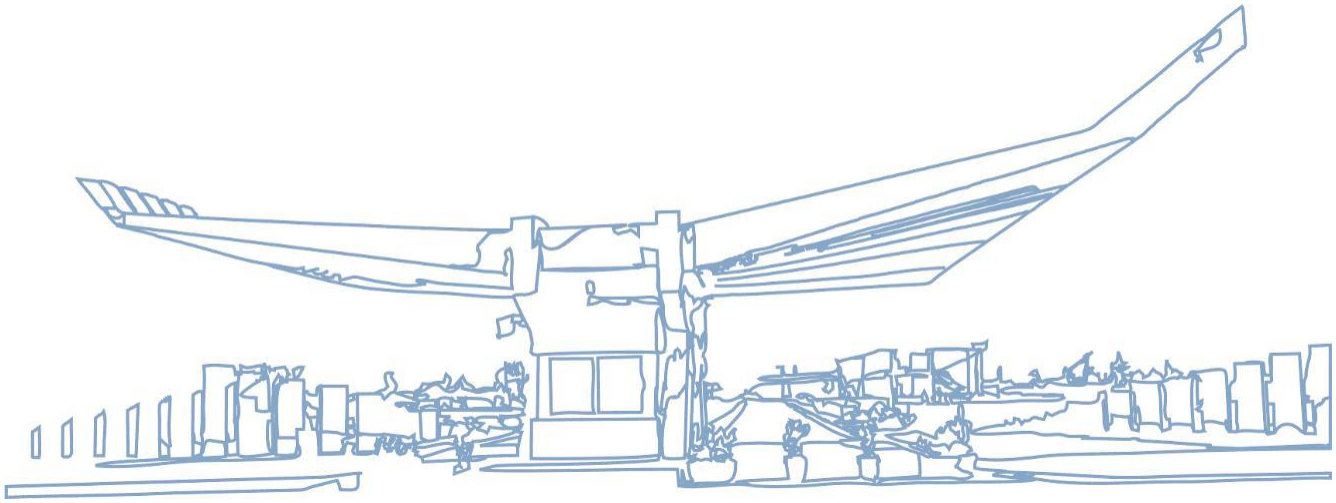
INTRODUCTION TO SOFTWARE ENGINEERING



Team Members:

Greidi Gjoka

Indrit Breiti

Kejsi Bushi

Klea Xhina

Rubin Aga

Table of Contents

1. Executive Summary

1.1 Project Overview

The topic of our project is creating a game that will make high school students like programming.

Almost every high school in Albania offers what is called "IT " classes where students start to understand more about technology and specifically programming.
Not everyone finds coding intriguing, so the purpose of this project is to make students change their opinion regarding programming. We were required to create a simple game that shows students that programming can be fun too.

To do this, our group created a game divided into two parts using python, which consisted of a fox that will grab coins, and at a specific moment, the game will pause to continue, only if the user answers a random question correctly. These questions are related to programming but asked in other ways so the user would not find this game monotonic, but at the same time, the student will learn more about coding.

This product is made for students that do not like programming in the age range 15-18 but can be used by anyone interested in playing interactive games.
Since the age of the users that are going to use this game is specified, we were focused to make this product likable for this range. Firstly, to have a successful game is essential to know what students don't like about coding. To gather that information, we conducted a survey where 168 students from both private and public schools gave their opinion regarding their difficulties in this field.

After we got all the information that we needed, we started to work on our project. Every team member gave their opinion regarding the results of the survey and how we can use that type of information on our project. Then, everyone got their task and started working. The choosing process was based on what every team member felt most secure to work with. As a group, we decided that after a certain task was done by a team member, he would upload it on our working space so everyone could see it and talk through a task that was unclear and needed to be improved.

At the end of the first meeting, we decided that:

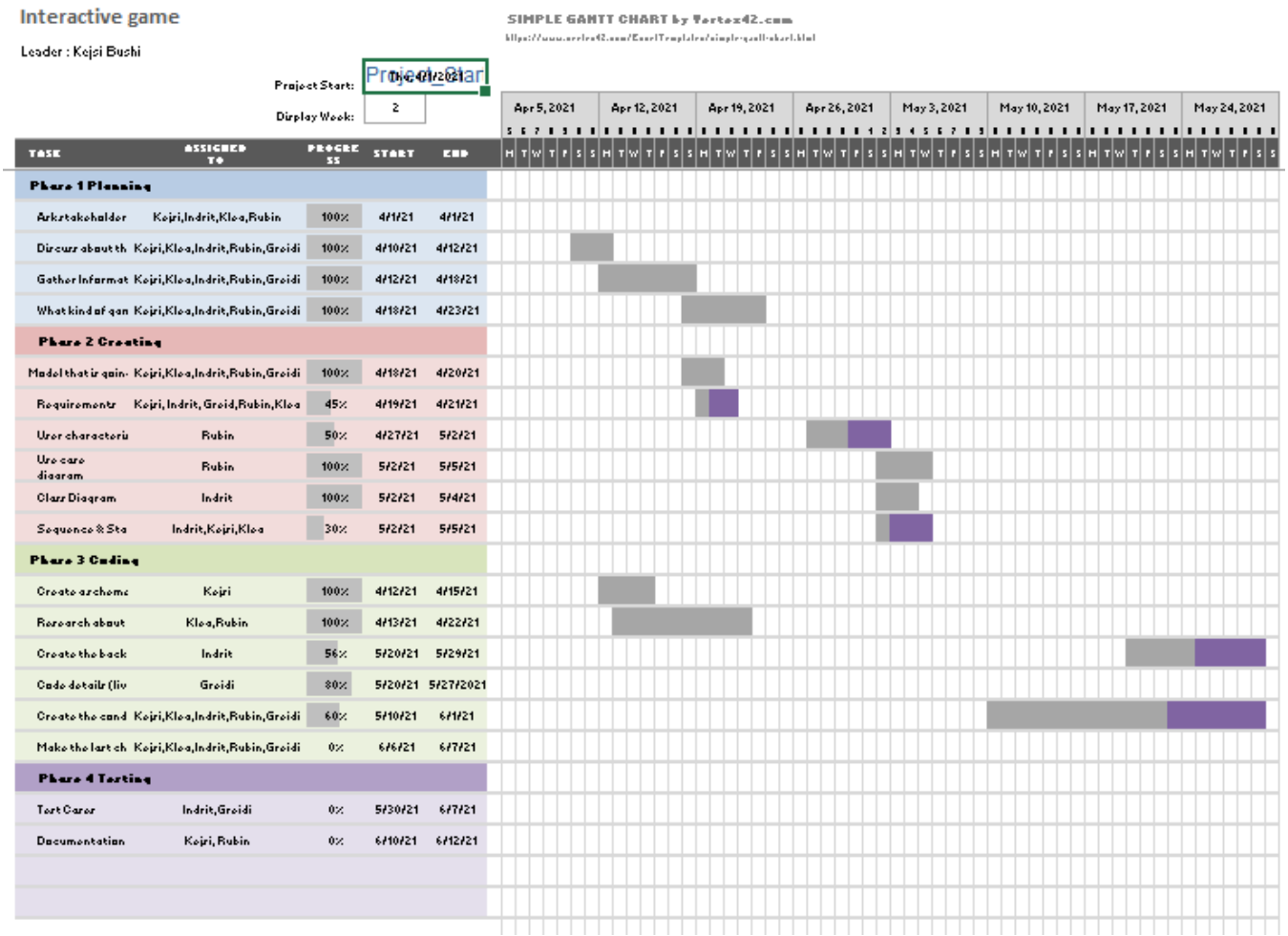
- For every phase of the project, every team member would do a task so everyone would know the progress and understand every step of the project.
- Kejsi and Klea would be more focused on the organization of the project, the designing part, and also analyzing the results of the survey.
- Indrit, Rubin, and Greidi would be more focused on preparing the diagrams like Class diagrams, sequence and state diagrams, and the coding part.

Greidi also prepared some information needed regarding the game.

For our kind of project and the requirement needed, we followed the Agile process model. We were also required to prepare our project using the framework Scrum. We agreed that we were going to conduct a meeting, either on an online platform or meet in person every two weeks, where we would discuss our progress and unclarified topics.

Since we followed the Agile process model, we had the opportunity to be more flexible on the changes that needed to be done during our work. These changes were made after we had discussed mainly tasks that included the design part of the game.
It is essential to have a structured plan to be followed so, no misunderstanding occurs. So, we used the Gantt chart, PERT diagram, and Network diagrams to organize every task. Gantt chart was beneficial since there you could see the date of the beginning and end of a task, the person that a task was assigned to, and the progress of the task making the organization easier.
Meanwhile, while working with the PERT diagram, we could predict for every activity the time that it needed to be completed. Below you will see both the Gantt chart and Pert table regarding our project.

Gantt chart was prepared by Kejsi Bushi



Interactive game

Leader : Kejsi Bushi

SIMPLE GANTT CHART by Vertex42.com

Project Start: Project_Start

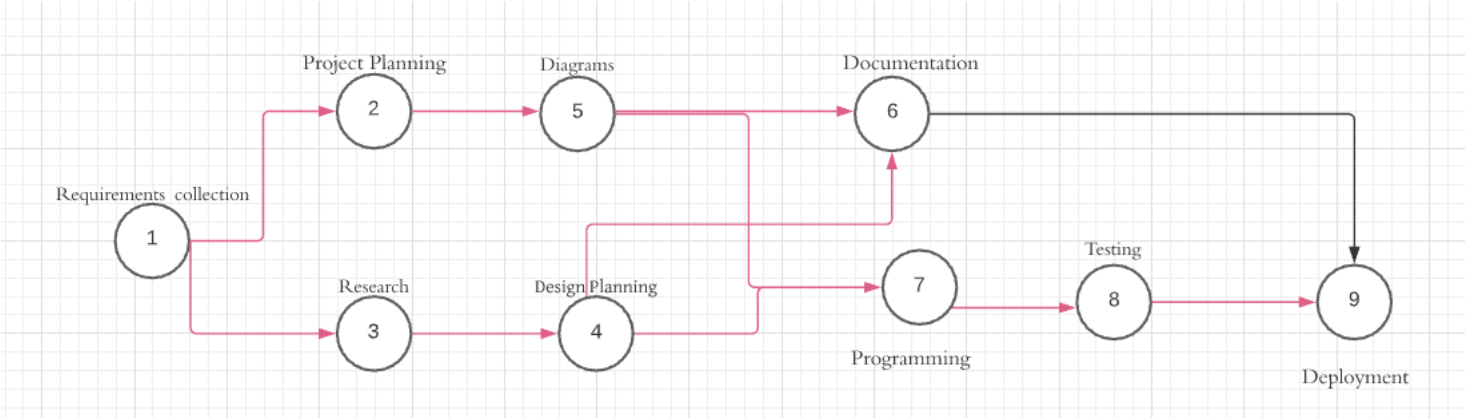
Display Week: 2

TASK	ASSIGNED TO	PROGRESS	START	END
Phase 1 Planning				
Arkz takeholder	Kejri,Indrit,Klea,Rubin	100%	4/1/21	4/1/21
Discuss about th	Kejri,Klea,Indrit,Rubin,Groidi	100%	4/10/21	4/12/21
Gather Informat	Kejri,Klea,Indrit,Rubin,Groidi	100%	4/12/21	4/18/21
What kind of gan	Kejri,Klea,Indrit,Rubin,Groidi	100%	4/18/21	4/23/21
Phase 2 Creating				
Model that ir qain	Kejri,Klea,Indrit,Rubin,Groidi	100%	4/18/21	4/20/21
Requirements	Kejri, Indrit, Groid,Rubin,Klea	45%	4/19/21	4/21/21
Uror characteriu	Rubin	50%	4/27/21	5/2/21
Uro care diagram	Rubin	100%	5/2/21	5/5/21
Class Diagram	Indrit	100%	5/2/21	5/4/21
Sequence & Sta	Indrit,Kejri,Klea	30%	5/2/21	5/5/21
Phase 3 Coding				
Create archeme	Kejri	100%	4/12/21	4/15/21
Research about	Klea,Rubin	100%	4/13/21	4/22/21
Create the back	Indrit	56%	5/20/21	5/29/21
Code detailr (liv	Groidi	80%	5/20/21	5/27/2021
Create the cand	Kejri,Klea,Indrit,Rubin,Groidi	60%	5/10/21	6/1/21
Make the last ch	Kejri,Klea,Indrit,Rubin,Groidi	0%	6/6/21	6/7/21
Phase 4 Testing				
Test Caser	Indrit,Groidi	0%	5/30/21	6/7/21
Documentation	Kejri, Rubin	0%	6/10/21	6/12/21

Apr 5, 2021 | Apr 12, 2021 | Apr 19, 2021 | Apr 26, 2021 | May 3, 2021 | May 10, 2021 | May 17, 2021 | May 24, 2021

Pert Diagram and Network Diagram were prepared by Rubin Aga

	TIME ESTIMATION (in weeks)			EXPECTED TIME
ACTIVITY	o	r	p	E= (o+4r+p)/4
Requirements collection	0.5	1	1.5	1
Project Planning	1.5	2	3	2.08
Research	0.5	1	2	1.08
Diagrams	0.3	1	1.5	1
Design Planning	1	2.2	3	2.13
Programming	3	4	5	4
Testing	0.5	1	1.5	1
Documentation	0.2	0.6	1.2	0.7
Deployment	0.2	0.4	0.6	0.4



1.2 Purpose and Scope of this Specification

The scope of this project is to help students to understand more about programming and how it works. Our idea was to create a game whose main purpose, besides helping them regarding the part of coding, would be increasing their interest in the technology field.

Our customer gave some ideas regarding the product. He gave us some requirements that we should include in our game. The customer explained that since this game will be mainly used by high school students, we should focus on their likes and dislikes to achieve the best product possible. We decided to create a questionnaire where high school students would be asked regarding the difficulties that they face during learning programming but also suggestions that would help them make this topic easier to understand.

To make this game even more interactive, we have chosen facts related to coding. We were careful to choose the information that will make the users understand that:

- Programming can be learned by anyone
- It has great importance and is related to every profession. To emphasize this idea, we found some information that will be displayed on the game.

Since our customer was focused on wanting this game to be as interactive as possible, we decided to divide this game into two parts.
In the first part, the user will try to collect coins, but at the same time, he will be faced with some simple multiple-choice questions where he will use the number of the alternative he thinks is correct to give his answer. The user will continue to play only if he answers the question correctly. To collect these coins, he will be only using "A" and "D" on the keyboard to move forward or backward. Also, during the game, the user will see some fun facts that will give him an extra "life" for his game. We wanted that this first part would give the idea of a fox running in a modern city.
Meanwhile, in the second part, the fox will have to move around, so we added two extra keyboard buttons. We also have inputted the concept of time. Here the user collects the coins and answers some questions till the time is finished.
The purpose of this is that the user while playing the game, will understand more about programming but also learn interesting facts.

2. Product/Service Description

The product/service we offer is an interesting game for students in high school who are interested or want to learn about programming.

2.1 Product Context

The game is independent and not related to any products. It is made to be accessible for all students in high school years who are willing to learn to program. This product can also be used by teachers as a tool, or method for teaching.

2.2 User Characteristics

The different profiles that will use this product are students and professors/teachers in high schools.
Students:

- The students are the main players.
- The levels will provide questions for them to answer, each question will become more difficult.
- Once they are done with the levels, they are given free time to collect coins and move freely.
- The game offers a leaderboard, which encourages students to compete and learn more about programming.

2.3 Assumptions

The game is assumed to be run on the computers in the school's PC labs, operating with Windows OS. An IDE to run the turtle is also assumed to be present.

2.4 Constraints

-The game is designed and tested on Windows devices.
-The game is functional on Windows and Linux.
-Older versions of Windows present no problem when the game is running.
-Online IDE such as repl.it causes lag and a lack of sound.
-It is recommended to download the game and run it with an IDE that runs turtle on python (pycharm is the number one suggestion) which will help in avoiding lag and -lack of sound when running the game.
-The free space required is 1,77 MB of free space.

2.5 Dependencies

The game must be updated by downloading the file through the team's GitHub for future updates and new questions. There, the user is also instructed on how to add questions by themselves (a feature for professors and teachers).

3 Requirements:

In all software programs, before coding the program, prerequisite requirements are needed so the work can flow properly, knowing what is needed from the program. These requirements are brief descriptions of processes or functions of the final product. Depending on the type, functional or non-functional they describe how the specific requirement works or what needs to be present for it to work. To create all the requirements and all the necessary precautions before coding the program, we conducted several meetings. These meetings varied from ones within the team to ones with the customer. The meetings with the customer were necessary to know the concrete expectations for the program, its structure and its goal. Within the team, the meetings consisted of discussions regarding the best way to achieve what the customer wanted. In these meetings, the theme of the software, the necessary data for the program and other details regarding the product were discussed. What we mainly discussed was split into two main parts, the functional requirements and the non-functional requirements.

Functional requirements:

- The system must show the rules of the game every time it is started.
- The system must allow users to enter their unique username.
- The system must help users understand how to play the game.
- The system must have two working levels and transition properly between them
- The system must reward players for achieving high scores
- The system must allow users to record their own scores within the high-score system embedded.
- The system must be granted internet access to open website pages related to the topics in the software.

These functional requirements are vital for the software to function. In order for our final product to achieve the maximum potential, all the bullet points must be ticked off.

Next, there are the non-functional requirements. non-functional requirement is a specification that describes the system's operation capabilities and constraints that enhance its functionality. These may be speed, security, reliability, etc. We split these into multiple points. Some of them being:

User Interface Requirements

- User interface needs to be intuitive.
- User Interface needs to be easy to the eye and welcoming for the user.

- User Interface needs to be lightweight and load quickly so the user does not have to wait.

Since the user interface is the first thing the user sees from the product it needs to convey what the program does. It needs to be pleasing to the eye and fun to use. To solve these issues we conducted many meetings. In these meetings we talked about the easiest way for the users to know how the program works, how to use it and to start the game immediately. The colors and background used were picked in consensus by the whole team. So were a good part of the visual indicators. The user interface does not only consist of the main page of the program but also how the user will interact with the program. The game is played using the arrow keys. Tips explaining how the game works are given throughout the game so the user always knows what is going on. The game functions in a 2D manner, with 2 levels. For every change the user is informed via a popup text message on the screen.

Usability and Compatibility

Since the program is in its early stages of development, currently it only supports Windows 10 and higher operating system. Only one person can play it at a time in a machine. However, scoring the scores of multiple players is supported. This means that while only 1 person can play at a time, the score of each person is recorded so if multiple people want to compete with each other, that is made possible by the leader board feature. The program does not conflict with any other processes in the operating system.

- The program is easily picked up by all users.

- Reaching the end of the program depends on the amount of questions and length of the session. Usually it does not take too long to reach the end of the program.

- Users have an easy time relaunching the program and starting the work immediately.

- The design is pleasant to use and mainly intuitive.

Availability

Since the program requires an internet connection for its full features, the availability of all the processes in the program depends on the internet connection, however many of these features are optional and do not affect the program itself. Only some processes, such as watching a video to regain a life or looking up topics is where the internet is needed. So, depending on the user's internet access, the program is downloaded, small and lightweight so there are no problems running it offline in the machine.

4. Design thinking methodologies

4.1 Negotiation

Negotiation is one of the most important parts of software projects. Its definition only means a discussion made between the team members that are holding the project and helping to reach an agreement. During the process of creating our project, we held various meetings discussing the problems we encountered and trying to find the most effective solutions. In addition to google meetings we also met each other in cafes, we also have used other social networks and applications such as WhatsApp, Trello, etc.

To improve the topic on which we would base the students' consent in the subject of IT, we created a form that we asked a specific number of students regarding their opinions on why they don't like programming. After that, we discussed with each other the ideas that each of us had about the game we were going to create. Everyone had different ideas. Indrit suggested creating a basketball game, Casey suggested creating a puzzle while Rubin and I preferred more a thematic cooking game.

Our idea was a bit difficult to create. So Greidi suggested the old Snake game. His idea was liked by all of us and so we continued on how the game would work. Instead of a snake, Indrit suggested creating a fox and the theme would be the same as the old game where the fox would catch coins. But in our game every time that the fox catches a coin a question pops up. After this, we had the main idea of what we would do and were very satisfied with each other.

When the game was created there needed some minor adjustments. At first the character moved faster than he should. Another small problem was the final part of the game where the score was not so distinct and we couldn't see it clearly. The critiques were made so we continued with the work on fixing them

4.2 Empathy

To have empathy means understanding, being aware of, being sensitive to, and vicariously experiencing the feelings, thoughts, and experiences of another person. In our project to be empathetic means to put ourselves in the shoes of high school students. What their opinion would be on programming and IT. How they feel about it what they like and what they dislike.

Firstly, we put ourselves in a high school student position. Students generally complain about the textbooks they have in the IT subject. In addition, they say that even teachers do not give them the right importance. Except for the user in our case the students, we tried to be empathetic with the team members too. For every problem we had, we asked each other and tried to help each other in every way. We tried giving ideas to improve the project. One case was the design of the game. What colors and background should the game have. Kejsi searched and learned what was the most attractive and pleasing colors. So that's why we have used light colors like blue yellow orange etc. Also, the group leader tried to share the tasks in the best way for each one of us.

4.3 Noticing

While working with the project, we even learn new things that we didn't know before. Firstly, we noticed the reality of programming consent here in Albanian high schools which was a bit disappointing So, to change this we tried to make programming enjoyable for everyone. We created this game so we can play and learn at the same time. You don't need books but only your logic. While searching for information we also learned some interesting facts like the first programmer was a lady, Ada Lovelace, Lord Byron's daughter. Also, the world's youngest programmer is a 6-year-old named Kautilyaa Katariya.

4.4 Critiques

When we had finished half the work, we left a meeting with professor Enea where we would show our progress so far and what opinion and criticism he would have. After we presented all the work he gave us some opinions on what to add.

Firstly, he wanted us to play more with the colors. Then he said that during the game it would be better if we gave some hints or help because it would make it even more enjoyable. Besides that, he said that the game should be short and practical. No one wants an endless game with so many tasks and subordinations. He asked for some curiosities too.

This would make it even more interesting and playable. And after every answer that the user gives we would give feedback like why was it true or false depending on the answer. He also said that it would be good to use the pert method which is one of the most common scheduling techniques for network analysis and is used to track and coordinate complex tasks.

It also helps to analyze the project work schedule by focusing on each task and calculate the minimum time required to complete the project. By the answer that we got from the form, another problem was the difficult language used so we should give an understanding and easy language to read. Also, if the user loses the games we could add some internet videos where we explain the mistake that he/she made. After we got all the feedback from the professor we continued with the changes that needed to be made.

5. Software Design

5.1 Use Case

The main use case of this software is making programming easier to learn for teens.

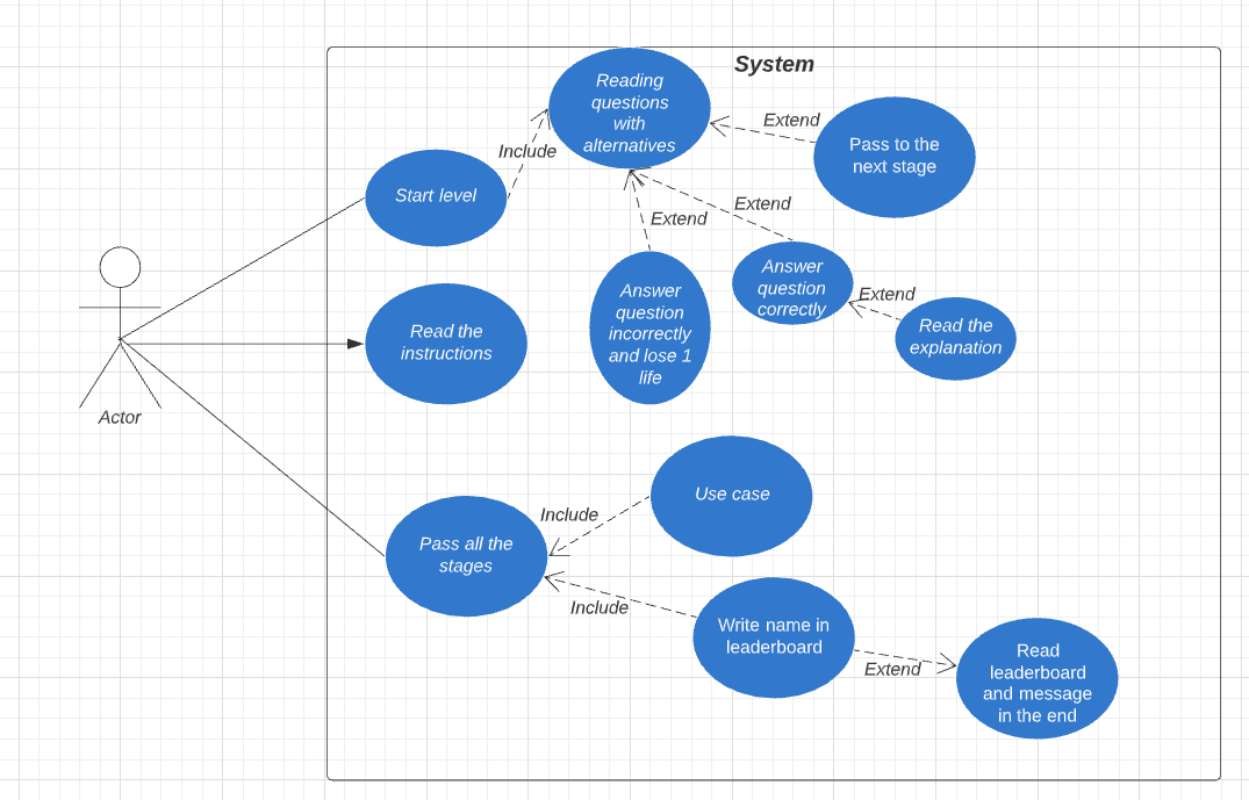
● The objective is to learn coding to teens through a game. The user will experience animations related to coding facts and questions. He will learn new things about programming while gaming.

● There is only one actor that interacts at a time, the player. However, the leaderboard shows the data for multiple users.

The actor will interact with the system using only the keyboard. First, he has to press any key to exit the rules screen. Then the user will be prompted for the username through a text dialog box.

In the first game mode the user can move back and forward using the keys 'a' and 'd'. When a question is stated, he can choose the desired option using keys '1', '2', '3' or '4'. On this game mode the user cannot move to the next question without solving the current one. For each wrong answer he loses 1 out of his 5 lives. Randomly, but rarely facts are spawned instead of question, each fact gives you 1 extra life. Only after finishing all the questions or loosing all his lives, he can move to the second game mode. In the second game mode the user can move using the keys 'w', 'a', 's' and 'd'. Choosing an answer is the same as in the first game mode. The user has to collect as many coins as possible each with a value of +1 point. Randomly some coins have a question which disables the player from moving until he answers. If he answers correctly he gets +10 points, otherwise he loses 1 of his 5 lives, the correct answer is displayed and question ends. This mode has a time limit. If the user finishes all the questions

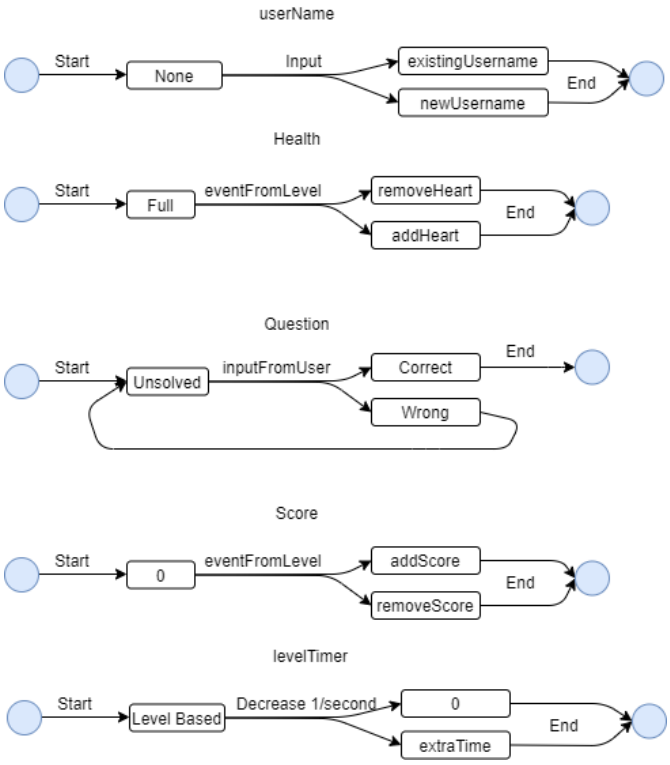
before the time limit, he can freely collect as many coins as he can until the time runs out. The game mode ends if the user runs out of time or out of lives.
After the last game mode, the leaderboard, a replay and an exit button are displayed



5.2 State Diagram

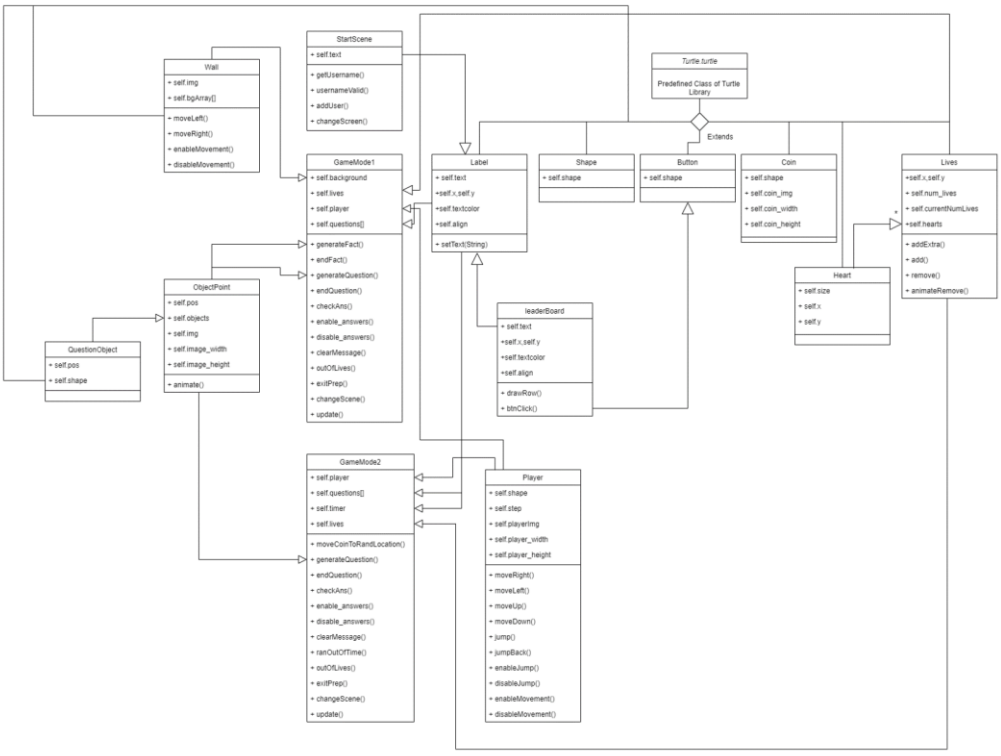
Each object of the game is created by using classes so controlling their state is as simple as calling the functions implemented in them.
Objects which change continuously like the timer are controlled by the update() function which is called each time the turtle screen updates.
All the other states are controlled by keyboard events or related events like player-object distance.
For example:

- The state of the question is controlled by the keypress of the key related to each option.
- The state of Score is controlled by question classes and coin objects.



5.3 Class Diagram

Most of the classes are associated with Turtle.turtle to make it easier to create custom turtles. For example, the label class is just a turtle with hidden turtle, penup, and a function to change the text when needed. Using this class you can create labels just using label=Label(text, position) without having to rewrite penup, hideturtle etc.



5.4 Class Based Modeling (CRC)

The code is developed using objects. This makes the code readable and maintainable. Objects like the coins can just be called as different instances and you can have full access to each instance. Some classes inherit the Turtle.turtle object, this way the class has all the members of the turtle and makes it much easier to create new objects based on it.

Class: Player	
Responsibility	**Collaborator**
Avatar Image	
Position	Keyboard Handler
Scaling	
Speed	
Jump	Keyboard Handler
Collecting Points	Coin
Enable/Disable Movement	

Class: Coin	
Responsibility	**Collaborator**
Coin Image	
Position	Keyboard Handler
Scaling	
Detecting Overlap with Player	Player

Class: Button	
Responsibility	**Collaborator**
Use turtle as Button	
Image Button	
Detect Click	

Class: Label	
Responsibility	**Collaborator**
Use turtle as Label	
Show Text	
Update Text	
Format Text	

Class: Score	
Responsibility	**Collaborator**
Show Current Score	User, Label

Class: Timer	
Responsibility	**Collaborator**
Show Timer	User, Label

Class: Lives	
Responsibility	**Collaborator**
Show Health	Heart, User
Add/Remove Heart	Heart

Class: Heart	
Responsibility	**Collaborator**
Draw Heart	

Class: RulesScene	
Responsibility	**Collaborator**
Display Commands	Label
Display Rules	Label
Detect Keypress	Keyboard Handler

Class: Wall	
Responsibility	**Collaborator**
Repeat Image as Backgroun	
Keep track of Player Position	Player
Enable/Disable Movement	

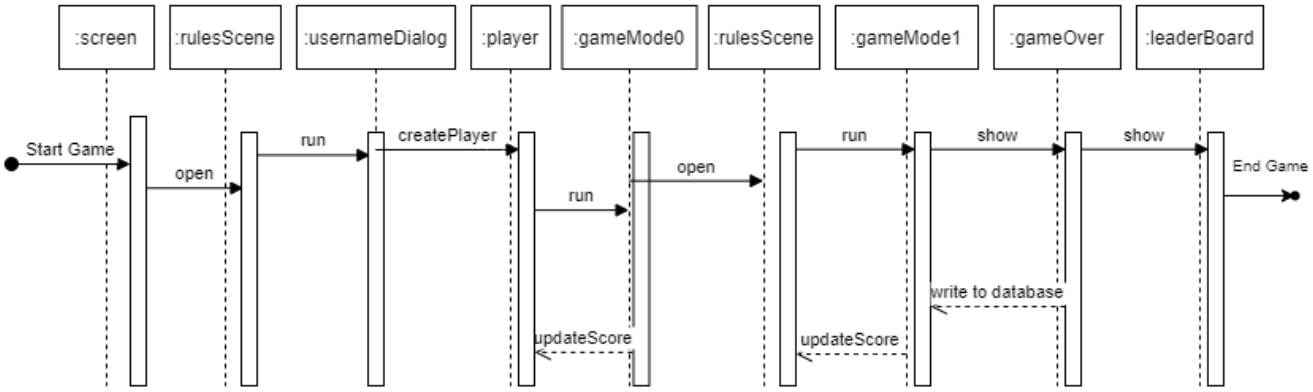
Class: QuestionPoint	
Responsibility	**Collaborator**
Detect Aproach of Player	
Display Question Image	
Start/End Question	GameMode

Class: GameMode	
Responsibility	**Collaborator**
Randomly Select Question	
Display Correct/Wrong Message	
Enable/Disable Answers	
Update Score	Score
Update Lives	Lives
Update Timer	Timer

Class: LeaderBoard	
Responsibility	**Collaborator**
Display Ranked Users	User, Label
Display Score	User, Label
Display Time	User, Label
Replay/Exit Button	Button

Class: User	
Responsibility	**Collaborator**
Get User Name	Welcome Scene
Keep Track of Points	Player
Keep Track of Time	
Save Data to Database	
Temporary Save Health	

5.4 Sequence Diagram



The game is divided in 5 main sections:

1. At first the user is presented with the game rules for the first game mode.
2. After reading the rules he has to enter a unique username to enter the first game mode.
3. After finishing all the questions or losing all his lives of the first game mode the rules of the second game mode are displayed.
4. After reading the rules the player moves to the second game mode. The second game mode finishes if the player runs out of time or out of lives.
5. After the second game mode has finished, the leaderboard is shown.

Each section (screen) is controlled on the main loop by a global variable. After each scene the screen is cleared and the new screen is initialized.

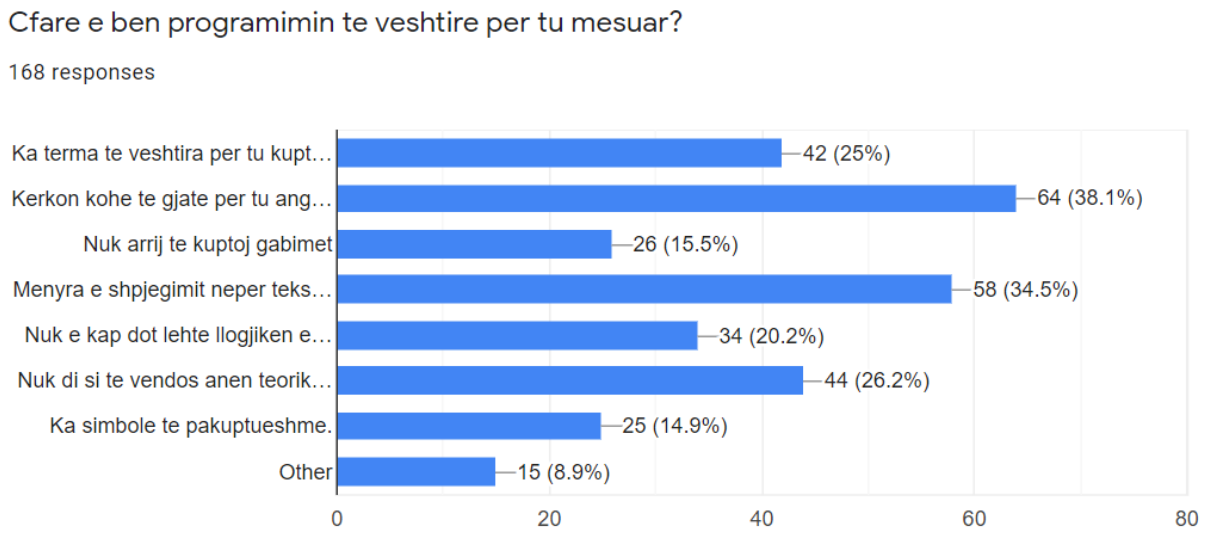
APPENDIX

Group Participation:

During these 14 weeks, we have organized ten meetings in total. From these ten meetings, three of them were with Mr.Enea, and others were a group meeting via Google meet or meeting in person. All the team members have participated equally and tried to give their best to complete all the requirements.

APPENDIX 1

To understand more what high schoolers think about the programming, we conducted a survey. This survey was made in the Albanian language, because, in the end, this game is going to be used by Albanian students. The results of the survey are shown below:
Link:
https://docs.google.com/forms/d/1OvRE907HRIOIKlPh6nlanOXmHYev3Un8NNVg4NIEfAw/edit#responses



Other

12 responses

Duhet praktike dhe shume ushtrime per te kuptuar logjiken dhe per tu aftesuar sa me mire.

Nuk na e mesojne mire

.

Nothing

Tekstet jane teper te prapambetura ne programime....programet azhornohen rregullisht dhe kjo i ben te veshtira per ti kuptuar.

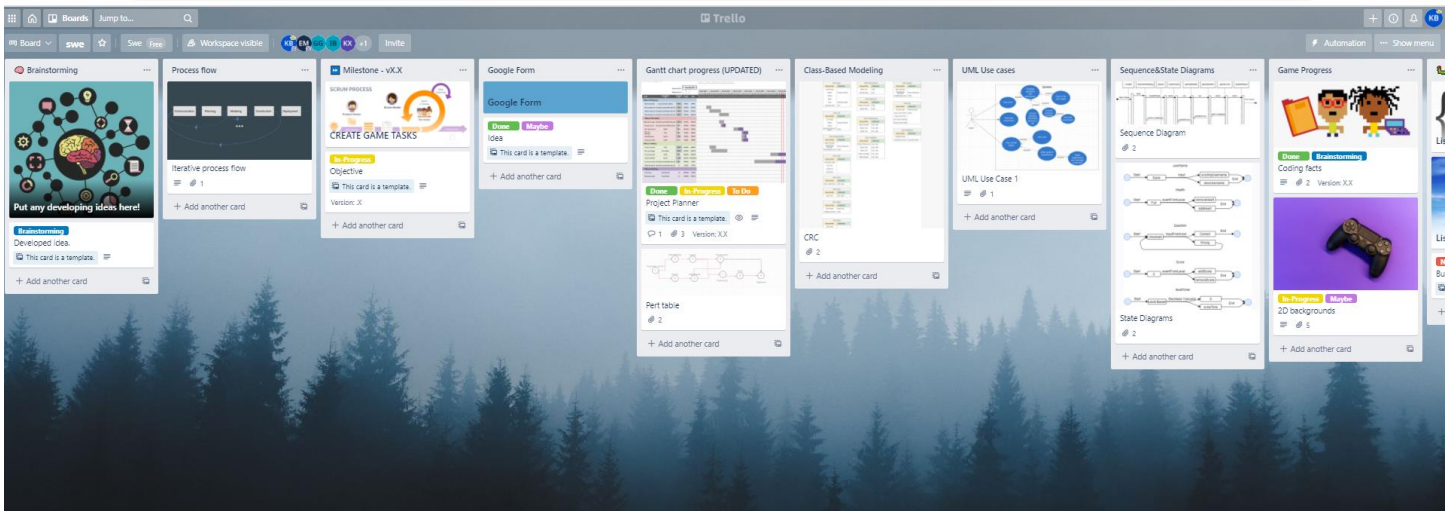
Mësimi në online ë më i vështirë

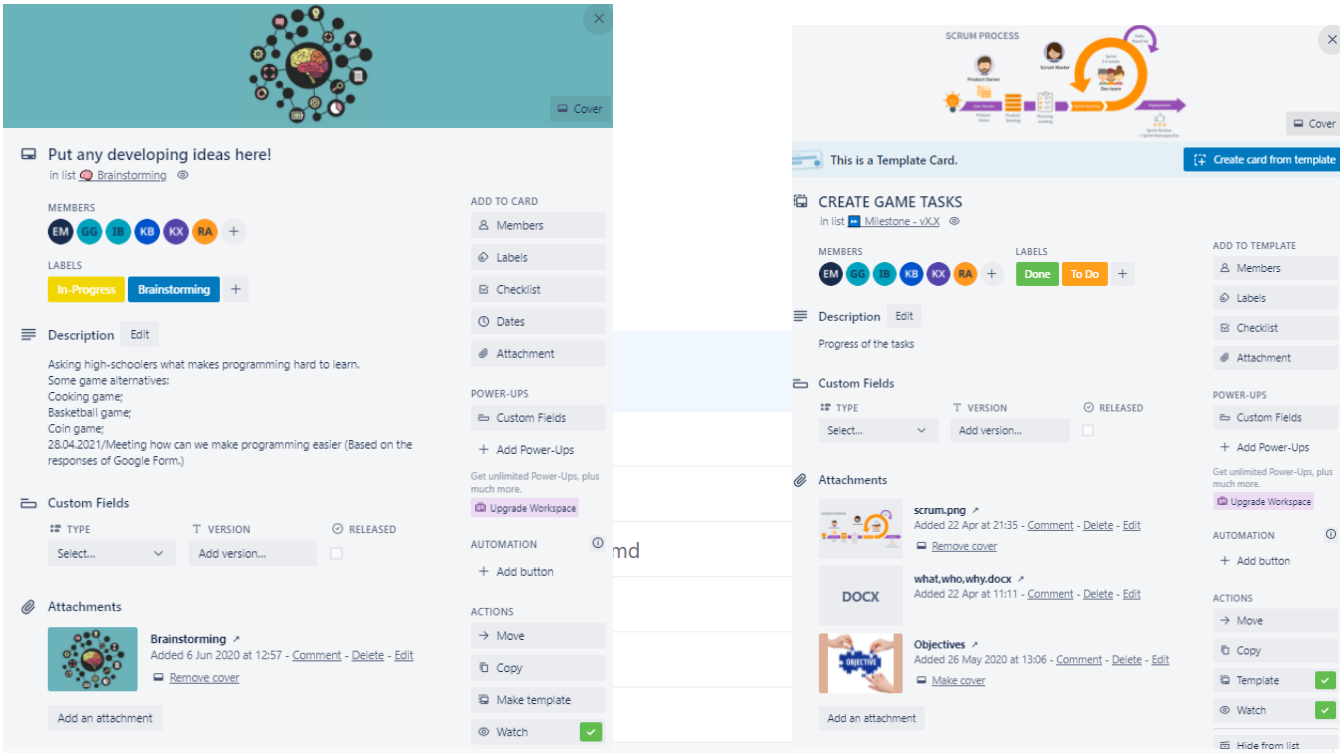
😡😡😡

Provoni shkollen si ne vendet nordike, pa shum gjera permendesh dhe tekste me koncize, me te formuluara shkurt qarte dhe mire

#3

We have worked with platforms: Trello, GitHub and WhatsApp. In Trello are presented our whole progress, from choosing what type of game we would work till the background alternatives. Meanwhile in GitHub we have all the coding process, the elements of the game, the questions that are going to be asked etc. Below are presented some screenshot of Trello and GitHub.





requirements.txt	Demo	3 days ago

omega0override Update README.md — 58e5a8a — 3 days ago — 23 commits

DocumentationFiles	Demo Version	3 days ago
Files	Demo Version	3 days ago
README.md	Update README.md	3 days ago
get-pip.py	Demo	3 days ago
levels.txt	Update levels.txt	16 days ago
main.py	Demo	3 days ago
requirements.txt	Demo	3 days ago

README.md

Turtle-Game