Investigation of the feasibility and development of a user-friendly platform for creating and hosting programming contests as a recruitment aid for software companies

Bachelor's thesis in Software Engineering

Sebastian Bellevik
Tobias Tikka
Daniel Bäckström
Marcus Olsson
John Petersson
Niclas Alexandersson

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2015

Investigation of the feasibility and development of a user-friendly platform for creating and hosting programming contests as a recruitment aid for software companies

SEBASTIAN BELLEVIK
TOBIAS TIKKA
DANIEL BÄCKSTRÖM
MARCUS OLSSON
JOHN PETERSSON
NICLAS ALEXANDERSSON

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2015

Abstract

This thesis describes the development of a platform for creating and hosting programming contests, intended to aid software companies in their, often time-consuming and expensive, recruitment processes. Conducted on behalf of a company named Software Skills, the project aimed to test the feasibility of using such a platform as a tool for attracting and getting in touch with potential new recruits, and making it user-friendly and stable enough to go into active use within the industry.

The project resulted in a fully-functioning platform, with support for participating in contests using three different programming languages, with access to high-level APIs, an online coding environment with syntax highlighting, console output, and graphical visualization of program behavior. New contests may be implemented in Python, with no need for working with other languages.

The thesis follows the project from its beginning to its end, describing the methodologies used, the challenges encountered, the solutions devised, and the final results.

Sammanfattning

Rapporten beskriver utvecklingen av en plattform för att skapa och tillhandahålla programmeringstävlingar, som syftar till att underlätta för programvaruföretag i deras, ofta tidskrävande och dyra, rekryteringsprocesser. Utfört på uppdrag av ett företag som heter Software Skills syftar projektet till att undersöka möjligheten att använda en sådan plattform som ett verktyg för att attrahera och komma i kontakt med potentiella nya rekryter, och göra det användarvänligt och stabilt nog att tas i aktivt bruk inom branschen.

Projektet resulterade i en fullt fungerande plattform, med stöd för att delta i tävlingar på tre olika programspråk, med tillgång till högnivå-API:er, en internetbaserad programmeringsmiljö med syntaxmarkering, konsol och grafisk visualisering av programbeteende. Nya tävlingar kan skapas i Python, utan att behöva arbeta med andra språk.

Rapporten följer projektet från dess början till dess slut, och beskriver de metoder som används, de utmaningar som uppkom, de lösningar som utarbetades, och slutresultatet.

Acknowledgements

We would like to express our deepest appreciation to Henrik Enström at Software Skills, who has been the commissioner of this product. It is thanks to him that we have been able to carry through the project in a successful manner. A special gratitude should also be given to Anders Mac Key, who helped us with our project process early on in the project. Anders works as an Agile Coach at Software Skills and has given us clarity in how to work with Scrum, and what parts of it that might be redundant in our case.

Furthermore, we would also like to acknowledge the postdoctoral researcher from Chalmers University of Technology, Musard Balliu, who helped us plan the security aspects of our application. A special thanks goes out to Arne Linde and Ramona Enache, who have given us feedback and suggestions on how to improve our project. We appreciate all the help that we have gotten from the user test participants who provided us with usable and necessary information regarding the security and user experience of the Honeypot product.

Vocabulary

Administrator	A person with rights to create, edit and publish challenges as well as managing customers.
API	Application Programming Interface
Autocompletion	A feature that predicts the rest of the word a user is typing
Backend	Server side application running on the server
Backlog	The waiting list of User Stories yet to be implemented in the project
Challenge	A coding challenge created by Software Skills used to initiate contests.
CMS	Content Management System
Contest	A contest is an instance of a challenge over a specified time period initiated by a customer
Contestant	Person participating in a coding contest
Customer	Company requesting and hosting a contest from Software Skills
Design Studio	Meeting were a project team comes together to visualize potential solutions to a design problem.
DOM	Document Object Model
Frontend	Client side application running in the web-browser
Honeypot	The name of the product and project
IDE	Integrated Development Environment
JSON	JavaScript Object Notation, a lightweight data-interchange format.
MVC	Model View Controller
Sprint	Set period of time that determines a development cycle in a project group
Test Participant	Person partaking in a user test
The client	Software Skills, the company for which this project was conducted
Theme	A structural separation of the product content, where each part contains features that are coupled
User Story	A description of some functionality of the product
User Story Points	Measurement of how complex a User Story should be considered to be.
WebSockets	WebSockets is an advanced technology providing full-duplex communication between a web browser and a server.

Table of contents

Chapter 1: Introduction

Software is still a rapidly growing industry, and companies compete fiercely to find talented developers. Companies who reside in a market with global talent networks and social media, struggle to get a hold of the few unemployed with the required skills and talent. Lacking satisfying tools to reach this talent there is room for improvements in the field of software talent acquisition.

1.1 Purpose

The purpose of this bachelor’s thesis is to investigate the feasibility of and develop a user-friendly platform for creating and hosting programming contests as a recruitment aid for software companies. Having access to such a platform would allow companies to use programming contests as a means to attract, evaluate the skills of, and get in touch with potential recruits without having to actively advertise a job opening.

The development portion of the thesis is being conducted on behalf of the Gothenburg-based company Software Skills. The aim is for the final product to go into active use within the industry, as a service for Software Skills to sell to its customers.

1.2 Background

In the ever-changing world of software engineering, it is of utter importance to have a natural talent for the craft. It is not enough to master just one, or even a few languages since they are updated constantly, and new ones are introduced every year [1]. A programmer needs to be flexible and able to adapt to new technologies. For a talented programmer, the language or technology in question is irrelevant. What is important is the ability to use programming for solving any given task. One can speculate on how to best define talent in the field of software engineering from a business perspective. It may just be a matter of perspective, and it may be generally definable. To further investigate, we will first look at examples of hiring processes of two companies and what talent characteristics they are looking for.

At Intel, the process is first and foremost based on posting job openings on their web page, where job applicants can create candidate profiles and apply for jobs [2]. Thus, the company keeps a database of candidates and their supplied talents and skills, and so, when applications arrive, “Intel representatives can examine applicant profiles; assessing skills, education and experience against the requirements of an open position.”[2] The few applicants chosen are invited for an interview, and generally this is done to verify that the talents and skills of the applicants match those described in their profiles.

Microsoft uses a special blog, “Be Your Future,” to promote job openings and post other information relevant for people looking for information about the company [3]. The hiring process via the blog consists of four steps. Applicants can fill in an online form where they specify interests, skills, and other relevant information. Microsoft staff will then review the forms and, if interested, contact the applicant to set her up for an online test. Upon completion, a video interview will be scheduled with successful applicants. If the interview goes well, the applicant will get to spend a day at one of Microsoft's main offices, along with other applicants. Although this is a rather straightforward process, Microsoft staff is required to review a large number of application forms to even get a chance at assessing an applicant's skills and talents.

These examples suggest that large companies tend to spend significant amounts of time and money on finding "hopefully" talented people, and even then additional time and money to validate applicants' claims. Since this process is long and costly, it is important that the candidates truly exhibit talent in the domain of the job opening. Notice that we use the word *talent*, not *skill*, which are two quite different concepts with differing implications when it comes to software engineering. The difference is well explained in an article by Dale Reynolds, who talks about why one should aim to hire for talent and not for skills: "There is a huge difference between talents and skills. Like riding a bike, a skill is something that almost anyone can learn. A talent is something that a person does naturally, although we may spend years honing it. A person talented in software development can learn a new skill, e.g., a new programming language, in a matter of a few days. This is because they have a natural ability to think in design terms, and they can easily use whatever tools are available. Just as Tiger Woods can easily embrace a new type of club or an unfamiliar course, a good software designer can easily embrace a new tool."[4]

What this project aims to accomplish is to remove the – often time-consuming and costly – first step that many companies use to gather and review application forms. Instead, it aims to let them find talented people directly, with the use of customizable language independent code challenges. After someone has completed a challenge, the company can simply filter out people with a low score and focus its attention on the ones with the most talent. As the company can choose challenges according to the relevant business domain, they can get an indication that people who achieve a good score have a natural talent in that domain.

One might wonder how these challenges can provide a good indication of talent. The answer would be that the company can use the proposed solution to assess the applicant's flexibility and ability to quickly learn how to use new tools. This is because the creator of the challenge specifies which tools and API the applicant will have at her disposal. As Dale Reynolds described above, this is considered a virtue of a talented software engineer.

The language used to solve the challenge has little relevance, since a talented software engineer can likely learn to use new languages in weeks. The purpose of such a challenge and the evaluation of the applicant's code is clearly described in an article by Hyam Singer: "Present the candidate with a problem and ask her to code a solution in the language of their choosing. The only requirements should be that the solution be thorough and correct, including addressing any edge conditions or potential errors. Remember, the goal here is to evaluate the candidate's (a) ability to problem solve, (b) knowledge of computer science, and (c) coding style. You are not at this point evaluating aptitude in a specific programming language." [5]

Another problem the product would help solve is the fact that most talented software engineers are not even in the market for a new job, as they are likely to already be employed. Because of the almost non-existent unemployment rate among software engineers, especially smaller businesses have a hard time reaching out to talent [6]. Also, potential talent that is already employed is less likely to look for a new job unless dissatisfied with their current one or convinced by someone else. What this project hopes to achieve is a solution for companies where they can easily reach out and pique the interest of anyone who enjoys programming and a good challenge. Using Honeypot, the companies can present their code challenges as a contest to reach a large group of talented people. If the contestant achieves a high enough score, the company can simply contact them, without spending a fortune on tech recruiters or wasting manpower on going through job applications.

1.3 Challenges

As stated and elaborated on in section 1.2, the aim of this bachelor's thesis is to develop a platform that uses programming contests as a means for helping software companies find potential recruits. Central to achieving this goal are three major groups of concerns: functionality, usability, and viability for production use. Functionality concerns involve the basic requirements for creating a working platform for hosting programming contests. Additional constraints are added by the fact that the platform is targeted at companies looking to use it as an aid in their recruitment processes: the platform needs to be attractive for both contestants and companies, making usability important. Finally, due to the fact that the platform is intended to go into active use within the industry, it must be ready for production in the sense that it is scalable, secure, stable, easy to maintain, and easy to extend with new types of contests. These three groups of concerns are the pillars on which all decisions made during the work with the thesis rest.

The following subsections will go into more depth regarding the challenges involved in developing specific aspects of the platform. While functionality, usability, and viability for production use are the underlying reasons behind these more specific requirements, they are poorly suited as a way of grouping said requirements. Therefore, the challenges have been divided into a different set of groups: those concerning the experience for contestants, those concerning the experience for company users, those concerning the experience for administrators, and those concerning the platform itself.

1.3.1 Contestant experience

To be an effective tool for helping companies find potential recruits, the platform needs to be able to attract people who actively choose to participate in the contests. Contestants need to be convinced to stay long enough to allow their performances to be evaluated. This places a requirement on the individual contests to be interesting enough to be worth the time of the contestants, and on the platform itself to present the contests in a way that does not deter potential contestants from participating.

To avoid deterring potential contestants, it is important that the platform makes it easy to get started and reduces unnecessary work on the contestants' part to a minimum. The platform should strive to make contestants feel comfortable within the environment, give clear and immediate feedback, and provide the necessary guidance to quickly learn how to use the platform and participate in the contest.

Because contestants may have different backgrounds and preferences, one key aspect of making them feel comfortable is to provide them with a number of different options as to which programming language to use. Programming in their language of choice allows them to focus on the actual contest instead of on getting the syntax right. Another step towards this goal is to ensure that each challenge provides a sufficiently high-level API to move the focus away from dealing with low-level problems such as parsing textual input and output, which is sometimes required in other programming contests. It might also be desirable to allow programmers to print output to the console as a way of debugging their solutions.

As most people who have ever written a program are aware, the code does not always work as the programmer intended. There may be syntax errors, mistakes, bugs, or other problems that cause a program to fail to compile, crash during runtime, or enter an infinite loop. These problems need to be detected when they occur so that the evaluation can be halted, and appropriate feedback be given to the programmer.

An important part of the success of a contest is exposure, and a major avenue for this is through social media. This can be taken advantage of by allowing competitors to share their results through these kinds of services.

Finally, there needs to be some form of incentive for contestants to provide the company hosting the challenge with their contact details. Ideally, this should be done as unintrusively as possible, as to not become a deterrent to participation.

1.3.2 Company experience

To be able to reach competent candidates through coding contests, companies will need the ability to request challenges that are relevant for their current goal. The available challenges should be easily accessible and should include a clear description, making the process of choosing a challenge for use in a contest as easy as possible. Before hosting a contest, the company should have the ability to brand it, making sure that contestants will understand which company is responsible. Other than branding, companies should have the option to add a description to clearly state the goal of the contest as well as awakening the interest of possible contestants.

During the contest, and especially afterward, one of the most important tasks for the responsible company will be to review the contestants' results. The reviewers should not only be able to view the individual scores and comparison of contestants, but also comprehensive details about each submission. If interesting candidates are found, the companies need a way of contacting them. Each contestant’s relevant contact details should, therefore, be made accessible while reviewing their results.

1.3.3 Administrator experience

To make this product viable for the client, the process of creating a challenge has been as simple and straightforward as possible. While the framework will support the use of multiple programming languages when solving a challenge, no extra burden must be placed on the administrator when constructing it. The framework provided for constructing new challenges must, therefore, be independent of which programming languages contestants will use to solve them. Communication between the separate languages must be taken care of in the background. The necessary tools needed for creating a challenge must be easily accessible and self-explanatory. Moreover, editing an existing challenge needs to be equally straightforward.

The administrator needs access to a list of companies with their associated contact details, which should be editable by the administrator to account for new and leaving customers. The administrator shall also have the option to decide which companies have access to a finished challenge.

1.3.4 Platform requirements

Challenges in this project are centered on writing code, with code execution as the primary means of evaluation. Thus, the perhaps most essential feature of the platform will be the ability to compile and execute said code. This must be possible in any of the supported languages and not just a single one. If additional files are needed for compilation or execution, these will need to be placed in the appropriate locations before compilation begins.

To be able to provide the service and its challenges to a larger audience in a successful manner, it is important that the system is both stable and secure, as well as scalable.

Because the evaluation of solutions involves executing untrusted code on the server, a significant concern is going to be security. Because the contests are going to be open to anyone, this might be seen as an invitation for attackers to try to execute malicious code on the server or, perhaps less harmfully, cheat. Competing programmers must not be able to make their programs access other contestants’ code, or otherwise wreak havoc on the server. Especially important, is mitigating the risks of having user data such as passwords and email addresses compromised.

If the service is to be able to grow and attract a large number of contestants, there are at least two things to consider. First, it should be possible to extend the code base and add new features to the platform with time; the code structure and the choices of technologies are therefore crucial from the beginning. Second, it should both be scalable in the sense that a lot of contestants should be able to take the challenge at the same time, as well as that a lot of customers should be able to subscribe and use the product.

1.4 Scope

This thesis applies a very specific solution to a much more general problem. Thus, it is important to understand that the final product is not intended to be used as a complete solution to all of a company’s recruitment problems, but rather as an additional tool in a much larger toolbox.

While the project aims to make it easier to find people with programming and problem-solving ability, it will limit itself to doing so through automatic evaluation of code functionality and performance alone. Code that performs well according to the criteria defined by the challenge will still need to be inspected manually for things like clarity, maintainability, and conformance to best practices. Also, contestants will need to be contacted for more traditional interviews to ensure that they will be a good fit for the company.

Though it might be possible to create automatically evaluated contests that assess qualities other than programming ability, such as memory or multitasking ability, doing so will not be within the scope of this project. While the format might be similar, the challenges involved in creating such a platform are very different from what is to be done in this project. This would require valuable development resources to be diverted from the project’s main focus.

1.5 Outline

Next chapter will shortly explain the methods and processes used throughout the project and contain a pre-study. Shortly after, a theory section will follow that will introduce the frameworks, techniques and technical theory used to lay a strong foundation for the project, and help in the implementation process. After that, in chapter 4, with the use of the technical background, the realization of the product is explained thoroughly. Chapter 5 explains the design and implementation of the developed application after which the result of the project is presented and discussed. Finally, a conclusion of the project is briefly described and disclosed.

Chapter 2: Method

The project was initially divided into three separate phases: pre-study, implementation and evaluation. Since it was clear from the beginning that the final product would be delivered to Software Skills and used in the industry, additional emphasis was put on the quality and reliability of the code. This chapter will elaborate on the three different phases and give a general picture of the scope of the project.

2.1 Pre-study

Before any work on implementing the product itself could commence, an informal pre-study was conducted. The purpose of the pre-study was to determine the ideal scope of the project and gather specific requirements for the final product. This was achieved by means of discussions with the client, to get a clearer understanding of their expectations; a competitor analysis, to see which other platforms are on the market and what they are capable of; and preliminary architecture planning, to predict potential problems and ensure that solutions for them were available. Further details about this process and the achieved results can be found in section 4.1.

2.2 Implementation process

During the implementation process of the project, the software development methodology used was Scrum, together with the concept of themes, which is a way of structurally separate a product into more manageable components, from the Lean UX methodology. The software Git was used for revision control together with the web-based Git repository hosting service GitHub.

2.3 Evaluation

To ensure the project was on the right track regarding usability, evaluation was performed throughout the project. This took the form of usability testing and an active dialog with the client who ordered the product.

Chapter 3: Theory

Before going further into the details of the rest of this thesis, there are a number of concepts that may be helpful for the reader to be familiar with. In this chapter, introductions are given to the MEAN-stack and its components, the basics of executing code programmatically, interaction design, web design, the purpose of modularity, and the principles of agile processes. Readers who are already familiar with some of these topics may skip the respective parts.

3.1 The MEAN-stack

The MEAN-stack is a collection of JavaScript technologies used to develop web applications [7]. The acronym MEAN stands for MongoDB, ExpressJS, AngularJS and Node.js. With the MEAN-stack, one can build large scale fully dynamic web applications utilizing just one programming language, namely JavaScript. The concept of using just one language throughout the web application can also be referred to as a "full-stack" web application. [8]

There are several advantages to using a uniform language throughout the layers of the whole application. It is easier to switch between contexts since the parts are written in a similar manner. With JavaScript throughout, one seamlessly uses JSON when communicating between the components and layers of the application. A developer working on the frontend can easily understand the backend code and database queries. With the same syntax and object structure throughout, one does not have to consider best practices of several languages. This reduces the barrier of entry for developers to understand the codebase.

Compared to traditional server-side page generation stacks such as LAMP (Linux, Apache, MySQL, PHP) [9], MEAN instead moves towards a client-side single-page application (SPA). The emphasis of the application is therefore moved to the client-side. With the use of AngularJS, it is possible to move the Model-View-Controller (MVC) artifacts from the backend to the frontend. One benefit of this is significant load reduction on the backend. This reduction, together with Node.js, makes it easier to write RESTful web services – further explained in section 3.1.3 – of asynchronous event-driven nature.

3.1.1 MongoDB

MongoDB is part of the NoSQL family of database systems. The name is inspired by the word "humongous" which alludes to the claimed ability to handle large data sets efficiently. The database system is document based, storing data in structured JSON-like formatted documents. The format is called BSON and like JSON follows a nested object-oriented structure. This makes integration into object-oriented languages such as JavaScript easy and fast. [10, 11]

The schemas of MongoDB are flexible and not required to be fully defined before populating the database with data. This enables a more agile approach to database modeling compared to “classical” SQL based database systems. In MongoDB, one can iteratively extend the schema without the process of time-consuming database migrations. In MongoDB, every document has an id property of the type ObjectId. By referring to other documents via ObjectId:s, it is possible to model the database with a relational model.

Mongoose

Mongoose is an object data modeling (ODM) library that makes it possible to model the schemas in the form of objects. [12] Mongoose makes the modeling environment for data rigorous, enforcing structure as needed while still maintaining the flexibility that makes MongoDB powerful. Mongoose is an active popular library that is used by large scale web applications like Linkedin and Trello. [12] The ease of modeling schemas in an agile manner without migrating data is something that fits agile processes well (see 3.6).

3.1.2 Node.js

Node.js is a server-based solution designed to facilitate scalability and speed in web applications.. The platform is written in Javascript and use event-driven, asynchronous I/O to minimize overhead and maximize scalability. [13]

When handling a web application with a traditional web server such as Apache, the interaction between the users and the application is thread-based. The user connects to the application and is placed on a queue on the web server, which is managed according to the principle of "first come first served". This can result in long load times during heavy traffic to the site. [14]. With Node.js, on the other hand, the interaction is event-based. The user connects to the application, creating an event (e.g. delivery of a page). Node.js continues to handle this request asynchronously with other requests, so that queues are less likely. [15]

3.1.3 Express.js and RESTful API’s

Express.js is a Node.js web application framework[16]. With Express.js it is easy to define stateless RESTful API’s, which is a software architecture style including guidelines and best practices for creating scalable web services [17]. An implemented Express.js service usually provides a RESTful API-interface consisting of lists of specified Hypertext Transfer Protocol (HTTP) routes [18]. Each route is specified with an HTTP verb (GET, PUT, POST, DELETE, etc.) to handle a request of the corresponding type [19].

3.1.4 AngularJS

AngularJS is a client-side framework not mainly focusing on document object model (DOM) manipulation. This sets the framework apart from the more common jQuery, which is a library used to manipulate the DOM, create animations, handle events, and develop Ajax applications. Instead, the emphasis lies on updating the state of the model in the MVC pattern, and then the DOM will be updated automatically via data-binding into the model.

A powerful feature of AngularJS is the possibility to extend HTML with new definable tags called directives. The main components used to handle front-end logic are controllers and services. Services are mainly used for handling state and communicating with the backend. Hence they correspond to the model in the MVC architecture pattern. Controllers, on the other hand, are used for initiating services and responding to events fired from HTML elements. [20, 21]

Modularity is natively supported by AngularJS using the built-in function angular.module() [22]. Here, a module is a container for everything that a part of the application needs in terms of AngularJS functionality, such as controllers, services and directives. The main application is then composed of several modules, each of which may in turn consist of several other modules.

3.2 Code execution

3.2.1 Running code

Running code on a computer requires that several aspects are taken into consideration. First of all, the tools needed to compile and execute code in a specific language need to be installed. One major difference between languages is that some are in need of compilation before the code can be executed, while others can be executed immediately. The process of compiling code consist of providing the code access to all required code libraries and then initiating the compilation. An example of a common programming language that need compiling before a program can be executed is Java [23]. When executing a program, the user running the code also requires the necessary privileges for accessing the processes needed. Once the code is executed, one or several processes are initiated and can receive input and send output.

3.2.2 Communication between processes

Interprocess communication (IPC) is a term used to describe the act of exchanging information between two or more separate software processes. There are several ways in which this exchange of information may take place. The UNIX operating system provides a number of facilities for IPC, including pipes and FIFOs, message queues, stream and datagram sockets and shared memory [24].

Pipes and FIFOs

Pipes allow data to be sent as byte streams between a parent and a child process, or between two different child processes of the same parent. FIFOs are a generalization of this, allowing the same means of communication between any two processes with access to the name of the FIFO. Pipes can be used to make one process write to the standard input, or read from the standard output or standard error of another. [24]

Message queues
Message queues can be used to send data as complete units – messages – instead of as continuous streams of bytes. In System V message queues, messages may be divided into different types, allowing processes to choose whether to retrieve messages of any type or only of a specific one. In POSIX message queues, each message has an associated priority determining its position in the queue. [24]

Sockets
Sockets are a way of achieving two-way communication between separate processes. Stream sockets allow information to be sent as continuous streams of bytes, while datagram sockets send their data as complete, separate units – datagrams. Sockets may be used to connect two processes residing locally on the same system, or as a way for two processes residing on separate systems to communicate over a network. [24]

Shared memory
Communication between processes can also be achieved by allowing them to read and write from a shared region of memory. Memory written to the region by one process may immediately be read by another. Shared memory imposes no restrictions on which order data may be read in, allowing random access to data anywhere within the shared region. A drawback of this type of IPC, however, is that it offers no built-in synchronization, meaning that all communication must be synchronized externally. [24]

3.2.3 Security through sandboxing

A Sandbox is an environment in which possibly malicious code may be run without risk of compromising the underlying computer system. There are a number of different approaches to sandboxing untrusted code. Each one comes with its benefits and drawbacks, and which one is most suitable depends on the requirements of the individual application.

Virtualization
One common approach to constructing a sandbox is Virtualization [25]. Sandboxed applications are executed in a Virtual Machine (VM), either at the process level or the operating system level. In both cases, the VM acts as a container for the sandboxed process, isolating it from the outside system. A possible drawback is that starting a VM for running a process can have some performance overhead.

Verification and rewriting
Another way to sandbox an application is to inspect the code or binary executable before execution, and either verify that it cannot perform any security violations [26], or rewrite it to replace potentially violating instructions with harmless ones [25]. For this to work, programs written in different programming languages need to be converted to a common format, or tools for inspection need to be used for each separate language.

Operating system security

Sandboxing can also be performed by making use of the operating system's security facilities, such as running the application in question using a non-privileged user with little or no access to system resources. A limitation of this way of sandboxing is that that it can be hard to migrate between platforms, since built-in security features may vary between different operating systems.

3.3 Interaction design

3.3.1 Usability

The ISO definition of usability is "[The] extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use" [27]. Effectiveness, in this case, concerns whether or not the user can achieve the intended goal while efficiency relates to the effort required [28]. A shoe can be as effective as a hammer for driving a nail through a board, but the hammer is a more efficient tool for the task. Moreover, the definition states that usability is specific for the user, her goals and the context in question. What this means is that it depends on the experience of said user, as well as her potential disabilities. *What* she is trying to achieve is important, as well as the technical platform, physical surroundings and social environment.

3.3.2 System Usability Scale

The System Usability Scale (SUS) is a standardized and well-used method of measuring subjective usability [29]. Its general nature makes it technologically agnostic, meaning that it can be applied to a wide variety of platforms. It was created to be a lightweight, "quick and dirty" tool for low-cost evaluation [30, 31]. The scale consists of ten statements that the user grades her agreement to on a scale from one to five. The even numbered statements are worded in a positive tone while the odd ones are worded negatively. The level of agreement to the statements is then used to calculate a total score, giving an indication of the system's general usability.

In detail, the scoring works as follows:

1. Sum the scores of each item. If it is a positive statement, this score is equal to the scale position minus 1. For a negative statement, it is equal to 5 minus the scale position.
2. Multiply the sum of the scores by 2.5 to obtain a total score.

The total score will be between 1 and 100. The higher the score, the better the usability.

While a single number does not say much about what specifically is good or bad in a system, it can be an indication of its *general* “goodness”. As for what constitutes a “good” score there are no set limits. Bangor et al. suggests in *An Empirical Evaluation of the System Usability Scale* [30] the following levels. A score of 70 or above can be considered “passable,” while “better products” receive between 70 and 80. To be a “truly superior” product, a score of above 90 is required, and falling below 70 would earn the label “at best mediocre”.

3.4 Web design

When building a web application, there are many things to consider. The web is ever-evolving, and new technologies are introduced all the time, but these new technologies are not always supported by all the major web browsers [32]. What technologies to incorporate is a hard question, where one has to look at the kind of website one is building and who the target audience is. The two main areas for consideration are which layout mode to use and how to animate things.

3.4.2 CSS layout modes

A CSS layout mode can be described as what determines how a box is positioned and sized within in a web page, based on the way it interacts with other boxes around it. For a long time, it has been difficult to build web pages that scale based on the web browser's viewport size as one would like. One would have to write a substantial amount of extra CSS, and sometimes even make use of JavaScript, to make the design of a web page look and behave as desired. This is because there is no easy way of controlling the direction of child elements within a box when using the conventional *block* layout mode, nor is there an easy way to make a box occupy all the available space of its parent element. *Flexible box*, on the other hand, is a new and modern layout mode available in CSS3, built specifically with these scenarios in mind [33]. It provides easy ways of achieving what the block layout mode fails to do and thus makes it easier to build complex designs with less and cleaner code.

3.4.3 Animations

There are two main ways of making animations happen on a website: either by using JavaScript or by using the animation and transition functionality available in CSS3. They each have their strengths and weaknesses, with JavaScript having better browser support while lacking the simplicity and performance optimization that one gets with CSS3. [34, 35]

3.4.4 Vendor prefixes

New CSS technologies are always first implemented in web browsers with web browser specific prefixes, which one has to know and use when writing the CSS to make use of the new technologies [36]. WebKit-based web browsers, for example, use -webkit-[the real CSS property name]. It can be difficult to remember when and what prefixes to use, and when not to. As a result, the CSS may become more complex, as in this example:

Without vendor prefixes

```
a {
    display: flex;
}
```

With the correct vendor prefixes

```
a {
    display: -webkit-box;
    display: -webkit-flex;
    display: -ms-flexbox;
    display: flex
}
```

This is where a so-called auto-prefixer is useful, a small program which, given unprefixed CSS, will add all the necessary prefixes automatically. [37]

3.5 Modularity

Modular programming is a way of developing software by looking at a program as the sum of its clearly separable parts. Each part, or *module*, is separate from the other parts and has a well-defined function. The modules are also supposed to be interchangeable, meaning they could be updated or replaced without changing the structure of a project. [40]

In large software projects, there is much to be gained by splitting a project into smaller, clearly separated modules. Large and complex systems can be difficult to understand for the individual, but by implementing modular programming, each developer only needs to understand a part of the project. Modular programming also facilitates the development of software when the development team is large, by making it possible for parts of the development team to work undisturbed on separate parts of the project. [38] Additional advantages of modular programming are that it becomes easier to maintain a program, as it is easier to debug a problem when the code is separated into separate parts, and that testing can be made more specific and only test one small part of the whole system. Another benefit of working with modularity is the reusability of the different modules. Code written in a module for a specific project may be reused with similar functionality in a completely different project with little to no modifications. [39]

3.6 Agile process

3.6.1 Project methodologies

To succeed in creating a product within a specified time frame, it is advisable to use a suitable and proven project methodology. A popular project management concept in software projects is the Agile methodology, where the development process is more liquid in its behavior compared to the more traditional, timeboxed waterfall methods [41].

3.6.2 Scrum

Scrum is a process framework that has been used with increased frequency throughout software projects since the early 1990s and is often applied in the management of complex product development projects [42]. Concepts that can be found in projects where Scrum is implemented are User Stories, Sprints and Backlog. According to Mike Cohn, who is one of the founders of Scrum Alliance, it is considered a best practice for user stories to follow the I.N.V.E.S.T. acronym; the user story should be Independent, Negotiable, Valuable to users and customers, Estimable, Small and Testable [43].

A sprint is a set period of time where the project group concentrates their efforts on a specified area of the product [44]. The focus area is determined in advance during the sprint planning, where the project team gathers and decide which User Stories to focus on under the coming sprint.

3.6.3 LEAN UX

The focus of the LEAN methodology is to manage a project or company in the same manner as a startup. To succeed in a competitive and fast-changing market, the process must adapt as well. A key concept is the Minimum Viable Product, which means that the development team is working towards producing a working implementation of the product in each iteration [45]. Using this approach, the development team can test their assumptions about the product at an early stage, and adjust the development strategy as new insights arise and data are presented to them.

Chapter 4: Realization

Supported by the method description in chapter 2, and the theoretical descriptions in chapter 3, this chapter will go more into depth as to how the project was realized. It will follow the structure from chapter 2, and go into details about how the pre-study was conducted, give a description of the implementation process, and finally go over how the user testing was carried out.

4.1 Pre-study

As mentioned in section 2.1, the pre-study consisted of three main parts: discussions with the client, a competitor analysis, and preliminary architecture planning. Due to the informal nature of the pre-study, there was some overlap among these three parts, both in time and in activities. Thus, it is important to understand that while they are described separately here, they were never intended to be independent and have actively impacted each other throughout the course of the pre-study.

4.1.1 Client discussions

To ensure that the final product would live up to the expectations of the client, multiple discussions took place throughout the course of the pre-study. Insights gained from the parallel work with the competitor analysis and architecture planning were used as a basis for discussion and helped identify concrete requirements for the platform.

Additional restrictions placed on the platform's design at the request of the client were that the platform should be web-based, and that there should only be a need to use a single programming language when implementing a new challenge, namely Python. Following research into language popularity [46], Java, JavaScript, and Python were chosen as the initial programming languages for the platform to support, with the option of adding C++, C#, and Ruby in the future. It was also determined that visualization of the behavior of contestants' programs should be used as a way of setting the platform apart from similar systems.

4.1.2 Competitor analysis

In order to understand the market of online coding evaluation platforms and to get an idea of what might be expected from such a platform, as well as determine what can be done better, a competitor analysis was conducted. To perform this analysis, a number of similar products were investigated, where their graphical user interface, performance, popularity and similarity to the Honeypot concept were compared against each other. After this, two of the evaluated competitors stood out: Codility and Kattis.

Codility is a platform where companies can subscribe on various kinds of programming problems to filter through applicants for programming jobs [47]. The client has been using Codility in the past and, therefore, has a good understanding of how their system is structured.

Kattis is, similarly to Codility, a way of testing the code quality of candidates [48]. Evaluating their user interface, it was found to be missing an intuitive way to evaluate code. When using Kattis, each candidate needs to send her code solution via email to their servers for evaluation, which was deemed an unnecessary step that might provoke frustration among the users of the product.

4.1.3 Architecture planning

A significant portion of the work with the pre-study was taken up by preliminary architecture planning. Planning the architecture before starting initial development allowed potential problems to be discovered early, and solutions to be devised for them. It also gave a clearer view of what would be feasible to achieve and helped determine a reasonable scope for the project. Finally, having a preliminary architecture simplified the planning of the development phase, by making the big decisions in advance and providing a basic idea of what work would need to be performed.

An early decision was that the platform should, to the greatest extent possible, be written in a single programming language. This was so that development could be sped up, by reducing the need to learn more than one language when working on separate parts of the system. The two main candidates being JavaScript and Python, it was decided that JavaScript should be used as the primary programming language due to a combination of it being natively supported in modern web browsers and prior experience of the team. It was later requested by the client that contests should be implemented in Python. This request was accommodated by deciding to separate the part of the platform used for evaluating contestants' solutions from the rest of the system and implementing only that part in Python, in an effort to minimize the need for communication between different programming languages in the same part of the platform.

One of the major challenges at the architectural level was how to combine the flexibility of allowing contestants to write their code in multiple programming languages with the comfort of having access to a high-level API. While the problem could be solved for individual types of contests by providing native code for handling communication between the contestant's solution and the evaluating code, this native code would need to be written in every supported language, for every separate type of contest. This would have made the implementation of new challenges unacceptably cumbersome; a better approach was needed. The final solution was to require the API for each type of challenge to be specified in JSON, and automatically generating the language-specific communication code for each programming language and challenge, based on this specification.

Another architectural challenge was how to visualize the behavior of contestants' solutions on the web, for all separate types of contests. Because the platform needed to support the easy creation of entirely new types of contests, the solution needed to be general enough to enable as broad a range of potential uses as possible. Specifically, implementing a new type of challenge should not require writing any JavaScript for visualizing it on the web. To accomplish this, it was decided that a simple Python library for allowing contest implementations to create and manipulate graphical objects over discrete time steps was to be created. All changes made to the created objects while running a contestant's solution would be recorded in Python and then be transmitted back as part of the result, to be played back in the browser using JavaScript.

4.2 Implementation process

The implementation process was divided into two major *themes*: Contest System and Customer Management System (see figure 1). Time was also set apart for documentation, user testing and code refactoring. These themes were later given a preliminary number of 2-week long *sprints.* The Product Backlog and the supposed functionality for the application was created during the initial stage of the project, using the pre-study as a base. It was continually changed throughout the project after discussions between the client and the project group, or product owner. Each sprint started out with a *Sprint Meeting* where *User Stories* from the Product Backlog were weighted against each other. Each project member was assigned the same amount of *User Story Points* each sprint during the sprint meeting.

Design drafts and the user experience of the product were decided during the *Design Studio* session that was held the first Sprint Meeting for each theme. The final graphical design was later assigned to one of the team members. After the design for a specific theme was determined, the development process could begin. Suitable development tools were chosen according to each team member's personal preferences. Git was used by all project members for version control and was stored at a private Organization Account on Github [49].

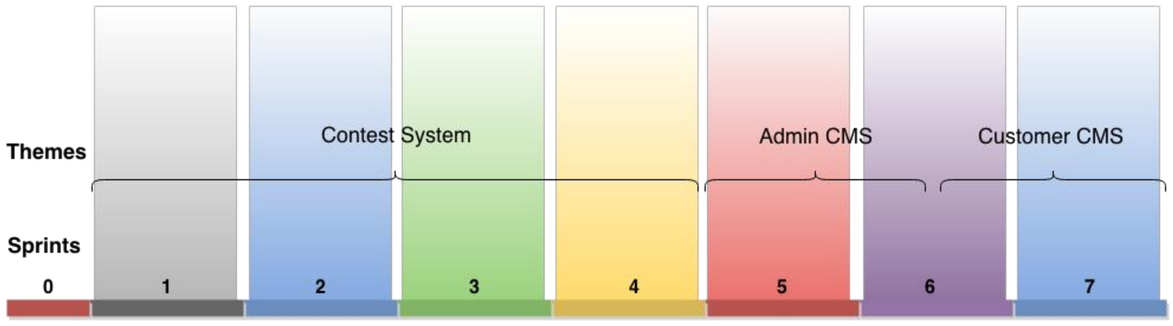


FIGURE 1 - IMPLEMENTATION PROCESS

4.3 Evaluation

For this project, a "quick and dirty"-approach [50] to evaluation was adopted, partly because the development team could be seen as representatives of part of the target group and, therefore, provide valuable input, and partly due to time and cost constraints. This approach eliminates the rigors of a lab environment and focuses more on informality and speed than on structure and correctness, and is useful for obtaining fast results with little or no budget.

The evaluation of the product was three-part: continuous evaluation throughout the project by the customer, observational user tests on views, and a live contest run under intended working conditions with a follow-up questionnaire.

With the time limitations considered, it was decided to not perform any tests during the early stages of development, and to instead wait until a working prototype was ready. As each view became ready enough for testing, small-scale user feedback sessions were conducted to gauge if the views were perceived as fulfilling the project's goals of being usable, intuitive and aesthetically pleasing. The observations obtained from these would then be incorporated into the development of the product.

Towards the end of the development phase, a larger-scale test was performed on the nearly fully featured live product in its intended working environment. This meant that participants used the software online in their natural environment, using their own computers. The accompanying questionnaire had a dual focus. In part, it contained as set of statements about to the System Usability Scale (see 3.3.2). The other emphasis of the questionnaire was to find as many software bugs and improvement areas as possible before the product was to be delivered to the client.

4.3.1 Client evaluation

Throughout the project, informal meetings were held at the conclusion of each sprint to keep the client updated on the progress. In addition to showcasing the work, these meetings served as feedback sessions where the input from the client was taken into account in the evaluation process.

4.3.2 Observational user tests

The tests were carried out by observing participants performing a series of tasks, at the end of which they were asked a set of questions. These questions consisted both of a series of statements concerning the system's usability and aesthetics that the participant graded her agreement to, and open-ended follow-up questions about suggestions for improvements and comments in general.

Test metrics

In addition to the qualitative metrics obtained from asking questions, a set of quantitative ones were measured. Since the main reasons for quantitative metrics are to track progress between releases and to assess one's competitive position compared to other companies [51], both of which were of limited concern to this project (as it will only contain one release and is sufficiently different from it is peers not to be reliant on performance for competition), they were kept simple. Task success percentage, average time on task and error rate were measured.

Test group size
Regarding to the test group, 6-12 participants are considered typical size. It has been shown that 80% of usability problems are discovered with five participants. The benefit of doubling that amount only results in a 10% increase [52]. The group sizes were therefore aimed towards the lower end of the typical measure at 6, given the project's limitations on timeframe and resources. In the case of the challenge view user testing, a clear trend could be seen early, and since results did not vary much it was decided to conclude testing after four participants.

What was tested
The plan was to test all of the views separately as each of them reached a working prototype state. However, since the client feedback on the customer and admin views was so positive in nature it was decided to down prioritize their testing. Therefore, the only view that was tested was the challenge view.

4.3.3 Live working conditions test with questionnaire

To keep evaluation lightweight, a method called System Usability Scale (SUS) was used (see 3.3.2).

The questionnaire was adapted to better suit the product in two ways (see figure 24 in appendix A). First by replacing the word "system" in the statements with the more domain relevant "application". Also, since the product is not intended for frequent use, the first statement, "I think that I would like to use this application frequently" was deemed misleading. Removing it, however, would have the undesired consequence of the total score being skewed and, therefore, unsuitable for its intended use – the comparison with other products. The statement was therefore altered to read "I think that I would like to use this application again". While these changes where small, they might cause the result to become somewhat misleading, and therefore have to be taken into account in its interpretation.

Bugs and changes feedback
In addition to the questions related to the SUS, the questionnaire also contained a section where participants were encouraged to supply feedback on what they would change in the application, and to recount all the software bugs that they had found while using the product.

How the test was performed
To have something to test on, a fully working example challenge was created and hosted on a temporary domain. To have as broad a base of results as possible, the link to this was then distributed via Facebook to anyone and everyone that might have been interested. Participants went through the completion of the example challenge without any extra test guidelines than those included in the actual application. Upon receiving their score, they were provided a link to a Google form containing a questionnaire that they were then asked to fill in.

Chapter 5: Design and implementation

This chapter will go into greater technical depth as to how the project was implemented. It will give an overview of the functionality that was implemented, and explain how the challenges identified in section 1.3 were dealt with.

5.1 Features

In order to easier understand the implementation of the platform, an overview of the functionality that was actually implemented may be helpful. This section will go over the main features which were included, and explain the reasoning behind their inclusion.

5.1.1 Online code editor

In order to make the coding process as streamlined as possible, and also to contribute to a positive contestant experience, a code editor was included as a part of the web application. The reason for this was to eliminate the tedious process of using several applications for solving a single problem, sending a solution for evaluation manually and waiting for a response. This setup was made to resemble a typical coding environment, making it recognisable for as many people as possible. It included autocompletion, error indication and syntax highlighting for the supported languages. In addition to the code editor the view includes a console for program output and a section for the API documentation.

5.1.2 Persistence of solution code

Some challenges might take more time to solve and can be difficult to complete in one session, making persistence of the contestants code important. A contestants written code in each of the available languages is stored and automatically restored upon returning to the contest.

5.1.3 Support for multiple languages

The languages Java, Javascript, and Python are all supported with full syntax highlighting in the code editor. Each language provides the contestants with access to a high-level API with support for function calls to and from the contestants' code, each following the conventions of its respective language. The scoring of a contestant's solution is not in any way affected by the choice of language.

5.1.4 Quick feedback

As the code editor is built into the web application, the project aims to give users the same kind of fast feedback that people would get while programming in any IDE. After the code has been sent to the backend for execution, there is only a short delay before the result is presented to the user in the form of console output in the embedded console and graphical updates on the canvas. Giving contestants access to the console output of their programs is a vital part in making the system accessible for as many people as possible, as debugging a program and receiving compile or runtime errors are an important part of many programmers' workflow. The graphical feedback is described in more detail in section 5.1.5.

5.1.5 Graphical visualization

One element that sets this product apart from other similar products is the graphical visualization when executing code. Other than standard console output, contestants can see the behavior of their programs, visualized on by animating objects on a virtual canvas. Being able to get some visual representation of what is happening makes for more variety when constructing code challenges, as well as more ways of giving the users feedback. If the goal of a contest is to get a character from point A to point B through a maze using the quickest route, the result will not only be displayed as a score in the console, but also shown on the canvas as the character moving through the selected maze. The section of the web application containing the graphical visualization and the playback controls of the animation can be seen in figure 2.

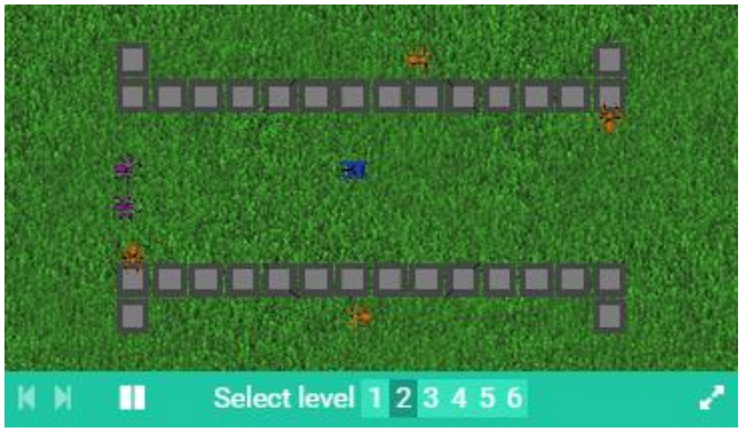


FIGURE 2 - CANVAS

5.1.6 Recording and playback of code solution

To make it possible for customers to judge how the contestants wrote their code, a tool capable of recording and playing back the process of writing that code was developed. This allows the customer to get an understanding of the contestant's thinking while developing the program, and how she progressed towards the finally submitted solution. With the tool, it is possible to step through the timeline of the submitted solution, effectively adding the recorded edits up to the final version of the submitted code.

5.1.7 Easy creation of challenges

An important step in making the final product viable for practical use, was ensuring that the creation of new challenges was as easy as possible. Therefore, the client is given access to a number of tools made to simplify the process of constructing a challenge. All code to implement the challenge is written in Python. Defining the API of a challenge is done via a simple graphical interface, and the code skeleton of the challenge is generated at the push of a button. The administrator constructing the challenge only has to focus on implementing the logic behind the functions needed, which is done in a syntax highlighted code editor, embedded in the CMS. Equally simple is the process of specifying the levels used for running and scoring the challenge, for which Python and a similar code editor are also used.

5.1.8 Responsive web design

An important part in making the contestant experience the best it can be for as many people as possible is making sure that the web application looks good and consistent, independent of what screen resolution the contestant might have. The way of achieving this is by building the web application using a *responsive* design. The concept of responsive web design was well explained by web designer Ethan Marcotte, in his article Responsive Web Design [53]: “Rather than tailoring disconnected designs to each of an ever-increasing number of web devices, we can treat them as facets of the same experience. We can design for an optimal viewing experience, but embed standards-based technologies into our designs to make them not only more flexible, but more adaptive to the media that renders them.”

5.1.9 Secure execution of solutions

To minimize the risk to the system, the execution of contestants’ code needs to be done in a safe manner. This is achieved by having the platform execute the contestants’ code within a sandboxed environment. The security features of the platform are described in greater detail in section 5.9.

5.1.10 Contestant rankings

All contestants are scored based on how well their solution performed on all of the levels of a challenge, and using the sum of these scores, the platform is able to rank the contestants. This ranking is available to the contestants after they have submitted their solution, in the form of a high score list showing all the contestants and their best scores. The goal of this high score list is to encourage the contestants to submit multiple times, in order to improve their score and position on the high score list. The ranking system is also used as a tool for the customers, to order the contestants that submitted solutions for a particular contest.

5.2 System architecture

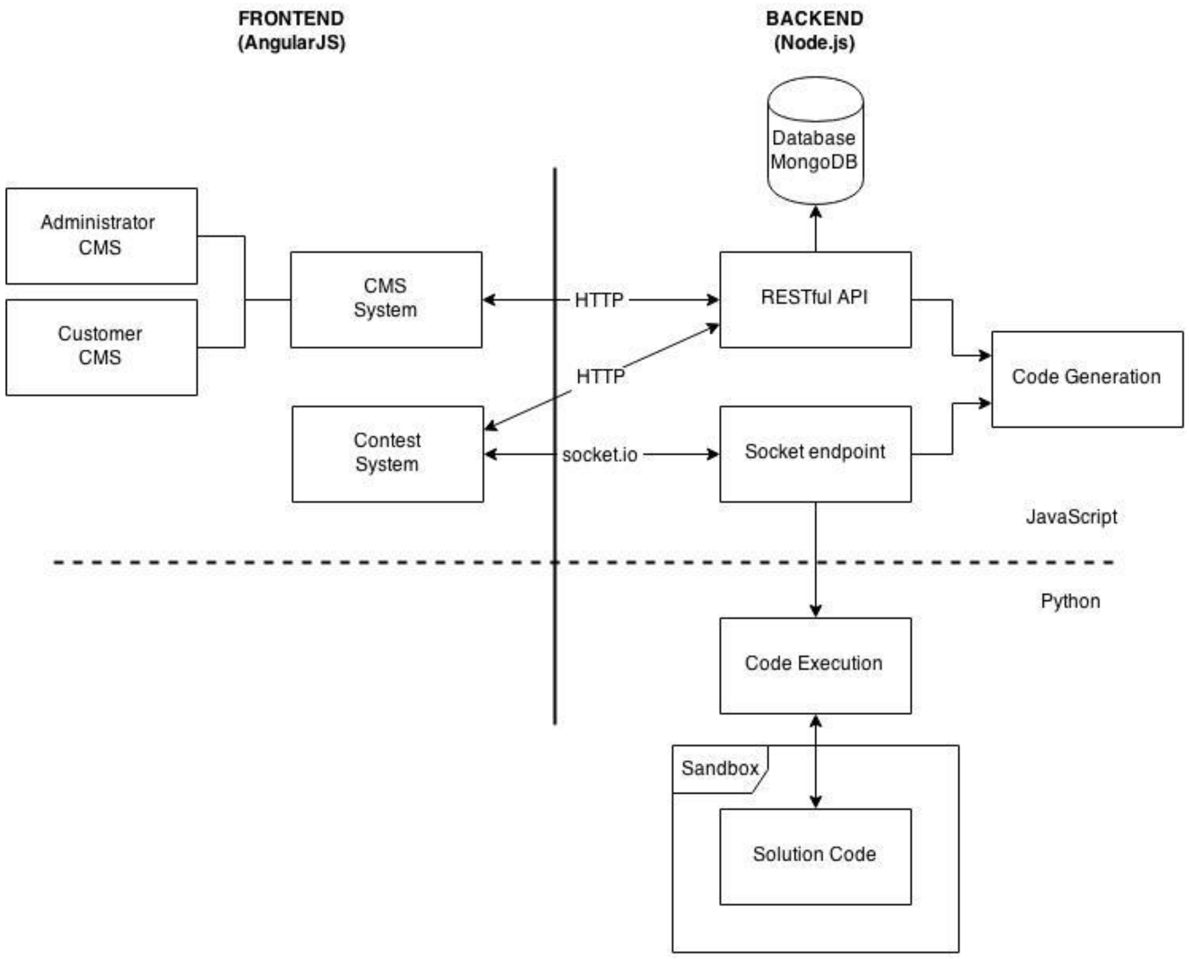


FIGURE 3 - ARCHITECTURE

The overall system architecture can be divided into three different parts; the AngularJS based frontend, the Node.js based web-application, and the Python based code execution subsystem. The frontend is divided into two separate systems, one handling the contests and one handling the content management systems. Communication with the backend is achieved utilizing HTTP and WebSockets. HTTP is used for communication with the RESTful API and sockets are used for evaluating solution code of contestants. How the three parts are connected on a higher level can be seen in figure 3.

5.2.1 Database

In dealing with a project as large as HoneyPot, with a very small timeframe to implement the application, it was concluded that seamless database integration into the JavaScript based backend would be a key factor to be able to deliver the application in time. For this reason, MongoDB was chosen as the database for the project.

MongoDB has a flexible schema that allows for variability between different documents in the same collection. That flexibility can be very powerful, as database schemas usually evolve over time. This way, one can easily sketch an ER-diagram before doing a requirements analysis over the whole system, only introducing the entities, relations, and fields that are relevant for the upcoming sprint. Then, after the ER draft is drawn, one can model or extend an existing model in mongoose without the need to perform migrations.

This process of editing and extending existing schemas and ER-diagrams after iterative requirement analysis was used during the full length of the project and resulted in the full-scale ER-diagram, figure 22, shown in Appendix A.

5.2.2 Modularity and logical separation of frontend

The frontend can be divided into two main parts; the contest system – the part that the contestants interact with – and the content management system – the part that the administrator and the customers interact with. These two systems can, in turn, be divided into even smaller and clearly separated modules. The contest system consists of a problem description, an API documentation, a code editor, a console, a canvas, and a separate thank-you page. The content management system (CMS) consists of two separate parts: the administrator CMS and the customer CMS. The administrator CMS can be divided into challenge management and customer management. The customer CMS can also be divided into multiple parts, one for customizing and branding a contest, and another for viewing information about the results and submissions for a specific contest.

Logical modules

The first step in making the application modular was to separate the project into logical modules. Using *angular.module* (see 3.14), the contest view's logical separation can be seen in figure 4, and the logical separation of the Content Management System can be seen in figure 5.

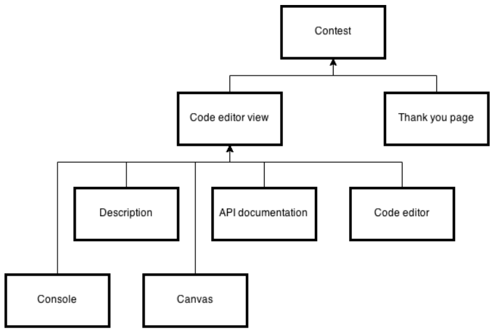


FIGURE 4 - LOGICAL SEPARATION OF CONTEST SYSTEM

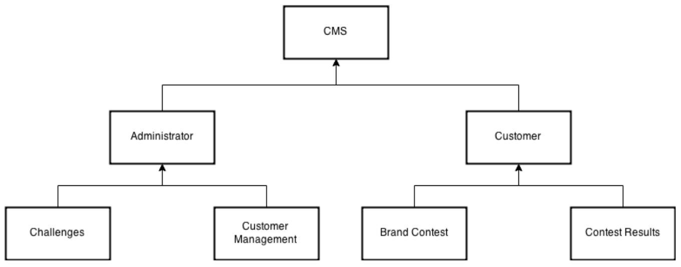


FIGURE 5 - LOGICAL SEPARATION OF CONTENT MANAGEMENT SYSTEM

5.3 Design

In designing the user interface, an effort was made to follow the guidelines, or “heuristics”, suggested by Jacob Nielsen [54]. This meant, for example, that efforts were made to remove all unnecessary information and keep the GUI clean, in order to achieve an “aesthetic and minimalist design”. Another example is that buttons have icons as well as text on them. This was done with the intention of decreasing memory load according to the principle of “recognition rather than recall”.

The choice of green as the main color for the application was inspired by research suggesting that it has positive influence on creativity [55]. This was considered fitting in order to create as conducive an environment as possible for contestant brilliance.

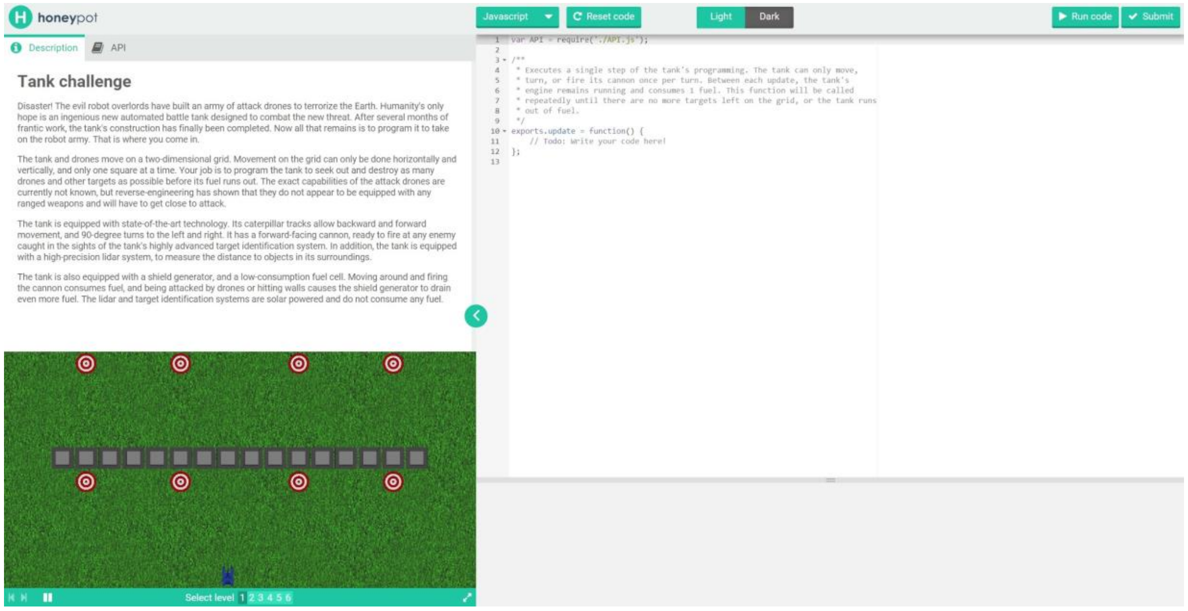
About 8% of men and 0.5% of women [56] are color blind. To accommodate for their needs considerations were made, among others that buttons change shape when hovered over instead of shifting color. Also there was a conscious effort to keep a high contrast between GUI elements.

5.4 Contestant experience

To get an initial understanding of what contestants will encounter during the process of participating in an active contest, a brief introduction, illustrated with images, will follow.

New contestants are initially met with a Landing Page, customized by the customer hosting the contest. On this page, contestants may find a description of what the contest might be about, and possibly what the customer might be looking for when recruiting.

On the following Contest Page (see figure 6), contestants are introduced to the main part of the application, where it is possible to write solutions for the described challenge. Two of the parts of the contest page are the challenge description and API, showcased in the top left of figure 6. The main part of the page on the right is the code editor and the console. This is where contestants may write their solution to the challenge in their chosen language. Also, in the bottom left, the contestants can see the graphical representation of a challenge-level. This is where graphical feedback of code execution is displayed after running. It is possible to get feedback on the solution from several levels, and their respective score, by clicking the *run* button in the top right corner.



Tank challenge

Disaster! The evil robot overlords have built an army of attack drones to terrorize the Earth. Humanity's only hope is an ingenious new automated battle tank designed to combat the new threat. After several months of frantic work, the tank's construction has finally been completed. Now all that remains is to program it to take on the robot army. That is where you come in.

The tank and drones move on a two-dimensional grid. Movement on the grid can only be done horizontally and vertically, and only one square at a time. Your job is to program the tank to seek out and destroy as many drones and other targets as possible before its fuel runs out. The exact capabilities of the attack drones are currently not known, but reverse-engineering has shown that they do not appear to be equipped with any ranged weapons and will have to get close to attack.

The tank is equipped with state-of-the-art technology. Its caterpillar tracks allow backward and forward movement, and 90-degree turns to the left and right. It has a forward-facing cannon, ready to fire at any enemy caught in the sights of the tank's highly advanced target identification system. In addition, the tank is equipped with a high-precision lidar system, to measure the distance to objects in its surroundings.

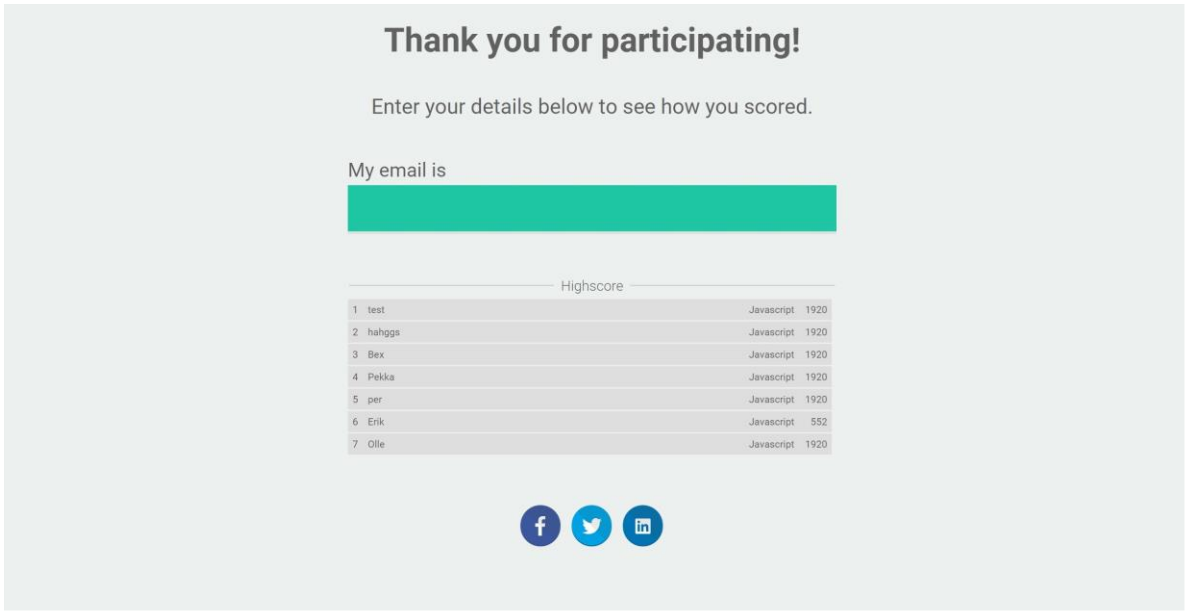
The tank is also equipped with a shield generator, and a low-consumption fuel cell. Moving around and firing the cannon consumes fuel, and being attacked by drones or hitting walls causes the shield generator to drain even more fuel. The lidar and target identification systems are solar powered and do not consume any fuel.

```javascript
var API = require('./API.js');

/**
 * Executes a single step of the tank's programming. The tank can only move,
 * turn, or fire its cannon once per turn. Between each update, the tank's
 * engine remains running and consumes 1 fuel. This function will be called
 * repeatedly until there are no more targets left on the grid, or the tank runs
 * out of fuel.
 */
exports.update = function() {
    // Todo: Write your code here!
};
```

FIGURE 6 - CONTEST PAGE

When a contestant is happy with her solution, she may submit it by clicking the *submit* button in the upper right corner, after which she will be taken to the Thank-You Page (see figure 7), where she may enter her personal details to find out how she scored.



Thank you for participating!

Enter your details below to see how you scored.

My email is

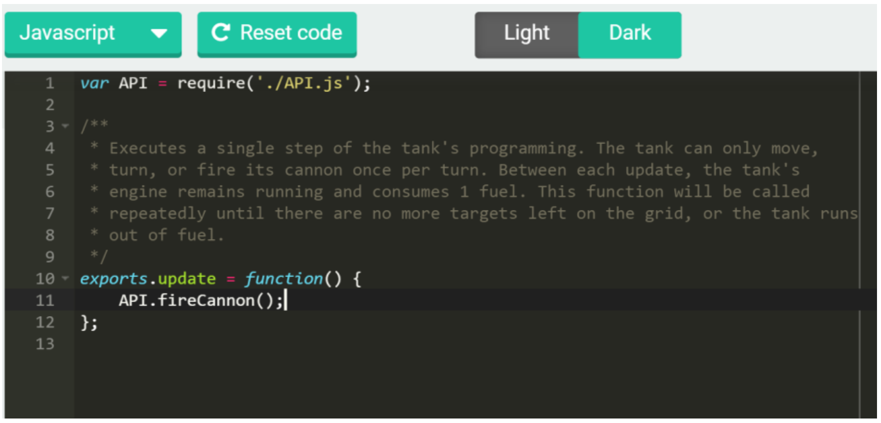
Highscore

1	test	Javascript	1920
2	hahggs	Javascript	1920
3	Bex	Javascript	1920
4	Pekka	Javascript	1920
5	per	Javascript	1920
6	Erik	Javascript	552
7	Olle	Javascript	1920

FIGURE 7 - THANK-YOU PAGE

5.4.1 Online code editor

For the online code editor, two options were considered; building one from scratch, or using an existing JavaScript online code editor. Building one from scratch was considered too large a project, and the editor that ended up being used for the project was Ace [57], mainly because of the extensive documentation available, and the active development and maintenance by Mozilla [58]. Ace is a very powerful online JavaScript code editor, with a large set of features that help in making the contestant experience a good one, the most important being syntax highlighting, automatic indent and outdent, and line wrapping, to name a few. Some of these features are showcased in figure 8 below.



```
var API = require('./API.js');

/**
 * Executes a single step of the tank's programming. The tank can only move,
 * turn, or fire its cannon once per turn. Between each update, the tank's
 * engine remains running and consumes 1 fuel. This function will be called
 * repeatedly until there are no more targets left on the grid, or the tank runs
 * out of fuel.
 */
exports.update = function() {
    API.fireCannon();
};

```

FIGURE 8 - CODE EDITOR

5.4.2 Persistence of solution code

Stored solution code needed to be fetched either from the database, or from the local storage of the contestant's browser. Storing the code locally seemed like the best option, since it would relieve the backend of requests, which may have otherwise required the backend to send large amounts of data. Furthermore, fetching the solution code from the backend would require open access to the solution code, if one just supplied a database solution ID. This would be a security threat, since other contestants might be able to fetch solutions via backend requests.

However, when implementing the local persistence, an issue was encountered; the size of local storage per domain may not exceed 5 MB [59], so there was a limit put on the number of solutions that could be saved. To solve this issue, the amount of code a contestant can write in the editor was limited. Also, the number of solutions for different contests was limited to at most 20 solutions; if the number exceeded 20, the 3 oldest solutions are removed. An implication of this is that it is not possible to solve 21 contests and have all the solutions saved. It was concluded that the limit was reasonable, since it was considered unlikely that a user with 21 unfinished solutions for contests would return to the 3 oldest and expect that the old unsubmitted code to still be there. Instead, this hypothetical user would have to start over on those contests.

5.4.3 Support for multiple languages and high-level API

Syntax highlighting for the supported programming languages is provided by the Ace editor. Initially, the skeleton code for all languages is fetched from the database, and then all updates to the code are stored locally in the browser for each separate language. When a contestant switches language, the contents of the code editor are replaced with the locally stored code for that language.

When a solution is to be executed, the code is sent to the server and saved within a temporary directory. In addition to the code written by the contestant, libraries and files written in the same language are copied to the same directory. Together, all the files copied into the directory form of a program which can be executed and communicated with using pipes, which were determined to be the simplest way of performing the necessary interprocess communication with the solution programs. By sending commands formatted as JSON to standard in, functions of the solution code can be called, and their results returned as JSON on standard out. Likewise, one of the files copied alongside the solution acts as the API, forwarding function calls from the solution code to the challenge API over standard out in the same manner as functions of the solution code are called over standard in.

5.4.4 Running code

When a contestant wants to see the result of her solution she clicks the “Run code” button, whereupon the currently written code is sent to the backend. The code is then run once per level (test case) of the current challenge, and the score is calculated individually for each level and sent back together with the graphical representation of how the code behaved. If the running of the code would result in an exception or error of any kind, this is included in the result returned to the frontend, and is displayed within the console part of the web application. The flow of information after a contestant clicks the “Run code” button can be seen in figure 9.

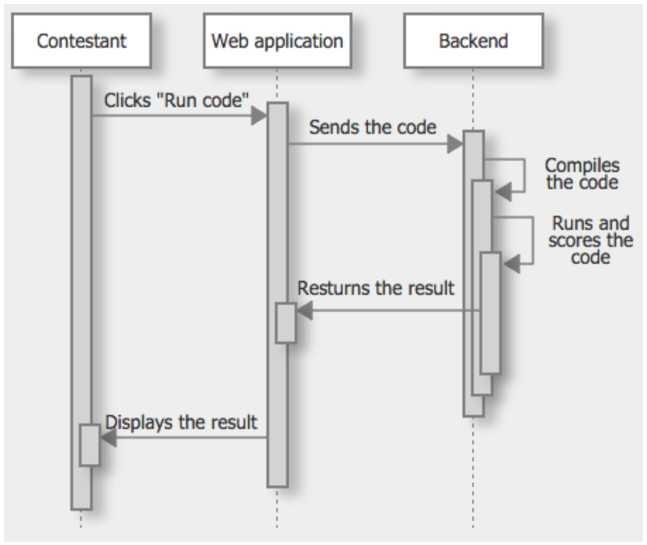


FIGURE 9 - RUNNING OF SOLUTION CODE

5.4.5 Quick feedback

In order to achieve near-instant feedback of running code in the code editor view, the code needed to be delivered to the backend for compiling and running in a fast and stable way, supporting two way communication. For this reason, network sockets were used, along with HTTP, for communication between frontend and backend. When the code view is loaded, a socket connection is established between the frontend and the backend, and the frontend is assigned a unique identification number. Once the connection is open, both ends are able to send information to the other without the need for a request. Handling code execution and evaluation this way makes for totally asynchronous communication. This way, the frontend resources are not made unavailable by waiting for a response from the backend. When information has been sent from the frontend to the backend, the frontend is able to perform its tasks as normal. The backend is then able to send information upon completion, whereupon an action is performed on the frontend side.

5.4.6 Graphical visualization

In order to be able to provide a graphical visualization to the contestant of how the solution performed, a *canvas* element is included in the web application. A canvas element is a new kind of HTML element, introduced in HTML5, which, using JavaScript [60], makes it possible to make drawings and animations on a web page. Writing JavaScript for animations by hand can be difficult, so for this project, a JavaScript framework specially designed for animating objects on a canvas element was used. The framework used was CreateJS [61], which provides a full suite of libraries for animating [62] and pre-loading assets used in the animations [63].

Using CreateJS and the graphical information in the result obtained from running a contestant's code, it is possible to animate the solution using 2D graphics. The graphical information in the result consists of a set of steps for every level, where a step consists only of the changes made to a set of graphical objects.

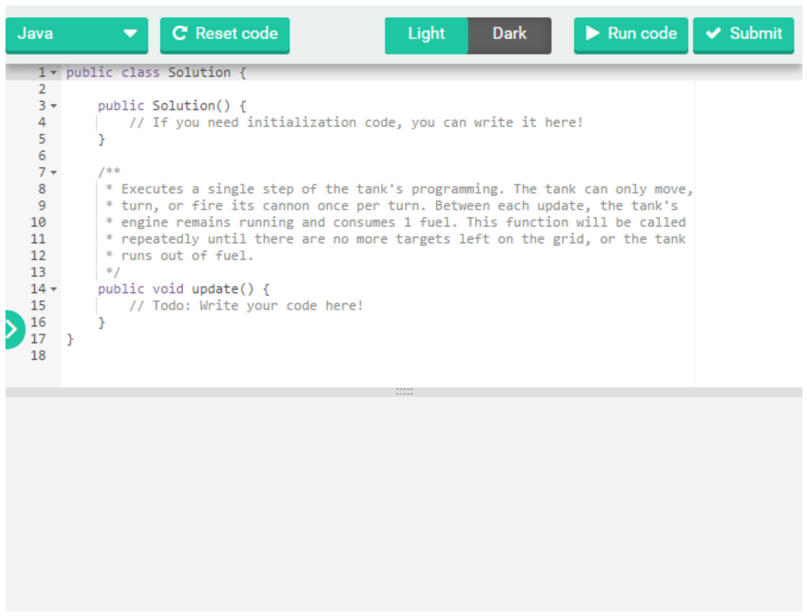
5.4.7 Recording of code solution

As the recording of the code written by a contestant is a part of the code playback feature of the system, it was important that the records of the code were only changes made to it, and that all changes were included. Conveniently, Ace fires a Javascript event every time the contents are modified [64], and provides information about what was changed. Using this event, the web application is able to record all changes made to the code and send information about these to the backend for storing. More information about how the information about the changes is used can be found in section 5.5.2.

Sending a HTTP POST request on each keystroke, however, would put unnecessary load on the server. Thus, it was decided that recorded edits should be sent in batches of 30 edits. The edits are cached until 30 edits are recorded, and then the whole batch of edits is sent. To make sure that no edits are left in the cache upon submission, all remaining edits are sent before moving on to the submission page (also referred to as the Thank-You Page, see 5.4).

5.4.8 Responsive web design

Given that the audience for this project are mainly programmers, the assumption was made that the users of the application would interact with it using modern web browsers. Under this assumption, the design of the web application was made responsive by making use of the new web technologies *flexible box* and the animations functionality in CSS3. These new technologies made it possible for the application to respond to all resolutions down to a width of about 980 pixels, with all features of the view clearly visible and easily accessible, as seen in figure 6. When the left panel, containing the canvas, challenge description and API documentation, is folded away, the minimum width were the view still looks good is lowered to about 670 pixels, as seen in figure 10. The main reason that this can not be any lower is because of the critical elements that the view must contain to fulfill its purpose, such as the buttons for changing the programming language and running the code.



```
public class Solution {

    public Solution() {
        // If you need initialization code, you can write it here!
    }

    /**
     * Executes a single step of the tank's programming. The tank can only move,
     * turn, or fire its cannon once per turn. Between each update, the tank's
     * engine remains running and consumes 1 fuel. This function will be called
     * repeatedly until there are no more targets left on the grid, or the tank
     * runs out of fuel.
     */
    public void update() {
        // Todo: Write your code here!
    }
}
```

FIGURE 10 - FOLDED CONTEST PAGE

In order to make sure that the design of the web application look as intended in as many web browsers as possible, and to help ease the development of the application, an auto-prefixer was used to help with the necessary vendor prefixes. In this project, the auto-prefixer used was Autoprefixer – the most commonly used auto-prefixer, which is recommended by Google [65, 66]. It adds the necessary vendor prefixes using data from the website Can I Use [67], a site which provides up-to-date data on web browser support for front-end web technologies.

5.5 Customer experience

The customer CMS is divided into three tabs: challenges, users, and settings. The settings tab is only used for changing password. Challenges is the home tab, which lists the challenges that the customer has access to, and those that the customer may request access to. In figure 11 below, the customer Software Skills is shown to have four unlocked challenges, with the option to request the Traveling Salesman challenge.

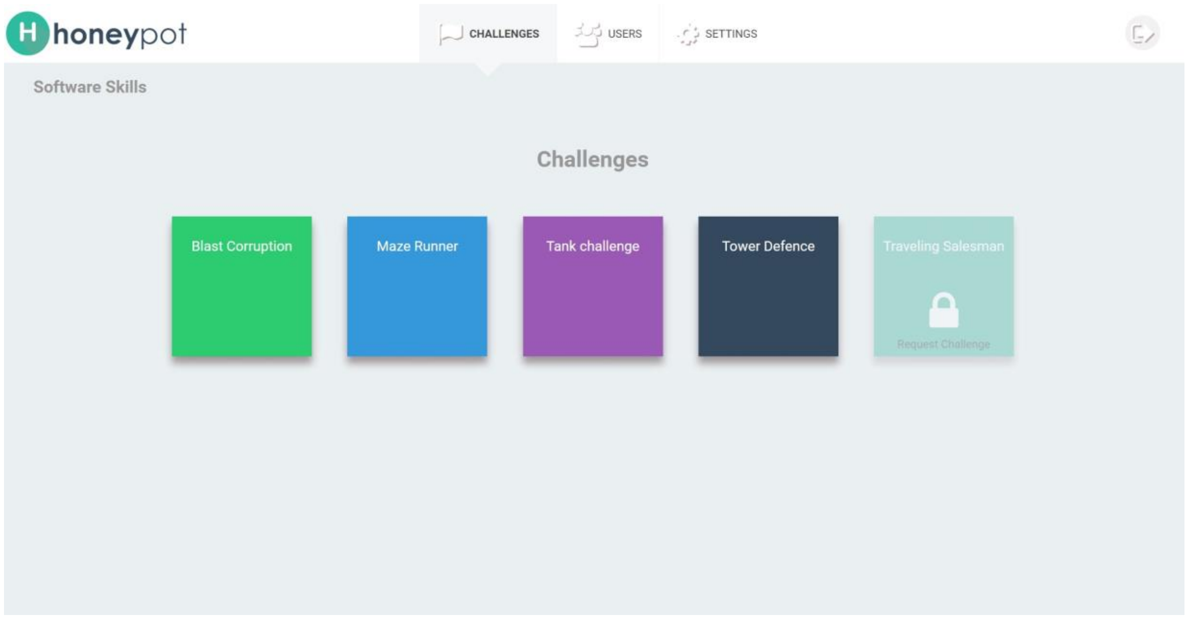


FIGURE 11 - CUSTOMER CHALLENGES

When clicking on a challenge that the customer has access to, a grid of customer-created contests for that particular challenge is shown (see figure 12). These contests can be visited, customized and one can display results of the contest contestants. Here, it is also possible to initiate new contests.

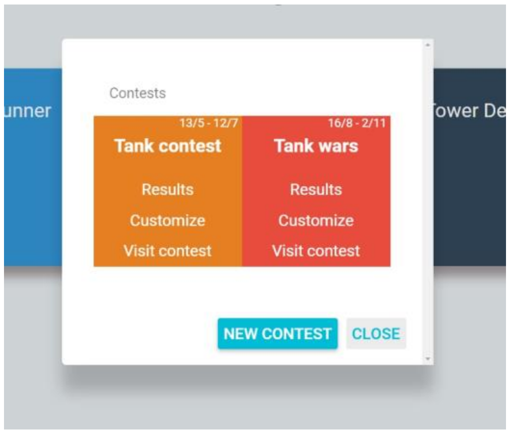
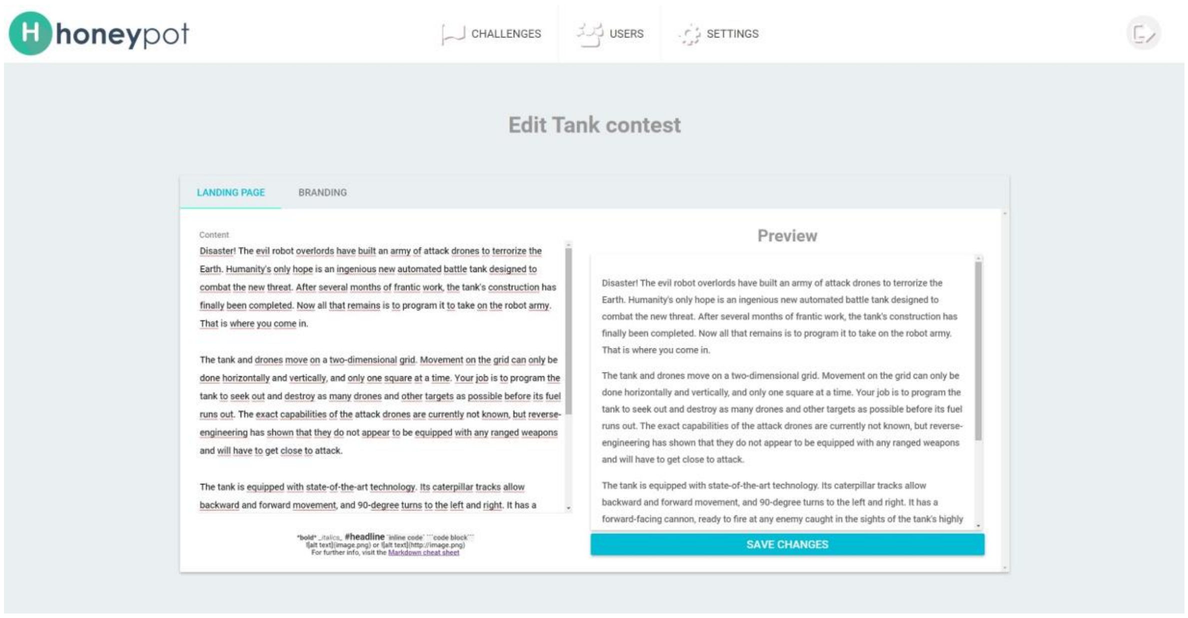


FIGURE 12 - CUSTOMER CONTESTS OF A CHALLENGE

If the customer chooses to customize a contest, she is taken to the page shown in figure 13. In this view, the customer may specify a description and a background image for the contest Landing Page.



Edit Tank contest

Content

Disaster! The evil robot overlords have built an army of attack drones to terrorize the Earth. Humanity's only hope is an ingenious new automated battle tank designed to combat the new threat. After several months of frantic work, the tank's construction has finally been completed. Now all that remains is to program it to take on the robot army. That is where you come in.

The tank and drones move on a two-dimensional grid. Movement on the grid can only be done horizontally and vertically, and only one square at a time. Your job is to program the tank to seek out and destroy as many drones and other targets as possible before its fuel runs out. The exact capabilities of the attack drones are currently not known, but reverse-engineering has shown that they do not appear to be equipped with any ranged weapons and will have to get close to attack.

The tank is equipped with state-of-the-art technology. Its caterpillar tracks allow backward and forward movement, and 90-degree turns to the left and right. It has a

bold _italics_ #headline \`inline code\` \`\`\`code block\`\`\`
![alt text](image.png) or ![alt text](http://image.png)
For further info, visit the Markdown cheat sheet

Preview

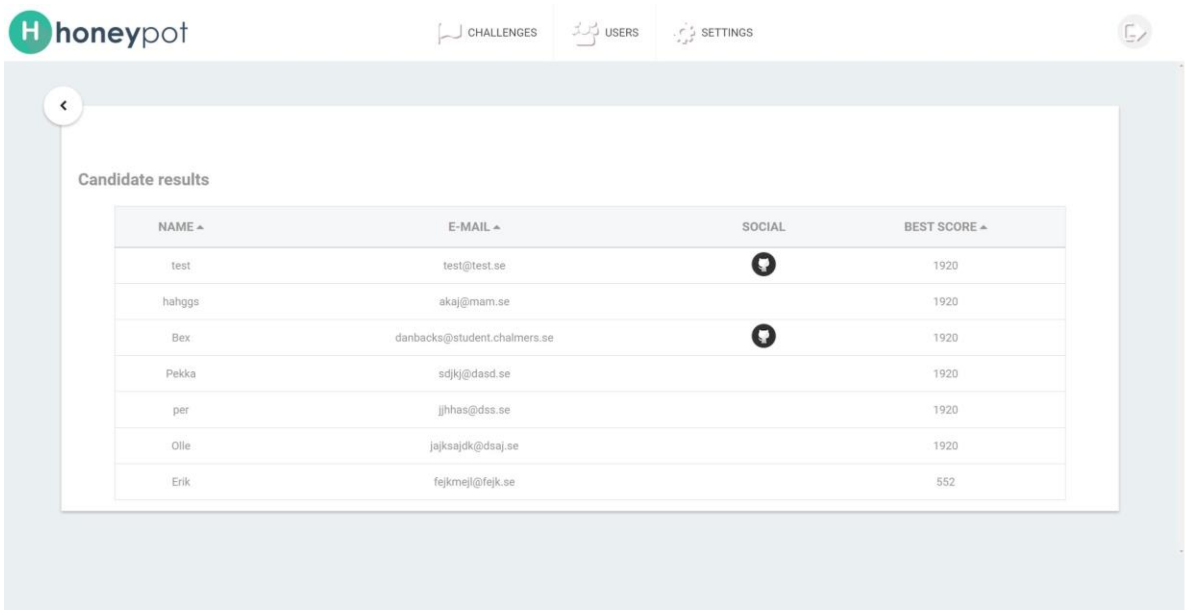
Disaster! The evil robot overlords have built an army of attack drones to terrorize the Earth. Humanity's only hope is an ingenious new automated battle tank designed to combat the new threat. After several months of frantic work, the tank's construction has finally been completed. Now all that remains is to program it to take on the robot army. That is where you come in.

The tank and drones move on a two-dimensional grid. Movement on the grid can only be done horizontally and vertically, and only one square at a time. Your job is to program the tank to seek out and destroy as many drones and other targets as possible before its fuel runs out. The exact capabilities of the attack drones are currently not known, but reverse-engineering has shown that they do not appear to be equipped with any ranged weapons and will have to get close to attack.

The tank is equipped with state-of-the-art technology. Its caterpillar tracks allow backward and forward movement, and 90-degree turns to the left and right. It has a
forward-facing cannon, ready to fire at any enemy caught in the sights of the tank's highly

FIGURE 13 - CUSTOMIZING CONTEST LANDING PAGE

If the customer instead chooses to view the results of the contest, the customer is taken to a page listing the contestants that have submitted solutions, along with their highest score, as seen in figure 14 below. Here, if a Github account has been supplied on submission, it is also possible review the contestant's Github activity.

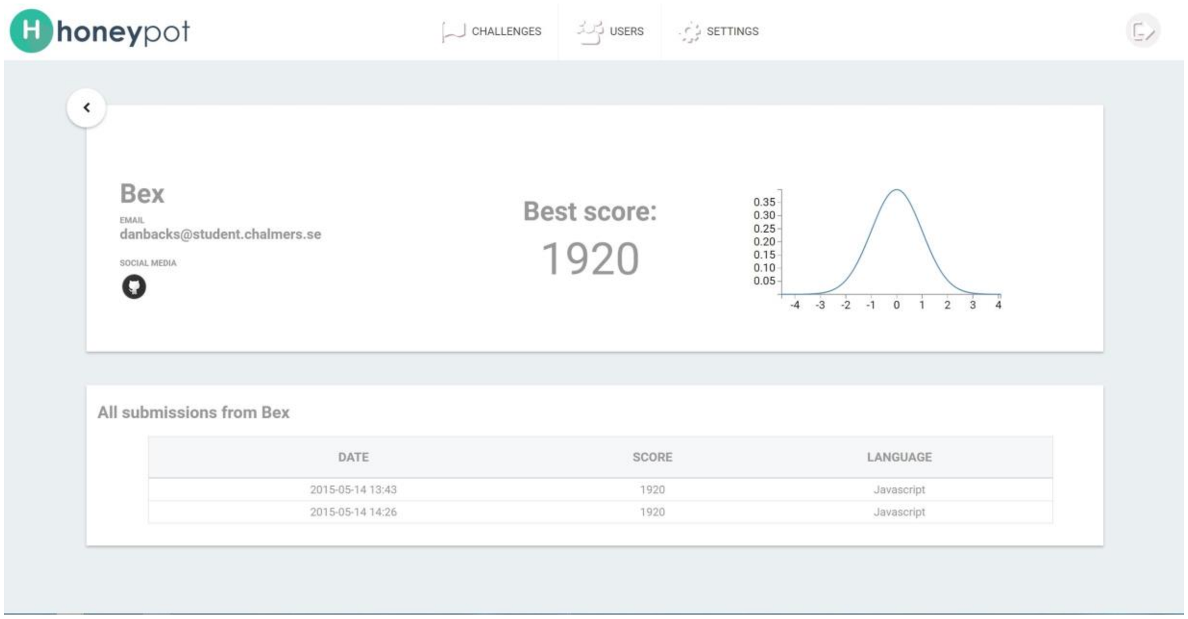


Candidate results

NAME ▲	E-MAIL ▲	SOCIAL	BEST SCORE ▲
test	test@test.se	●	1920
hahggs	akaj@mam.se		1920
Bex	danbacks@student.chalmers.se	●	1920
Pekka	sdjkj@dasd.se		1920
per	jjhhas@dss.se		1920
Olle	jajksajdk@dsaj.se		1920
Erik	fejkmejl@fejk.se		552

FIGURE 14 - RESULTS OF CONTESTANTS

Clicking on a contestant row takes the customer to a more detailed page of the contestant's submissions for the contest, as seen in figure 15 below. Here, all submissions made by the customer may be reviewed, along with information about the submission date, final score, and the programming language used.

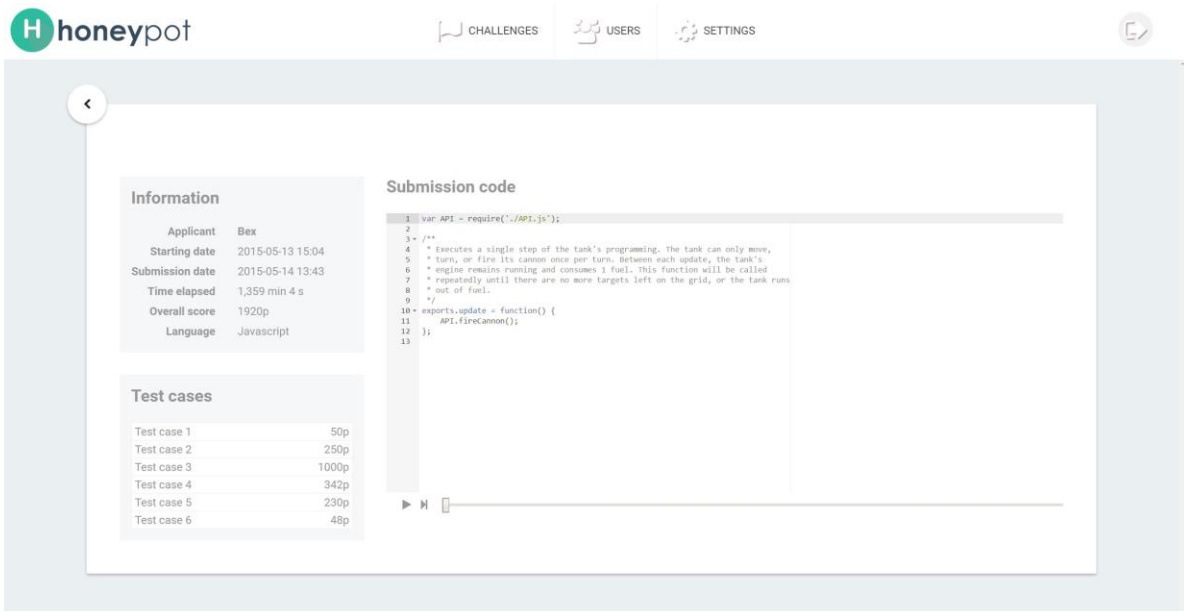


All submissions from Bex

DATE	SCORE	LANGUAGE
2015-05-14 13:43	1920	Javascript
2015-05-14 14:26	1920	Javascript

FIGURE 15 - CONTESTANT DETAILS AND SUBMISSIONS

The customer can then choose to review a specific submission to get more detailed information about it, as seen in figure 16. Here, the customer can review how long time the contestant needed to solve the challenge, and how the contestant scored on each test case. Also, as can be seen in the center-right of the figure, it is possible to play back the candidate's coding-process for the specific solution that was submitted.



Information

Applicant	Bex
Starting date	2015-05-13 15:04
Submission date	2015-05-14 13:43
Time elapsed	1,359 min 4 s
Overall score	1920p
Language	Javascript

Test cases

Test case 1	50p
Test case 2	250p
Test case 3	1000p
Test case 4	342p
Test case 5	230p
Test case 6	48p

Submission code

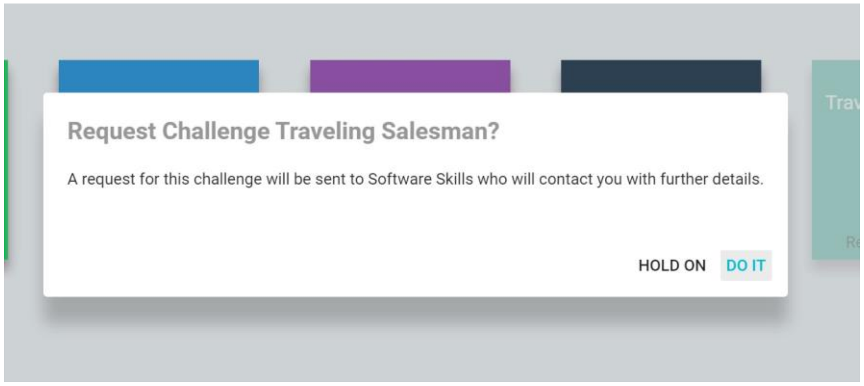
```
var API = require('./API.js');

/**
 * Executes a single step of the tank's programming. The tank can only move,
 * turn, or fire its cannon once per turn. Between each update, the tank's
 * engine remains running and consumes 1 fuel. This function will be called
 * repeatedly until there are no more targets left on the grid, or the tank runs
 * out of fuel.
 */
exports.update = function() {
    API.fireCannon();
};
```

FIGURE 16 - SUBMISSION DETAILS

5.5.1 Requesting challenges and initiating contests

Requesting challenges is simple and just one click away for a customer, as seen in figure 17. This feature was implemented utilizing *Nodemailer*[68] in the backend to automatically send request emails to the client. An administrator can then, likely after payment, unlock access to the requested challenge for the customer (see figure 21). The customer is then allowed to initiate contests with the unlocked challenge.

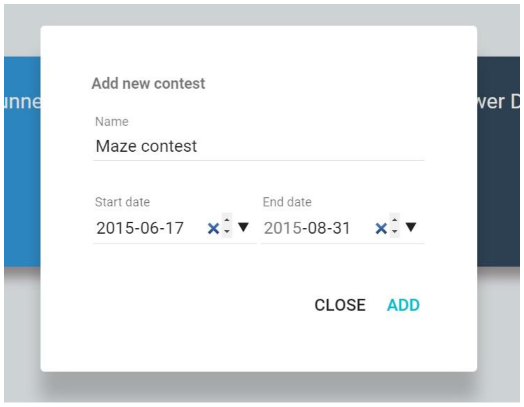


Request Challenge Traveling Salesman?

A request for this challenge will be sent to Software Skills who will contact you with further details.

FIGURE 17 - REQUEST CHALLENGE

Initiating a contest is a simple two-step process. First, the customer specifies contest information such name and start and end dates, as seen in figure 18. Second, the customer may customize the contest Landing Page. The features are implemented using AngularJS services, communicating with the backend using RESTful API routes.



Maze contest

2015-06-17

2015-08-31

FIGURE 18 - INITIATE CONTEST

5.5.2 Evaluation of contestants and playback of code solutions

The final code submission from a contestant is tested, in parallel, on all of the levels of the challenge taken, and the results on these levels are stored in the database, together with the code itself and other details about the submission, such as the date, time spent writing the code, and details about the contestant.

Using the information stored in the database about the changes made to the code, it is possible to build the code from the initial state to the final one, by applying the changes one by one. This is made possible by using an instance of the Ace editor in the solution details view as well, as the editor has functions for applying all of the corresponding changes made to the document that cause its *change event* to trigger. When clicking play in the code playback, a code change is applied every 50 milliseconds, and the result is akin to watching a video recording of how the contestant wrote the code, with playback controls and a timeline, as can be seen in figure 16.

5.6 Administrator experience

The administrator CMS has three main tabs: challenges, customers and settings. The settings tab is only used to change the password of the account. The challenges tab provides the administrator with a grid of the created challenges, seen in figure 19. In the challenges tab, challenges can be published for all customers to request and use in their respective CMS. It is also possible to edit existing challenges, and to create new ones.

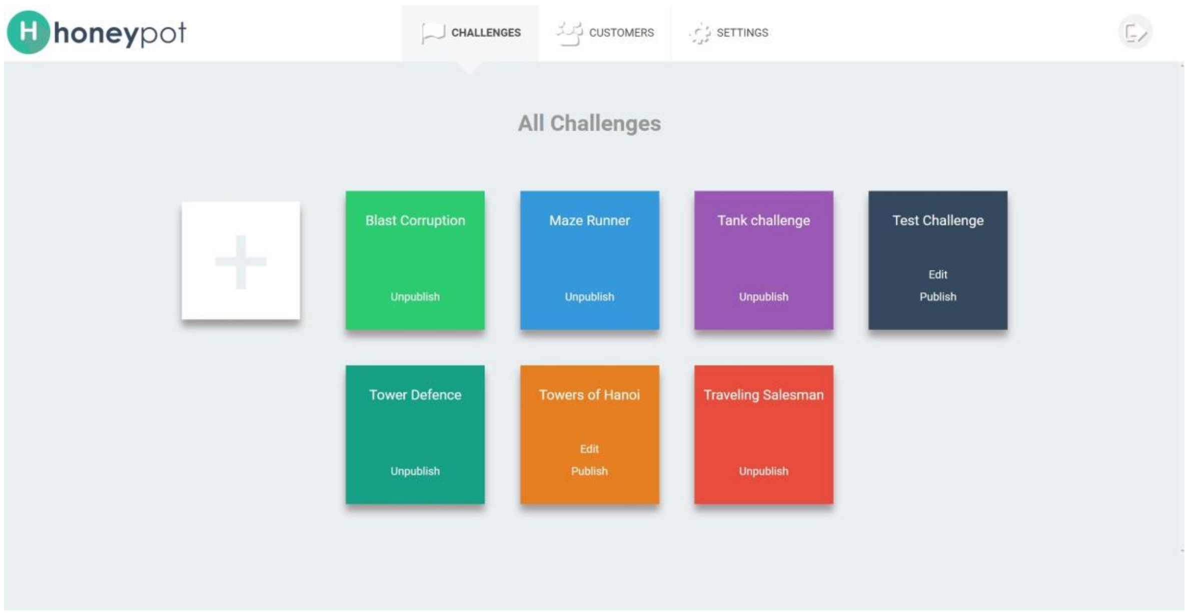


FIGURE 19 - ADMINISTRATOR CHALLENGES

Editing or creating a challenge takes the administrator to a page with multiple tabs, used to describe, specify, and implement a challenge, as seen in figure 20. On this page, the administrator can specify the challenge description and its API, upload images and sprites to be used for graphical visualization, and implement the challenge and its level specifications (test cases).

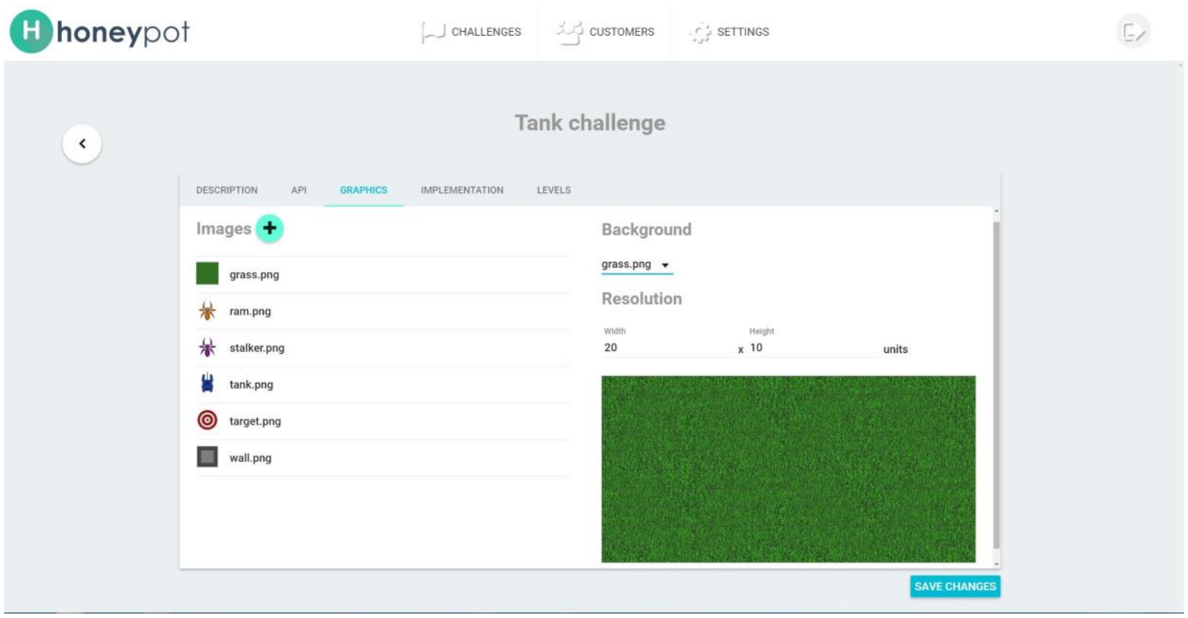


FIGURE 20 - EDITING CHALLENGE

In the customers tab, the admin has the ability to view customers and their unlocked and locked challenges, as seen in figure 21. Here, administrator can unlock access to a challenge that a customer has requested. The administrator can also add, edit, and remove customers.

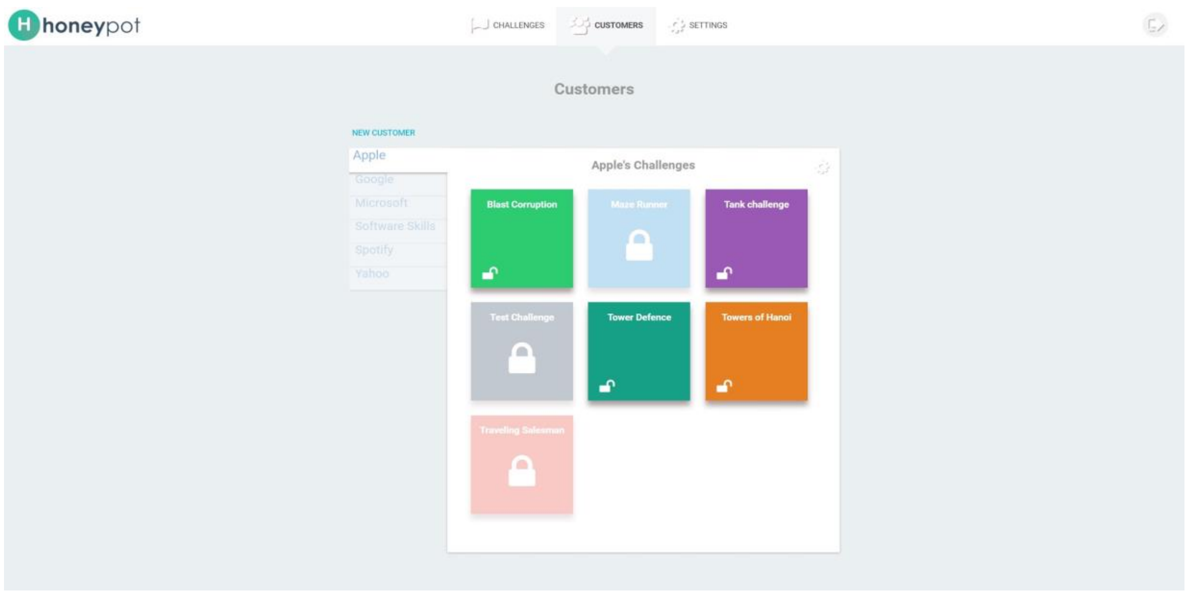


FIGURE 21 - CUSTOMERS

5.6.1 Creation and editing of challenges

The key to allowing an administrator to quickly create new challenges, is by doing most of the work for supporting multiple languages in the background. The implementer is required to specify the API of the challenge before writing any implementation code. By having an API specification, the necessary code for communicating with contestants' solutions can be automatically generated based on this specification, reducing the amount of work required by the implementer. The Ace editor is used to implement challenges and specify levels (test cases). Changes to the API are appended to the end of the implementation file.

5.7 Scalability

By placing the MVC structure on the AngularJS frontend, the logic and processing on the backend was reduced to just handle authentication, session management, the RESTful API (see 3.1.3), database communication, code generation, and code evaluation. With the use of Node.js (see 3.1.2) and its event-loop with asynchronous non-blocking I/O combined with asynchronous socket-based code execution for evaluation of solutions, the system was developed in an event-driven, non-blocking manner. This allows for large numbers of requests to be processed simultaneously, even on a low performance server. The backend is completely stateless, hence the memory consumption of the server does not rise over time. It only increases by the amount of requests currently being processed. The evaluation of levels (test cases) of a contestant's solution runs in parallel, with compilation happening at most once per solution, which increases performance and reduces the processing on the backend.

5.8 Stability

When working with the stability of the platform, there were two main areas of concern: the platform had to remain stable under a lot of pressure, and the backend could not be allowed to crash. Several steps were taken in order to make the sure that the platform would remain stable under high load. Concerning the compilation and execution of the contestants' code, a limit was placed on the maximum number of solutions running simultaneously, effectively securing system resources for the main system, in collaboration with some of the measures described in section 5.9. This, however, was not enough to make sure that the system would be stable enough for production use. Measures were also put in place to ensure that the same contestant could not send multiple simultaneous code-running requests to the backend. The button to run the code from the web application is disabled when a request is sent, and remains so until the response arrives. Steps have also been taken in the backend to make sure that this cannot happen: if the backend detects a code-running request from a socket connection that already has a code running-request in queue, the new request is simply terminated.

Another thing that needed to be considered when designing the RESTful API (see 3.1.3) of the backend to be stable was to ensure that simultaneous requests that may change the database were implemented in a concurrently safe manner. This means that updates of the database have to happen atomically, without any other database request interleaving, for example, a query or an object update. If an update of the object document, which effectively increases the version number, happens after the query but before the save, MongoDB will throw a version error and crash the backend, since the expected version does not match the actual. Because of this, the project is implemented using special MongoDB *findAndModify* queries, which are atomic to prevent these crashes [69].

In order to make sure that the platform does not crash without restarting, the backend is started using the Node.js module PM2 [70]. With PM2, the node process is monitored and restarted if it crashes. Also, PM2 is configured so that the node process is started automatically as a service on server restart. PM2 allows easy monitoring of the node process, providing information about things such as memory consumption, uptime, restarts and CPU usage. If a crash occurs, the error log will be saved, allowing the reasons behind the crash to be reviewed and the issue to be solved in a later patch.

5.9 Security

5.9.1 Authentication and session handling

To authenticate user sessions, Honeypot uses session-cookies handled by an AngularJS service. The cookie data is sent with each HTTP request to the backend where it is processed by the express-session middleware [71], which uses connect-mongo [72] session storage to store and validate user sessions. This allows the routes of the RESTful API to check for authentication of different types of user accounts and thereby invalidate requests lacking the required privileges. Access to restricted data is hence an administrator-only feature, and the customer users only have access to data related to the customer in question.

In addition, publicly accessible API routes are strictly limited to the routes used in the contest application, only exposing data that is considered non-sensitive. Therefore, it is, for example, not possible for contestants to make backend requests and get other contestants submitted code.

To protect password data of users, bcrypt [73] is adopted for password storage. Bcrypt is a key derivation function for passwords utilizing the Blowfish cipher [74]. This ensures that possibly leaked user data will have a strongly encrypted password field.

5.9.2 Code execution

Since contestants are permitted to submit arbitrary code for execution, it is important that this code is prevented from performing any harmful actions. This is achieved by executing all code written by contestants within a sandbox. Because all contestant solutions are run on the same system, need to be run with low overhead, and may be written in different programming languages, the sandbox is built using the built-in security features of the operating system. The sandbox is designed to protect against illegal access to files or resources, and to limit the potential damage of a denial-of-service attack.

AppArmor
The main defense against contestants trying to illegally access files or resources, is AppArmor, a Linux Security Module for mandatory access control [75]. While access control in Linux is normally based on user permissions, AppArmor adds an additional layer of security by association permissions with specific programs. Furthermore, when set to enforce a security profile for a program, AppArmor's default behavior is to deny access to any file or resource not explicitly specified as permitted within the profile.

Unprivileged user
While the primary defense against illegal access is provided by AppArmor, there may still be flaws within the profiles used to execute sandboxed code. To limit the potential damage where such a flaw is present, contestants' code is run by a special user created for that specific purpose. Beyond running such code, the user has no other special privileges on the system.

Resource limits
To defend against denial-of-service attacks carried out by contestant submissions, two different operating system mechanisms are used. The *setrlimit* [76] system call is used to impose limits on the maximum amount of CPU time a solution process is allowed to use, and to put an upper limit on the number of processes the user is allowed to run simultaneously. In addition, *cgroups* are used to limit the maximum memory and swap available to solution processes, as well as set a maximum limit on CPU shares.

Chapter 6: Result and discussion

The purpose of the project has been to investigate the feasibility of and develop a user-friendly platform for creating and hosting programming contests as a recruitment aid for software companies. The project team has made pre-studies in form of competitor analysis and research suggesting the best programming languages to support in the application. The goal of the product was stated early on in the process, but features were continuously added and removed following discussions with the client.

The project resulted in a production-ready application, which will be a part of Software Skills' product range as of June 2015. This was the goal all along, even though the vision of the product continuously evolved throughout the course of the project.

6.1 Evaluation and usability

The contest system was well received
The results from the contest system tests were positive. All participants completed all tasks with none taking unexpectedly long, and no errors were encountered. The graphical interface and its manipulation was considered intuitive, aesthetically pleasing, and professional. Some feedback regarding potential improvements was received. For example, one participant would have liked to have a *close-all* function for the API tabs, which was implemented in the subsequent sprint.

CMS views testing was left out
As previously stated (see 4.3.2), testing of the customer and admin views was down-prioritized and consequently left out of the project. This was partly due to receiving very positive feedback regarding the views from the client and partly because of time constraints.

Usability was found to be of good standards
With a result base of 24 participants, the total score of the SUS was 82. When comparing this to the levels proposed by Bangor et al (see 3.3.2), it would earn the project a place above "passable", among the "better products". This could be interpreted as the product having good usability standards, while still needing development before being able to be classed as a top product.

6.2 Process

Within Scrum, there are several central concepts that are considered necessary in order to get the full potential out of the framework. However, during the development process for Honeypot, only a handful of these concepts were deemed necessary in order to complete the project in a successful manner. The largest contribution from Scrum that was utilized were the concepts of using a backlog, sprints, and the planning this involves, as well as the roles of Scrum Master, Product Owner and Developer. Early on, it was determined by the development group that concepts such as Sprint Reviews would generate much value to the group in relation to the time it would require to implement them. Thanks to good communication tools such as Slack, it was easy for the group to reflect on and discuss its work continuously throughout the project.

The project role Scrum Master was not considered necessary after the first couple of sprints, and was therefore excluded from the remainder of the process. Also, the developer role changed over time. It was first said that every team member would work with all parts of the application in order to both learn as much as possible and avoid scenarios where the group is dependent on the presence of a single person. Towards the later part of the project, however, every team member ended up having a personal area of expertise to focus on. This was not considered a problem since it resulted in faster progress and shorter product iterations.

The process structure, in the form of themes, made the process clear and agile. Iteratively extending the system worked well, and made the team focus on delivering finished parts of the system, efficiently managing to build a large, competitive application.

Using design studios for creating mockups of the design of all the platform’s parts worked very well in this project. It boosted the team’s creativity, and resulted in a design incorporating the best parts of multiple mockups, and contributed to all members of the team being on the same page regarding what the final product would look like.

6.3 Quality of the application

Thanks to a clear project process in combination with good communication between the project members, the result is a well-implemented final product. Every major decision regarding the architecture was thoroughly discussed in the project group before its implementation. The process also provided structured release cycles, since the result of each sprint was considered a working instance of the final product. Therefore, it was relatively easy to foresee what the project team needed to focus on in future sprints in order to finish the project in time.

The user tests (see figure 25 in appendix A) showed that a considerable number of users found the graphical user interface appealing and easy to use. Some test participants, however, did have valuable input as to possible improvements of the interface. They, for example, missed a way of resetting the code editor once they had started writing a solution. A button to fill this need was therefore added. Due to lack of time, it was not possible to take all input that was received from the user testing into consideration. The product had already reached a satisfactory level for both the project group and the client. It is, however, worth mentioning that none of the project members had any comprehensive experience with interaction design. This is a competence that might have yielded benefits in terms of product quality.

The user tests did however find some critical issues regarding the sandboxing of the product, which was of high priority for the project group to address and correct. It was for this very reason that the user tests were carried out; when the minimum viable product had been implemented, it also needed to be used by the intended user base to ensure that it would function properly.

Chapter 7: Conclusion

The purpose of this project has been to create a framework for easy creation of web-based code challenges. The work has resulted in a fully functional product, which has been sold to the client. The code base has been passed on to the client, together with thorough documentation of how the different parts of the application work, and how support for new languages can be added to the existing platform. This result has met, and exceeded, the original expectations of the client, and shows that developing a user-friendly platform for creating and hosting programming contests, and using it as a recruitment aid for software companies, is indeed a very real possibility.

7.1 Future work

There are several features to consider regarding the future development and maintainability of the project. The number of languages supported for writing contest solutions should likely be increased, to make sure that potential contestants are not lost because they are unable to write code in their favorite language. This would increase the competitiveness of the application in the market.

The application and the client would likely benefit from a payment subsystem. This would open up the possibility for customers to more directly get access to challenges without Software Skills having to manually handle payments and access. A dedicated page for the Honeypot application may also be added to increase sales and provide links to customers and their contests. This would open up possibilities for advertising revenues.

Another feature which might be considered is a way to preview challenges when initiating contests as a customer or when editing challenges as an administrator. This would make it easier to create challenges, and give the customer a taste of the challenge they may want to use.

7.2 Critical analysis

It was said in the beginning of the project that the development process would be performed using a test-driven approach. Unit tests would be written for every part of the application, preferably even before its development had started. This would have made the development process structured and clear, in addition to ensuring that the code would hold a high quality and preventing bugs. It was, however, determined that this method could not be fully implemented, since there were too many unknown factors in the product. Using this test-driven method would therefore have prevented the project group from working in a sufficiently high pace; instead, a fast, iterative approach was implemented.

It is interesting to speculate how the process would have looked if the project group had assigned one or two members for working with quality assurance only. The project would, in that case, have had more test cases to work with, and it is possible that bugs would have been found in greater numbers and earlier in the process. The intention with this project was, however, partly to give every team member the opportunity to learn as much as possible about the technologies used; a quality assurance group would have been a contradiction to this. For further work with the Honeypot product, it could be argued that the project structure would need to be restructured since the code base is relatively big. The project group, however, wishes to argue that the process that has been implemented has been one of the key factors to achieving the successful end result, which suggests that the process was well-suited to the task.

References

[1] “Timeline of programming languages”, Internet: http://en.wikipedia.org/wiki/Timeline_of_programming_languages, [visited 18 May 2015]

[2] Intel, “Jobs at Intel - Hiring Process and Tips”, Internet: http://www.intel.com/content/www/us/en/jobs/hiring.html, [visited 13 Feb 2015]

[3] Microsoft, “Application process”, Internet: http://careers.microsoft.com/careers/en/gb/gradapplicationprocess.aspx, [visited 13 Feb 2015]

[4] D. Reynolds, “The Best Way to Hire Software Developers”, Internet: http://news.dice.com/2013/05/31/hiring-software-developers/, May. 2013 [visited 13 Feb 2015]

[5] H. Singer, “In Search of the Elite Few – Finding and Hiring the Best Software Developers in the Industry” Internet: http://www.toptal.com/freelance/ in-search-of-the-elite-few-finding-and-hiring-the-best-developers-in-the-industry, February 2014, [visited 13 Feb 2015]

[6] M. Niehaus, “How can you hire top engineers on a startup budget? Like this.”, Internet: http://venturebeat.com/2014/09/25/ how-can-you-hire-top-engineers-on-a-startup-budget/, September 2014, [visited 13 Feb 2015]

[7] S. Davis, “The MEAN Stack: MongoDB, ExpressJS, AngularJS, Node.js”, Internet: http://blog.mongodb.org/post/49262866911/the-mean-stack-mongodb-expressjs-angularjs-and, Sept. 9, 2014, [visited 13 Feb 2015]

[8] linnovate, inc., “The Friendly & Fun Javascript Fullstack for your next web application”, Internet: http://mean.io/#!/, [visited 13 Feb 2015]

[9] P.J. Murray & K. Oyri. “Developing online communities with lamp (linux, apache, mysql, php) - The imia osni and chirad experiences” Connecting medical informatics and bio-informatics, , pp. 361-, 2005.

[10] PCQuest, “MongoDB 2.4.7: An open source document database”, Internet: http://search.proquest.com/docview/1519705804?accountid=10041, Oct. 30, 2013, [visited 13 Feb 2015]

[11] J.Pokorny. “NoSQL databases: A step to database scalability in web environment” International Journal of Web Information Systems, vol. 9, pp. 69-82, 2013.

[12] Learnboost, “Mongoose: elegant mongodb object modeling for node.js”, Internet: http://mongoosejs.com/, [visited 13 Feb 2015]

[13] S.Tilkov & S.Vinoski. “Node.js: Using JavaScript to Build High-Performance Network Programs” IEEE Internet Computing, vol. 14, pp. 80-83, 2010

[14] P.J. Murray & K. Oyri. “Developing online communities with lamp (linux, apache, mysql, php) - The imia osni and chirad experiences” Connecting medical informatics and bio-informatics, , pp. 361-, 2005.

[15] J. Kim, E. Levy, A Ferbrache, P. Stepanowsky, C. Farcras, S. Wang, S. Brunner, T. Bath, Y. Wu, L. Ochno-Machado. “MAGI: a Node.js web service for fast microRNA-Seq analysis in a GPU infrastructure” Bioinformatics, vol. 30 , pp. 2826-2827, 2005.

[16] A. Mardan. Express.js Deep API Reference. Apress, CA: Berkeley, 2014

[17] S. Vinoski. "RESTful Web Services Development Checklist" IEEE Internet Computing, vol. 10, pp. 95-96, 2008.

[18] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, Y. Lafon, J. F. Reschke, “Hypertext Transfer Protocol -- HTTP/1.1”, Internet: http://www.w3.org/Protocols/HTTP/1.1/rfc2616bis/draft-lafon-rfc2616bis-03.html, June 2007, [visited 16 May 2015]

[19] T. Berners-Lee, R. Fielding, H. Nielsen, “Method Definitions”, Hypertext Transfer Protocol -- HTTP/1.0. IETF, , pp. 30-32, 2004

[20] J. Likness, “10 Reasons Web Developers Should Learn AngularJS”, Internet: http://java.dzone.com/articles/10-reasons-web-developers, Oct. 13, 2014, [visited 13 Feb 2015]

[21] G. Stovall, “AngularJS: An Overview”, Internet: http://glennstovall.com/blog/2013/06/27/angularjs-an-overview/, Jun. 27, 2013 [visited 13 Feb 2015]

[22] AngularJS, “What is a Module?”, Internet: https://docs.angularjs.org/guide/module, [visited 13 Feb 2015]

[23] Oracle inc., “Essentials, Part 1, Lesson 1: Compiling & Running a Simple Program”, Internet: http://www.oracle.com/technetwork/java/compile-136656.html#comp, [visited 13 Feb 2015]

[24] M. Kerrisk. Linux Programming Interface: A Linux and UNIX System Programming Handbook, No Starch Press, 2009

[25] M. Stephens. Sandbox. Encyclopedia of Cryptography and Security, 2011, pp. 1075-1078

[26] D.S. Peterson, M. Bishop; R. Pandey. “A Flexible Containment Mechanism for Executing Untrusted Code” Proceedings of the 11th USENIX Security Symposium, Aug. 2002

[27] “ISO 9241-11:1998(en)”, Internet: https://www.iso.org, 1998, [Visited: 18th of May 2015]

[28] Usability.net, “What is usability?”, Internet: http://www.usabilitynet.org/management/b_what.htm, 2006, [Visited: 18th of May 2015]

[29] J. Lewis and J. Sauro, “The factor structure of the System usability Scale”, Internet:, http://dx.doi.org/10.1007/978-3-642-02806-9_12, 1st of january 2009 [Visited: 18th of may 2015]

[30] A. Bangor, P.T. Kortum, J.T. Miller “An Empirical Evaluation of the System Usability Scale”, International Journal of Human-Computer Interaction, Internet: http://dx.doi.org/10.1080/10447310802205776, July. 30, 2008, [Visited 18th May 2015]

[31] J. Brooke, "SUS-A quick and dirty usability scale." Usability evaluation in industry,1996, p.189

[32] J. Kyrnin, “CSS Vendor Prefixes: What Are They and Why You Should Use Them”, Internet: http://webdesign.about.com/od/css/a/css-vendor-prefixes.htm, [visited 13 Feb 2015]

[33] Mozilla Developer Network, “Using CSS flexible boxes”, Internet: https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Flexible_boxes, [visited 13 Feb 2015]

[34] Mozilla Developer Network, “Using CSS animations”, Internet: https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Using_CSS_animations, [visited 13 Feb 2015]

[35] Mozilla Developer Network, “Using CSS transitions”, Internet: https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Using_CSS_transitions, [visited 13 Feb 2015]

[36] J. Kyrnin, “CSS Vendor Prefixes: What Are They and Why You Should Use Them”, Internet: http://webdesign.about.com/od/css/a/css-vendor-prefixes.htm, [visited 13 Feb 2015]

[37] A. Sitnik, “Autoprefixer: A Postprocessor for Dealing with Vendor Prefixes in the Best Possible Way”, Internet: https://css-tricks.com/autoprefixer/, Aug. 7, 2013, [visited 13 Feb 2015]

[38] I. Rammer & C. Weyer, “Modularizing AngularJS Applications”, Internet: http://henriquat.re/modularizing-angularjs/modularizing-angular-applications/modularizing-angular-applications.html, [visited 13 Feb 2015]

[39] C. Berry, “Modular AngularJS app design”, Internet: http://clintberry.com/2013/modular-angularjs-application-design/, Apr. 29, 2013, [visited 13 Feb 2015]

[40] Techopedia, “Modular Programming”, Internet: http://www.techopedia.com/definition/25972/modular-programming, [visited 13 Feb 2015]

[41] J. Bowes, “Agile vs Waterfall: Comparing project management methods”, internet: http://manifesto.co.uk/agile-vs-waterfall-comparing-project-management-methodologies/, 17 July 2014, [visited 13 Feb 2015]

[42] K. Schwaber & J. Sutherland, “The Scrum Guide”, p. 3, Internet: http://www.scrumguides.org/, July 2013, [visited 13 Feb 2015]

[43] M. Cohn. User Stories Applied. Addison-Wesley, 2003, pp. 17

[44] K. Schwaber & J. Sutherland, “The Scrum Guide”, p. 7-8, Internet: http://www.scrumguides.org/, July 2013, [visited 13 Feb 2015]

[45] J. Gothelf and J. Seiden, Lean UX: applying lean principles to improve user experience, 1st edition, O'Reilly Media, 2013, p. 55

[46] C. Zapponi, “Programming Languages and Github”, Internet: http://githut.info/, [visited 14 May 2015]

[47] Codility inc., “Codility: We test coders”, Internet: http://codility.com/, [visited 13 Feb 2015]

[48] Kattis, “Kattis: We help you hire great technical talent”, Internet: https://www.kattis.com/, [visited 13 Feb 2015]

[49] K. Neath, “Introducing Organizations”, Internet: https://github.com/blog/674-introducing-organizations, Jun. 29, 2010, [visited 13 Feb 2015]

[50] H. Sharp; Y. Rogers; J. Preece, Interaction design: beyond human-computer interaction, 3rd edition, Wiley, 2011, p. 341

[51] J. Nielsen, “Usability Metrics”, Internet: http://www.nngroup.com/articles/usability-metrics/, Jun. 21, 2001, [visited 1st of May 2015]

[52] J.S. Dumas and J. Redish, A practical guide to usability testing, Rev. edition, Intellect Ltd, 1999, p. 127

[53] E. Marcotte, “Responsive Web Design”, Internet: http://alistapart.com/article/responsive-web-design/, May 25, 2010, [visited 13 Feb 2015]

[54] J. Nielsen, Usability engineering, Academic Press, Boston, 1993
and Y. Rogers, H. Sharp and J. Preece, Interaction design: beyond human-computer interaction, 2nd edition, John Wiley & Sons, 2011, p.408

[55] S. Lichtenfeld et al, “Fertile Green: Green Facilitates Creative Performance”, Personality and Social Psychology Bulletin, 38 (6) , pp. 784-797, 2012

[56] C. Newman, “Considering the color-blind”, Web Techniques, 5(8), 59-61. Internet: http://search.proquest.com/docview/274911012?accountid=10041, 2000 [Visited; 10th of May 2015]

[57] Cloud 9 inc., “High performance code editor for the web”, Internet: http://ace.c9.io/#nav=about, [visited 13 Feb 2015]

[58] Mozilla, “Vi bygger ett bättre internet”, Internet: https://www.mozilla.org/, [visited 13 Feb 2015]] and Cloud9 IDE [Cloud 9 inc., “Your development environment, in the cloud”, Internet: https://c9.io/, [visited 13 Feb 2015]

[59] E. Kitamura, “Working with quota on mobile browsers”, Internet: http://www.html5rocks.com/en/tutorials/offline/quota-research/, Jan. 28, 2014, [visited 13 Feb 2015]

[60] Mozilla Developer Network, “<canvas>”, Internet: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/canvas, [visited 13 Feb 2015]

[61] gskinner inc., “CreateJS: A suite of Javascript libraries and tools designed for working with HTML5”, Internet: http://www.createjs.com/, [visited 13 Feb 2015]

[62] gskinner inc., “TweenJS: A JavaScript library for tweening and animating HTML5 and JavaScript properties”, Internet: http://www.createjs.com/TweenJS, [visited 13 Feb 2015]

[63] gskinner inc., “PreloadJS: A JavaScript library that lets you manage and co-ordinate the loading of assets”, Internet: http://www.createjs.com/PreloadJS, [visited 13 Feb 2015]

[64] Cloud9 inc., “Editor”, Internet: http://ace.c9.io/api/editor.html#Editor.event.change, [visited 15 May 2015]

[65] M. Kearny & M. Gaunt, “Add a Build Process”, Internet: https://developers.google.com/web/fundamentals/tools/build/setupbuildprocess#dont-trip-up-with-vendor-prefixes, Sept. 25, 2015, [Visited 1 May]

[66] PostCSS, “Autoprefixer”, Internet: https://github.com/postcss/autoprefixer, [Visited 1 May]

[67] A. Deveria, “Can I use... Support tables for HTML5, CSS3, etc”, Internet: http://caniuse.com/, [Visited 1 May]

[68] A. Reinman, “Nodemailer: Send e-mails with Node.JS – easy as cake! E-mail made in Estonia”, Internet: http://www.nodemailer.com/, [Visited 1 May]

[69] MongoDB, “db.collection.findAndModify()”, Internet: http://docs.mongodb.org/manual/reference/method/db.collection.findAndModify/, [Visited 1 May 2015]

[70] DigitalOcean, “How To Use PM2 to Setup a Node.js Production Environment On An Ubuntu VPS”, Internet: https://www.digitalocean.com/community/tutorials/how-to-use-pm2-to-setup-a-node-js-production-environment-on-an-ubuntu-vps, [Visited May 1 2015]

[71] Strongloop inc., “express-session”, Internet: https://www.npmjs.com/package/express-session, [Visited May 1 2015]

[72] J. Desboeufs & K. Banner, “connect-mongo”, Internet: https://www.npmjs.com/package/connect-mongo, [Visited May 1 2015]

[73] PassLib, “passlib.hash.bcrypt - Bcrypt”, Internet: http://pythonhosted.org/passlib/lib/passlib.hash.bcrypt.html [Visited May 1 2015]

[74] B. Schneier, “The Blowfish Encryption Algorithm”, Internet: https://www.schneier.com/blowfish.html, [Visited May 1 2015]

[75] M. Bauer. “Paranoid penguin: an introduction to Novell AppArmor” Linux Journal, vol. 148, 2006

[76] The Open Group, “IEEE Std 1003.1™, 2013 Edition”, Internet: http://pubs.opengroup.org/onlinepubs/9699919799/functions/getrlimit.html . [Visited May 1 2015]

Appendices

Appendix A

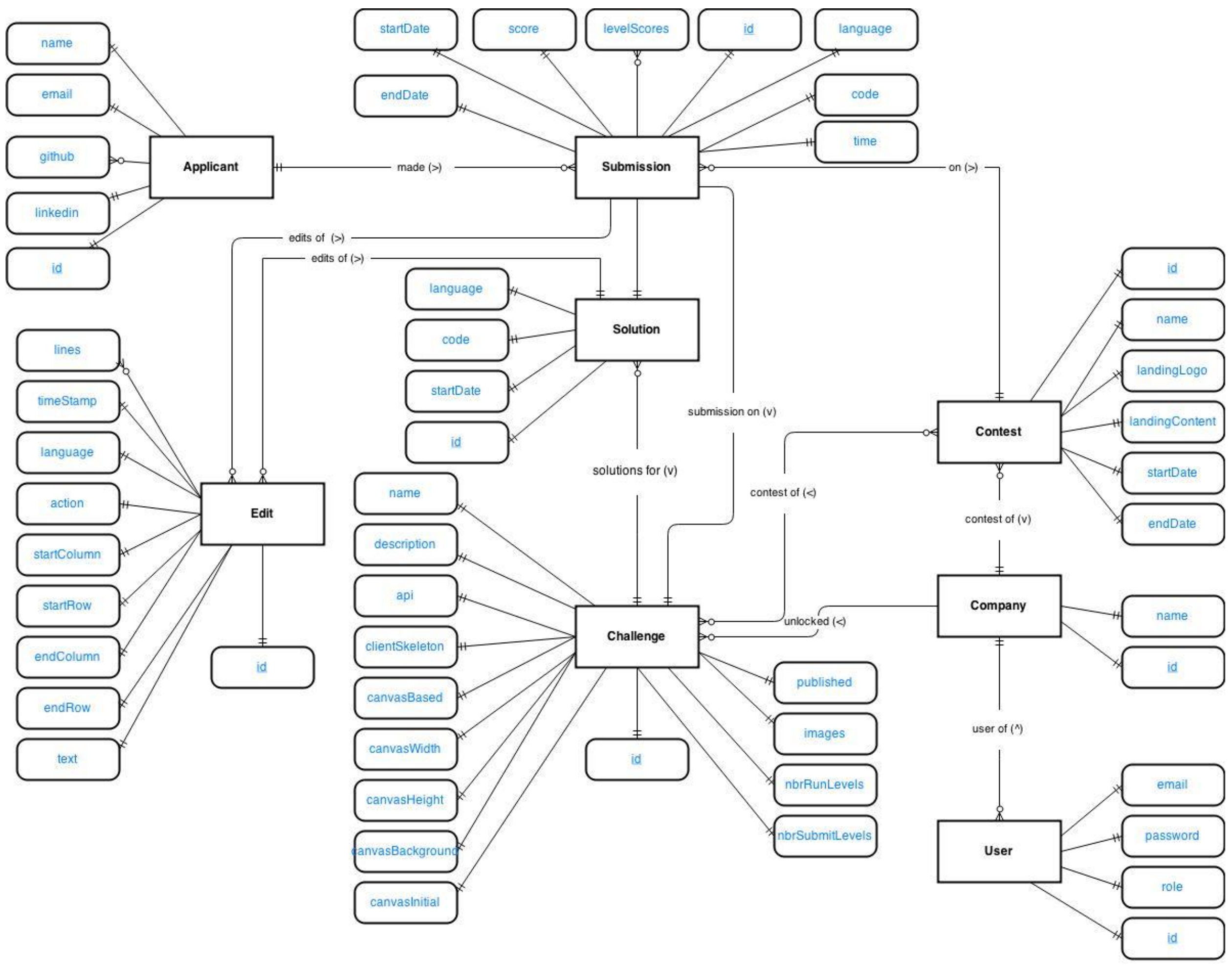


FIGURE 22 - DATABASE ER-DIAGRAM

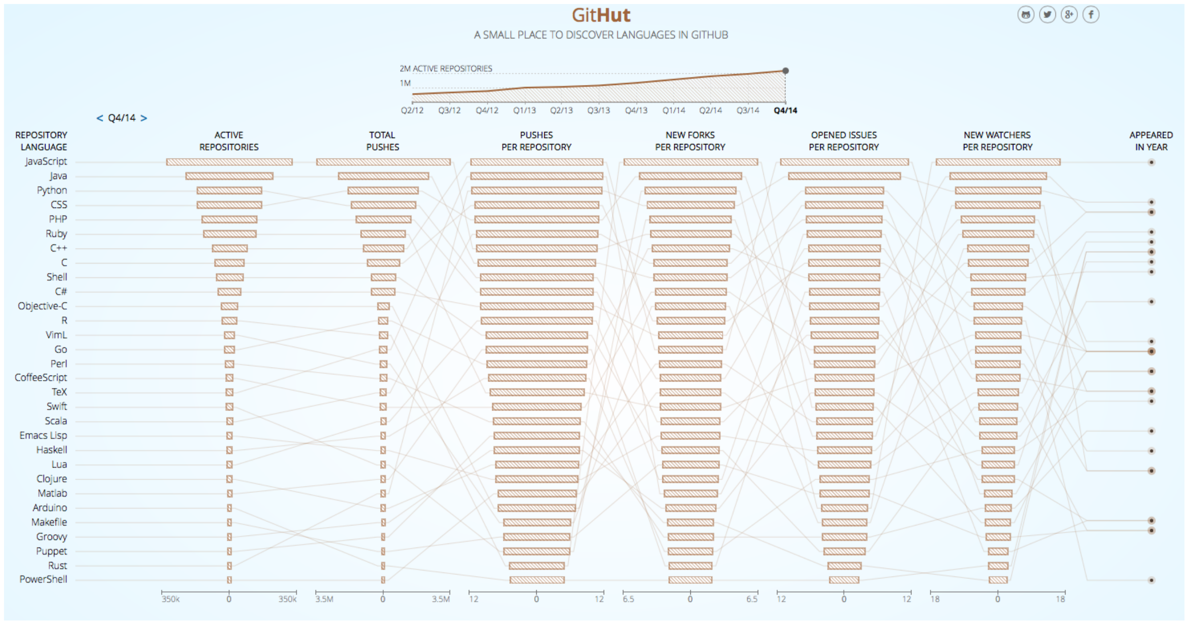


FIGURE 23 - GITHUB LANGUAGE POPULARITY
[C. Zapponi, “Programming Languages and Github”, Internet: http://githut.info/, [visited 14 May 2015]]

Original	**Modified**
I think that I would like to use this system frequently.	I think that I would like to use this application again.
I found the system unnecessarily complex.	I found the application unnecessarily complex.
I thought the system was easy to use.	I thought the application was easy to use.
I think that I would need the support of a technical person to be able to use this system.	I think that I would need the support of a technical person to be able to use this application.
I found the various functions in this system were well integrated.	I found the various functions in this application were well integrated.
I thought there was too much inconsistency in this system.	I thought there was too much inconsistency in this application.
I would imagine that most	I would imagine that most people

people would learn to use this system very quickly.	would learn to use this application very quickly.
I found the system very cumbersome to use.	I found the application very cumbersome to use.
I felt very confident using the system.	I felt very confident using the application.
I needed to learn a lot of things before I could get going with this system.	I needed to learn a lot of things before I could get going with this application.

FIGURE 24 - SYSTEM USABILITY SCALE STATEMENTS

FIGURE 25 - User Testing Results for Challenge View

Scenarios - Challenge view								
Find the description for API-method “move down”			ok	ok	ok	ok		
Change language to javascript			ok	ok	ok	ok		
Minimize description/canvas part of application			ok	ok	ok	ok		
Make the console really small			ok	ok	ok	ok		
Run your code			ok	ok	ok	ok		
Hand in solution your for grading			ok	ok	ok	ok		
Scenarios - Thank you view								
Submit your name and email			ok	ok	ok	ok		
Share your score on facebook			ok	ok	ok	ok		
								4.666666667

Statements, graded 1 to 5						Average
I think the application is easy to use	5	4	5	4		4.67
I think the application is aesthetically pleasing	5	3	4	4		4.00
The application makes me want to use it	5	4	5	4		4.67
I think the user interface is intuitive	5	4	5	4		4.67
I think the interface looks professional/like a polished product	5	5	5	5		5.00
Average:	5	4	4.8	4.2		4.60
Open ended questions						
What would you change?	-	-	-			
Any general comments?	Nice with soft minimizing, scroll bar and close all (maybe floating at the top) in api, stylish, focus on what is important, no null-lull, clean. Color blind adaptation? Debugging - what if the you have a partial solution and want to test the latter half of it, it is slow to go through all the steps					
	Looks good, easy to understand, clean - not many choices					
	no changes, looks professional					
Metrics						
Do any of the scenarios take unexpectedly long?	no	no	no	no		
Scenario success	all	all	all	all		
Errors - critical/non-critical	only errors where mac/pc related - alt 2 changed tab instead of writing an @					
Errors	none	none	none	none		