

„ALEXANDRU IOAN CUZA” UNIVERSITY OF IAȘI  
**COMPUTER SCIENCE UNIVERSITY**



DISSERTATION THESIS

# Algorithm Compendium

proposed by

***Vasile-Octavian Fermuș***

**Session:** *July, 2016*

Scientific coordinator

**Asist. Dr. Vasile Alaiba**

„ALEXANDRU IOAN CUZA” UNIVERSITY OF IAȘI  
COMPUTER SCIENCE UNIVERSITY

# Algorithm Compendium

***Vasile-Octavian Fermuș***

**Session: *July, 2016***

Scientific coordinator

**Asist. Dr. Vasile Alaiba**

---

## STATEMENT REGARDING ORIGINALITY AND RESPECT COPYRIGHT

I hereby declare that the dissertation thesis with the title “Algorithm Compendium” is written by me and wasn’t submitted to another university or higher education institution in the country or abroad. Also, I hereby declare that all the sources, including the ones retrieved online are appropriately cited in this paper, respecting the rules of avoiding Copyright infringement:

- all exactly reproduced excerpts, including translations from other languages, are written using quotes and hold an accurate reference to their source;
- rephrased texts written by other authors hold an accurate reference to the source;
- the source code, images, etc. taken from open-source projects or other sources are used by respecting copyright ownership and hold accurate references;

summarizing ideas of other authors hold an accurate reference to the original text

Iași,

**Graduate: Vasile-Octavian Fermuș**

---

## STATEMENT OF CONSENT

I hereby declare that I agree that the dissertation thesis titled “Algorithm Compendium”, the application source code and other content (graphics, multimedia, test data, etc.) presented in this paper can be used by the Faculty of Computer Science.

Also, I agree that the Faculty of Computer Science, “Alexandru Ioan Cuza” University of Iași can use, modify, reproduce and distribute the application, executable and source code, in non-commercial purposes, created by me for the current thesis.

Iași,

**Graduate: Vasile-Octavian Fermuș**

---

# Introduction and Motivation

Through this application I propose a system for collecting algorithm that would be useful to programmers in any field of IT that search for solutions to problems that exist in personal applications, representing an alternative to search engines.

## Subject approach motivation

The reasons I am picking this theme form my dissertation thesis are the following:

- My passion for Web Development, cultivated in the university years
- One year of experience as a front-end developer on web applications and my desire to continue working in this field of IT
- The necessity of familiarizing myself with the back-end part of web development in a continuously more competitive environment
- My experience as a student and developer, always in need of answers regarding homework or work related applications.

At the present time, there are several platforms where users can find help to their problems but none of them treat the issue the way I plan to. Safe to mention are platforms as “W3Schools”, a platform that comprises a high array of guides for web development and tutorials, but without user input, or “Stack Overflow”, a platform that tries to help users by letting them ask questions regarding various topics while other users answer, but without great built-in support (at least in my opinion).

What I plan on doing is creating a blend between the group and private messaging functions of Facebook, the comment system of blog services, the ask/respond model of platforms like Stack Overflow and the possibility of commenting certain parts of content, as seen on web applications such as Soundcloud.

This way, users can request certain algorithms and others can create a post based on that request, leaving others to discuss said post as a whole or at certain lines in the algorithm, for a complete feedback and to help viewers understand the shown processes.

---

## Contributions

The theme of this dissertation thesis was picked by me and approved by my coordinator, Asist. Dr. Alaiba Vasile.

I managed on developing this application based on my experience as a web developer and understanding of how the workflow of a web application and the building process work. Without prior advanced knowledge of back-end eff, I had to learn to use MySQL, PHP and its framework Laravel effectively, in order to complete the thesis, which was one of my objectives.

The student-coordinator collaboration was based on periodic reports on my advancement and the group created on the Facebook social platform to answer any of my questions and possible issues.

I also used guidance in the form of tutorials and guides over the internet, mainly about the back-end part of the development cycle, which I was not accustomed to.

From halfway through the application development I started using a version control system, using GitExtensions, uploading the application on the online project hosting website Github.

By using a version control system, my progress was trackable by anyone, including my coordinator, by progress meaning any line of code written or deleted in the application, along with added/deleted functions and features.

This way I managed to create this thesis application, to the best of my knowledge, trusting that I achieved what I planned on accomplishing.

## Used Technologies

### Front-End

- HTML5 for creating the view of the application
  - SCSS/CSS3 for styling pages
  - JavaScript with the jQuery library for scripting and ajax
  - jQuery-UI for certain UI elements such as drag-ables
  - Ace editor JavaScript library for the algorithm view
  - Tag-it JavaScript library for tags on algorithm search
-

## Back-End

- PHP5 as the back-end language
- MySQL was used to store data
- Laravel was the framework I chose to build my website, due to familiarity, the offered template and session management

## Thesis Contents

In the first chapter I will show and comment other platforms and applications that also target helping developers, along with details about differences and similarities between them and my platform.

In the second chapter of the thesis I will present the application that this thesis is built upon. I will present the structure, used technologies and use in detail, along parts of code that are to be considered important.

In the third and last chapter I will write down my conclusions and present ways in which this type of platform can be improved and extended, along difficulties met along the way.

---

# Table of Contents

Chapter I – Alternatives and Examples .....	10
1.1 Stack Overflow .....	10
1.2 W3Schools .....	11
1.3 Code School .....	12
1.4 Conclusions .....	13
Chapter II – Implementation Details.....	14
2.1 Structure and Functioning Mode.....	14
2.2 Implementation.....	15
2.2.1 Landing Page .....	15
2.2.2 Home page .....	17
2.2.3 Edit Algorithm Page .....	24
2.2.4 Posts Page .....	25
2.2.5 Notifications page .....	28
2.2.6 Private Messaging and Groups .....	29
2.2.7 Profile Page .....	34
2.2.8 The Administrator page .....	36
2.3 Graphical Interface Details.....	40
2.4 Used Technologies – Front-End.....	41
2.4.1 HTML5 (HyperText Markup Language) .....	41
2.4.2 CSS3 (Cascading Style Sheets 3).....	42
2.4.3 JavaScript .....	42
2.4.4 jQuery Library.....	42
2.4.5 Bootstrap Framework .....	43
2.4.6 Grunt .....	43
2.5 Used Technologies – Back-End .....	43
2.5.1 PHP5 .....	43

---



2.5.2	MySQL .....	44
2.5.3	Laravel 4.2 Framework.....	44
Chapter III – Conclusions .....		46
Bibliography .....		47

# Chapter I – Alternatives and Examples

In this chapter I will present different existing alternatives to my thesis application. Besides presenting their pros and cons, I will also compare them to my own approach. The application I built has its roots in these existing applications so I find it important to discuss them, building upon the viability of my approach.

## 1.1 Stack Overflow

Stack Overflow is a question and answer site for professional and enthusiast programmers.<sup>1</sup>

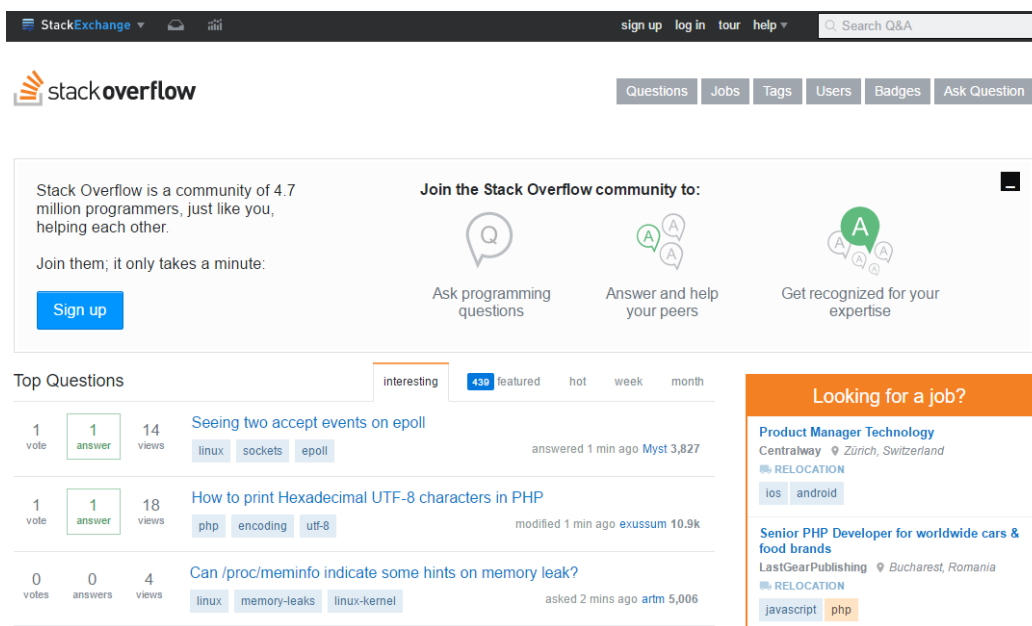


Figure 1 - Stack Overflow Front Page

It allows users to input questions for others to answer, provided that they are related to programming and are concise enough. Anyone can see the contents of the website but only registered users can ask questions or answer them. Most programmers use this website in search for answers due to its abundant array of answered questions, the website being created in 2008.

---

<sup>1</sup> <http://stackoverflow.com/tour>

Despite its advantages, it lacks in-site support for code writing, users having to resort to outside applications and links to provide advanced help, such as showing and explaining code in a clean, visible way. Even if the website revolves around questions and answers, users cannot request code, but only ask for help regarding already existent pieces.

Comparison between Stack Overflow and Algorithm Compendium	
Similarities	Differences
<ul style="list-style-type: none"><li>• Both applications aim to help people improve their coding experience</li><li>• Both applications store information with the help of the user base only</li><li>• The usefulness of the application grows based on user interaction</li></ul>	<ul style="list-style-type: none"><li>• Stack Overflow doesn't allow requesting algorithms, only encouraging questions based on existent code and issues</li><li>• Stack Overflow allows unregistered users view content</li></ul>

## 1.2 W3Schools

W3Schools is a web developers site, with tutorials and references on web development languages such as HTML, CSS, JavaScript, PHP, SQL, and JQuery, covering most aspects of web programming. The site derives its name from the World Wide Web (W3), but is not affiliated with the W3C. W3Schools was originally created in 1998, by Refsnes Data, a Norwegian software development and consulting company.<sup>2</sup>

It's a really useful resource for web developers, since it features a lot of examples and guides, from basic to more advanced. It offers a clean interface, albeit outdated. The code is easy to see and it offers the possibility of trying out and experimenting with existent code on the website, without resorting to third party applications. On the other side it doesn't offer any kind of user interaction or support, lacking comment section or discussion regarding the existent content.

---

<sup>2</sup> <http://www.w3schools.com/about/default.asp>

Comparison between W3Schools and Algorithm Compendium	
Similarities	Differences
<ul style="list-style-type: none"> <li>Both applications aim to help people improve their coding experience</li> <li>Both applications offer a clean layout to display code</li> <li>Code snippets or full algorithms are supported on both websites</li> </ul>	<ul style="list-style-type: none"> <li>W3Schools doesn't support user interaction</li> <li>W3Schools doesn't let users add content and improve upon the existent content in any way</li> <li>Algorithm Compendium offers a broader spectrum of topics, while W3Schools only works with Web related programming</li> </ul>

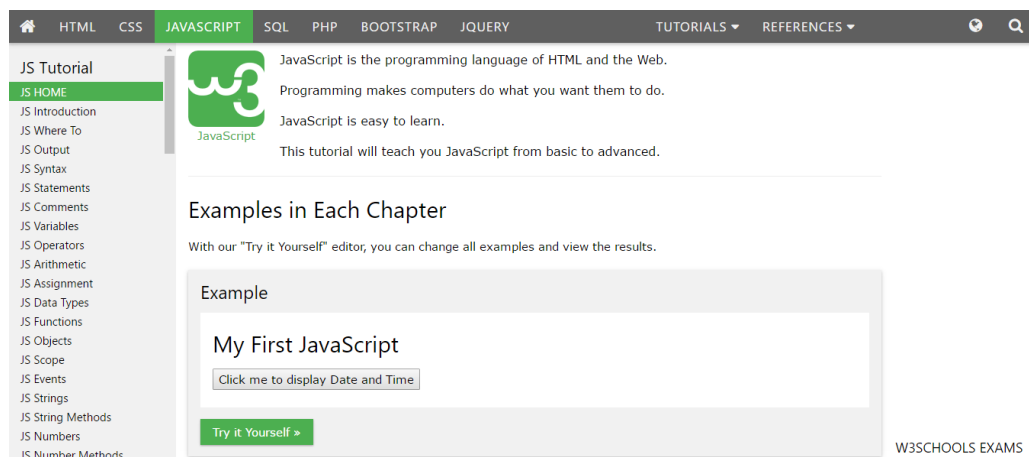


Figure 2 - W3Schools Website Example page

## 1.3 Code School

Code School is an online learning destination for existing and aspiring developers that teaches through entertaining content. Each course is built around a creative theme and storyline so that it feels like you're playing a game, not sitting in a classroom. We combine gaming mechanics with video instruction and in-browser coding challenges to make learning fun and memorable.<sup>3</sup>

This website is a paid service that offers high quality guides, along videos created by highly talented programmers. User interaction is nonexistent though, and a normal user cannot add

<sup>3</sup> <https://www.codeschool.com/about>

content to the website or request certain content. It is a great way to learn programming and it offers a wide array of guides, complete with quizzes and tests.

It must be said that there are multiple sites that offer this kind of service, this being just an example I am accustomed with, learning JavaScript using its courses.

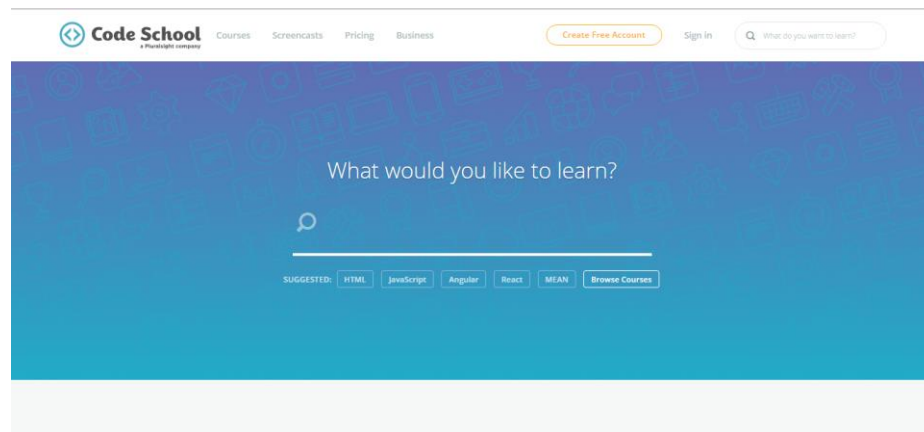


Figure 3 - Code School snippet of the main page

Comparison between Code School and Algorithm Compendium	
Similarities	Differences
<ul style="list-style-type: none"><li>Both applications aim to help people improve their coding experience</li><li>Both applications offer a clean layout to display code</li><li>Code snippets or full algorithms are supported on both websites</li></ul>	<ul style="list-style-type: none"><li>Code School doesn't let users add content and improve upon the existent content in any way</li><li>Algorithm Compendium does not have a subscription fee, being a free service.</li><li>Algorithm Compendium doesn't offer video support</li></ul>

## 1.4 Conclusions

This application takes example from many types of websites that are directed to programmers, but it's unique in its design, taking a bit of each to create a website that gathers data in order to build a collection of algorithms, with feedback for users and discussions revolving said algorithms for further understanding. My application also has unique features such as private messaging and groups, to increase user interactivity and commenting not only the algorithm as a whole but also sections of it.

# Chapter II – Implementation Details

## 2.1 Structure and Functioning Mode

In computing, a web application or web app is a client–server software application which the client (or user interface) runs in a web browser.<sup>4</sup>

Algorithm compendium is a web application itself, meaning that the only way to access it is through a web browser, online. The general purpose of this application is to create a collection of algorithms with the help of registered users in order to help students and programmers to learn new things or accomplish tasks related to IT faster. It's based on the same “Question/Answer” model as, for example, Stack Overflow, with the difference that Algorithm Compendium lets users request algorithms that other users can write and publish. Those algorithms can then be discussed in a multitude of ways and graded by the user base. Users can improve their trust by being commended by the other users.

Misconduct is punished by first warning users about their mistakes or banning them if the situation demands it. Banned users cannot use the site anymore using that account, unless they are unbanned.

The website is to be used free of charge, no presenting a subscription fee or any other kind of payment.

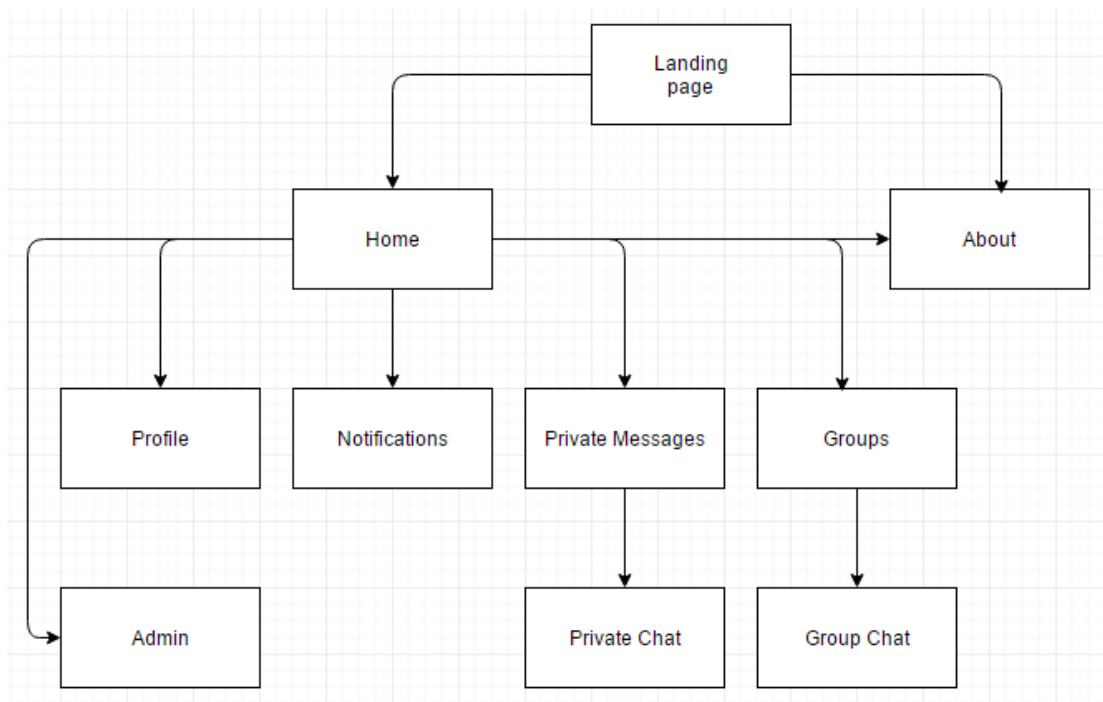
On the front page, the user can register or log in using a created account. If the user is not banned, it will be transferred to the main page, where he/she can submit his/her own algorithms, whether based on a user request or not or view other algorithms and user profiles, along with their list of submitted posts.

A user can message any other user privately, create or join groups where more people share the same private chat room. This feature has been implemented to ease user interaction and offer the possibility of private conversations between users in a clean, friendly manner.

---

<sup>4</sup> [https://en.wikipedia.org/wiki/Web\\_application](https://en.wikipedia.org/wiki/Web_application)

In the case of algorithm/profile comments or replies, the user is notified, having a notification page where he/she can see all the notifications received.



*Figure 4 - Crude map of the website navigation*

## 2.2 Implementation

In this subchapter I will discuss the features of the application, complete with pieces of relevant code and related APIs and database tables.

It is worth noting that most requests are called using jQuery's AJAX functions, sending and receiving JSON that the application then parses and displays to the user.

### 2.2.1 Landing Page

This is the first page that the user interacts with, allowing him to create an account or logging in with an existent account. Also, the user can check one other page, named "About", which contains a link to the documentation and some general information. This page is only accessible if a user is not logged in.

Algorithm Compendium
Home
About

Start using Algorithm Compendium!

This application takes example from many types of websites that are directed to programmers, but it's unique in its design, taking a bit of each to create a website that gathers data in order to build a collection of algorithms, with feedback for users and discussions revolving said algorithms for further understanding.

Algorithm Compendium also has unique features such as private messaging and groups, to increase user interactivity and commenting not only the algorithm as a whole but also sections of it.

Register
Login
Forgotten Password

Login

University of Alexandru Ioan Cuza  
Fermus V. Vasile-Octavian

Figure 5 - Landing page

### 2.2.1.1 Controllers and function details

The controller that implements the page features is UsersController, using the “users” table to store register data and check login credentials.

Below, the “users” table is displayed.

The „user\_type” field refers to the status of the users, as follows:

- 0: Banned user
- 1: Normal user (default)
- 2: Moderator: can ban normal users
- 3: Admin: can ban any users and demote or promote users and moderators

#	Name	Type	Collation	Attributes	Null	Default	Extra
1	id	int(10)		UNSIGNED	No	None	AUTO_INCREMENT
2	email	varchar(100)			No	None	
3	user_type	int(11)			No	None	
4	last_name	varchar(255)			No	None	
5	first_name	varchar(255)			No	None	
6	password	varchar(64)			No	None	
7	created_at	timestamp			No	0000-00-00 00:00:00	
8	updated_at	timestamp			No	0000-00-00 00:00:00	

Figure 6 - “users” table



## APIs

**users/create (method: post)** – This receives as input the user's email, last name, first name and password. If the input is valid the new user is inserted into the database and the account is logged in.

**users/signin (method: post)** – This receives as input the user's email and password. If the data corresponds to an existent account, the user is logged in.

### 2.2.2 Home page

This is the default page upon logging in and the main hub for the user. From here, the user can navigate anywhere on the website, aside from the admin page if the account is not marked as a moderator or administrator. The page has three main tabs:

- My posts – Offers a list of algorithms the user has created
- Search – Lets the user search for algorithms on the website; if he cannot find an algorithm he can create a request.
- Post new algorithm – Offers an interface for the user to create a new post regarding an algorithm, complete with description and if applicable original post link if the algorithm was copied from an outside source. When posting, the user can create an algorithm based on an existent request.

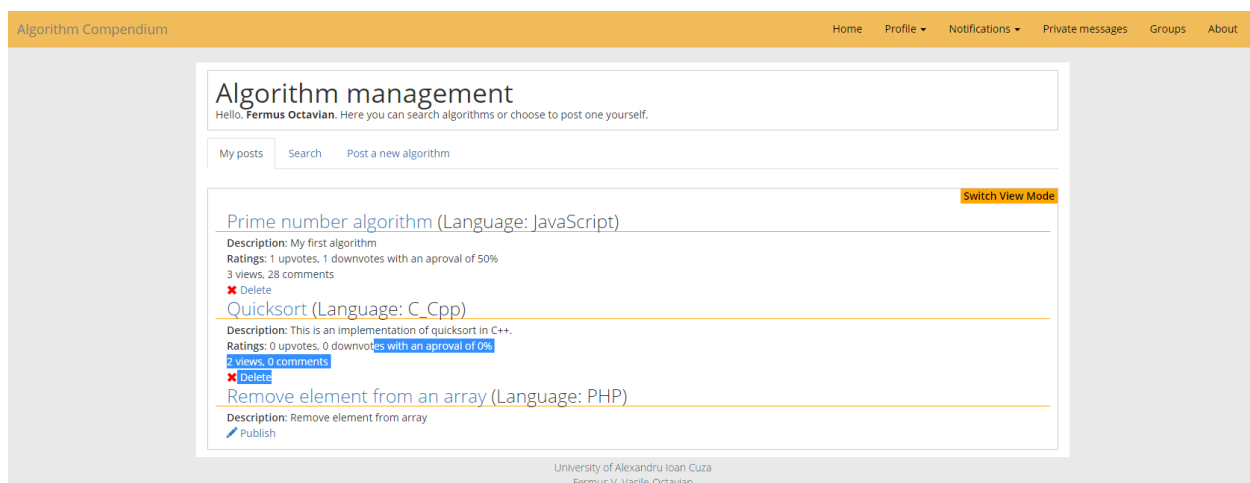


Figure 7 - Home page

### 2.2.2.1 Controllers and function details

The controllers that implement the functionalities of this page are:

- NotificationsController – Notifies user when something that might interest him has happened. The used table is „notifications” and the used API is „notifications/all”.
- PostController – Used to create or search algorithms on the website. The main used table is „algorithms”, although it has helper tables for vote and view counts. The used APIs are „post/myalgorithms”, „post/search” and „post/pushalgorithm”
- RequestController – Used to handle algorithm request creation, searching and voting. The main used table is „algorithm\_requests”, although it also uses a table for voting requests called „algorithm\_request\_votes”. The HTTP requests used are „requests/all”, „requests/vote” and „requests/submit”
- ReportController – This controller handles the report functions, where a user can notify moderators and admins regarding “bad” content. The used table is „reports” and the API is “reports/report”.

#### APIs

**notifications/all (method: get)** – fetches all the notifications of that user, returning them in a JSON which is used to populate the notifications list in the navigation bar, allowing the user to see the newest ones.

Note: “notifications/all” is called in every page when the user is logged and will not be explained in other sections, except the Notifications Page, where it is used differently.

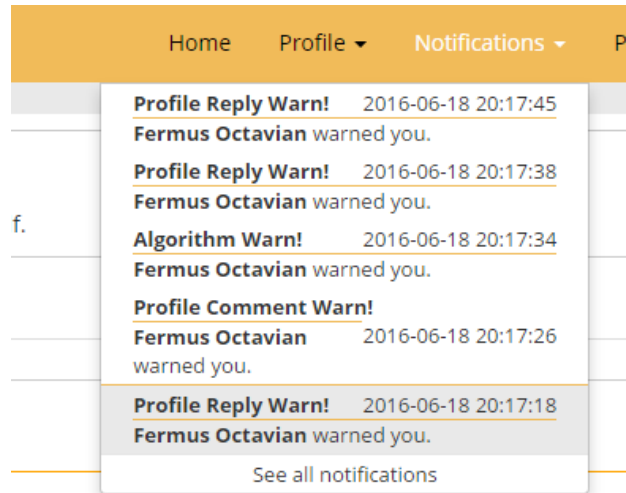


Figure 8 - A notification bar example

As we can see, the newest five notifications are shown, differentiating clicked and unclicked notifications, to provide visual cues as to what the user already took notice of.

**post/myalgorithms (method: get)** – This API returns an object that contains a list of all the algorithms the user has created, templated or published.

**post/delete (method: delete)** – This accepts as input an id. If there exists an algorithm with that id that was created by the user calling the API it will delete the algorithm from the database.

**post/publish (method: put)** – Similar to “post/delete”, with the difference that it also checks if the post algorithm content is empty. If it is not, it sets the algorithm as published in the database, making it visible in search results.

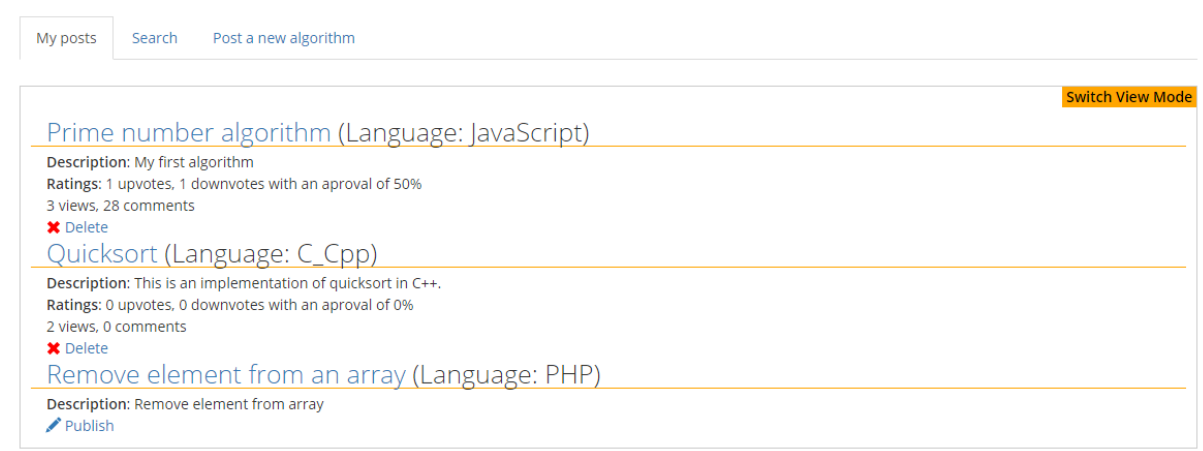


Figure 9 - “My posts” tab in the home page

**post/search (method: post)** – As input, this HTTP request receives a “tags” field which is a string of words separated by space, a “language” field which contains a string and a “ratio” Boolean field, returning a JSON object containing a list of algorithms that include one or more of the tags and are written in the requested programming language. The ratio Boolean field decides if any algorithms that have more dislikes than likes should be included in the returned list.

Example of Request Object	Example of Response Object
<pre>{   tags: "prime",   language: "javascript",   ratio: true }</pre>	<pre>[   {     id: 1,     user_id: 1,     name: "Prime number algorithm",     reported: 0,     upvotes: 1,     user_id: 1,     username: "Fermus Octavian"     views: 3   } ]</pre>

[My posts](#)
[Search](#)
[Post a new algorithm](#)

**Keywords**

prime x

**Programming Language**

javascript

☐ Positive like/dislike ratio
 ☐ Don't include my own algorithms

Can't find what you are searching for? [Submit a request](#)

[Search](#)

## Prime number algorithm (Language: JavaScript)

By: [Fermus Octavian](#)

Description: My first algorithm

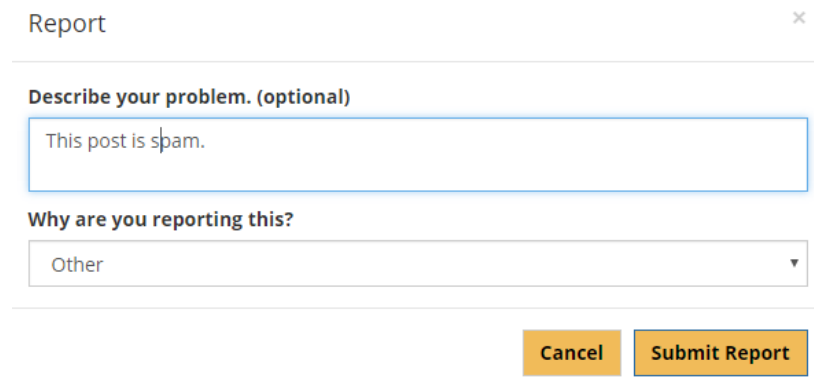
Ratings: 1 upvotes, 1 downvotes with an aproval of 50%

3 views, 0 comments

Figure 10 – “Search” tab on Home Page

**reports/report (method: post)** – If any kind of irrelevant/indecent content is found a report can be filed via a form and submitted using Ajax. The collected data will be used to create a comprehensive visualization of the problem without necessarily visiting the reported page. A user cannot report his/her own content.

Note: This request is used for every report on the application and will not be explained in other sections.



Report

Describe your problem. (optional)

This post is spam.

Why are you reporting this?

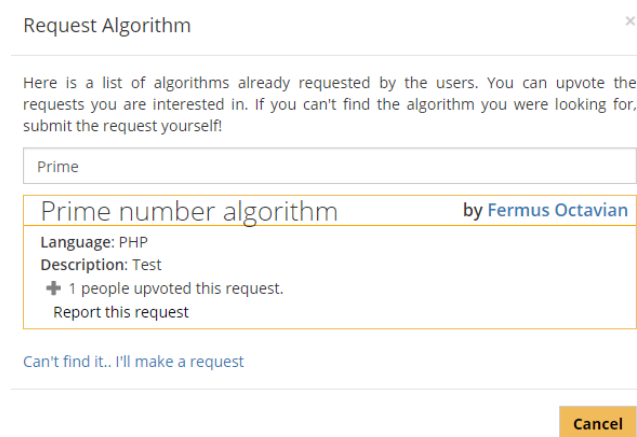
Other

Cancel Submit Report

Figure 11 - Default report modal

**requests/all (method: get)** – This returns the list of existent requests. Its inputs accept a Boolean field called “all\_requests” that, if false, will not return requests submitted by the logged user. That was implemented to disallow users to create algorithms based on their own requests.

**requests/vote (method: put)** – This request adds a vote point to an existent request based on a valid request id or removes the vote if it was already existent.



Request Algorithm

Here is a list of algorithms already requested by the users. You can upvote the requests you are interested in. If you can't find the algorithm you were looking for, submit the request yourself!

Prime

Prime number algorithm by Fermus Octavian

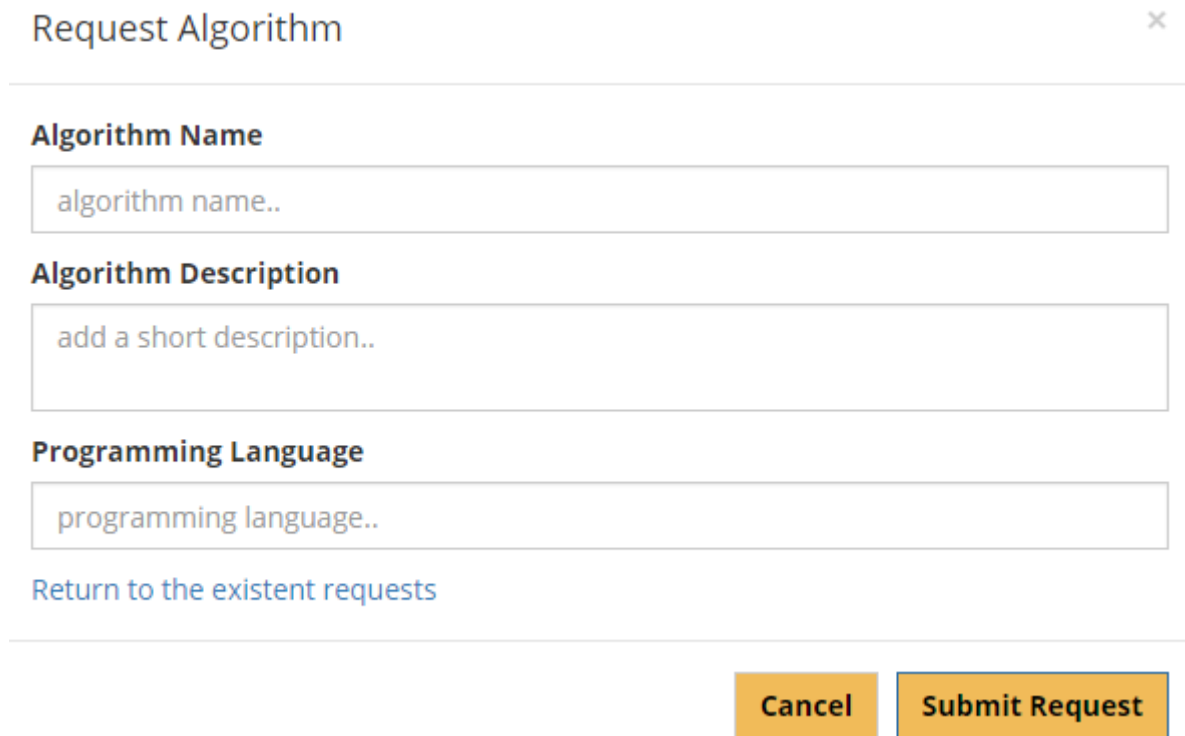
Language: PHP  
Description: Test  
+ 1 people upvoted this request.  
Report this request

Can't find it.. I'll make a request

Cancel

Figure 12 - Request algorithm modal example

**requests/submit (method: post)** – This accepts three mandatory fields that create a new request in the database, giving it a vote from the user by default.



Request Algorithm x

---

**Algorithm Name**

**Algorithm Description**

**Programming Language**

[Return to the existent requests](#)

Cancel Submit Request

*Figure 13 - The request algorithm modal used if no similar requests are found*

**post/pushalgorithm (method: post)** – Using this request a user creates a new post that contains the mandatory fields “name”, “description”, “language” and “template” (which, if set to false will make the post visible and uneditable). Also, there are two optional fields, “byrequest” and “original\_link”. The field “byrequest” is an id of the request chosen to answer and “original\_link” is filled if the algorithm is taken from an outside source, to avoid copyright infringement. The “content” field is optional if the user chooses to save the post as a template, for future editing, but becomes mandatory upon publishing.

[My posts](#) [Search](#) [Post a new algorithm](#)

Reset progress | [I want to create an algorithm based on a request..](#)

Algorithm name

algorithm name..

Programming Language

algorithm language..

Description

add a short description..

Original Link (optional)

original algorithm location..

Continue

Figure 14 - The “Post a new algorithm”, before filling the first three fields

[My posts](#) [Search](#) [Post a new algorithm](#)

Reset progress | [I want to create an algorithm based on a request..](#)

ultimate test

Language: JavaScript

By [Fermus Octavian](#)

Original Link: [Find the original link here](#)

Description: Let's see if everything by now is working

1

Save as template Publish

Figure 15 - The “Post a new algorithm”, after pressing “continue” in the first part of creating the post. A preview of the post page is created.

23 | Page

### 2.2.3 Edit Algorithm Page

This page is accessible by clicking the title of an unpublished algorithm in the “My algorithms” tab in the Home page. As I previously stated, a user can choose to save the post as a template for future editing, before publishing since he/she cannot change the contents after the post is made public. Here, the user can also choose to delete the unpublished algorithm, if so desired.

The controllers that implement the functionalities of this page are:

- NotificationsController – Same as on every page, outside the dedicated Notifications page.
- PostController – Used to edit an unpublished algorithm. The used APIs are „post/template”, „post/edit” and „post/delete”.
- RequestController – Used to handle algorithm request creation. The HTTP requests used is „requests/all”.

#### APIs

##### **notifications/all (method: get)**

**post/template (method: get)** – This API fetches an algorithm based on a valid id if the post is owned by the connected user and it is not currently published. The retrieved data is used to populate the UI so the user can change the contents.

**post/edit (method: post)** – It behaves similarly to „post/push”, with the difference that instead of creating a new post, it edits an existent, unpublished one that is owned by the user that called the API.



[Reset progress](#) | [I want to create an algorithm based on a request..](#)

**Algorithm name**

**Programming Language**

**Description**

**Original Link (optional)**

```

1 <?php
2     $a1=array("a"=>"red","b"=>"green","c"=>"blue","d"=>"yellow");
3     $a2=array("a"=>"purple","b"=>"orange");
4     array_splice($a1,0,2,$a2);
5     print_r($a1);
6     ?>

```

Delete Save as template Publish

Figure 16 - An example of "Edit algorithm" page contents

## 2.2.4 Posts Page

This page is used to view a published post. It allows a user to view the algorithm, comment certain lines of it or the algorithm as a whole in the comments section. The creator can be commended for his work and the algorithm can be upvoted or downvoted. Also, the view count of the post is incremented each time a logged user first visits it.

The controllers that implement the functionalities of this page are:

- NotificationsController – Same as on every page, outside the dedicated Notifications page.
- UserController – Used to commend the creator of the post or take back an existent commend point. Commendations are used to represent the number of users that like the content posted by creators. The used API is „put/commend”.
- PostController – Used to retrieve the contents of the post.

## APIs

### **notifications/all (method: get)**

**post/data (method: get)** – This API fetches a published post and the data related to it, namely algorithm data, view count, votes and comments and the number of commendations of the creator. It accepts an id as input and will return data if the id is of an existing, published algorithm.

**post/vote (method: post)** – It accepts a post id field and a vote field as input. If the vote already exists, this API will modify the vote the user gave to the post. Otherwise, it creates a new row in the database, containing the details about the user's action.

**users/commend (method: put)** – It accepts a user id field as input, adding or deleting a commendation given from a user. To note that a user can only receive one commendation from another and can be removed at any given moment. A user with a high commendation number is generally seen as a better and more valued content creator.

**post/commentline (method: post)** – As input, this API accepts a post id field, a line field which represents on which line the comment was submitted and the text of the comment, adding a new row in the line comments table, returning the new comment list for update. When commenting, a notification will be sent to the user that created the post and the users that commented on the same line.

**post/votecommentline (method: post)** – Similar to „post/vote”, with the difference that the input contains line comment id instead of the id of the post.

**post/deletecommentline (method: delete)** – Accepts the comment id. If the id is of a comment posted by the user, it will set a field in the database called „visible” to false. This way, the message still exists, but its contents are not shown. A user cannot restore a deleted comment.

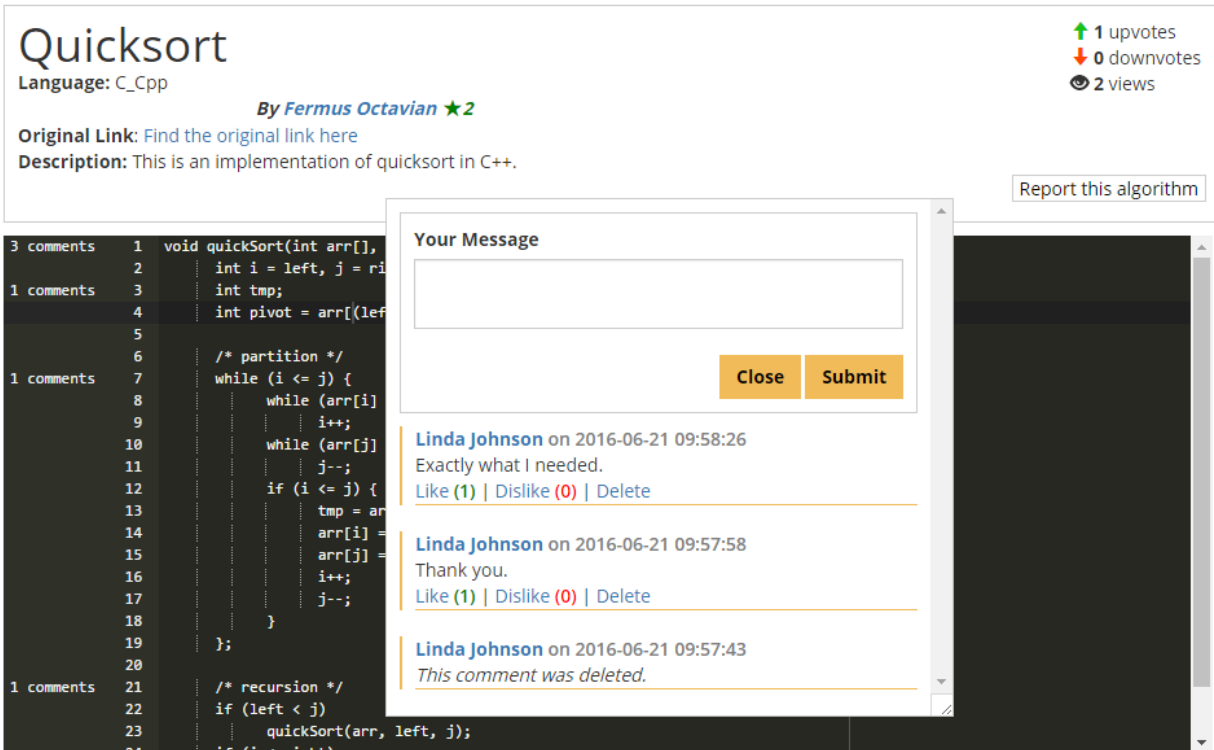


Figure 17 - The “Post” page inline comments functionality

**post/comment (method: post)** – The request accepts a valid post id and the text contained in the comment, similar to „post/commentline”, returning the updated comment list. When commenting, a notification will be sent to the user that created the post and the users that commented on the same algorithm.

**post/votecomment (method: post)** – Similar to „post/vote”, but interacts with the comments table, instead.

**post/deletecomment (method: delete)** – Similar to „post/deletelinecomment”.

**post/reply (method: post)** – The request accepts a valid post id and the text contained in the comment, similar to „post/commentline”, returning the updated comment list and it also accepts a comment id instead of a line id as input. When commenting, a notification will be sent to the user that created the post and the users that commented on the same algorithm.

**post/votereply (method: post)** – Similar to „post/votecommentline”, but interacts with the comment reply table instead.

**post/deletereply (method: delete)** – Similar to „post/ deletecomment”.

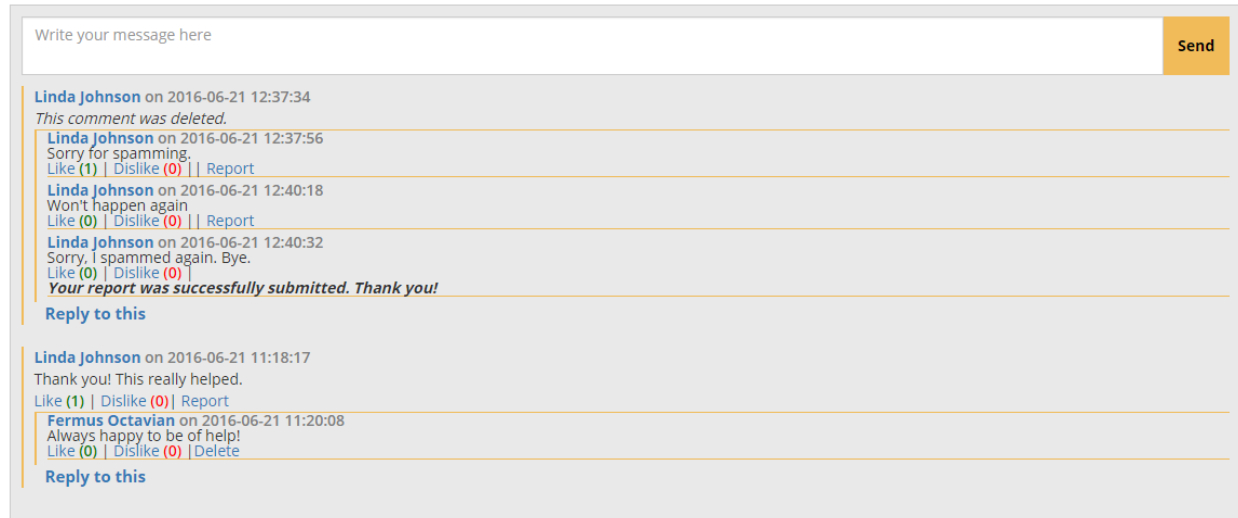


Figure 18 - Example of a comment section.

## 2.2.5 Notifications page

This page is used dedicated to the visualization of received notifications, which at the current time range from comments and replies to warnings given by moderators and administrators. Notifications that have been clicked are differentiated by color from notifications that weren't.

The controller that implement the functionalities of this page is NotificationsController, using „notifications/seeall” and „notifications/delete”.

### APIs

**notifications/seeall (method: get)** – Calling this API will return a full list of the notifications received by the user that were not deleted by him, giving him enough information to create a clear visualization of what each notification means. A user can then click on the link button to get to the content that was referred to or delete the notification with „notifications/delete”.

**notifications /delete (method: delete)** – This API receives a notification id and checks if the notification exists and it is directed to the connected user. If it is, it gets deleted from the database, since there is no need to set it to „not visible”, having no other purpose than being informative.

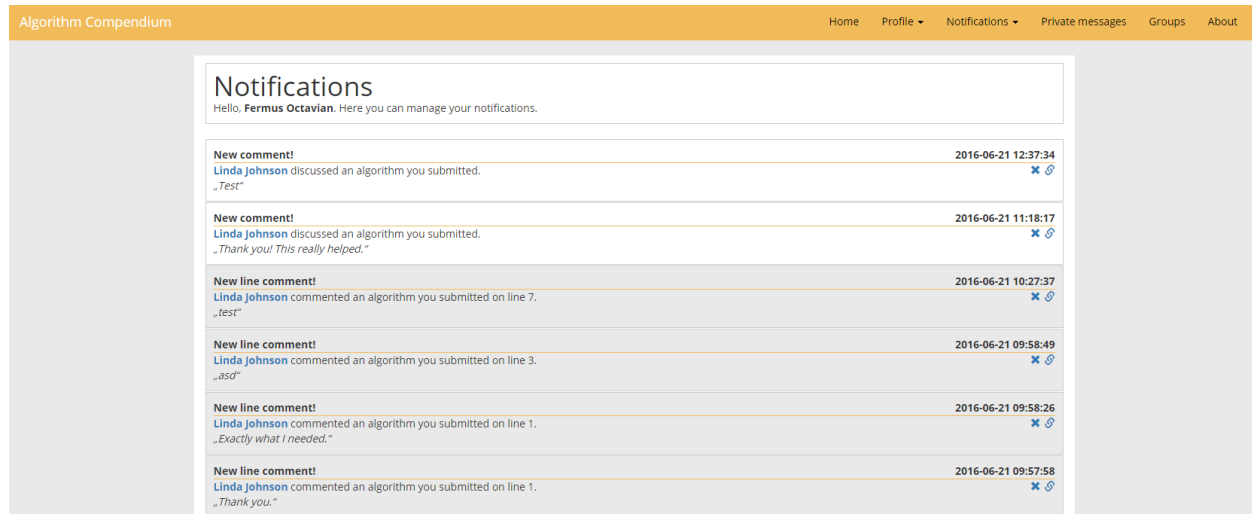


Figure 19 - Notifications page snippet

## 2.2.6 Private Messaging and Groups

The application offers the possibility of users talking privately to increase user interaction and decrease spam on the posts and profile pages. This can be done either one on one or in groups where more persons can be a part of. There are four views that help to achieve this purpose, namely the two main message/group pages where the user can execute different actions that are not directly related to chatting and the pages that implement the chatting options.

To be noted that the chat only gives the feeling of being in real-time, when it actually sends requests each second, to check for new messages.

The controllers that implement the functionalities of the messaging and group pages are `MessagingController` and `NotificationsController`.

- `NotificationsController` – Same as on every page, outside the dedicated Notifications page.
- `MessagingController` – Offers the possibility of private messages between two users and the tools to create private groups.

### APIs related to private messaging between two users

**messaging/messagecrumb (method: post)** – This API receives a timestamp or a null input and returns a list that holds the last message from each conversation sorted descending by time, and a new timestamp that decides if anything new was received. With this data, the app creates a

list of links on the sidebar. This API is called once every second, but only gathers data and updates the view if the timestamp is different.

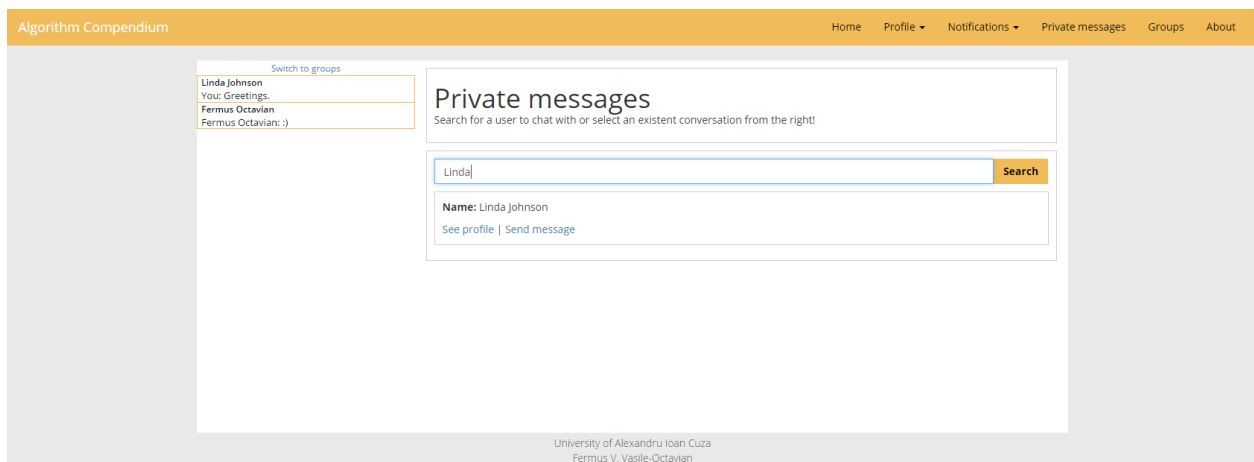


Figure 20 - The main private messages page

**messaging/getuserhistory (method: get)** – This API receives a user id and, if the user exists it returns the message history between the the two users, along with the list from „messaging/messagecrumb”. The first list will populate the main chat. This API is called once every half a second, but only updates if the timestamp received is different from the last.

**messaging/messageuser (method: post)** – Similar to the comments, it receives a text and the user id the message is destined to and updates the message history.

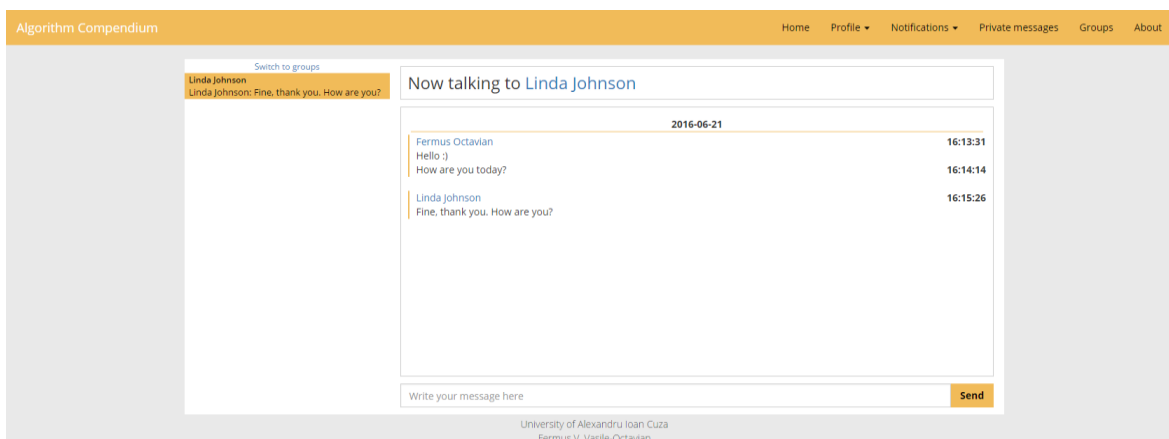


Figure 21 - Private chat example

## APIs related to groups

**„messaging/groupcrumb” (method: get)** – Similar to „messaging/messagecrumb” but in this case it applies to groups, retrieving the read status of the last messages from the groups the user is in.

**„messaging/groups” (method: get)** – Based on the authenticated user, the API sends a list of the groups the user is a member of or has requested entry to, along with some details about each given group, such as description and member count.

**„messaging/leavegroup” (method: post)** – This API is used when the connected user wants to leave a group he is in. If any remaining members exist in the group and the user that leaves is the current leader, the ownership is transferred to the oldest member of the group. Otherwise, the group is set to “invisible”, meaning it will not be found in searches or by direct link.

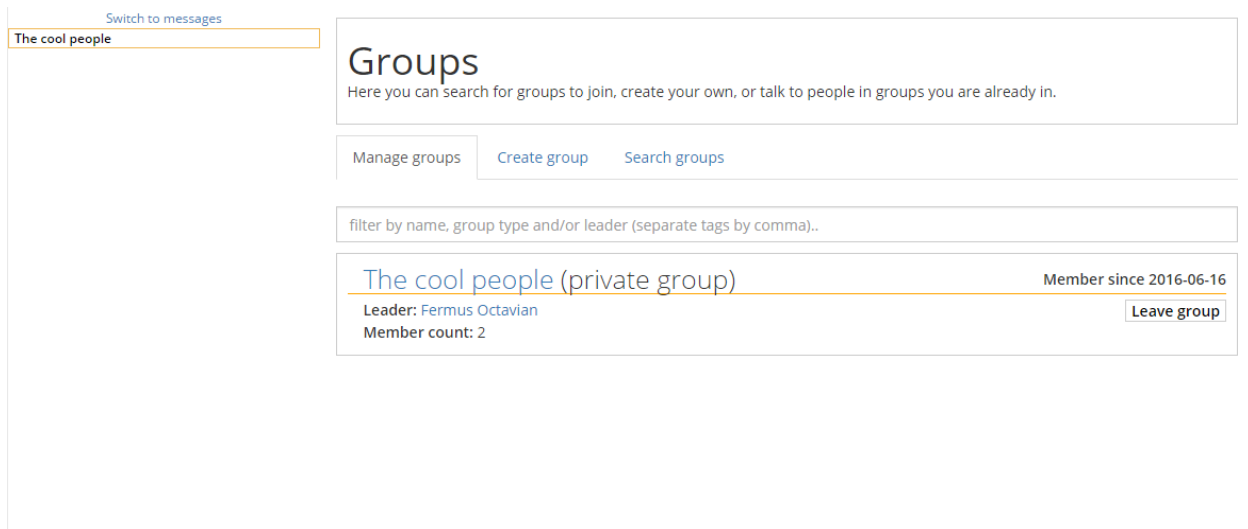


Figure 22 - The "Manage groups" tab of the "Groups" page

**„messaging/creategroup” (method: post)** – As input there exist two mandatory fields, named “group\_name” and “group\_type” and an optional group called “group\_description”. When the group is created, the creator is automatically appointed as leader. If the group is private, a user must request entry to the group and be accepted, while if the group is public, he will get entry automatically when the user applies for entry.

Figure 23 - The "Create group" tab of the "Groups" page

**„messaging/searchgroup” (method: post)** – When calling this request, a string field with the search data is submitted, returning all the existent visible groups with a name or description that matches the given input, along with details like current leader and member count.

**„messaging/joingroup” (method: post)** – As input, this API receives a group id. If the id is valid, one of two things happen: If the group is public, the user is granted entry automatically. If the group is private, a request is created instead. The leader of that group can accept or deny entry to the requesting user.

**„messaging/cancelrequest” (method: delete)** – This API cancels a request, deleting the field from the database if it exists and it is related to the connected user.

Figure 24 - The "Search groups" tab of the "Groups" page



**„messaging/groupinitialdata” (method: get)** – This API retrieves all the necessary data about the group if the user is a member, similar to “messaging/messagehistory”.

**„messaging/messagegroup” (method: post)** – Similar to “messaging/messageuser”, but with a group id instead of a user id.

**„messaging/kickfromgroup” (method: post)** – This API accepts a user id and a group id. It then checks if the user is the leader of the given group and if the user id belongs to another user in it. If that is found true, the user is kicked, also getting a notification.

**messaging/acceptgrouprequest” (method: post)** – This API is used to accept a member into a private group and it’s usable only by a leader. Similar to “messaging/kickfromgroup”, it accepts a user id and a group id and creates a notification for the accepted member.

**„messaging/denygrouprequest” (method: post)** – The inverse of the API called “messaging/acceptgrouprequest”. The difference is that instead of setting the “accepted” field to true, it deletes it from the table.

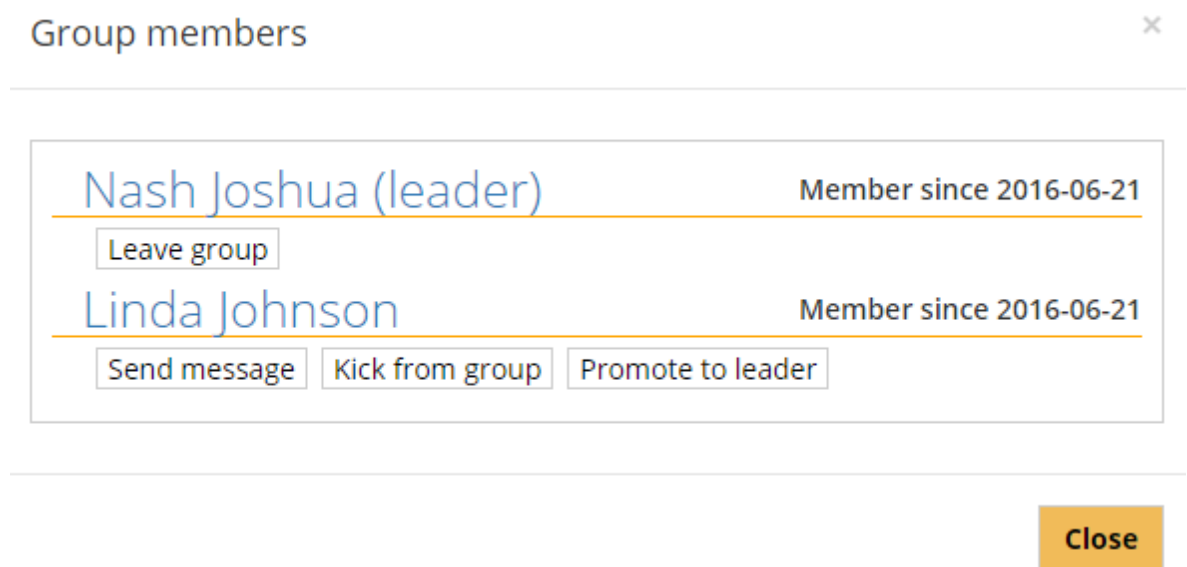


Figure 25 - A group member list from the point of view of the group leader

**„messaging/convertgroupprivate” (method: post)** – This API can also only be used by a group leader, switching the status of the group from public to private. It accepts a group id as input.

**„messaging/convertgrouppublic” (method: post)** – The inverse API to “messaging/convertgroupprivate”. The reasoning behind creating two APIs that do almost the same thing is that in the case of “messaging/convertgrouppublic”, any join requests are automatically accepted.

**„messaging/messagegroup” (method: post)** – Similar to “messaging/messageuser”, but with a group id instead of a user id.

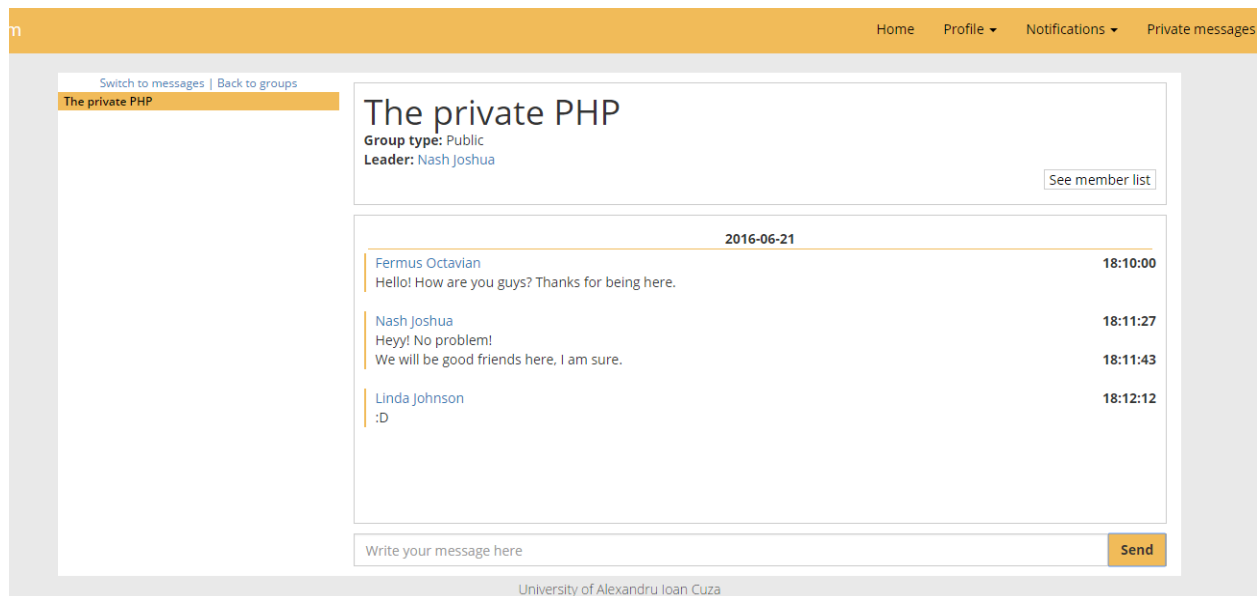


Figure 26 - Example of the group chat

## 2.2.7 Profile Page

The users of “Algorithm Compendium” will have their own personal profile page, where people can comment on or commend the user. It mixes together functionalities previously seen, such as commenting and replying, getting the list of algorithms and reporting content. Of course, the commendation feature is also in place. If the visited profile is owned by the logged account, the user can access the settings tab and change the basic account information. Since the functionalities of each API is similar to other APIs belonging to other controllers, I will just name

most of them and note the similarities to other APIs already presented, unless it's something not discussed before.

The controllers that implement the functionalities of the profile page are

- UsersController – Fetches necessary data and allows the possibility of commenting, replying or commending the profile's user.
- ReportController – As in all other pages, its main function, “reports/report” is used to send notifications to admins about unwanted content.

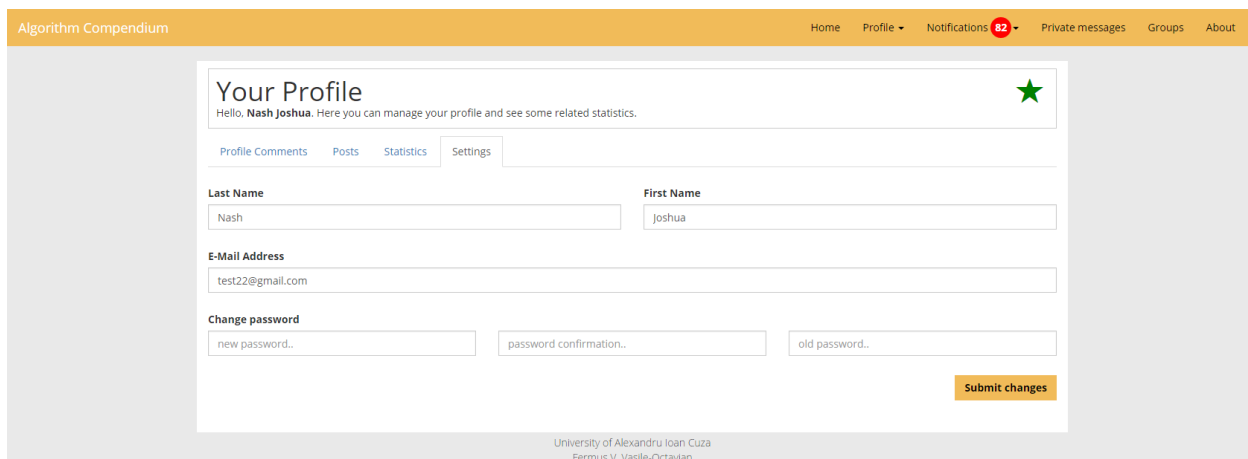
## APIs

**„users/commend” (method: put)**

**„users/profiledetails” (method: get)** – Fetches the data related to the profile, based on a valid user id. The data includes the user's post list, commendation number, comments, replies and likes linked to its comment section and some general statistics for the „statistics tab”.

**„users/commend” (method: put)** – As previously stated, this API is used to “commend” a user.

**„users/changeinformation” (method: post)** – This API is used to change the user info, that being first and last name, password and email. Of course, if the password is changed, password confirmation and old password inputs are needed, otherwise all three password fields are optional.



The screenshot shows a web application interface for a user profile. At the top, there's a navigation bar with links: Home, Profile, Notifications (with a red badge showing '82'), Private messages, Groups, and About. The main content area is titled 'Your Profile' and includes a greeting: 'Hello, Nash Joshua. Here you can manage your profile and see some related statistics.' Below this, there are tabs: Profile, Comments, Posts, Statistics, and Settings. The 'Settings' tab is selected. The settings form includes fields for 'Last Name' (filled with 'Nash'), 'First Name' (filled with 'Joshua'), and 'E-Mail Address' (filled with 'test22@gmail.com'). There is also a 'Change password' section with three input fields: 'new password..', 'password confirmation..', and 'old password..'. A 'Submit changes' button is located at the bottom right of the form. The footer of the page mentions 'University of Alexandru Ioan Cuza Ierusalim' and 'Fermus V. Vasile-Octavian'.

Figure 27 - An example of a profile page, on the settings tab

**„users/comment” (method: post)** – Similar to the other comment APIs, the difference being that the input received is a user id.

„users/votecomment” (method: post) – Similar to the other voting APIs, the difference being that the input received is a user id.

„users/reply” (method: post) – Similar to the other reply APIs, the difference being that the input received is a profile comment id.

„users/votereply” (method: put) – Similar to the other voting APIs, the difference being that the input received is a user id.

„users/deletereplay” (method: post) – Similar to other delete reply APIs, it receives a profile comment reply id.

„users/deletecomment” (method: post) – Similar to other delete comment APIs, receiving a profile comment id.

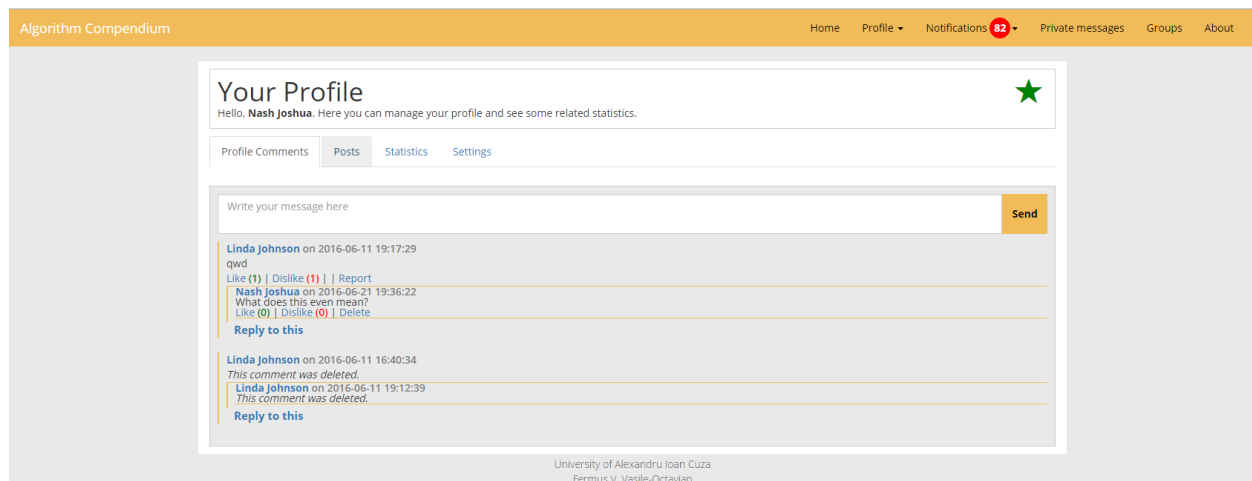


Figure 28 - An example of a profile page, on the comments tab

## 2.2.8 The Administrator page

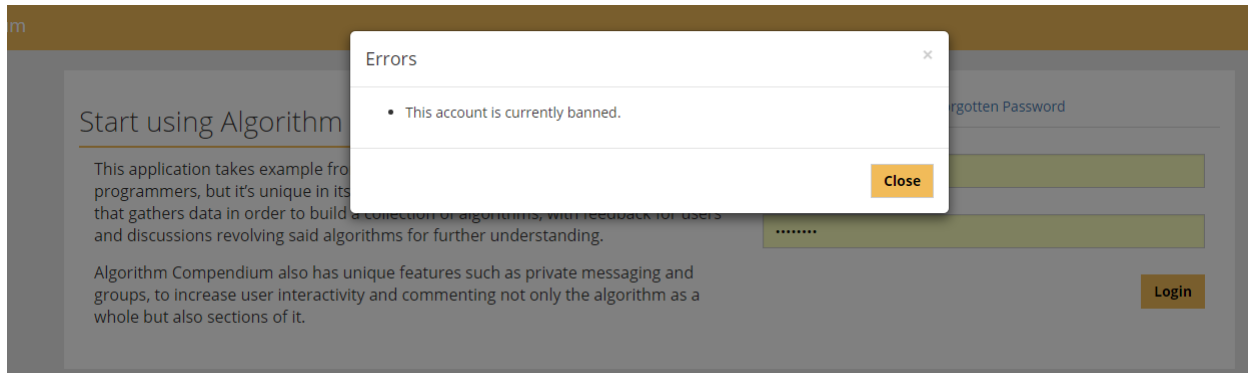
The users of the application are divided in four types:

- **Banned users** – Banned users are not allowed to call any APIs and cannot log in. Upon trying to log in the application, they are greeted with a notification modal stating that they cannot access the site. Any content previously created by the banned account is still public but can be removed by moderators/administrators.
- **Normal users** – These users can do anything on the website, aside from accessing the Admin page and use the moderating related APIs.

- **Moderators** – Moderators are users that have been promoted by the administrator of the website. Aside from the normal user privileges, a moderator can warn/ban normal users or promote them to moderator status.
- **Administrators** – This type of user has the highest status, having all the moderator privilege and being able to ban moderators.

## Warnings

Users and moderators can receive warnings as notifications, with references to the reported content. If a high enough amount of warnings is received, the moderator can choose to ban the user. A user can be banned even without having a single warning if the posted content is highly inappropriate.



*Figure 29 - The notification modal that tells the user that the account is unusable*

To help moderators and administrators, the administrator page has been created. It is visible for all the moderators and administrators and it contains three lists: “Unanswered reports”, “Answered reports” and “User list”.

The first list, “Unanswered reports” shows a list of reports sent by users for various reasons. The administrator on the page can see a representation of the problem, along a link to the content and the status of the users at that point and a set of actions the moderator can take, from warning one or both users to banning the users. After it is considered that the appropriate actions were taken, the report can be set as “answered”, being passed to the “Unanswered reports”, which is similar in design to the first tab, with the difference that it only contains answered reports and no actions can be taken against the users anymore, regarding those reports.

The third list is a list of current users, along with actions that can be taken against of them. Currently, the actions are “ban”, “promote” and “demote”.

The controllers that implements the functionalities of the page is AdminController.

## **APIs**

**„administrative/fetch” (method: get)** – This API takes nothing as an input but checks if the user has the required privilege (user type moderator or higher) and it returns a list for each tab in the page. The lists are presented above.

**„administrative/warn” (method: post)** – As input, it receives the ids of the of the reported person and the report id. It creates a warning and notifies the user about his misconduct, which he can see in his notification tab. The reported user will not know who the reporter is but will know who the moderator/administrator that warned him is.

**„administrative/banuser” (method: put)** – Receives the same input as “administrative/warn” but instead of creating a notification and adding a warning, it sets the user’s type to 0 (banned). If the user is logged he will be logged out at his next API request, notifying him of the ban on the landing page.

**„administrative/setanswered” (method: put)** – As input, this API receives a report id and updates it if it’s valid, setting it’s answered status to true and the administrator that answered the report as the calling user.

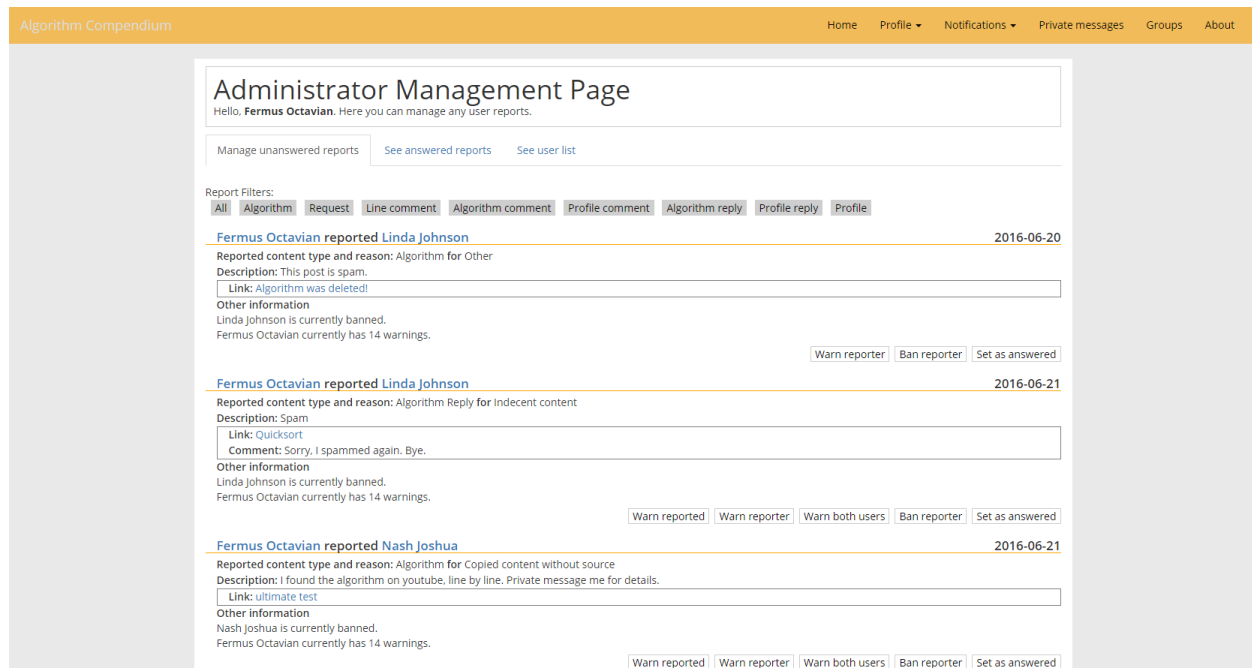


Figure 30 - The "Unanswered reports" tab of the Admin page

**„administrative/unbanuser” (method: post)** – Similar to ba user, but in this case the user type is set to 1 (normal), and a notification is sent to the user, telling him the name of the administrator that unbanned him.

**„administrative/demoteuser” (method: post)** – Receives an id as input, and it sets the type of the user to 1 (normal user) if it previously was 2 (moderator). Only administrators can use this function. A notification is sent to the demoted user.

**„administrative/promoteuser” (method: post)** – Similar to the demote API, but here, the current API promotes the user from user type 1 (normal user) to user type 2 (moderators) instead.

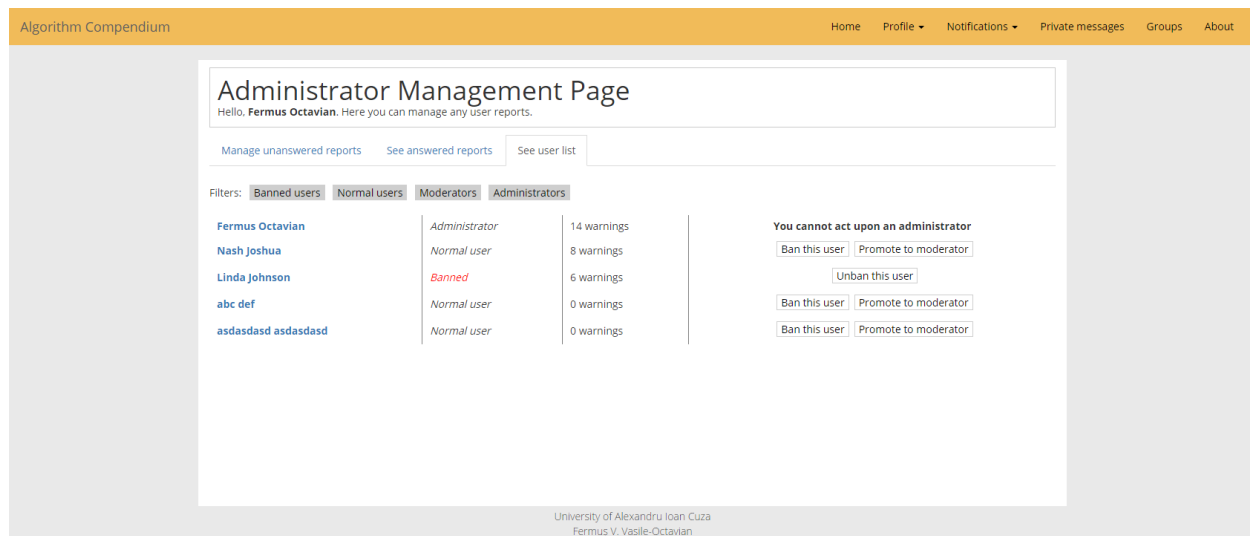


Figure 31 - The "User list" tab of the admin page

## 2.3 Graphical Interface Details

The graphical interface was created using HTML5 (HyperText Markup Language 5), and stylized using CSS3 (Cascading Style Sheets 3), using JavaScript along the jQuery framework to implement front-end implementations of the features and communicate with PHP via APIs. For some functions I used other JavaScript frameworks such as Ace Editor or jQuery-UI to create interface elements that would take considerable time to create and perfect otherwise, in order to work on any type of browser in a proper way. To make the application responsive I used the front-end framework called Bootstrap, mainly to profit of its "grid" feature.

Also, to template my pages I used the templating feature of the Laravel Framework.

I used grunt, along the libraries "grunt-contrib-sass" to convert SCSS files to CSS and "grunt-contrib-watch" to be able to convert the said files in real time, without calling grunt with each change.



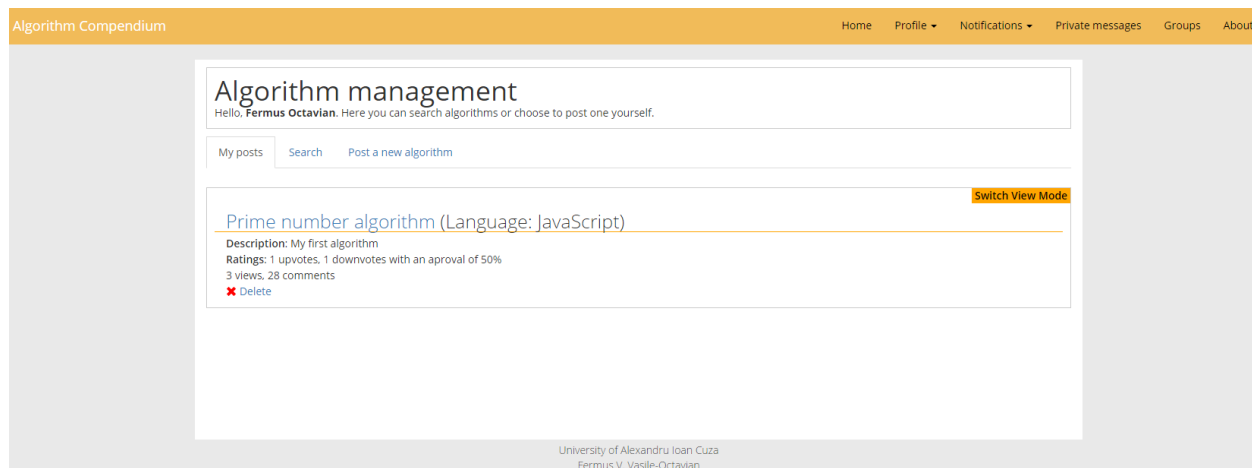


Figure 7 - Algorithm Compendium Graphical User Interface

## 2.4 Used Technologies – Front-End

### 2.4.1 HTML5 (HyperText Markup Language)

HyperText Markup Language, commonly abbreviated as HTML, is the standard markup language used to create web pages. Along with CSS, and JavaScript, HTML is a cornerstone technology used to create web pages.<sup>5</sup>

HTML5 is just the next version of HTML. The HTML5 specification was finalized in 2013 by the W3C. For the previous few years, the specification had been in flux, still being worked on by the W3C working groups. Browser manufactures have chosen to implement some of the specifications, but some high-profile companies are backing off the technology (Facebook for example). Coupling this with the specification previously not finalized, adoption has been slow.<sup>6</sup>

<sup>5</sup> "JavaScript - The definitive guide" by Flanagan, David (page 1)

<sup>6</sup> "Modern Web Essentials using JavaScript and HTML5" by David Pitt (page 24)

### 2.4.2 CSS3 (Cascading Style Sheets 3)

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language.<sup>7</sup> Although most often used to set the visual style of web pages and user interfaces written in HTML and XHTML, the language can be applied to any XML document, including plain XML, SVG and XUL.<sup>8</sup>

CSS3 is the latest evolution of the Cascading Style Sheets language and aims at extending CSS2.1. It brings a lot of long-awaited novelties, like rounded corners, shadows, gradients, transitions or animations, as well as new layouts like multi-columns, flexible box or grid layouts.<sup>9</sup>

### 2.4.3 JavaScript

JavaScript is a cross-platform, object-oriented scripting language. It is a small and lightweight language. Inside a host environment (for example, a web browser), JavaScript can be connected to the objects of its environment to provide programmatic control over them. JavaScript and Java are similar in some ways but fundamentally different in some others. The JavaScript language resembles Java but does not have Java's static typing and strong type checking. JavaScript follows most Java expression syntax, naming conventions and basic control-flow constructs which was the reason why it was renamed from LiveScript to JavaScript.<sup>10</sup>

### 2.4.4 jQuery Library

jQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers.<sup>11</sup>

---

<sup>7</sup> "CSS developer guide" by Mozilla Developer Network

<sup>8</sup> "Web-based Mobile Apps of the Future Using HTML 5, CSS and JavaScript" by HTMLGoodies

<sup>9</sup> "CSS developer guide" by Mozilla Developer Network

<sup>10</sup> "JavaScript developer guide" by Mozilla Developer Network

<sup>11</sup> <https://jquery.com/>

This library, in my opinion is a must use library in JavaScript programming, due to the multitude of features that it holds. Aside from basic event scripting, I used this library to create Ajax requests and manipulate data, making the connection with the Laravel's controllers and APIs.

### 2.4.5 Bootstrap Framework

Bootstrap is a front-end framework designed to kick-start development of Web apps and sites. For questions related to a version of Bootstrap also use the specific version's tag from "twitter-bootstrap-2", "twitter-bootstrap-3" and "twitter-bootstrap-4" tags.

Bootstrap is a front-end framework designed to kickstart the front-end development of webapps and sites. Among other things, it includes base CSS and HTML for typography, icons, forms, buttons, tables, layout grids, and navigation, along with custom-built `jQuery-plugins` and support for responsive layouts.<sup>12</sup>

I used bootstrap mainly for the grid and navbar, allowing me to create html much easier, although most of the other CSS was built by me.

### 2.4.6 Grunt

Grunt is a task runner that automates repetitive tasks. I used it to compile SCSS files in CSS without repetitively using Ruby to compile when needed and the portability it presents once NPM has been installed to the device.

## 2.5 Used Technologies – Back-End

### 2.5.1 PHP5

PHP (recursive acronym for *PHP: Hypertext Preprocessor*) is a widely-used open source general-purpose scripting language that is especially suited for web development and can be embedded into HTML.

What distinguishes PHP from something like client-side JavaScript is that the code is executed on the server, generating HTML which is then sent to the client. The client would receive the results of running that script, but would not know what the underlying code was. A programmer

---

<sup>12</sup> <http://stackoverflow.com/tags/twitter-bootstrap/info>

can even configure his web server to process all your HTML files with PHP, and then there's really no way that users can tell how he did his implementation.<sup>13</sup>

In the implementation of “Algorithm Compendium” I used a minimal amount of PHP in my HTML files, although the files had the extension “.blade.php” due to the use of templating via the Laravel framework.

## 2.5.2 MySQL

MySQL, the most popular Open Source SQL database management system, is developed, distributed, and supported by Oracle Corporation.

The MySQL Web site (<http://www.mysql.com/>) provides the latest information about MySQL software.

MySQL is a database management system.

A database is a structured collection of data. It may be anything from a simple shopping list to a picture gallery or the vast amounts of information in a corporate network. To add, access, and process data stored in a computer database, you need a database management system such as MySQL Server. Since computers are very good at handling large amounts of data, database management systems play a central role in computing, as standalone utilities, or as parts of other applications.<sup>14</sup>

I used MySQL to store all the data related to the application, creating queries using Laravel.

## 2.5.3 Laravel 4.2 Framework

Laravel is a web application framework with expressive, elegant syntax. Laravel aims to make the development process a pleasing one for the developer without sacrificing application functionality. Laravel is accessible, yet powerful, providing powerful tools needed for large, robust applications.<sup>15</sup>

---

<sup>13</sup> <http://php.net/manual/en/intro-what-is.php>

<sup>14</sup> <https://dev.mysql.com/doc/refman/5.7/en/what-is-mysql.html>

<sup>15</sup> <https://laravel.com/docs/4.2/introduction>

Even if version 5 has recently surfaced, I used 4.2 because it was safe to assume that the probability of a bug of the framework that could impede the progress of the implementation could exist.

The reason I used Laravel 4.2 over Symfony or Silex is my slight familiarity of the framework and the ease I could create controllers, database queries and page templates because of it.

# Chapter III – Conclusions

The created web application aims to ease the life of programmers and IT students by offering them a platform where they can find or request needed resources related to programming.

They are offered the possibility of building a reputation on the website as developers or learning from talented persons via posts, groups and private messaging.

Even if there are hundreds of websites that target learning and improving individuals as developers, none implement all my ideas in one application. I took my inspiration from web applications (not all related to IT and programming) such as YouTube (the post and comment system), Soundcloud (commenting content at certain lines of code) and Stack Overflow (the ask and answer system).

I used a wide variety of technology to build the thesis application, although in Web Development, an application usually comprises a large amount of used technologies by default.

The main difficulty I had with the application was creating the Back End, since my experience in that field was not very great, although I did pick this thesis to improve my knowledge.

As ways of improvement, there could be added more ways to give incentive to users to post algorithms and be a helping hand in the community.

Making the site registration free could be a way to improve the interactivity with the application, but I fear that it would make the user interactivity decrease without the possibility of commenting, replying and requesting algorithms. Of course, the possibility of allowing anonymous users to comment, post algorithms or request them is possible but it is not viable due to possible persons with bad intents.

I strongly believe that I improved as a programmer myself by creating my dissertation thesis and I would like to thank Asist. Dr. Vasile Alaiba for his invaluable help in coordinating me and Conf. Dr. Sabin-Corneliu Buraga for making me take interest in Web Development.

As a closing remark, I think that this application could succeed in its purpose and be used as a real website, maybe improving upon it and using it as a hub for the students of “Alexandru Ioan Cuza” University.

# Bibliography

- [1] **Stack Overflow** official site, “Tour”, <http://stackoverflow.com/tour>
- [2] **W3Schools** official site, “About” - <http://www.w3schools.com/about/default.asp>
- [3] **Code School** official site, “About” - <https://www.codeschool.com/about>
- [4] **Wikipedia**, “Web Application” - [https://en.wikipedia.org/wiki/Web\\_application](https://en.wikipedia.org/wiki/Web_application)
- [5] **Flanagan David**, "JavaScript - The definitive guide"
- [6] **David Pitt**, "Modern Web Essentials using JavaScript and HTML5"
- [9] **Mozilla Developer Network**, “CSS developer guide” – <https://developer.mozilla.org/en-US/docs/Web/Guide/CSS>
- [10] **Scott Clark**, “Web-based Mobile Apps of the Future Using HTML 5, CSS and JavaScript” - <http://www.htmlgoodies.com/beyond/article.php/3893911/Web-based-Mobile-Apps-of-the-Future-Using-HTML-5-CSS-and-JavaScript.htm>
- [11] **Mozilla Developer Network**, “JavaScript developer guide” - <https://www.developer.mozilla.org/en-US/docs/Web/JavaScript/Guide>
- [12] **JQuery** official site - <https://jquery.com/>
- [13] Daniel Tan, “About twitter-bootstrap” - <http://stackoverflow.com/tags/twitter-bootstrap/info>
- [14] **PHP** official site, “What is PHP?”, <http://php.net/manual/en/intro-what-is.php>
- [15] **MySQL official site**, “What is MySQL” - <https://dev.mysql.com/doc/refman/5.7/en/what-is-mysql.html>
- [16] **Laravel** official site, “Introduction”, <https://laravel.com/docs/4.2/introduction>