An Interactive Web Application to motivate learning of programming

GYOKAY ALI

Banner ID 000762233

Supervisor: **YASMINE ARAFA**

Final Year Individual Project COMP1682

A dissertation submitted in partial fulfilment of the University of Greenwich undergraduate degree programme

BSc (Hons) Software Engineering

18th April 2016

Word count: **7760 (less than 14,000 not counting preamble and appendices)**

ABSTRACT

Programming is the language of the future and preparing students to this future is crucial, because there is no other solution other than learning to program. This project has a mission to investigate and help to understand the key motivating factors affecting learning of programming among first year undergraduate students. Why students are afraid of programming in such level? Is it too difficult or just lack of motivation? What is the solution to this problem?

The outcome of this piece of paper finds an answer to these aforementioned questions by conducting a deep research into past literature by credible authors and sources who tried to solve or highlighting the same problem. Different approaches to motivate ICT students into programming will be discussed in detail. The most substantial factors will be taken under serious consideration and applied into real application.

As a result an interactive web application will be developed to support this project and hopefully it will find a place in practice to motivate students to program in a more fun and engaging way. For the purpose, due to tight deadlines ASP.NET will be used as it allows fast development. Particular attention will be drawn into Human-Computer Interaction (HCI), and accessibility and usability of the web application.

ACKNOWLEDGEMENTS

I would like to take this opportunity to show great appreciation to all the support and help given to me during this project's lifecycle.

I would like to thank my supervisor Dr Yasmine Arafa for her commitment, advices and support and co-supervisor Dr Markus Wolf for his excellent lectures in ASP.NET that helped me a lot.

Also, I am very grateful for the support of my family and my girlfriend.

CONTENTS

1 INTRODUCTION

1.1 Background Information

In the recent years there has been a huge demand of recruiting IT professionals worldwide to fill the hundreds of thousands positions, even millions currently available, a problem which continues to grow much more intensively. It is expected by 2020 the demand to show ”*increases ranging from 28% to 32%”* (Thibodeau, 2012). This is not due to the lack of IT graduates or specialists, but due to lack of quality and competent IT professionals. Quality is very substantial requirement, especially in the software industry.

To produce quality and competent individuals, who will be ready to start an immediate career after graduation, organizations and universities has to cooperate with each other to find a solution or improve the already available ones. This is a rapidly changing environment. The solution is to keep students motivated all the way through and inject them with the idea that there is nothing to be afraid of in programming. A solution, which is to teach and help them improve and develop the newly acquired knowledge from the lectures, on their own pace. This must be achieved in an engaging, interactive and fun way, because the beginners get bored very easily after seeing their first error on the screen. This is where they get lost and confused.

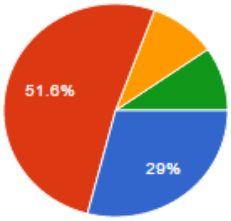
One of the deliverables of this project was to develop an interactive web application that will help motivating the beginners of this area of study. The application will be developed according to current standards of the industry with emphasis on HCI to make it an interactive approach to learning. Accessibility and usability issues will be considered very carefully to make the application available to anyone willing to tackle the difficulties they are facing in programming. The product is also intended to support the findings of the research study.

1.2 Project Scope

The purpose of this project, is firstly to conduct a research into the field, to identify the key factors that mostly motivate the youth of this era to learn programming. Also to investigate in detail and compare similar products already available, with greater attention to HCI, to acquire a better understanding of the concept and to give a better user experience with the intended final product. Prior to development process, some research into Accessibility and Usability is mandatory, to achieve a quality product that adheres to standards. After all this an interactive web application will be developed to support the arguments highlighted in the Literature review of the report. A good process will result in a good product.

The proposed web application will let the students to practice and learn in depth a chosen programming language on their own. After an online questionnaire was sent out to First Year Undergraduate students of the CMS Department in the University of Greenwich, the exact programming language to be used in the product was finally decided, which is C# (C Sharp). The results can be clearly seen from the summary of responses (see Appendix B). A detailed discussion on this will be available in the next chapters.

Which Programming language would you like to learn next or most?



Java	9	29%
C#	16	51.6%
JavaScipt	3	9.7%
Other	3	9.7%

The application will involve some quizzes and small games, which will make the time spent more fun and motivating. Also there will be implemented a score and progress functionality. At first everything will be very easy for the beginners, but it will get more difficult as the student progress through the learning environment by collecting points.

If the project meets all of the requirements and deliver all components, it will be deemed as successful one.

1.3 Project Aim

The aim of this project is to develop an innovative learning application/website for Year 1 CIS student to learn and practice programming basics by "doing". The long term goal is to motivate learners.

1.4 Project Objectives

A detailed set of objectives were set at the beginning of the project. Please refer to the Project Proposal (see Appendix A) for full version. Below is a summarization of the main objectives:

- Conduct a thorough research in the relevant field and write an Investigation report consisting of: Literature review, Review of existing products and Technical review.
- Produce the Design documentation consisting of separate documents in detail: Statement of requirements (Requirements elicitation, Questionnaire results review, Functional and Non-functional requirements); UML and ERD design; User Interface design.
- Product development: Web application, Web service? and SQL Database
- Produce Test documentation as a result of thorough Black box, White box and Unit testing
- Write Self-evaluation and Product and Process evaluation
- Conclude the project with a Reflection report

1.5 Project Deliverables

Two products will be delivered at the end of this project. A result of systematic investigation and knowledge gathering through reading and analysing quality literature.

- An Interactive Web Application
- Final Year Project Thesis

1.6 Project Methodology

As the life of the current project is unpredictable and open to changes many times, this will require the Rapid Application Development (RAD) Prototyping approach using the 'MoSCoW' methodology to control the process in an evolutionary way. Building an initial prototype as early as possible will be a very significant step. Also an iterative concept will be required in order to test and get feedback from the end-user.

The reason for choosing RAD is due to the fact that is time and cost effective methodology, which puts more emphasis on development to deliver a quality end product, rather spending time on planning. It is completely opposite of the Waterfall model, which puts more emphasis on planning. The RAD Prototyping is suitable in this case, because it allows going back to make corrections to the requirements as knowledge is gathered along the way of development.

The evolutionary methodology 'MoSCoW' will be used to identify the high priority and low priority requirements. In this way requirements are divided into two: high level requirements (MUST and SHOULD) and low level requirements (COULD and WOULD). More attention will be given to the high level requirements as they will be enough to have a functioning prototype delivered. And if time permits the low level requirements will be taken under consideration and implemented to have a fully functioning faultless quality product. But this is not the main goal, as the future of the project is unpredictable.

Drawbacks may occur due to the nature of RAD, which are reduced scalability, reduced features, limited functionality as the developed application starts as a prototype and evolves during the development life cycle into a finished product. More than one prototype will be developed along the way to ensure quality and better functionality. In this case three prototypes will be delivered, each one an evolved version of the previous one.

Adjustments would be made if required during the length of the project. Monitoring the plan at all times will reduce the risks of fail and keep up with deadlines (see Appendix A).

2 THE FEASIBILITY STUDY

An email that contains short description, aim and purpose of the study, with a link to the online questionnaire, was sent out to Year 1 CMS students by the supervisor of this course. The results were a little bit disappointing at first, because only 31 students entered their answers to the questionnaire. Nevertheless, the summary of responses by the students were quite enough to prove the feasibility of this study.

The results (see Appendix B) show that 61.3% of the students identified themselves as 'Beginners' in programming, also with 19.4% and 6.5% respectively as 'Intermediate' and 'Advanced' programmers. Only 4 students (12.9%) were identified as 'No knowledge' (no previous experience). As the aim of the final product is to motivate mainly beginners, the study acquired good results.

When asked 'Which Programming languages do you know already?' responses show that most of the students (72.4%) are already familiar with 'Java'. This is due to the fact that most of the First Year students are enrolled on computer programming courses with 'Java'. A third of them (37.9%) also familiar with 'C#', a knowledge and experience more likely gained through college or self-study. As these two languages are the most common among students, the rest of the answers will be ignored. Ignored because students shared that they would like to practice and learn 'C#' the most (51.6%), followed by 'Java' (29%).

Most of the students (25.8%) were not confident and sure with their programming skills, which reveals a serious problem. Hopefully the end product will be able to motivate them in their studies. To mention, a reason for motivation to the author of this project, was the positive answers of the students who mostly liked the idea of the project. This means that they believe in the success of this project.

Another area to note is students approach positively the idea of using interactive learning environment with games (87.1%) answering with 'Yes' and small quizzes (48.4%). They believe that this approach will help them to get motivated in programming.

To conclude due to tight deadlines only one language will be available to practice at this point, which is 'C#', the champion with 51.6%. Moreover more emphasis will be on implementing simple games to grab the attention of the students and if time permits some quizzes will be added as well. Future improvements to the learning environment could be the added in feature to practice more than one language (e.g. C#, Java, JavaScript and etc.) and availability of more interactive content not limited to only one or two.

Please check the questionnaire and summary of responses available at (see Appendix B).

3 LITERATURE REVIEW

3.1 Introduction

This literature review, a part of the report is intended to discuss the findings of an extensive and deep investigation. Including to that knowledge gathered by the analysis of existing research studies carried out by quality academics and professionals. The topics covered by them are quite wide in range such as e-learning, robotics, issues in programming, learning motivation, pedagogy and other problems of the field relevant to this project.

The author will be addressing the problem and solution to the problem divided into few sections, and finally conclude his arguments and findings with a summary of the problems and solutions investigated and analysed. This will include the difficulties of first year undergraduate students facing in programming and the underlying reasons. The key factors of motivation among students will be the primary topic of this review, which will also help to form and develop the end product of the project. Hence before moving to the key factors that help in motivation, firstly will be looked at the demotivating factors. What is so frustrating or bad in the curriculum or the system that cannot motivate the students and meet their expectations? Their expectations are very important point to look at and discuss.

A summarized list of key motivation factors and concepts to be discussed and considered in the following sections are: use of e-learning environments, e-assessment, and web based interactive solutions, use of visualization and animations, Virtual Reality (VR), programming robots, programming games and learning styles.

3.2 Difficulty of learning to program

“"Programming" is a complicated business” (Jenkins, 2001). Jenkins is not the only one who argues that learning programming is very difficult and not the only one who proposes few possible approaches to solve this problem. The different approaches trying to find solution to this issue are yet to come in the next section, in detail. Moreover in nearly all research papers used for this literature review, the difficulty of programming was the first thing to point out by the authors. Everything about programming was *“recognized as a difficult* endeavour*”* (Kelleher and Pausch, 2007), especially for first year students who are novice and cannot assimilate the overwhelming material and abstract concepts, that the traditional approach to teaching offers to them (Kuo-En Chang et al., 2000).

Regardless of their many efforts, programming teachers also find themselves in struggle to teach the important aspects of the material and grab the attention of the students (Drumond, Brandao and Salles, 2014). The authors also indicate that every little problem or stress reduces student’s enthusiasm to learn.

According to Hwang et al., *"coding to solve problems"* on its own as a single activity is not good enough to develop students' cognitive learning (Hwang et al., 2008). They suggested applying multiple activities in proper, following each other from simple to advanced concepts. Coding to solve problems is a common approach, however as mentioned by the authors single activities can start a feeling of monotony and ineffectiveness in students. Again the traditional programming and the way it is taught to students is not engaging, relevant and fun. Markham and King indicated in their conference paper that examples as introduction to variables, expressions, conditionals and etc., and their use in assessments are *"purely academic, designed to practice the concepts taught"* (Markham and King, 2010). They conclude with the argument that this approach is no longer suitable for attracting the youth of this era into computer programming courses. Moreover the traditional approach cannot do much to prevent the high dropout rates of first year undergraduate students. Apparently the debate on practicing the best teaching methodologies will never end (Markham and King, 2010).

Some examples used in teaching programming are either trivial or not specific to the real world. For instance, Anderson et al., gave examples with the "Fibonacci sequence" and "hello world!" and tried to show how some students may fail to understand the application of similar examples, beyond computing (Anderson et al., 2015). The authors also concluded that students must work on real life problems in order to prevent this kind of false impression. To program the aforementioned examples could be easy, however the idea of programming itself becomes difficult, as long as the students don't know why they are doing and learning something in particular. Some could think of it as a waste of time and may decide to skip that lesson, even abandon the course as a whole. A concern resulting in "misbeliefs" such as "programming is difficult".

Jenkins is not the only who claimed that programming is difficult, as mentioned previously (Jenkins, 2001). Lahtinen, Ala-Mutka and Järvinen also supported his arguments and added that the difficulty is in the nature of programming (Lahtinen, Ala-Mutka and Järvinen, 2005). The authors believed the *"lack of personal instruction"* in students was another reason. Programming is a skill, supported by knowledge or even by aptitude. Why not! It requires correct understanding and dedication (time) to grasp the abstract concepts in its nature. Lahtinen, Ala-Mutka and Järvinen argued in their research paper that the most important reason to face difficulties in programming, is because students new to programming don't know how to apply what they have learnt in class. Students have to be encouraged to *"learning by doing"* by their teachers (Lahtinen, Ala-Mutka and Järvinen, 2005). Practical exercises and assignments are a key to acquire deeper understanding of programming.

A research carried out by Butler and Morgan, included surveying 150 students from their organization, ended up with unsurprising results – a high percentage of the students did have difficulties in implementing high level programming concepts, especially challenged with Object-oriented programming (OOP) (Butler and Morgan, 2007). As a result this showed that nearly all novice

programmers don't feel comfortable with conceptual material. Esteves and Mendes strongly agreed with the previous argument and clearly stated the difficulties faced with conceptual aspects in their conference paper (Esteves and Mendes, 2004).

Gomes and Mendes summarized a list of possible determining factors of student failure. Factors such as: Static format of learning versus dynamic behaviour of a program; Abstract concepts hard to visualize; Emphasis on theoretical knowledge and memorization; Limited access to the individual support of their teachers; Different skillset; Lack of visual representation to non-English speaking students (Gomes and Mendes, 2007). To add on this some important factors of failure and demotivation were identified by several researchers: lack of study, practice, determination and time, wrong learning strategy, subject difficulty, unsuitable teaching method and imbalanced treatment (Hawi, 2010; Hijon-Neira et al., 2014). Other factors that affect the learning of students were proposed by Jenkins too, in his substantial piece of work: multiple skills, multiple processes, the language, and educational novelty, interest, reputation and image, and pace (Jenkins, 2001).

"Although young people interact with digital media all of the time, few of them can create their own games, animations, or simulations. It's as if they can "read" but not "write"" (Resnick et al., 2009).

To rectify all these problems there needs to be introduced means of motivation. By means it was meant either one of the following possible solutions or a combination of them: PBL, robots, e-learning environments (web or desktop), e-assessment environments, interactive visualisation tools and games. The following section is dedicated to discuss some of these methodologies used as a way to motivate students in different organizations.

To finalize this section, Jenkins concluded his arguments with *"Programming is learned by programming, not from books"* (Jenkins, 2001), thus calling for a new approach into teaching and learning of programming.

3.3 Motivation

This section is intended to discuss the key factors of motivation, among students enrolled on computer programming courses, identified by quality researchers. The author will expose his findings to explain the key factors of motivation, as well as possible approaches and methodologies to enhance students' learning process. The main goal found in all research papers was to motivate and attract students into programming. Moreover to prepare them to the future real life of a Software Engineer, by developing their soft skills, in addition to technical.

3.3.1 Key factors of motivation

Law, Lee and Yu in their hypothetical research paper proposed that the motivation of students and the factors affecting them are strongly correlated with students‘ efficacy (Law, Lee and Yu, 2010). They introduced two types of motivation: ‘intrinsic’ and ‘extrinsic’. To identify a set of motivation factors the authors categorized the two types as:

Intrinsic:

- ‘individual attitude and expectation’
- ‘challenging goals’

Extrinsic:

- ‘clear direction’
- ‘reward and recognition’
- ‘punishment’
- ‘social pressure and competition’

(Law, Lee and Yu, 2010)

In their research study, Law, Lee and Yu set two hypotheses out of three, to check the correlation level of students’ efficacy against each motivation factor. To summarize students who valued ‘intrinsic’ or ‘extrinsic’ factors showed higher degree of efficacy (Law, Lee and Yu, 2010). Undoubtedly there is a strong relationship between efficacy and performance, which as a result showed that students with higher level of efficacy would perform better in learning computer programming. Drumond, Brandao and Salles also proposed the use of ‘intrinsic’ type of motivation, with strategies related to games, activities to simulate logical thinking and letting students set their own objectives (Drumond, Brandao and Salles, 2014).

Interesting results of a study conducted by Cummins, Azhar and Sklar, using robotic-based learning in programming courses with novice students, revealed several factors to have a positive effect on motivation: ‘absolute integration’, ‘same programming environment’, ‘simplicity of programming’, ‘robust communication’ and ‘robust hardware’ (Cummins, Azhar and Sklar, 2008). Factors, which present comfortable environment to students as they would like to see, as well as helping develop their soft skills. Other researchers also agreed with this and concluded that students’ learning capabilities were enhanced when they work in comfortable and usable environment (Feldgen and Clua, 2004; Jurado, Redondo and Ortega, 2012). Lahtinen, Ala-Mutka and Järvinen surveyed students in their organization and ended up with interesting answers by students, who shared that studying and programming by themselves, ‘learning by doing’ and interesting examples were key motivation for them (Lahtinen, Ala-Mutka and Järvinen, 2005). But the authors noted that the one has to be careful when judging student opinions, because they contradicted teachers’ opinions in the same research study. Feldgen and Clua also indicated two types of learners: ‘analytic’ and ‘global’ learners (Feldgen and Clua, 2004). Their piece of work was quite significant to take into account.

"We know that people learn best, and enjoy it most, when they are working on personally-meaningful projects" (Resnick et al., 2009).

An approach applied by Binas and Pietrikova in their organization showed significant improvement and better motivation to students. The factors adopted were:

- Letting students program at the soonest
- Not using unfamiliar concepts, prior teaching
- Teaching step by step
- Using attractive examples
- Teaching how to debug their programs
- Sharing the solution of difficult problems
- Avoiding teaching irrelevant material

(Binas and Pietrikova, 2014)

To conclude Jenkins and Lykke et al. shared the same vision in their quality pieces of work and indicated the substantial significance of the different motivation types 'extrinsic', 'intrinsic', 'social' and 'achievement'. (Jenkins, 2001; Lykke et al., 2014). The latter publication also stated that the 'comfort level' of students is one of the strongest factors in motivation (Lykke et al., 2014).

3.3.2 Different approaches to motivate students

This subsection was aimed to discuss the available methodologies and approaches recommended by credible sources, which have significant effect on motivating students and attracting them more into computer programming. The findings are not limited to one good solution, but encouraging to combine few or more to suit the needs of the students.

Third hypothesis set by Law, Lee and Yu encouraged the use of facilitative e-learning systems, due to positive linkage with the students' learning efficacy (Law, Lee and Yu, 2010). Others agreed with his statements too. Another similar approach to motivate programming was introduced using 'Wanda', a tool to create card-based games encouraging the use of algorithms, and improved the algorithmic thinking of students (Drumond, Brandao and Salles, 2014). A challenging and competitive environment that can be used by teachers to teach their students the basic concepts of algorithms. Survey conducted by Drumond, Brandao and Salles gathered significantly helpful data about students' experience with 'Wanda'. About 90% of the students claimed that this kind of learning environment was very interesting and fun, and they wanted to see similar activities in other courses too (Drumond, Brandao and Salles, 2014). This proves that the use of interesting and engaging methodologies in the course could have big influence on students' interests.

A study carried out by Kuo-En Chang et al., introduced the 'completion strategy', which encouraged the use of examples in programming (Kuo-En Chang et al., 2000). This approach was used to let students program by completing, modifying and extending the given examples. According to Kuo-En Chang et al., this strategy was very beneficial in motivating students. Other activities like 'gap filling' and 'peer assessment' were found to be very helpful in the cognitive development of students, if there was an assisting tool (Hwang et al., 2008). The authors conducted interviews with students and the derived information showed that activities like 'coding to solve problems' and 'peer assessment' enabled to improve students' problem solving and evaluation skills (Hwang et al., 2008).

Yet another approach introduced by Ball et al. was the use of 'LEGO Mindstorms', which aimed to prove that Computer Science could be exciting and fun, despite the wrong perceptions of people, e.g. being 'boring' and difficult (Ball et al., 2012). 'LEGO Mindstorms' was accepted by the authors as a creative and innovative way to encourage and motivate children and students, by showing them the importance of working as a team and solving problems (Ball et al., 2012; Cummins et al., 2008). Similarly a research by Markham and King exposed the positive effect of robots in programming motivation, and how working with them was fun and satisfactory (Markham and King, 2010). Other than motivation to learn, 'robot-based' learning was stated as a very effective way to improve students' problem solving, team working and communication skills. Anderson et al. also added the effect of using 'robotics' in education, including other ones like 'media computation' and 'animations' (Anderson et al., 2015). They strongly suggested the use of real life applicable projects and to keep deadlines flexible.

"When creating curricula based on games, they need to be sure that the games they choose really are able to motivate students of both genders" (Kelleher and Pausch, 2007). Diversity is a substantial aspect to consider. That's why in 'game-based' learning teachers have to consider both genders, and create environments suitable for both sides. For instance, if a 'game-based' learning environment was proposed, which included only games in the sort of 'first-person shooters' and 'sports', as a consequence we would put barriers in front of the female learners. Another game-based learning environment for novice programmers was also introduced, called 'Greenfoot', with significant results in motivating students and facilitated learning. The study results indicated that more than 60% of the students were satisfied with this approach and 65% rated positively its usefulness, effectiveness and productiveness (Hijon-Neira et al., 2014).

Other approaches that might help to increase the motivation, involved the use of 'concurrent programming models' (Leavens, 2008), and 'pedagogical (elementary) programming patterns' (Nunes de Barros et al., 2005). However the arguments given to these approaches did not fulfil the expectations of the author and were not measured as very helpful in this particular case study.

Esteves and Mendes in their study indicated the significant benefits of using 'animation based simulation' to reduce the difficulties of students (Esteves and Mendes, 2004). According them this kind of approach could help to interact better with program's dynamics and let student learn on their own pace. Novice students into programming frequently face problems while using the complicated Integrated Development Environments (IDE), as they are professional tools and meant for advanced programmers. That's why the introduction of learning environments with pedagogical goals could be more suitable to help students learn (Esteves and Mendes, 2004).

Yet again another product that was praised by students with positive comments was 'AulaWeb', a "*Web Assisted Self-assessment*" application (Garcia-Beltran and Martinez, 2006). The authors stated that this approach is not to assess the students, but motivate and encourage them. Moreover self-assessment objects e.g. quizzes help to "evaluate their own understanding and misconception" (Matthews, Hin and Choo, 2014). Respectively an Intelligent Tutoring System (ITS) proposed by others, was described as an ideal way of Computer-Assisted Learning (CAL) approach, which guided and helped students in their learning process to acquire the best skills (Jurado, Redondo and Ortega, 2012).

The difficulties of learning to program and the need to find a proper motivation method were considered and discussed by nearly all researchers so far. One more way to attain this was by introducing a web-based 'Intelligent assistant', with exact goals similar to the previous examples (Konecki, Kadoic and Piltaver, 2015). Similarly an interesting approach to motivate students was the use of Virtual Reality (VR) in computer programming courses (Chandramouli, Zahraee and Winer, 2014). The authors described it as a fun-based learning in 3D environment, which could help to reduce the cognitive overload on students.

The author also analysed an interesting comparison study by Lykke et al., which had for purpose to compare different approaches and their effectiveness against each other (Lykke et al., 2014). Three approaches were put in comparison: Project Based Learning (PBL), PBL + LEGO Mindstorms (robot-based approach), and the traditional way of teaching. The results of the study indicated there wasn't too much difference between each and their effectiveness was questioned (Lykke et al., 2014).

Jiau, Chen and Ssu clarified the difference between traditional game-based assignments and commercial games, and indicated why the commercial ones are so successful (Jiau, Chen and Ssu, 2009). They took similar approach in their study, and observed the improvements that traditional approach wouldn't have achieved. The key was to develop competitive games, where *"outcomes are largely under the control of the students"* (Jiau, Chen and Ssu, 2009). This is called 'game-based metrics approach'. All student are familiar with games, therefore understand a problem better this way, as if they were trying to win the game (Feldgen and Clua, 2004). Apparently game-based assignments and game-based metrics are one of the best ways to motivate students today.

Simple, but meaningful conclusion to this section –

"A little bit of programming can go a long way" (Resnick et al., 2009).

3.4 Conclusion

To conclude, the investigation process of extensive amount of quality literature and their analysis let the student to obtain significantly in-depth understanding of the problem stated. The gathering of proven arguments and applying them to form the literature review of this project, did help the student to improve his critical-thinking, self-study, evaluation, writing and research skills. The credible sources mostly agreed with the difficulties students are facing in learning and understanding the conceptual aspects in programming. Motivating students is the key to teach them programming.

Throughout the whole process of reviewing the findings, the motivation problem of students in learning programming, the source of these problems and what we can do to solve them were clearly stated in detail. The investigation resulted in with the general fact that programming will be always 'difficult', however if we approach the problem with carefully planned methodologies and apply them, there will be significant changes in the learning process of students. After the investigation the student can tell that, students need to work in their comfortable environments in order to learn and practice programming. This means the curriculum has to put efforts to understand students' expectations from a learning activity or methodology. This argument was proven to be correct in the literature review, indicating the fact that students learn the best in their own way, the way they feel more comfortable (e.g. playing games and following strategy instructions).

Several factors of motivation were discussed in the review, e.g. intrinsic, extrinsic, social, achievement and others. Each student has their own way of learning, therefore the factors that motivate them vary. For instance some could have only one way to motivate themselves, but also some others could have a mix of different types of motivation. Also many different approaches were proposed to help student motivation: web and desktop-based e-learning environments, web-based self-assessment environments, intelligent assistive environments, visualization and animal tools, game-based learning, robot-based learning, Virtual Reality (VR) and pedagogically oriented development environments. Each of the mentioned approaches had proven benefits in motivating students to learn programming. Other benefits of using these ways of teaching were indicated as: reduced drop-out rates, increased success in assignments, increase in the knowledge, problem-solving and critical-thinking skills of the students.

Finally the investigation showed that the most appropriate way to motivate students in this case would be the use of interactivity with 'game-based approach', especially it is substantial to take into account the 'game-based metrics approach'. Overall this was a very useful and helpful process of learning.

4 LEGAL, SOCIAL, ETHICAL AND PROFESSIONAL ISSUES AND CONSIDERATIONS

5 REVIEW OF EXISTING PRODUCTS

5.1 Introduction

This chapter is intended to review existing products in the market available for free or fee, which are similar to this project's final product. By similar it was meant having the same principles, approach and aim, including running on the same platform (web-based). Only the most appealing and well-known to the audience products will be reviewed. Because an investigation on existing products ended up with too many results, which would have required too much time to test and review, if we consider the tight deadlines of this project. As a result three products were reviewed, which are pretty interactive and game-based with the purpose to ease and motivate learning of programming. The chosen web-based products are: 'CodeCombat', 'CodinGame' and 'CodeMonkey'. Each product will be reviewed in a separate section.

Product Name	URL to Website
CodeCombat	https://codecombat.com/
CodinGame	https://www.codingame.com/start
CodeMonkey	https://www.playcodemonkey.com/

5.2 Product 1 - CodeCombat

5.2.1 Review

CodeCombat is a web application with the aim to teach programming with 'game-based' approach. There are no lessons, but instructions and quests (assignments) to follow in the game. At first the levels are pretty easy to complete, but as the player continues to progress the game difficulty increases too. As you progress the need for better problem-solving and algorithmic-thinking skills increase too. CodeCombat is doing great in motivating learners, as the players have good control over the game, e.g. being clan member, editing avatar, play desired level, choice of programming language or contributing to the project itself, which is open source. Also there is a feature specially added for teachers' use, where they can create a class for their students and teach them programming in a fun way.

Currently the available programming languages to learn and practice are as follows: Python, JavaScript, CoffeScript, Clojure and Lua. Unfortunately the choice of available languages is limited to scripting ones. From my personal experience with this learning environment, Python was particularly easy to start coding. Surprisingly there are wide range of languages to choose, to be exact more than 50. This is excellent as it will allow anyone from anywhere in the world to learn programming in their native language, if they don't know or prefer English. CodeCombat heavily relies on open source code and high

quality services, with more than 40 open source languages and technologies used in the development process of the web application. The core languages are CoffeeScript, Jade, Sass and Markdown. One more thing very important to note is that the code editor used is called ‘Ace’, which is very customizable, open source and embeddable in any web environment.

CodeCombat is a very interactive learning environment, which uses good visualization and animations in the games, therefore very attractive option especially for the younger audience. Navigation is pretty easy and it is obvious where each button links to. At first this application may seem complicated for new comers, but it gets very easy to use after few tries. Overall CodeCombat presents very interesting, attractive and interactive environment to learn programming, while having fun. Very similar to ‘game-based metrics’ approach. Also their source code is available on GitHub. Not suitable for mobile devices.

5.2.2 Appearance

For more images please refer to Appendix C.

5.3 Product 2 – CodinGame

5.3.1 Review

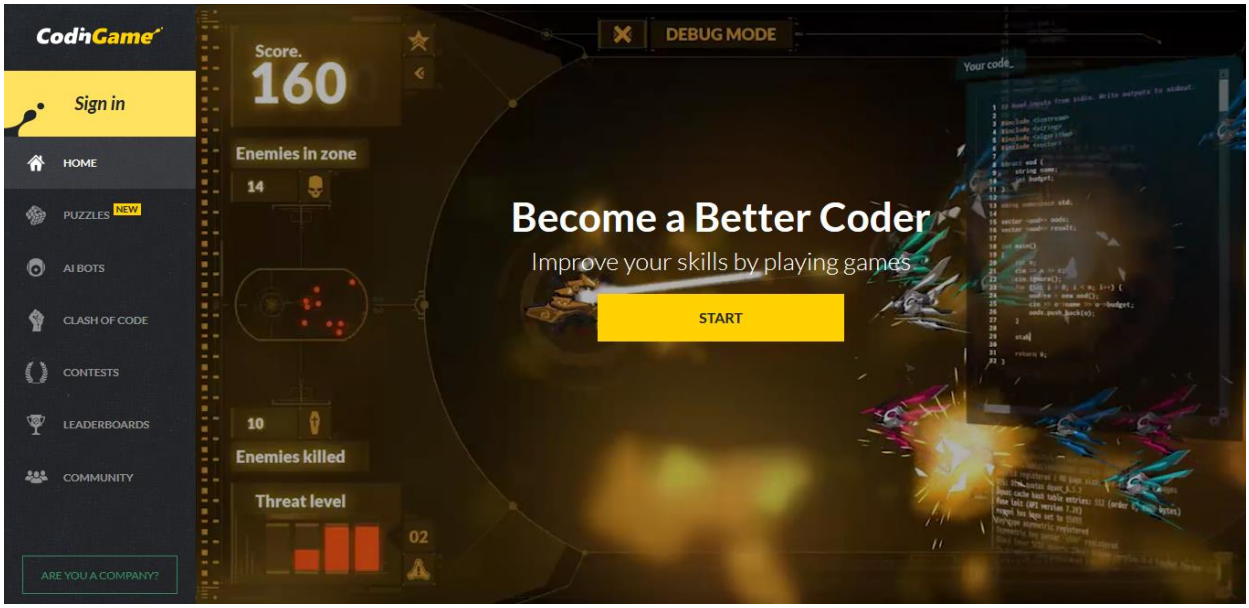
CodinGame is a web-based innovative learning environment with ‘game-based’ approach. They have the goal to help learning and improving programming skills of the coders, no matter if they are qualified professionals or just starting to program. There are many features in CodinGame and the games are categorized in four levels of difficulty – easy, medium, hard and very hard. So they present a challenging environment to the programmers. The games are called puzzles and you play them in three different modes – Single player (solving algorithms), Optimization (getting the highest possible score), Code golf (achieving the shortest possible code). Also another feature called AI Bots, allows the programmers to build smart bots to ‘fight their opponents’. CodinGame is a very challenging environment, because the

players also have the option to participate in challenges with other players to solve algorithmic problems as fast as possible. Something very interesting to mention is the option to participate in contests with great prizes for the top 10 programmers. Finally if a player becomes consistent and very good in solving problems, they have the chance to be offered a real job, which is the greatest motivation to start learning programming.

There are more than 20 programming languages to practice, which makes it suitable for anyone. Unfortunately only English and French are available to choose from, which is very disappointing. What about someone who wants to learn programming, but doesn't speak either of those languages? This is an issue that needs more attention. Again the code editor in CodinGame is 'Ace', which is important to note for later investigation on its use. However this product is not open source and it is not available to the community, whereas CodeCombat is and heavily reliant on community's help for improvements.

Navigation is very straightforward, however the application is neither easy nor difficult to use. CodinGame unfortunately has a bit more confusing interface than CodeCombat, where the game, instructions, code editor and test cases are in the same page. Basically it is overcrowded with too many features. Still a very good learning environment to practice programming, especially for skilful programmers. CodinGame could be a bit difficult for newcomers into the field. Not suitable for mobile devices.

5.3.2 Appearance



For more images please refer to Appendix C.

5.4 Product 3 – CodeMonkey

5.4.1 Review

CodeMonkey is an online game for teaching and learning basic programming concepts. It is intended for the very young audience from the age of 9 and up, but it is also suitable for older learners with no previous knowledge or little experience in programming. CodeMonkey follows the 'game-based' approach to motivate the learners. Basic concepts are covered throughout the learning process by playing the game, e.g. objects, function calls, arguments, loops, variables, arrays, for loops, function definitions, until loops, if and if-else conditions and boolean operators. The game is very simple and entertaining, where the main goal is to control the monkey to walk and catch the bananas. The players are encouraged to get 3 stars as achievement for each game level (e.g. 1 – to catch the banana; 2 – use of all concepts taught to complete the game; 3 – short code and to the point). Other than playing the basic game, the learners who have active subscription have the option to use features like 'challenge builder' and 'skill challenges'. Moreover teachers and organizations can use CodeMonkey to create classes and teach their students programming in an interactive and fun way, but again subscription is required.

The game is translated into 18 languages, making it accessible to a wider audience. The only programming language used by the learners is CoffeeScript, which compiles to JavaScript and resembles the way we write in English, due to its simple syntax. Looks like the open source code editor 'Ace' helps many of the similar environment used to practice and learn programming. 'Ace' is used by CodeMonkey too.

Unfortunately this learning environment is not available for mobile devices too. But we have to bear in mind that, coding in a mobile device wouldn't be very pleasant and easy. Programming is better done using computers, so a web-based application is very suitable for this purpose. CodeMonkey is designed very attractively and 'fun', also it is very easy to navigate and use. Overall a very good product for beginners to learn programming in a very entertaining and engaging way.

5.4.2 Appearance

For more images please refer to Appendix C.

5.5 Key issues to use in the design and implementation

After the investigation and review of current successful products several issues were taken under consideration. Hopefully applying or avoiding few of them, will have a positive impact on the intended final product. Key issues and aspect to consider are:

Positive:

- The overall structure must be consistent;
- The navigation must be available on all pages and easy to navigate;
- Access to home page from other pages must be as easy as one click;
- Straightforward registration and login;
- Any text included must be conformant to standards (size, colour, font and etc.);
- Any imagery and animation should be considered carefully (relevance, copyright and etc.);
- Any sound/music could be attractive, but should be considered carefully (irritation, copyright);
- The colour scheme used must be chosen carefully (accessibility, standards and etc.);
- Simpler layout and content would be helpful;
- The product must be easy to use;
- The products must be accessible;
- The product must be interactive;
- The product could use the ‘game-based’ approach;
- The product could have more than one language or way to teach;
- The use of open source technologies would be helpful;

Negative:

- Avoid inconsistency;
- Avoid content that may cause confusion and irritation;
- Avoid over using imagery, animations and sounds;

5.6 Conclusions

To conclude all of the products were developed to teach and help learning to program by introducing interactive game-based environments and were presented up to good standard. They all have the same goal – to motivate learning and make programming fun and engaging. This is achieved by solving problems to complete challenging, competitive and fun games. Learning to code, while playing games is a wonderful idea especially for the younger audience.

Again programming must be easy and fun, not difficult and boring. Aforementioned examples really succeeded in attracting children and adults into programming, no matter of their experience and future career intentions. Programming definitely shouldn't be perceived as a way to career in the IT industry, but more as an essential skill and a way to improve a person's problem-solving, critical-thinking and other soft skills. Basically being able to write program and think analytically shouldn't mean the person has to work as a programmer.

We can categorize the reviewed products according their difficulty level as: CodeMonkey more suitable for beginners, CodeCombat for beginners and intermediate programmers and finally CodinGame for experienced and advanced programmers.

6 REQUIREMENTS ANALYSIS

6.1 Introduction

In order to successfully complete the design phase and finally move on to product development, firstly we have to carefully gather and analyse the requirements. For the purpose a questionnaire was created to get information about students' expectations from this type of product and measure their experience and skills in programming. The results from the questionnaire were already discussed to approve the feasibility of the project (see Chapter 2), as well as the further actions to be carried out. For a detailed view, please refer to the questionnaire and summary of responses (see Appendix B).

Also a significant amount of information was derived by reviewing existing products, which aimed to motivate learners and ease learning of programming by 'game-based' approach' (see Chapter 5). The whole review process was very helpful to determine the positive and negative issues arising, and to identify the functional and non-functional requirements.

6.2 Target audience

The target audience of the end product could be anyone, as it is a web-based learning environment and will be accessible from any location, as long as they have access to computer and Internet. Mainly the product is intended to be used by Year 1 students enrolled on computer programming courses. Students who experience difficulties in learning programming and where the traditional way of teaching cannot motivate them. Also the product will be suitable for both male and female students, in order to have gender diversity.

6.3 Statement of requirements

The statement of requirements consists of two subsections: Functional requirements (FR) to show what the product is intended to achieve and Non-functional requirements (NFR) to show how they will be achieved and possible constraints that may arise. Each of the requirements will be categorized as 'high' or 'low' level priority using the MoSCoW methodology. Requirements with high priority will be indicated with 'MUST' or 'SHOULD' as they are the most significant ones to be achieved. Also requirements with low priority, which are not necessarily significant to achieve, but might be considered if time permits will be indicated with 'COULD' or 'WOULD'. A list of functional and non-functional requirements is below:

6.3.1 Functional requirements

FR	Description	MoSCoW
1.	User to register successfully	MUST
2.	User to login successfully using their ID and Password	MUST
3.	User login details to be kept in a Cookie and Session	MUST
4.	User must have access to learning material after authorization	MUST
5.	User to logout from the system successfully	MUST
6.	System to retrieve data from MS SQL Server database	MUST
7.	System to alter data in MS SQL Server database	MUST
8.	System to insert data into MS SQL Server database	MUST
9.	User must be able to navigate to desired pages and default content	MUST
10.	User must be able to choose practice activity (programming quiz or game)	MUST
11.	User to start a quiz	MUST
12.	User to play a programming game	MUST
13.	System to generate questions in a random order	SHOULD
14.	User to check personal progress (level, experience and results)	SHOULD
15.	System to keep progress data in MS SQL Server database	SHOULD
16.	User to check solutions	SHOULD
17.	User to change account password	SHOULD
18.	Bugs in games and quizzes	COULD
19.	User to find information in F.A.Q. section	COULD
20.	User to get in touch with the admin/owner/developer (email or phone)	COULD
21.	User to penetrate and threaten the system using hacking tricks (injection tricks)	WON'T

6.3.2 Non-functional requirements

NFR	Description	MoSCoW
1.	Usable – The application must have user-centric design and must be kept clear and simple. The application must be easy to use and navigate through pages back and forward.	MUST
2.	Easy to learn - Consistent interface, with additional instructions for advanced task and actions	MUST
3.	Easy to understand – Simple and unambiguous content	MUST
4.	Engaging – Must be pleasant, comforting and satisfying to use (interactive content and games).	MUST
5.	Security – Unauthorized access and manipulation to the system must be prevented.	MUST
6.	Privacy – Personal information must be protected and data to be stored securely.	MUST

	(e.g. email and password)	
7.	Accessible - The user should be able to reach the website from any browser and platform. Also any user with disabilities should be able to use the application. (appropriate colour scheme, compliant to standards)	SHOULD
8.	Performance – The application should perform well. (loading, request/response). Also the application should be effective and efficient.	SHOULD
9.	Error tolerant – Errors prevented and recovered caused by user's interactions	SHOULD
10.	Reliable – The application must work to a certain extent to satisfy user's needs	SHOULD

7 DESIGN OF ……?

Discuss your creative methods

User testing of these

Discuss the different alternatives and explaining why you chose the ones you did.

Visual design: e.g.

- Include some images for illustration of key points
- Refer to appendices containing story boards and more.

Technical design. e.g.

- Use accepted notation
- UML use case, sequence, class, interaction, state
- Explain how ERD etc were arrived at
- Use un-accepted notation – rich pictures
- Navigational structure of product.
- Show simple diagrams
- Refer to appendices containing detail technical stuff.

LSEPi in your design

Justify your decisions referring back to Literature review.

8 DEVELOPMENT OF…….

Based on your designs now discuss how you implemented then

Did you have to make changes?

Were there any technical problems you hadn't foreseen?

What advanced coding did you create / adapt?

Put relevant snippets in the text, refer to more complex examples in the appendix

Explain what and why you did what you did.

Don't give lots of boring detail explain the skilful decisions and techniques you used and implemented

Discuss any interim testing you did.

LSEPi in your development

Refer back to the research you did.

9 TESTING

Your approach to testing

LSEPi in your testing

Your white/black box testing.

Run accessibility tests.

Test schedules and results – may continue in an appendix

10 EVALUATION

Your approach to evaluation

LSEPi in your evaluation

Evaluation by users and clients

11 CONCLUSION

Your review of the product

Your review of the process of the project.

How the project may be taken forward by you or others.

Summarise your key thoughts about the project.

All projects are open to criticism, it is better that this comes from you before anyone else has a chance to criticise.

Critical evaluation of…

Your **process** – the entire project from start to finish

Your **product** – objective evaluation of the thing you made

Your **person** - yourself – what did you bring to the project, how have you changed through this experience, what have you learned – you may resort to first person grammar here but you get extra brownie points if you can do this in the third person.

BIBLIOGRAPHY

Anderson, R., Ordóñez, R., Pham, P. and Tribelhorn, B. (2015) A Data Programming CS1 Course, In *SIGCSE '15 Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, ACM New York, NY, USA, pp. 150-155, [online] Available at: https://homes.cs.washington.edu/~mernst/pubs/data-programming-sigcse2015.pdf (Accessed 19 February 2016).

Ball, C., Moller, F. and Pau, R. (2012) The Mindstorms effect: A gender analysis on the influence of LEGO Mindstorms in computer science education, In *Proceedings of the 7th Workshop in Primary and Secondary Computing Education*, ACM, pp. 141-142, [online] Available at: http://cs.swan.ac.uk/~csfm/Pubs/wipsce12.pdf (Accessed 20 February 2016).

Binas, M. and Pietrikova, E. (2014) Useful recommendations for successful implementation of programming courses, In *Emerging eLearning Technologies and Applications (ICETA), 2014 IEEE 12th International Conference on*, IEEE, pp. 397 - 401.

Butler, M. and Morgan, M. (2007) Learning challenges faced by novice programming students studying high level and low feedback concepts, In *Providing choices for learners and learning. Proceedings ascilite Singapore 2007*, pp. 99-107, [online] Available at: http://www.ascilite.org/conferences/singapore07/procs/butler.pdf (Accessed 19 February 2016).

Chandramouli, M., Zahraee, M. and Winer, C. (2014) A Fun-Learning Approach to Programming: An Adaptive Virtual Reality (VR) Platform to Teach Programming to Engineering Students, In *Electro/Information Technology (EIT), 2014 IEEE International Conference on*, IEEE, pp. 581 - 586.

Cummins, J., Azhar, M. and Sklar, E. (2008) *Using Surveyor SRV-1 Robots to Motivate CS1 Students*, AAAI, pp. 23-27, [online] Available at: https://www.aaai.org/Papers/Workshops/2008/WS-08-02/WS08-02-005.pdf (Accessed 20 February 2016).

Drumond, R., Brandao, A. and Salles, C. (2014) Wanda: A Framework to Develop Card Based Games to Help Motivate Programming Students, *Computer Games and Digital Entertainment (SBGAMES), 2014 Brazilian Symposium on*, pp. 158-164.

Esteves, M. and Mendes, A. (2004) A simulation tool to help learning of object oriented programming basics, In *Frontiers in Education, 2004. FIE 2004. 34th Annual*, IEEE, pp. F4C-7-12 Vol. 2.

Feldgen, M. and Clua, O. (2004) Games As A Motivation For Freshman Students To Learn Programming, In *34th ASEE/IEEE Frontiers in Education Conference*, IEEE, pp. S1H-11-S1H-16.

Garcia-Beltran, A. and Martinez, R. (2006) Web Assisted Self-assessment in Computer Programming Learning Using AulaWeb, *Int. J. Engng Ed.*, 22(5), pp. 1063-1069, [online] Available at: http://www.ijee.ie/articles/Vol22-5/18_ijee1790.pdf (Accessed 20 February 2016).

Gomes, A. and Mendes, A. (2007) An environment to improve programming education, In *Proceedings of the 2007 International Conference on Computer Systems and Technologies*, ACM, p. 88, [online] Available at: https://www.researchgate.net/publication/220795506_An_environment_to_improve_programming_education (Accessed 19 February 2016).

Hawi, N. (2010) Causal attributions of success and failure made by undergraduate students in an introductory-level computer programming course, *Computers & Education*, 54(4), pp. 1127-1136.

Hijon-Neira, R., Velazquez-Iturbide, A., Pizarro-Romero, C. and Carrico, L. (2014) Serious games for motivating into programming, In *Frontiers in Education Conference (FIE), 2014 IEEE*, IEEE, pp. 1-8.

Hwang, W., Wang, C., Hwang, G., Huang, Y. and Huang, S. (2008) A web-based programming learning environment to support cognitive development, *Interacting with Computers*, 20(6), pp. 524-534.

Jenkins, T. (2001) The motivation of students of programming, *SIGCSE Bull.*, 33(3), pp. 53-56.

Jiau, H., Chen, J. and Ssu, K. (2009) Enhancing Self-Motivation in Learning Programming Using Game-Based Simulation and Metrics, *IEEE Trans. Educ.*, 52(4), pp. 555-562.

Jurado, F., Redondo, M. and Ortega, M. (2012) eLearning standards and automatic assessment in a distributed eclipse based environment for learning computer programming, *Comput Appl Eng Educ*, 22(4), pp. 774-787.

Kelleher, C. and Pausch, R. (2007) Using storytelling to motivate programming, *Communications of the ACM*, 50(7), p. 58.

Konecki, M., Kadoic, N. and Piltaver, R. (2015) Intelligent assistant for helping students to learn programming, In *Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2015 38th International Convention on*, IEEE, pp. 924 - 928.

Kuo-En Chang, Bea-Chu Chiao, Sei-Wang Chen, and Rong-Shue Hsiao, (2000) A programming learning system for beginners-a completion strategy approach, *IEEE Trans. Educ.*, 43(2), pp. 211-220.

Lahtinen, E., Ala-Mutka, K. and Järvinen, H. (2005) A study of the difficulties of novice programmers, *SIGCSE Bull.*, 37(3), p. 14.

Law, K., Lee, V. and Yu, Y. (2010) Learning motivation in e-learning facilitated computer programming courses, *Computers & Education*, 55(1), pp. 218-228.

Leavens, G. (2008) Use concurrent programming models to motivate teaching of programming languages, *ACM SIGPLAN Notices*, 43(11), p. 93.

Lykke, M., Coto, M., Mora, S., Vandel, N. and Jantzen, C. (2014) Motivating programming students by problem based learning and LEGO robots, In *Global Engineering Education Conference (EDUCON), 2014 IEEE*, IEEE, pp. 544 - 555.

Markham, S. and King, K. (2010) Using personal robots in CS1: experiences, outcomes, and attitudinal influences. In *Proceedings of the fifteenth annual conference on Innovation and technology in computer science education (ITiCSE '10).*, New York, NY, USA, ACM, pp. 204-208, [online] Available at: http://ims.mii.lt/ims/konferenciju_medziaga/ITiCSE'10/docs/p204.pdf (Accessed 19 February 2016).

Matthews, R., Hin, H. and Choo, K. (2014) Learning Object to Enhance Introductory Programming Understanding: Does The Size Really Matter?, *TOJET: The Turkish Online Journal of Educational Technology*, 13(1), pp. 174-183, [online] Available at: http://www.tojet.net/articles/v13i1/13115.pdf (Accessed 20 February 2016).

Nunes de Barros, L., Paula dos Santos Mota, A., Valdivia Delgado, K. and Megumi Matsumoto, P. (2005) A tool for programming learning with pedagogical patterns, In *Proceedings of the 2005 OOPSLA workshop on Eclipse technology eXchange (eclipse '05)*, ACM, pp. 125-129, [online] Available at: http://dl.acm.org/citation.cfm?id=1117722&dl=ACM&coll=DL&CFID=582778302&CFTOKEN=77452920 (Accessed 20 February 2016).

Resnick, M., Maloney, J., Monroy-Hernandes, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B. and Kafai, Y. (2009) Scratch: Programming for all, *Communications of the ACM*, 52(11), pp. 60-67, [online] Available at: http://web.media.mit.edu/~mres/papers/Scratch-CACM-final.pdf (Accessed 19 February 2016).

Thibodeau, P. (2012) IT jobs will grow 22% through 2020, says U.S., *Computerworld*, [online] Available at: http://www.computerworld.com/article/2502348/it-management/it-jobs-will-grow-22--through-2020--says-u-s-.html (Accessed 19 February 2016).

APPENDIX A - Project Proposal

An Interactive Web Application to motivate learning of programming

Gyokay Ali

BSc (Hons) Software Engineering

000762233

1 Overview

The project will investigate the different ways of learning programming as well as the key motivating factors affecting learning among first year undergraduate CIS students. Nowadays Software Engineering is one of the most sought after professions all over the world, therefore as future IT specialists the students should possess very good applicable practical skills, in addition to the theoretical knowledge. Without an adequate motivation it is typically very difficult for new students to develop good programming skills, which requires a lot of practice.

"*The challenge of preparing graduates for a fast-changing work environment calls for the development of an effective learning framework. In this regard, technology is often used to enhance students' engagement in learning and their academic achievement*" (Carle et al., 2009, Roth et al., 2008, Tan, 2006 and Yu et al., 2006).

"*However, students nowadays will easily lose enthusiasm and interests in learning computer programming, especially when they experience repetitive failure in practising on their own. The need to improve the teaching and learning of computer programming thus calls for special attention to the factors affecting students' learning motivation*" (Jenkins, 2001).

The proposed web application will let the students to practice a chosen programming language (Java, C#, SQL or JavaScript) on their own. The exact programming language is not decided yet, but it will be clear after a questionnaire is sent out to the students. The application will involve some quizzes and small games, which will make the time spent more fun and motivating. Also there will be implemented a score and progress functionality. At first everything will be very easy for the beginners, but it will get more difficult as the student progress through the learning environment by collecting points.

Apparently the development of the application or at least a working prototype will not be an easy task. However I believe that I have the necessary technology skills in order to have a successful final product. The courses Web & Application Development, Advanced Programming and Database Applications Technologies from 2nd Year were very pragmatic and useful. The skills required for this project's application are ASP.Net with C#. SQL, OO concepts and little bit of HTML and CSS. First class grades acquired from the courses above motivate me that this project will be a successful one.

In order to evaluate the effectiveness of this application, it will be tested by students to see if it would have any impact on their learning and if it helps to get them more confident in programming.

Keywords: learning, motivation, e-learning, programming, self-study, education, interactivity

2 Aim

The aim of this project is to develop an innovative learning application/website for Year 1 CIS student to learn programming basics by "doing". The long term goal is to motivate learners.

3 Objectives

3.1 An Investigation Report [16.0]

3.1.1 The investigation report will consist of literature review and review of existing similar products.

3.1.2 Literature Review [11.0]

3.1.2.1 Search for articles, journals and books [2.0]

3.1.2.2 Read the related material [7.0]

3.1.2.3 Write and finish off the Literature review [2.0]

3.1.3 Existing Products Review [5.0]

3.1.3.1 Search for similar products [1.0]

3.1.3.2 Test, compare and evaluate the products [2.0]

3.1.3.3 Write and finish off the Product review [2.0]

3.2 Design documentation [14.0]

3.2.1 Several design documents will be produced as a result. It includes the statement of requirements, series of design diagrams and the user interface design.

3.2.2 Statement of Requirements [4.0]

3.2.2.1 Create a questionnaire [1.0]

3.2.2.2 Analyse results from the questionnaire [1.0]

3.2.2.1 Gather and write the requirements [2.0]

3.2.3 UML Diagrams [3.0]

3.2.3.1 Design Use-Case diagrams [1.5]

3.2.3.2 Design Class diagrams [1.5]

3.2.4 Entity Relationship Diagram (ERD) [1.0]

3.2.5 User Interface Design [6.0]

3.3 Product Implementation [17.5]

3.3.1 This objective involves creating and populating the database to be used and development of the product.

3.3.2 Database Implementation [3.5]

3.3.2.1 Create the tables [2.0]

3.3.2.2 Apply normalization rules [1.0]

3.3.2.3 Populate the tables [0.5]

3.3.3 Web application development [14.0]

3.4 Testing [7.5]

3.4.1 The testing documentation involves testing the web application as well as the web application

3.4.2 Database Testing [3.0]

3.4.2.1 Create test plan [1.0]

3.4.2.2 Test the database [0.5]

3.4.2.3 Review the results [0.5]

3.4.2.4 Fix if necessary [1.0]

3.4.3 Web application Testing [4.5]

3.4.3.1 Create test plan [1.0]

3.4.3.2 Test the web application [1.0]

3.4.3.3 Review the results [0.5]

3.4.3.4 Fix if necessary [2.0]

3.5 An Evaluation Report [5.0]

3.6 A Reflection Report [5.0]

Project estimated duration [65.0]

4 Legal, Social, Ethical and Professional

This project's final product will be developed only for the needs of Year 1 undergraduate students, so it will not require any authorization from the university (as long as they are over 18). Therefore there won't be any underage student involved in the process of requirements gathering and testing. But as the requirements gathering involves handing questionnaires to collect relevant information, a permission by the Research Ethics Committee will be required. Also the questionnaire forms must be designed and written in compliance with the Data Protection Act (1998).

The confidentiality of sensitive data must be signed off by the end user. The privacy aspects will be considered and an encryption will be required when retrieving and saving data to the database. To prevent misuse and going against the law, at the end of the development all collected data will be destroyed.

The project will involve comparing similar products available (functionality and design). All the material (images, sounds and content) to be used in the web application must comply with the Copyright, Designs and Patents Act (1988). Also this involves properly citing and referencing all the sources used while writing the Literature Review or simply the whole project.

To prevent any social issues it will not be required from the users to enter their religion, sex, gender and age, when they register.

A professional manner will be followed by taking under consideration BCS and IEEE standards. Properly planning and managing the project will prove this.

5 Planning (see appendix A)

As the life of the current plan is unpredictable and open to changes (probably many times) at some point, this will require the Rapid Application Development (RAD) Prototyping approach using the 'MoSCoW' methodology to control the process in an evolutionary way. Building an initial prototype as early as possible will be a very significant step. Also an iterative concept will be required in order to test and get feedback from the end-user.

In general, RAD approaches to software development put less emphasis on planning tasks and more emphasis on development. In contrast to the waterfall model, which emphasizes rigorous specification and planning, RAD approaches emphasize the necessity of adjusting requirements in reaction to knowledge gained as the project progresses (Wikipedia, The Free Encyclopedia, 2015).

Using UML will be a big part of the design phase. Its role is to show the actions of the users involved in the system and help to create the prototype by simply designing the Use Case and Class diagrams.

The project objectives will be managed by creating a Gantt chart in MS Project. Adjustments would be made if required during the length of the project. Monitoring this plan at all times will reduce the risks of fail and keep up with deadlines.

6 Initial References

Websites:

BCS, (2011) Code of Conduct, [Online], Accessed on 24 Oct 2015, http://www.bcs.org/category/6030#profession

Data Protection Act 1998, (1998) London: HMSO, [Online], Accessed on 24 Oct 2015, http://www.legislation.gov.uk/ukpga/1998/29

IEEE, (2012) Publications & Standards, [Online], Accessed on 24 Oct 2015, http://www.ieee.org/publications_standards/index.html

Copyright, Designs and Patents Act 1988, (1988), [Online], Accessed on 24 Oct 2015, http://www.legislation.gov.uk/ukpga/1988/48/contents

Rapid application development. (2015, October 15). In *Wikipedia, The Free Encyclopedia*. Retrieved 13:10, October 24, 2015, from https://en.wikipedia.org/w/index.php?title=Rapid_application_development&oldid=685812708

Articles and Conference Papers:

A.C. Carle, D. Jaffee, D. Miller (2009). Engaging college science students and changing academic achievement with technology: A quasi-experimental preliminary investigation: *Computers & Education*, 52, (pp. 376–380)

Jenkins, T. (2001). The motivation of students of programming. In *Proceedings of ITiCSE 2001: The 6th annual conference on innovation and technology in computer science education* (pp. 53–56).

O. Tan (2006). Development of a thinking programme for engineering students: *Innovations in Education and Teaching International*, 43 (3) (pp. 245–259)

V. Roth, V. Ivanchenko, N. Record (2008). Evaluating student response to WeBWorK, a web-based homework delivery and grading system: *Computers & Education*, 50, pp. 1462–1482

Yu, Y. T., Poon, C. K., & Choy, M. (2006). Experiences with PASS: Developing and using a Programming Assignment aSsessment System. In *Proceedings of QSIC 2006: The 6th international conference on quality software* (pp. 360–365).

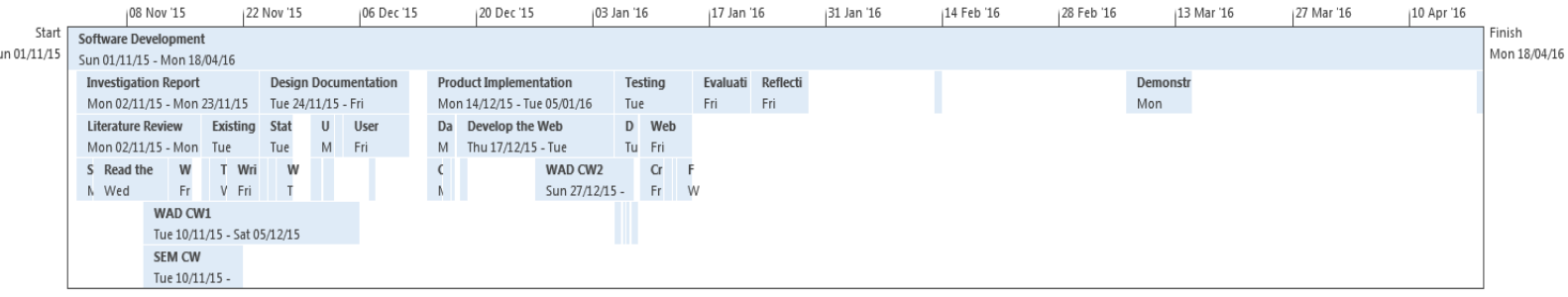
Appendix A – Gantt Chart

	Investigation Report	**17 days**	**Mon 02/11/15**	**Mon 23/11/15**
	Literature Review	**12 days**	**Mon 02/11/15**	**Mon 16/11/15**
	Search for reading material	2 days	Mon 02/11/15	Tue 03/11/15
	Read the related material	7 days	Wed 04/11/15	Thu 12/11/15
	Write Literature Review	2 days	Fri 13/11/15	Sun 15/11/15
	Existing Product Review	**5 days**	**Tue 17/11/15**	**Mon 23/11/15**
	Search for similar products	1 day	Tue 17/11/15	Tue 17/11/15
	Test, compare and evaluate findings	2 days	Wed 18/11/15	Thu 19/11/15
	Write the Existing Products Review	2 days	Fri 20/11/15	Mon 23/11/15

Task Mode	Task Name	Duration	Start	Finish
	Design Documentation	**14 days**	**Tue 24/11/15**	**Fri 11/12/15**
	Statement of Requirements	**4 days**	**Tue 24/11/15**	**Fri 27/11/15**
	Create a questionnair	1 day	Tue 24/11/15	Tue 24/11/15
	Analyse results	1 day	Wed 25/11/15	Wed 25/11/15
	Write requirement	2 days	Thu 26/11/15	Fri 27/11/15
	UML Diagrams	**3 days**	**Mon 30/11/15**	**Wed 02/12/15**
	Design Use Case diagrams	1.5 days	Mon 30/11/15	Tue 01/12/15
	Design Class diagram	1.5 days	Tue 01/12/15	Wed 02/12/15
	ERD	1 day	Thu 03/12/15	Thu 03/12/15
	User Interface Design	6 days	Fri 04/12/15	Fri 11/12/15

Task Mode	Task Name	Duration	Start	Finish
	Product Implementation	**17.5 days**	**Mon 14/12/15**	**Tue 05/01/16**
	Database Implementatio	**3.5 days**	**Mon 14/12/15**	**Thu 17/12/15**
	Create the tables	2 days	Mon 14/12/15	Tue 15/12/15
	Apply normalizatio	1 day	Wed 16/12/15	Wed 16/12/15
	Populate tables	0.5 days	Thu 17/12/15	Thu 17/12/15
	Develop the Web Application	14 days	Thu 17/12/15	Tue 05/01/16
	Testing	**7.5 days**	**Tue 05/01/16**	**Thu 14/01/16**
	Database testing	**3 days**	**Tue 05/01/16**	**Fri 08/01/16**
	Create test plans	1 day	Tue 05/01/16	Wed 06/01/16
	Test the database	0.5 days	Wed 06/01/16	Wed 06/01/16
	Review results	0.5 days	Thu 07/01/16	Thu 07/01/16
	Fix if necessary	1 day	Thu 07/01/16	Fri 08/01/16

Task Mode	Task Name	Duration	Start	Finish
	Web Application testing	**4.5 days**	**Fri 08/01/16**	**Thu 14/01/16**
	Create test plan	1 day	Fri 08/01/16	Mon 11/01/16
	Test the web application	1 day	Mon 11/01/16	Tue 12/01/16
	Review the results	0.5 days	Tue 12/01/16	Tue 12/01/16
	Fix if necessary	2 days	Wed 13/01/16	Thu 14/01/16
	Evaluation Report	5 days	Fri 15/01/16	Thu 21/01/16
	Reflection Report	5 days	Fri 22/01/16	Thu 28/01/16
	Project Delivarables			
	Demonstration of Prototype	1 day?	Mon 07/12/15	Mon 07/12/15
	Initial Contextual Report	1 day?	Fri 18/12/15	Fri 18/12/15
	Interim Report	1 day?	Sat 13/02/16	Sat 13/02/16
	Demonstration and Viva	6 days?	Mon 07/03/16	Mon 14/03/16
	Final Report Upload	1 day?	Mon 18/04/16	Mon 18/04/16
	Other Course Deliverables			
	WAD CW1	20 days	Tue 10/11/15	Sat 05/12/15
	WAD CW2	10 days?	Sun 27/12/15	Thu 07/01/16
	SEM CW	10 days?	Tue 10/11/15	Sat 21/11/15



APPENDIX B - Online Questionnaire

1 Questionnaire Form

*Required

How good do you think you are in Programming? *

Even if it is only one language, that would be fine.

- ○ No knowledge
- ○ Beginner
- ○ Intermediate
- ○ Advanced
- ○ Expert

Which Programming languages do you know already? *

- ☐ Java
- ☐ C#
- ☐ JavaScript
- ☐ Other:

Could you rate your level of confidence with the language/s you have chosen above? *

Please choose 5, if you say "I don't know" or "I am not sure".

	0	1	2	3	4	5	6	7	8	9	10	
Bad/Struggling	○	○	○	○	○	○	○	○	○	○	○	Very good/No problems

Which Programming language would you like to learn next or most? *

- ○ Java
- ○ C#
- ○ JavaScipt
- ○ Other:

Do you have struggle with any of the following? *

- [ ] Operators and Expressions
- [ ] Conditional Statements
- [ ] Loops
- [ ] Arrays
- [ ] Other:

How do you think the traditional teaching of programming might be improved?

Do you think little games implemented in such application will make you more motivated to learn/practice a Programming language? *

- ( ) Yes
- ( ) No

Do you think quizzes in such application will make you more motivated to learn/practice a Programming language *

- ( ) Yes
- ( ) No

What sort of game would be interesting to see in such environment? *

Any particular one that is not mentioned here? Please fill the box for the purpose.

- [ ] Puzzle with drag and drop
- [ ] Crossword puzzle
- [ ] Question challenge
- [ ] Labyrinth
- [ ] Card game

Any particualar game that you would like to see, which is not mentioned above?

What puts you off mostly when you are to study/practice programming? *

- [ ] I am being distracted by other things
- [ ] It is not fun
- [ ] Lack of time
- [ ] I don't see point in learning programming
- [ ] The material taught is not relevant to my expectations
- [ ] Programming seems very complicated to me
- [ ] Other:

Which of the following online learning/self-study environments have you used? *

- [ ] AppInventor
- [ ] Scratch
- [ ] CODE.org
- [ ] CodinGame
- [ ] CodeCombat
- [ ] Other:

Could you rate your experience with the learning environment/s you have chosen above? *

Were they helpful and motivating to you or not? Please chose 5 if you say "I don't know" or "I am not sure".

	0	1	2	3	4	5	6	7	8	9	10	
Not at all	○	○	○	○	○	○	○	○	○	○	○	Very helpful/motivating

What do you do like the most about the learning experience in the environments you chose above?

What was the worst about them?

What would you like to see that wasn't included? How do you think they could be improved to enhance your learning experience?

Do you like the idea and goal of this project? *

	0	1	2	3	4	5	
No	○	○	○	○	○	○	Yes

Would you have any suggestions to improve?

2 Consent Form

A Study carried out by Gyokay Ali,

Final Year BSc Software Engineering student at the University of Greenwich

The aim of this project is to develop an innovative learning application/website for Year 1 CIS students to learn programming basics by "doing". The long term goal is to motivate learners. The proposed web application will let the students to practice a chosen programming language on their own. The exact programming language and motivating methods are not decided yet, but it will be clear after this questionnaire. Basically, help me to help you.

Please read the disclaimer carefully and if you accept them you can move on to the questions. If you have any queries about the questionnaire or just want to know the results and final product at the end, please feel free to contact me at ag306@gre.ac.uk and I will get back to you ASAP.

DISCLAIMER

Thank you for agreeing to participate in this study, which is being conducted by Gyokay Ali, a student at the University of Greenwich, as part of the project for their BSc Software Engineering. The project is supervised by Dr Yasmine Arafa.

The Questionnaire:

- As part of this study, you will be asked to fill out a questionnaire.
- The purpose of the study is to collect data about students' experiences with various programming languages. To check what is a potential method that will motivate them to learn or practice one. And to get opinion and suggestions from the students, if have any possible.
- While you are under no obligation to answer any of the questions we would appreciate if you could provide an answer to all of the questions.
- The entire process should take about 5 minutes at the most of your time.
- If at any time you wish to withdraw from the study, you can do so.

What we will do with your answers:

- The answers you provide will be analysed and the data will be stored.
- We will keep the data for research purposes only. Some of the data we have gathered may be published as part of the project report.
- All data gathered will be destroyed on completion of the project.

For further information about this study please contact:

Gyokay Ali
Email: ag306@gre.ac.uk

Dr Yasmine Arafa
Faculty of Architecture, Computing and Humanities
University of Greenwich,
Old Royal Naval College, London SE10 9LS
Tel: 020 8331 8388
Email: Y.Arafa@gre.ac.uk

Thank you for taking the time to complete this questionnaire

3 Summary of Responses

Figure 1

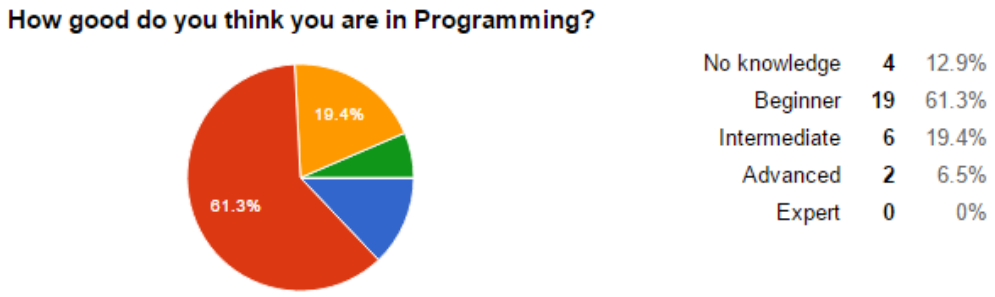


Figure 2

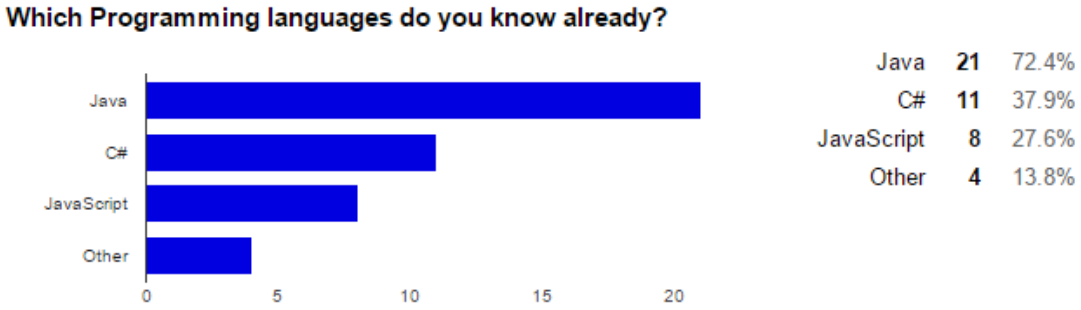


Figure 3

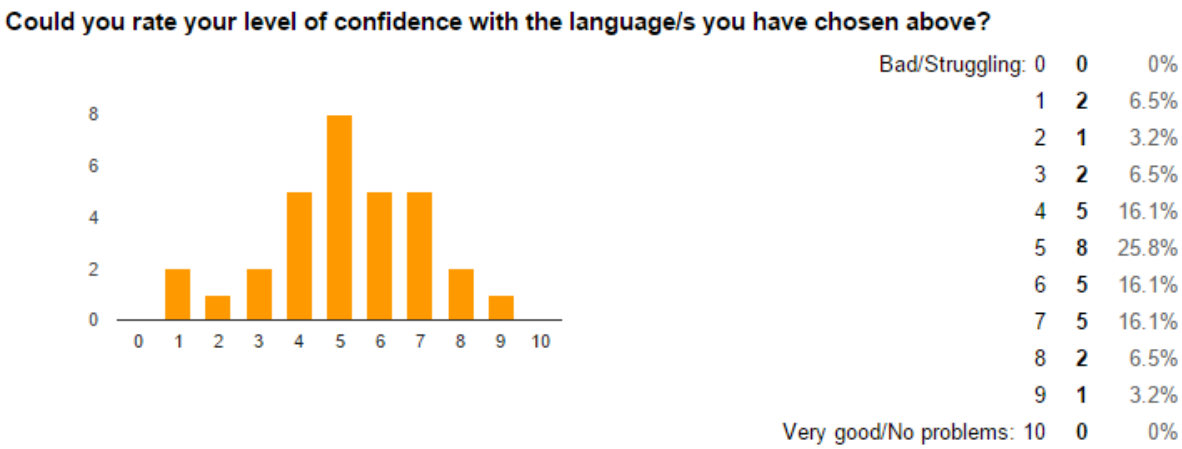


Figure 4

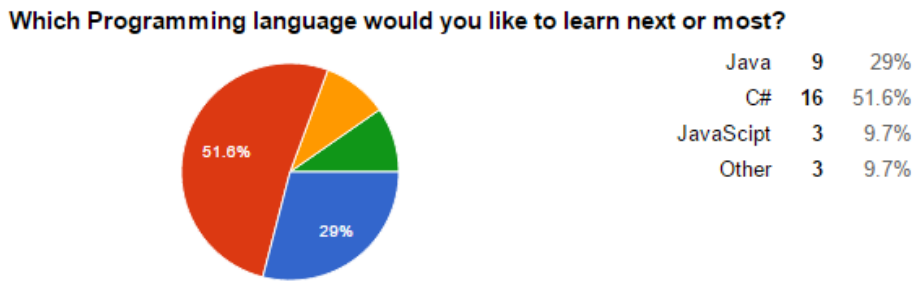


Figure 5

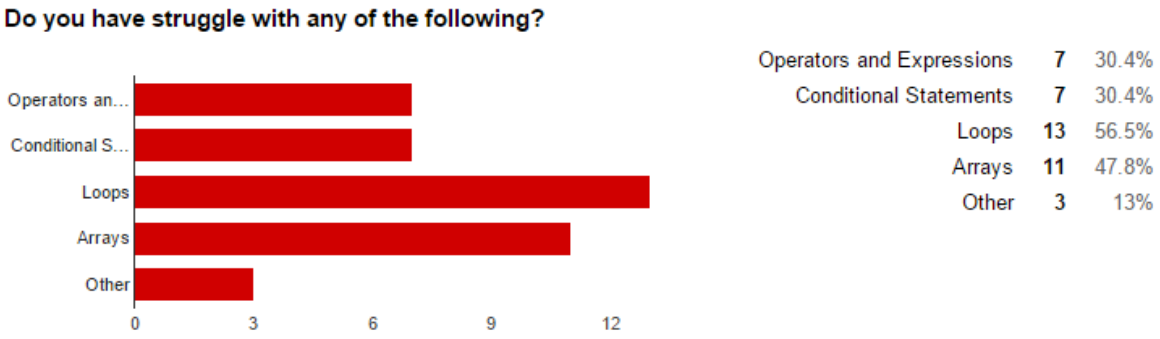


Figure 6

Figure 7

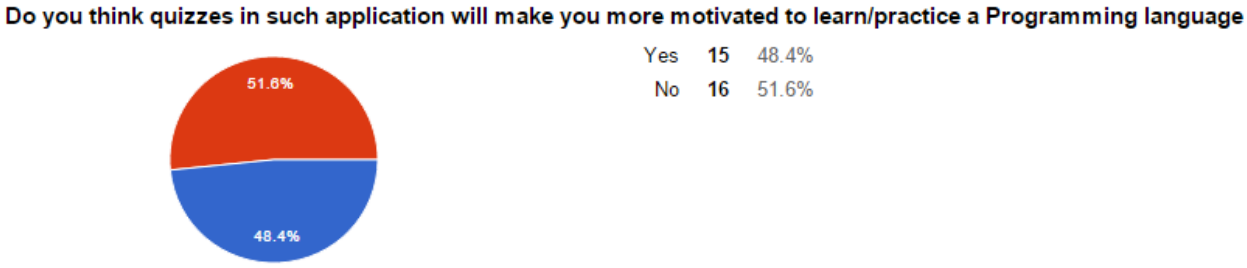
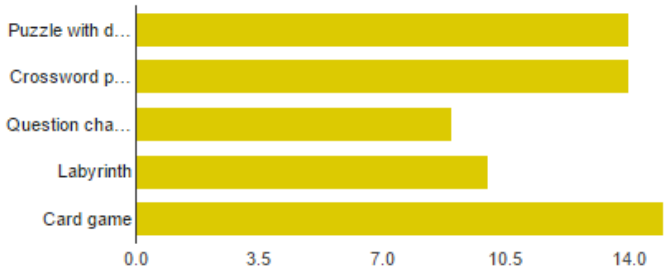


Figure 8

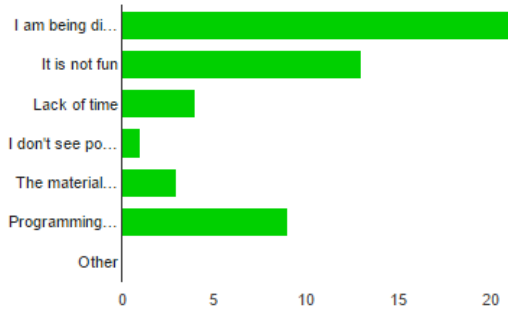
What sort of game would be interesting to see in such environment?



Puzzle with drag and drop	14	45.2%
Crossword puzzle	14	45.2%
Question challenge	9	29%
Labyrinth	10	32.3%
Card game	15	48.4%

Figure 9

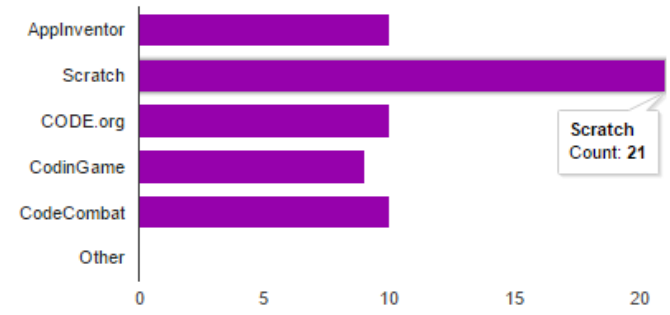
What puts you off mostly when you are to study/practice programming?



I am being distracted by other things	25	80.6%
It is not fun	13	41.9%
Lack of time	4	12.9%
I don't see point in learning programming	1	3.2%
The material taught is not relevant to my expectations	3	9.7%
Programming seems very complicated to me	9	29%
Other	0	0%

Figure 10

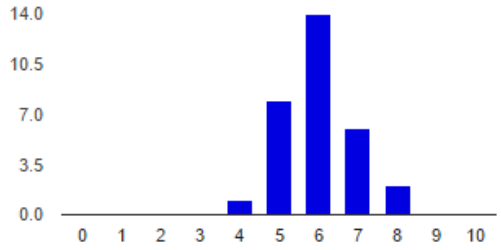
Which of the following online learning/self-study environments have you used?



AppInventor	10	32.3%
Scratch	21	67.7%
CODE.org	10	32.3%
CodinGame	9	29%
CodeCombat	10	32.3%
Other	0	0%

Figure 11

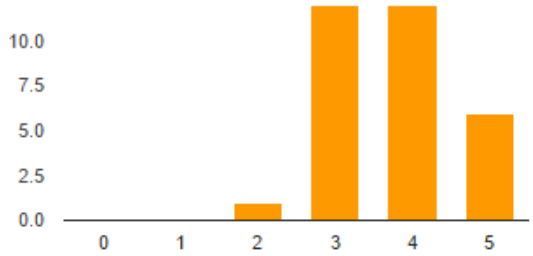
Could you rate your experience with the learning environment/s you have chosen above?



Not at all: 0	**0**	0%
1	**0**	0%
2	**0**	0%
3	**0**	0%
4	**1**	3.2%
5	**8**	25.8%
6	**14**	45.2%
7	**6**	19.4%
8	**2**	6.5%
9	**0**	0%
Very helpful/motivating: 10	**0**	0%

Figure 12

Do you like the idea and goal of this project?

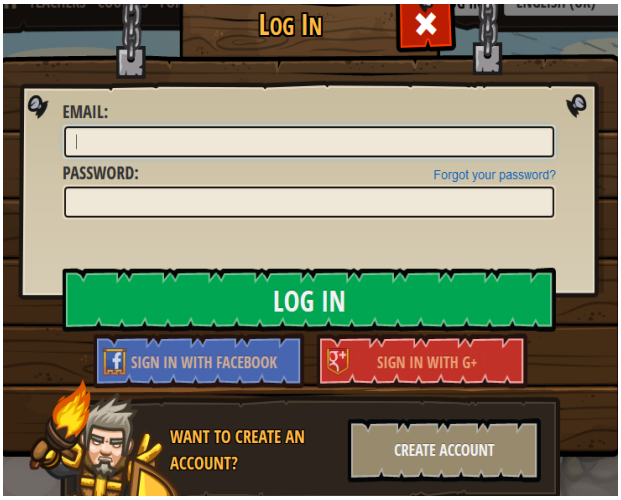


No: 0	**0**	0%
1	**0**	0%
2	**1**	3.2%
3	**12**	38.7%
4	**12**	38.7%
Yes: 5	**6**	19.4%

APPENDIX C - Existing Products Imagery

1 CodeCombat

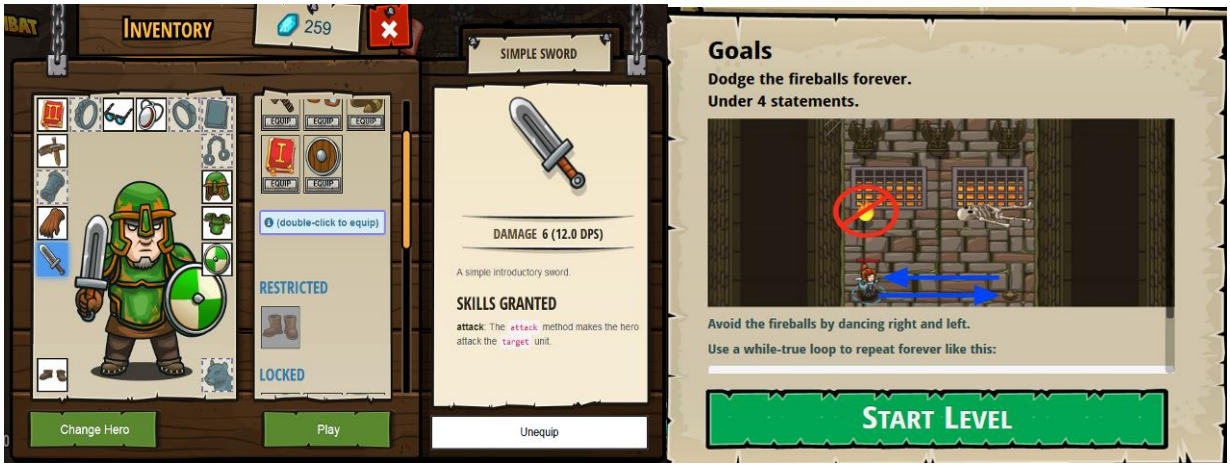
Registration / Login



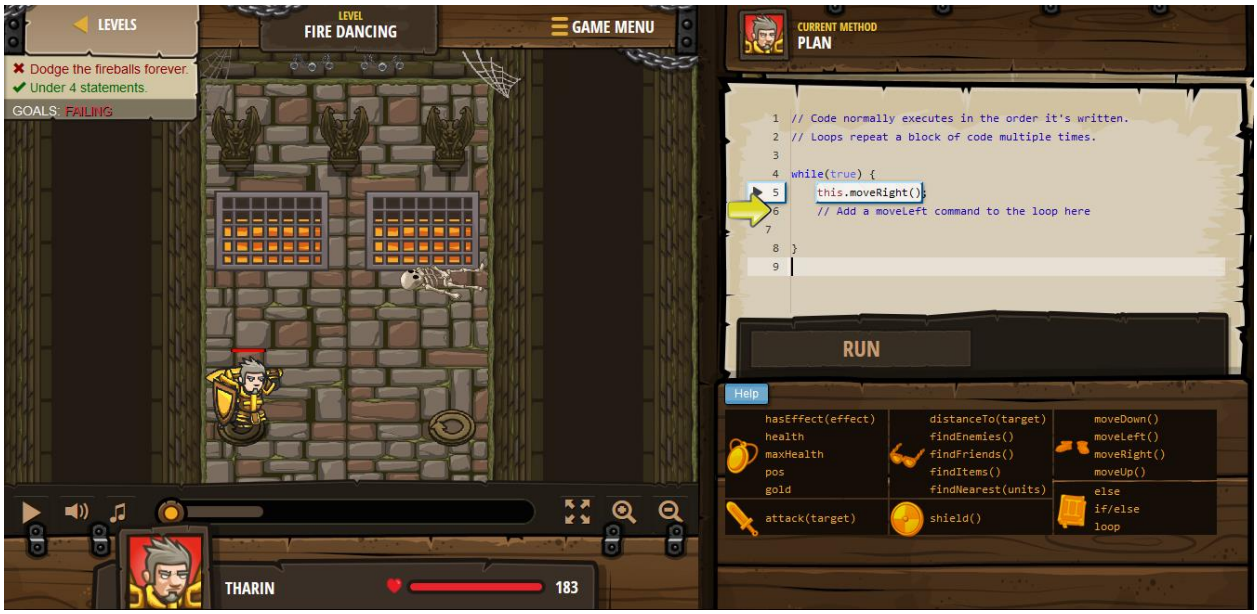
Community

Levels

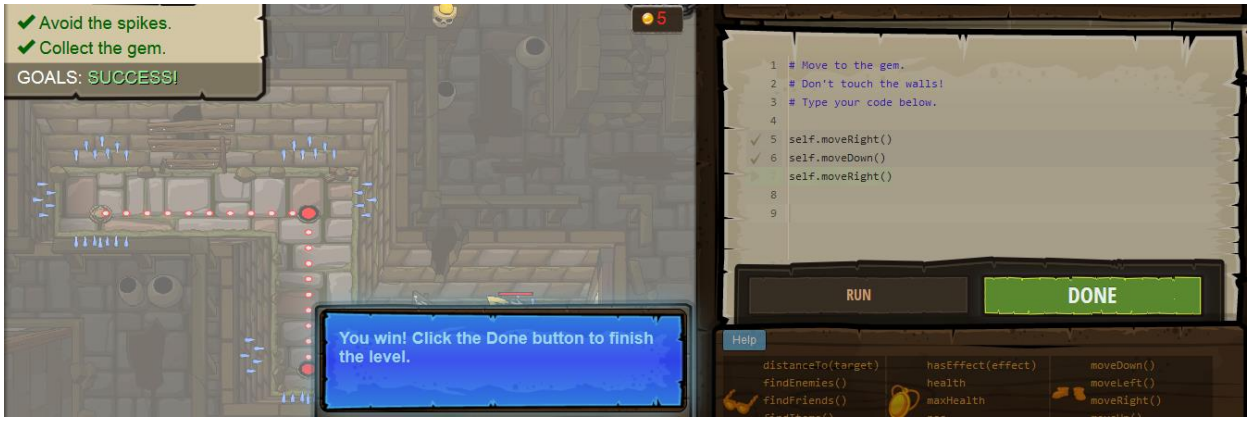
Hero Inventory / Instrunctions



Gameplay



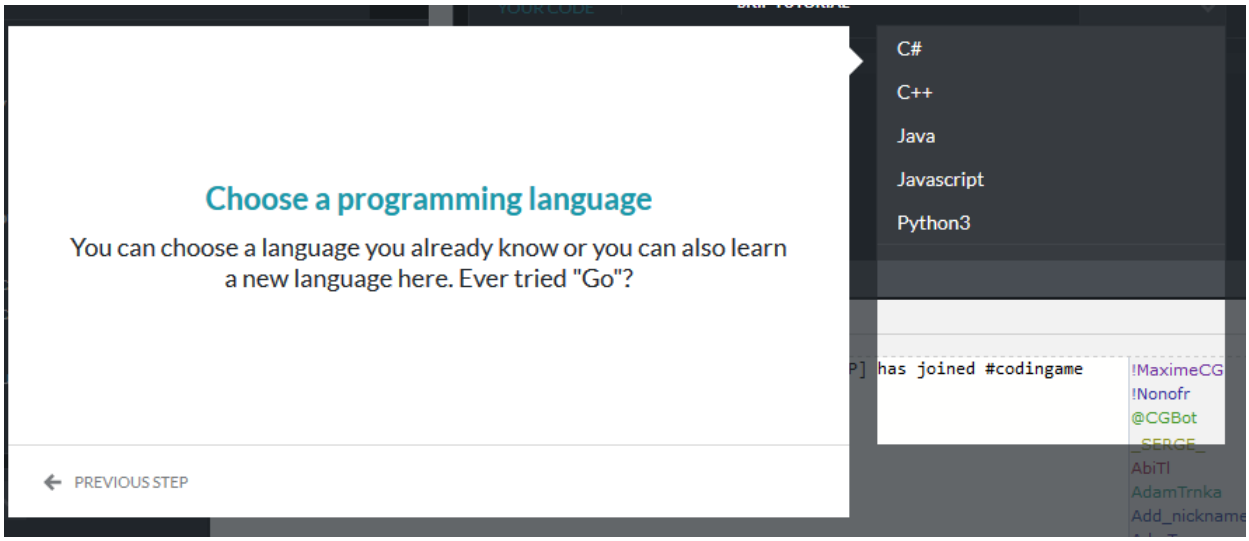
Correct code



Victory / Survey

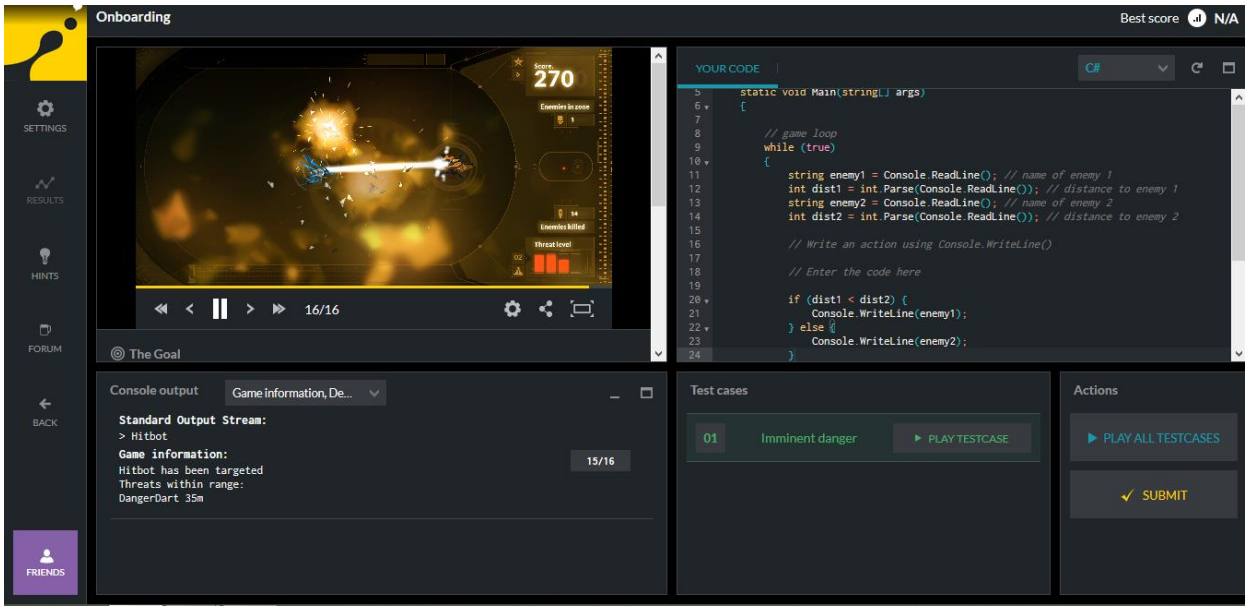
2 CodinGame

Instructions 1

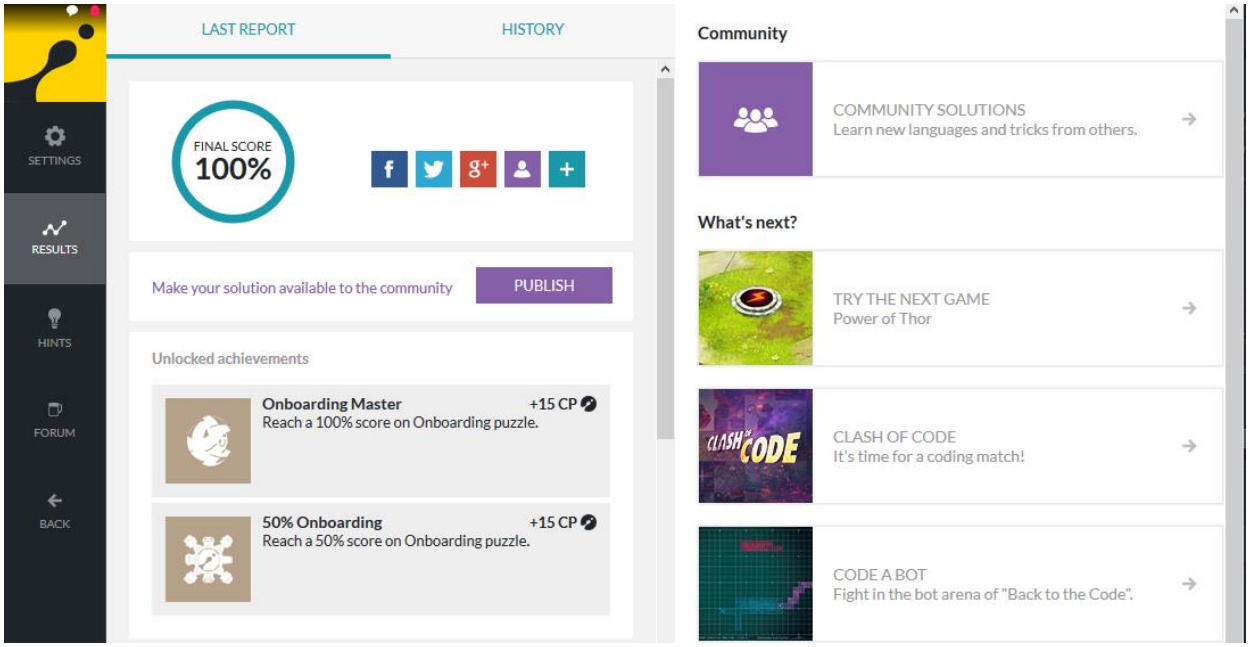


Instructions 2

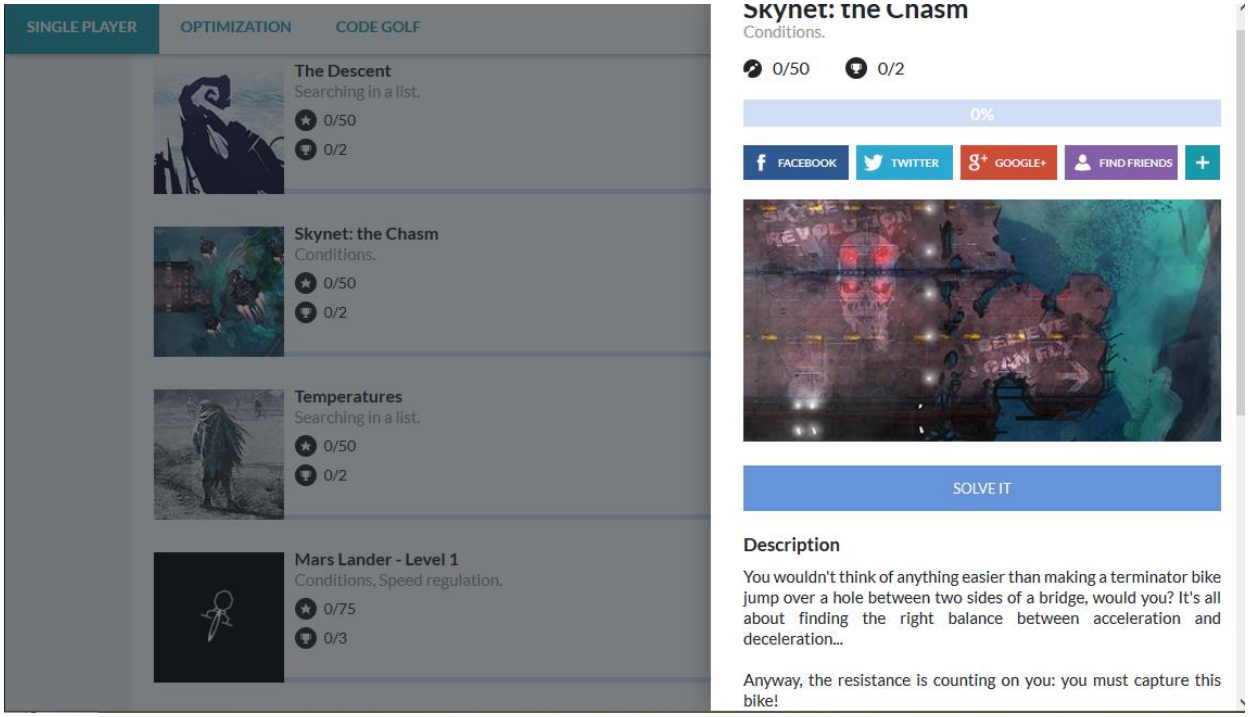
Gameplay / Code editor



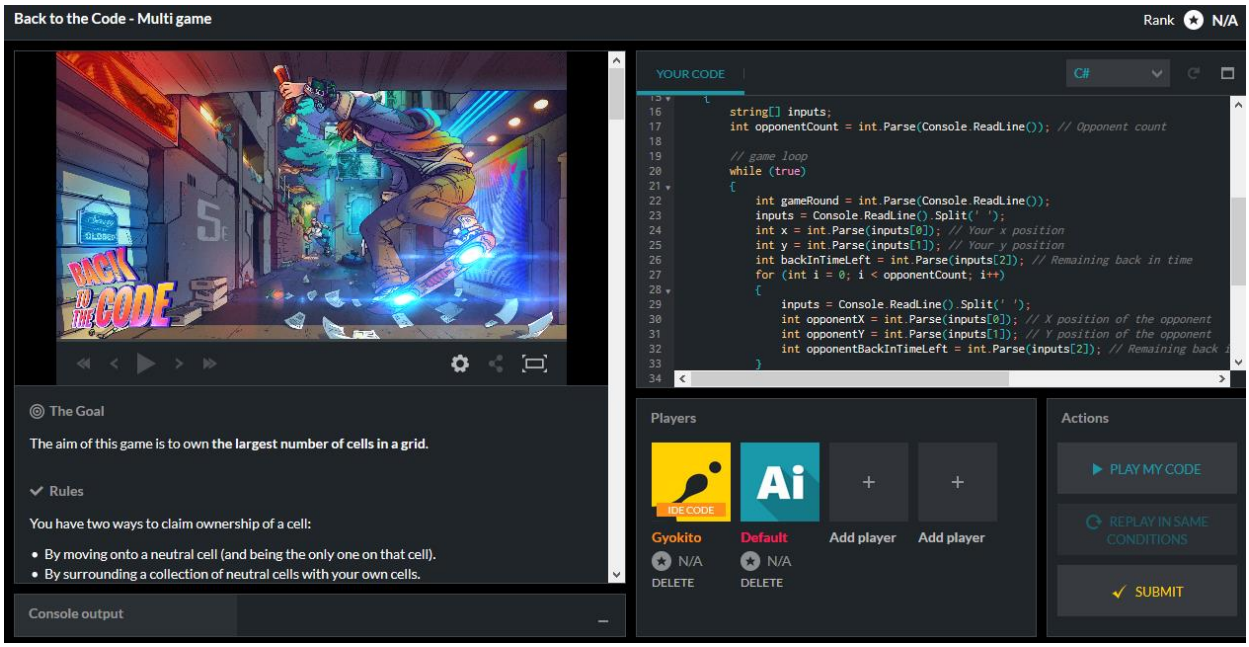
Results



Puzzles and descriptions

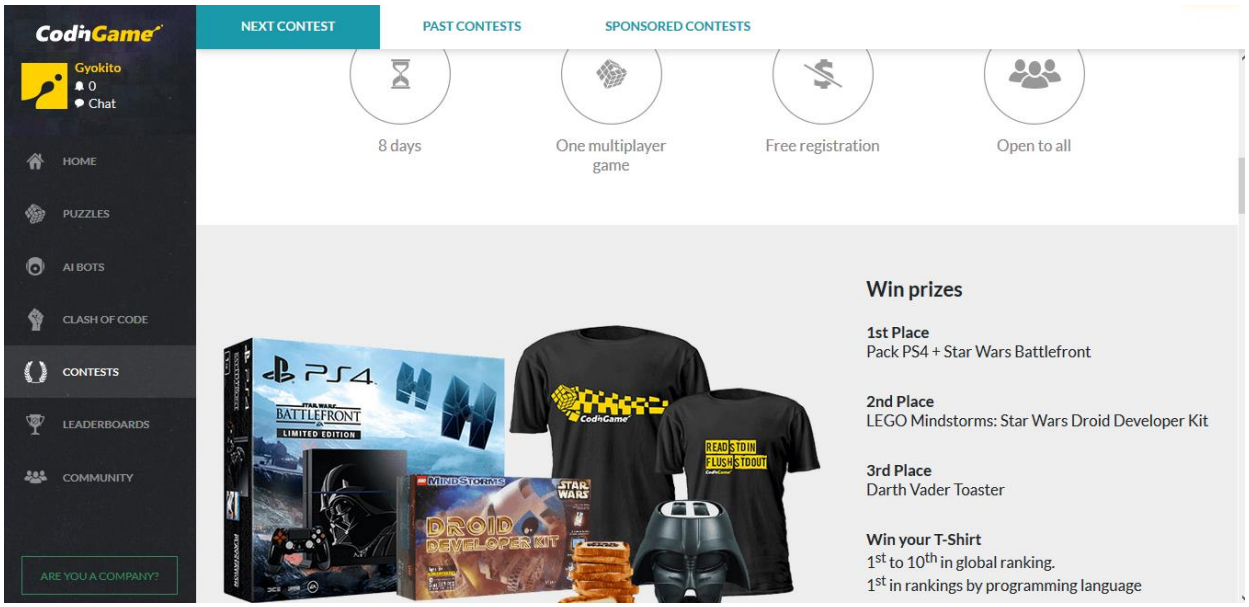


Programming AI Bot



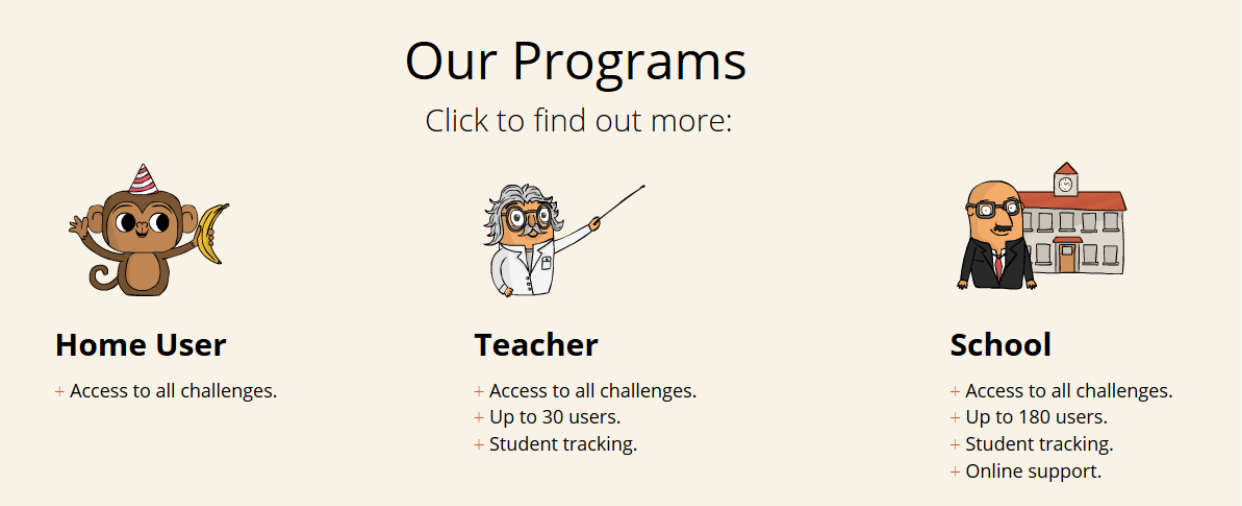
Challenges for advanced players / programmers

Contests



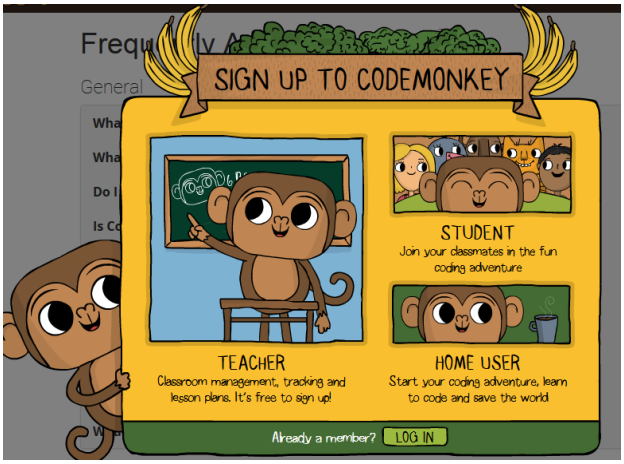
3 CodeMonkey

Information

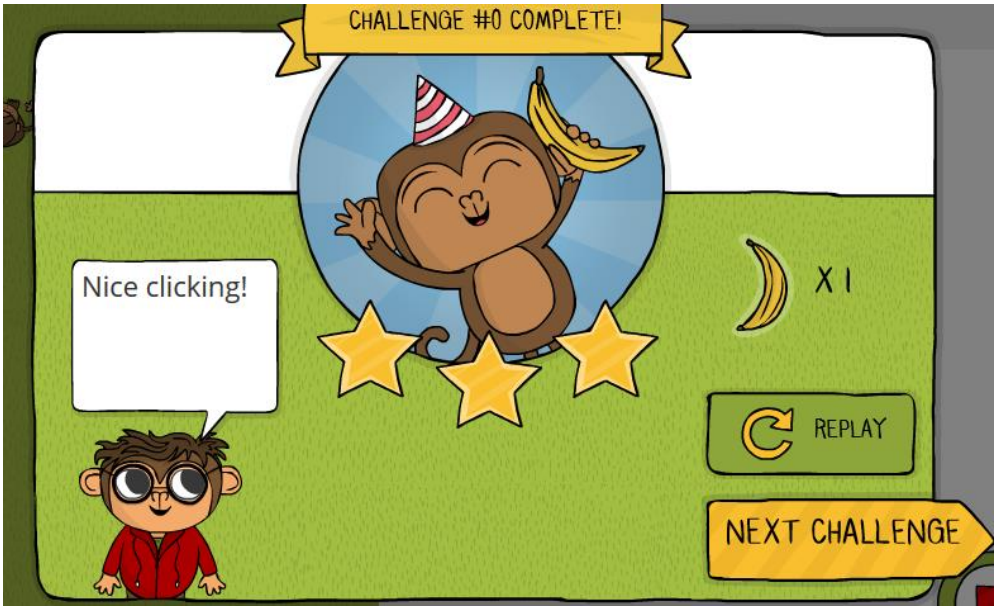


Registration / Login

More info before sign up



Gameplay and coding



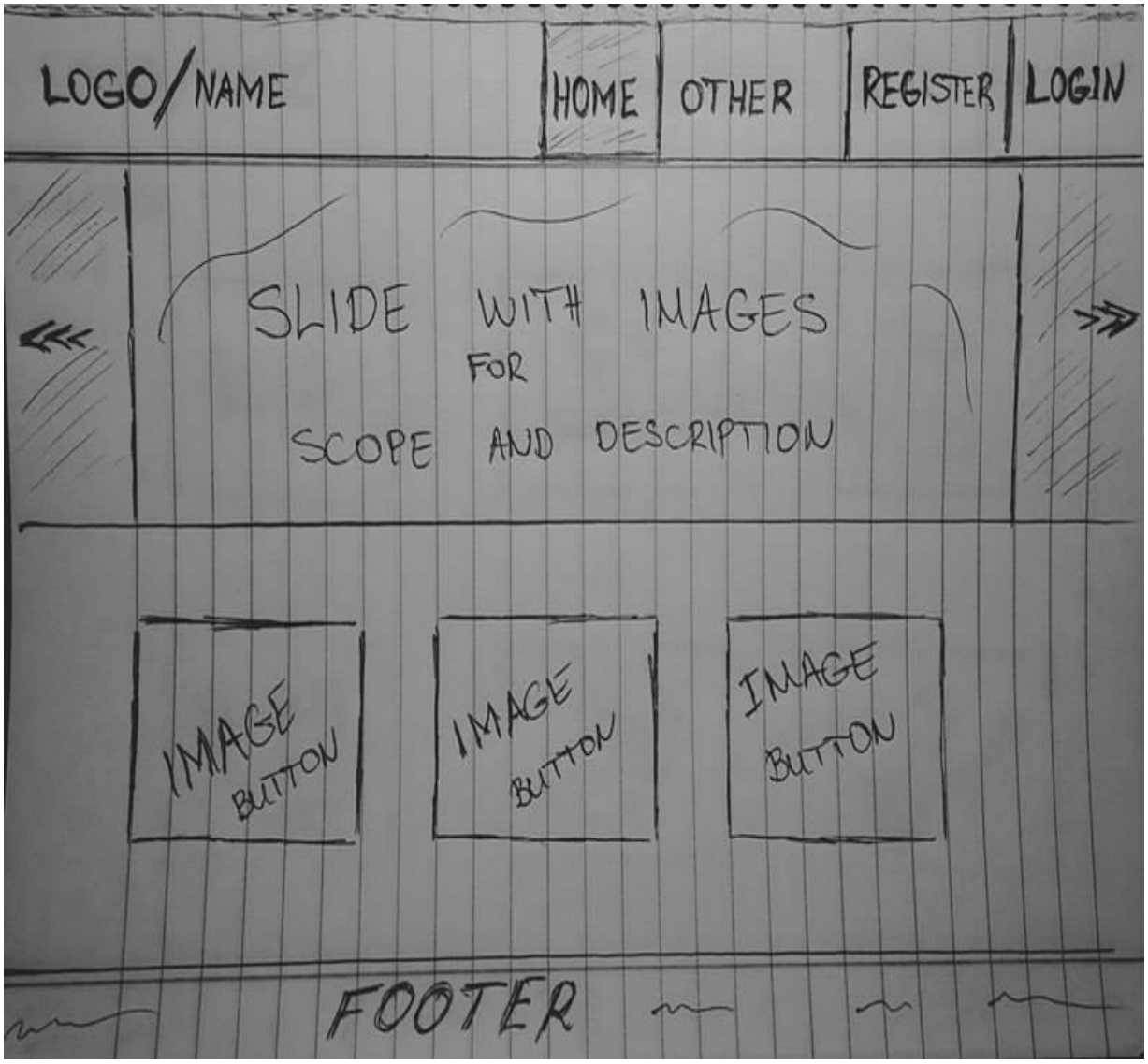

CODEMONKEY
CHALLENGE #60
for b in bananas
turnTo b
step distanceTo b
RUN!
step
turn
turnTo
distanceTo
say
for


CODEMONKEY
CHALLENGE #89
chase = (t) ->
until near t
turnTo t
step 1
safeCollect = (m) ->
until cat.sleeping()
wait()
chase m
grab()
chase pile
drop()
safeCollect match
STOP
step
turnTo
distanceTo
grab
drop
function
until
near
sleeping
wait
chase
safeCollect

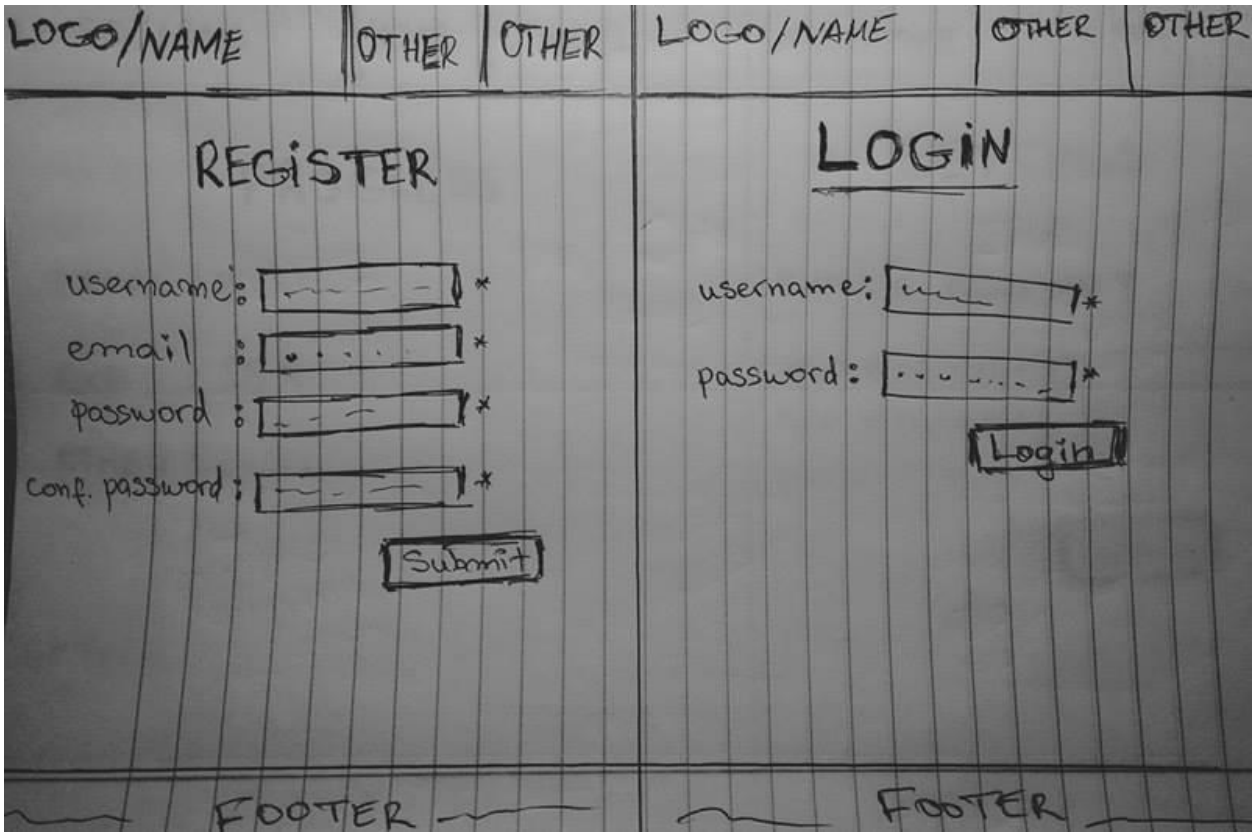

CODEMONKEY
CHALLENGE #107
for b in bananas
goat.goto b
goat.hit()
# if else here:
RUN!
say
goto
hit
frozen
green
if else

APPENDIX D - Low Fidelity Wireframe Sketches

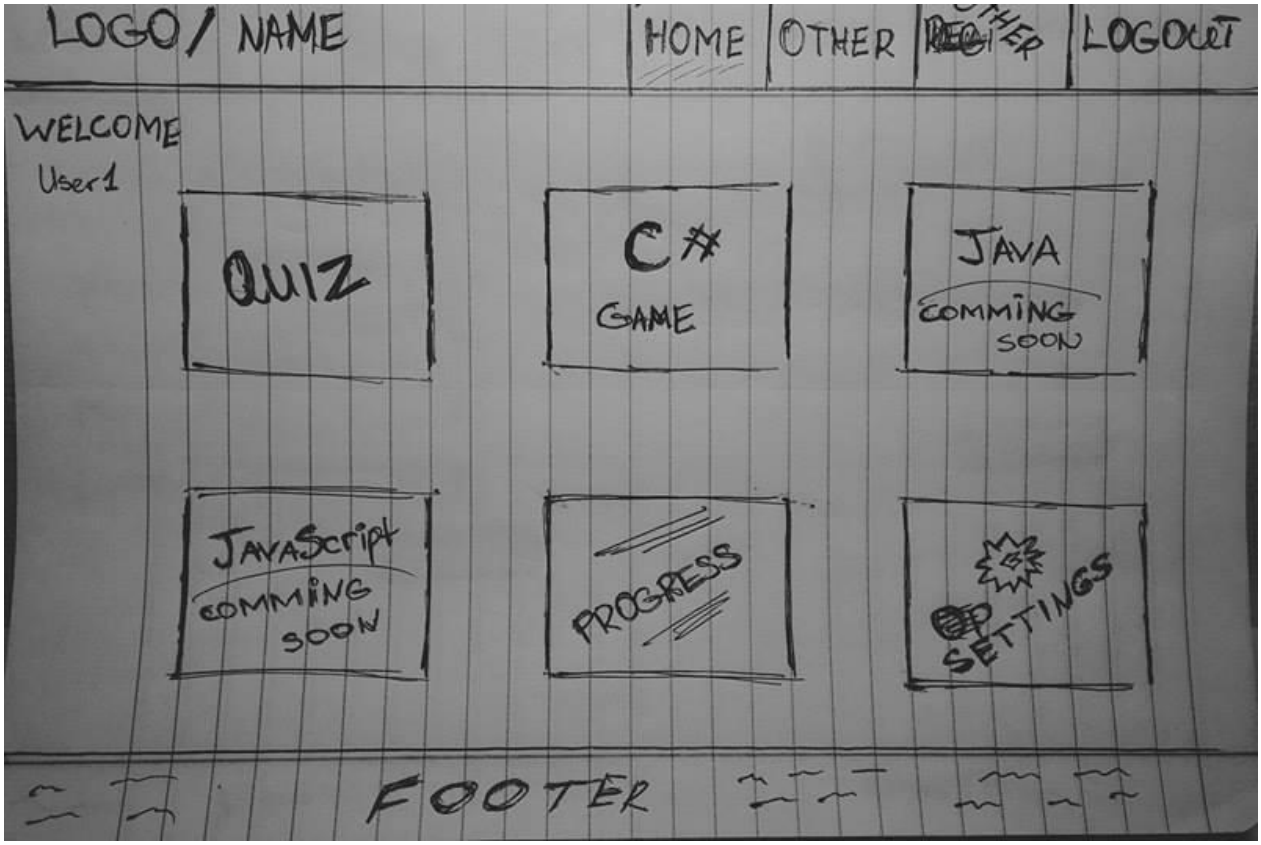
Home / Default page view



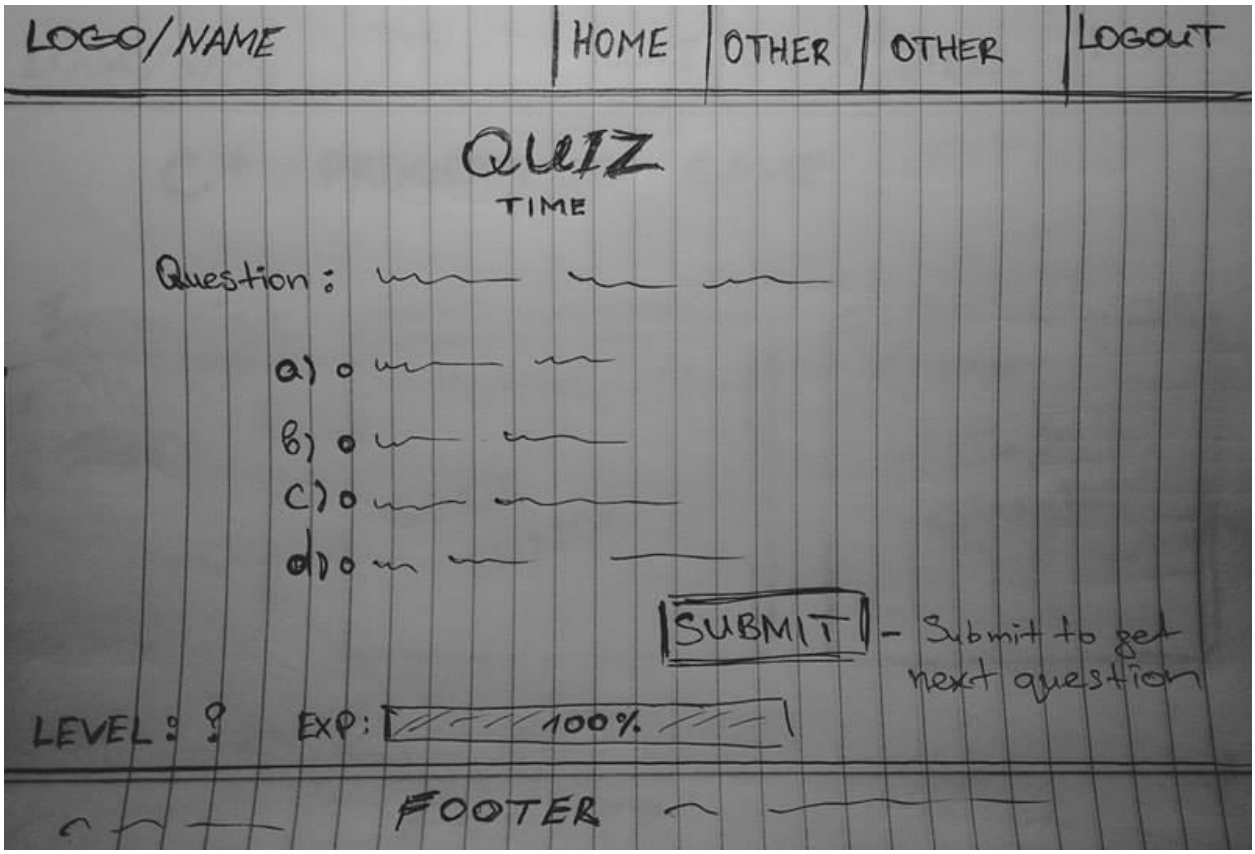
Register / Login page view



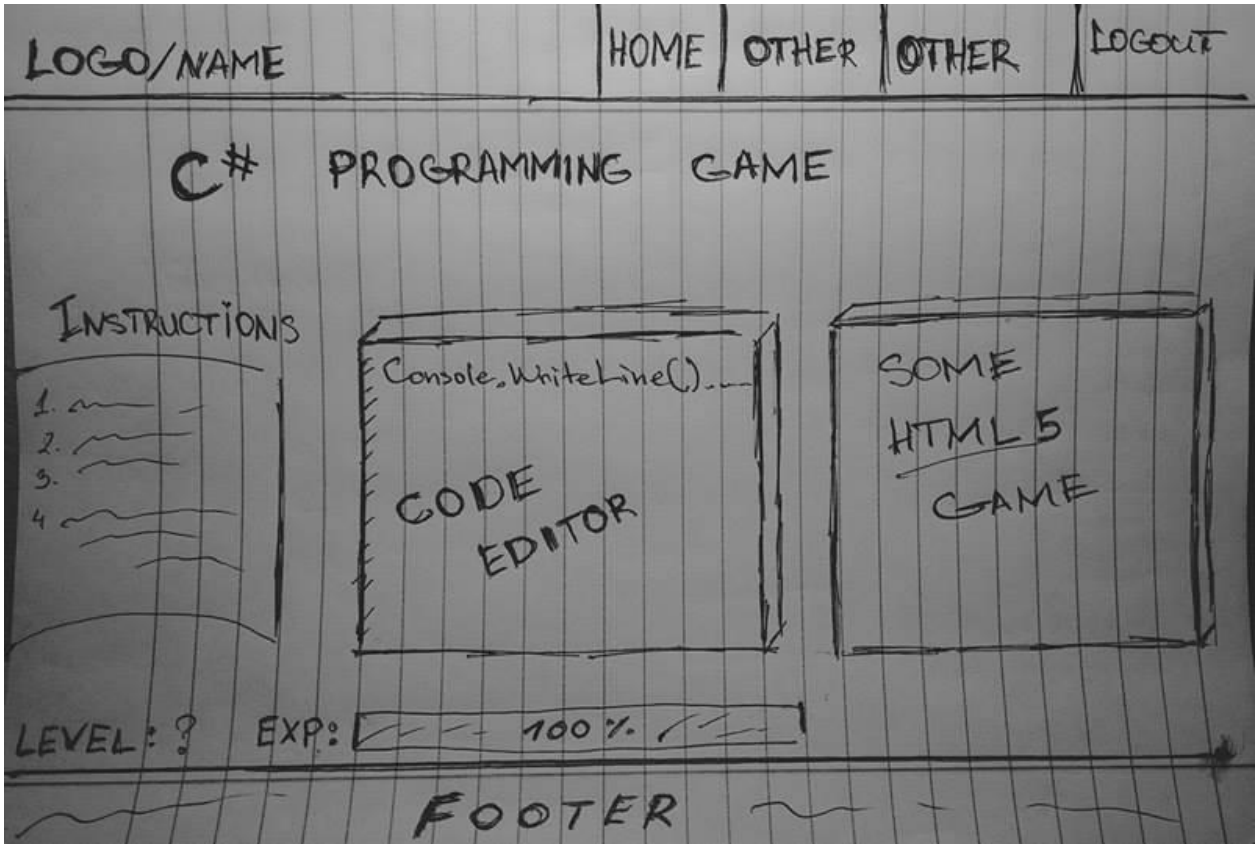
Member's area page view



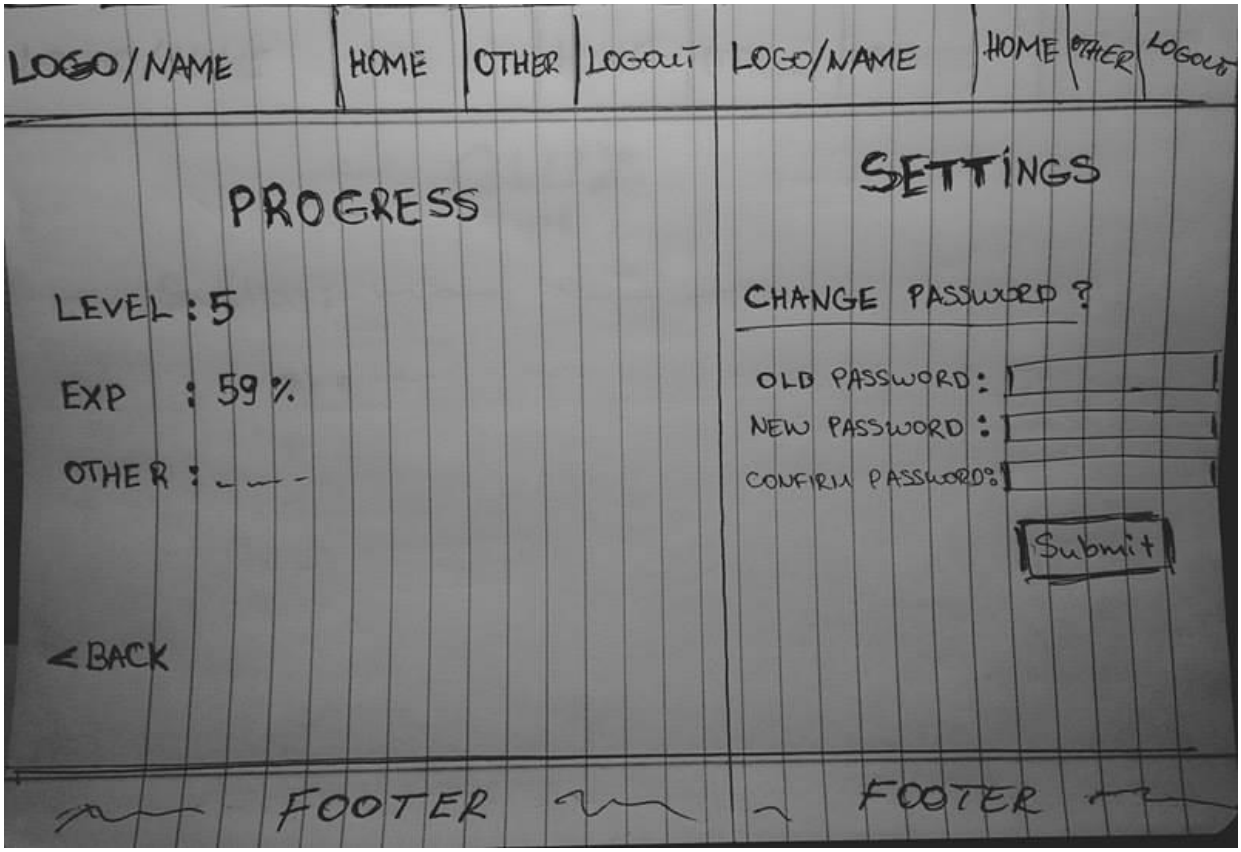
Quizzes page view



Games page view



Progress / Settings page view



APPENDIX E - Prototypes

1 PROTOTYPE No.1 – Default Page