:30h | 02:30h |
| 2nd week | Sprint retrospective | 01:00h | 05:00h |
| 2nd week | Sprint planning | 01:00h | 05:00h |
| 2nd week | Advisor review | 00:45h | 03:45h |
| **Total** | | | **22:30h** |

## 9.1. Time Tracking

An estimated time and the executed work time get assigned and tracked to each issue (= task) during the sprint plannings and the following sprint, allowing a consistent time tracking per sprint.

### 9.1.1. Time Tracking Overview

| Sprint | Planned | Tracked | Diff |
|---|---:|---:|---:|
| 01 | 54:15h | 56:30h | + 02:15h |
| 02 | 68:00h | 64:00h | - 04:00h |
| 03 | 84:15h | 86:25h | + 02:10h |
| 04 | 63:00h | 57:20h | - 05:40h |
| 05 | 64:30h | 70:40 | + 06:10h |
| 06 | 58:00h | 57:30h | - 00:30h |
| 07 | 52:30h | 49:30h | - 03:00h |
| **Total** | **444:30h** | **441:55h** | **-02:35h** |

### 9.1.2. Time per person

| Sprint | Adrian | Benjamin | Dominik | Jason | Marco |
|---|---:|---:|---:|---:|---:|
| 01 | 10:30h | 13:45h | 10:15h | 11:45h | 10:15h |
| 02 | 15:55h | 10:30h | 13:55h | 08:30h | 15:10h |
| 03 | 19:00h | 12:05h | 17:05h | 22:10h | 16:05h |
| 04 | 13:00h | 13:20h | 13:00h | 11:30h | 06:30h |
| 05 | 12:30h | 21:05h | 13:30h | 13:05h | 10:30h |
| 06 | 10:00h | 03:00h | 15:00h | 17:00h | 12:30h |
| 07 | 05:45h | 17:00h | 07:45h | 06:30h | 12:30h |
| **Total** | **86:40h** | **90:45h** | **90:30h** | **90:30h** | **83:30h** |

## 9.2. Evaluation

All team members spent about the same amount of time on the project. The total project working time is composed as follows:

| **Description** | **Amount** | **Time/meeting** | **Total in project** |
|---|---:|---:|---:|
| Weekly sync | 14x5 | 00:15h | 17:30h |
| Sprint refinement | 5x5 | 00:45h | 18:45h |
| Sprint planning | 7x5 | 01:00h | 35:00h |
| Sprint review | 7x5 | 00:30h | 17:30h |
| Retrospective | 7x5 | 01:00h | 35:00h |
| Advisor review | 5x5 | 00:45h | 18:45h |
| Sprint work | | indv. | 441:55h |
| **Total** | | | **584:25h** |

The required working hours for the module are 30h * 4ECTS * 5 members = 600h. In addition to our tracked hours (584:25h) we spent about 20h on the preparation of the presentation.
We achieved the time box pretty well because we did a time evaluation after each sprint. Then we could dynamically decide whether to work on more or less issues in the next sprint.

### 9.2.1. Burndown

The burndown contains all product specific issues. You can recognize the burndown relevant issues by the labels 'must', 'should' and 'could' on Gitlab. The issue estimations were done by 'ideal hours' (alternative to story points).
The MVP line is reached as expected in time in sprint 05. There is some remaining work after the last sprint. This is an expected state, since the team was aware at the project planning that not all 'could' issues are going to be done.

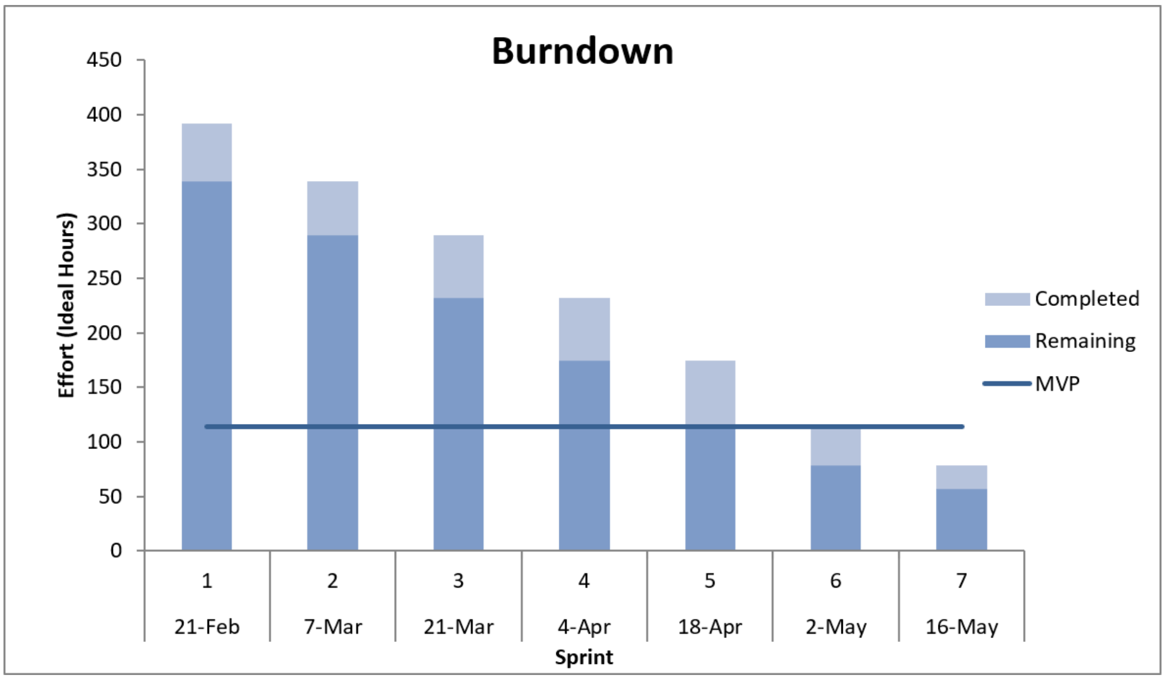


Figure 9.1.: Burndown

### 9.2.2. Velocity

The velocity (estimated time / tracked time) fluctuates in each sprint. The reason for this is that we had relatively little working time per sprint (only two days per person per sprint). Therefore, a small deviation in the estimate is already significant. On average, we had a velocity of 1.03. Thus, our estimates were relatively good overall. In the future, we would plan longer sprints with this amount of work to reduce the overhead and make a more consistent estimate.

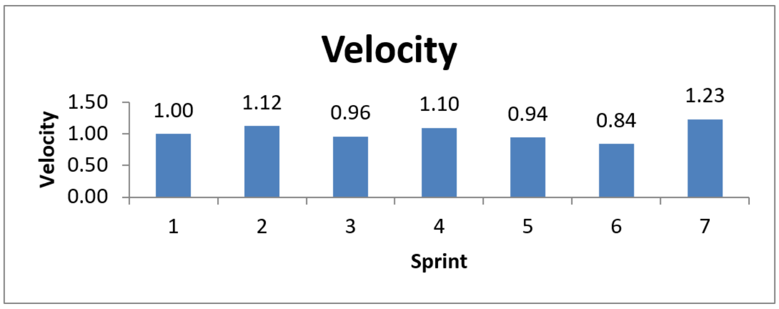


Figure 9.2.: Velocity

# 10. Personal Reports

At the end of the last sprint, a project retrospective was conducted. There the whole project was reflected as a team. The details can be found in the corresponding meeting minutes (see section 11.19).

## 10.1. Adrian Locher

**Positives:** Our team in this project worked very well with each other. Critique was always welcomed and acted upon.
Splitting our work could not have been easier, since we all had our own fields of expertise. There was never a task that was left undone, because no one could or wanted to do it.

**Improvement opportunities:** We could still improve, if we would have accepted when we weren't able to make a feature working in time and ask for help. Asking for help happened rather seldom.
In the beginning we also spent a lot of time discussing everything, instead of delegating tasks. This could have been improved especially in the early stages of the project.

**Highlights:** My absolute highlight was, seeing how after a few sprints everything started to come together. All of a sudden a working piece of software emerged from independently programmed pieces.

## 10.2. Benjamin Kern

**Positives:** Overall I believe the organization and synchronization with the team was nearly optimal. Tasks were distributed such that dependencies were largely mitigated. The workload was distributed evenly across teammembers.

**Improvement opportunities:** I believe sprints should be longer than 2 weeks, considering that we only have 1 workday per week that we work on the SE project. Essentially a sprint lasted only 2 days, which is way too short. When working with a new framework and to some degree unfamiliar technologies some minor bug or unforeseen technical limitation can easily cost half or even a whole workday (so 25-50% of the entire time allocated for the sprint). This made it difficult (and sometimes impossible) to hit deadlines while also ensuring that tasks fulfill the DoD. Writing tests, refactoring and merging require a lot of time that should not be underestimated (particularly if the merge requests require multiple teammembers to communicate).

**Highlights:** My highlights mostly coincide with the learning opportunities I encountered. I got familiar with ASP.NET, Blazor, the dotnet entity framework and C# in general, as well as writing code using the MVVM model. Mostly I learned how unpredictable time estimations of unfamiliar technologies can be. On a more general note: The overhead of starting up a project should never be underestimated. A great deal of planning is required to develop software with multiple people that has high cohesion and low coupling. Scaling vertically (adding more people to the team) is much more costly than anticipated: Many questions such as how should the workflow of branches take place and how will we mitigate dependencies between tasks that people work on simultaneously are questions that simply do not appear or at the very least are optional considerations when working alone.

## 10.3. Dominik Ehrle

**Positives:** The teamwork throughout this project was great. Discussions were on a respectfull and collegial basis. Constructive criticism could be given and also got accepted, but a lot of positive feedback and appreciation was also given.
The assignment of issues worked very well, due to the fact that every team member wanted to get involved in the project and contribute as much as the others do. Thus, issues were mostly completed on time, leading to a recognizable progress throughout the whole project what was quite motivating.

**Improvement opportunities:** What could be improved is the asking for help. If one team member got stuck on a particular task, most often it took rather long until he asked for help and therefore some issues took longer than expected. Following on that, espacially during the project start, issues were worked on rather late and just before the due date, what made it difficult to help if necessary or do a good review of the work done.

**Highlights:** The highlight for me was seeing the first, running product improve over the weeks with new features and designs beeing implemented. We grew alongside our product, improving our skills in coding, communicating and working as a team and were able to apply what we have learned in practice.

## 10.4. Jason Benz

**Positives:** We were allowed to build the team ourselves. This meant we could work with people who were nice to work with. On the whole, the collaboration went very well. Everyone knew his area of responsibility in the team and did his job. You could also see the progress of the project from sprint to sprint and how the application was evolving, what was very motivating.

**Improvement opportunities:** GitLab is a great tool for source code and pipeline management. Personally, I didn't like the tool as a kanban board and to manage issues with different levels (epic, feature, user story, task and milestones as well). In the future, I would choose a different tool for the management part (e.g. Azure DevOps).
The overhead for the sprint meetings was quite large compared to the time spent on coding. In another project with this development capacity, I would run longer sprints. I believe three week sprints would have been ideal for us.

**Highlights:** Before my time at OST, I was already working as a software developer. In the company, we were developing an existing software solution, which had a lot of legacy issues. Therefore, the highlight for me was definitely that we were able to create a larger program completely from scratch. This also included the fact that we were allowed to create the planning and architecture ourselves.

## 10.5. Marco Agostini

**Positives:** The project for me is an overall success on mulitple levels. For sure we had points where we had issues, but the team managed to work together to accomplish even difficult task in the proposed time frame. The most positive part for me was the learning from other colleagues. This is the first real software project for me, and I was able to learn a lot from my peers. And because everybody has a different point of focus in the area of software engineering, there is always one peer who knows more than you in the area you are struggling in.

**Improvement opportunities:** As aforementioned we always found a solution for every problem within the team. But one thing to improve for myself is the time i use before i ask my peers. I always wanted to solve the problem by myself first, and from time to time wasted a lot of time with a topic, another peer already would have known the solution for.

**Highlights:** I really enjoyed creating working software in a team. The amount of joy you feel, when you can deploy the application for the first time is indescribable. And also, the help of the peers was one thing I really appreciated.

# 11. Meeting Minutes

## 11.1. Kickoff meeting

| | |
|---|---|
| **Meeting date/time:** | 21.02.2022, 09:00 |
| **Meeting purpose:** | Discussing the initialization of the project |
| **Notetaker:** | Jason Benz |

### Attendees:

- Adrian Locher
- Benjamin Kern
- Dominik Ehrle
- Jason Benz
- Marco Agostini

### Agenda

| Item | Description |
|---|---|
| Project Planning | What methods to use |
| Infrastructure | Where to deploy the application |
| Version Control | Discussing branching strategies |
| Time Tracking | How to track the time used for the project |
| Language | What language to use in the Project |

### Discussion

| Item | Who | Notes |
|---|---|---|
| Project Planning | Everyone | We work according to ”Scrum+” (Scrum + RUP). There will be a minimum of one milestone per sprint. A sprint is a two week long working phase and contains all work-items, that are done between the meetings with our supervisor. All work-items will be recorded as an Issue in gitlab. |
| Infrastructure | Everyone | For a hosting server we will request a server in the ost server infrastructure. What operating system to use will be decided by 28.02.2022. |
| Version Control | Everyone | There will be one production branch which is also the main branch. There will be a branch for each feature, where one feature consists of an amount of work-items which in return are recorded as issues. |
| Time tracking | Everyone | Time tracking will be done as comments in the issues in gitlab Commands to use: /estimate <time>/spend <time> |
| Lanugage | Everyone | Meetings and Reviews are held in german. Reports are written in english. |

### Additional notes

- Todos for everyone:
  - Install VisualStudio or JetBrains Rider
  - Integrate the IDE with our gitlab repository
  - Get familiar with time-tracking, milestones, etc. on our test-repository
  - Work through the official Microsoft-Blazor tutorial
  - Think of requirements for the project

## 11.2. Finalizing Inception & Starting Elaboration

| | |
|---|---|
| **Meeting date/time:** | 28.02.2022, 09:00 |
| **Meeting purpose:** | Deciding on requirements and roles & starting the ellaboration phase |
| **Notetaker:** | Adrian Locher |

**Attendees:**

- Adrian Locher
- Benjamin Kern
- Dominik Ehrle
- Jason Benz
- Marco Agostini

**Agenda**

| Item | Description |
|---|---|
| Requirements | Deciding on Requirements, all team members have thought of by today, and fixate them in our planning |
| Roles | Choosing and assigning roles to each team member |
| Meeting schedule | Selecting what meetings to hold, and when to hold them |
| Longterm Planning | Defining phases and milestones |
| Shortterm Planning | Defining what is to be done for the next two weeks |

**Discussion**

| Item | Who | Notes |
|---|---|---|
| Requirements | Everyone | Basic requirements of our product were defined (see product documentation) |
| Roles | Everyone | Roles were associated with project members (see project documentation) |
| Meeting schedule | Everyone | Meeting times, their schedule and times were defined (see project documentation) |
| Longterm Planning | Everyone | Milestones were defined and associated with Phases (See project documentation) |
| Shortterm Planning | Everyone | Formalize and document requirements, Start tracking time in gitlab, risk management, visualize timelime, order virtual server infrastructure |

**Additional notes**

- Todos:
  - Adrian: Send review-invitations to supervisor
  - Marco: Book a server for deployment of the application (Ubuntuserver)
  - Marco: Configure CI
  - Everyone: Think of different risks to our project until 03.03.2020
  - Jason: Include milestones in Versioncontrol
  - Jason: Create template for risk-graphics
  - Adrian: Formalize Requirements / Use-cases

## 11.3. Risk Management & Short Term Planning Meeting

| | |
|---|---|
| **Meeting date/time:** | 03.03.2022, 18:00 |
| **Meeting purpose:** | Risk Management & Short Term Planning |
| **Notetaker:** | Benjamin Kern |

**Attendees:**

- Adrian Locher
- Benjamin Kern
- Dominik Ehrle
- Jason Benz
- Marco Agostini

**Agenda**

| **Item** | **Description** |
|---|---|
| Risk Management | Probabilities, categorization, mitigation and effects of functional and nonfunctional risks |
| Short term planning | Define work items to be assigned during next sprint planning. |
| Time tracking | Get timetracking up to date and consistent across all teammates (meaning time is recorded in the appropriate issue) |
| Preparation for Review 1 | Going through review checklist once again to ensure all points are OK |

**Discussion**

| **Item** | **Who** | **Notes** |
|---|---|---|
| Risk Management | Everyone | Probabilities of personal and technological risks discussed, as well as categorization of said risks. Possible risk mitigation as well as their effects on the project in the long term discussed. Thorough discussion of effects of CI/server failure, API failure/change as well as the impacts of these risks on our architecture. |
| Short term planning | Everyone | Created issues for the work items to be assigned during the next sprint and added those to the appropriate milestones. Tagged with appropriate sprint label. |
| Time tracking | Everyone | As defined in agenda |
| Preparation for Review 1 | Everyone | As defined in agenda |

## 11.4. Review 1 - Project Plan & Sprint 02 planning

| | |
|---|---|
| **Meeting date/time:** | 07.03.2022, 10:10 |
| **Meeting purpose:** | Review 1 & planning of Sprint 02 |
| **Notetaker:** | Dominik Ehrle |

**Attendees:**

- Adrian Locher
- Benjamin Kern
- Dominik Ehrle
- Jason Benz
- Marco Agostini
- Thomas Kälin (Project advisor)

**Agenda**

| Item | Description |
|---|---|
| Review 1 & sprint retrospective | First Review with Thomas Kälin addressing the Project Plan with discussion/feedback of sprint 01 |
| Sprint planning | planning of sprint 02 and distribution of Issues |

### 11.4.1. Review 1 - Feedback/Improvements

- Define what the basic functions are
- Add description, what the job of a "supervisor" is
- create project backlog
  - define all features and estimate required time until when all "must" requirements are fulfilled → planed for sprint 02
- change estimated time for sprint review and sprint retrospective (sprint review 15-30 min, retrospective 60 min
- suggestion for definint sprint objective; choose a User story, which spanns across all working fields and thus covers everything in a small portion → allows to discuss a specific funcionality with the customer and define what has to be done in the background to fullfill this task (technical requirements) → Brings a communicational benefit towards the customer
- time tracking can be morre generic, mainly for a brief overview whether time schedule matches and for future projects as a reference
- in chapter 8 (time tracking) less detailed plan for a sprint, but more high level metrics like; how much time was spent and how much is left, are we still in our time schedule, how much time did each teammember invest, etc.
- rework and expand details for guidelines and define if linthing should be used, DoD (Definition of Done) should be used for features, etc.
- add or omit Vision to documentation
  - matches +- project proposal with a small enhancement
  - Mopckup of UI in Management Summary, not in Vision
- what could be done better:
  - focus planning around functionalities and build up from there → first define features and build backlog
  - risk analysis; get more out with less effort → better argue with measures insted of heavy

calculations
  - detailed branching strategie
  - automated spellcheking for document
  - and more small extra-tasks
- for review 2:
  - how are requirements defined
  - rough list of requirements with a few of them fine-tuned

**Discussion**

| **Item** | **Who** | **Notes** |
|---|---|---|
| sprint 2 planning | Everyone | - main objective; working login → technical requirements were defined and issues created/assigned<br>- creation of burndown chart<br>- new user story: define Workflow / Guidelines |
| realization of improvements | Everyone | implement improvements / changes from feedback of review 1 and what has to be done for review 2 |

## 11.5. Sprint review

| | |
|---|---|
| **Meeting date/time:** | 21.03.2022, 08:30-09:00 |
| **Meeting purpose:** | Sprint review |
| **Notetaker:** | Jason Benz |

### Attendees:

- Adrian Locher
- Benjamin Kern
- Dominik Ehrle
- Jason Benz
- Marco Agostini

### Agenda

| **Item** | **Description** |
|---|---|
| Issue presentation | Everybody present his issues |
| Status check | Check if every issue is closed |

### Discussion

| **Item** | **Who** | **Notes** |
|---|---|---|
| Frontend — Mockup | Dominik Ehrle | The login / register mockup is done. A home site and media overview / search site draft is created. The other sites could not be designed yet, because of different open questions. Therefore the team decided to take this issue into the next sprint. |
| Login implementation | Adrian Locher | The merge request is open, but it will be done until midday. If the merge request will pass, the issue can be closed (update: issue was closed in time) |
| All other issues | Everyone | All other issues are completly done |

## 11.6. Sprint retrospective

|  |  |
|---|---|
| **Meeting date/time:** | 21.03.2022, 09:00-10:00 |
| **Meeting purpose:** | Sprint retrospective |
| **Notetaker:** | Jason Benz |

**Attendees:**

- Adrian Locher
- Benjamin Kern
- Dominik Ehrle
- Jason Benz
- Marco Agostini

### Agenda

| Item | Description |
|---|---|
| Positive points | What went well in the sprint? |
| Negative points | What didn't go well in the sprint? |
| Suggestions for improvement | What can be done better in future sprints? (specific measures) |

### Discussion

| Item | Who | Notes |
|---|---|---|
| + Fewer ambiguities | Everyone | In the first sprint, there were some open questions (about procedures, collaboration, tooling, etc.). These were clarified and it is clear to all team members what the processes etc. look like. |
| + Improved performance | Everyone | In the last sprint there was an overhead because many topics were discussed by the entire team. In this sprint, the work was more speedy and the tasks were well divided. Smaller topics were discussed in groups of two and only the result was presented to the whole team. This saved some time. |
| - CI crashes | Everyone | The documentation CI did some crashes. Decision: Everyone is responsible for making the CI run more reliably. Some errors were IDE-dependent. Care should be taken that only executable elements are pushed and everyone should check their IDE. |
| - Uncompleted issue | Everyone | One issue could not be completed because it was too big. Decision: In the future, issues should be made smaller. The risk that an issue will not be completed is thus smaller. If an issue is still not finished, only a small piece has to be transferred to the new sprint instead of a large one. |

## 11.7. Review 2 - Requirements

|  |  |
|---|---|
| **Meeting date/time:** | 21.03.2022, 10:10 - 10:55 |
| **Meeting purpose:** | Review 2 & planning of Sprint 03 |
| **Notetaker:** | Marco Agostini |

**Attendees:**

- Adrian Locher
- Benjamin Kern
- Dominik Ehrle
- Jason Benz
- Marco Agostini
- Thomas Kälin (Project advisor)

**Agenda**

| Item | Description |
|---|---|
| Review 2 | The second review with Thomas Kälin addressing the requirements of the project. |

### 11.7.1. Review 2 - Feedback/Improvements

- Domain model
  - Correct the cardinality in the model
  - The rating association should have an attribute
  - The domain model is missing the verification email
- User stories
  - There may is a need for defined acceptance criteria for a user story
  - We need to make sure we can estimate how many user stories we are able to finish
- Put the epic infrastructure at the end of the epic list, since the customer is not interested in this epic
- NFR 11 - the verification method is missing
- Naming - make sure to name things consistent in the document
  - Naming 2.1.3 names accounts and profiles but the domain model lacks the use of this definitions
  - Naming 2.1.4 names name but the DB lacks the use of this definition

**Discussion**

| Item | Who | Notes |
|---|---|---|
| NFR | Everyone | When do the NFR get checked in the project? |
| Mockups for Users stories | Everyone | For very user story there should be made a mockup before realising it. |
| Domain Model | Everyone | There should be a description of the matching algorithm in order to understand the domain model better. |
| Consistency | Everyone | In order to boost the consistency of wording and technical terms in the project a glossary should be introduced. |
| Performance | Everyone | Also the description of the execution of the matching algorithm should be written down because it could have an impact on the NFR (Performance). |

## 11.8. Sprint review

| | |
|---|---|
| **Meeting date/time:** | 04.04.2022, 08:30-09:00 |
| **Meeting purpose:** | Sprint review |
| **Notetaker:** | Jason Benz |

**Attendees:**

- Adrian Locher
- Benjamin Kern
- Dominik Ehrle
- Jason Benz
- Marco Agostini

### Agenda

| **Item** | **Description** |
|---|---|
| Issue presentation | Everybody present his issues |
| Status check | Check if every issue is closed |

### Discussion

| **Item** | **Who** | **Notes** |
|---|---|---|
| Frontend — Mockup | Dominik Ehrle | Some mockups are refined, some other created. Since more refinements are needed, the task is going to stay open until the end of M5 - Alpha (May 02). |
| Media API selection & integration | Benjamin Kern | A refactoring and integration tests are required. This task will be closed, but a follow up task will be created. |
| Public profile | Jason Benz | There is a bug and more tests need to be written. That will be done in this sprint because it's just minor adjustments. (update: everything is ok now, issue is closed) |
| All other issues | Everyone | All other issues are completly done |

## 11.9. Sprint retrospective

| | |
|---|---|
| **Meeting date/time:** | 04.04.2022, 09:00-10:00 |
| **Meeting purpose:** | Sprint retrospective |
| **Notetaker:** | Jason Benz |

**Attendees:**

- Adrian Locher
- Benjamin Kern
- Dominik Ehrle
- Jason Benz
- Marco Agostini

### Agenda

| **Item** | **Description** |
|---|---|
| Positive points | What went well in the sprint? |
| Negative points | What didn't go well in the sprint? |
| Suggestions for improvement | What can be done better in future sprints? (specific measures) |

### Retro structure

Today's retro structure was a lot better than last time. This should be continued in this way.
Procedure:

1. Everyone thinks about what **went well** in the last sprint for 5 minutes.
2. After the time is up, everyone sends their input to the chat at the same time.
3. The (positive) results are discussed.
4. Everyone thinks about what **didn't go well** in the last sprint for 5 minutes.
5. After the time is up, everyone sends their input to the chat at the same time.
6. The (negative) results are discussed.
7. Specific improvements for the next sprint are proposed and discussed.
8. The note taker writes everything down to the protocol.

### Discussion

| Item | Who | Notes |
|---|---|---|
| + Teamwork | Everyone | Cooperation works very well and uncomplicated (regardless of whether it is an official SEProj day) |
| + Coding | Everyone | A large part of the time can be invested in product development. This is enjoyable and motivating. |
| + Know-how transfer | Everyone | Everyone has found their place in the team and the knowledge gained is shared. |
| + Task splitting | Everyone | The tasks are well divided. Thus, they are realistic to implement and there are few dependencies. |
| + Familiarity | Everyone | After a few initial difficulties (e.g. Microsoft login functionality) the team is now familiar with the code. |
| - Lack of tests | Everyone | Very few tests were written. Decision: Tests are part of the DoD. Reminder to all, to comply with the DoD and write more (meaningful) tests. |
| - Workflow deviation | Everyone | The workflows regarding version control (branching strategy) are partially not considered. Decision: All read the corresponding documentation chapter again and try to apply it. |
| - Unpunctuality | Everyone | At least someone is late for every meeting. Decision: Send calendar entries for all meetings. Meetings will start on time. If it doesn't improve by the next retro, consider a punishment system (e.g., beer for the team). |
| - Dev setup documentation | Everyone | The development setup is not documented. Decision: Document the development setup in the readme of the repo. |
| - Task delimitation | Everyone | It is partly unclear whether e.g. the UI implementation also belongs to the task. Decision: Tasks must be better defined in the refinement. |
| - Last minute changes | Everyone | Some tasks are changed or finished last minute. Therefore the tasks could not be reviewed in time. Decision: Try to finish the tasks in the middle / on Friday of the week before the review. Then a merge request can be done in time. |

## 11.10. Review 3 - End of Elaboration

|  |  |
|---|---|
| **Meeting date/time:** | 04.04.2022, 10:10 |
| **Meeting purpose:** | Review 3 |
| **Notetaker:** | Benjamin Kern |

**Attendees:**

- Adrian Locher
- Benjamin Kern
- Dominik Ehrle
- Jason Benz
- Marco Agostini
- Thomas Kälin (Project advisor)

**Agenda**

| Item | Description |
|---|---|
| Review 3 | Discussion of end of elaboration |

### 11.10.1. Review 3 - Feedback/Improvements

**Positive Feedback**

- Prototype is good - End-to-End Functionality (”Durchstich”) achieved
- Ahead of normal schedule, lots of runnable code compared to other teams

**Improvements / Negative Feedback**

- Refine long term project plan:
  - Presumably less time for should/could requirements than currently planned. Do not underestimate amount of admin tasks for presentation / end of project.
- Risks R11, R12, R13 have been mitigated or eliminated, should be apparent in documentation.
  - Answer from team: Is apparent from residual risk matrix, occurred risks are mentioned explicitly in text.
- UI Mockups are only accessible through gitlab. Until they are final, a link should be added in the documentation. Add directly to documentation when they are final.
- Quality measures: 80% test coverage should be mentioned in documentation.
- Domain model’s comments should be adapted once the recommendation algorithm is established and implemented.
- Architecture:
  - Show defined interfaces (method signatures) between API’s, as this is what matters to potential new developers joining the team.
  - Add shadow to media api to indicate that it may represent several concrete api’s
  - Indicate that the currently used media api is TMDB.
  - Containers are not conform with C4 standard: Containers are seperately runnable and deployable units. There should be 3 containers: Database, the SUD and the Media API.
  - Figure of architecture (Figure 4.4); is this color coded? Lots of similar colors, but no 1 to 1 mapping.
  - If possible, simplify components structure

  - Document architecture decisions: Why was TMDB API chosen over IMDB?
  - Figure of navigationflow should maybe be moved from architecture to requirements analysis.

**What to expect from Review 4**

- Show that our code satisfies established quality measures e.g. test coverages.
- Code Review on a very high level: Readability, structural desicions into classes. Main purpose is establishing whether a new developer would be able to follow and contribute to the project.

## 11.11. Sprint review

| | |
|---|---|
| **Meeting date/time:** | 19.04.2022, 08:30-09:00 |
| **Meeting purpose:** | Sprint review |
| **Notetaker:** | Jason Benz |

### Attendees:

- Adrian Locher
- Benjamin Kern
- Dominik Ehrle
- Jason Benz
- Marco Agostini

### Agenda

| Item | Description |
|---|---|
| Issue presentation | Everybody present his issues |
| Status check | Check if every issue is closed |

### Discussion

| Item | Who | Notes |
|---|---|---|
| Frontend — Mockup | Dominik Ehrle | No new changes. The task is going to stay open until the end of M5 - Alpha (May 02). |
| Delete user profile | Jason Benz | The estimated time was not required. Therefore some time was invested into exception handling and some code improvements. |
| Movie overview styling | Marco Agostini | Since this issue is covered by issue #39 no time were spent on this specific issue. |
| Media Api Integration Tests | Benjamin Kern | Another more priorized issue needed more time. Therefore this issue is not completed. Will be finished in the next sprint. |
| Chat — Basic UI | Benjamin Kern | Due to unexpected problems, this issue was not finished. It will be completed in the next sprint. |
| All other issues | Everyone | All other issues are completly done. |

## 11.12. Sprint retrospective

|  |  |
|---|---|
| **Meeting date/time:** | 19.04.2022, 09:00-10:00 |
| **Meeting purpose:** | Sprint retrospective |
| **Notetaker:** | Jason Benz |

**Attendees:**

- Adrian Locher
- Benjamin Kern
- Dominik Ehrle
- Jason Benz
- Marco Agostini

### Agenda

| Item | Description |
|---|---|
| Positive points | What went well in the sprint? |
| Negative points | What didn't go well in the sprint? |
| Suggestions for improvement | What can be done better in future sprints? (specific measures) |

### Discussion

| Topic | Who | Notes |
|---|---|---|
| + Meeting start | Everyone | Meeting start is now on point. Keep it running like this! |
| + Progress | Everyone | The progress of the software is visible. A lot of features were added. This is motivationg. |
| + Workflows | Everyone | Everyone is sticking to the workflows discussed and has internalized them. E.g. branching strategy. |
| + Code review | Everyone | The code reviews (merge requests) are done with a high quality. |
| + MVP | Everyone | All must requirements are in progress. The goal to finish the MVP by the next sprint is realistic. |
| + Time estimations | Everyone | Overall, the time estimates are correct. |
| - Logging | Everyone | There is no logging library in use. Only exceptions are thrown, but no log messages are written. Decision: New issue for the next sprint. |

## 11.13. Review 4 - Quality

| | |
|---|---|
| **Meeting date/time:** | 25.05.2022, 10:10 |
| **Meeting purpose:** | Review 4 |
| **Notetaker:** | Marco Agostini |

#### Attendees:

- Adrian Locher
- Benjamin Kern
- Dominik Ehrle
- Jason Benz
- Marco Agostini
- Thomas Kälin (Project advisor)

#### Agenda

| **Item** | **Description** |
|---|---|
| Review 4 | Discussion of quality measurements |

### 11.13.1. Review 4 - Feedback/Improvements

#### Positive Feedback

- Nice demo of the search mechanism and the new feature to rate media.
- The collaboration techniques (Git) are used very good and in a consistent way.
- Quality measures are good for the size of the project.
- Time accurate invitations and good moderation of the meetings.

#### Improvements / Negative Feedback

- Testing
  - The team has chosen unit, usability and E2E tests should be used in the project. The issue is in the code are some integration test which should be E2E since this is state in the project documentation.
  - There are some unit tests which only test the mock. Please change these tests to useful test scenarios.
  - Tests with other teams are planned and should be started with the alpha version. This should be stated somewhere in the documentation.
- Code Review
  - Figure of architecture components (Figure 4.3); We have different modules on the components graph compared to the code structure of the product. We may should consider to reorganise the product to match the module types of the components graph.
  - Figure 4.3 shows feature modules in the business model, which are not separated in the code.
  - Figure 4.3 does show infrastructure and configuration, but where is this found in the code?
  - There is a profile service in the persistency layer, which cannot be found in the code base.
  - Figure of architecture (Figure 4.4); What does the ring infrastructure mean and where can it be found in the code. The infrastructure ring does not make much sense.
  - The configuration / DI ring could be additional described with the informations which

namespace it does use.

  - The Gitlab repository does need some cleaning, since there are lot of unused files.

- General
  - Consider to create to create a final acceptance step with a plan of which features which person does test of the software. - This is planned with other EPJ teams.

## 11.14. Sprint review

|  |  |
|---|---|
| **Meeting date/time:** | 02.05.2022, 08:30-09:00 |
| **Meeting purpose:** | Sprint review |
| **Notetaker:** | Jason Benz |

### Attendees:

- Adrian Locher
- Benjamin Kern
- Dominik Ehrle
- Jason Benz
- Marco Agostini

### Agenda

| Item | Description |
|---|---|
| Issue presentation | Everybody present his issues |
| Status check | Check if every issue is closed |
| MVP | Internal presentation of the MVP |

### Discussion

| Item | Who | Notes |
|---|---|---|
| Manage contacts | Marco Agostini | The time estimation is realistic, but the issue is not finished, because of some other additional work (code cleanup / automated testing). The issue will be completed in the next sprint. |
| User suggestion | Jason Benz | The user suggestions requirements of the MVP are fulfilled. During development some possible improvements have been identified which will be done in the next sprint. |
| Chat | Benjamin Kern | The code output matches with the minimal requirements of the MVP. There will be needed some additional work. Therefore a following up issue is created. |
| All other issues | Everyone | All other issues are completly done. |
| MVP | Everyone | All must requirements are completed. Therefore the MVP / Alpha version is ready and the milestone is reached in time. The different parts of the MVP were presented by the team members. The user tests with the other SE Project team will be initiated. |

## 11.15. Sprint retrospective

| | |
|---|---|
| **Meeting date/time:** | 19.04.2022, 09:00-10:00 |
| **Meeting purpose:** | Sprint retrospective |
| **Notetaker:** | Jason Benz |

### Attendees:

- Adrian Locher
- Benjamin Kern
- Dominik Ehrle
- Jason Benz
- Marco Agostini

### Agenda

| Item | Description |
|---|---|
| Positive points | What went well in the sprint? |
| Negative points | What didn't go well in the sprint? |
| Suggestions for improvement | What can be done better in future sprints? (specific measures) |

### Discussion

| Topic | Who | Notes |
|---|---|---|
| + MVP | Everyone | The team reached a pretty solid project progress. The MVP / Alpha state is reached. |
| + Feedback | Everyone | The feedback process is working. Much of the feedback is applied by the team members. |
| - Dependencies | Everyone | Some issues have close dependencies. Decision: The MVP is completed. Therefore the team can work on issues with less dependencies to the other issues in the sprint. No action is required. |
| - Infrastructure work | Everyone | Infrastructure work (like SonarQube configuration, pipline improvement etc.) are not calculated within the estimations. Decision: Create also issues for this tasks. |
| - Time estimations | Everyone | Some unexpected problem occured. Therefore the time estimations were inaccurate. Decision: The time estimations should include some reserves. |

## 11.16. Sprint review

| | |
|---|---|
| **Meeting date/time:** | 16.05.2022, 08:30-09:00 |
| **Meeting purpose:** | Sprint review |
| **Notetaker:** | Jason Benz |

### Attendees:

- Adrian Locher
- Benjamin Kern
- Dominik Ehrle
- Jason Benz
- Marco Agostini

### Agenda

| **Item** | **Description** |
|---|---|
| Issue presentation | Everybody present his issues |
| Status check | Check if every issue is closed |
| MVP | Internal presentation of the MVP |

### Discussion

| **Item** | **Who** | **Notes** |
|---|---|---|
| Manage contacts | Marco Agostini | Issue is nearly done. Some little improvements are required. Will be done at the start of this week. Therefore it won't be moved to the next sprint. |
| User suggestion improvement | Jason Benz | The issue is nearly done. The only thing missing is the integration of the contact module (open issue above). Will be done at the start of this week as well. |
| Feed | Jason Benz | The issue is nearly done. The only thing missing is the integration of the contact module (open issue above). Will be done at the start of this week as well. |
| Media comment | Dominik Ehrle | The issue is done. Only the code review is pending. Will be done today. |
| Chat | Benjamin Kern | Isn't done. More improvement is required. Will be moved to the next sprint. |
| All other issues | Everyone | All other issues are completly done. |

## 11.17. Sprint retrospective

| | |
|---|---|
| **Meeting date/time:** | 16.05.2022, 09:00-10:00 |
| **Meeting purpose:** | Sprint retrospective |
| **Notetaker:** | Jason Benz |

**Attendees:**

- Adrian Locher
- Benjamin Kern
- Dominik Ehrle
- Jason Benz
- Marco Agostini

### Agenda

| Item | Description |
|---|---|
| Positive points | What went well in the sprint? |
| Negative points | What didn't go well in the sprint? |
| Suggestions for improvement | What can be done better in future sprints? (specific measures) |

### Discussion

| Topic | Who | Notes |
|---|---|---|
| + Status | Everyone | Overall, we have achieved a good output in this project. |
| + Wide range of work | Everyone | Different topics are worked on (docker, pipelines, unit testing, different modules, frontend, backend etc.). It is a diversified job. |
| + Git workflow | Everyone | The git workflow is improving every sprint. |
| - Dependencies | Everyone | Again some issue dependencies. |
| - Issue completion | Everyone | Not all issues were done in time. Decision: Ask for help earlier! |

## 11.18. Review 5 - Architecture

| | |
|---|---|
| **Meeting date/time:** | 16.05.2022, 10:10 - 10:55 |
| **Meeting purpose:** | Review 5 |
| **Notetaker:** | Benjamin Kern |

**Attendees:**

- Adrian Locher
- Marco Agostini
- Dominik Ehrle
- Jason Benz
- Benjamin Kern
- Thomas Kälin (Project advisor)

**Agenda**

| Item | Description |
|---|---|
| Review 5 | Discussion of the project architecture. |

### 11.18.1. Review 5 - Feedback/Improvements

- Previous feedback
  - Mostly taken into account
  - Integration tests are missing in the documentation
- SPA Webapplication with clean architecture
  - Well chosen, cleanly applied
  - Modern technologies is a plus
  - MVVM and DI applied where useful
- NFR
  - Tested and fulfilled
- Databases in Figure 4.2 should be inside dashed line
- Some design decisions are missing in documentation
  - Why Blazor server and not ASP.NET Core with Razor pages?
  - TMDB over IMDB, SendGrid, MSSQL, EF Core, etc.
- Documentation should be cleaned up (instructions left over etc.)
- section 2.2 should describe NFRs, but does not.
- section 5.1 should describe execution of usbility tests, but does not.

**Discussion**

| Item | Who | Notes |
|---|---|---|
| Domain Model | Everyone | Seems to be out of date - has been updated in the meantime, it is now accurate again. |

## 11.19. Project retrospective

| | |
|---|---|
| **Meeting date/time:** | 30.05.2022, 09:00-10:00 |
| **Meeting purpose:** | Project retrospective |
| **Notetaker:** | Jason Benz |

### Attendees:

- Adrian Locher
- Benjamin Kern
- Dominik Ehrle
- Jason Benz
- Marco Agostini

### Agenda

| **Item** | **Description** |
|---|---|
| Positive points | What went well in the project? |
| Improvements | What can be done better in future projects |

### Positive points

- The teamwork worked very well
- Visible progress was evident in every sprint
- It was a lot of pleasure to create a project from scratch
- We were involved in every part of the software development process (planning, infrastructure setup, programming, deployment, etc.)
- The processes (e.g. Git workflow) were further optimized in each sprint
- The retrospective helped us to improve
- Criticism was taken seriously and acted upon
- The MVP and the desired optional features have been implemented

### Improvements

- Since sprint meetings create overhead and we were only working two days per person per sprint, longer sprints would be better
- The tasks were partially completed on the last day of the sprint. Therefore, it was not possible to help each other in case of problems. A longer sprint would help over here as well
- We would evaluate the working tools (time tracking, issue management, ci / cd, etc.) next time before the project starts