Table of Contents

[1. Introduction 2](#_Toc507874602)

[2. Motivation 2](#_Toc507874603)

[3. Basic functionalities, user categories and terms used 3](#_Toc507874604)

[4. Details regarding User categories 4](#_Toc507874605)

[4.1. Guests 4](#_Toc507874606)

[4.2. Registered Users 4](#_Toc507874607)

[4.3. Administrators 4](#_Toc507874608)

[4.4. Possible extensions 4](#_Toc507874609)

[5. Details regarding problems 5](#_Toc507874610)

[5.1. Test cases 5](#_Toc507874611)

[5.2. Points awarded 5](#_Toc507874612)

[5.3. Problem solution 5](#_Toc507874613)

[5.4. Possible extensions 5](#_Toc507874614)

[6. Functionalities 6](#_Toc507874615)

[6.0. Basic functionalities 6](#_Toc507874616)

[6.1. Guest functionalities 6](#_Toc507874617)

[6.2. Registered user functionalities 6](#_Toc507874618)

[6.3. Administrator functionalities 7](#_Toc507874619)

[6.4. Look at the future 8](#_Toc507874620)

[7. Implementation overview 9](#_Toc507874621)

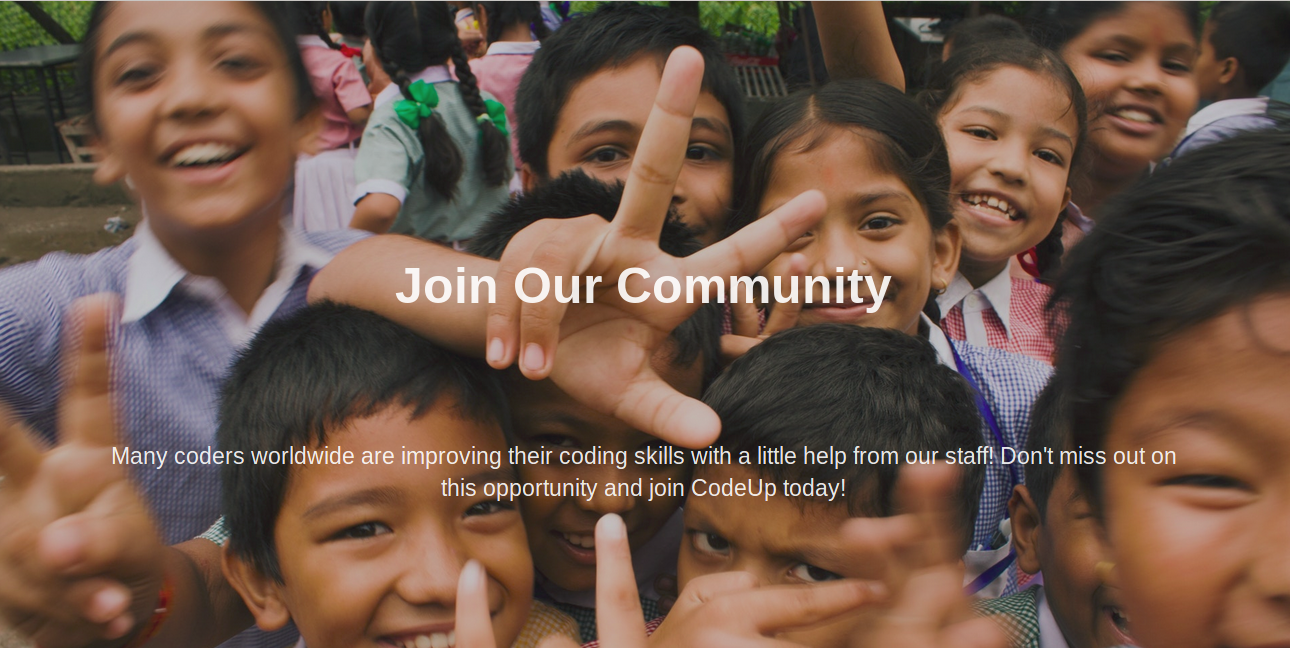
[8. Constraints and concerns 9](#_Toc507874622)

# 1. Introduction

Our project should be small, but a fully functional web platform for solving programming challenges and learning new data structures / algorithms. This platform will, at first, only have core functionalities, but we will be paying extra attention to modular design throughout the platform implementation because the number of possible functionality extensions is incredibly large.

# 2. Motivation

Inspired by the ones such as HackerRank and Project Euler, we will try to build a platform that is easy-to-use and yet has a vast amount of problem categories that could be used by our users to advance their knowledge in the specified domain. We hope that coding enthusiasts throughout the world will love and use our platform.



# 3. Basic functionalities, user categories and terms used

In the following section, we will bold every term that is important for further understanding of the platform functionalities.

CodeUp gives **User**s a registration option, which is needed if users want to display basic information about them (**User Profile**), submit problem solutions and compete with others based on their **total score**.

Users without an account (**Guest**s) can see problem statements, but all other platform functionalities are disabled for them.

**Administrator** can make other users administrators, and add/modify:

1) **Problem statement**s

2) **Problem solution**s

3) **Test cases**

**Problem domain**s are used to group problems of similar type (i.e. Algorithms, Data Structures, Math problems, etc.). Domains are further divided into **problem sub-domains**

which are used for finer granulation of problem domains (i.e. Algorithms can be further divided into Divide & Conquer, Dynamic programming, etc.).

Adding problem domains and sub-domains is only limited by time and the collective knowledge of the platform administrators.

Users can find and choose one of the problems from problem domain they are most interested in, and type their solution into the online **editor** (which again, can support multiple languages – only limited by the amount of time implementers have to provide additional functionalities). The solution, in the end, should be graded based on the amount of test cases it successfully passes; still, it is hardly realistic that the two of us will provide problems with many test cases at the beginning of this project’s development.

# 4. Details regarding User categories

## 4.1. Guests

Only have access to web-pages with problem statements. When they open the platform’s Web Page, they are asked if they want to register or log in. Further explanation on why one would want to register is presented to the user.

## 4.2. Registered Users

Each user has their own profile, which can be customized to their own liking. The profile provides information such as an username, country of origin, total points earned solving problems so far, and possibly rank, both in user’s own country and globally. Registered users will have an email address stored in the database, which can be used for sending notifications regarding new site functionalities, programming competitions, new problem domains and so forth.

## 4.3. Administrators

Administrators would have every functionality one might imagine. As stated before, they would add problem (sub) domains, individual problems and corresponding test cases, edit existing site resources and possibly ban users who show malicious intent (i.e. sharing problem solutions on public message boards or trying to sell problem solutions to other users).

## 4.4. Possible extensions

Need for a new user category, **recruiter**, may arise in the future. A recruiter might sort registered users by some criteria and then contact ones which are to his liking.

Registered Users could form **team**s and compete with other teams in a **tournament**-like fashion.

Users may be awarded with **badge**s that correspond to reached milestones.

# 5. Details regarding problems

## 5.1. Test cases

Each problem must have a corresponding set of test cases which are to be compared with the user’s program output. Set of test cases need to be provided at the same time when a new problem is posted, but may be extended to encompass more of them in the future. There needs to be a small set of public test cases which enable users to have a better understanding of the problem. Other test cases are private, and therefore not visible to the users. Extra care has to be exercised in order to make sure that test cases really do produce a good output.

## 5.2. Points awarded

Each problem, when solved correctly, awards users with some variable number of points, depending on the perceived difficulty of the problem. Problems themselves belong to some level of difficulty, e.g. easy, medium, hard, expert.

Users may only get a partial number of points for some problem, if their solution produces valid output to a subset of test cases. Total number of user points is used for ranking users by their expertise.

## 5.3. Problem solution

Platform’s problem solution is provided to the user once their solution passes all the test cases successfully. Solution explanation and/or animations might be provided.

## 5.4. Possible extensions

Adding some sort of virtual currency, this can be both purchased and earned through solving problems. Currency can be used for peeking at some of the private test cases, or viewing platform’s solution before one solved the problem (passed all of the test cases).

Adding support for additional languages is, as stated before, possible as time passes.

# 6. Functionalities

## 6.0. Basic functionalities

*6.0.1 Registering*

Guests will be provided with the option of registering, so that they can access the set of site’s functionalities corresponding to the registered user functionalities. E-mail, username and password are required for creating an account.

*6.0.2 Logging in*

Using username and password, users can log in and resume where they left off. After logging in, they get to access the site’s functionalities corresponding to the account type.

*6.0.3. Browsing problem statements*

Users/Guests access problems by navigating through a hierarchical structure of problem domains and sub-domains.

## 6.1. Guest functionalities

Guests have access only to the basic site functionalities

## 6.2. Registered user functionalities

In addition to basic functionalities, they also have access to the following:

*6.2.1. Browsing other users*

Users can browse through the list of all users, view their profiles, including their own.

*6.2.2. Submitting problem solutions*

Upon solving a problem, users can submit their own solutions. The solution is submitted by typing the program in the editor window and, upon finishing program construction, sending that program over the internet to one of our servers where the validity of the program will be tested against all public and private test cases.

*6.2.3. Modifying personal information*

Users can make changes to their personal profile (e.g. username, password, email address, country etc.) and can modify the visibility of their personal information to other users.

*6.2.4 Modifying personal preferences*

Users should be able to sign up for notifications about tournaments, new problem statements for some problem (sub)domain and so on. Users can also cancel their subscription at any time.

*6.2.5. Requesting a new feature*

We are trying to be as user-friendly as possible, and therefore providing a mechanism for user feedback is one of our concerns. Registered users may request anything that they think will increase platform’s usefulness, e.g. a larger base of problems or even an introduction of novel concepts to the site.

*6.2.6. Reporting a problem*

Each page has a reference to ’Report a problem’ page where users can notify us about any inconveniences while browsing our site.

## 6.3. Administrator functionalities

In addition to basic and registered user functionalities, administrators also have access to the following:

*6.3.1. Changing account type*

Administrators can modify other non-administrator account types, either by upgrading them (e.g. from a registered user to an admin) or by downgrading them (e.g. from user to banned user).

*6.3.2. Adding new content to the site*

Administrators can add new problem (sub)domains, new individual problem statements or new test cases to already existing problems.

*6.3.3 Browsing user complaints/suggestions*

Users can request features from our staff, or report a problem with the way our site is functioning. Administrator is the first one who should look at those requests, remove spam/inappropriate requests (this could possibly be automated in the future) and forward any quality requests that were in the request pool to the developers team.

## 6.4. Look at the future

While the amount of possible modifications to the set of functionalities is endless, we will try to list some of the most exciting ones. It is never out of the question that our platform’s core functionality set will change according to what users find most useful – such is the life of software developers.

*6.4.1. Adding new user account types, for example: recruiters, senior problem solvers, etc...*

*6.4.2. Each problem has a forum for discussion, where users might provide ideas about the problem or its solution*

*6.4.3. Direct messaging between users*

*6.4.4. Hackathons, Tournaments and other user-to-user or group-to-group competitions*

*6.4.5. Exotic problem domains:* Adding a problem domain such as distributed programming would require changing a good part of our infrastructure, for example routing programs to multiple servers, which would then communicate using some not-yet-implemented protocol to arrive at the solution.

*6.4.6. Tutorials, explanations of core concepts/algorithms using animations*

*6.4.7. Customizing editor’s look:* Users might be used to certain type of IDE’s – and we will try to enable them in choosing the type of editor they would use if they were programming on their own computers. That may include changes to font type and size, tabulation size, and one of the existing editor styles (e.g. Dracula, Monokai, etc.).

# 7. Implementation overview

For platform implementation, the following will be used:

1. MySQL for storing, retrieving and deleting data in a database,

2. HTML/CSS/JavaScript for the front-end design,

3. PHP, popular language the server-side scripting.

We might consider using NoSQL for manipulating data in a database.

Our team will possibly use some third-party software, for features that would be difficult to implement from scratch.

# 8. Constraints and concerns

Because this will be by far the most complex project that we have ever worked on, the list of concerns is pretty big:

* How to provide an online editor (one possibility is using third-party party software) ?
* How are we going to compile user’s code, especially in a way that will enable us to add new language support without modifying server-side by a substantial amount ?
* Can we do a good enough job of error reporting (compilation-time errors and their details, run-time error and their details) ?
* Users might provide us with a malicious code (infinite loops, code that might attempt to destroy our database records, etc.). How are we going to make sure that that code will never run on our machines, or is terminated before causing some damage ?

As far as constraints go, it is hard to reason about them upfront. We have never implemented anything similar and therefore we can’t really reason about trade-offs of our potential solution.

One obvious constraint is a small time window until due-date of submission. There are multiple technologies that we have to master in the meantime, and probably thousands of lines of code we will have to write, along with proper testing and documentation. How we are going to do all of that with our 2-man team, remains to be seen.

***9. Version history***

|  |  |  |
| --- | --- | --- |
| **Version** | **Date** | **Changes** |
| **1.0** | 03.03.2018. | Initial version |
| **1.1** | 22.03.2018 | Revisited “basic functionalities”. Removed “customize editor’s look” from basic to possible future extensions. Added “Browsing user’s suggestions” to admin’s basic functionalities. Decided that server side technology is Python’s framework Flask. Formatting. |
| **1.2** | 21.05.2018 | After the initial phase of implementation, we have revised this document in order for it to reflect our experience with this project. In the end, we used PHP (no frameworks) as the implementation language. Plans to reimplement project in Laravel (Popular PHP Framework). |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |