

Experimental web-based social network with advanced  
activity analysis system

Study program: Software Engineering and Technology  
Author: Viktor Kozhemiakin

Supervisor: Ing. Michal Lucki, Ph.D.  
Feb, 2024

## I. Personal and study details

Student's name: **Kozhemiakin Viktor** Personal ID number: **499361**  
Faculty / Institute: **Faculty of Electrical Engineering**  
Department / Institute: **Department of Computer Science**  
Study program: **Software Engineering and Technology**

## II. Bachelor's thesis details

Bachelor's thesis title in English:

**Experimental web-based social network with advanced activity analysis system**

Bachelor's thesis title in Czech:

**Tvorba experimentální webové sociální sítě s pokročilým systémem analýzy aktivit**

Guidelines:

This work aims to create a web-based social network that provides an evaluation system (likes, comments). It analyzes the history of reactions to a post and creates statistics in a form of graphs, identifies automatic bots and their possible negative reactions (attacks) to posts. The social network can be completely autonomous or enable login via Facebook or Google. The project will be created in Java Script (Full Stack) and will include a graphical user interface (Frontend). Verification of the functionality and testing the system will be carried out on the target group based on the instructions from the supervisor.

Bibliography / sources:

[1] A. Barengi et al., Snake: An End-to-End Encrypted Online Social Network, 2014 IEEE Intl Conf on High Performance Computing and Communications, 2014 IEEE 6th Intl Symp on Cyberspace Safety and Security, Paris, France, 2015  
[2] Y. Lin, Chatbot Script Design for Programming Language Learning, 2022 IEEE 5th Eurasian Conference on Educational Innovation (ECEI), Taipei, Taiwan, 2022  
[3] U. Sa'adah et al., Implementing Singleton method in design of MVC-based PHP framework, 2015 International Electronics Symposium (IES), Surabaya, Indonesia, 2015

Name and workplace of bachelor's thesis supervisor:

**Ing. Michal Lucki, Ph.D. Department of Telecommunications Engineering FEE**

Name and workplace of second bachelor's thesis supervisor or consultant:

Date of bachelor's thesis assignment: **30.01.2023** Deadline for bachelor thesis submission: **05.02.2024**

Assignment valid until: **22.09.2024**

Ing. Michal Lucki, Ph.D.  
Supervisor's signature

Head of department's signature

prof. Mgr. Petr Páta, Ph.D.  
Dean's signature

## III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

\_\_\_\_\_  
Date of assignment receipt

\_\_\_\_\_  
Student's signature

## Acknowledgements

I would like to thank Ing. Michal Lucki, Ph.D. for becoming my supervisor, who believed in me and my project. Also, I would like to thank my family, because they believed in my success.

## Declaration

I hereby declare that I have completed this thesis independently and that I have listed all the literature and publications used.

## Abstract:

The main goal of the work is to develop a web application that is a social network and explore the problem of manipulation. The application is intended to provide a comfortable platform for communication among users. Academic papers that explore the problem of manipulation were read. Then existing popular social networks were analyzed and the best features and options considering the disadvantages and common problems were selected. Creating own social media helps to understand their main problems from the owner's side and the technical side. Next application functions, implementation, and technologies were represented – the solution of the problem. Finally, the results of the work were analyzed and comprehended.

**Keywords:** Web application, social network, JavaScript, ReactJS, bots, figma, microservices, microfrontends, Node.js, paid commenters

## Abstrakt:

Hlavním cílem práce je vyvinout webovou aplikaci, která je sociální sítí a prozkoumat problém manipulace. Aplikace má poskytovat pohodlnou platformu pro komunikaci mezi uživateli. Byly přečteny akademické práce, které zkoumají problém manipulace. Poté byly analyzovány existující populární sociální sítě a vybrány nejlepší vlastnosti a možnosti s ohledem na nevýhody a běžné problémy. Vytváření vlastních sociálních médií pomáhá porozumět jejich hlavním problémům ze strany majitele i po technické stránce. Představeny byly další aplikační funkce, implementace a technologie – řešení problému. Nakonec byly výsledky práce analyzovány a pochopeny.

**Klíčová slova:** Webová aplikace, sociální síť, JavaScript, ReactJS, boty, figma, microslužby, microfrontends, Node.js, placené komentátory

# Contents

<b>Declaration</b>	<b>2</b>
<b>Contents</b>	<b>5</b>
<b>List of figures and tables</b>	<b>7</b>
<b>Chapter 1</b>	<b>9</b>
Introduction	9
1.1 Motivation	9
<b>Chapter 2</b>	<b>11</b>
Research	11
2.1 Material and methods	11
2.2 Research of existing applications	11
2.2.1 Social network with photos	12
2.2.2 Social network with texts	14
2.2.3 Social network with videos	14
2.2.4 Messenger	15
2.2.5 Common traits of social networks	16
2.2.6 Comparison of social networks	17
2.3 Strategic analysis and planning	18
2.3.1 SWOT analysis	18
2.3.2 PEST analysis	18
2.3.3 5F analysis	19
2.4 Choice of technical solution	20
2.4.1 Frontend	21
2.4.1.1 Angular	22
2.4.1.2 Vue	22
2.4.1.3 React	22
2.4.1.4 Summary	23
2.4.2 Backend	24
2.4.2.1 Express	24
2.4.2.2 Nest	24
2.4.2.3 Summary	25
2.4.3 Manipulation	25
2.4.3.1 Bots	26
2.4.3.2 Paid commenters	27
2.4.3.3 Scam	28
2.4.4 Conclusion	29
2.5 Requirements	29
2.5.1 Functional requirements	29
2.5.2 Use cases	31
2.5.2.1 Posts	31
2.5.2.2 Comments	32
2.5.2.3 Users and roles	33
<b>Chapter 3</b>	<b>35</b>

Design	35
<b>3.1 Class diagram</b>	<b>35</b>
<b>3.2 Deployment diagram</b>	<b>36</b>
<b>3.3 UI Design</b>	<b>37</b>
3.3.1 Create a new post	37
3.3.2 Comments	38
3.3.3 Followers	39
3.3.4 User post	40
3.3.5 Conclusion	42
<b>Chapter 4</b>	<b>43</b>
Implementation	43
4.1 Backend	43
4.1.1 Authentication and authorization	45
4.1.2 Endpoints	47
4.1.3 Bot prevention	49
4.2 Frontend	49
4.2.1 Authentication and authorization	51
4.2.2 Components and pages	51
4.3 Implementation of project requirements	53
4.3.1 Writing posts	53
4.3.2 Open opinion	54
4.3.3 Attaching poll	55
4.3.4 Reactions	55
4.3.5 Adding options	55
4.3.6 Voting for options	56
4.3.7 Comments	56
4.3.8 Replies on comments	57
4.3.9 Viewing a post statistics	58
4.3.10 Subscribes and subscribers	58
4.3.11 Viewing a user's statistics	59
4.3.12 Spam reporting	59
4.3.13 Requests about moderator	60
4.3.14 Ban users	61
4.3.15 Ban moderators	61
4.3.16 Consideration of request	61
4.3.17 End of moderator rights	61
4.3.18 Removing comments and posts by moderator	61
4.3.19 Removing comments and posts by admin	61
4.4 Deployment	61
4.4.1 Environment	61
4.4.2 Public access	64
<b>Chapter 5</b>	<b>65</b>
Testing	65
5.1 Automated tests	65



5.2 Paid commenters	66
5.3 Bots	68
5.4 Conclusion	70
<b>Chapter 6</b>	<b>71</b>
Conclusion	71
6.1 Total	71
6.2 Future plans	72
<b>Bibliography</b>	<b>73</b>
<b>Acronyms</b>	<b>78</b>

## List of figures and tables

- [Figure 2.2: Structure of Instagram posts](#)
- [Figure 2.3: Structure of Telegram comment with reply and multiple emojis](#)
- [Table 2.4: Comparison of social networks](#)
- [Table 2.5: SWOT analysis](#)
- [Table 2.6: PEST analysis](#)
- [Figure 2.7: Architectures of website](#)
- [Table 2.8: Comparison of client-side JS technologies](#)
- [Figure 2.9: NestJS component structure](#)
- [Figure 2.10: The architecture of a botnet](#)
- [Figure 2.11: The architecture of a factory](#)
- [Figure 2.12: Full information of votes on vk.com](#)
- [Figure 2.13: Use case model of posts](#)
- [Figure 2.14: Use case model of comments](#)
- [Figure 2.15: Use case model of users' rights](#)
- [Figure 3.1: A class diagram of the system](#)
- [Figure 3.2: A deployment model](#)
- [Figure 3.3: Creating post page](#)
- [Figure 3.4: List of comments](#)
- [Figure 3.5: Followers page](#)
- [Figure 3.6: User's post page](#)
- [Figure 4.1: Project structure of the post microservice](#)
- [Figure 4.2: Project structure of the api-gateway microservice](#)
- [Figure 4.3: Structure of token](#)
- [Figure 4.4: An endpoint](#)
- [Figure 4.5: Swagger UI](#)
- [Figure 4.6: User controller](#)
- [Figure 4.7: User service](#)
- [Figure 4.8: Project structure of shell microfrontend](#)
- [Figure 4.9: Project structure of user microservice](#)

- [Figure 4.10: Single Post page](#)
- [Figure 4.11: Poll component](#)
- [Figure 4.12: Mock comments](#)
- [Figure 4.13: Create post page](#)
- [Figure 4.14: Single post](#)
- [Figure 4.15: Post with poll](#)
- [Figure 4.16: Post with voted options](#)
- [Figure 4.17: Comment list](#)
- [Figure 4.18: A process of replying](#)
- [Figure 4.19: A list of subscribers](#)
- [Figure 4.20: Statistics of subscribers](#)
- [Figure 4.21: A modal window of report](#)
- [Figure 4.22: User info page](#)
- [Figure 4.23: Dockerfile for ms-api-gateway](#)
- [Figure 4.24: Docker-compose.yaml file](#)
- [Figure 4.25: Main page of the site](#)
- [Figure 5.1: Tests for User service](#)
- [Figure 5.2: Test of createComment function](#)
- [Figure 5.3: Comments of user](#)
- [Figure 5.4: Statistics of likes](#)
- [Figure 5.5: Statistics of subscribers](#)
- [Figure 5.6: Statistics of post's votes](#)

# Chapter 1

## Introduction

This chapter describes the motivation and the aim of the project. The motivation is the explanation of why the main problem is important and relevant, the aim is the desired outcome of this thesis.

### 1.1 Motivation

Social networks are one of the most important parts of public life today. People can message their friends, share their thoughts with everyone and read posts from other users or watch videos. But today social networks are not only sites for communication, they also are an important tool of influence. A person can influence other people or be influenced by famous people or trends. According to statistics, many people use social networks almost every day. [1]

This influence can be used for misinformation and scam. Many modern social networks have a weak and non-transparent system of rating. Users can't see the complete picture of a post: the number of likes and dislikes, and the author of a post/video can delete unwanted comments, which can contain information about the author's lie or misleading. Influencers can manipulate people, who don't see the real statistics and the full situation. Also, internet bots and paid commenters are another huge problem. They simulate the activity of the network. Many special bots can boost and change the numbers of likes/views/votes, paid commenters can write hypocritical custom-made comments [2]. Then a user gets a distorted picture of reality with fakes and lies. It's a brazen manipulation and a usual person is defenceless against this.

Social networks have user-friendly designs and high performance, but they may have disadvantages, such as a non-transparent and complex for users system of rating: likes/dislikes. Moreover these networks are afraid of changes.

### 1.2 Aims

The project aims at creating a modern user-friendly social network that provides an opportunity for comfortable communication between users through posts with video, photos, comments, and polls that contain options. Users can vote for these options and see statistics, also they can add a new option if the post doesn't have the option yet. Users can react to posts and comments through likes and dislikes, which cannot be disabled. Thanks to a user-friendly interface, the network is available for all ages and levels of computer skills.

Also, this network should be resistant to bots and paid commenters. The network should recognize bots, special accounts that were created only for spam and number boosting (likes and dislikes). Special farms can create thousands of bots for self-interest manipulations or misinformation, and the network should prevent this and automatically identify these bots. Paid commenters are real people, but every professional paid commenter, who works on a farm, can have 1-10 accounts and simulate the behavior of a normal user. [3] However, the social network can provide some public information about every account: their followers, comments and reactions, and people can decide for themselves: is this account a bot or not.

Before coding, technical documentation was developed: functional requirements, deployment diagram, use cases, class model, list of languages and technologies, architecture, etc. All this helped to create a modern social network that can do its work quickly and effectively.

The first step was the Frontend. The Frontend part was created first because it shows all moments and all endpoints that a site needs. The Frontend was written in the modern library for JavaScript – ReactJS, which is used for Instagram and Twitter. Also, this Frontend was separated into different small parts - microfrontends: every solid part is a small application. This software architectural style simplifies scalability.

The second step was the Backend. The Backend was written in NestJS with PostgreSQL – a [SQL](#) database. The backend also uses the separation style - microservices: every solid functionality is a small application.

The final step was testing: unit tests were written for the public [API](#) methods and non-trivial functions as well as end-to-end frontend interface tests. The public API methods and non-trivial functions were documented and described. Also, this part includes usability testing.

GitHub was used for version control. Every major change was committed and commented. It ensures quality and the opportunity to find and correct errors. Personal web server was used for deployment and hosting.

# Chapter 2

## Research

This chapter contains the analysis of existing social networks, a technical solution, and requirements. The requirements of the project were determined based on the advantages and disadvantages of modern applications.

### 2.1 Material and methods

#### **Creating an experimental web application as a model for testing**

The first step in problem-solving is to create a website as a model for testing. This experimental website should be used for constructing and testing different possible situations. Also, the website will show unexpected moments and errors.

#### **Analysis of activity**

Every action in the system should be registered and added to the database. Analysis of all data provides detailed statistics, then it will be used for advanced analysis and data visualization in the form of charts. Firstly, analysis of activity helps to identify bots, secondly, data visualization helps users to see a complete picture of a post/account.

#### **Creating “fake” bots and paid commenters**

Creating fake bots will support testing the antibot system and simulate the behavior of a typical farm of bots or paid commenters. These bots will attack the application: boosting likes, dislikes, or subscriptions. The stress test shows the efficiency of the system and helps to find new methods of identification of bots. If the system passes this stress test, then it is successful and working.

#### **Analysis of the other experience and academic papers**

The main problem of this work is well researched and many academic papers explore this problem. Also, solving the problem is relevant for companies, which create new solutions.

### 2.2 Research of existing applications

Social networks with millions of users were analyzed because a modern and competitive application should use real-life best practices. These applications are a suitable choice for analysis.

The main criteria are:

- User-friendly post system. A post (video, photo, text) is the main unit of content. Other features such as comments, likes/dislikes depend on the posts;
- Transparency of data. Transparent data provides full statistics and enables users to make informed decisions about the content. These users cannot be misled;

- Comment system. Comments are an important part of a post that complements the post. Comments provide feedback from users and help other users to understand the post better through detailed reviews.
- Variety of reactions. Social networks should offer different types of reactions because users can react to posts or comments differently. Reactions as comments also provide feedback from other users and help other users understand the post better using quick statistics based on numbers.

Social networks with a large number of users are most relevant, because these networks use best practices for attracting users and creating engaging content. [Table 2.1](#) shows the most popular social networks in January 2022.

<b>Social network</b>	<b>Number of users (million users)</b>
Facebook	2910
YouTube	2562
WhatsApp	2000
Instagram	1478
TikTok	1000
Telegram	550
Pinterest	444
Twitter	436

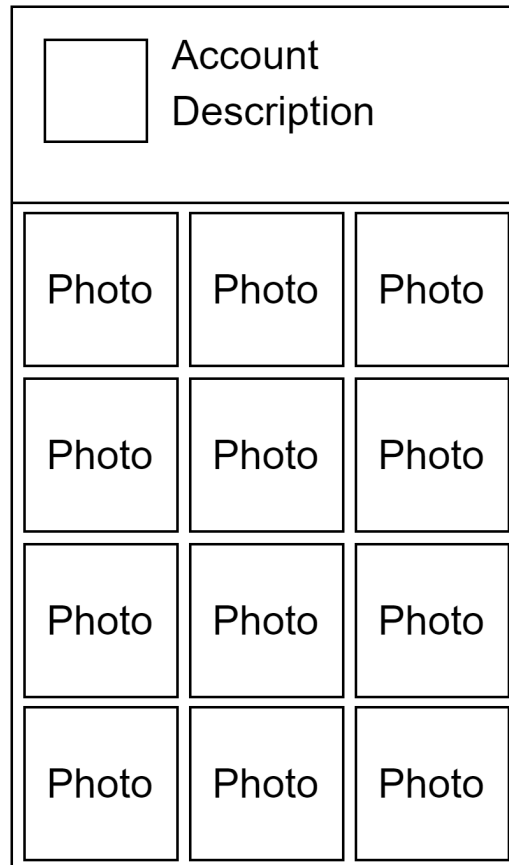
*Table 2.1: Most popular social networks worldwide as of January 2022, ranked by the number of monthly active users*

### 2.2.1 Social network with photos

Different types of social networks exist: with a focus on photos, videos and texts. The first type for analysis is a social network with photos - Instagram, which was created by Meta. Instagram is one of the largest social platforms in the world. Instagram has 1.5 billion users. The core ideas of these social networks are showing and exchanging photos. Users can watch photos of friends or famous influencers and post their photos with short text for sharing with other users. Also, Instagram provides the ability to promote and grow businesses, where business owners can show their services or products in photos and communicate with customers. [4]

The main focus of Instagram is photos. Therefore Instagram has a required photo for a post. It has advantages, for example, Instagram allows users to see all posts in a list with 3 photos per line, and a photo serves as an identifier for each

post. A user can scroll through all Instagram posts to the oldest post and can find any post if he knows which photo the post has. [Figure 2.2](#) shows the structure of Instagram posts.



*Figure 2.2: Structure of Instagram posts.*

But the comment system is not user-friendly. Replies are sorted in ascending order, but a user can see only 3 last comments, if he wants to see more than 3 comments, then he must click again. Finally, if a user wants to see the first reply to a comment with 150 replies, he must click "show more replies" 49 times  $(150-3)/3$ . It takes a lot of time.

Instagram has only likes, but it can be disabled. It makes Instagram a very non-transparent social network. Meta company writes: "We tested hiding like counts to see if it might depressurize people's experience on Instagram. What we heard from people and experts was that not seeing like counts was beneficial for some, and annoying to others, particularly because people use like counts to get a sense for what's trending or popular, so we're giving you the choice." [5] However, a non-transparent system is very vulnerable to bots and paid commenters. For example, an author can buy subscribers for an account, hide comments and promote some product. It can be noticed that an account has many subscribers and some likes, but without any number, however people can believe this advertisement. If

someone wants to see likes, it's possible through special services or extensions, but most people will not do this, because they are lazy, and most people don't pay attention, and they don't know the full situation.

### 2.2.2 Social network with texts

A social network with a text focuses on text. This network can have a video, or a photo, but the main content and goal is text. Twitter (now "X") is a large social network for short messages - tweets. Users use text as the main unit of information, they divide text into multiple parts or put text into an attached picture if text exceeds the limit.

The main problem of Twitter's comments is the fact that every comment, reply in Twitter is a tweet. Twitter uses a microblogging structure. [6] Twitter has no separation of comments and posts (twitts). It creates a chaotic heap of everything. But this comment system is comfortable for viewing replies, because under each tweet the replies - retweets are displayed. It allows users to concentrate on one topic and keep their attention.

Twitter has only likes like Instagram. It also creates problems with transparency of public opinion, however the number of likes on Twitter cannot be disabled, as on Instagram. On the contrary to Instagram, Twitter has a transparent system for polls. A poll has a limited time from 5 minutes to 7 days, then the poll ends. The voting in the poll is anonymous, the poll shows only results. The poll is closed, only the author can define options, because it's a common practice of online-polls in social networks.

### 2.2.3 Social network with videos

A social network with a focus on videos has videos as its main content. YouTube is a video hosting and a social network with videos. YouTube has many interesting technical features that can serve as examples for inspiration.

YouTube had a likes/dislikes system, which could represent the viewers' preference. [7] But now dislikes are disabled, nevertheless, people can install an extension for the browser and see dislikes, however dislikes make no sense because most people don't use the extension, people are lazy, and not everyone has time for this.

Comments are a very inalienable part of a video because people can discuss the video and communicate with each other, comments may increase the total number of viewers. [8] YouTube has a complex and advanced system of comments. Users can sort comments by date and by popularity, which is computed by YouTube algorithms. This mode sorts comments in an actual order by the number of likes, period and trends. If a video was popular some years ago, and it became relevant again, users see new popular comments, YouTube system understands that an old



comment (5 years ago) with 1000 likes is less important than a new comment (1 day ago) with 800 likes. Also, the author of the video can like and pin comments, it helps to display some important information about the video, because pinned comments have high priority. Finally, the comment shows all replies. But YouTube comments have one major disadvantage: videos for children have no comments and so do videos for non-children users too.

YouTube provides a lot of data for detecting the boost on YouTube Analytics. YouTube comment system is well-developed and responsive, but YouTube has problems with bots, spambots on YouTube are a common thing. YouTube has its own moderation system for comments and this can prevent provocative comments, also the author of the channel can add their own "red" words. Another serious problem is clickbaits: an author of a channel shows a picture with a new movie, and writes that this video is a teaser of the new movie, but in reality this is a video with the opinion of the author with a black background.

YouTube is a very popular platform among paid commenters and bots. Bots are used for like/dislike/subscription boosting. Some channel wants to add new subscribers. An author can buy 10000 new subscribers for 1000\$, or buy positive comments for their own video. The rise in popularity makes YouTube vulnerable to bots and paid commenters. [9] However, YouTube can prevent bots, creating bots for YouTube is hard.

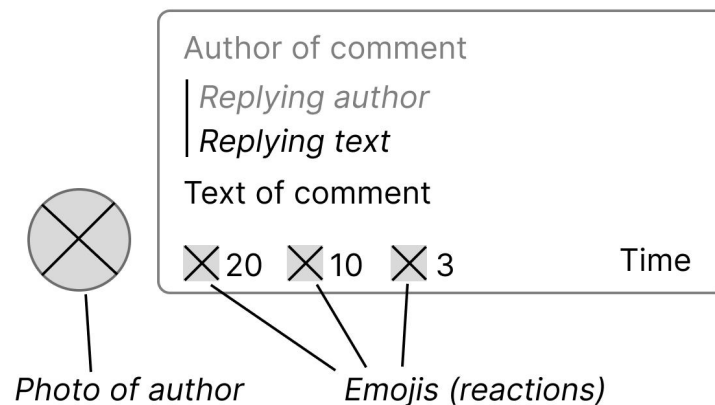
#### 2.2.4 Messenger

Telegram is not a social network. But it's a messenger that evolves into a social network. Telegram has public channels with messages - posts with optional photos, videos, and audio messages. Also, people can comment on these post and message each other.

The main advantages of Telegram are its abilities. Telegram is the most extensible platform among others. The Telegram platform helps to get feedback from the audience of the channel. People can use Telegram bots (not spam bots) for automation of processes: premium subscriptions to private channels, support bots for answering, etc. Bots have become a new opportunity to receive information from followers and communicate with them directly. [10] Telegram has the best reactions with custom emojis.

A technical problem of Telegram is its comments system, which means an uncomfortable list of comments. One comment is good, because it can contain images, audio messages and people can put any available emoji, including custom, if it's not disabled. But Telegram doesn't provide any sorting system for the list of comments, only an infinite stream of comments sorted by date, and you can't see all replies, however the application at least shows the original comment that has this

reply. One comment can have different emojis. [Figure 2.3](#) shows the structure of a Telegram comment



*Figure 2.3: Structure of Telegram comment with reply and multiple emojis.*

Telegram became a very popular platform among manipulators and scammers. Telegram has the best system of emotions, because the standard pack includes: likes, dislikes, smiling, sadness, anger, crying, etc, and also users can create their own custom emojis: from cars and words to rabbits and ponies. Author of the channel can add some emoji, but he also can disable some emojis, or even all of them. And in this situation Telegram becomes the most non-transparent social media where it is possible to know only the number of views and no more. In contrast to YouTube, disabling emojis and comments are usual and normal practice, because Telegram has a chat format for short information. Users don't trust YouTube video with disabled comments and even disabled likes, but the Telegram channel has no problems with this. Besides the technical side, the user-experience side is very important. Because every social network is a network of users, technologies are only means of communication between people. And manipulators with disabled emotions have more chances in Telegram than in YouTube.

### 2.2.5 Common traits of social networks

After analyzing these 4 social networks, some trends and frequently used practices can be identified:

#### 1. Only likes

These sites prefer to use only likes. The main reason is resistance to negativity. If potential haters have no opportunities, they can't hate someone, or this hate has less impact. But sometimes dislikes or alternative reactions must exist. Users can prevent bots' comments through the report system because their comments usually are spam. But paid commenters' comments are harder to detect. These comments look like usual ones, but paid commenters have a goal to persuade users or mislead them, in most cases these comments are provocative, and dislikes can expose this.

#### 2. Non-transparent information.

These sites prefer to hide information. Sites do not consider transparency necessary, because the main goal and meaning of their existence is commercial profit. Also people don't need transparency and don't ask for it. Transparency is an advanced feature that is unusual for the average users, who are used to trusting themselves and short statistics (number of likes). However, transparency is needed for advanced users, advertisers and journalists, which can analyze statistics and draw conclusions.

Why do people dislike something? There are many reasons, but the main 4 reasons were selected:

1. Scam: illegal casinos or financial pyramids, this reason is illegal, and this content will be deleted.
2. Lie of the author. Author can promote overpriced low-quality products or some famous person. It's legal, but it's a lie because people trust their favorite blogger, but this blogger can manipulate own followers.
3. Users don't like it. An author can change their own style, and people, who love the consistent style, don't like this. Or the new author's experiment is bad. But through dislikes the author can understand what the problem is and can fix this. The author can read comments, but it doesn't show the full size of the situation and requires much time.
4. Default hate. When people hate someone or this person doesn't like their favorite blogger: conflicts between famous media persons. Existing companies use this argument for non-transparent data: For example, YouTube writes: "We want every creator to feel they can express themselves without harassment." [\[11\]](#)

### 2.2.6 Comparison of social networks

This table was created after analyzing these 4 social networks. The table shows every important and relevant factor. Grades are relative: 1/5 for the comment system in Instagram means that only the comment system of Instagram is worse than YouTube, Twitter, Telegram. Telegram's 5/5 doesn't mean that the comment system in Telegram is the best in the world, but the system is the best among the other 3 networks.

In addition, low grades don't mean that a network cannot be used for analysis. These applications have many users and this relevance has its own serious reasons, every application has strong points which are good practices. [Table 2.4](#) shows the final comparison.

	Instagram	Twitter	YouTube	Telegram
Comment system	1/5	2/5	5/5	3/5
Post system	4/5	3/5	5/5	3/5
Reactions (including likes)	1/5	2/5	3/5	5/5
Transparency of data	1/5	2/5	4/5	3/5
<b>Final grade</b>	<b>1.75/5</b>	<b>2.25/5</b>	<b>4.25/5</b>	<b>3.5/5</b>

Table 2.4: Comparison of social networks

## 2.3 Strategic analysis and planning

This chapter contains three different analytic methodologies: [SWOT](#), [PEST](#), 5F

### 2.3.1 SWOT analysis

SWOT analysis is a technique of strategic planning and management that helps to evaluate the strengths, weaknesses, opportunities, and threats of a project. SWOT analysis helps to find the best way to use the available resources and capabilities, to avoid risks and to maximize benefits. Strengths and weaknesses are internal conditions that depend only on the company. Opportunities and threats are the external environment of an enterprise [12]. [Table 2.5](#) shows SWOT analysis of the project

<b>Strengths</b>	<b>Weaknesses</b>
<ul style="list-style-type: none"> <li>● Low cost of development</li> <li>● Bold and innovative decisions</li> <li>● Clean community</li> </ul>	<ul style="list-style-type: none"> <li>● Non user-friendly interface</li> <li>● Lack of advertising</li> </ul>
<b>Opportunities</b>	<b>Threats</b>
<ul style="list-style-type: none"> <li>● Exploration of new markets</li> <li>● Transition of personalities from other platforms</li> </ul>	<ul style="list-style-type: none"> <li>● Regulation changes</li> </ul>

Table 2.5: SWOT analysis

### 2.3.2 PEST analysis

PEST analysis is a framework of macro-environmental factors used in strategic management. Standard basic PEST analyzes four factors: political, economic, social and technological. [13] Political factors analyze the role and influence of the

government in the economy and the current political situation. Economic factors include the inflation rate, economic growth, taxes, prices. Social factors analyze the population and socio-cultural aspects. Technological factors include the technological level of the society: automation, innovation, and importance of technologies in public life, technological capabilities of the country and the business. [Table 2.6](#) shows the PEST analysis of the project.

PEST analysis is unique for each society and country. The project focuses on Czechia and EU countries.

<b>Political</b>	<ul style="list-style-type: none"> <li>● Czechia is a member of the Europe Union</li> <li>● Political situation in Czechia and EU is stable</li> <li>● However EU regulates many aspect of the Internet</li> </ul>
<b>Economic</b>	<ul style="list-style-type: none"> <li>● Inflation is medium</li> <li>● Overall economic situation is stable</li> </ul>
<b>Social</b>	<ul style="list-style-type: none"> <li>● Social networks and internet communication has significant role in public live</li> <li>● Social situation is stable</li> </ul>
<b>Technological</b>	<ul style="list-style-type: none"> <li>● Czechia and EU are high-urbanized societies with high availability of devices</li> <li>● Almost each person and household has the Internet</li> <li>● Technological situation is stable</li> </ul>

*Table 2.6: PEST analysis*

### 2.3.3 5F analysis

Porter's Five Forces framework analyzes five competitive forces that shape each project and industry. These five competitive forces affect the status and development of an industry. [\[14\]](#)

- **Threat of substitutes:**  
Main substitutes for social networks are messengers. Threat of substitutes is real.
- **Bargaining power of suppliers:**  
The project is independent from external suppliers.
- **Bargaining power of customers (buyers):**  
Target market is the default people and influencers. Customers have alternatives for communication and creativity in social networks, but only a few social networks provide transparent information.
- **Competitive rivalry:**  
Many competitive social networks exist, but the project has unique features that can be successful among users.

- **Threat of new entrants:**

New entrants will not become a barrier, because the social network has its own unique community. However, social network is a profitable idea. Threat is real, but not critical.

## 2.4 Choice of technical solution

The beginning of each project is understanding the problem because technologies must solve the problem, but not vice versa.

JavaScript is a programming language. JavaScript is a fundamental piece of modern web apps that's used to construct a variety of systems, including web apps with sophisticated user interfaces. [15] JavaScript is a universal language that supports [OOP](#) principles, but using OOP isn't required. The main reason for selecting [JS](#) is the universality and efficiency of the language, which can be used for everything: from web programming (frontend and backend) to artificial intelligence. GitHub statistics shows that JavaScript has been in 1st place for the last 9 years. A large amount of JavaScript frameworks exist for frontend: Angular, Vue, React and for backend: Express, Nest [16].

The project uses a modern standard application style: client + server application as most social networks. [17] Client and server are independent applications. Client is frontend, server is backend. Nowadays, two software engineering paradigms dominate modern enterprise application development: monolithic and microservice-based architecture. A microservice architecture decomposes a business domain into small pieces implemented by autonomous, self-contained, loosely coupled, and independently deployable services. [18] Every service solves its own task. The main advantage of a monolith is that it is easy to write. Monolith doesn't require any extra tools like Kafka, Redis, etc. But monolith is harder to scale, because all project pieces are closely related. Microservice-based architecture is a more modern style, that can solve problems with scalability, availability and maintainability. However, microservice-based applications have their own challenges, like increased consumption of computing resources and maintaining data consistency and transaction management across microservices. But for modern large projects like a social network microservices are the best solution. A social network is a complex and extensible system.

Microfrontend style is based on microservice style, but only for frontend. [19] Every microfrontend is a small application that is responsible for its own task. Microfrontends have the same advantages and disadvantages as microservice-based architecture. Microfrontends were selected for the frontend part of the project. Differences between architectural styles are shown in [Figure 2.7](#).

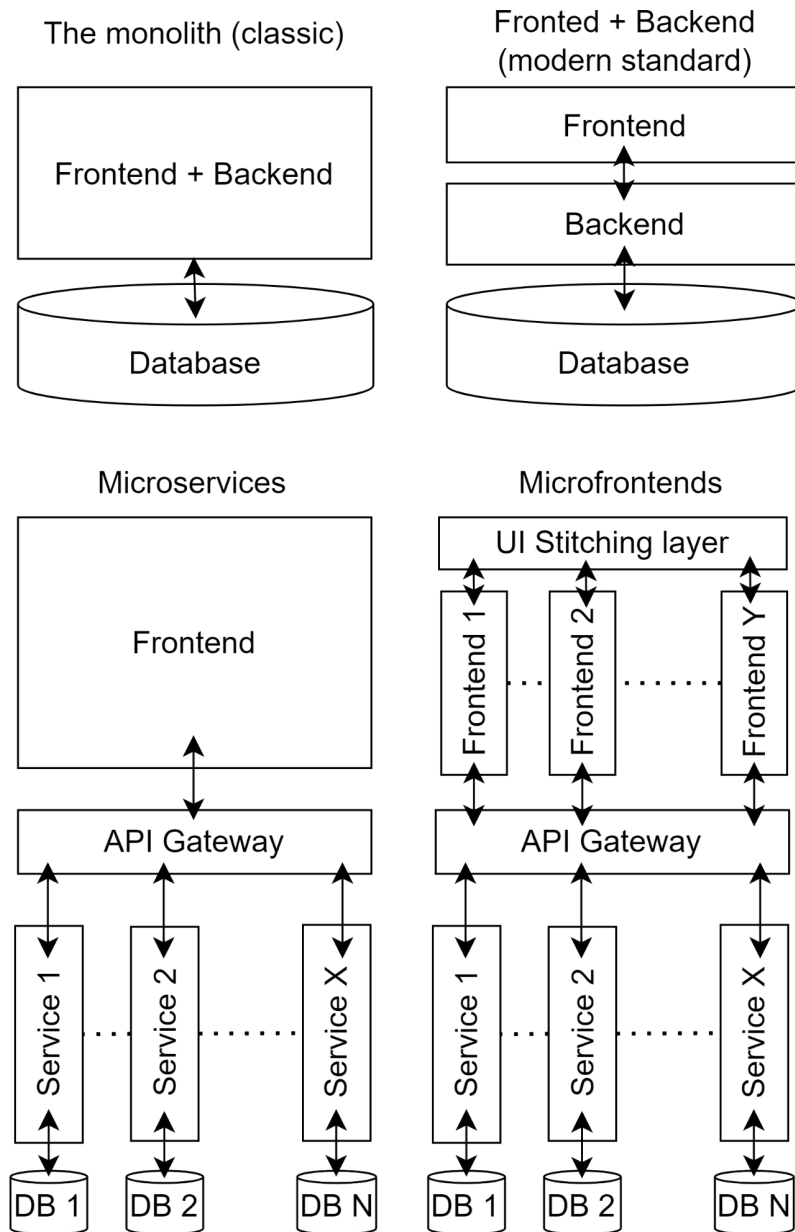


Figure 2.7: Architectures of websites

### 2.4.1 Frontend

Frontend development is one of the main aspects of web applications, as it determines how the user interface is designed and implemented. However, frontend development can also pose many problems, especially for large and complex applications that require frequent updates, scalability, and performance.

JavaScript frameworks are an important feature of frontend development because they provide tools for creating scalable, dynamic applications. Frameworks aim to accelerate the process of developing a website. [20] JavaScript has many frameworks and libraries for the Frontend. Three main technologies for frontend are

Angular, Vue and React. In this subsection, some basic information about the libraries and frameworks is described.

#### 2.4.1.1 Angular

AngularJS is an open-source JavaScript framework for creating front-end single page web applications using TypeScript, a superset of JavaScript that adds static typing and other features. [21] TypeScript is not mandatory for Angular, but is recommended. Angular was created by Google in 2010 and is maintained by a large community of developers. Angular uses a component-style architecture, where each component consists of a template, a class - TypeScript, and metadata (decorators). Angular also provides different features such as data binding, dependency injection, routing, forms, testing, and more.

One of the main advantages of Angular is that it is a complete and opinionated framework that offers a consistent way of developing [SPAs](#). Angular also benefits from the TypeScript language, which helps to catch errors at compile time and improve code quality and readability. Angular is suitable for large and complex applications that require high performance and scalability.

However, some of the drawbacks of Angular are that it has a steep learning curve, a large bundle size, and frequent breaking changes.

#### 2.4.1.2 Vue

Vue is an open-source progressive framework for building web user interfaces based on JavaScript. Vue was created by former Google engineer Evan You in 2014, and is supported by a small core team and a large community of contributors. [22] Vue uses a component-style architecture, where each component consists of a HTML template, [CSS](#) styles and [JS](#) script. Vue also provides different features such as data binding, directives, transitions, routing, state management, and more.

One of the main advantages of Vue is that it is lightweight and flexible, which makes it easy to use. Vue is suitable for small to medium-sized applications that need a simple and elegant UI.

One of the main disadvantages of Vue is that Vue has less community and less demand in the labor market than React and Angular. It means that Vue has less supporting libraries and frameworks for solving problems, and that it has some compatibility issues with older browsers.

#### 2.4.1.3 React

React is an open-source library for building user interfaces based on JavaScript or [JSX](#), a syntax extension that allows writing [HTML](#)-like code in JavaScript while using React features. React was created by Facebook, and was first launched in



2013. [22] React uses a component style, where components are reusable pieces of the [UI](#) that have their own logic, styles and state. React, like Vue, uses its own virtual [DOM](#), which is a representation of the real DOM in memory, to efficiently update the UI when the state changes. React DOM compares the new values to the previously stored values and only re-renders if there is a difference between the two states. [23] React is compatible with TypeScript, which is optional.

One of the main advantages of React is that it is declarative, which makes it easy to create interactive and dynamic UIs. React is suitable for small to large-sized applications. React also has an active community that provides many tools and libraries to enhance its functionality, such as React Router for routing, and Next.js for server-side rendering. However, some of the disadvantages of React are that it is only a UI library, which means that it is necessary to choose and integrate other technologies for the rest of the application stack, and that it has a high initial setup cost.

#### 2.4.1.4 Summary

In conclusion, Angular, React, and Vue are three main libraries/frameworks for building client-side JavaScript applications. [24] Each of them has its own advantages and disadvantages. React was selected as the core library for frontends, because React has a solid ecosystem of tools and libraries from an active community for solving different tasks. React was created by Meta company, and is used for Instagram, Facebook, and Twitter - modern social networks. React is a suitable choice for the social networks because React uses a component style: every part of the interface can be separated into one component, every button, every image, and every block of content. The project consists of posts and comments - reusable components. [Table 2.8](#) shows a comparison of these 3 technologies.

	React	Angular	Vue
<a href="#">NPM packages</a>	251641	70067	82301
Stack Overflow questions	467175	299943	105656
Weekly downloads on NPM web	21385508	8611884	3960281
LinkedIn job positions (Czechia)	1036	793	558
<a href="#">Using by developers (by Stack Overflow)</a>	42.62%	20.39%	18.82%
Jobs.cz job positions	133	68	42

*Table 2.8: Comparison of client-side JS technologies*

Relevance of React allows one to find a programmer or a technical solution for almost every problem. According to information from npm, LinkedIn and StackOverflow, React is the most discussed among other JS technologies.

## 2.4.2 Backend

Backend development is the second important aspect of web applications, as it determines how the business logic is designed and implemented. The server is responsible for centralized work with data of users and storage. Frameworks work with data, the database is the storage. Every microservice has its own small database for its own part of data.

Databases can be relational and nonrelational. Relational databases use SQL. SQL - Structured Query Language - is a language for working with relational databases, which consist of a set of tables where the data are classified by category and type. SQL - typed databases have fixed and predefined schemes. The main benefit of relational databases is stability.

NoSQL databases are non relational. NoSQL databases can be document - typed, graph or key - values. Non Relational databases have dynamic schemes. The main benefit of non relational databases is flexible storage [25].

PostgreSQL was selected as the [RDBMS](#) for all services of the project. PostgreSQL is open-sourced, supports the transaction function and data integrity checking. [26] PostgreSQL is suitable for the social network because SQL is stable. However, NoSQL databases also can be suitable for social networks, because NoSQL is more scalable and is good for BigData. Backend can interact with databases directly or through [ORM](#) tools. Prisma was selected as the ORM for the project.

JavaScript is a language for Backend, not only for Frontend. It's possible due to NodeJS, which is based on C++ and allows the use of JS for working with hardware. Two main JavaScript frameworks for the backend are Express and Nest. In this subsection, some basic information about the frameworks is described.

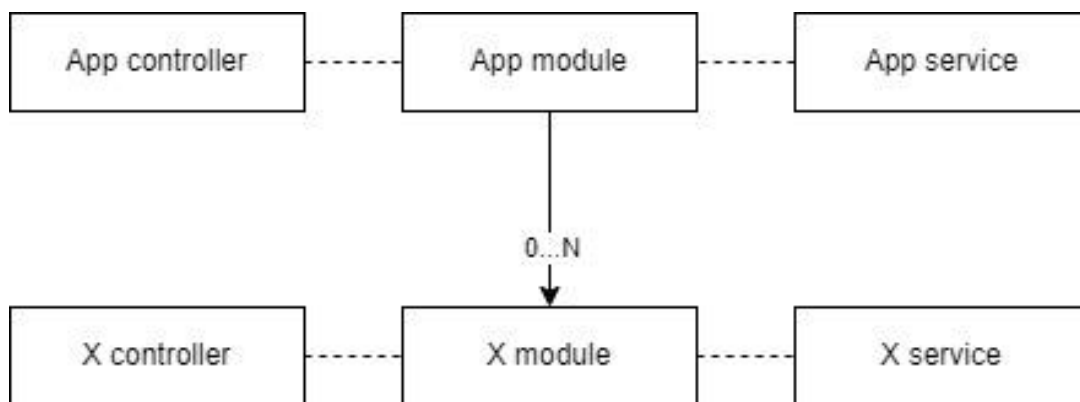
### 2.3.2.1 Express

Express is a Node.js minimalistic and flexible open source web application framework that provides a set of features that extends a standard NodeJS. Express [JS](#) has no specific structure, it has a free architectural style and code organization. [27] Express JS extends NodeJS through its own features, such as routing, error handling, middleware, and template engines. Many JS frameworks are based on Express: Blueprint, NestJS, Locomotive, Sails.

### 2.4.2.2 Nest

Nest is a progressive Node.js framework for building highly scalable server-side applications. [28] It is based on TypeScript and JavaScript. Nest is used over default Express. Nest uses Express and extends it. One of the advantages of Nest is that

anything supported in Express is also supported in Nest. Nest is inspired by Angular and uses Angular - like syntax with annotations and components. Nest has a modular structure, which is shown in [Figure 2.9](#).



*Figure 2.9: NestJS component structure*

Every module is microservice-like, Module exports dependencies from libraries and other modules, and imports objects and functions which can be used by other modules. Each module can have its own controller and service. A service is responsible for business logic. A controller defines the [API](#): endpoints. App module is the core module of the application.

#### 2.4.2.3 Summary

NestJS was selected for the project. NestJS is an Express based framework with its own structure and features. Express is faster, but Nest provides more tools and features for solving the problems. Also the Nest syntax with annotations allows for a more usable configuration.

Nest provides a structured architecture that is good for large and scalable projects. Nest is compatible with microservice architecture. Each service works through the core app module.

Each microservice can communicate with another through a message broker. A message broker sends messages (commands/requests) from one microservice to another. Apache Kafka uses publish - subscriber model for asynchronous communication. Apache Kafka was selected as the message broker because Kafka is compatible with Nest, and Kafka systems are decoupled; hence they are easier to maintain and scale. [\[29\]](#)

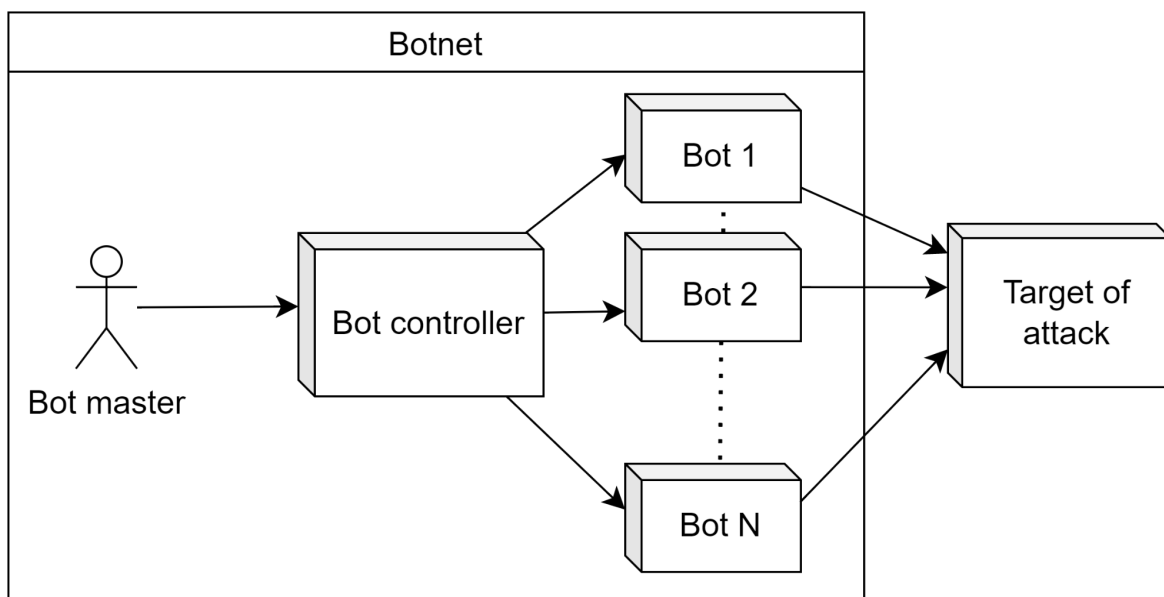
#### 2.4.3 Manipulation

Social networks have to be resistant against bots, paid commenters and scam information.

### 2.4.3.1 Bots

Bots are special fake accounts, often they are empty and without photos and additional information. A set of bots creates a botnet. A botnet is a set of Internet-based computers under a common controller, the architecture of the botnet is shown in [Figure 2.10](#). Large groups of bots (botnets) are controlled by the same entity, called a bot master, acting behind the scenes in a command-and-control fashion. There are many ways in which social bots can disrupt or influence online discourse, such as spamming hashtags, scamming twitter users, and astroturfing. [\[30\]](#)

Traditional methods of detecting bots include feature based and text-based methods. Feature-based methods involve extracting features from an account such as word distribution in a text, posting rates, or followers. Text-based methods utilize frameworks in natural language processing when examining user profiles and posts. Feature-based approaches do not perform well as recent bots can avoid detection by emulating behaviors of real users. A text-based approach to detecting bots is vulnerable to newer bots whose posts also contain real posts from human users. Graph-based methods involve constructing users as nodes and relationships among nodes as edges for the initial graph data, such as relational graph convolutional networks. [\[31\]](#)



*Figure 2.10: The architecture of a botnet*

Bots are used for manipulation of numbers of likes/followers or for creating fake activity. These actions have three main goals:

1. Changing public opinion. If people see a big number of likes, they are likely to believe it. Big number of dislikes can undermine trust to comment or post, or even lead to deleting this.
2. Abusing the recommendation system. Big number of likes can boost a post or person for trends or personal recommendation.

3. Creating a good picture for advertising. Price of adverts depends on different parameters, such as likes, views, followers. If bloggers or media are popular with many followers and views, they earn more money. Advertisers assess price for ads based on numbers and decide for themselves.

The main problem of bots is suspicious activity. A real human has own account with subscriptions, messages, posts and personal recommendations. A human is connected with own account, an account is a digital identity. The behavior of bots differs from real accounts. Botmaster should create many accounts without free time for customization and setting. An account with 1000 subscribers, a five-years history of activity, has equal like or another reaction as an account, which was created two days ago for boosting.

#### 2.4.3.2 Paid commenters

Paid commenters are accounts, which are created by special factories. A factory worker can manage many accounts (from 1 to 15), in contrast to bots, paid commenters are not automatic, in most cases workers login and directly write each comment or post. A factory is shown in [Figure 2.11](#).

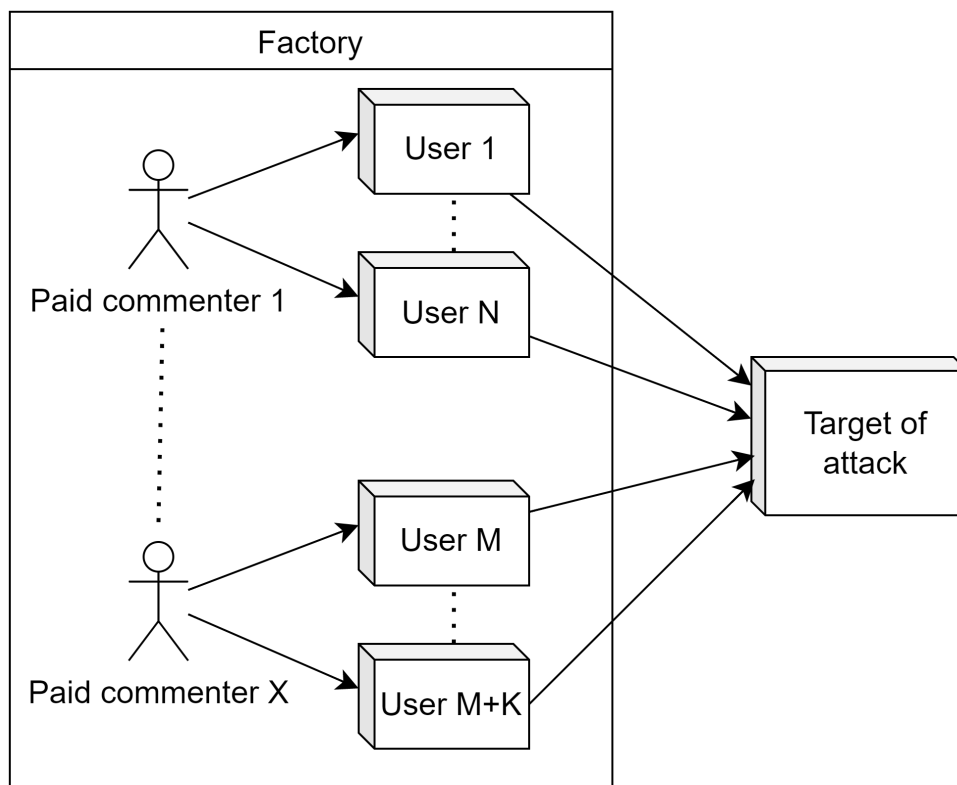


Figure 2.11: The architecture of a factory

Main goals of paid commenters are provocations and disinformation. [32] It's possible if information cannot be checked quickly because nobody has free time for deep fact-checking and analysis. When a paid commenter writes some comment with controversial information, people can see replies and compare their positions.

Nevertheless, comments and a number of likes have a big impact on the opinion of people. A person doesn't understand who is right, a human is a social creature, and it's normal if a person believes the majority. If data is transparent, a paid commenter has no chances for misleading, because people can see the full situation easily.

The behavior of a paid commenter differs from a usual user, because a paid commenter must write many comments, and these comments are similar to each other. Real users can be active, but their comments are different and natural based on the situation.

The project shows all comments of users, because it's public information and each user can try to find and collect all comments independently, but it requires a lot of time. This project doesn't need any extra services for providing data (such as Socialblade, etc), standard providing data is enough, also this includes creating graphs.

### 2.4.3.3 Scam

Authors can lie and manipulate the facts for their own profit. This problem has the same reasons as paid commenter's manipulation - lack of information. A person doesn't see many dislikes, which exist because the author lies, and believes the author. Providing the information is possible on the technical side, but many social networks provide less information than they can. But full information allows the users to see the full situation and analyze this for themselves. [Figure 2.12](#) shows an example of full and transparent statistics of the poll with a chart.



Figure 2.12: Full information of votes on vk.com site

### 2.4.4 Conclusion

Technical implementation of the project isn't hard or revolutionary. The Internet suggests many technical solutions and working methods. The main reason for manipulation is the reluctance of people. They think that non-transparent data is their advantage, but it's not right.

## 2.5 Requirements

Project requirements are part of software engineering. Requirements describe what the project should be able to do. Requirements are used for business, although the project is not commercial, nevertheless these requirements help to simulate the real project.

### 2.5.1 Functional requirements

#### **Business Goal (BG)**

Create a social media network with advanced statistics for 5 – 20 million users.

#### Business Requirements (BRQ)

##### 1. **Posts**

Users should be able to create/read/update/delete posts and add optional photos and a video to their posts because this will allow them to create the content for the site. Every post also can have a poll with different options.

##### 2. **Comments**

Users should be able to write comments for posts and options and they can communicate with other users.

##### 3. **Users' role and bans**

Admins should be able to manage rights of users and appoint moderators because admins need to maintain public order.

##### 4. **Statistics**

Users should be able to see statistics of post reactions, votes for polls and subscribers, so everyone can view results, understand the situation about public opinion.

#### Functional Requirements (FRQ)

The word "manage" means CRUD operations

Admins have all rights of moderator, moderators have all rights of user

##### 1. **Writing posts (BRQ 1)**

Users can manage their posts with optional photos and a video.

- 2. Open opinion (BRQ 1, BRQ 4)**  
Likes/dislikes for posts cannot be disabled.
- 3. Attaching poll (BRQ 1)**  
Users can attach a poll to their posts while creating them
- 4. Reactions (BRQ 1, BRQ 2)**  
Users can react to posts and comments (like/dislike)
- 5. Adding options (BRQ 1)**  
Users can add one option for a poll if it's allowed
- 6. Voting for options (BRQ 1)**  
User can vote for any option of any post and only once
- 7. Comments (BRQ 2)**  
Users can manage their comments
- 8. Replies on comments (BRQ 2)**  
Users can reply to comments
- 9. Viewing a post statistics (BRQ 1, BRQ 4)**  
Users can watch the statistics of the poll and post reactions
- 10. Subscribes and subscribers (BRQ 3)**  
Users can subscribe to other users and have their own subscribers
- 11. Viewing a user's statistics (BRQ 3, BRQ 4)**  
Users can watch the statistics of the user's followers
- 12. Spam reporting (BRQ 3)**  
Users can report spam/misinformation
- 13. Requests about moderator (BRQ 3)**  
Users can send a request to become a moderator when they want to become one
- 14. Ban users (BRQ 3)**  
Moderators can ban or mute any user for a selected time
- 15. Ban moderators (BRQ 3)**  
Admins can ban any moderator or user
- 16. Consideration of request (BRQ 3)**  
Admins can accept/deny a request to become a moderator
- 17. End of moderator rights (BRQ 3)**  
Admins can stop moderator rights for any moderator
- 18. Removing comments and posts by moderator (BRQ 1, BRQ 2, BRQ 3)**  
Moderators can remove any non-moderator posts and comments



## 19. Removing comments and posts by admin (BRQ 1, BRQ 2, BRQ 3)

Admins can remove any posts and comments

Non – functional requirements (NRQ)

### 1. Integration into browsers

The site is available and works equally for different browsers (Firefox, Edge, Chrome, Opera)

### 2. Compatibility (Responsive design) with mobile devices

The site has an adaptive design for all devices and displays

### 3. Password protection

The site will encrypt passwords for effective protection

### 4. User–friendly interface

The site has a user–friendly and intuitively understandable interface

## 2.5.2 Use cases

Use cases help to model all actions of users and distribute these actions between different user roles. Use - case diagram complexity has a major impact on the quality of the resulting system. [33] The project has three parts in the UC diagram: posts, comments and user relationships.

### 2.5.2.1 Posts

A post is one of two main entities of content. The social network consists of a set of posts and comments. [Figure 2.13](#) shows how users can interact with posts. A user can create posts and make CRUD actions with their own posts. User can create a post with optional photos, a video and a poll that contains options, then the user must allow or forbid adding new options. If adding of options is allowed, any user can add a new option once. Also users can react to any post, including their own posts. If a post or some option has inappropriate content such as spam, misinformation, hate speech, moderators and admins can delete the post or the option. Users can report about violations, and admins and moderators can accept or deny these reports.

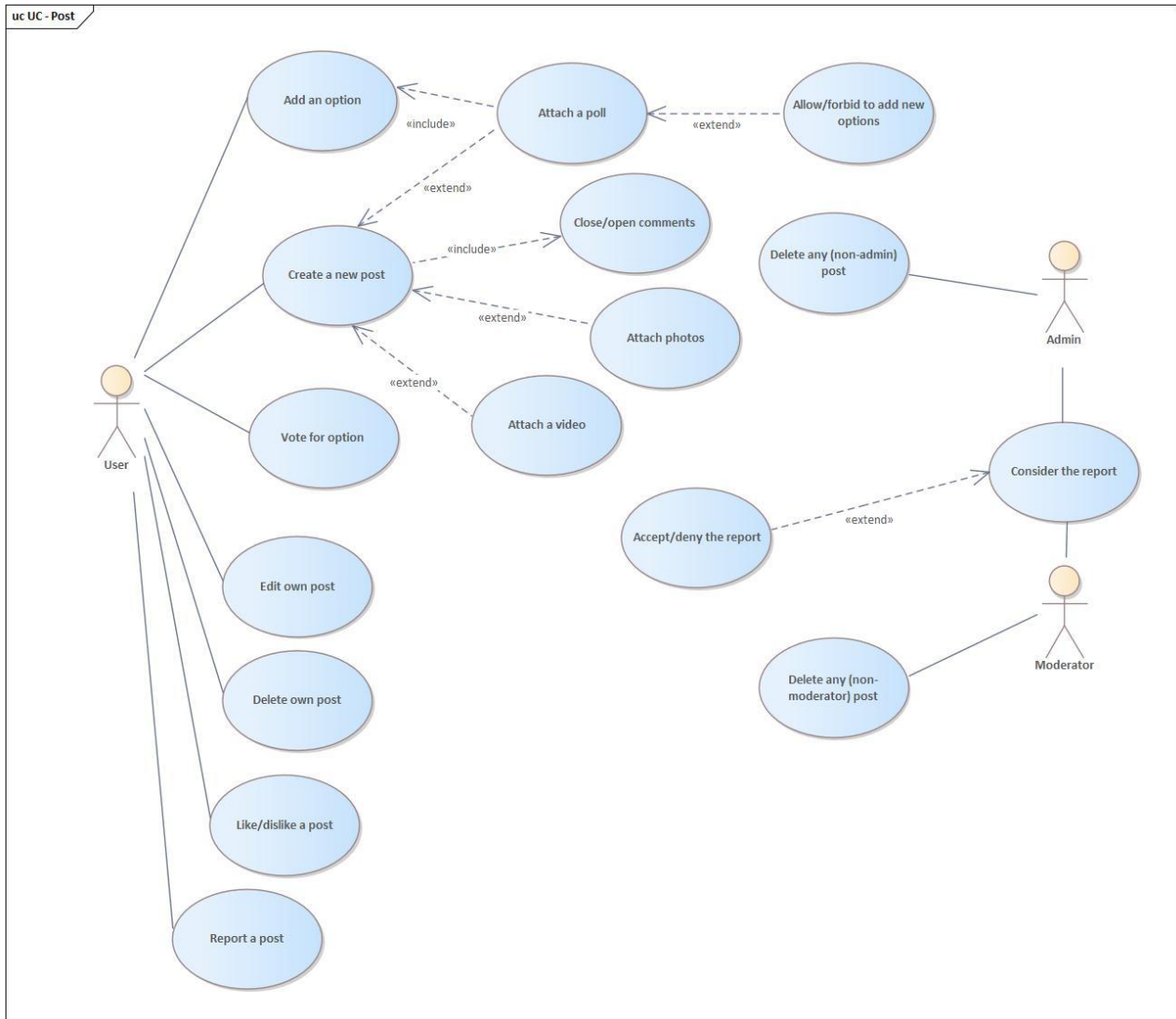


Figure 2.13: Use case model of posts

### 2.5.2.2 Comments

A post is the second of two main entities of content. A comment can be written under a post or an option. [Figure 2.14](#) shows how users can interact with comments. User can write their own comments and make CRUD actions with these comments. Users can react to comments and reply. If a comment has inappropriate content such as spam, misinformation, hate speech, moderators and admins can delete the comment. Users can report about violations, and admins and moderators can accept or deny these reports.

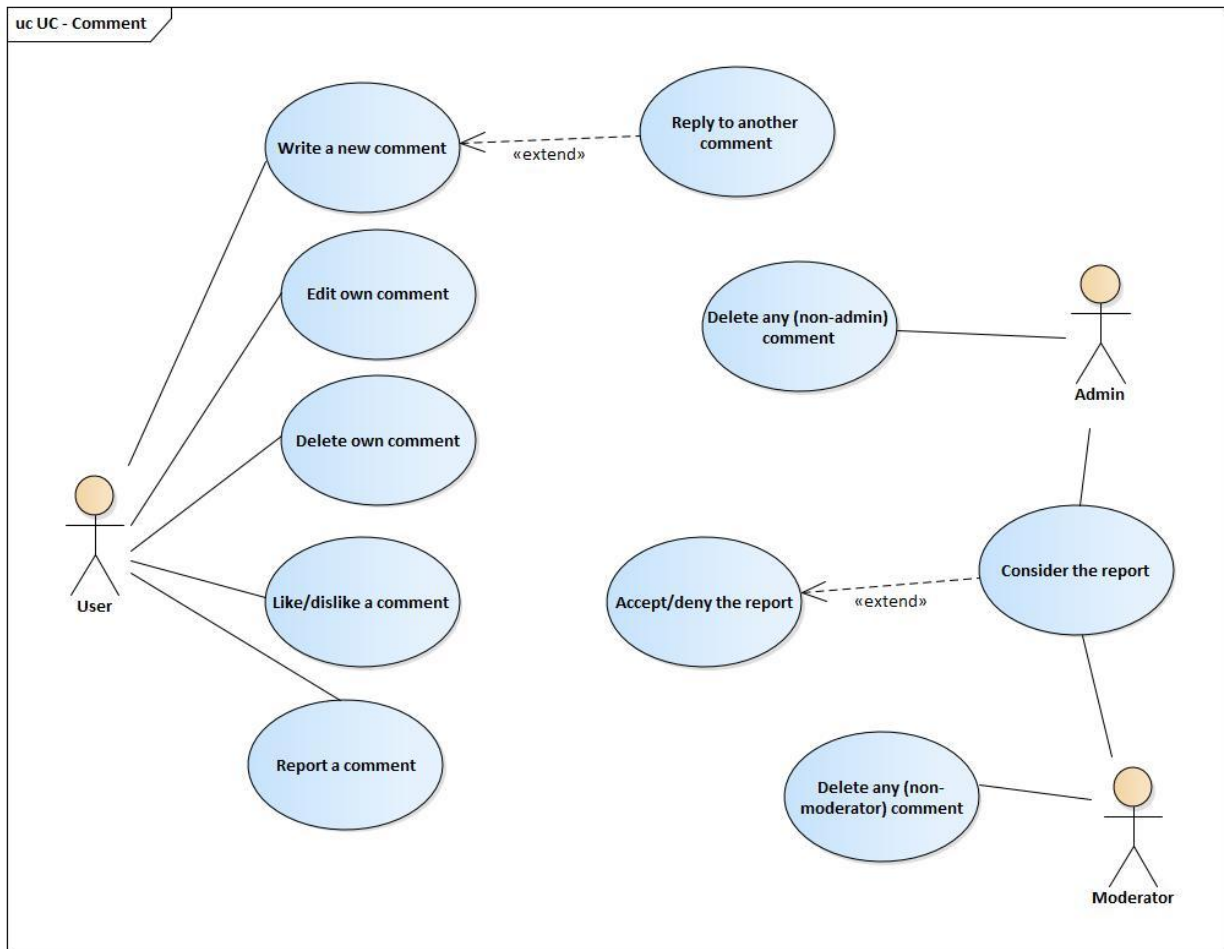


Figure 2.14: Use case model of comments

### 2.5.2.3 Users and roles

Default users are most users in the system. Users fill information about themselves while registering. Moderators and admins have to maintain order and prevent violations. If a default user wants to become a moderator, this user can send a request. If an admin accepts this request, the user will become a moderator. Admin can suspend moderator rights. Admins are defined by the system and programmers. Admins and moderators can ban users, banned users have no rights, and blocked accounts are saved, but cannot be used. [Figure 2.15](#) shows interactions between different users and hierarchy.

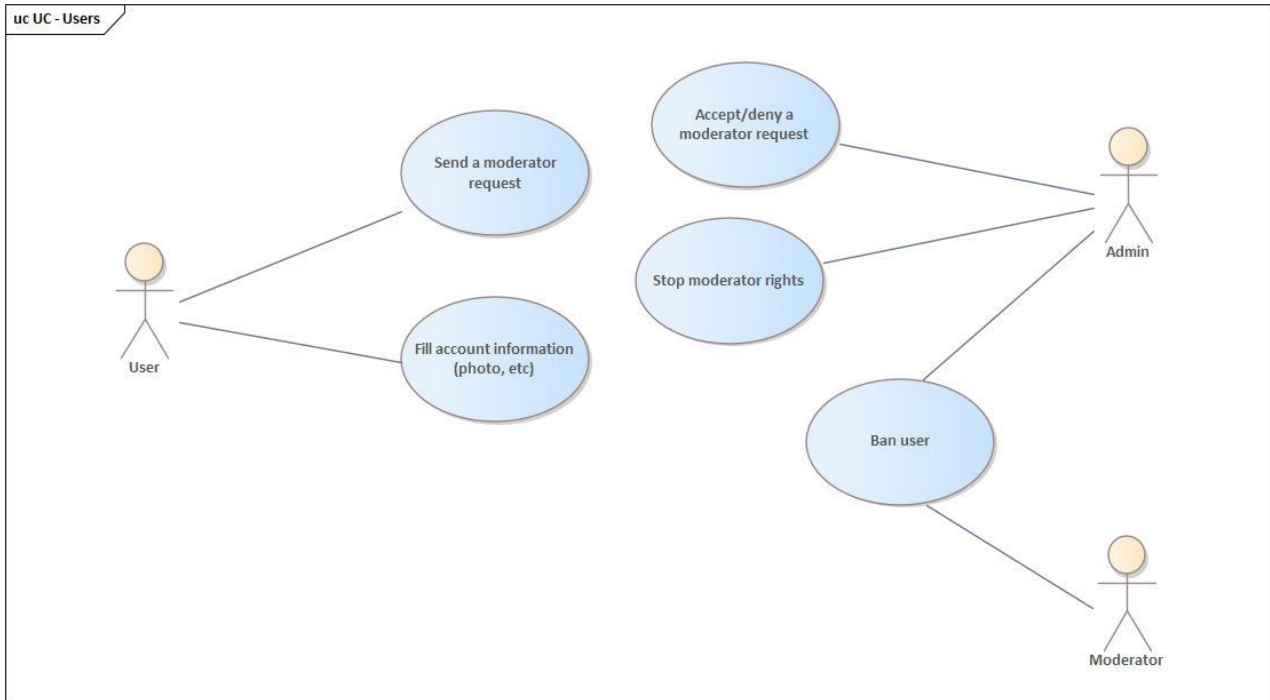


Figure 2.15: Use case model of users' rights

# Chapter 3

## Design

### 3.1 Class diagram

A class diagram is a type of diagram in [UML](#) that shows the structure and relationships of classes in an object-oriented system. Class diagram describes the class structure of a system, attributes of these classes, hierarchy and relations among different classes. [\[34\]](#) The main goal of the class diagram is the representation of the project as an interrelated collection of entities.

Entities are connected by two-way relations:

- **0..1**: A can have a B or not
- **1..1**: A has a B
- **0..N**: A has any instances of B
- **1..N**: A has 1 or more instances of B

These relations work also in the other direction, for example: User can write any number of posts: 1..N, but one post can be written only by one user: 1..1. Class diagram helps to design database. [Figure 3.1](#) shows the class diagram of the project.

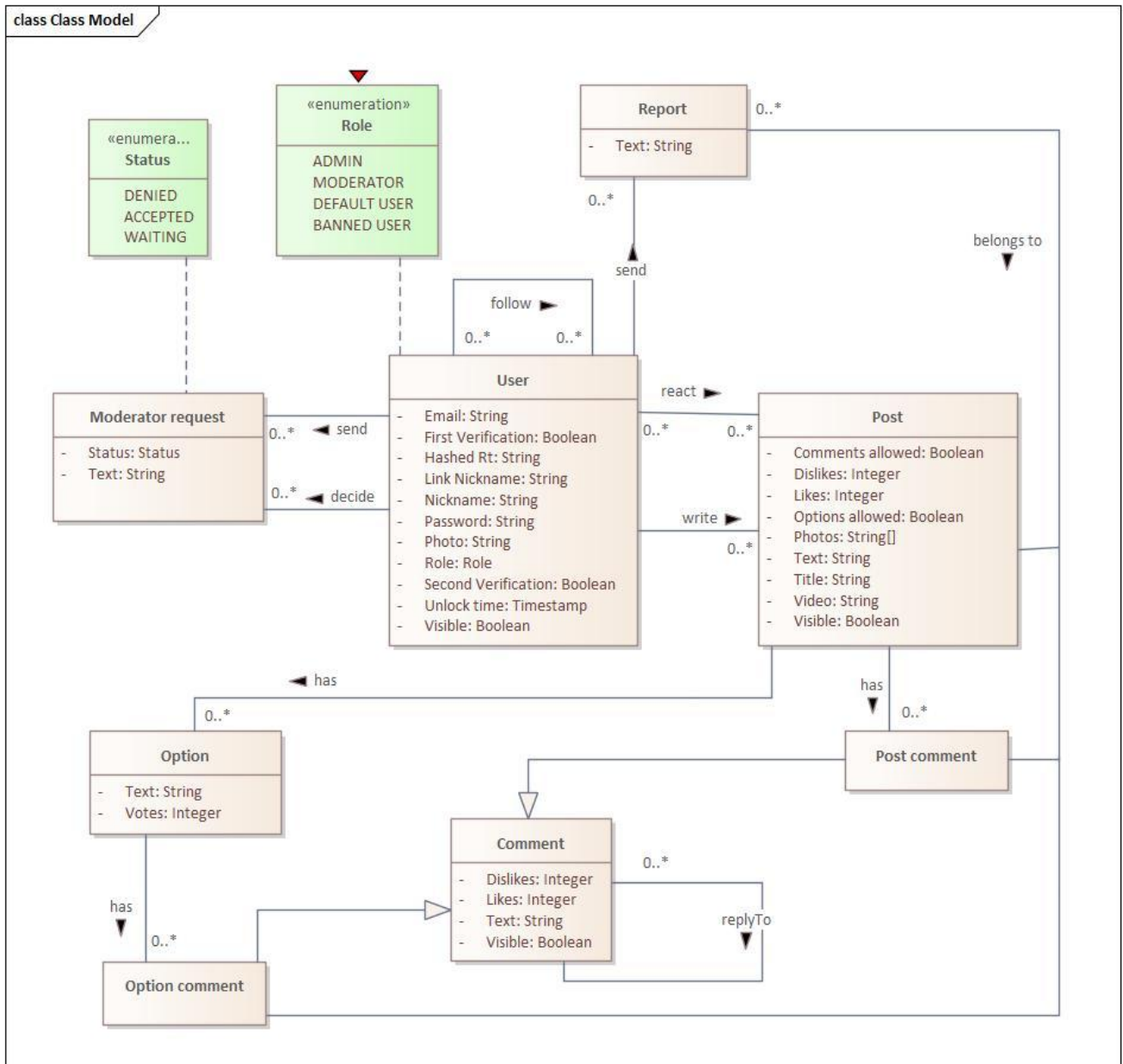


Figure 3.1: A class diagram of the system

### 3.2 Deployment diagram

A deployment diagram is a type of UML diagram that shows how a system is deployed on different hardware or software components, in other words - it is used to represent the deployment view of an application. [35] This diagram allows one to view relationships between every service and connection with the database or relationships between every frontend. The diagram is used to model the physical aspects of an object-oriented system, such as the configuration of nodes and connections. A deployment diagram can help to model the topology of a system, the distribution of components across different nodes, and the communication paths between these nodes.

Frontend is a client, each client works with frontend on their own computer independently. Backend is the centralized remote server which accepts requests from clients and returns responses back. Frontend part consists of four microfrontends on React, then microfrontends communicate with backend through [API](#) Gateway, API Gateway also is one of four microservices, which is responsible for external communication and authentication. Each microservice has its own database in PostgreSQL. [Figure 3.2](#) shows the deployment diagram of the project.

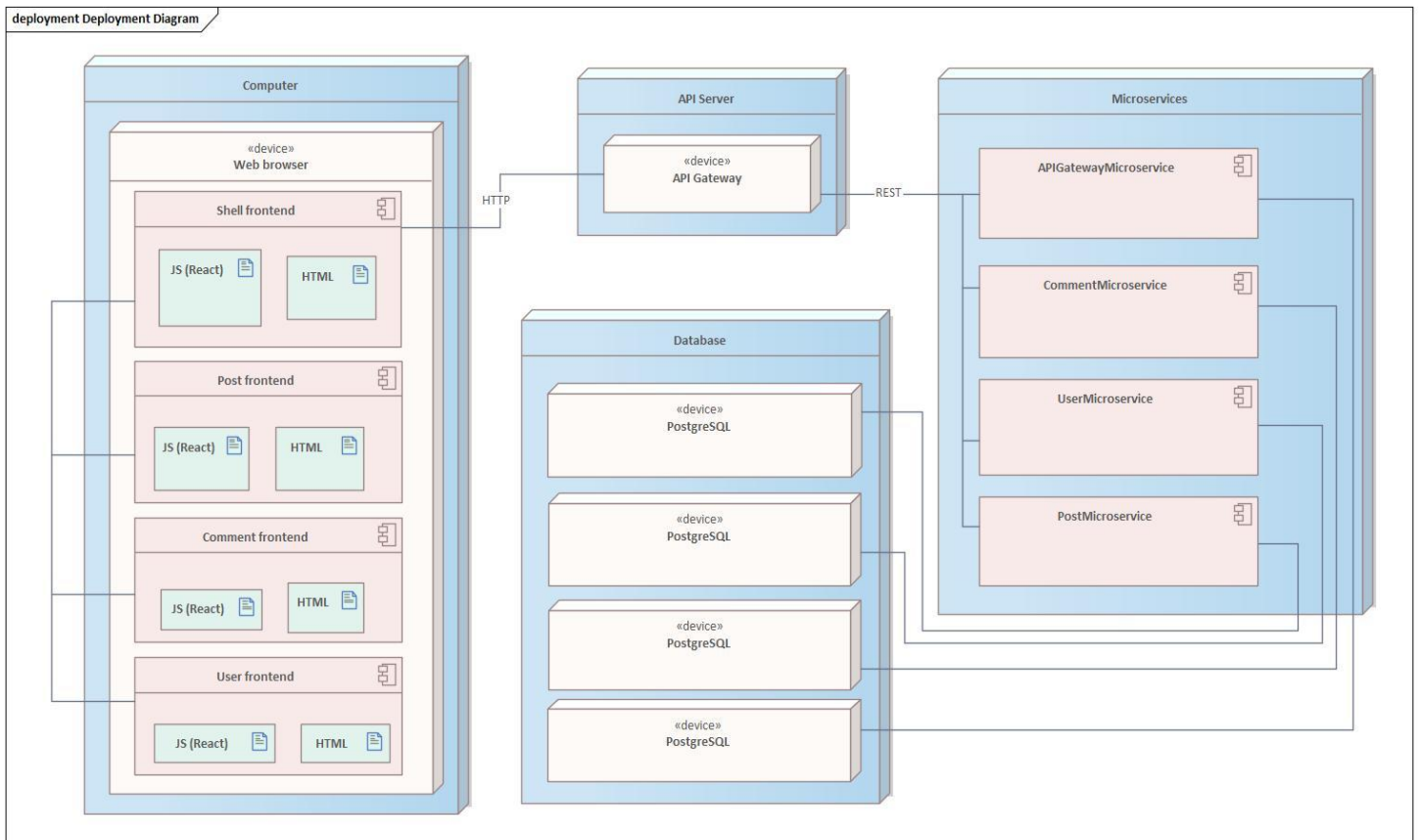


Figure 3.2: A deployment model

### 3.3 UI Design

The creation of UI/[UX](#) design is described in this section. This design was created in Figma. [UI](#) design must show how the system will look like, UI is a visual representation of the layout structure. UX analyzes the behavior of users. [\[36\]](#)

#### 3.3.1 Create a new post

Users can create posts on a special page. This page includes a form with a title, a text, photos, a video and a poll that contains options.

MYSELECT  Write a post

---

**Write a post**

**Title**

**Text**

**Photos**

**Video**

**Poll**

People can add options

People can comment the post

---

+420-123-456-789  
 mail@myselect.com  
 Copyright © 2022 myselect.com, you can answer yoy every wotking day: 08:00 - 20:00

*Figure 3.3: Creating post page*


Text and title are required fields. The maximum number of photos is ten. If options are empty then the post will be without a poll.

Also each page has a menu on the top with the project name “Myselect”, with a search field for posts and users, with a button for creating new posts and a login/logout button. [Figure 3.3](#) shows creating post page.


### 3.3.2 Comments



Each post or option can have comments if this is allowed.





MYSELECT  Write a post 



---


Sort by: Date 

 Person 10:09 04/01/2021 



Hello, It's my comment for reply, Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis condimentum augue id magna semper rutrum. Duis pulvinar. Aliquam erat volutpat. Duis pulvinar. Duis ante orci, molestie vitae vehicula venenatis, tincidunt ac pede. Nullam lectus justo, vulputate eget mollis sed, tempor sed magna. Donec ipsum massa, ullamcorper in, auctor et, scelerisque sed, est. Pellentesque sapien. Integer imperdiet lectus quis justo.

 4 321  987 Reply



 User456 10:09 04/01/2021 

 Person  
Hello, it's my comment for reply, Lorem ipsum dolor sit amet, consectetur adipiscing elitfa afaaf aff...

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis condimentum augue id magna semper rutrum. Duis pulvinar. Aliquam erat volutpat. Duis pulvinar. Duis ante orci, molestie vitae vehicula venenatis, tincidunt ac pede. Nullam lectus justo, vulputate eget mollis sed, tempor sed magna. Donec ipsum massa, ullamcorper in, auctor et, scelerisque sed, est. Pellentesque sapien. Integer imperdiet lectus quis justo.

 4 321  987 Reply

---

 Person 

Hello, it's my comment for reply, Lorem ipsum dolor sit amet, consectetur adipiscing elitfa afaaf aff...

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis condimentum augue id magna semper rutrum. Duis pulvinar. Aliquam erat volutpat. Duis pulvinar. Duis ante orci, molestie vitae vehicula venenatis, tincidunt ac pede. Nullam lectus justo, vulputate eget mollis sed, tempor sed magna. Donec ipsum massa, ullamcorper in, auctor et, scelerisque sed, est. Pellentesque sapien. Integer imperdiei


Write a comment

Figure 3.4: List of comments

Each comment has a nickname of the author, time of creating, text, number of likes/dislikes and reply button. If a comment is a reply to another comment, this comment has a short text of the replying comment. [Figure 3.4](#) shows a list of comments.

### 3.3.3 Followers

Each user can follow (or subscribe) to another user. Users can see their followings as a list and unfollow (unsubscribe) from them. If a user clicks on the nickname of a following user, he will be redirected to the public user page, which contains all posts of this user. [Figure 3.5](#) shows a list of subscribers.

MYSELECT  Write a post 

---

My followings

















	User123	124 567 followers	Unsubscribe	
	User456	390 500 followers	Unsubscribe	
	User789	567 followers	Unsubscribe	
	UserPro	38 901 followers	Unsubscribe	
	VIPerson	97 followers	Unsubscribe	
	XXX	32 489 followers	Unsubscribe	
	Plane123	14 567 followers	Unsubscribe	




Figure 3.5: Followers page

### 3.3.4 User post


The User's page in [Figure 3.6](#) contains a list of all posts of the user. A post has short info about the author, a button for adding new options if it's available. Post shows a text first, then options, photos, and a video if the post has this. Bottom part of the post has likes/dislikes and a button for the number of comments. If comments are allowed, a user can click on the button and he will be redirected to a page with a list of comments.

Clicking on the option means that the user votes for this option, and can see the statistics of all options. If the number of options is more than 5, users can use a search field for finding a preferred option. Clicking on the option after voting redirects the user to the comments page.

 User123    124 567 followers    [Subscribe](#)    

 User123    10:09 04/01/2021     




Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis condimentum augue id magna semper rutrum. Duis pulvinar. Aliquam erat volutpat. Duis pulvinar. Duis ante orci, molestie vitae vehicula venenatis, tincidunt ac pede. Nullam lectus justo, vulputate eget mollis sed, tempor sed magna. Donec ipsum massa, ullamcorper in, auctor et, scelerisque sed, est. Pellentesque sapien. Integer imperdiet lectus quis justo.



 Search

**Option 1**


**Option 2**




**Option 3**

 4 321     987     1 234

 User123    10:09 04/01/2021    

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis condimentum augue id magna semper rutrum. Duis pulvinar. Aliquam erat volutpat. Duis pulvinar. Duis ante orci, molestie vitae vehicula venenatis, tincidunt ac pede. Nullam lectus justo, vulputate eget mollis sed, tempor sed magna. Donec ipsum massa, ullamcorper in, auctor et, scelerisque sed, est. Pellentesque sapien. Integer imperdiet lectus quis justo.






 4 321     987     1 234

Figure 3.6: User's post page

### 3.3.5 Conclusion

The design was created. Design must direct the frontend part and be a manual guide that explains how the frontend looks like. The final version can have differences from the design, but the main goal of the design is solving core problems and overall definition of the user interface of application and user experience from using the application.

# Chapter 4

## Implementation

Implementation of the project was divided into 2 parts: backend + frontend. In this chapter the process of implementation is described.

### 4.1 Backend

Backend part has 4 microservices:

- **Api - gateway:** the main microservice that has authorization, guard protection and accepts all requests from client and redirects them through Kafka to another microservices.
- **User:** works with user's data: full information, followers, users' statistics
- **Post:** works with posts and polls: reactions, options, votes, posts' statistics.
- **Comment:** works with comments to posts and options.

Each microservice was created through the npm console by the command "nest new project\_name --strict", this command creates a project on NestJS with basic configuration and with the use of TypeScript.

Api - gateway, post and comment microservices have a short table of users with main data: email, role, link nickname that provides short author info for authentication/posts/comments without extra requests to the user microservice.

Post, comment and user microservices have similar structure, because they are microservices with their own task. [Figure 4.1](#) shows the structure of the post microservice:

- /dist - an automatic folder for the built application
- /node\_modules - an automatic folder, where the installed libraries are
- /prisma - database schema and history of all migrations - changes of DB
- /src - main folder, which contains source code of the project
- /src/dtos/ - has DTO (Data Transfer Object) files, which define the accepting and requesting form of data
- /src/prisma - prisma service in this big microservice, defines how database must work
- /src/types - defines custom types of data for objects in TypeScript
- /src/app - files with core code of microservice, for post microservice these files contains functions for creating, deleting and working with posts
- /src/main - configuration of microservice and start file for running
- /test - tests
- .env - configuration of database, env is Environment

- .eslintrc.js - ESLint configuration - special plugin for formatting of the code
- .gitignore - which files should be ignored, when project is loaded to GitHub
- .prettierrc - automatic supporting file for configuration
- nest-cli.json - automatic supporting file for configuration
- package.json - list of libraries and dependencies, which will be installed in folder “node\_modules”
- package-lock.json - supporting file for “package.json”
- README.md - automatic file with description of project
- tsconfig.json - configuration of TypeScript and building of project, because compiler works with clean [JS](#), but TypeScript is used for development, after project building TypeScript will be transformed to JavaScript
- tsconfig.build.json - file for building [TS](#) project

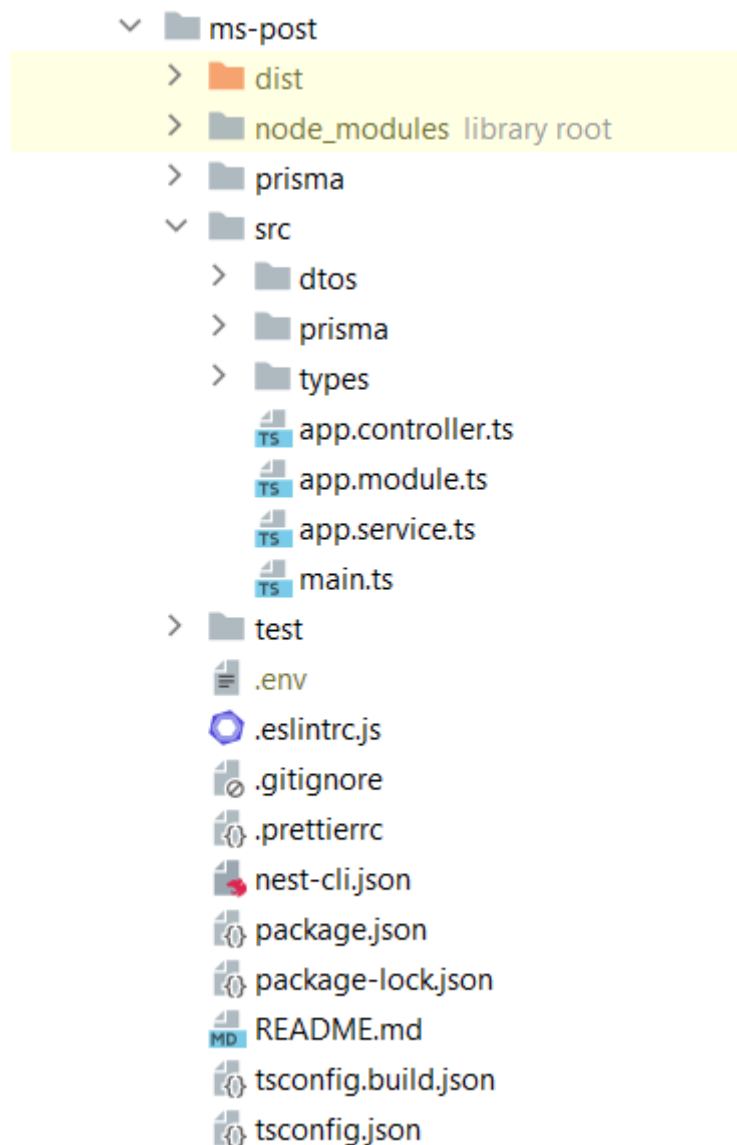


Figure 4.1: Project structure of the post microservice

Api - gateway is the main microservice, and has its own structure, which is described in [Figure 4.2](#). The microservice has differences only in the /src folder

- src/auth - [HTTP](#) endpoints for external interactions with authentication: tokens, login, logout
- src/comment - HTTP endpoints for external interactions with comments
- src/post - HTTP endpoints for external interactions with posts
- src/user - HTTP endpoints for external interactions with users
- app.module.ts - connects all modules
- main.ts - running start file with auth protection

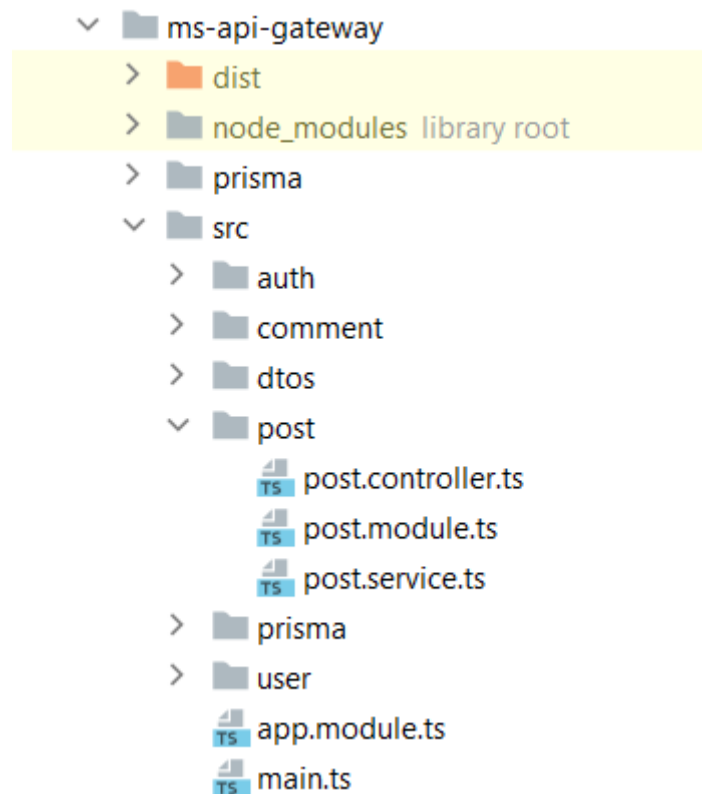


Figure 4.2: Project structure of the api-gateway microservice

#### 4.1.1 Authentication and authorization

Some actions are available without authentication, such as getting a list of all posts. But the majority of actions require authentication. The project provides its own authentication system.

After successful authentication, users get their own access and refresh token. A token contains information, such as in [Figure 4.3](#), where the token has info about id, email, role. When the site gets the token in the header of the HTTP request, the site can decode the token and read info. The access token is used for interactions with endpoints that require authentication, the access token has a small time of life - 45 seconds. But the refresh token has a large lifetime of life - 90 days. The refresh token is used for getting a new pair of tokens and so that the site could check if the

user is signed up or not. Tokens are stored in cookies on the Frontend side in cookies.

Encoded <small>PASTE A TOKEN HERE</small>	Decoded <small>EDIT THE PAYLOAD AND SECRET</small>				
<pre>eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjkwYmRmYTJmLTZjYXV4MmYzLkZiYmZlZTY3OWQ2OSIsImVtYWlsIjoiamFuX25vdmFrQG1haWwzIiwiLCJyb2x1IjoieEVVGQVMVF9VU0VSIiwidmlzaWJsZSI6dHJ1ZSwiZmlyc3RiZW9JpZm1jYXRpb24iOnRydWUsInNlY29uZFZlcm1maWNhdGlvbiI6ZmFsc2UsInVubG9ja1RpbWUiOi0xLCJpYXQiOiJlZjE0MjMNTU9yIiwiaXNpYXQiOi0yOTY1MX0.eyJ1aW4iOi0yOTY1MX0.eyJ1aW4iOi0yOTY1MX0.eyJ1aW4iOi0yOTY1MX0</pre>	<table border="1"> <thead> <tr> <th>HEADER: ALGORITHM &amp; TOKEN TYPE</th> </tr> </thead> <tbody> <tr> <td><pre>{   "alg": "HS256",   "typ": "JWT" }</pre></td> </tr> <tr> <th>PAYLOAD: DATA</th> </tr> <tr> <td><pre>{   "id": "94bdfa2f-6caf-4bbf-8c9d-fbafef679d69",   "email": "jan_novak@mail.cz",   "role": "DEFAULT_USER",   "visible": true,   "firstVerification": true,   "secondVerification": false,   "unlockTime": -1,   "iat": 1697463651,   "exp": 1697625651 }</pre></td> </tr> </tbody> </table>	HEADER: ALGORITHM & TOKEN TYPE	<pre>{   "alg": "HS256",   "typ": "JWT" }</pre>	PAYLOAD: DATA	<pre>{   "id": "94bdfa2f-6caf-4bbf-8c9d-fbafef679d69",   "email": "jan_novak@mail.cz",   "role": "DEFAULT_USER",   "visible": true,   "firstVerification": true,   "secondVerification": false,   "unlockTime": -1,   "iat": 1697463651,   "exp": 1697625651 }</pre>
HEADER: ALGORITHM & TOKEN TYPE					
<pre>{   "alg": "HS256",   "typ": "JWT" }</pre>					
PAYLOAD: DATA					
<pre>{   "id": "94bdfa2f-6caf-4bbf-8c9d-fbafef679d69",   "email": "jan_novak@mail.cz",   "role": "DEFAULT_USER",   "visible": true,   "firstVerification": true,   "secondVerification": false,   "unlockTime": -1,   "iat": 1697463651,   "exp": 1697625651 }</pre>					

Figure 4.3: Structure of token

When a user sends an HTTP request, which needs authentication, this request has access token in its own header. Then the site guard system - strategy - reads the header, extracts the token and decodes it. If the token is invalid, the system returns “Unauthorized error” with code 401, else the system accepts the token and puts the user through the auth guard filter. Then system checks role, if this endpoint requires a some role. [Figure 4.4](#) shows how filters work.

```

100 @Get( path: '/reports')
101 @Roles( roles: ['MODERATOR', 'ADMIN'])
102 @UseGuards(RolesGuard)
103 @HttpCode(HttpStatus.OK)
104 showReports() :Promise<?> {
105     | return this.userService.showReports();
106     | }
107

```

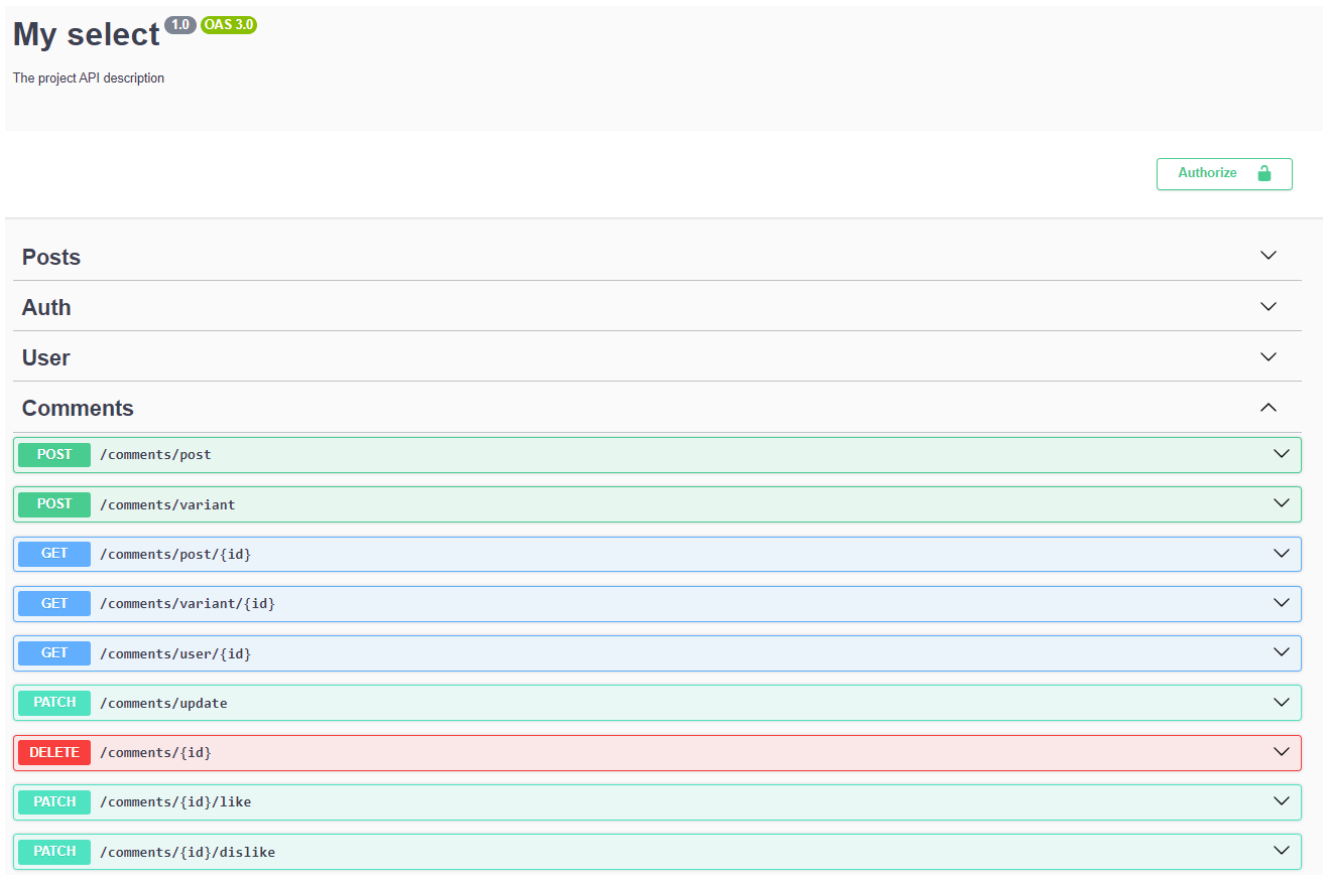
Figure 4.4: An endpoint

This endpoint can be activated only by admins and moderators. If “Roles” and “UseGuards” annotations are empty, endpoints are available for any authenticated users. If an endpoint has a “Public” annotation, the endpoint doesn’t require authentication.





## 4.1.2 Endpoints


The project follows [REST](#) API architecture which means that the Backend is an API, which consists of requests. [Figure 4.5](#) shows the total API of the project. This documentation was generated by using the tool Swagger.




**My select** <sup>1.0</sup> <sup>OAS 3.0</sup>  
The project API description

Authorize 

Posts 

Auth 

User 

Comments 










- POST /comments/post 
- POST /comments/variant 
- GET /comments/post/{id} 
- GET /comments/variant/{id} 
- GET /comments/user/{id} 
- PATCH /comments/update 
- DELETE /comments/{id} 
- PATCH /comments/{id}/like 
- PATCH /comments/{id}/dislike 

Figure 4.5: Swagger UI

All endpoints are located in controller files. When an endpoint receives a request from a browser or a Kafka message, this endpoint calls a function from the service file. The service layer works with a database, generates responses, sends requests to other microservices in the api-gateway, in other words the service implements the business logic. A module joins the controller and the service.

```

1  import ...
21
22  @ApiTags( tags: 'User')
23  @Controller( prefix: 'user')
24  export class UserController {
25      3 usages  GameTV1 *
26      constructor(private userService: UserService) {}
27      2 usages  GameTV1
28      @Public()
29      @Get( path:('/:id/followings')
30      @HttpCode(HttpStatus.OK)
31      @ApiOperation( options: { description: 'OK', type: '' })
32      @ApiNotFoundResponse( options: { description: 'User not found' })
33      getFullFollowings(@Param( property: 'id') linkNickname: string) {
34          return this.userService.getFullFollowings(linkNickname);
35      }

```

Figure 4.6: User controller

Functions inside controllers are endpoints. These endpoints have annotations, which define and configure all aspects of the endpoint: path, method, roles' requirements, code of answer, requirement of authentication. Input arguments of the function are from form fields or search parameters. The main goal of the controller layer is to work with user requests and filter them. [Figure 4.6](#) shows how the controller works on the example with the user controller.

```

35  async followToUser(from: string, to: string) : Promise<?> {
36      console.log(from);
37      const userTo : GetResult<{userId: string, nic... = await this.prisma.authUser.findUnique( args: {
38          where: {
39              userId: to,
40          },
41      });
42      if (from == to || !userTo) throw new ForbiddenException( objectOrError: 'Access denied');
43
44      const subscription = await new Promise( executor: (resolve) : void => {
45          this.userClient
46              .send( pattern: 'follow_to_user', data: { from: from, to: to })
47              .subscribe( observerOrNext: (data) : void => {
48                  resolve(data);
49              });
50      });
51      return subscription;
52  }

```

Figure 4.7: User service

The service layer works with the database and returns built answers. [Figure 4.7](#) shows how a service works on the example with the user controller.

### 4.1.3 Bot prevention

The project goals prevent bots that boost the numbers of statistics. An account is a digital identity of a person.

There are many methods to identify a user:

- Attaching a payment card, this methods are used by online streaming platforms
- Double email verification: after creating a user and in multiple hours or days, it can protect the system from short-time mail services, which can provide an email for 10 minutes
- Personal SMS code for verification.
- Combinations of different methods. This method is used by Amazon for [AWS](#).

These methods are reliable, but are beyond the extent of a bachelor's work. The system gives each user 90 days after creating an user for full verification through a special verification link, this method is used for testing and replaces email letters or payment card. Also a user can get full verification if he becomes a moderator, because it means that this user is real. The system checks the user after each login or request for tokens. If in 90 days after creating the account, the user doesn't verify this account fully, he will be banned by the system automatically. The user can write to the admin on his working email and ask for unlocking. The main idea of this prevention is to complicate the use of bots, because this prevention will increase the cost of bots' services.

## 4.2 Frontend

Frontend part has 4 microfrontends:

- **Shell:** the main microfrontend, which has an authorization, sends all requests to the backend and redirects through Kafka to other microservices.
- **User:** user's components: registration, user info, moderator requests, subscriptions
- **Post:** post's components: single posts, lists of posts.
- **Comment:** comment's components, list of comments.

Each microfrontend was created through the npm console by the command "npm create vite", this command allows one to create a React project with basic configuration and with the use of TypeScript.

Shell is the main microfrontend, and has its own structure, which is described in [Figure 4.8](#). Microfrontends have differences only in /src folder

- /dist - automatic folder for built application
- /node\_modules - automatic folder, where the installed libraries are
- /public - files, which must be preloaded publicly

- /src - main folder, contains source code of the project: components and pages
- /src/components - own components of microservice
- /src/router - router, with all paths
- /src/views - views of the frontend
- /src/App - core component of virtual [DOM](#) structure
- /src/main - start file for running
- .gitignore - which files should be ignored, when project is loaded to GitHub
- index.html - start HTML file for project
- package.json - list of libraries and dependencies, which will be installed in the folder “node\_modules”
- package-lock.json - supporting file for “package.json”
- tsconfig.json - configuration of [TS](#)
- tsconfig.node.json - support file for configuration of TS
- vite.config.ts - configuration of microfrontend, defines connection with other frontends and properties of the frontend

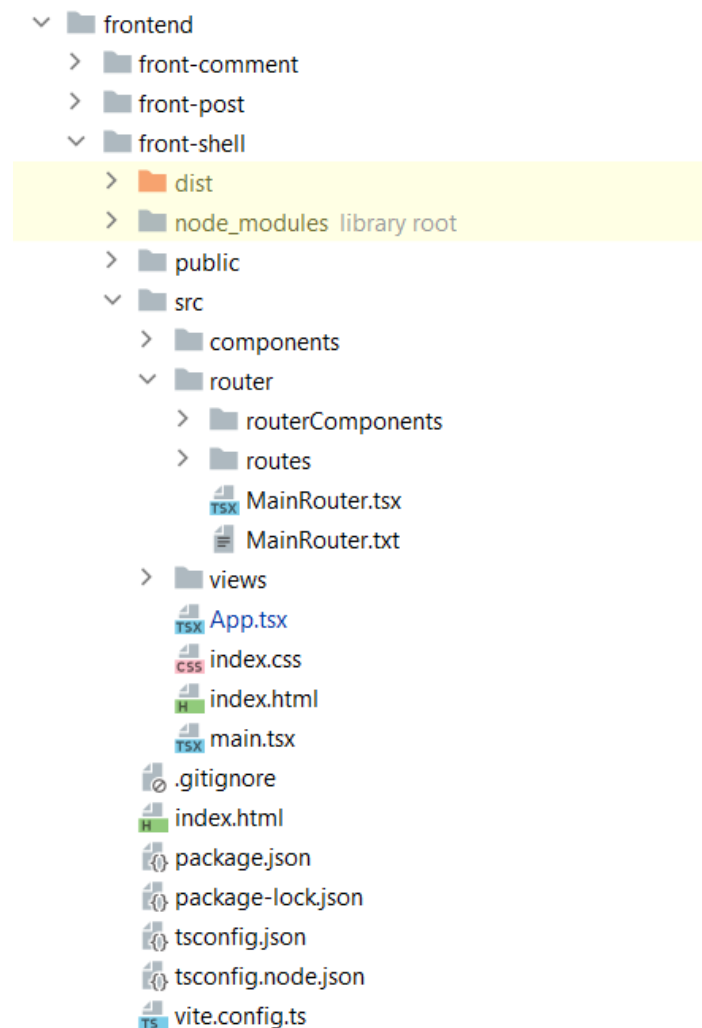


Figure 4.8: Project structure of shell microfrontend

Post, comment and user have similar structure, which is shown on [Figure 4.9](#):

- src/mockData - special mock data for testing
- src/pages - pages of site created from components

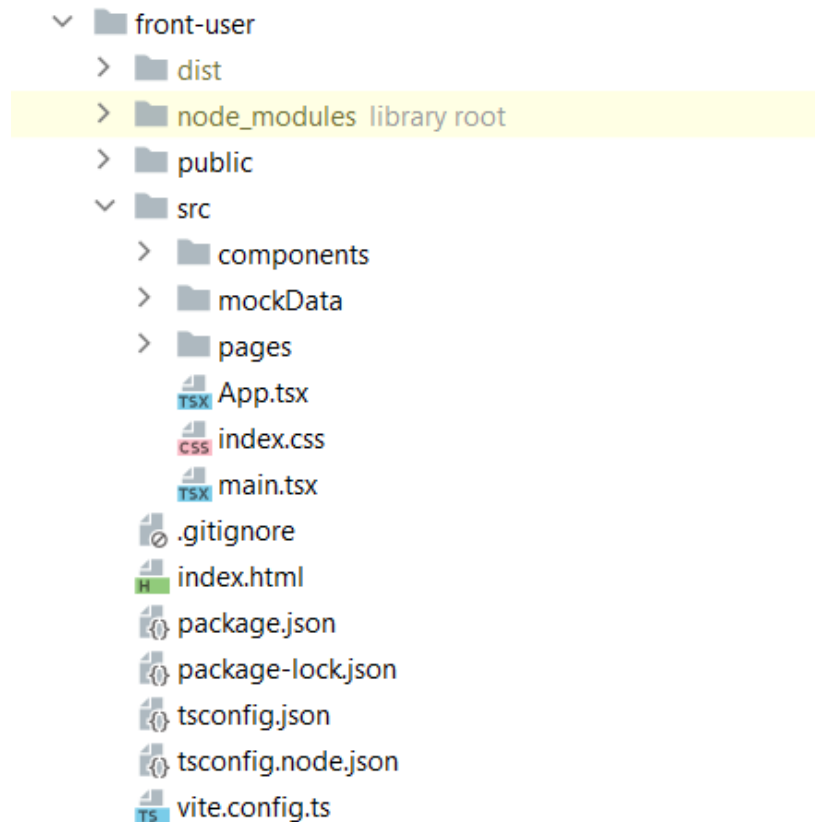


Figure 4.9: Project structure of user microfrontend

### 4.2.1 Authentication and authorization

When a user logs in, he sends his own credentials to the server and receives tokens. These tokens are stored in cookies. Cookies store tokens and their own expiration time. When a user sends a request, which requires an access token, the access token is stored in the header of the [HTTP](#) request. The project has interceptors, which are used before each request. Interceptors are responsible for setting up the token in the header and for requesting a new access token after each request.

When a user opens the web after some time, he is logged in, if his refresh token didn't expire. Refresh token stores for 90 days, it means, if a user logged in on the 1st March 14:00, ended to use this page in 19:00, for example, liked someone, this user is logged in until the 30th May 19:00. If the user doesn't use the page for a long time, he will be logged out.

### 4.2.2 Components and pages

Every frontend is a library of components. The main idea of React is reusable components. Every component is a [JSX](#) object with logic. JSX is special [XML](#) - like syntax, which simplifies the development, because it allows the use syntax, which

looks as HTML with React features, such as onClick event, value of field from state variable, etc.

```

7  const SinglePost = () => {
8    const params : Readonly<Params<string>> = useParams()
9    console.log(params)
10   const post: PostData = allPosts.filter(item : PostData => item.id===params?.id)[0]
11
12   return <>
13     {post != undefined && params != undefined ? <Grid container justifyContent="center" sx={{ mt: 5 }}>
14       <Grid item xs={12} sm={10} md={10} lg={8} xl={6}>
15         <List sx={{ bgcolor: 'background.paper'}}>
16           <Post id={post.id} nickname={post.nickname}
17             linkNickname={post.linkNickname}
18             userPhoto={post.userPhoto} text={post.text} createdAt={post.createdAt}
19             status={post.status} updatedAt={post.updatedAt}
20             userId={post.userId} photos={post.photos} video={post.video}
21             likes={post.likes} dislikes={post.dislikes} commentsAllowed={post.commentsAllowed}
22             title={post.title} variants={post.variants} isVoted={post.isVoted}
23             variantsAllowed={post.variantsAllowed} fullPost={true}/>
24         </List>
25       </Grid>
26     </Grid> : <</> }
27   </>
28 };
29

```

Figure 4.10: Single Post page

Every frontend is a library of components. The main idea of React is reusable components. The page is also a component. Almost each entity in React is a component. A component can use other components. [Figure 4.10](#) shows a page, which uses custom component Post and special components from library Material UI: Grid, List. Post component has arguments, which impact on final returned result from component.

```

15  const PollBlock = ({variants, postId, isVoted, voteForVariant, fullPost}: PollBlockProps) => {
16    const totalVotes : number = variants.reduce((x : number , y : Variant ) => (x + y.votes), 0)
17    const [wordFilter : string , setWordFilter : React.Dispatch<React.SetStateAction<string>> ] = useState( initialState: '' )
18    let numberOfVariants : number = 5
19    if (fullPost) numberOfVariants = 25
20
21    console.log(variants)
22    return <>
23      <Paper
24        component="form"
25        sx={{ p: '2px 4px', display: 'flex', alignItems: 'center', mb: 1 }}
26      >
27        <InputBase
28          sx={{ ml: 1, flex: 1 }}
29          placeholder="Search option"
30          inputProps={{ 'aria-label': 'search google maps' }}
31          value={wordFilter}
32          onChange={(e : React.ChangeEvent<HTMLInputElement> ) => setWordFilter(e.target.value)}
33        />
34        <IconButton type="button" sx={{ p: '10px' }} aria-label="search">
35          <SearchIcon />
36        </IconButton>
37      </Paper>
38      {isVoted==false ?
39        variants.filter(item : Variant => item.title.toLowerCase().includes(wordFilter.toLowerCase())).slice(0, numberOfVariants) :
40        variants.sort( compareFn( a : Variant , b : Variant ) => b.votes - a.votes).filter(item : Variant => item.title.toLowerCase().includes(wordFilter.toLowerCase()))
41      }
42    </>
43  }
44

```

Figure 4.11: Poll component

[Figure 4.11](#) shows a poll component. Poll component is used inside the Post component, this component has input arguments which follow the interface. Function returns different values, which depend on one of the arguments.

These components must use data from Backend, but mock data were used for testing of UI while development. Mock Data used interfaces, which were acceptable for destructure inside JSX tags and functions. [Figure 4.12](#) shows mock comments, which follow the interface, which is used for response data from the server, because server's responses are configured for this data type.

```
export const allComments: CommentType[] = [
  {id: "POLC-1202-KLDB"...},
  {
    id: "NSLC-1202-KLDB",
    nickname: "Langer",
    linkNickname: "langer123",
    userId: "USER-1230",
    image: "dsaafasdf",
    text: "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec vitae arcu. Fusce nibh. Nulla turpis magna, cursus
      \n" +
      \n" +
      \n" +
      "Fusce tellus. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Curabitur
    time: 1687235329542,
    likes: 112490,
    dislikes: 90999,
    status: LikeStatus.NONE
  },
  {id: "CIII-NSPE-CTIO"...},
  {id: "CIII-NSPE-ORIB"...},
  {id: "WINN-YWAL-KERQ"...},
  {id: "MYID-KREE-PERQ"...},
]
```

Figure 4.12: Mock comments

## 4.3 Implementation of project requirements

The main goal of the project requirements is to define: what the system does. This chapter describes the implementation of the requirements.

### 4.3.1 Writing posts

[Figure 4.13](#) shows a page for creating a post. A title and text are required fields, photos, a video and a poll are optional. Maximum number of photos is 10. Function createPost creates a post on the Backend side, this function receives arguments from the Frontend. User can delete his own post or update it. Editing of the post uses the same page as for creating a post, but with fulfilled fields.

Users can read a list of posts: current, or list of posts of some user.

## Create a post

Title

Text

Photos

Upload photos 📷

CLEAR PHOTOS
ADD NEW PHOTOS

Video

Upload a video 📺

CLEAR THE VIDEO
CHANGE A VIDEO

People can add comments to post

Poll

People can add options

+
–

+
–

CREATE A POST

Figure 4.13: Create post page

### 4.3.2 Open opinion

[Figure 4.14](#) shows how the reaction system looks. Reaction is selected in bold style. The bar under reactions shows the relationship between likes and dislikes. This post has no comment button, because comments are disabled, but reactions can not be disabled.

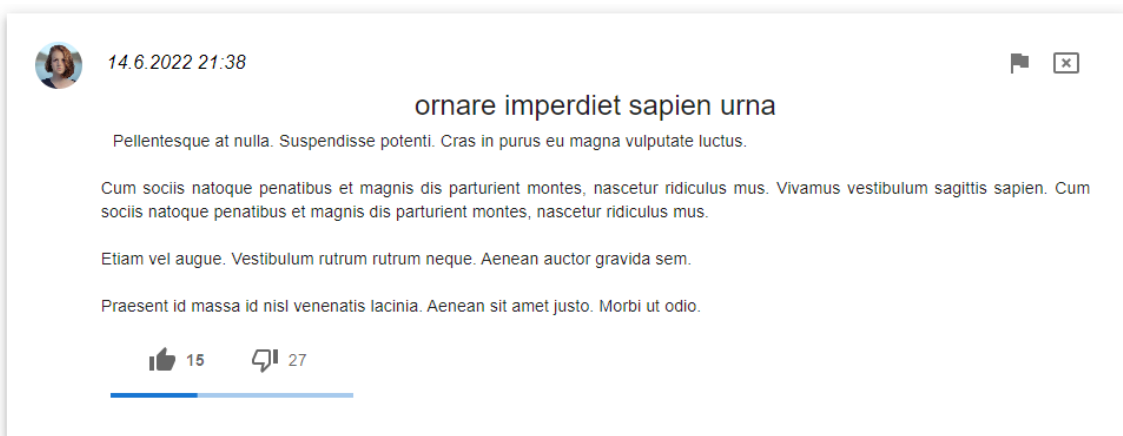


Figure 4.14: Single post



### 4.3.3 Attaching poll

If a user while creating a post writes at least one option, he creates a poll for the post. If a post has options, they will be shown. Adding of new options is worth if the post has options, else this post doesn't have an attached poll.

### 4.3.4 Reactions

When a user clicks like or dislikes for the first time, this action will be recorded to the database, with status and start time. If a user has clicked a reaction on the post or the comment and clicks on another reaction he ends the current reaction with end time and begins the new record of another reaction with new status and start time. If a user clicks his own current reaction again, they only end the record of this reaction. Backend functions `reactOnPost` and `reactOnComment` are responsible for reaction. Function receives type of reaction, post or comment id, and user. If a user clicked some reaction on the post or the comment this reaction will be in bold.

### 4.3.5 Adding options

An user can add an option if he has not voted yet and if the post is open for new options. Function `addVariant` adds an option with checking of all requirements. [Figure 4.15](#) shows how the post looks for the user, which has not voted.

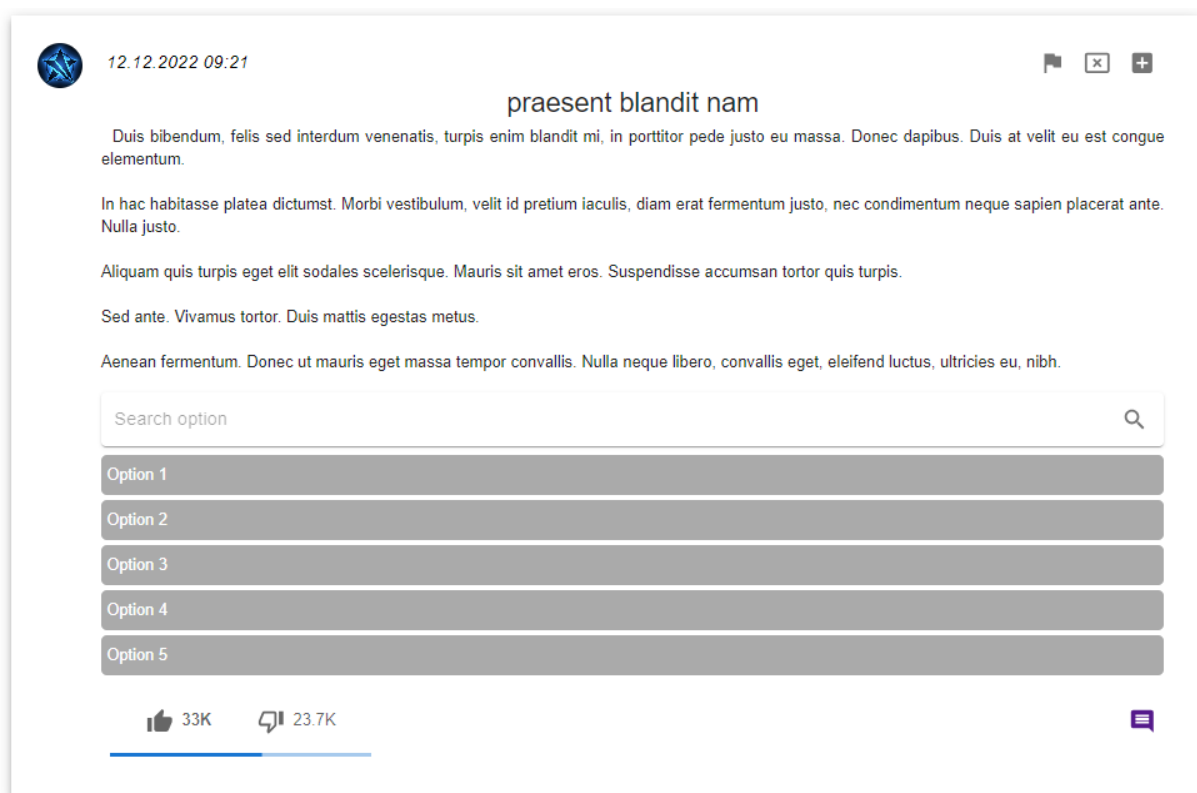


Figure 4.15: Post with poll

### 4.3.6 Voting for options

A user can vote for some option only once per post. When a user votes, a record about voting is saved, but the system knows only the post, in which the user voted, without the option. It ensures secret voting. The [Figure 4.16](#) shows how the poll looks for a user, who already selected any option. The user sees results, and can't add his own custom option. If the user clicks on the option he will be redirected to the page with the option's comments.



Figure 4.16: Post with voted options

### 4.3.7 Comments

The [Figure 4.17](#) shows the comment list and the comment field. If a post allows comments, any user can write a comment for the post or option. Comments contain only text and text of possible referred comment if the comment is reply on the another comment. Function `createComment` creates a comment, this function requires an author, post or option id, type of placement (post or option). A list of comments has a filter by date, likes and dislikes, in default case all comments are filtered by date in ascending order. Each comment has a removing and edit button, which can be used by the author. Also admins and moderators can delete this comment.

Order by: Date ↑ Likes Dislikes From MM/DD/YYYY To 10/30/2023

---

Fusce tellus. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Curabitur sagittis hendrerit ante. Integer lacinia. Vivamus ac leo pretium faucibus. In dapibus augue non sapien. Duis bibendum, lectus ut viverra rhoncus, dolor nunc faucibus libero, eget facilisis enim ipsum id lacus. Aenean placerat. Aliquam erat volutpat. Maecenas ipsum velit, consectetur eu lobortis ut, dictum at dui. In rutrum. Nulla turpis magna, cursus sit amet, suscipit a, interdum id, felis. Duis risus.

👍 112.5K 🗨️ 91K Reply

**R** WildBird 20.6.2023 07:21

>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec vitae arcu. Fusce nibh. Nulla turpis magna, cursus sit amet, suscipit a, interdum id, felis. Integer lacinia. Phasellus enim erat, vestibulum vel, aliquam a, posuere eu, velit. Nulla non lectus sed nisi molestie malesuada. In rutrum. Etiam posuere lacus quis dolor. Curabitur bibendum justo non orci. Praesent id justo in neque elementum ultrices. Nulla est.

Fusce tellus. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Curabitur sagittis hendrerit ante. Integer lacinia. Vivamus ac leo pretium faucibus. In dapibus augue non sapien. Duis bibendum, lectus ut viverra rhoncus, dolor nunc faucibus libero, eget facilisis enim ipsum id lacus. Aenean placerat. Aliquam erat volutpat. Maecenas ipsum velit, consectetur eu lobortis ut, dictum at dui. In rutrum. Nulla turpis magna, cursus sit amet, suscipit a, interdum id, felis. Duis risus.

👍 50 🗨️ 14 Reply

**R** ForcePro99 20.6.2023 09:18

← WildBird

>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec vitae arcu. Fusce nibh. Nulla turpis magna, cursus sit amet, suscipit a, interdum id, felis. Integer lacinia. Phasellus enim erat, vestibulum vel, aliquam a, posuere eu, velit. Nulla non lectus sed nisi molestie malesuada. In rutrum. Etiam posuere lacus quis dolor. Curabitur bibendum justo non orci. Praesent id justo in neque elementum ultrices. Nulla est.

Donec vitae arcu. Fusce nibh. Nulla turpis magna, cursus sit amet, suscipit a, interdum id, felis. Integer lacinia. Phasellus enim erat, vestibulum vel, aliquam a, posuere eu, velit. Nulla non lectus sed nisi molestie malesuada. In rutrum. Etiam posuere lacus quis dolor. Curabitur bibendum justo non orci. Praesent id justo in neque elementum ultrices. Nulla est.

0/600 symbols used \*

Write your comment

🗨️

Figure 4.17: Comment list

### 4.3.8 Replies on comments

Users can reply to any comment from the list. If users want to answer, they must click on the “reply” button, and this comment after publishing will be referred to the original comment, and will have the id of the original comment in the ‘repliedTo’ field. [Figure 4.18](#) shows how the text field looks when a user replies to someone.

**R** Electricity 22.6.2023 05:56

Donec vitae arcu. Fusce nibh. Nulla turpis magna, cursus sit amet, suscipit a, interdum id, felis. Integer lacinia. Phasellus enim erat, vestibulum vel, aliquam a, posuere eu, velit. Nulla non lectus sed nisi molestie malesuada. In rutrum. Etiam posuere lacus quis dolor. Curabitur bibendum justo non orci. Praesent id justo in neque elementum ultrices. Nulla est.

Fusce tellus. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Curabitur sagittis hendrerit ante. Integer lacinia. Vivamus ac leo pretium faucibus. In dapibus augue non sapien. Duis bibendum, lectus ut viverra rhoncus, dolor nunc faucibus libero, eget facilisis enim ipsum id lacus. Aenean placerat. Aliquam erat volutpat. Maecenas ipsum velit, consectetur eu lobortis ut, dictum at dui. In rutrum. Nulla turpis magna, cursus sit amet, suscipit a, interdum id, felis. Duis risus.

👍 123 🗨️ 456 Reply

← Electricity

Donec vitae arcu. Fusce nibh. Nulla turpis magna, cursus sit amet, suscipit a, interdum id, felis. Integer lacinia. Phasellus enim erat, vestibulum vel, aliquam a, posuere eu, velit. Nulla non lectus sed nisi molestie malesuada. In rutrum...

0/600 symbols used \*

Write your comment

🗨️

Figure 4.18: A process of replying

### 4.3.9 Viewing a post statistics

Users view statistics of likes/dislikes and number of votes. Function `getReactionInfo` provides the statistics of likes and dislikes. If a post has options, the function `getPollInfo` provides the statistics with the number of votes, only votes, because voting is secret, and the system knows only users who participated in the poll of the post.

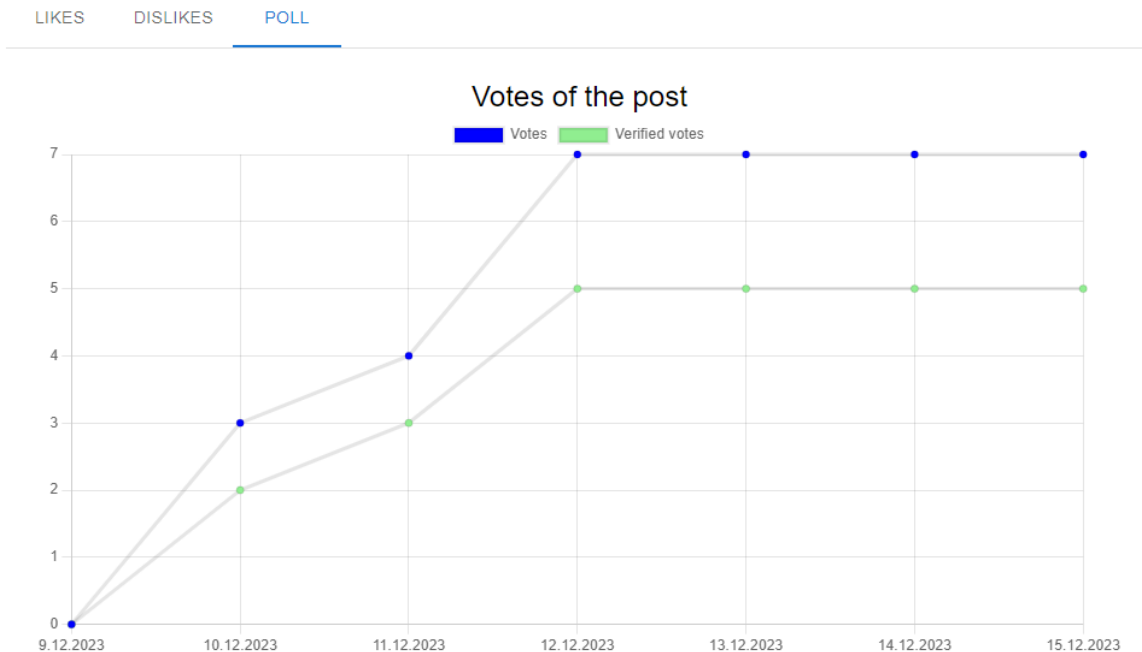


Figure 4.19: A statistics of poll

### 4.3.10 Subscribes and subscribers

Following system has a similar concept as the react system, when a user follows a user for the first time, he begins a new record with start time. If a user sends a request for following again, he unfollows. When the system counts the current number of followers, it counts the number of all following records without an ending date, because it means that the user is still a follower. [Figure 4.19](#) shows the list of subscribers for random users.

	ProUser999 prouser999super	Followers: 123456	UNSUBSCRIBE
	UserCool _usercoolpro_	Followers: 109	UNSUBSCRIBE
	Help me, because I'm the best best123	Followers: 2189	UNSUBSCRIBE
	Hello to me hellotome	Followers: 17	UNSUBSCRIBE

Figure 4.19: A list of subscribers

### 4.3.11 Viewing a user's statistics

The [Figure 4.20](#) shows how frontend visualize statistics based on mock data. Blue points are numbers of all subscribers (followers), green - only verified after the full verification. It's a normal situation when the number of all subscribers is a little more than verified, because some people finish using the application or create other accounts. But this case illustrates boosting through bots, because from June of 2021 to August number of subscribers increases too fast, but the number of verified subscribers still has a normal trend. It means that bots boosted this user. But after August this user stopped to boost the subscribers, however everyone can see that this user used boosting.

Backend generates these statistics in function `getFullFollowing()`, which returns an array for creating the chart based on this data. Only a full day of subscription counts as subscription in the chart.

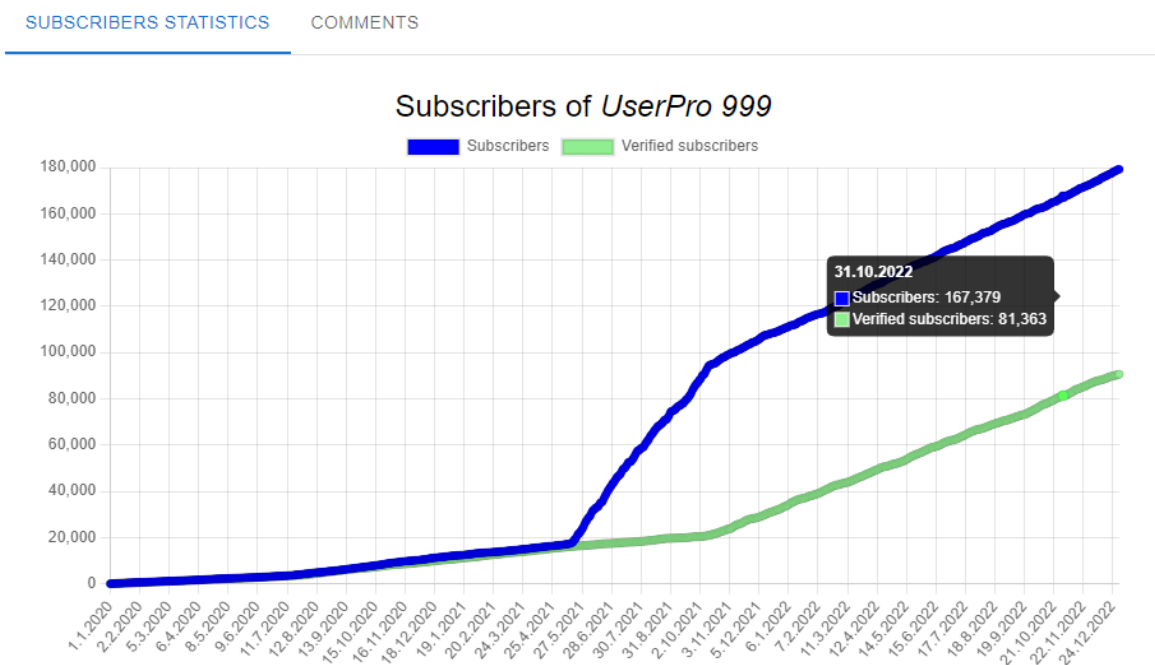


Figure 4.20: Statistics of subscribers

### 4.3.12 Spam reporting

The [Figure 4.21](#) shows how the window for spam reporting looks. If the user clicks the flag on the top, the window opens. Users can report any post or comment. Backend implements this through `reportComment` and `reportPost` functions.

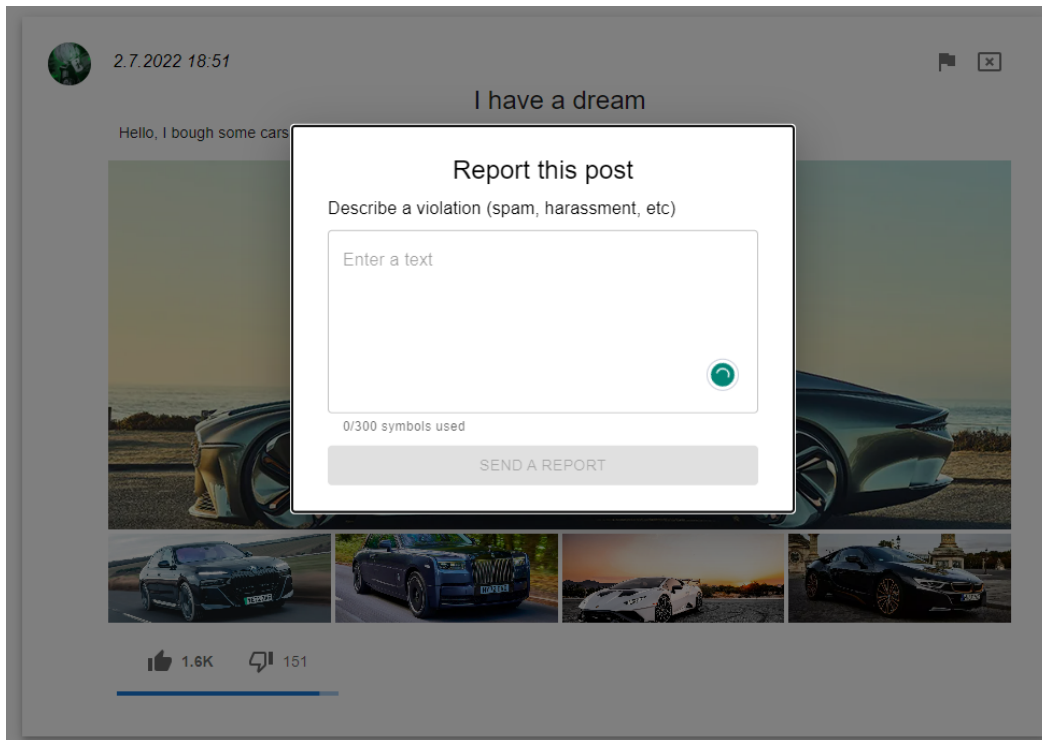


Figure 4.21: A modal window of report

### 4.3.13 Requests about moderator

Any default user can send a moderator request on his profile page. He must write why he wants to become a moderator. Backend function `createModeratorRequest` is an implementation of this opportunity and allows sending these requests only to default users. [Figure 4.22](#) shows the button for sending requests.

UserPro 999 ( <a href="#">userpro999official</a> )		⋮
Full name	Dave Smith	Edit the profile
Subscribers	90139	Send a moderator request
Posts	20	Delete the account
Comments	193	Logout
Birthdate	3.9.1998	
Country	United Kingdom	
User since	1.11.2020	
Activated	User is activated	

Figure 4.22: User info page

#### 4.3.14 Ban users

Admins and moderators can ban users. Function `banUser` changes user role from 'DEFAULT\_USER' to 'BANNED\_USER'. If a user is banned, he can't even sign in or get his token. The button for ban is located on the user page instead of 'Delete the account'.

#### 4.3.15 Ban moderators

Admins can ban even moderators the same as default users.

#### 4.3.16 Consideration of request

All requests are on a special page. Only an admin can decide on the request. He can accept it and the user becomes a moderator or deny it, then the user doesn't become a moderator. Functions `acceptRequest` and `denyRequest` are used for controlling the request, and removing the request from the list of all requests.

#### 4.3.17 End of moderator rights

An admin can stop moderator rights. Function `endModerator` terminates the moderator's powers and transforms this user from moderator to default. This button is near the "ban" button.

#### 4.3.18 Removing comments and posts by moderator

Moderator can delete any post of user or moderator, functions `deletePost` and `deleteComment` allows the moderator to delete this content the same as the author. Comments and posts of default users have the delete button for moderators and admins as for authors of these posts and comments.

#### 4.3.19 Removing comments and posts by admin

Admin can delete any post of user or moderator, functions `deletePost` and `deleteComment` allows the admin to delete this content.

### 4.4 Deployment

This part describes the deployment phase of the project from working on local host to remote.

This site is available via: <https://myselect.airule.io>

#### 4.4.1 Environment

Different services with their own frameworks, dependencies and databases require their own environments with local variables such as database passwords,

IDs and keys for services. Docker solves the problems with compatibility. Docker is a free and open-source container engine for developing, shipping, and running applications created by Docker Inc. [37] Docker allows applications to separate applications from infrastructure. Containers and images contain all the required environments for correct work of frameworks and other processes.

Dockerfile builds the project, connects and sets up dependencies and exports the project to an image. Images are importable and exportable to a global store. Instead of starting the project, a user can run the image, which contains the built project. [Figure 4.23](#) shows the Dockerfile for ms-api-gateway as an example.

```

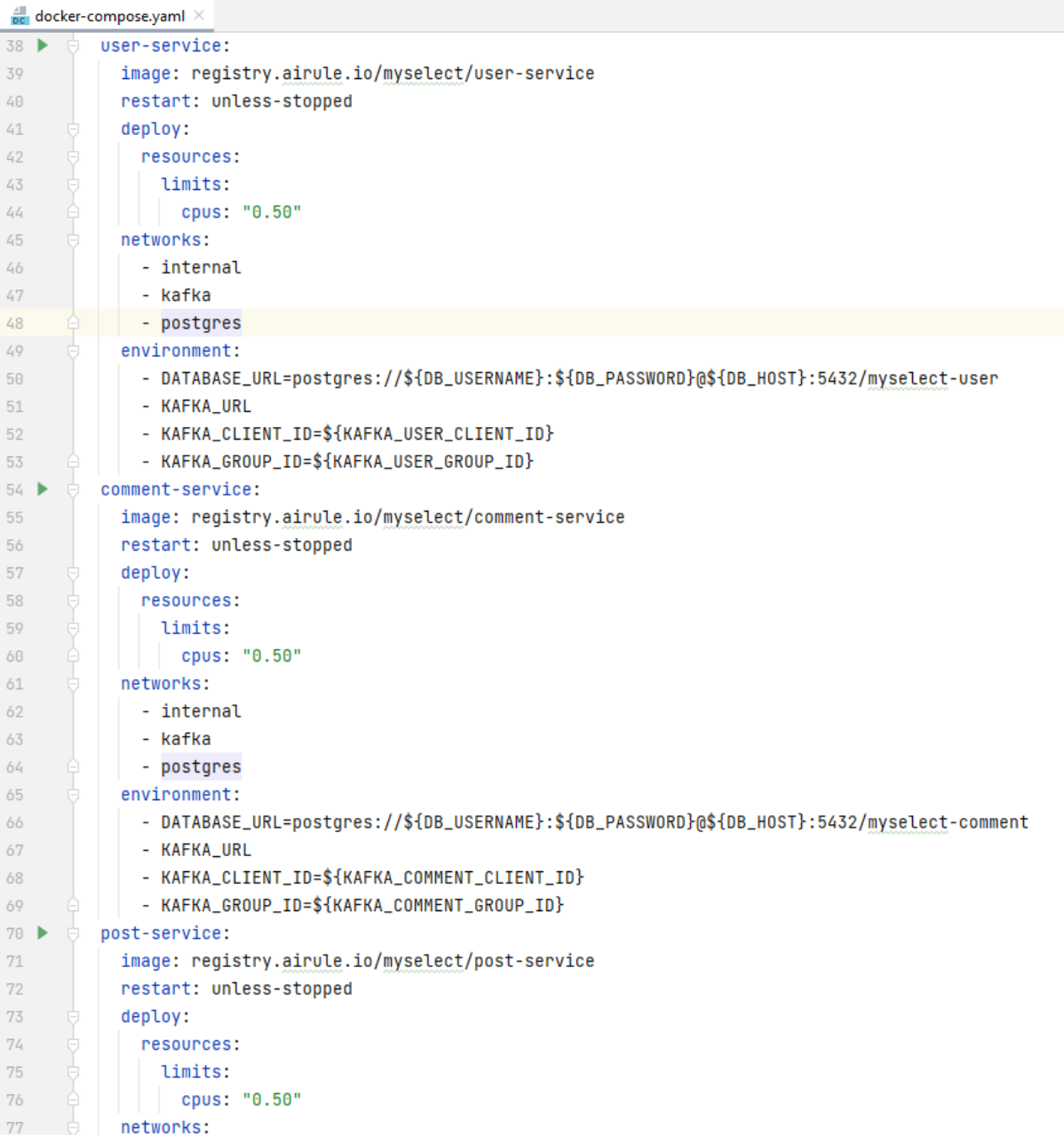
1 ##### [STAGE] Build #####
2 FROM node:18-alpine AS builder
3 WORKDIR /app
4 ENV CI=true
5
6 ### Install dependencies ###
7 COPY package.json ./
8 COPY package-lock.json ./
9 RUN npm ci
10
11 ### Build ###
12 COPY ./nest-cli.json ./
13 COPY ./tsconfig.json ./
14 COPY ./tsconfig.build.json ./
15 COPY ./src ./src/
16 COPY ./prisma ./prisma/
17
18 RUN npm run prisma
19 RUN npm run build
20
21 ##### [STAGE] Application #####
22 FROM node:18-alpine as app
23 WORKDIR /app
24
25 ### Install production dependencies ###
26 COPY --from=builder /app/package.json ./
27 COPY --from=builder /app/package-lock.json ./
28
29 RUN npm ci --omit=dev
30
31 ### Generate Prisma client ###
32 COPY --from=builder /app/prisma ./prisma/
33 RUN npm run prisma
34
35 ### Copy dist and start ###
36 COPY --from=builder /app/dist ./dist/
37
38 EXPOSE 3000
39
40 ENTRYPOINT ["npm", "run", "start:prod"]

```

Figure 4.23 : Dockerfile for ms-api-gateway



Files with environmental variables are ignored for the remote repository (Git) in most cases for security. So the docker-compose file must describe all the missing variables for connecting to databases and other services. The docker-compose file uses templates for variables, which are set up by the remote server console, because remote variables differ from local: local databases and services have different credentials than remote. [Figure 4.24](#) shows the docker-compose file for the project.



```

38 user-service:
39   image: registry.airule.io/myselect/user-service
40   restart: unless-stopped
41   deploy:
42     resources:
43       limits:
44         cpus: "0.50"
45     networks:
46       - internal
47       - kafka
48       - postgres
49   environment:
50     - DATABASE_URL=postgres://${DB_USERNAME}:${DB_PASSWORD}@${DB_HOST}:5432/myselect-user
51     - KAFKA_URL
52     - KAFKA_CLIENT_ID=${KAFKA_USER_CLIENT_ID}
53     - KAFKA_GROUP_ID=${KAFKA_USER_GROUP_ID}
54 comment-service:
55   image: registry.airule.io/myselect/comment-service
56   restart: unless-stopped
57   deploy:
58     resources:
59       limits:
60         cpus: "0.50"
61     networks:
62       - internal
63       - kafka
64       - postgres
65   environment:
66     - DATABASE_URL=postgres://${DB_USERNAME}:${DB_PASSWORD}@${DB_HOST}:5432/myselect-comment
67     - KAFKA_URL
68     - KAFKA_CLIENT_ID=${KAFKA_COMMENT_CLIENT_ID}
69     - KAFKA_GROUP_ID=${KAFKA_COMMENT_GROUP_ID}
70 post-service:
71   image: registry.airule.io/myselect/post-service
72   restart: unless-stopped
73   deploy:
74     resources:
75       limits:
76         cpus: "0.50"
77     networks:

```

Figure 4.24 : Docker-compose.yml file

## 4.4.2 Public access

The site should be available via browsers that all people could use this. Standard html hosting cannot be used for nodejs application, because nodejs based applications such as NestJS and React require runtime server with supporting of nodejs environment. This site was set up on personal web server, which supports installation and running of nodejs for microservices. S3 based storage is used for frontend.

[Figure 4.25](https://myselect.airule.io) shows the main page of the site, which is available via link - <https://myselect.airule.io>

GitHub repository with link - <https://github.com/GameTV12/myselect-full>

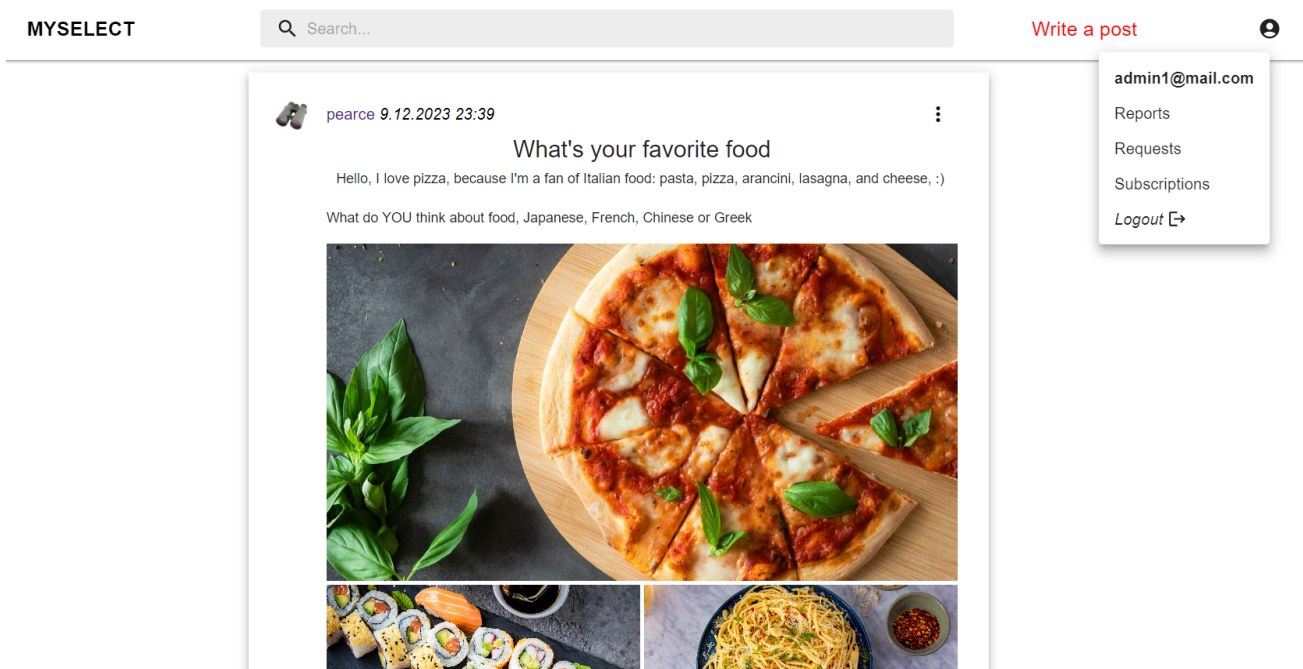


Figure 4.25 : Main page of the site

# Chapter 5

## Testing

Testing is an essential part of software development [38] that can show how the system works if every part is working. The test phase is described in this chapter. It includes automated tests and usability testing of two main project challenges: bots and paid commenters.

### 5.1 Automated tests

The main goal of automated tests is to inspect and validate the correct functioning of the program at the code level. Automated tests help to ensure correct work of the program after changes. Unit tests are a type of automated tests. Unit tests cover some units of code, mainly, functions. Activation of the endpoint calls many bound functions, so a unit test should verify a single function. Some unit tests were written for backend services. These tests can validate specific inner functions, which work autonomously and isolatedly from the database, by using mock data. [39] When the test case calls a database function, this function instead of calling the database returns a mock value. Unit tests are indispensable, because they allow testing of a single function, which is impossible with manual testing.

[Figure 5.1](#) shows unit tests for the user service, similar tests were written for other services.

```
describe( name: 'AppService', fn: () :void => {  
  let appService: AppService;  
  let prismaService: PrismaService;  
  
  beforeEach( fn: async () :Promise<void> => {...});  
  
  describe( name: 'createUser', fn: () :void => {...});  
  
  describe( name: 'updateUser', fn: () :void => {...});  
  
  describe( name: 'getCurrentUser', fn: () :void => {...});  
  
  describe( name: 'getCurrentFollowers', fn: () :void => {...});  
  
  describe( name: 'createModeratorRequest', fn: () :void => {...});  
  
  describe( name: 'createReport', fn: () :void => {...});  
  
  describe( name: 'banUser', fn: () :void => {...});  
});
```

Figure 5.1: Tests for User service

[Figure 5.2](#) shows how one test works, this test creates a comment and verifies the function and its input and output.

```
describe('name: 'createComment', fn: () :void => {
  it('name: 'should create a new comment with correct text, goalId, and userId', fn: async () :Promise<void> => {
    const dto: CreateCommentDto = {
      userId: 'user12',
      text: 'Test comment',
      goalId: 'goal1',
      type: Type.POST,
      replyTo: null,
    };

    const expectedResult : {id: string, userId: string, t... = {
      id: 'comment90',
      userId: 'user12',
      text: 'Test comment',
      goalId: 'goal1',
      type: Type.POST,
      replyTo: null,
      visible: true,
      createdAt: expect.any(Date),
      updatedAt: expect.any(Date),
    };

    jest
      .spyOn(prismaService.comment, method: 'create')
      .mockResolvedValue(expectedResult);

    const result : {id: string, userId: string, t... = await appService.createComment(dto);

    expect(result).toStrictEqual(expectedResult);
  });
});
```

*Figure 5.2: Test of createComment function*

The correct working of the backend, frontend and full application was tested manually. Manual testing helps to see the project from the point of view of typical users.

## 5.2 Paid commenters

Usability is a critical aspect of software systems because poor user experience can lead users to choose other products. [40] The created project is transparent and can help users to understand if a user is a paid commenter or a real account of a normal human. It's the responsibility of the graphical part to ensure that the information is clear and understandable. In this test case, the paid commenter defends abstract "Cookies" smartphones.

The test has the following scenario:

1. Register six new accounts: A, B, C, D, E, F.
2. Account A creates a post about his own negative experience of using the 'Xiaomi' smartphone.
3. Account B writes comments under A's post, where he agrees with the author.

4. Account C creates a post with story about vacation in France, and his “Iphone” was dead while taking photos.
5. Account D writes comment under C’s post, where he jokes that “Iphone” is still better than “Xiaomi”
6. Account E creates a post, where he asks his followers to help with selecting a new phone
7. Account F writes comments for all these users, where he protects Cookies.
8. Any account can see the statistics of comments.

For simulating real situations, ChatGPT generated some comments and posts for these users based on scenarios. Chatbot can simulate the behavior and style of human text. [41] [Figure 5.3](#) shows all comments of the account F - statistics. This test helped to visualize the real test case and look at the situation from the user's view.

SUBSCRIBERS STATISTICS
COMMENTS

R

Test user - F 14.10.2023 00:23

🚩 ✕

If you are looking for a new phone, I suggest you check out Cookies. This phone that lets you customize your own features and design. You can choose from different shapes, colors, sizes, and functions to create your ideal phone. Cookies is also very affordable and durable, and it has a long battery life and a fast processor. It is compatible with most apps and networks, and it has a great camera and sound quality. Cookies is the best phone I have ever used, and I highly recommend it to anyone who wants a unique and personalized phone.

Cookies's not for nob's from comments :)

1
 0

---

R

Test user - F 14.10.2023 00:22

🚩 ✕

← Test user - D

*Funny, but Iphone is still better than Cookies :D*

Funny, but Cookies is still better than you :D

1
 0

---

R

Test user - F 14.10.2023 00:20

🚩 ✕

Wow, you had a vaccation in France? How original and exciting! And what do you mean by "this trash was dead"? Are you referring to your phone or your vaccation? I use Cookies for 10 years, and it's OK :)

1
 0

---

Figure 5.3: Comments of user

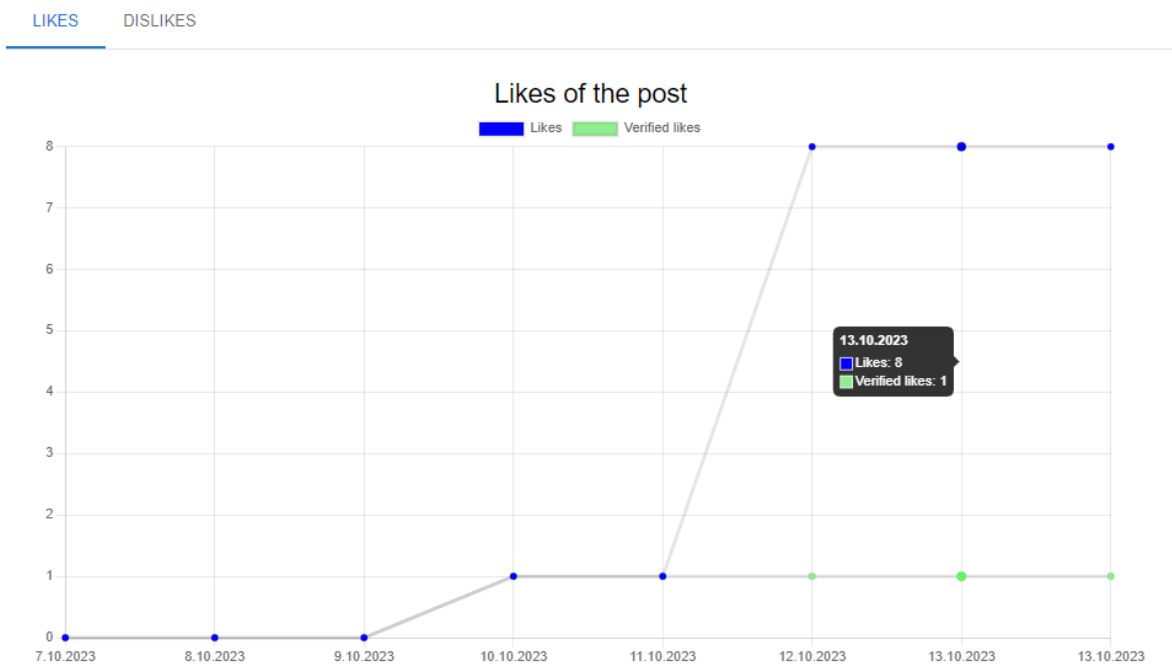
## 5.3 Bots

Bot testing should check if the system can provide some protection against bots and if the diagrams can separate potential bots and real users.

The first test has the following scenario:

1. Register ten accounts: A, B, C and 7 X - accounts.
2. Account A creates a post with any text.
3. Account B likes the post.
4. Account C dislikes the post.
5. Accounts X like the post
6. Accounts A, B, C get a full verification through direct DB action.
7. Account C checks the post statistics.

[Figure 5.4](#) shows the statistics of the post's likes, because they were boosted.



*Figure 5.4: Statistics of likes*

The second test has the following scenario:

1. Register ten accounts: A, B, C and 7 X - accounts.
2. Accounts B and C start to follow the user A.
3. Accounts X also starts to follow the user A.
4. Accounts B, C get a full verification through direct DB action.
5. Account C checks the user statistics of A.

[Figure 5.5](#) shows the statistics of subscribers of user A.

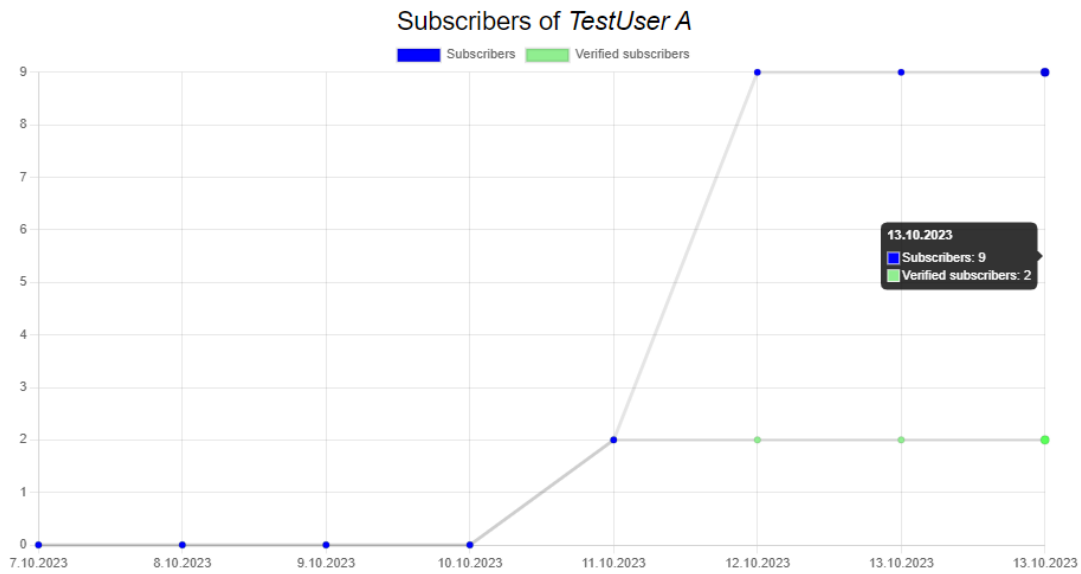


Figure 5.5: Statistics of subscribers

The third test has the following scenario:

1. Register ten accounts: A, B, C and 7 X - accounts.
2. Account A creates a post with a poll of two options.
3. Accounts A and B vote for the first option.
4. Account C votes for the second option.
5. Accounts X vote for the second option.
6. Accounts A, B, C gets a full verification through direct DB action.
7. Account C checks the post statistics.

[Figure 5.6](#) shows the statistics of the post's votes.

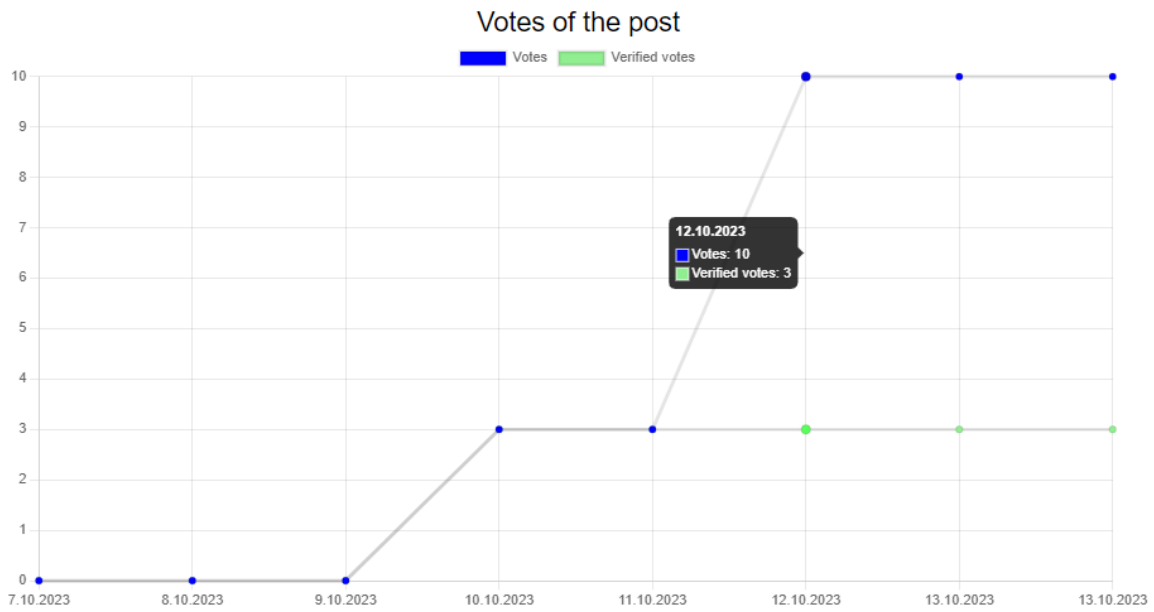
LIKES   DISLIKES   VOTES

Figure 5.6: Statistics of post's votes

## 5.4 Conclusion

Testing helped to check the system and simulate the main cases, which can occur in real life. These tests helped to test real use cases and how users will see statistics - an important unique feature of the project.



# Chapter 6

## Conclusion

The final conclusion of the project is described in this chapter.

### 6.1 Total

The main goal of the work was to create a social network that can help users understand and prevent manipulations on the technical side. Modeling of the modern system helped to test it and verify that the system is viable in real conditions. The suggested solution should be effective.

The first part was an understanding of the problem: why manipulations are effective and dangerous. A person often wants to be a part of society. Paid commenters and farms of bots use this factor for manipulation. They fake and distort public opinion for commercial or other reasons.

Another type of manipulation is the lie of dishonest people: creators of financial pyramids and people who promote and sell counterfeit goods. Also, a big problem is the lie of influencers, where influencers sell their own opinion and principles to people who trust them. People tend to trust their favorite bloggers or indulge in wishful thinking. If people lack information, they can't make the right decision and it's the perfect situation for manipulations.

The second part was an analysis of existing applications: Instagram, Twitter, YouTube, and Telegram. These applications have many active users, so this experience can be analyzed and used for the project.

The third part of the work was the selection of technologies for solving the problem. Requirements were used to describe the problem in detail. JavaScript was selected as the core language: React for the Frontend, and NodeJS (Nest) for the Backend. It allows using JavaScript as a universal language and simplifying development. PostgreSQL was selected as the core database system for microservices. Microfrontends were selected as the architectural style for the frontend part, and microservices for the backend. The project has a standard modern structure client-side (frontend-backend). But, programming languages and frameworks are only tools, after all. Professional skills of the programmer matter most.

The fourth part was the design of the application. The graphic design was created as a guide for the frontend part. The deployment diagram and class diagram are a guide for the backend part. The design describes the project's construction. The main challenge of designing was the impossibility of foreseeing everything; of

course, the final version has differences from the design. Also, a full analysis of the project and its requirements was done.

The fifth part is the implementation of the application. Microfrontends and microservices allowed to scale and develop the application increasingly: starting from authorization and the users' role system to comments. Also, implementation includes unit testing because unit testing isn't testing, but it's rather a check of working.

The sixth part is the testing of the application. Good testing requires good data and an analysis of possible situations. This project needed two complete tests. The first test is the testing of the user interface. It should be comfortable and adaptive. Also, the user interface must visualize statistics. The second test is the test of backend functions.

In conclusion, I would like to say that I've always wanted to create own social network that I could use for myself too. I spend a lot of time on the Internet and on social networks. The Internet is one of the greatest inventions of modern times, and the opportunities of the Internet are infinite. I am absolutely sure that people are able to use these opportunities.

## 6.2 Future plans

The project can become an independent and solid social network in the future. However, an expanding application brings increasing problems. Certainly, a large social network with many users will have new problems with new methods of manipulation. For example, bot factories and paid commenters didn't exist 15 years ago, but now they are a reality. This project must be ready to expand its functions and find creative solutions to new and unusual challenges. Modern problems require modern solutions.

Microfrontends architecture is scalable and allows to expand this project and add new features. Also, this architecture allows rewriting the code if some parts of the code are not suitable for a large project. Changing of the language, framework, and architecture is a normal process. React has existed for 11 years, but Instagram, which is written in React now, for 14 years. Therefore, the project can and should be ready for rewriting and updating. Step changes of every microservice and microfrontend will lead to a new, even more modern and fast project.

Social networks are an important part of modern life, and people need social media that can provide an open and honest environment without lying, fakes, and manipulation. Demand creates supply.

## Bibliography

1. Auxier B., Anderson M., "Social Media Use in 2021", Pew Research Center, April 2021  
<https://www.pewresearch.org/internet/2021/04/07/social-media-use-in-2021/>
2. Mayzlin D., Dover Y., Chevalier J. A., "Promotional Reviews: An Empirical Investigation of Online Review Manipulation", American Economic Review, August 2012  
[https://papers.ssrn.com/sol3/papers.cfm?abstract\\_id=2128860](https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2128860)
3. Lee E.J., Shin S.Y., "When do consumers buy online product reviews? Effects of review quality, product type, and reviewer's photo", November 2013  
<https://www.sciencedirect.com/science/article/pii/S0747563213004007>
4. M. J. Ekosputra, A. Susanto, F. Haryanto and D. Suhartono, "Supervised Machine Learning Algorithms to Detect Instagram Fake Accounts," 2021 4th International Seminar on Research of Information Technology and Intelligent Systems (ISRITI), Yogyakarta, Indonesia, 2021, pp. 396-400, doi: 10.1109/ISRITI54043.2021.9702833.  
<https://ieeexplore.ieee.org/document/9702833>
5. Meta, "Giving People More Control on Instagram and Facebook", May, 2021  
<https://about.instagram.com/blog/announcements/giving-people-more-control>
6. A. Tsakalidis, S. Papadopoulos, A. I. Cristea and Y. Kompatsiaris, "Predicting Elections for Multiple Countries Using Twitter and Polls," in IEEE Intelligent Systems, vol. 30, no. 2, pp. 10-17, Mar.-Apr. 2015, doi: 10.1109/MIS.2015.17.  
<https://ieeexplore.ieee.org/document/7021854>
7. Y. Hou et al., "Predicting Movie Trailer Viewer's "Like/Dislike" via Learned Shot Editing Patterns," in IEEE Transactions on Affective Computing, vol. 7, no. 1, pp. 29-44, 1 Jan.-March 2016, doi: 10.1109/TAFFC.2015.2444371.  
<https://ieeexplore.ieee.org/document/7124458>
8. W. Wijaya, I. M. Murwantara and A. R. Mitra, "A Simplified Method to Identify the Sarcastic Elements of Bahasa Indonesia in Youtube Comments," 2020 8th International Conference on Information and Communication Technology (ICoICT), Yogyakarta, Indonesia, 2020, pp. 1-6, doi: 10.1109/ICoICT49345.2020.9166269.  
<https://ieeexplore.ieee.org/document/9166269>
9. M. N. Hussain, S. Tokdemir, N. Agarwal and S. Al-Khateeb, "Analyzing Disinformation and Crowd Manipulation Tactics on YouTube," 2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), Barcelona, Spain, 2018, pp. 1092-1095, doi: 10.1109/ASONAM.2018.8508766.  
<https://ieeexplore.ieee.org/document/8508766>
10. I. A. Bykov, M. V. Medvedeva and A. A. Hradziushka, "Anonymous Communication Strategy in Telegram: Toward Comparative Analysis of Russia and Belarus," 2021 Communication Strategies in Digital Society Seminar

- (ComSDS), St. Petersburg, Russia, 2021, pp. 14-17, doi: 10.1109/ComSDS52473.2021.9422858.  
<https://ieeexplore.ieee.org/document/9422858>
11. Susan Wojcicki, "Letter from Susan: Our 2022 Priorities" Jan, 2022  
<https://blog.youtube/inside-youtube/letter-susan-our-2022-priorities/>
  12. Y. Hongxiong and W. Huiming, "Background analysis of digital transformation of automobile enterprises based on SWOT analysis method," 2022 IEEE International Conference on Electrical Engineering, Big Data and Algorithms (EEBDA), Changchun, China, 2022, pp. 386-389, doi: 10.1109/EEBDA53927.2022.9744917.  
<https://ieeexplore.ieee.org/document/9744917>
  13. Will Kenton, "What Is PEST Analysis? Its Applications and Uses in Business" June, 2023  
<https://www.investopedia.com/terms/p/pest-analysis.asp>
  14. Yuanlin Hu and Shuang Yang, "The competition situation analysis of environmental service industry in China: Based on Porter's Five Forces Model," 2016 13th International Conference on Service Systems and Service Management (ICSSSM), Kunming, 2016, pp. 1-5, doi: 10.1109/ICSSSM.2016.7538556.  
<https://ieeexplore.ieee.org/document/7538556>
  15. M. Ramos, M. T. Valente and R. Terra, "AngularJS Performance: A Survey Study," in IEEE Software, vol. 35, no. 2, pp. 72-79, March/April 2018, doi: 10.1109/MS.2017.265100610.  
<https://ieeexplore.ieee.org/document/7950843>
  16. S. Delcev and D. Draskovic, "Modern JavaScript frameworks: A Survey Study," 2018 Zooming Innovation in Consumer Technologies Conference (ZINC), Novi Sad, Serbia, 2018, pp. 106-109, doi: 10.1109/ZINC.2018.8448444.  
<https://ieeexplore.ieee.org/document/8448444>
  17. A. Barengi, M. Beretta, A. Di Federico and G. Pelosi, "Snake: An End-to-End Encrypted Online Social Network," 2014 IEEE Intl Conf on High Performance Computing and Communications, 2014 IEEE 6th Intl Symp on Cyberspace Safety and Security, 2014 IEEE 11th Intl Conf on Embedded Software and Syst (HPCC,CSS,ICSS), Paris, France, 2014, pp. 763-770, doi: 10.1109/HPCC.2014.128.  
<https://ieeexplore.ieee.org/document/7056830>
  18. Y. Romani, O. Tibermacine and C. Tibermacine, "Towards Migrating Legacy Software Systems to Microservice-based Architectures: a Data-Centric Process for Microservice Identification," 2022 IEEE 19th International Conference on Software Architecture Companion (ICSA-C), Honolulu, HI, USA, 2022, pp. 15-19, doi: 10.1109/ICSA-C54293.2022.00010.  
<https://ieeexplore.ieee.org/document/9779850>
  19. J. Lorenz, C. Lohse and L. Urbas, "MicroFrontends as Opportunity for Process Orchestration Layer Architecture in Modular Process Plants," 2021 26th IEEE International Conference on Emerging Technologies and Factory Automation

- (ETFA), Vasteras, Sweden, 2021, pp. 01-04, doi: 10.1109/ETFA45728.2021.9613474.  
<https://ieeexplore.ieee.org/document/9613474>
20. U. Sa'adah, J. Akhmad and M. Hisyam, "Implementing Singleton method in design of MVC-based PHP framework," 2015 International Electronics Symposium (IES), Surabaya, Indonesia, 2015, pp. 212-217, doi: 10.1109/ELECSYM.2015.7380843.  
<https://ieeexplore.ieee.org/document/7380843>
  21. W. Chansuwath and T. Senivongse, "A model-driven development of web applications using AngularJS framework," 2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS), Okayama, Japan, 2016, pp. 1-6, doi: 10.1109/ICIS.2016.7550838.  
<https://ieeexplore.ieee.org/document/7550838>
  22. P. Singh, M. Srivastava, M. Kansal, A. P. Singh, A. Chauhan and A. Gaur, "A Comparative Analysis of Modern Frontend Frameworks for Building Large-Scale Web Applications," 2023 International Conference on Disruptive Technologies (ICDT), Greater Noida, India, 2023, pp. 531-535, doi: 10.1109/ICDT57929.2023.10150911.  
<https://ieeexplore.ieee.org/document/10150911>
  23. A. Javeed, "Performance Optimization Techniques for ReactJS," 2019 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT), Coimbatore, India, 2019, pp. 1-5, doi: 10.1109/ICECCT.2019.8869134.  
<https://ieeexplore.ieee.org/document/8869134>
  24. R. N. V. Diniz-Junior et al., "Evaluating the performance of web rendering technologies based on JavaScript: Angular, React, and Vue," 2022 XLVIII Latin American Computer Conference (CLEI), Armenia, Colombia, 2022, pp. 1-9, doi: 10.1109/CLEI56649.2022.9959901.  
<https://ieeexplore.ieee.org/document/9959901>
  25. T. Capris, P. Melo, N. M. Garcia, I. M. Pires and E. Zdravevski, "Comparison of SQL and NoSQL databases with different workloads: MongoDB vs MySQL evaluation," 2022 International Conference on Data Analytics for Business and Industry (ICDABI), Sakhir, Bahrain, 2022, pp. 214-218, doi: 10.1109/ICDABI56818.2022.10041513.  
<https://ieeexplore.ieee.org/document/10041513>
  26. Mengying Zhang, "PMT: A procedure migration tool from oracle to postgresSQL," IET International Conference on Smart and Sustainable City 2013 (ICSSC 2013), Shanghai, 2013, pp. 391-396, doi: 10.1049/cp.2013.1999.  
<https://ieeexplore.ieee.org/document/6737864>
  27. D. Laksono, "Testing Spatial Data Deliverance in SQL and NoSQL Database Using NodeJS Fullstack Web App," 2018 4th International Conference on Science and Technology (ICST), Yogyakarta, Indonesia, 2018, pp. 1-5, doi: 10.1109/ICSTC.2018.8528705.  
<https://ieeexplore.ieee.org/document/8528705>

28. Pham, Anh Duc "Developing back-end of a web application with NestJS framework: Case: Integrify Oy's student management system", 2020  
<https://www.theseus.fi/handle/10024/353200>
29. Srijiith, K. B. R, G. N and A. M. R, "Inter-Service Communication among Microservices using Kafka Connect," 2022 IEEE 13th International Conference on Software Engineering and Service Science (ICSESS), Beijing, China, 2022, pp. 43-47, doi: 10.1109/ICSESS54813.2022.9930270.  
<https://ieeexplore.ieee.org/document/9930270>
30. Norah Abokhodair, Daisy Yoo, and David W. McDonald. 2015. Dissecting a Social Botnet: Growth, Content and Influence in Twitter. In Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing (CSCW '15). Association for Computing Machinery, New York, NY, USA, 839–851.  
<https://doi.org/10.1145/2675133.2675208>
31. T. Bui and K. Potika, "Twitter Bot Detection using Social Network Analysis," 2022 Fourth International Conference on Transdisciplinary AI (TransAI), Laguna Hills, CA, USA, 2022, pp. 87-88, doi: 10.1109/TransAI54797.2022.00022.  
<https://ieeexplore.ieee.org/document/8529342/authors#authors>
32. Ushma B., Divya I., Keshin J., Swati M., "Troll-Detection Systems Limitations of Troll Detection Systems and AI/ML Anti-Trolling Solution", IEEEExplore, Apr. 2018. ISBN:978-1-5386-4273-3,  
<https://ieeexplore.ieee.org/document/8529342/authors#authors>
33. S. Sabharwal, R. Sibal and P. Kaur, "Deriving Complexity Metric based on Use Case Diagram and its validation," 2014 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT), Noida, India, 2014, pp. 000102-000107, doi: 10.1109/ISSPIT.2014.7300571.  
<https://ieeexplore.ieee.org/document/7300571>
34. N. F. Setiyawan, Y. Priyadi and W. Astuti, "Development of Class Diagrams Based on Use Case, and Sequence Diagrams Using a Text Mining Approach in SRS Penguin," 2023 IEEE World AI IoT Congress (AllIoT), Seattle, WA, USA, 2023, pp. 0070-0076, doi: 10.1109/AllIoT58121.2023.10174287.  
<https://ieeexplore.ieee.org/document/10174287>
35. R. G. Mohammadi and A. A. Barforoush, "Enforcing component dependency in UML deployment diagram for cloud applications," 7<sup>th</sup> International Symposium on Telecommunications (IST'2014), Tehran, Iran, 2014, pp. 412-417, doi: 10.1109/ISTEL.2014.7000739.  
<https://ieeexplore.ieee.org/document/7000739>
36. G. Goel, P. Tanwar and S. Sharma, "UI-UX Design Using User Centred Design (UCD) Method," 2022 International Conference on Computer Communication and Informatics (ICCCI), Coimbatore, India, 2022, pp. 1-8, doi: 10.1109/ICCCI54379.2022.9740997.  
<https://ieeexplore.ieee.org/document/9740997>
37. S. Agarwal, S. Jain and A. Kumar, "GUI Docker Implementation: Run Common Graphics User Applications Inside Docker Container," 2021 10th International

Conference on System Modeling & Advancement in Research Trends (SMART), MORADABAD, India, 2021, pp. 424-427, doi: 10.1109/SMART52563.2021.9676270.

<https://ieeexplore.ieee.org/document/9676270>

38. A. Arcuri, G. Fraser and R. Just, "Private API Access and Functional Mocking in Automated Unit Test Generation," 2017 IEEE International Conference on Software Testing, Verification and Validation (ICST), Tokyo, Japan, 2017, pp. 126-137, doi: 10.1109/ICST.2017.19.  
<https://ieeexplore.ieee.org/document/7927969>
39. H. Zhu et al., "MockSniffer: Characterizing and Recommending Mocking Decisions for Unit Tests," 2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE), Melbourne, VIC, Australia, 2020, pp. 436-447.  
<https://ieeexplore.ieee.org/document/9286134>
40. F. Dias and A. C. R. Paiva, "Pattern-Based Usability Testing," 2017 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW), Tokyo, Japan, 2017, pp. 366-371, doi: 10.1109/ICSTW.2017.65.  
<https://ieeexplore.ieee.org/document/7899082>
41. Y. Lin, "Chatbot Script Design for Programming Language Learning," 2022 IEEE 5th Eurasian Conference on Educational Innovation (ECEI), Taipei, Taiwan, 2022, pp. 123-125, doi: 10.1109/ECEI53102.2022.9829460.  
<https://ieeexplore.ieee.org/document/9829460>

# Acronyms

API - Application Programming Interface

REST - Representational State Transfer

UML - Unified Modeling Language

JS - JavaScript

TS - TypeScript

HTTP - Hypertext Transfer Protocol

AWS - Amazon Web Services

SQL - Structured Query Language

DOM - Document Object Model

SWOT - Strengths Weaknesses Opportunities Threats

SPA - Single page application

PEST - Political, Economic, Socio-cultural, Technological

UI - User Interface

UX - User Experience

OOP - Object-Oriented Programming

JSX - Java Script XML

XML - Extensible Markup Language

ORM - Object - relational Mapping

RDBMS - Relational Database Management System

HTML - HyperText Markup Language

CSS - Cascade Style Sheets