

**Opa**

Documentation

Computer Architecture project

**Created by:**

Edvardas Ražanskas,

Justinas Rimavičius,

Alanas Pauša,

Laurita Radiševska,

Benas Sakalauskas,

Matas Lekavičius,

Ignas Vigelis

Vilnius University

2022

ANNOTATION

This document contains the documentation of the “Opa” system. Work intended to describe the information system “Opa”, its development process, in detail analysis, objectives, principles of operation, difficulties encountered, and tasks performed.

<https://project-opa.netlify.app/>

Table of Contents

[1. SCOPE OF THE PROJECT 4](#_Toc121993021)

[1.1. Introduction to the project 4](#_Toc121993022)

[1.2. Aims 4](#_Toc121993023)

[1.3. Team structure 4](#_Toc121993024)

[1.4. Development tools 4](#_Toc121993025)

[1.4.1. Systems and platform used 5](#_Toc121993026)

[1.4.2. Technologies used 5](#_Toc121993027)

[1.5. Functional requirements 6](#_Toc121993028)

[1.6. Non-Functional requirements 7](#_Toc121993049)

[2. WORKFLOW 8](#_Toc121993062)

[2.1. Prototype 8](#_Toc121993063)

[2.2. Back-end 8](#_Toc121993064)

[2.2.1. “Twitter” Scraping 8](#_Toc121993065)

[2.2.2. „Reddit“ Scraping 8](#_Toc121993066)

[2.3. Front-end 9](#_Toc121993067)

[2.4. Database 9](#_Toc121993068)

[3. Functionality walkthrough 10](#_Toc121993069)

[3.1. A home page 10](#_Toc121993070)

[3.2. Log in/Sign up pages 11](#_Toc121993071)

[3.3. Platform selection 13](#_Toc121993072)

[3.4. Keyword input without registration 13](#_Toc121993073)

[3.5. Keyword input for registered user 14](#_Toc121993074)

[3.6. Word cloud 15](#_Toc121993075)

[3.7. Profile view (was expected to do) 15](#_Toc121993076)

[4. QUALITY ASSURANCE AND TESTING 16](#_Toc121993077)

[5. GRAPHS AND VISUALIZATION 17](#_Toc121993079)

[Gantt diagram 17](#_Toc121993080)

[User case diagram 17](#_Toc121993081)

[Activity diagram 18](#_Toc121993082)

[Navigation model 19](#_Toc121993083)

[Timeline 19](#_Toc121993084)

# SCOPE OF THE PROJECT

## Introduction to the project

Initially, our team was planning on making an online casino, however, this would create an unevenly distributed workload in our team, so we decided to ditch this idea and reorientate our project to a web scraping site. The main idea behind it is to create a system, which would show the user the popularity of requested keywords from “Twitter” and “Reddit” platforms, showing its count for unregistered users and visually presenting data with graphs for registered ones.

## Aims

* To create a website that scrapes the most popular posts on “Twitter” or “Reddit” and displays the frequency of the keyword asked by the user.
* The main goal is for the user to see what words are used the most and make the posts popular in a specific genre/subject (by keywords).

## Team structure

* **Edvardas Ražanskas** – Project lead, back-end
* **Justinas Rimavičius** – Front-end, Back-end
* **Alanas Pauša** – Back-end responsible for „Reddit“ web scraping
* **Laurita Radiševska** – Sketch and design of the website, documentation
* **Benas Sakalauskas** – Front-end, Back-end
* **Matas Lekavičius** - Back-end responsible for „Twitter“ web scraping
* **Ignas Vigelis** – Documentation

## Development tools

### Systems and platform used

Figma – a platform used for user interface and user experience design.

Trello – web-based list-making application for task planning/management.

Git - a version control system that lets to manage and keep track of source code history.

GitHub – cloud-based hosting service used to manage Git repositories.

Draw.io – web page used for creating documentation diagrams.

### Technologies used

JavaScript – an object-oriented scripting programming language based on the principle of prototypes. Most often, the language is used to realize the interactivity of web pages, but it is also used as an opportunity to manipulate certain programs with scripts.

Python – a high-level, general-purpose programming language.

HTML – a markup language used to create websites.

CSS – a programming language used to style documents written in HTML.

Firebase – a cloud-hosted database.

ReactJS- JavaScript library used to create user interface components

NodeJS- an open-source server environment.

## Functional requirements

|  |  |
| --- | --- |
| Nr.1 | Requirement type: functional |
| The system must let the user create an account. Fields that need to be filled: email, repeat email, password, repeat password. | |
| Nr.2 | Requirement type: functional |
| The system must let the user sign in to his account. Fields that need to be filled: email, password. | |
| Nr.3 | Requirement type: functional |
| The system must let the user choose which site to scrape – “Reddit” or “Twitter”. | |
| Nr.4 | Requirement type: functional |
| The system must scrape your word from the chosen site. | |
| Nr.5 | Requirement type: functional |
| The system must show the user a list of words that are in posts with “keyword”. | |
| Nr.6 | Requirement type: functional |
| If the user is signed in to his account, the system must also let the user see results in a word cloud. | |

## Non-Functional requirements

|  |  |
| --- | --- |
| Nr.1 | Requirement type: non-functional |
| The website must be functional on all browsers. | |
| Nr.2 | Requirement type: non-functional |
| The list of words from posts needs to be presented from the most frequent to the least frequent. | |
| Nr.3 | Requirement type: non-functional |
| The scraping results need to be presented clearly. | |
| Nr.4 | Requirement type: non-functional |
| The results of scraping need to be presented in less than 10 minutes. | |

* 1. Users

|  |  |
| --- | --- |
| Role | Functions |
| Unregistered user | 1. Registration 2. Search posts by “keyword” from “Twitter” 3. Search posts by “keyword” from “Reddit” |
| Registered user | 1. Logging into the account 2. Search posts by “keyword” from “Twitter” 3. Search posts by “keyword” from “Reddit” 4. View data in the word cloud |
| Administrator | 1. System updates 2. Manage the content of the website |

# WORKFLOW

## Prototype

When the team finally arranged the main idea before the start of the project creation, it was essential to produce the website prototype. After the agreement on the structure of the website was made, Laurita Radiševska created two versions of the design with different elements layouts and color pallets. The team has chosen a design, which is more user-friendly and minimalistic. It was built using Figma, which allowed us to test the flow of the web page and point out its flaws.

# Back-end

One of the main points of our project was to scrape the data from social media platforms such as “Twitter” and “Reddit”. To do that Matas Lekavičius and Alanas Pauša needed to get these websites API‘s and produce a perfectly functioning code with filtered-out repeating words.

### “Twitter” Scraping

To get access to Twitter API, Matas Lekavičius, responsible for “Twitter” scraping, registered to “[developer.twitter.com](https://developer.twitter.com/)”. This web page provided our team with keys and tokens for access to the Twitter API. However, first challenges emerged, and it appeared that our team was given only “Essential access”, which means that a few Tweet search commands were unavailable. To get all the commands, it was necessary to fill out the request for “Elevated access”, which was approved in the span of a few days. For our project, it meant that monthly scraping limits increased from 500 thousand to 2 million Tweets, which would make our work results more elaborate.

## “Reddit“ Scraping

To scrape the data from “Reddit“ it was also essential to get access to its API. This task was done by Alanas Pauša, who registered to get a client ID and client secret. Later, this information was put into a code for “Reddit“ scraping. However, having only ID and secret in the code was not enough and it did not create any output. So, it was decided to add PRAW(Python Reddit API Wrapper) library to the code, which provided us with simple access to Reddit’s API.

## Front-end

The first step was choosing tools. It was not hard, since Justinas Rimavičius has experience doing work as a full stack developer and he chose that we do the project using JavaScript, ReactJS, and NodeJS. Making the front-end was simple, however after doing the authentication and basic design of the website the first problem came upon us – Python code integration with web technologies. Fortunately, this hassle was easily fixed by using the NodeJS ecosystem which has many different modules. When the user writes the “keyword” it first goes to the webpage server (NodeJS) and at the same time, the server calls forth the Python codes created by Matas Lekavičius and Alanas Pauša.

* + 1. Word cloud

We decided to have the option of showcasing the results in a “Word cloud” chart. Benas Sakalauskas worked on it. He managed to do it quite easily, however, he faced a few difficulties. The first one was obviously to make the chart look pretty and the second was to make the “words” proportionally big or small depending on how many times the word was in the posts.

## Database

For the database, we used “Firebase”, which is a platform backed by Google. In our case, this app development platform was convenient because it provides an authentication tool. One of the main advantages of this choice is that data is encrypted automatically, therefore authentication encryption on the website is not needed. Also, if the system would be updated with a new set of data(e.g., saving input of the user for data visualization), “Firebase” would be used for data storage. In addition, this system has monitoring features which means it is possible to observe the size of written/read data in real time.

# Functionality walkthrough

In this section will be presented expected visuals(prototype) and descriptions of the main functions along with real results.

### A home page

Any time the page is booted up the user will be seeing this home page.

The user is given three options to choose from:

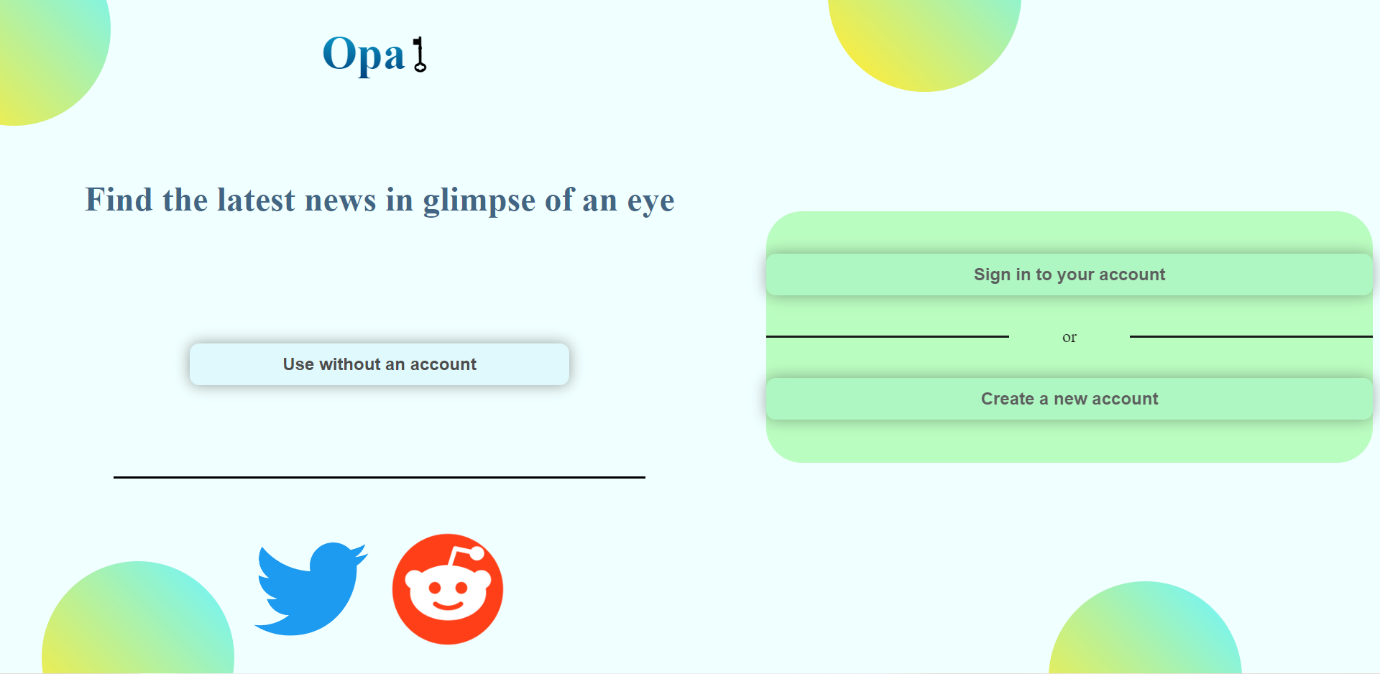
1) Create an account for the website

2) Log in to an existing account

3) Try using the site without an account

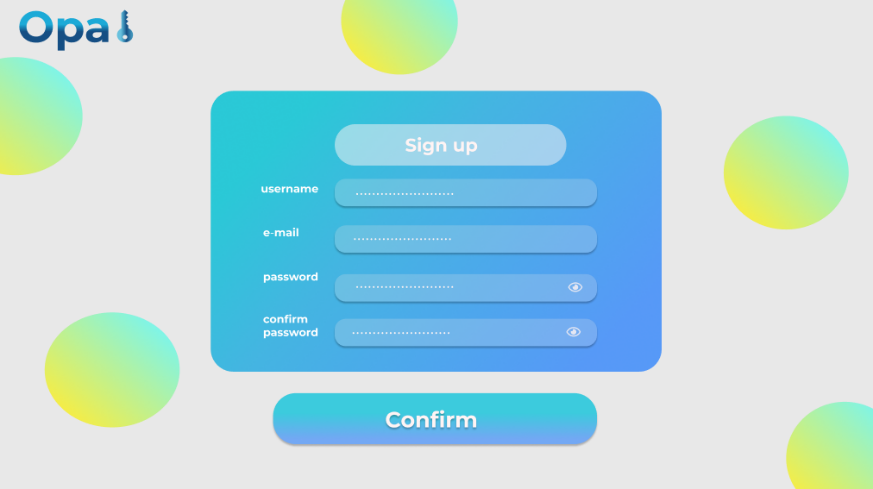
Paveikslėlis, kuriame yra žinutė

Automatiškai sugeneruotas aprašymasExpected results:

 Real outcome:

### Log in/Sign up pages

If the user decides to sign up, he will be asked to create a username, enter a valid E-mail, and password and to confirm it information by rewriting.

Expected results:

Real outcome:

Paveikslėlis, kuriame yra žinutė

Automatiškai sugeneruotas aprašymasExpected results:

Real outcome:

### Platform selection

Paveikslėlis, kuriame yra žinutė

Automatiškai sugeneruotas aprašymasOn this page the user has the ability to choose which site does he want to scrape.

### Keyword input without registration

When the user clicks on one of the two icon-buttons, he will see basic keyword search page without any detailed view. He will only have basic functions of the site – to search posts with “Your word” and to see the results.

Expected results:

Graphical user interface, application, chat or text message

Description automatically generated

Real outcome: Paveikslėlis, kuriame yra žinutė

Automatiškai sugeneruotas aprašymas

### Keyword input for registered user

If the user decided to register or log in, he will be able to see the results in a “Word cloud” chart

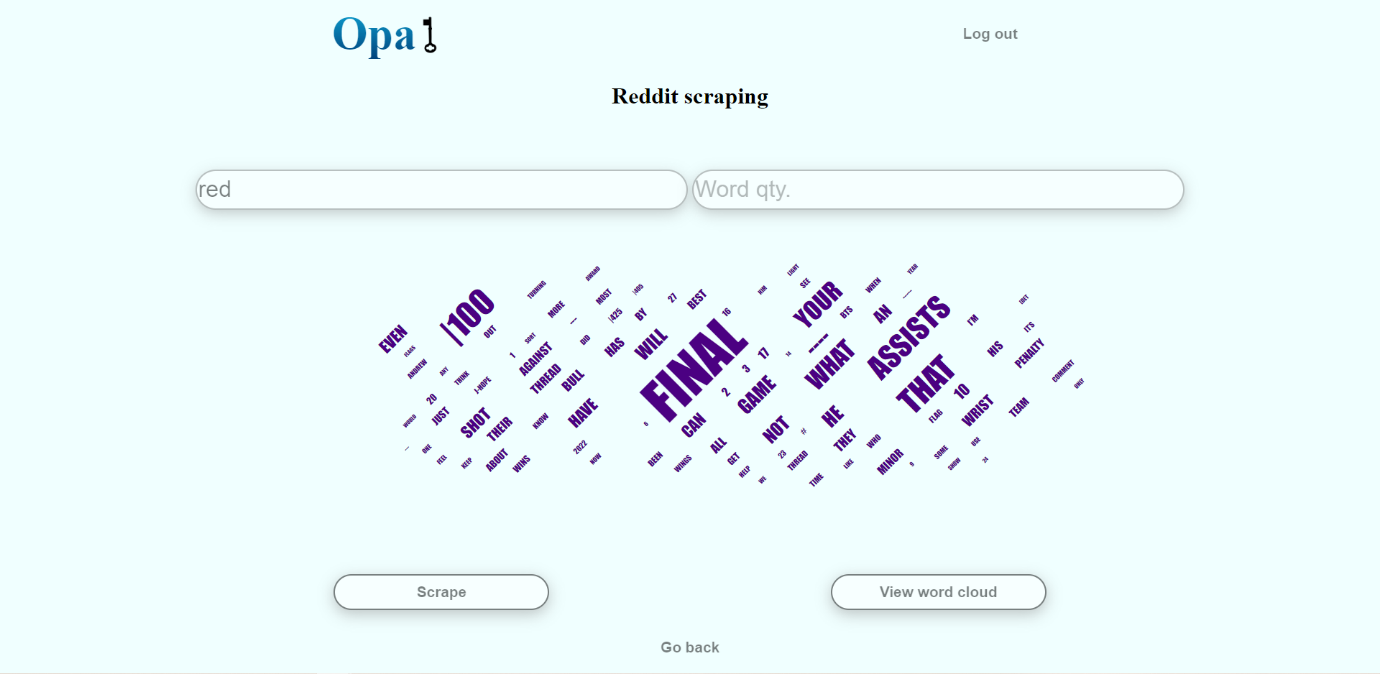
Graphical user interface, text, application, chat or text message

Description automatically generatedExpected results:

Paveikslėlis, kuriame yra žinutė

Automatiškai sugeneruotas aprašymasReal outcome:

### Word cloud

Registered user is able to see the scraping results in a word cloud by pressing “View word cloud”

### Profile view (was expected to do)

Registered user will be able to enter his personal profile, where he would find saved search results and information about them.



After choosing “Detailed data view” the user will be able to see the word cloud with the chosen keyword.

## QUALITY ASSURANCE AND TESTING

Automated unit testing was performed on separate modules of the website - Login and Sign-up functions. For this task it was chosen to use React testing library(for ReactJS code) and Jest(for JavaScript).

Other functional and non-functional requirements of the website were tested manually by the members of the team.

## GRAPHS AND VISUALIZATION

### User case diagram

Diagram

Description automatically generated

### Activity diagram

Diagram

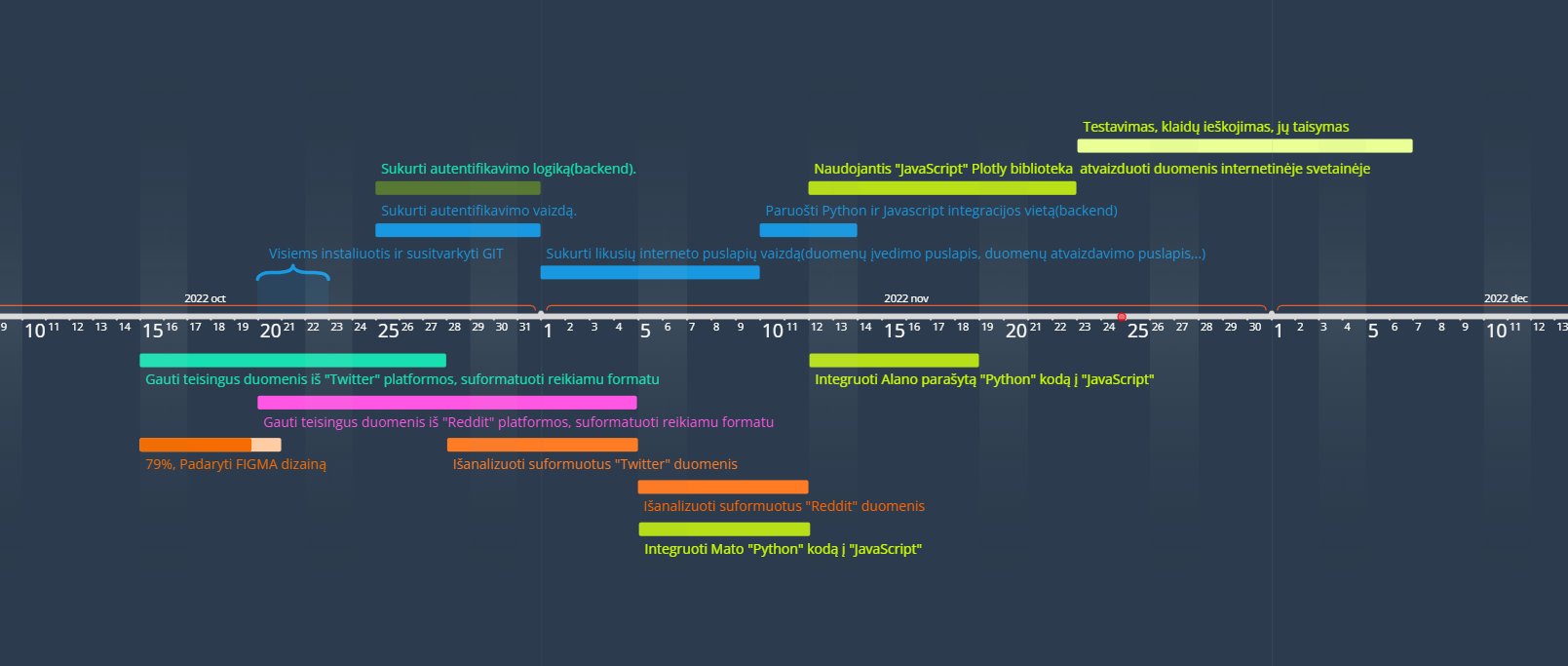
Description automatically generated

### Navigation model

A picture containing diagram

Description automatically generated

### Timeline



### Data model

|  |  |
| --- | --- |
| **Name** | **Type** |
| Email | String |
| Password | String |
| ScrapedWord | String |
| ResultWords | String |