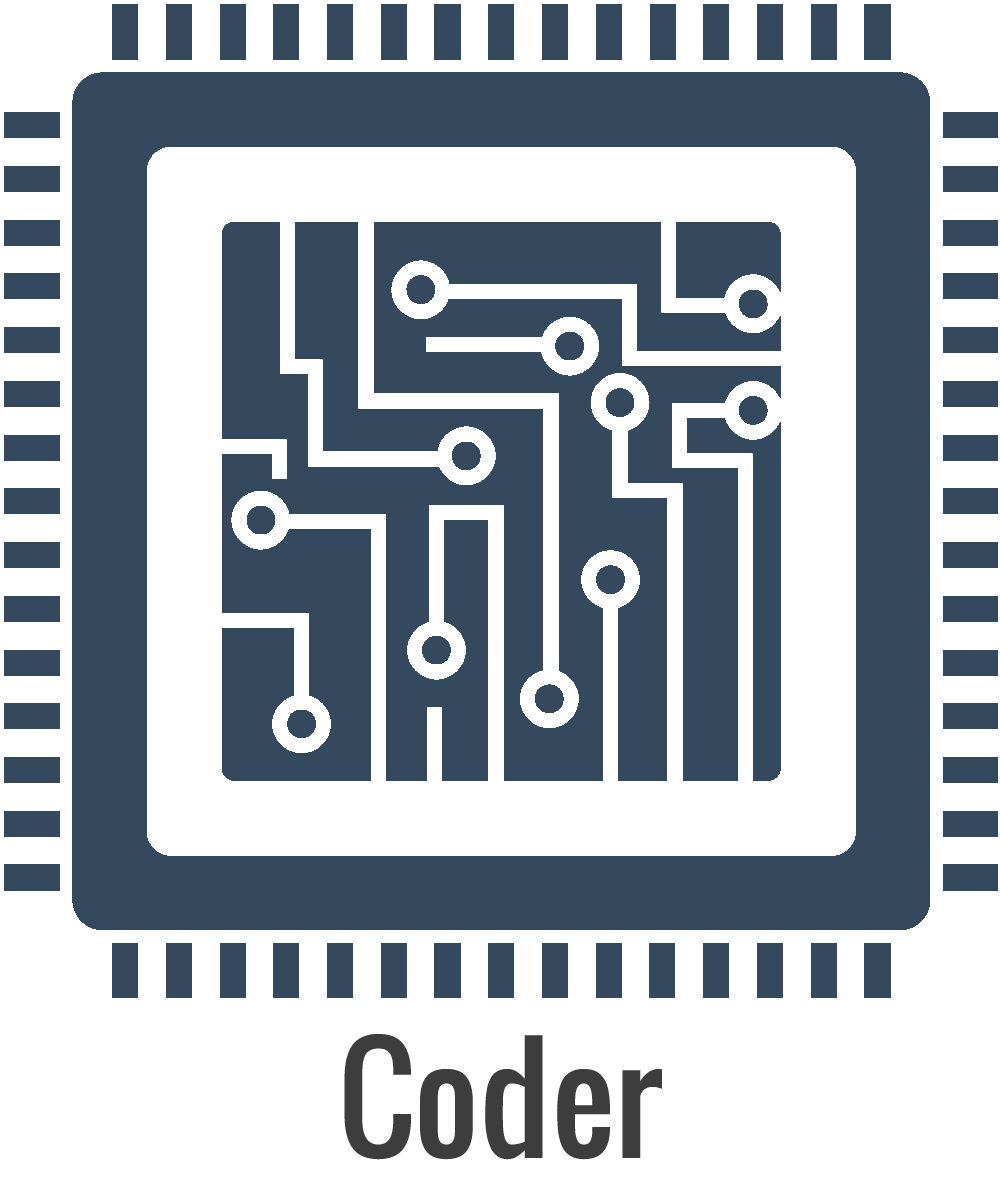
Háskólinn í Reykjavík, 29. apríl ‘16

Daníel Brandur Sigurgeirsson

VLN2 - Vorönn 2016



Requirement analysis report

**Group 7**

Andri Andrésson

Gunnar Torfi Steinarsson

Orri Arnarsson

Steinar Freyr Kristinsson

Þorgrímur Jónasarason

Table of Contents

[Introduction 3](#_Toc449605245)

[User groups 4](#_Toc449605246)

[Interviews 5](#_Toc449605247)

[Use cases 6](#_Toc449605248)

[Requirement list 10](#_Toc449605249)

[Usability Goals 12](#_Toc449605250)

[Rules 12](#_Toc449605251)

[Final words and conclusion 14](#_Toc449605252)

[Appendix 15](#_Toc449605253)

# Introduction

This requirement analysis report will look into user groups, use cases, requirements, interviews and usability goals in a new web application tool called Coder.

Coder is supposed to be a better and more user friendly version of Mooshak, which is the current programming assignment checker. The general idea is to offer students studying programming instant feedback when solving programming problems posed by their teachers. The teacher defines an assignment with a given input and output and the students can submit their solutions and see clearly if their solution is solving the problem correctly.

Coder should be very easy to use and should display user and course statistics clearly. The main difference from the current Mooshak is that Coder will not be as contest minded as its precursor. Mooshak does a lot of things very well, but there is always room for improvement. The goal of Coder is to be the next generation of Mooshak.

# User groups

The system shall assume that there are 3 types of users: Students, Teachers and Admins.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **User Group** | **Background** | **System usage** | **Environment** | **Goals** |
| 1 | **Students**  **Importance:** Very high | **Age:** 18+  **Gender:** Both male/female  **Education:** Varies.  **Abilities/disabilities:** None  **Computer knowledge:** Above average | **Usage:** Used around once a week. Up to two hours per visit.  **Training:** Moderate.  **Attitude:** Excited.  **Est. number of users:** <1000. | **Technical environment:** Computer.  **Real environment:** Home/RU.  **Other environment:** None. | -Submitting assignments  -Get feedback |
| 2 | **Teachers**  **Importance:** High | **Age:** 18+  **Gender:** Both male/female  **Education:** Varies.  **Abilities/disabilities:** None  **Computer knowledge:** Good. | **Usage:** Average three times per week.  **Training:** Good.  **Attitude:** Excited.  **Est. number of user:** <100 | **Technical environment:** Computer.  **Real environment:** Home/RU.  **Other environment:** None. | - Creating assignment.  - Review student assignments. |
| 3 | **Admins**  **Importance:** High | **Age:** 18+  **Gender:** Both male/female  **Education:** Varies.  **Abilities/disabilities:** None  **Computer knowledge:** Very good. | **Usage:** Every few weeks.  **Training:** Very good.  **Attitude:** Thrilled.  **Est. number of users:** <10 | **Technical environment:** Computer.  **Real environment:** Home/RU.  **Other environment:** None. | -Add new users to the system.  -Reset passwords and general error handling. |

# Interviews

After speaking with a couple of students at RU and asking them how they feel about the current mooshak the conclusion is that although the current version does what it is supposed to. They think the system can be improved quite a lot visually as well as functionally. Most of the interviewees asked for an upgraded UI and said that the error messages could be clearer. One student specifically asked for a moving melon background, the group gave no promises about that request.

The teachers feel that the system could be more efficient and that some more statistics on each assignment would be useful. One TA (*teacher assistant*) asked for a better way to review valgrind errors so that students would ask him less about the meaning of their error messages.

star

# Use cases

|  |  |
| --- | --- |
| Name | **Creation of users and courses, link users to courses** |
| Number | 1 |
| Priority | A |
| Precondition | Admin is logged into the system. |
| Description | An Administrator runs the web application. (S)he should be able to create users and define their role, as well as create courses and link users to those courses. |
| Basic flow | Admin logs into the system and creates a course.Admin creates a new user and defines its role. Admin then links the user to a selected course. |
| Alternative flow | Username or course name has already been used. Admin is asked to find another name that is currently not in use. |
| Postcondition | Users and courses have been added to the database, links have been registered. |
| Requirements | 1, 2, 3, 4, 5, 6, 7, 24, 31 |
| Actor | Admin |

|  |  |
| --- | --- |
| Name | **Creation of assignments** |
| Number | 2 |
| Priority | A |
| Precondition | Teacher and course are linked in the database. |
| Description | Teachers must be able to create assignments within a course. Each assignment should optionally contain multiple parts, where each part is X% of the grade for the assignment. Teacher saves the assignment. |
| Basic flow | Teacher logs in and goes to a specific course. Teacher creates a new assignment and optionally creates multiple parts (subtasks) inside the assignment. Teacher posts assignment to system. |
| Alternative flow | - |
| Postcondition | Assignment has been created. |
| Requirements | 1, 2, 8, 9, 10, 22, 23, 31 |
| Actor | Teachers |

|  |  |
| --- | --- |
| Name | **Inputs and expected outputs for assignments** |
| Number | 3 |
| Priority | A |
| Precondition | Course and assignment are linked. |
| Description | Each assignment part defines a program which should be written (example: a program which adds two numbers). For any program, it should be possible to define input/output pairs, where a correct program implementation will give the specified output for the given input. Example: a program which should add two numbers could have two such pairs defined, the first has the input 5 and 4, output 9, and the second has the input 4 and -2, output 2. There should be no practical limits to the number of input/output pairs associated with a given part. |
| Alternative flow | If there is no output given, the teacher has to write an expected output. |
| Basic flow | Teacher logs into the system. The teacher selects a project and then a task within that project. In the task, the teacher types in as many input/output pairs needed and saves the task. |
| Postcondition | Assignment gets updated with inputs/outputs. |
| Requirements | 1, 2, 11, 12, 20, 27, 32 |
| Actor | Teachers. |

|  |  |
| --- | --- |
| Name | **Hand in and submission** |
| Number | 4 |
| Priority | A |
| Precondition | Student and course are linked in the database. |
| Description | For each part, a student should be able to hand in his/her solution to that part. There should usually be no restrictions to the number of submissions allowed from a given student for each part (however, see below), and the system should keep a record of all the submissions, including if it was successful or not, such that students and/or teachers should be able to view this history. |
| Basic flow | Student logs into the system. The student then goes to a course page. Student picks the assignment he wants to submit. Student picks a subtask (if applicable). Student chooses file to submit from his local machine. Finally the student submits a file to the server. |
| Alternative flow | Student has not submitted any solutions to the assignment. Student will not have any history to display.  Many factors could play part in the student not being able to upload his solution. User is asked to re-upload. |
| Postcondition | Assignment gets submitted. |
| Requirements | 1, 2, 13, 14, 15, 16, 19, 26, 28, 29, 33, 34 |
| Actor | Students |

|  |  |
| --- | --- |
| Name | **Display students’ solutions and grades.** |
| Number | 5 |
| Priority | A |
| Precondition | An assignment has been created in the course. |
| Description | The teacher should be able to view the results, such that they can be moved someplace else (such as MySchool or to some other LMS system). When listing the results for a given student, the best result for each assignment part should be listed. I.e. if the student handed in 3 solutions to a given part, and the second solution turned out to be the one which gave the most correct answers. |
| Basic flow | Teacher is logged in and looks for an specific assignment. He finds a result for a specific student. The student submitted three results. The second one is the best one (according to the system tests). The teacher selects that one. The teacher reviews that results and gives the student a grade for it. |
| Alternative flow | A student has not submitted a solution to an assignment, a default grade of 0 is given. |
| Postcondition | Every student in the course can see his result and solution for the assignment. |
| Requirements | 1, 2, 17, 18, 29 |
| Actor | Teachers, students. |

# Requirement list

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Name and description** | **Use Case num.** | **Priority** | **Status** |
| 1 | Users can log into the system | 1, 2, 3, 4, 5 | A |  |
| 2 | Users can log out of the system | 1, 2, 3, 4, 5 | A |  |
| 3 | Admin is able to create users | 1 | A |  |
| 4 | Admin can define roles of users | 1 | A |  |
| 5 | Admin should be able to create courses | 1 | A |  |
| 6 | Admin should be able to link users to courses and vice versa | 1 | A |  |
| 7 | Admin should be able to link courses to users | 1 | A |  |
| 8 | Teacher can create an assignment within a course | 2 | A |  |
| 9 | Teacher can define parts inside of the assignment | 2 | A |  |
| 10 | Teacher can give grades for each part in the assignment | 2 | A |  |
| 11 | Teacher can add assignment description for each part | 3 | A |  |
| 12 | Teachers can assign input/output pairs to test each assignment solution, as many pairs a desired | 3 | A |  |
| 13 | Students can turn in each assignment as many times as desired | 4 | A |  |
| 14 | The system keeps a record of all submitted solutions to each assignment | 4 | A |  |
| 15 | Teachers can view the system records as desired | 4 | A |  |
| 16 | Students should be able to review their own records | 4 | A |  |
| 17 | Teacher should be able to post results to an LMS | 5 | A |  |
| 18 | When displaying results, the best part of each assignment should be listed | 5 | A |  |
| 19 | Link users together in assignment | 4 | A |  |
| 20 | Teacher can add overall assignment description | 3 | A |  |
| 21 | Be able to view assignment partners solution | 4 | A |  |
| 22 | A teacher and/or students could define student groups, such that submissions from all group members are visible to other members. | 2 | A |  |
| 23 | A teacher could define a limit to the number of submissions accepted for a given assignment part, or a penalty for a number of submissions above a certain limit. | 2 | B |  |
| 24 | Auto email user info | 1 | B |  |
| 25 | Multiple programming languages | - | B |  |
| 26 | Text editor & file upload | 4 | B |  |
| 27 | Teacher can attach files into assignment description | 3 | B |  |
| 28 | Duplicate submissions not allowed | 4 | B |  |
| 29 | Solution statistics for each assignment (both for students and teachers) | 4, 5 | B |  |
| 30 | Users can comment on each task | - | B |  |
| 31 | Users can copy the input given just by clicking a button | - | B |  |
| 32 | Multilingual web application | - | C |  |
| 33 | Facebook login, with access code | 1, 2 | C |  |
| 34 | Markdown editor for description | 3 | C |  |
| 35 | The ability to debug programs inside the application | 4 | C |  |
| 36 | The ability for students to edit the code online simultaneously | 4 | C |  |
| 37 | The ability for students to invite their teacher to their code | - | C |  |
| 38 | The ability to link this solution to the Centris LMS API | - | C |  |
| 39 | The ability to change the theme color of the system | - | C |  |
| 40 | Ability to show only “Accepted” submittions in each assignment. | - | C |  |
| 41 | Activity/Notifications feed | - | C |  |
| 42 | Students can broadcast live stream to other students/teachers | - | C |  |
| 43 | All users can view other live streams and old broadcasts | - | C |  |

# 

# Usability Goals

So far no tests have been done on the old system or the new one (because the new one currently does not exist). Nonetheless we created usability goals to use as guidelines.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Usability Factor** | **Data Collected** | **Worst Case** | **Preferred case** | **Best Case** | **Value Now** |
| Efficiency | Average response time of submission. | NA | <10 sec. | NA | NA |
| Efficiency | Click count to complete submission of an assignment. | >10 | 7 | 6 | NA |
| Effectiveness | Is able to submit an assignment (rate%) | 0% | 100% | 100% | NA |
| Satisfaction | Average satisfaction rating. | 0% | 80% | 100% | NA |

# Rules

We thought of a couple of rules that we believe will make Coder more natural to use, they will be listed here:

* You can only create 1 instance of each course. (Unique course name)
* Each student can only receive one grade for each assignment.
* Students can only upload certain types of files. (e.g .zip/.py)
* A student can only review his/his group’s solution.
* Students must submit solution in the programming language given.
* 100 is normally the highest grade, but teachers can give up to 120 points for each assignment.

# Final words and conclusion

There are obviously a lot of things that can be done to improve the current system. Mooshak does what it needs to do, but not much more. We believe we have the data, ideas and manpower to make Coder quite a lot better than Mooshak in many aspects.

Keep in mind that nothing in this report is final and can/will be changed throughout the implementation of Coder.  
  
Having said that, we have described *what* can be done and next up for us is to write a design report that consists of *how* the things mentioned in this report will be implemented.

# 

# 

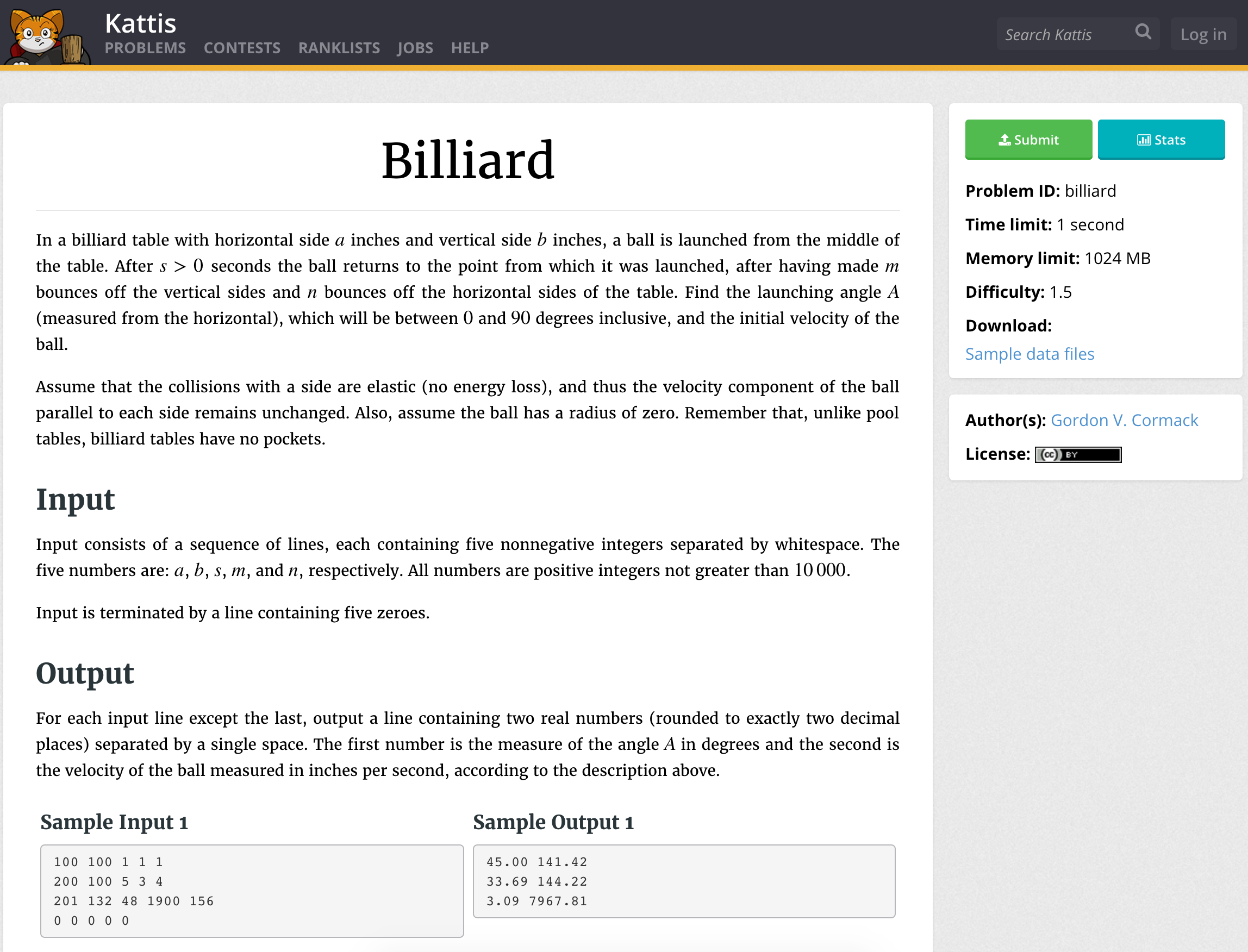
# 

# Appendix

These are the questions we asked during the interviews

* Background?
* Age, gender, education, abilities/disabilities, general computer knowledge?
* The use of the system?
* How much is it used (how often and how much each time), the skills of using this system, the attitude, the number of users?
* The context of use?
* The real environment, the technical environment?
* The main users’ tasks?
* What do users want to do?
* How do they do this today?

**Related systems**  
We took a look at two similar systems; Mooshak and Kattis.



*Image 1: How problems are represented in Kattis.*

*Image 2. How submissions are represented in mooshak.*