Jevgenij Ivanov, 20748055, jevgenij.ivanov.2021@mumail.ie
CS353, Project Documentation, Group 8
Screencast and Project repo was uploaded by Jevgenij Ivanov.

CombatClicker.io

Chapter 0: Introduction to Agile Development

Agile development is a popular software development approach that emphasizes flexibility, collaboration and rapid iteration. It was first described in 2001 in the Agile Manifesto and has since become the largest software development approach, with an estimated 85% of organizations using some form of agile methodology according to the State of Agile report published in 2019. There are many reasons why agile has become so popular since its conception, it focuses on delivering working software quickly and is highly flexible, allowing teams to adapt to changing requirements quickly throughout development, this is attractive to businesses as it allows them to respond to changing customer needs and markets. Agile development is based on the idea of building small increments of software within sprints, which are usually 2-4 weeks. This incremental approach allows teams to receive and respond to feedback from users and stakeholders throughout the entire development cycle.

There are many different forms of agile, such as Scrum, Extreme Programming and Lean. Each utilizes different tools and practices, but all share the same core values of Agile as set about in the Agile Manifesto. For our project we used the Scrum approach, one of the more popular methodologies of agile development. Scrum was first developed in the 1990s by Ken Schwaber and Jeff Sutherland. It is an incremental and iterative approach with a focus on short sprints. It has 6 key principles: control over the empirical process, self-organization, collaboration, value-based prioritization, timeboxing and Iterative development.

Scrum consists of three primary roles: the Product Owner, the Development Team, and the Scrum Master. The product owner is responsible for representing the interests of the product's stakeholders. They are responsible for creating the product-backlog, which is a list of features and requirements that the team will need to fulfil. They need to be able to communicate the product vision clearly and concisely to the development team, and will make decisions on what features should be built next. It is important that they are not vague, as the clearer the task, the easier and quicker the development team can complete it. It also ensures that stakeholders get what they ask for, if requirements are left for the developers to interpret it can lead to a vastly different product than what was originally envisioned and invested into. They will evaluate and reject or accept work done by the development team and should clearly outline what needs to be built and ensure the team understands these tasks. They are the key decision maker of the process and decides what work the development team will do.

The development team are responsible for building the software, they need to choose appropriate technology and techniques to complete their work, deliver working software by the end of each sprint. They work with the product owner to refine and clarify the backlog and to communicate their progress and any obstacles to they have encountered throughout development. Good communication, clearly defined tasks and collaboration with the product owner are key to a successful scrum team. The development team will need to participate in daily scrums, sprint reviews and in sprint planning meetings regularly. Their main task is to fulfil the requirements within the product-backlog by completing the tasks in their sprint-backlog.

The scrum master is the person responsible for ensuring that the development team follows the scrum process and is responsible for aiding in resolving any issues that arise during development. They are responsible for organizing all of the daily scrum meetings, sprint planning sessions and sprint reviews. They will also aid the team throughout development and will help clear any obstacles which they encounter. It is their responsibility to protect the team from disruptions and will need to coach and mentor development team members. The product owner will work closely with the scrum master. Together these three roles make up the scrum team and are all essential for a productive development cycle.

The sprint planning meeting is a time-boxed event that occurs every 2-4 weeks. In this meeting, the team will plan the upcoming sprint. The team will collaborate to identify the most important tasks from the product backlog and from this create a sprint backlog. This is a list of tasks assigned to different development team members to be completed in the upcoming sprint. The development team will estimate each task's size and complexity and will discuss any risks that may impede their work. It is an opportunity for the development team and product owner to work together to ensure that the team is focusing on the most valuable work.

The daily scrum is a short daily meeting in which the development team will discuss and plan the next 24 hours. Usually, it is about 15 minutes where each team member will explain what they did yesterday to help the sprint reach its goal, what they will do today to help the team meet their goal and what obstacles they encountered or any that are currently in their way. This meeting should be short and to the point, it is not a meeting to solve problems in but rather it is a chance for the team to coordinate together and to make sure they are on track to reach the goals of their sprint.

The sprint review will occur at the end of each sprint, it allows the development team to demonstrate their work to the product owner and stakeholders. Here the stakeholders will give feedback and ask questions about the product. The main goal of the review is to gain feedback and to ensure that the project is up to standards and on track. This is a chance for the development team to show their progress and report any problems they encountered. The stakeholders will also get to see their product be built and improved incrementally and will allow them to make any changes in requirements as they might change their mind after seeing the working feature or may want to add, remove or change features.

The product backlog is a prioritized list of work which is continuously refined and reprioritized as work on the project continues. It acts as a guide during sprint planning meetings and is compromised of user stories with a description of the feature that the user requires. These user stories are usually written in natural non-technical language it will showcase the exact requirements of each feature, these are listed in a priority-based order with more urgent features listed at the top. User stories are taken from the product backlog and are estimated in how long they will take to complete. The tasks for these user stories are identified and then added to the sprint backlog, which is to be completed during the next sprint. This way the most important tasks are worked on and completed first, and the product is built up in iterations.

The burn-down chart is a graphical representation of work remaining in a scrum project. It is used to track and progress and to visualize the amount of remaining work in the current sprint. Typically, it is updated on a daily basis and helps the development team and stakeholders understand the state of the project and helps identify issues that need to be addressed. As the sprint progresses the time remaining on the sprint, represented of the x-axis heads towards zero. The amount of work remaining is shown on the y-axis. This is to show that the team is making progress on the sprint goals. It is a simple way of tracking progress and to ensure that the team remains on track to reach its goals. It is also a way to demonstrate to stakeholders that work is being done on the project.

These are the main factors to make up a scrum project. It is a well-defined and easy to follow framework with 3 roles, 3 events and 3 main artefacts which help the team to remain on track, get feedback and to deliver working products quickly. This approach is highly flexible and allows the project to adapt to changes throughout its lifespan which might seriously disrupt different approaches. It gives a huge benefit to stakeholders who can see reliable progress being made and allows them to easily communicate with the team so that they can give feedback. It provides a clear framework for maintain and tracking progress, which helps teams to stay organized.

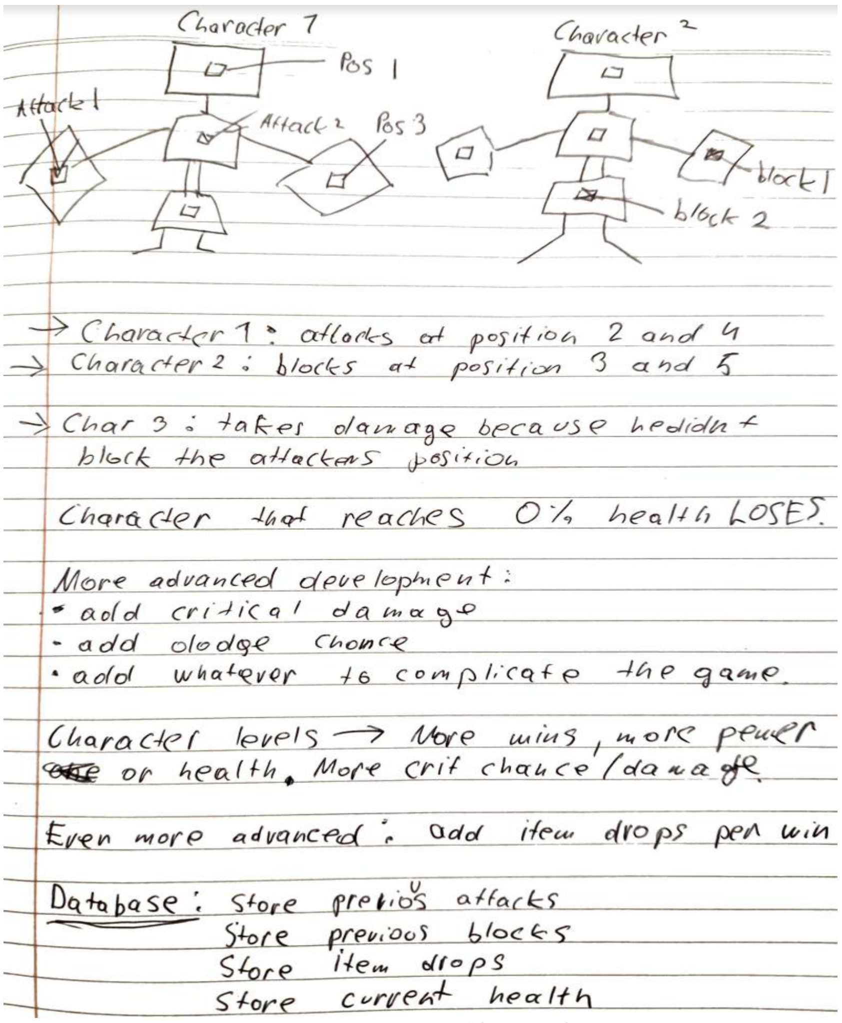
There are some disadvantages and situations where scrum is not ideal for. Scrum requires a significant amount of commitment and discipline from the team members and can be difficult to maintain and implement if team members are not fully invested in the project, if it were a side project for example where team members could not work on it reliably sprints become much less effective. It requires frequent meetings, which can be time-consuming and might not be suitable for teams with fewer resources. With larger organizations comprised of many teams it is difficult to implement and will require a high amount of coordination and communication between teams which can be time-consuming and the cost of resources required to maintain might outweigh the benefits of scrum. For projects with strict timelines and limited budgets it may not be suitable, as the focus on flexibility on changing requirements and features may make it difficult to commit to some of the core fixed requirements. It is designed for complex projects and may not be necessary for more simple, low complexity projects. Also, for projects that are in stricter and regulatory environments, Scrums focus on flexibility and delivery may not be suitable where strict controls and procedures are in place. Also, for projects which rely on external teams and systems, it may be too difficult to coordinate across different teams.

For most software development projects, the scrum framework would be suitable. Especially in long projects that may take many years to complete, as the initial vision and requirements for the product are likely to change over long periods. For example, a competitor might release new a product, market changes, company changes or laws and regulations could be introduced which would require the project to adapt. Outside of software development, scrum is used in other industries like finance and marketing. Both start-ups and large organizations use scrum, which is a testament to its flexibility and easy to follow procedures. The ability to deliver a working product quickly to stakeholders and customers makes it especially popular as it gets more feedback and boosts confidence in the project, this can be beneficial in getting more funding as-well as investors can see that significant progress is being made throughout development.

Chapter 1: Project Proposal

In our very first group meeting, our team of 6 gathered at the table and presented the project ideas that we will be working with for the next 10 weeks or so. The main 3 project idea proposals that we had were as follows: Denis came up with a Sports Management app idea, he had laid out some basic sketches and wireframes. Conor suggested making a Crypto wallet and had some sketches ready as well. And the third idea we had was a browser game that I came up with. The rest of the team did not have any concrete ideas or sketches to present, so we had to chose one proposal from the three that were on the table. In the end, each team member voted for the project idea that they were interested in, and my idea won the majority decision. We decided to call this browser game as "CombatClicker".

The game I suggested to make was quite simple, something of a mixture between "rock, paper, scissors" and "battleships". A turn based game where the user attacks 2 positions out of 5, and then defends any of these positions in the next turn. So in our game the 5 positions were Head, Body, Waist, Legs, Arms. The player selects Head and Body, and if the opponent selected Head and Body to defend, he would not get hit. Similarly, in "Battleships", if one player selects position x to attack, and the opponent does not have a ship in that position, the opponent does not take damage. After a bunch of rounds of attacking and defending, the player who reaches 0 life points loses. Below is the sketch and description of the game that I presented to the team during the project proposal meeting:



→ Character 1 : attacks at position 2 and 4
→ Character 2 : blocks at position 3 and 5

→ Char 3 : takes damage because hedidnt block the attackers position

Character that reaches 0% health LOSES.

More advanced development:
• add critical damage
- add dodge chance
• add whatever to complicate the game.

Character levels → More wins, more power ~~one~~ or health. More crit chance/damage.

Even more advanced : add item drops per win

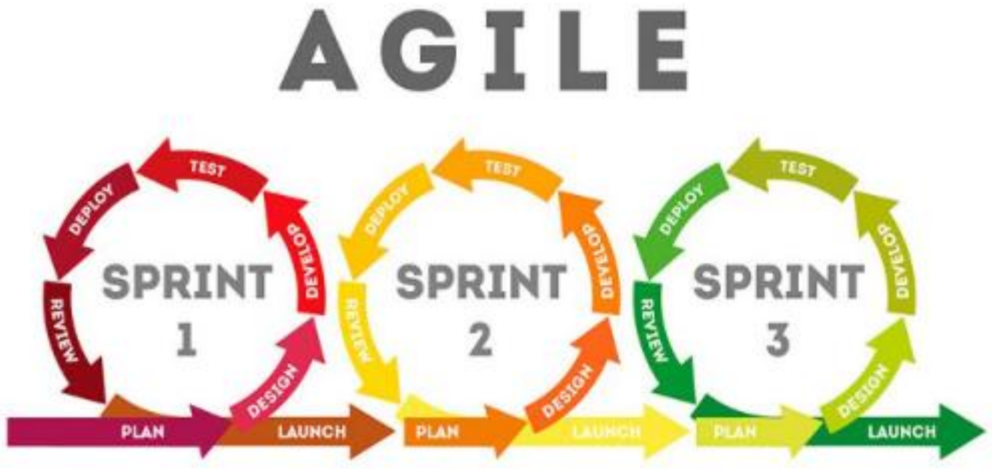
Database : Store previous attacks
Store previous blocks
Store item drops
Store current health

Initial sketch of proposed project "CombatClicker" [fig 1.0].

There were many reasons as to why was my project proposal was selected. In my opinion, one of the main reasons was that I have already pre-planned the core mechanics of the game and how it would look like on a browser page. I described the game in detail to my team and explained clearly how the game works, what technologies we can use, how difficult or simple some parts of development could be and so on. What was appealing in my idea, was that it was quite simple to build on a fundamental level. Basically, we just had to make five check boxes to represent positions to attack and defend, and add a couple of buttons to perform certain actions. Even though the core idea was simple and easy to understand, there was a lot of room for adding various complex functionality to the game after the fundament of the game has been laid out. I have articulated some of these functionalities to my team during the project proposal meeting. Some of the complex functionalities that I described were:

- Adding a multiplayer game between 2 people, either through local or network connections.
- Adding character progression, in form of levels, attack power and other stats.
- Adding various combat functionality, such as critical hits, dodge chances, damage over time and so on.
- Adding character inventory, which could store various items that would affect character power and stats.
- Add an adventure campaign where a user would defeat enemies and progress through game chapters and achieve higher levels.

This setup would allow us to work in an Agile environment, where people in my team would design a functionality, write code for it, test if it works, upload it to our GitLab repository, and then we would review it during our next scrum meeting. The project that I proposed worked perfectly for this purpose and everybody in the team agreed. Members of our team could deliver a working version of our game in small increments each sprint, and everyone in the team could interact and play with it, as well as see improvements and changes to the product quite often.



Agile framework from scrumstudy.com [fig 1.1].

After talking about all the possibilities and content that we can develop during the project, it was clear that we have a solid plan and structure for the project, so everyone in the team were on board and excited to start developing this browser game.

Another reason why my project idea was selected, is that everybody in my team has previously played some video games and enjoyed them quite a bit. We had an opportunity to create something with intrinsic motivation and have fun while doing it. With very little effort, we could create a playable game from the very beginning, and then play around with it during our future meetings and add or suggest new functionality or changes for our already playable game. This would keep pushing our team to be creative and interact with the project in an interesting way.

The ideas just kept coming and coming from everybody in the team. People just kept expanding on existing ideas and improving/changing them. We started discussing similar games and what we can take and use from them. For example, Pearse told us that he played a similar game in the form of a football penalty shooter, where a user would have to select a position where to kick the ball (e.g. left corner, right corner), and the goalie would either catch the ball in that position or miss it. So, from day one we already had something to kickstart our project from, and it felt good that everybody in the team liked my idea and were on board to start developing it.

Our team started researching and finding similar games online to get some inspiration and direction of our project. As stated earlier, Pearse found the penalty kicker game and showed it to the team. It helped us to visualise of how we might structure our own game. In my personal research, found a similar game online on the website https://www.carnage.ru/. I opened the game during the meeting for everybody in the team to have a look and take notes of what they like and don't like, which parts might be useful to our game and what some of the user interface elements could fit well in our game. Our team did not directly copy any of the code from the game as it was way too complex and advanced for our project, but we acquired some valuable information and ideas just from looking at the game and its structure.

I have also found a code for "rock, paper, scissors" game online. Our group played around with it during our meeting and all mutually agreed that we could use the code from this game to start writing our own code for CombatClicker. The idea behind it was that we could take the rock, paper, scissors options from the game and change them to head, body and legs. The link for the website with "Rock, Paper, Scissors" is https://codepen.io/nevan/pen/kVBmom. The code we used from this website was mostly HTML, and did not use any CSS code because the style just did not fit for our project. JavaScript code was also taken, but almost all of it was changed eventually to match the core gameplay of our CombatClicker game. It was just a good starting point to begin coding up our own browser game.

Since our team had different project proposals and ideas, we decided to not let it go to waste, and use some of the proposal ideas for the CombatClicker game. For example, from a Sports Management app that Denis suggested, we could use some of the wireframes for the menus. Our game could use a functional menu for the end user to navigate through. Since Denis had some sketches ready, he volunteered to start developing various menus for our game. The menus included a Login page, Settings page, Character profile page and many more. Conor's idea of a crypto wallet was also quite useful for CombatClicker. He volunteered to develop functionality that would send various information to a database such as Firebase. He could take for example log in details of a user, or record positions of attacks and defence from a game and store it in the database. This would also be extremely beneficial to our project. Everybody in the team could use something from their initial project ideas and contribute to the main project we all agreed on. This also helped to assign tasks to everybody in the team easily. Everybody in the team knew exactly what they needed to do and what is expected from them during the following sprints.

In conclusion, we had some very interesting project proposals during our first meeting, and I'm glad that people that had their ideas rejected did not lose motivation to work on the project that I suggested. In fact, I felt the opposite, as people gained a lot of motivation and drive for this project. We were all looking forward to the upcoming sprints and excited to see what we can create in the future.

Chapter 2: Technologies/Architecture/Testing

The entire project was built on vanilla JavaScript, HTML and CSS. We chose JavaScript because it is commonly used in the development of web-based applications, which is perfect for our browser game, CombatClicker. JavaScript is also an object-oriented language, which is a huge bonus to our project since we can use objects such as Player, Computer, Items and many more. Using object-oriented programming helped us split the game into smaller chunks and made it much easier to develop the game, because when encountered an issue in a specific part of the game, it was easily solvable by navigating to the class where the issue occurred. If the computer did too much damage, we could just go to computer class and edit the values for damage inside that class. JavaScript also has been used to handle various game logic, user input and other interactive features during the combat. The majority of code for CombatClicker was written in JavaScript.

HTML (HyperText Markup Language) is a markup language that was used to structure and format all of CombatClicker's content on the web. HTML was the foundation upon which our browser game was built, and it was where our project began. It allowed us to specify the layout of the game, its structure and various content such as character images and background wallpapers. Our HTML content also included static content, such as different styled texts appearing on-screen whenever some action happened during gameplay. For example, whenever one of the five checkboxes was selected, a text would appear on-screen informing the user of which position he or she selected. Of course, some JavaScript code was also written to support the functionality of the HTML content. Additionally, our team created game menus, log in and registration pages using HTML code, and it worked very well. The user interface and experience was designed through HTML and styled with CSS.

CSS played a huge role in developing CombatClicker. CSS is short for Cascading Style Sheets, and it is a stylesheet language that is used to control the appearance and formatting of the content we have on the web. HTML and CSS worked hand in hand together to create a visually appealing user interface for our browser game. It allowed our team to specify how our HTML elements should be displayed on the page, including various colours and gradients, text fonts, layouts and many more. A lot of trial and error technique was performed when working with CSS code, to find the most fitting visual appearance of different parts of the game. In our project, we used CSS to assign positions to different components on the web page, for example, some components had to be set to absolute, fixed or relative to avoid overlaps and correct display. The positions played quite an important role in making sure that the game could be opened on any device or laptop, and the components would not overlap even if the screen resolution would be very small.

Our project needed a database service to store various information about the game and the users that play it. We chose to work with Firebase as it is a popular platform that offers powerful tools and services for building and managing applications. Firebase allowed us to store and manipulate data in a number of ways. It provided us with a real-time database service that allowed us to store game content such as game states, record the combat log of previous matches, record leaderboards (who has beaten the computer most times) and many more. In CombatClicker project, Firebase was primarily used for storing the login information of users that registered their account for the game. This information included emails and passwords. Firebase also provides a comprehensive authentication service that allowed our project to implement user authentication. The user authentication we implemented was for the login details. During the registration, a user would have to provide a legit email in xxx@xxx.x format, as well as a password that must contain lowercase and uppercase letters, numbers and special characters. After the user registered, this data was then stored in our Firebase database. In order for the user to enter gameplay, he must input previously registered details on the log in screen, otherwise the access would be denied. After the user inputs log in details and presses the login button, our Firebase would then check if such account details exist, and then authenticate the user and allow log in. Additionally, if a user breaches the terms and conditions of CombatClicker, or abuses the game in any way, we are able to block or restrict the account of the user through Firebase. In the future development of CombatClicker, Firebase will be used to store stats of each player and their levels, as well as store item's information that a specific user has equipped or stored in their stash.

Our team had a plan to host our CombatClicker game online on a web server, however, we did not find a service that allowed us to use a domain name and web hosting for free. We looked at different host providers such as GoDaddy, Bluehost and HostGator. Since nobody in the team wanted to pay, we have found a workaround solution that allowed us to host a website for free. Using VSCode, we created a folder and added all of our project files (HTML, CSS, JavaScript and images) into it. We then opened Google Drive and uploaded our project folder there. Then, if you right-click the uploaded folder, there is an option to share and make "Anyone on the internet can view with link". You can then go to "drv.tw", click "Google Drive" and link it to the Google account that we created just for this purpose. After all this was done, we could see our browser game with the folder name "CombatClicker". We then click on the shareable public link, and CombatClicker is now accessible from any device. CombatClicker was now hosted online, and we could send the link to other people in our course to try out the game and give us some feedback.

During development of CombatClicker, our team looked around for JavaScript frameworks and libraries that would help the development process and enhance the flexibility of our game. We found that React JavaScript would have been a perfect library to use for developing a browser game such as CombatClicker. However, it was difficult to get all six team members to switch to React in the middle of the project, and we already had thousands of lines of code written. The project deadlines did not allow us to rewrite code and switch to React in time. There are a few reasons why starting the project with React JS would have been a great idea. One of the reasons is that React allows us to create reusable components that can be easily reused and shared throughout the game. This would help us write cleaner and more maintainable code from the start, and make it easier to develop and update our game if we continued to develop and implement more functionality to the game over time. React also has quite a strong community, with a huge number of resources such as tutorials and third-party libraries available. It was our mistake to not use React from the start of development, however we are now aware of how valuable React technology is, and we will definitely consider using it for future updates of CombatClicker.

To ensure that our game did not contain bugs in the code, our team had to perform various tests. The project did not require tests such as Unit tests or Integration tests, as there were not that many functions or components to work with. Most of the game testing was done manually to ensure the high quality of our game. Manual testing included playing through the game and trying every available feature. There were numerous of bugs throughout the development process, for example, the computer would sometimes choose the same position to attack twice, which was unfair because the player was only allowed to block once. Additionally, we performed "End-to-end" tests, and asked students from other groups to try out our game from an end user perspective. This was very beneficial to us because there were moments when someone who never seen the game before would try it, and spot some inconsistencies or bugs that were not so obvious to developers who seen the game hundreds of times. Our team would then write down the issues, and then try to reproduce it in our own time. In the end, all of the bugs were fixed, and no new bugs were spotted during play testing of over hundreds of times.

Overall, the technologies and architectures we used in CombatClicker played a significant role in determining its functionality and performance. By utilizing a combination of JavaScript, HTML and CSS, as well as a Firebase database for storing and manipulating user information, our team was able to create a functional and fun online browser game that could be accessed and played by other people from anywhere around the world.

Chapter 3: The Scrum Process

Initial SCRUM meeting:

In our first meeting, we had a lot to discuss as a team in order to decide on a project and start making progress. Our first task was to present our different proposals for the project to one another, and we carefully considered the pros and cons of each one. Some of the proposals were seen as much more ambitious than others, and we ultimately decided to vote out the ones that we felt were beyond our capabilities or resources to develop. Other proposals were considered too simple or vague, and we also voted out the ones that didn't have the support of the entire team.

After much discussion and debate, we were down to our final two proposals, and we held one final vote to determine which one we would move forward with. In the end, we agreed that our project would be CombatClicker.

With the project chosen, we then began brainstorming for ideas and creating a vision for the completed project. To get a sense of what we wanted to achieve, we searched the internet for other similar games and looked at their functionality and features. We then came up with a list of features that we would need to work on and the rough format of the website.

Each member of the team also discussed their skills and preferences in web development, and it became clear that many of us were much better at and preferred working on the front end. For example, one of our members was a multimedia student who volunteered to create game sounds and music, as well as work on creating logos and art for the project as svg's. Other members preferred working on the back end and with databases, and they volunteered to fill these roles. By the end of this discussion, our team of six was split into primary roles for the duration of the project, with three members working on the front end and design side, and three working on the back end.

We also spent some time discussing what frameworks, libraries, and other tools would be useful for the project, but we didn't make any final decisions at this point. We then created a WhatsApp group chat that we would use as our primary form of communication, and finally, we were all assigned tasks to complete during the first week's sprint.

Project Estimation:

One of the first tasks in our project was to estimate how long features would take to complete. There were many features which were much easier to estimate than others. Planning poker is a technique used for estimating the time required to complete tasks throughout a project. Before we could use planning poker, we first had to create some user stories, a user story is a short description of a feature written from the point of view of a variety of users that helps describe the functionality of a feature that a user wants from a system. These are written in normal, non-technical language. Each team member wrote some user stories which we then discussed in our next meeting.

From these stories we identified many of the core features which our project needed and more trivial features which were not as urgent and which we could put aside for the time being. We used days as our units of time for this process and assigned each of these tasks an estimation on how many days it would take one person to complete. For example, we agreed that making a basic home page in html would only take 1 day but that creating functionality for the gameplay would take much longer. Some of these tasks took much longer than expected as unexpected problems arose and team members often had to learn to use new tools which extended the time required to complete a task.

Admittedly, we did not follow and use planning poker as the project continued. If we had used it throughout it would have helped us identify which areas needed more attention and would have allowed us to visualize our progress much more clearly. But in the first weeks of the project it was a good tool which we used to build the basics of our project.

Meeting minutes:

During our weekly minutes one person was assigned the task of recording our minutes. Which was a summary on what we discussed, what work we had completed in the previous sprint, what tasks we were going to work on in the next sprint and any issues that arose. These were then posted to our MS Teams group channel. Below are the minutes for each week.

Week 1:
Starting with showing off the index cards showing each other what we expect of the game and where we think we can bring it. Moved onto planning poker phase seeing how long the registration would take which we think will be 3 days consisting of username email password. Home screen we will have nav bar and allow to start game session from there. Made our gitlab group and everyone joined Named game CombatClicker.io

Denis, Pearse, Ciaran – front end

Daniel, Jevgenij, Conor – back end

Week 2:
Conor added team to firebase. Set up Google VM, Decided tasks for next week:
Set up CRUD
Make base game in JS
Create SVG's?
Add some gameplay elements
Base Homepage

Week 3:
Denis and Ciaran are working on the settings page, improving functionality and design. Pearse is designing a background for the home page and login pages, it needs to be an SVG file so that everyone in the group can access it. Jevgenij is looking into connecting the main menu with the gameplay, as well as adding more functionality to gameplay, such as adding check boxes. Conor and Daniel are working together to connect the front end login system to the back end crud, and making the database work properly.

Week 4:
Denis got the settings page up and running with basic options. He will work on adding additional functionality. Ciaran has created a basic layout for the game page, he will implement this with Jevgenij's work to create a better page. Conor and Daniel have worked on creating an API for the login system. We will continue to advance the API and connect the front end to the database to create, save and update users.

Week 5:
Denis upgraded the login page drastically, adding a lot of new UI, UX, new background and more functionality. With the feedback he received, he is planning to change the background a bit to match a medieval style.

Ciaran has created a new prototype for main gameplay, adding a visual character for computer and improving more UI during gameplay. He will work with Jevgenij to combine his prototype and main game to make one great gameplay page. Conor has worked with firebase database, and he managed to fill login details into the database and make it work. Next step for Conor to add more tables to database, such as game combat log and other features. Daniel worked on some more menus, including login page and added more functionality to it. For the next sprint, Daniel will try and implement Routing to our project, enabling switching pages when menu buttons are pressed. For example when user presses Play, routing redirects the user to gameplay page. Pearse is working with a title screen, creating an svg image that appears when you first launch the game. Additionally, Pearse will use music programming and add some game music in the background, and add sounds to button presses. Jevgenij has developed more of the core mechanics of the game. Removing previous buttons and move all of the button functionality to check boxes. Jevgenij added a combat log to keep track all of the events that happened during the battle. He also added more visual improvements to the gameplay, switching colors when functionality switches between attack and defend. Added background image. Additionally, Jevgenij split the code into various classes, such as Player, Computer and Checks classes (OOP). This will add clarity to the work project, enabling other team members to be able to work on the code as well and make changes to different classes. Main objective of the whole team work right now is to connect our individual pages (Title screen, Login page, Main menu, Setting menu and Gameplay screen), so that with correct button press, the user is redirected to another page. This will be achieved through routing.

Week 6:
Conor will make a post method to update changelog and other data. Ciaran will add routing so that pages will be switched while in node. Gamesounds have been created and will be added in, some more may be created. Debugging continues. Denis will work on implementing the front end for the settings page.

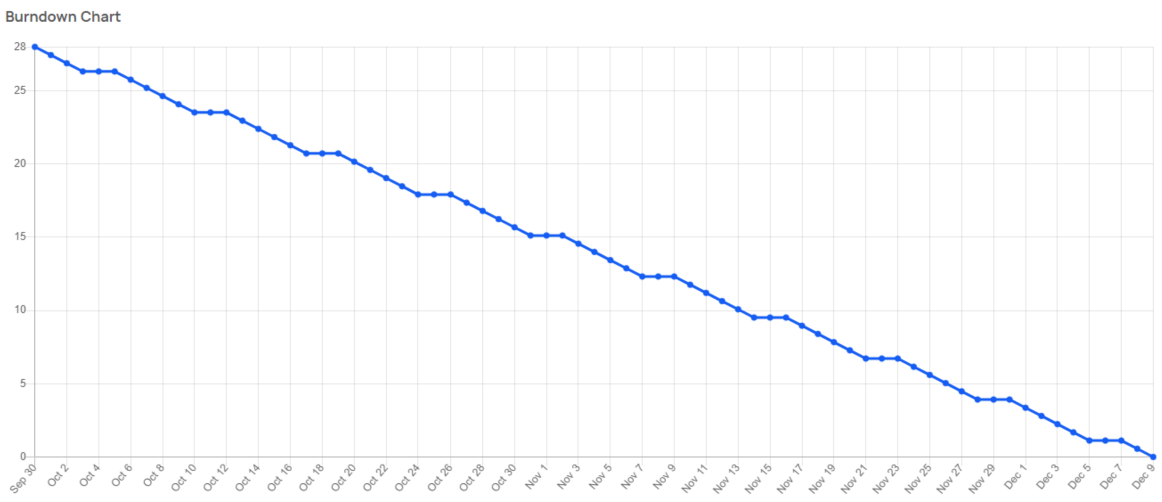
Week 7:
Denis was working on getting the background music to play on page load, he was only able to get it working through an audio player which defeats the purpose. He will continue to work on it and get it to automatically play. I was working on some documentation and fixing some small bugs. Jevgenij was working on documentation also and testing/fixing bugs. Conor and Ciaran managed to get the page routing done and implemented the database functionality.

Overall, the way our group recorded team minutes was not ideal. Often times not all of the data was recorded, because someone forgot or left the meeting early. It was hectic assigning a different person every week to record meeting minutes. It would have been much better to have one person work on the minutes for the whole duration of the project, or have each person to write their own minutes into a separate file. Week 5 was the best recording for a meeting done, it went into every detail of what each team member is doing and will do. Our team learned from this and there are many ways we could improve for future projects. For example, instead of writing minutes into team chat, we could create an excel spreadsheet with each team members name on a separate row. Then have a column for meeting minutes, a column for work to be done, and a column for marking tasks completed: green (completed) and red (to be completed). This would keep the project workflow more organised and all of the team minutes recorded correctly.

Burndown chart:

A burndown chart is another common tool used in SCRUM projects; it is a graphical visualization used to demonstrate how much work is left in a project. As seen in the graph below the y-axis represents the total units of work left in a project, these are called story points and are made and assigned values in the story mapping phase. The larger are more complex the task the more points it is worth. The x-axis then represents the time remaining in a project, usually in days or weeks. As these story points are completed the line slopes down towards zero, ideally this chart should remain constant as work is reliably completed. It is a tool used to identify any deviations in progress compared to what is expected. If the line slopes upwards, it means work is falling behind schedule and that the development team is encountering issues. This is one way a team manager can see if action is needed to aid in the productivity of the project. It is also a way to estimate when the project will be completed and if it will reach the given deadline, this way delays can be expected and addressed months or weeks in advance.

For our project we did not utilize a burndown chart as we felt that it may be an unnecessary tool that we would have to divert time and effort into tracking and updating, but it would have been a useful representation to show how much progress we made by then end of our project and if deadlines and our expectations were met.



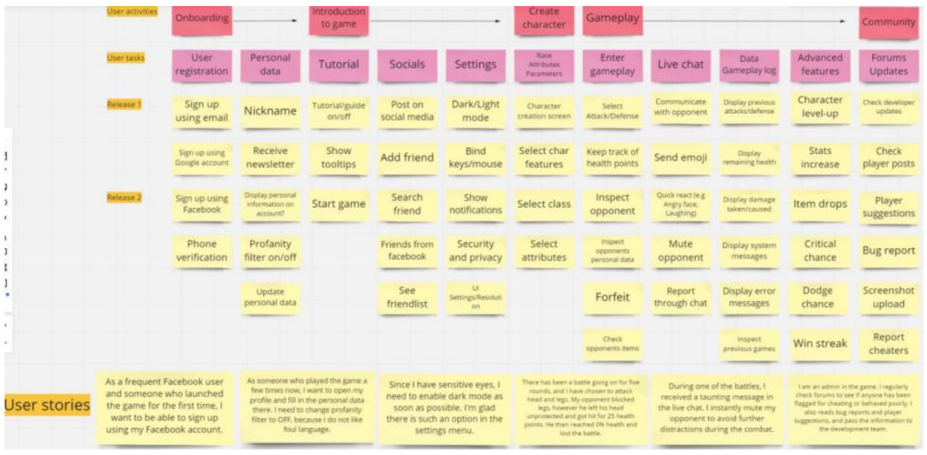
Burndown chart [fig 1.2].

Story Maps:

A story map is a graphical representation of the user stories that make up a project. It helps to visualize how a user will go through the different stages of the project, starting with the initial sign-up process and logging in, and then moving on to the homepage or other features of the project. These features are organized into categories, which are usually represented as columns on the story map. For example, the login and sign-up features might be placed in the "user management" column.

The user stories that make up the story map are simplified down to their core functionality and are placed in these columns according to their importance. The most basic and important features are listed at the top, while the less important ones are placed lower down on the map. This process gives an excellent overview of the ideas that we had for the features of the project, and helps to identify what work needs to be done.

It's worth noting that not all of the features that are listed on the story map will necessarily be implemented. Some of them might be deemed too complex or not necessary for the project, and they may be removed or put on hold. In the case of the project mentioned in the original statement, it seems that some of the multiplayer functionality was not implemented, though it was listed on the story map.



User activities	Onboarding		Introduction to game			Create character	Gameplay				Community
User tasks	User registration	Personal data	Tutorial	Socials	Settings	Race Attributes Parameters	Enter gameplay	Live chat	Data Gameplay log	Advanced features	Forums Updates
Release 1	Sign up using email	Nickname	Tutorial/guide on/off	Post on social media	Dark/Light mode	Character creation screen	Select Attack/Defense	Communicate with opponent	Display previous attacks/defense	Character level-up	Check developer updates
	Sign up using Google account	Receive newsletter	Show tooltips	Add friend	Bind keys/mouse	Select char features	Keep track of health points	Send emoji	Display remaining health	Stats increase	Check player posts
Release 2	Sign up using Facebook	Display personal information on account?	Start game	Search friend	Show notifications	Select class	Inspect opponent	Quick react (e.g Angry face, Laughing)	Display damage taken/caused	Item drops	Player suggestions
	Phone verification	Profanity filter on/off		Friends from facebook	Security and privacy	Select attributes	Inspect opponents personal data	Mute opponent	Display system messages	Critical chance	Bug report
		Update personal data		See friendlist	UI Settings/Resolution		Forfeit	Report through chat	Display error messages	Dodge chance	Screenshot upload
							Check opponents items		Inspect previous games	Win streak	Report cheaters
User stories	As a frequent Facebook user and someone who launched the game for the first time, I want to be able to sign up using my Facebook account.	As someone who played the game a few times now, I want to open my profile and fill in the personal data there. I need to change profanity filter to OFF, because I do not like foul language.		Since I have sensitive eyes, I need to enable dark mode as soon as possible. I'm glad there is such an option in the settings menu.		There has been a battle going on for five rounds, and I have chosen to attack head and legs. My opponent blocked legs, however he left his head unprotected and got hit for 25 health points. He then reached 0% health and lost the battle.		During one of the battles, I received a taunting message in the live chat. I instantly mute my opponent to avoid further distractions during the combat.		I am an admin in the game. I regularly check forums to see if anyone has been flagged for cheating or behaved poorly. I also reads bug reports and player suggestions, and pass the information to the development team.	

CombatClicker story map with most of the features [fig 1.3].

Testing:

As in any software project testing was an incredibly important process. Whenever a new feature was added or changed it would need to be tested. It was important to ensure that features and functionality were working as intended. It was also important and useful to test each other's code as it was uploaded to make sure that it was working on different machines and browsers. We also needed to test features while being hosted locally by node as the live preview feature in VSCode is not sufficient. Often a feature would work in preview but then would not when hosting, this allowed us to fix import issues for example.

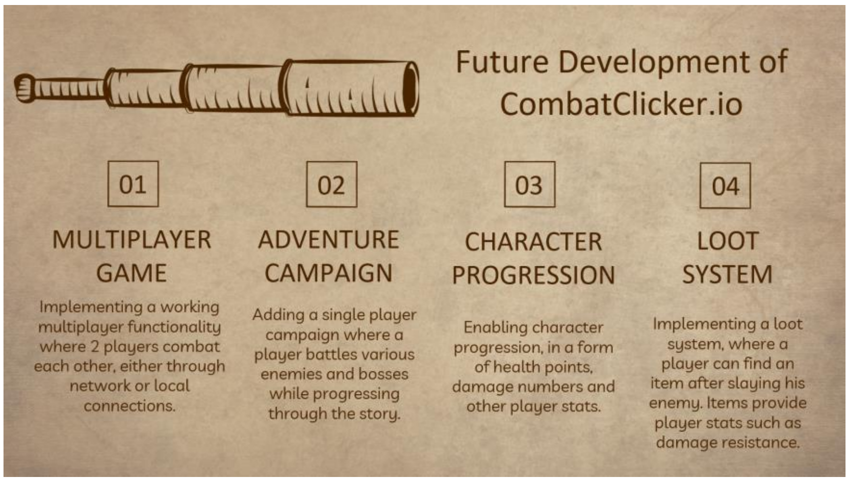
Running the entire project in node is a good example of system and integration testing, through this we saw many issues that arose when running the project as a whole and while trying to integrate different units of code written by various people. Often problems arose when running someone else's code as the required tools that were needed to run it were not specified for example.

By regularly testing eachothers code we could find bugs and issues that needed to be addressed, we could then report these to our teammates so that they could be fixed. For example, when testing the front-end visual issues were spotted or when the project was updated previously working features became broken. Another issue was signing up not working on every machine or unexpected firebase errors.

One method of testing which we found incredibly beneficial getting friends and other non-team members to try some end-to-end testing. This is a testing method where a user will test a system from start to finish. Getting people who didn't have a computer science background was incredibly useful as a way of seeing if the regular person could use our website. We had our friends sign up, then login and play the game and asked them to try and break it. Doing this we found ui and ux issues where users did not know how to navigate through the website properly and found some glitches in the gameplay. These tests allowed us to get a fresh perspective on the project from people who hadn't worked on it and didn't understand all its functionality. Through this method and testing on our own we fixed most of the bugs and glitches, especially those found within gameplay. Our user interface was also redesigned midway through the project based on the feedback we received.

Future releases/sprints:

Our team has many great ideas for future development of CombatClicker.io. The four main features that will be added in future sprints are Multiplayer game, Adventure campaign, Character progression and Loot system.


Future Development of CombatClicker.io

01 MULTIPLAYER GAME
Implementing a working multiplayer functionality where 2 players combat each other, either through network or local connections.

02 ADVENTURE CAMPAIGN
Adding a single player campaign where a player battles various enemies and bosses while progressing through the story.

03 CHARACTER PROGRESSION
Enabling character progression, in a form of health points, damage numbers and other player stats.

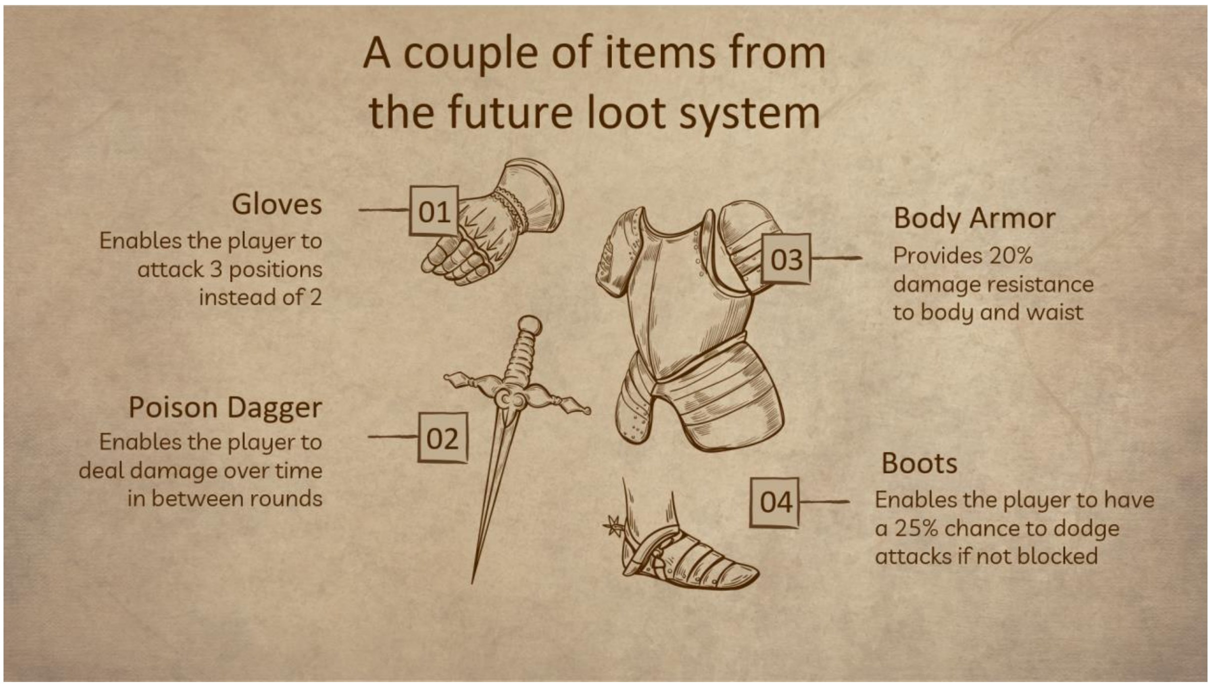
04 LOOT SYSTEM
Implementing a loot system, where a player can find an item after slaying his enemy. Items provide player stats such as damage resistance.

Multiplayer game would be a great addition to the game, either through local or network connections. The game plays quite similarly to "battleships" or "rock, paper, scissors", therefore a lot of mind games could come into play during the gameplay between two players. This would be a large and complicated task but would improve the project massively. There could be a player vs player mode, a global leader board and ranking system, a chat and social system. Adding multiplayer features would also require some form of moderation in terms of preventing cheaters or chat moderation.

Adventure campaign would be the continuation of already existing combat against a rat warrior. We could add ten more enemies that a player has to defeat, lets say we add ten more rats. First enemy would be the rat warrior we currently have with 100 health points, second rat could be a rat mage with 120 health points, and the final rat could be the most challenging boss with 500 health points. This campaign would be a great addition to the game and immerse the player for long periods of time.

Character progression is another important feature for future releases of the game. Any player of video games likes a sense of progression for their character. In CombatClicker, we will add a Leveling System, where a character would gain power in form of Health Points, Damage Points and other Character Stats. The character would become stronger with each enemy defeated, weather it is in adventure campaign or a multiplayer battle.

A loot system in CombatClicker would be the most interesting addition. There are so many ways you can change the functionality of the character. We can add an item for gloves, which would enable your character to select 3 positions to attack instead of 2.


A couple of items from the future loot system
Gloves
Enables the player to attack 3 positions instead of 2
01
Poison Dagger
Enables the player to deal damage over time in between rounds
02
Body Armor
Provides 20% damage resistance to body and waist
03
Boots
Enables the player to have a 25% chance to dodge attacks if not blocked
04
Examples of items from the future loot system [fig 1.4].

The image above shows more examples of possible items that can be added to the game to improve user experience. Additionally, a player could gain gold for defeating enemies or completing quests, with which they could purchase gear and items from a merchant.

Code versioning:

Version control is the practice of managing changes to a project over its development cycle. It allows developers to collaborate on a project and track changes easier. For our project we created a team in GitLab and would upload our code to a shared repository. This allowed us to work on different parts of the project concurrently without overwriting each other's changes and gave us the option to revert to previous versions of code. Every sprint we worked on our code from home and then uploaded this to the repository with version numbers, combatclicker1.1, 1.2, 2.1, etc… This was a powerful tool for allowing us to track record changes that were being made to the project.

We found that GitLab was excellent for sharing and uploading our code. It is a repository manager similar to GitHub, which was an advantage as many of us had experience with GitHub prior to this project for our own personal repository. It allowed us to create different branches, we created these at the beginning but found it unnecessary and decided to use the main branch instead.

GitLab has many advantages and tools which we didn't use in this project as it was unnecessary but would be incredibly useful in larger projects over longer timespans. For one it is free and open source but can be upgraded to the 'enterprise version' if a user wishes to. It is secure and has built in security scanning and the option to set up two factor authentication. A team can also choose to implement a number of plugins into their repository. However, it can be complex to new users at first who have no experience when it comes to GitLab or other Git like repository sites. I think that GitLab is a good fit for small to medium sized teams.

Team communication/ management

Communication and collaboration are major elements of an effective SCRUM project. In SCRUM there is an emphasis on face-to-face meetings and discussions which we participated in every week. There was also the benefit of most of the group sharing the same modules as we could discuss the project outside of our weekly meetings and ask for help and give suggestions mid-way through a sprint. For communication we used both MS Teams and a created WhatsApp group chat during this project. While MS Teams is much better at file sharing, screensharing and other features, WhatsApp was used as our primary method of communication as we found it much more convenient to use compared to MS Teams when talking day to day.

One of the elements missing from our team was a leader. This is an important role in any software development project and obviously in industry every team is going to have some form of management. Every week someone was assigned the role of Scum master, but outside of asking questions had no actual management responsibility. I think the project would have benefitted from a single leader to make decisions and to assign tasks as it would have increased productivity.

Remote working:

The vast majority of work done throughout this project was via remote working from home. The work done in the weekly meetings was mostly bug fixing and demonstrating our code to other team members. Due to the fact that this is a college project we did have many other assignments and duties to commit to, so some weeks finding enough time to do work on the project was a challenge. Remote working did present some challenges, mainly communication and collaboration as it was difficult to get help and to work together during sprints as team members all had different schedules. For example, when integrating another team member's code it was sometimes difficult to communicate and get assistance. Members could be away from home and be unable to work simultaneously when not at home or when doing assignments, this meant that when an issue arose it could take many hours or days until it was able to be fixed.

Compare this to an in-person environment where team members are all working and available, if an issue arises or if a teammate needs assistance, it can be addressed quickly and in person rather than over text. This is why we left a lot of the bug fixing to the weekly meetings as we could sit down with each other and go through and fix the issue together.

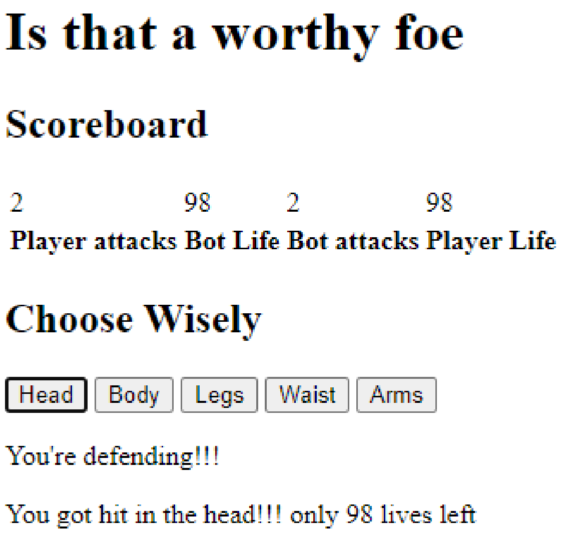
There are some advantages to remote working. It offers much more flexibility as team members can choose when and where they work, this is beneficial for people with non-traditional work hours and allowed us to do most of our work late in the evening for example if we needed to. Working remotely also meant that team members wouldn't have to commute onto campus to work on the project and could do it from their own home. Many of our team members also feel that they have increased productivity when working remotely as they can choose to work on their own time freely. Also working from home allowed us to rely on more reliable internet communications and allowed many of us to work on our own personal desktops and set-ups as opposed to laptops and college computers.

The flexibility and freedom of remote working has both advantages and disadvantages, it relies on good communication and planning by the team to remain effective.

Chapter 4: My contribution

From the very beginning, I felt responsible to develop the core game and its mechanics from scratch because it was my idea for the project and I knew better than anyone how the game should work and look. During our first SCRUM meeting, I let everybody in the team know that I will write code for the base of the game, and then anyone could join in and add new functionality and improve the game visuals. While I do that, we collaborated in our SCRUM meeting and assigned what each person will do. Initially, I suggested that three people will work on the various menus, settings, and Firebase, and three people work on the core game and its mechanics.

I began the project by laying out the core mechanics of the game. I took some parts of the code from “rock, paper, scissors” game from (1)https://codepen.io/nevan/pen/kVBmom as a base for CombatClicker. I have changed around the names, buttons and some functionality to match the CombatClicker gameplay. The following image is how the first version of the game looked and worked [fig 2]:


Is that a worthy foe

Scoreboard

```
2                98        2             98
Player attacks Bot Life Bot attacks Player Life
```

Choose Wisely

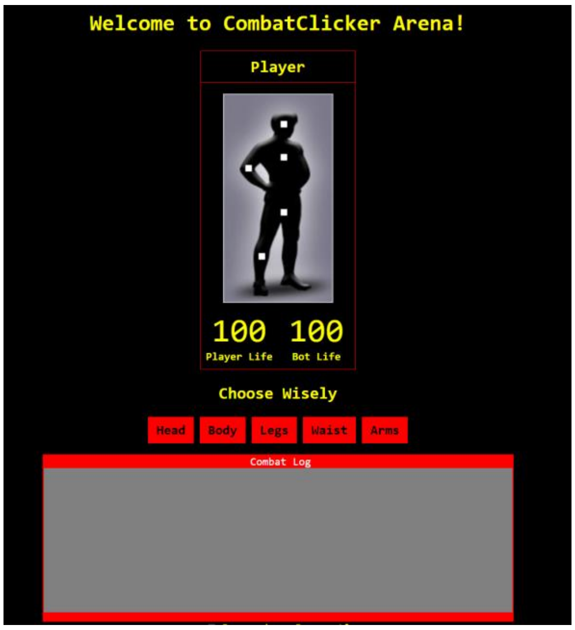
You're defending!!!

You got hit in the head!!! only 98 lives left

First version of CombatClicker [fig 2].

The above base functionality was the seed for building up the game’s mechanics in Javascript.

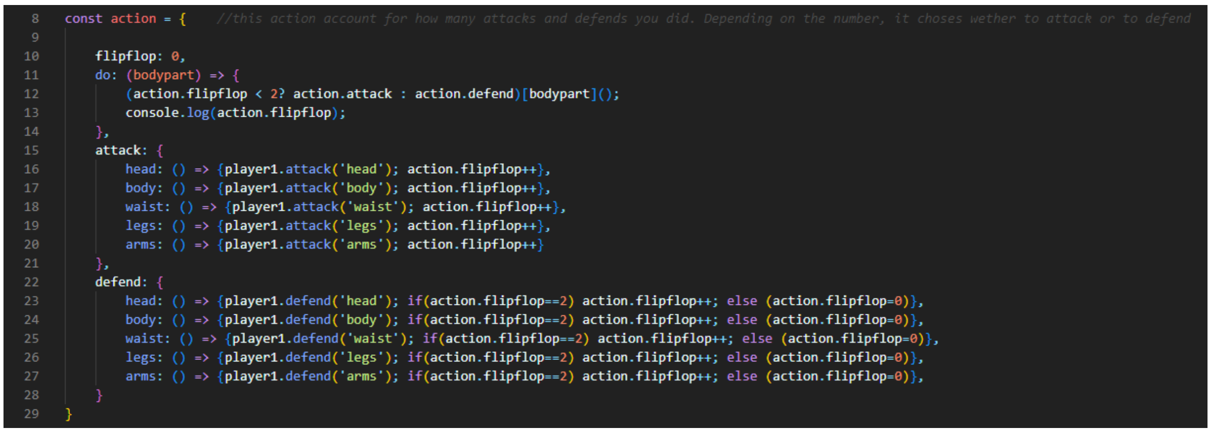
For the second sprint, I decided to work on CSS code, so that our group can have somewhat of a visual representation of the game, instead of just having a couple of functional buttons and text. I used the first version of CombatClicker from fig 2, and played around with different styles for the buttons and text to find the style that would fit. I added a (2)silhouette of a person as our character and placed it into a separate div box. After that, I placed the (3) checkboxes on top of the image, to represent attack/defend positions. And finally, I’ve added a combat log at the bottom which would keep track of all actions happened during combat. The following screenshot is the second version of the game after sprint 2:



Second version of CombatClicker [fig 2.1].

Even though the game was starting to look like something, there was absolutely no functionality behind it (except the five red buttons) at that moment, it was pure HTML code. The next challenge for me was to move the functionality from the five red buttons to the five checkboxes I created.

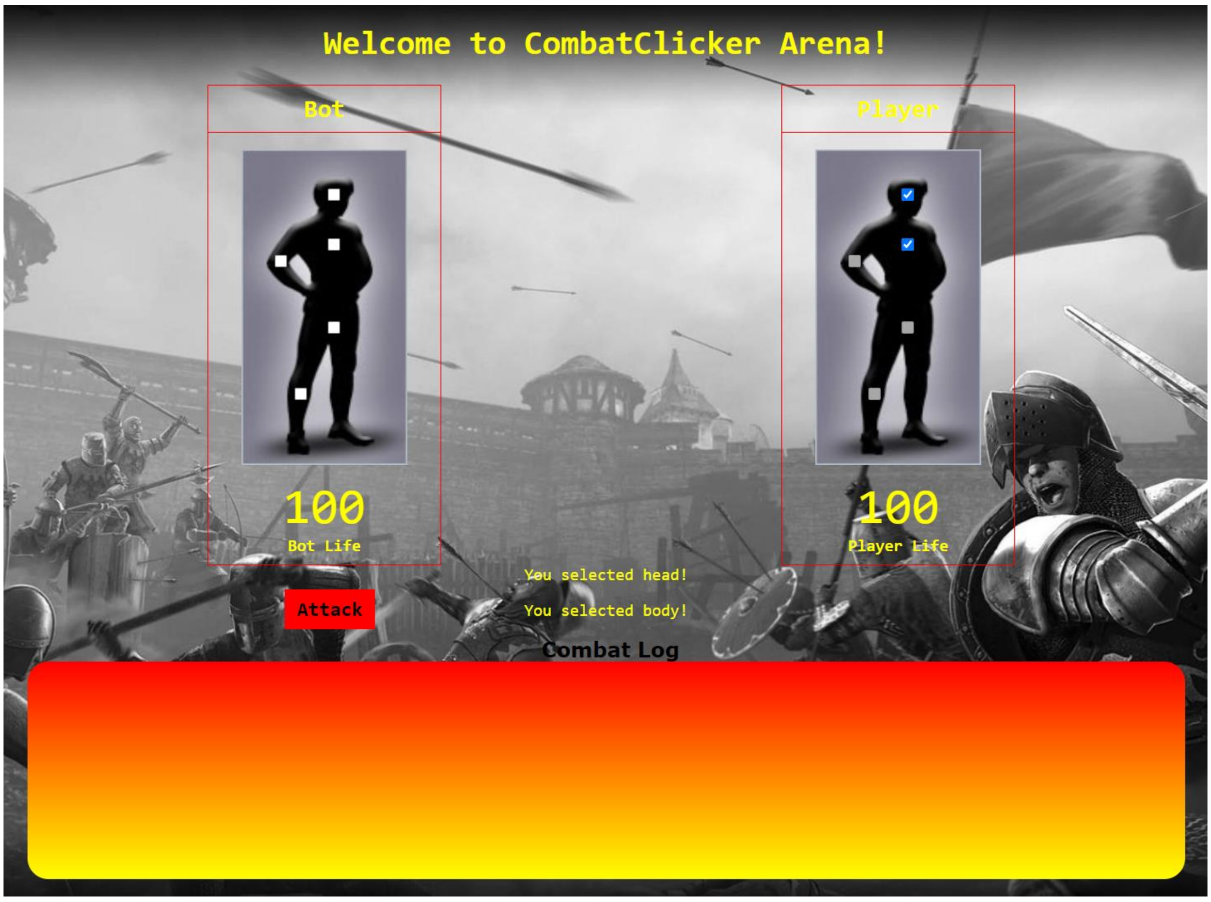
For the next couple of sprints, there was a lot of work done for the game on my side. I have added a (4)background image of a medieval combat. The second character on screen was just a pure copy of the first character <div> in HTML. The five red buttons from fig 2.1 were gone, and all of its functionality now worked with the checkboxes. The way I wrote code for checkboxes, was that you're only allowed to check two at a time, and the others would become unclickable. After the checkboxes were selected, a red button "Attack" would appear, and the player would then be able to submit the positions he selected to attack. After the "Attack" button was pressed, all of the functionality was then moved onto the other character on screen, and now the player would have to defend. This was achieved with a flipflop function that I created, which would read the game and flip between Attack and Defend states, as well as activating checkbox functionality depending on the state.



```
const action = {    //this action account for how many attacks and defends you did. Depending on the number, it choses wether to attack or to defend

    flipflop: 0,
    do: (bodypart) => {
        (action.flipflop < 2? action.attack : action.defend)[bodypart]();
        console.log(action.flipflop);
    },
    attack: {
        head: () => {player1.attack('head'); action.flipflop++},
        body: () => {player1.attack('body'); action.flipflop++},
        waist: () => {player1.attack('waist'); action.flipflop++},
        legs: () => {player1.attack('legs'); action.flipflop++},
        arms: () => {player1.attack('arms'); action.flipflop++}
    },
    defend: {
        head: () => {player1.defend('head'); if(action.flipflop==2) action.flipflop++; else (action.flipflop=0)},
        body: () => {player1.defend('body'); if(action.flipflop==2) action.flipflop++; else (action.flipflop=0)},
        waist: () => {player1.defend('waist'); if(action.flipflop==2) action.flipflop++; else (action.flipflop=0)},
        legs: () => {player1.defend('legs'); if(action.flipflop==2) action.flipflop++; else (action.flipflop=0)},
        arms: () => {player1.defend('arms'); if(action.flipflop==2) action.flipflop++; else (action.flipflop=0)},
    }
}
```

Flipflop function for switching between attack/defend CombatClicker [fig 2.2].

There were also onclick events for each of the checkboxes. Whenever a checkbox was selected, it would trigger text that would appear on-screen, indicating the player what position he or she has selected. After the positions have been selected and submitted, the computer would randomly select 2 out of 5 positions to defend. If the computer selected the wrong spots to defend, he would take damage ranging between 1 and 25. Flipflop function is then triggered, and now the player has to select 2 out of 5 positions again, but this time the player is defending, and the computer is randomly selecting 2 positions to attack. All of this functionality would repeat until either the player or computer reached 0 life.

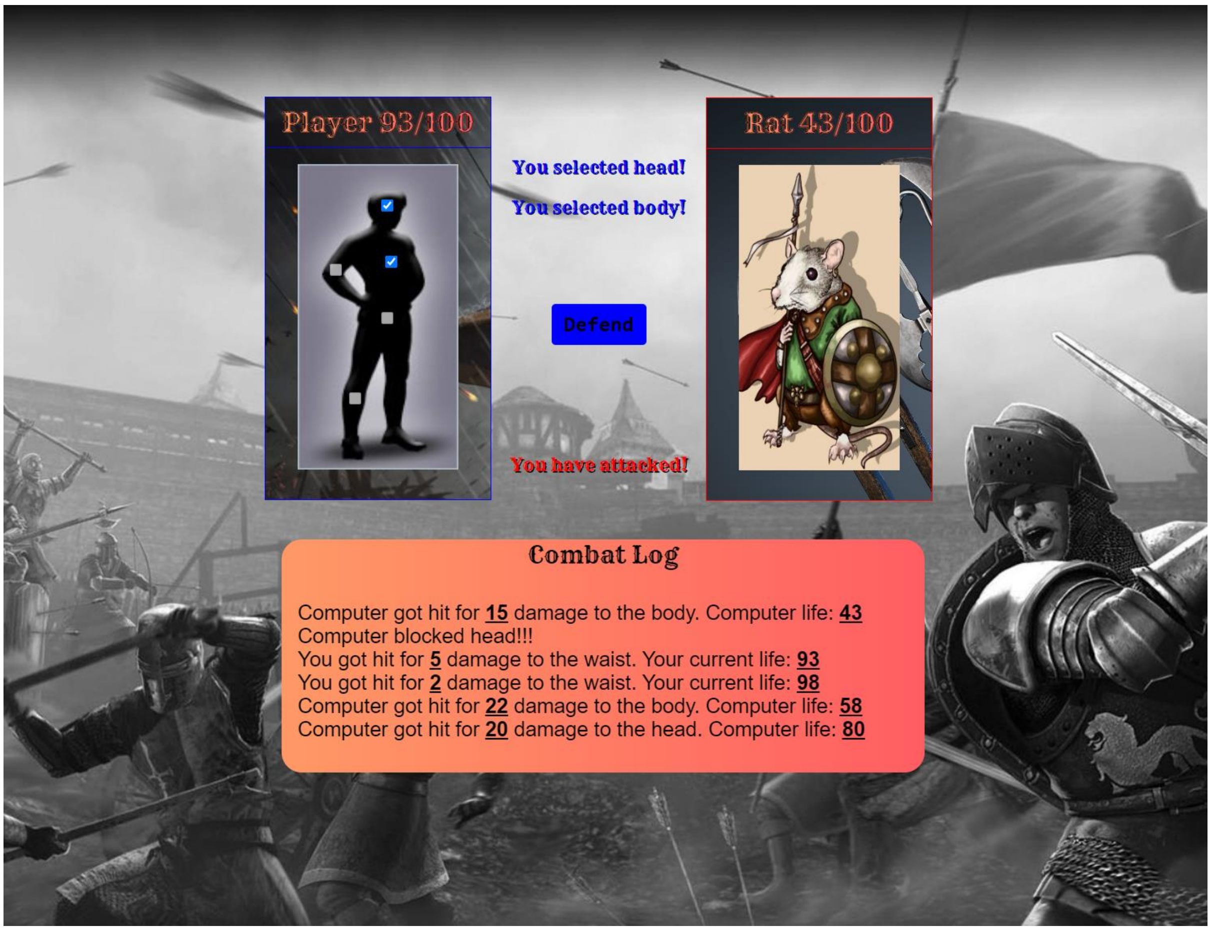


Third version of CombatClicker [fig 2.3].

The above image shows how the game looked like at this point. It was starting to get interesting, however, there was still a lot of missing functionality and work to be done.

For the next couple of sprints, I worked on creating an opponent and implement a fully functional combat from start to finish. I decided to add a (5)Rat Warrior as my computer opponent, just to add some fantasy flavour to the game. Player and computer health points were moved on top of characters, and now the damage dealt/received would instantly update above characters. Added images behind players, just to make the characters stand out and be more visually appealing. The player boarders would now switch between red and blue, red boarder means the character is attacking, blue boarder means the character is defending. The colours would switch after each round using the flipper function from fig 2.2. These colours were also implemented for the appearing text and buttons. After the end user selects the checkboxes and submits, the checkboxes would now disappear from one character and appear on the other.

During these few sprints, I also worked on CSS code, changing the font of all visible text, adding more colours and gradients, and changing the display of the Combat Log, as well as implementing a scrolling function for it. The text in Combat Log would now appear at the very top, and scroll down if too many lines would appear. I also emboldened the damage numbers in combat log, for visual clarity.

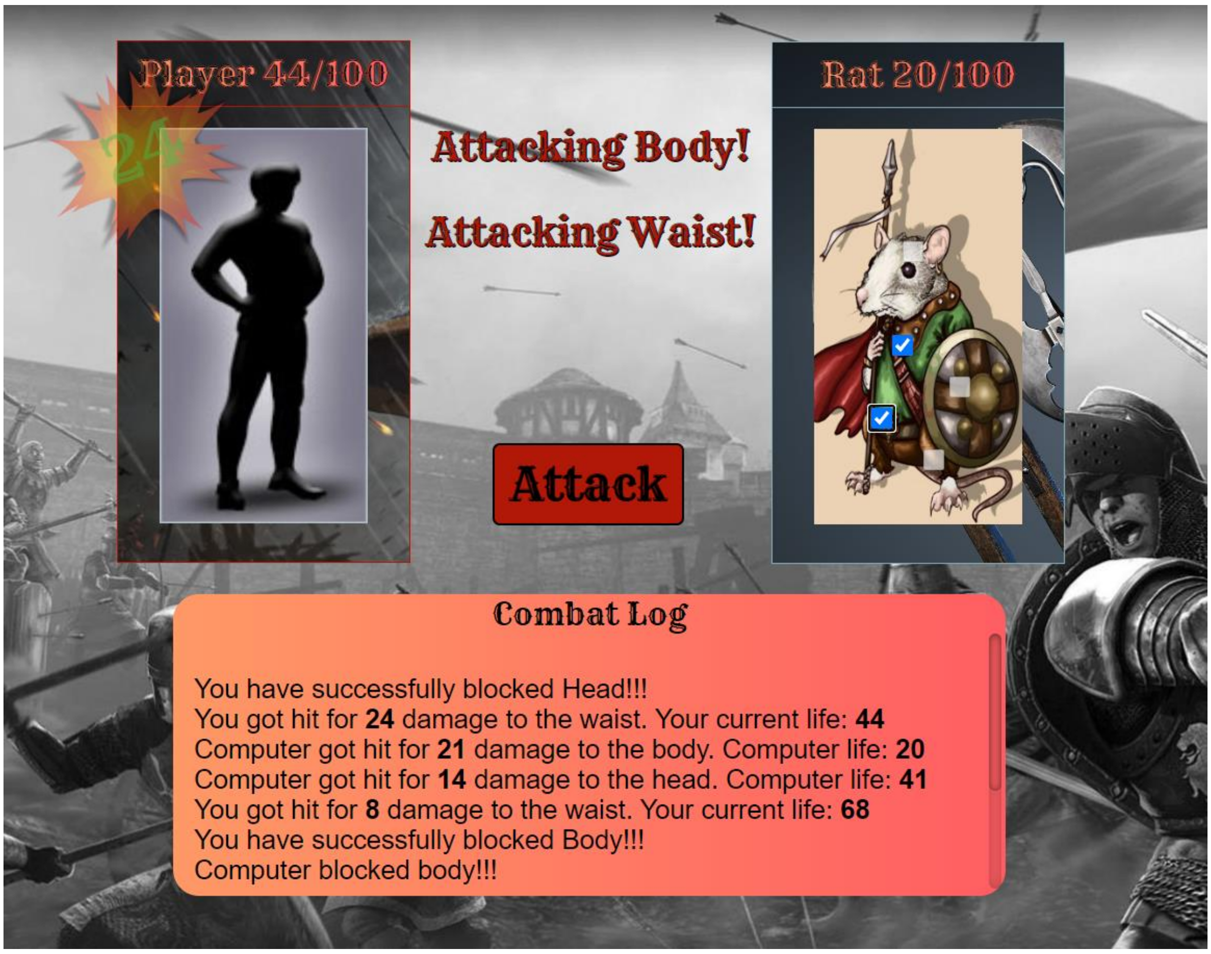


Computer got hit for **15** damage to the body. Computer life: **43**
Computer blocked head!!!
You got hit for **5** damage to the waist. Your current life: **93**
You got hit for **2** damage to the waist. Your current life: **98**
Computer got hit for **22** damage to the body. Computer life: **58**
Computer got hit for **20** damage to the head. Computer life: **80**

Fourth version of CombatClicker [fig 2.4].

Figure 2.4 above shows the fourth version of the game. The game was now playable from start to finish. I asked some students from other groups to try out the game and play it from an end user perspective. It was very enjoyable to see other people play the game that I have created. Letting other people to play the game was also very useful, as they would spot some small bugs or inconsistencies that I would just not notice. In conclusion, I learned that End-to-end tests are very useful for spotting bugs and making high quality games.

During the final sprint, I was working improving user interface and user experience. I made the Attack/Defend button to stand out more, and styled the pop-up text to be visually pleasing. The biggest changes that I had to make during the final sprint, were to create various divs to split the components, and set the position of these components to either absolute or relative. These changes allowed the game to be playable on different screen resolutions and avoid overlaps. The following image is how the final product has turned out.

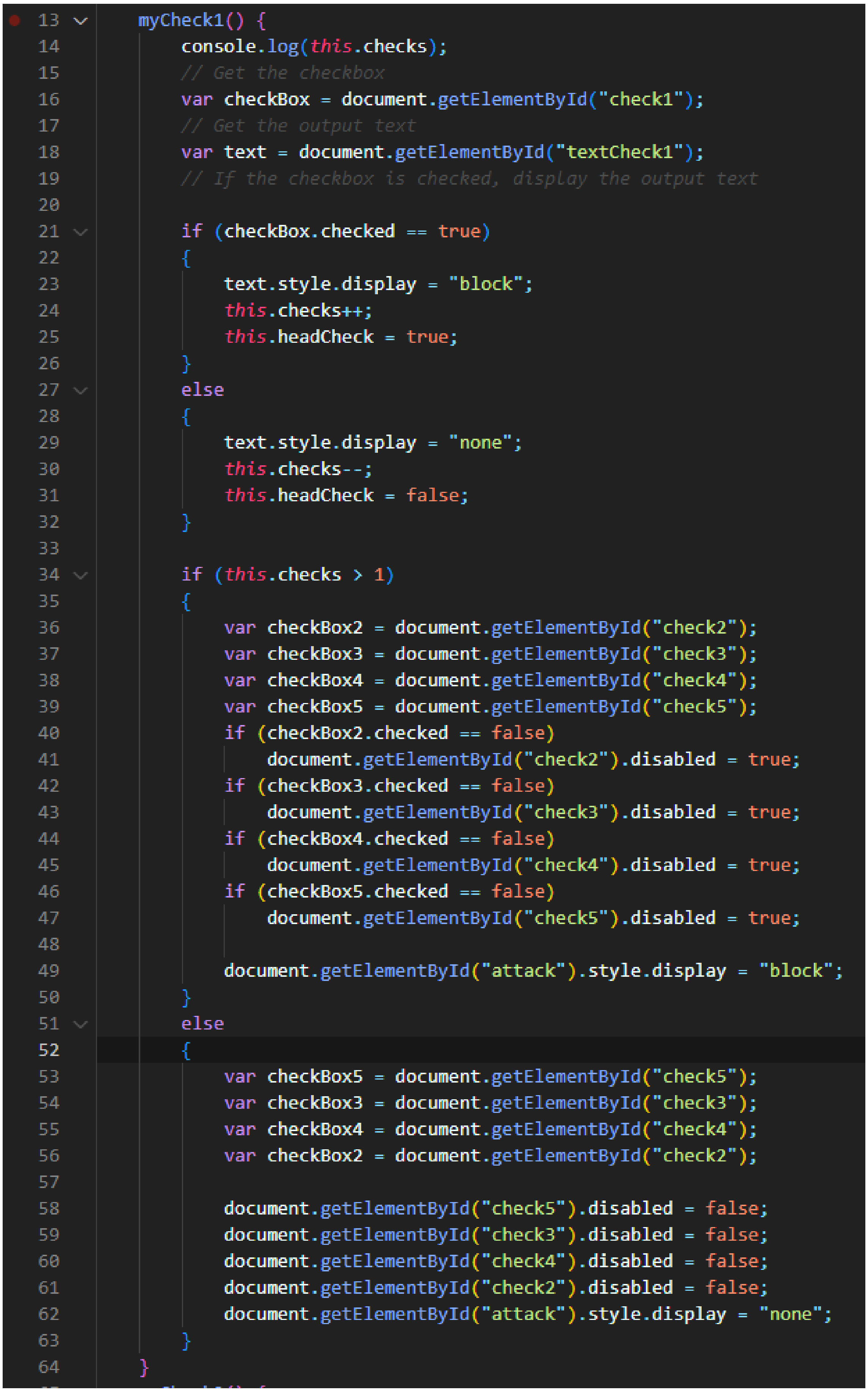


You have successfully blocked Head!!!
You got hit for **24** damage to the waist. Your current life: **44**
Computer got hit for **21** damage to the body. Computer life: **20**
Computer got hit for **14** damage to the head. Computer life: **41**
You got hit for **8** damage to the waist. Your current life: **68**
You have successfully blocked Body!!!
Computer blocked body!!!

Fifth and final version of CombatClicker [fig 2.5].

Observe, in the above image, the damage number in the top left corner. This was a (6)"Bang" effect that I put into the game to improve user experience. The damage from two attacks/defends would get calculated and display on top of a "Bang" icon with a fade out effect. The main game has now been finished, and I am very proud and satisfied with the end result.

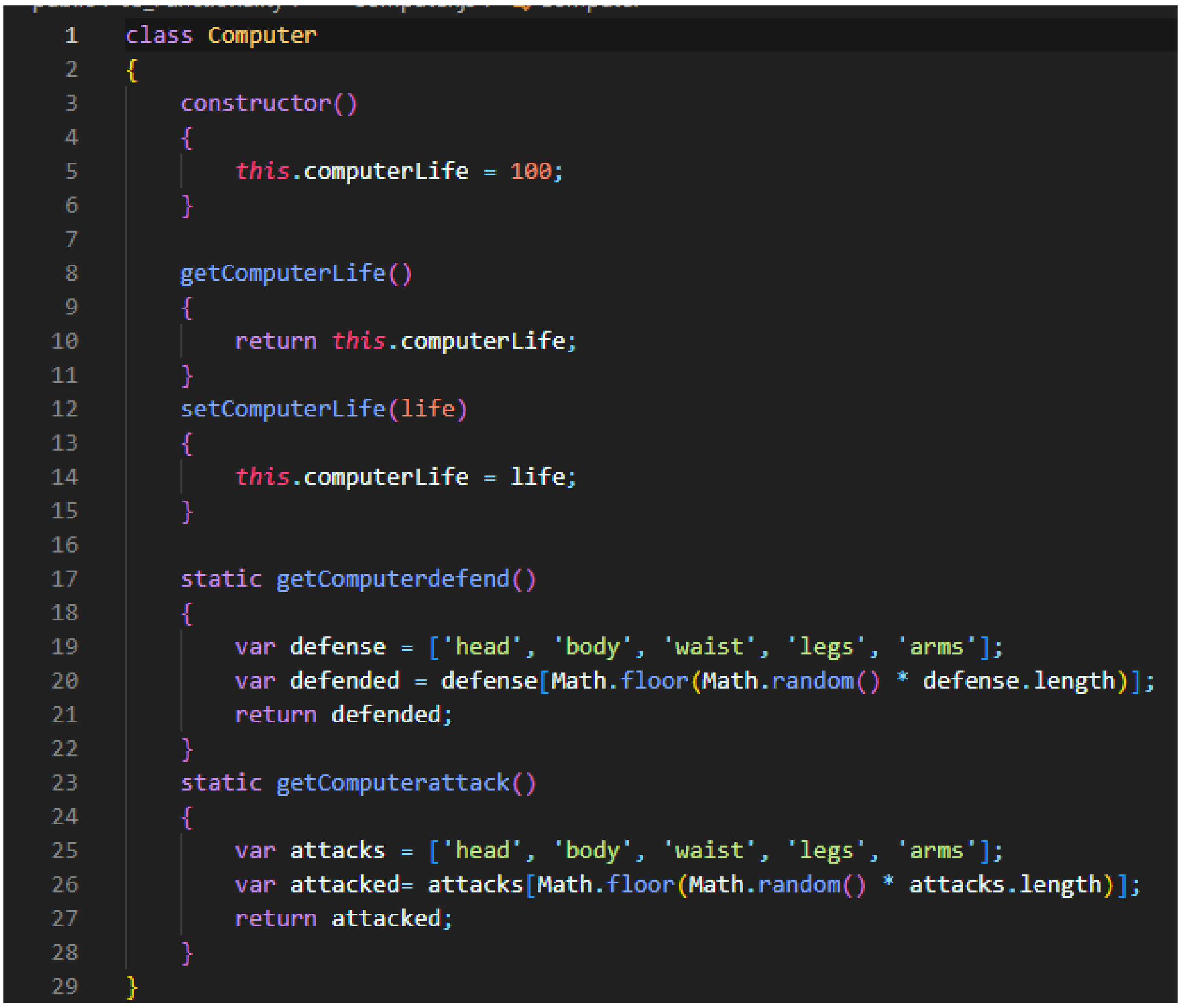
Over a thousand of lines of code have been written by me, between JavaScript, HTML and CSS. The below images are some of the notable pieces of code written by me that were particularly challenging.



```
    myCheck1() {
        console.log(this.checks);
        // Get the checkbox
        var checkBox = document.getElementById("check1");
        // Get the output text
        var text = document.getElementById("textCheck1");
        // If the checkbox is checked, display the output text

        if (checkBox.checked == true)
        {
            text.style.display = "block";
            this.checks++;
            this.headCheck = true;
        }
        else
        {
            text.style.display = "none";
            this.checks--;
            this.headCheck = false;
        }

        if (this.checks > 1)
        {
            var checkBox2 = document.getElementById("check2");
            var checkBox3 = document.getElementById("check3");
            var checkBox4 = document.getElementById("check4");
            var checkBox5 = document.getElementById("check5");
            if (checkBox2.checked == false)
                document.getElementById("check2").disabled = true;
            if (checkBox3.checked == false)
                document.getElementById("check3").disabled = true;
            if (checkBox4.checked == false)
                document.getElementById("check4").disabled = true;
            if (checkBox5.checked == false)
                document.getElementById("check5").disabled = true;

            document.getElementById("attack").style.display = "block";
        }
        else
        {
            var checkBox5 = document.getElementById("check5");
            var checkBox3 = document.getElementById("check3");
            var checkBox4 = document.getElementById("check4");
            var checkBox2 = document.getElementById("check2");

            document.getElementById("check5").disabled = false;
            document.getElementById("check3").disabled = false;
            document.getElementById("check4").disabled = false;
            document.getElementById("check2").disabled = false;
            document.getElementById("attack").style.display = "none";
        }
    }
```

JavaScript code for checkbox functionality [fig 3].

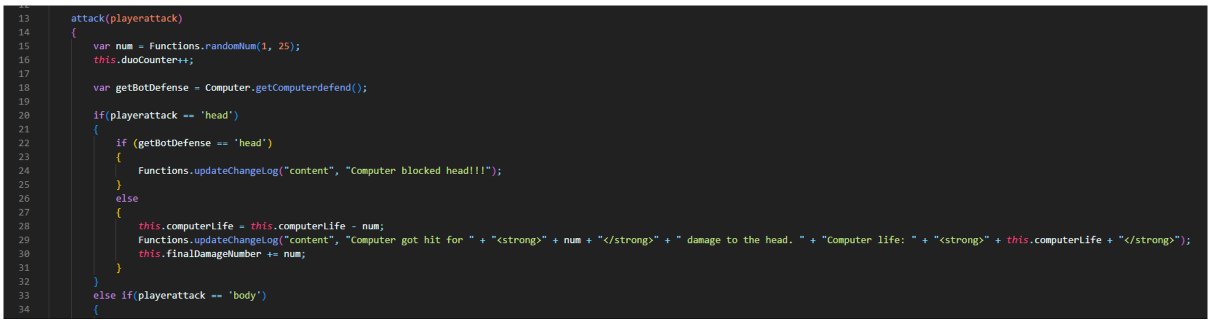
The above image is how the functionality for checkboxes was written. There were five of these functions, one for each checkbox. These functions kept track of which checkboxes are selected, which ones have to be disabled and what text must be displayed.



```
class Computer
{
    constructor()
    {
        this.computerLife = 100;
    }

    getComputerLife()
    {
        return this.computerLife;
    }
    setComputerLife(life)
    {
        this.computerLife = life;
    }

    static getComputerdefend()
    {
        var defense = ['head', 'body', 'waist', 'legs', 'arms'];
        var defended = defense[Math.floor(Math.random() * defense.length)];
        return defended;
    }
    static getComputerattack()
    {
        var attacks = ['head', 'body', 'waist', 'legs', 'arms'];
        var attacked= attacks[Math.floor(Math.random() * attacks.length)];
        return attacked;
    }
}
```

Computer class, used as an object in other classes [fig 3.1].

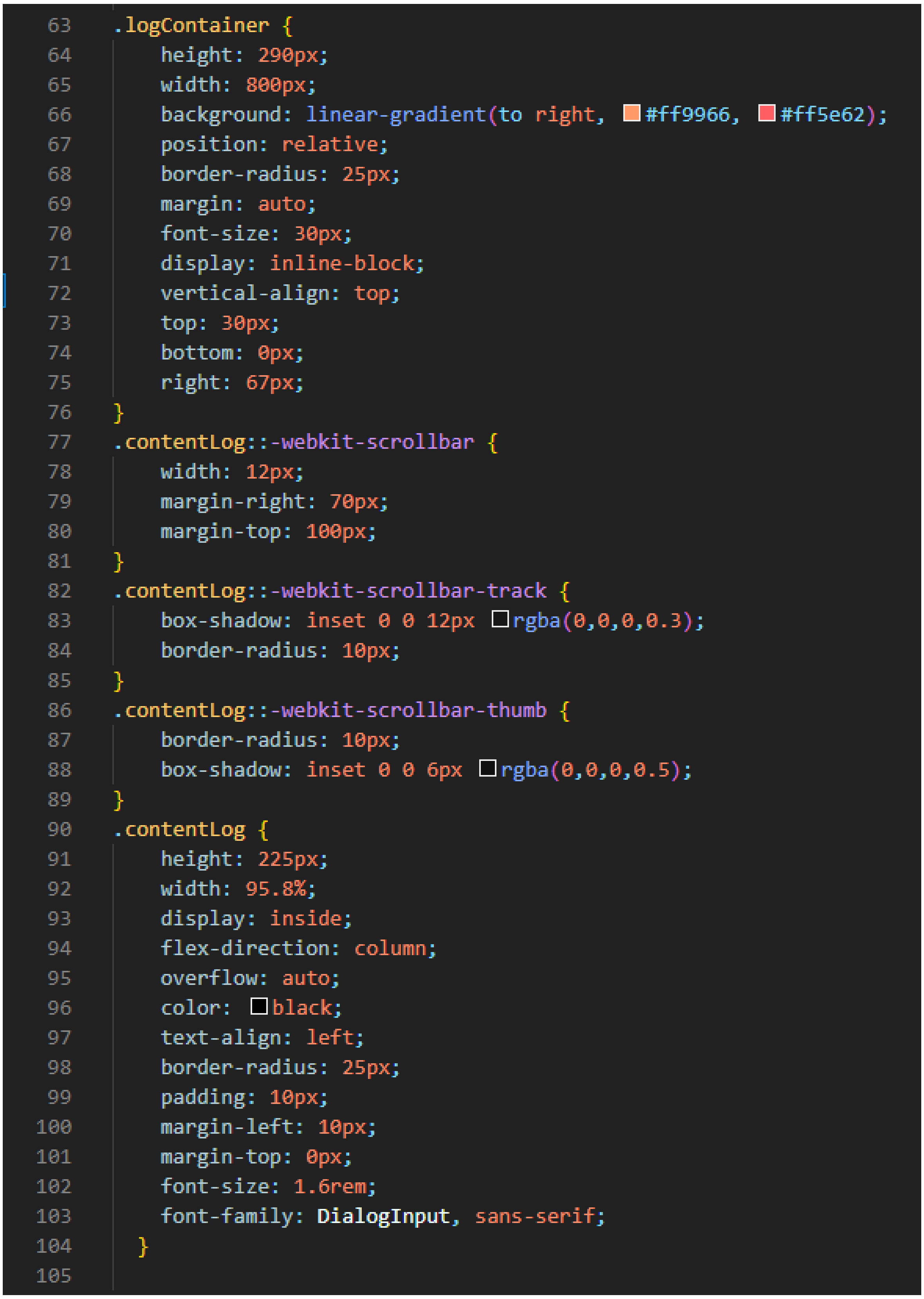
Figure 3.1 shows an example of object-oriented programming usage. There were more classes like these, which were used to create objects and then passed into different functions and classes.



```
    attack(playerattack)
    {
        var num = Functions.randomNum(1, 25);
        this.duoCounter++;

        var getBotDefense = Computer.getComputerdefend();

        if(playerattack == 'head')
        {
            if (getBotDefense == 'head')
            {
                Functions.updateChangeLog("content", "Computer blocked head!!!");
            }
            else
            {
                this.computerLife = this.computerLife - num;
                Functions.updateChangeLog("content", "Computer got hit for " + "<strong>" + num + "</strong>" + " damage to the head. " + "Computer life: " + "<strong>" + this.computerLife + "</strong>");
                this.finalDamageNumber += num;
            }
        }
        else if(playerattack == 'body')
        {
```

Main functionality of attacking the computer [fig 3.2].

Figure 3.2 shows an example of how the functionality worked for when a player attacks the computer. If a player selects head, the computer then generates a random position to defend. If the player attacks the same position as the computer defends, a block state would be triggered and a string would be passed into the combat log.



```css
.logContainer {
    height: 290px;
    width: 800px;
    background: linear-gradient(to right, #ff9966, #ff5e62);
    position: relative;
    border-radius: 25px;
    margin: auto;
    font-size: 30px;
    display: inline-block;
    vertical-align: top;
    top: 30px;
    bottom: 0px;
    right: 67px;
}
.contentLog::-webkit-scrollbar {
    width: 12px;
    margin-right: 70px;
    margin-top: 100px;
}
.contentLog::-webkit-scrollbar-track {
    box-shadow: inset 0 0 12px rgba(0,0,0,0.3);
    border-radius: 10px;
}
.contentLog::-webkit-scrollbar-thumb {
    border-radius: 10px;
    box-shadow: inset 0 0 6px rgba(0,0,0,0.5);
}
.contentLog {
    height: 225px;
    width: 95.8%;
    display: inside;
    flex-direction: column;
    overflow: auto;
    color: black;
    text-align: left;
    border-radius: 25px;
    padding: 10px;
    margin-left: 10px;
    margin-top: 0px;
    font-size: 1.6rem;
    font-family: DialogInput, sans-serif;
  }

```

CSS code for the combat log [fig 3.3].

The above image is just an example of how the styling and visual content was created in CombatClicker. To achieve the perfect style that fit into the game, a lot of trial and error has been done for each of the components.

Chapter 5: Summary

This whole project was quite an enjoyable experience. This was the first time I have worked on an application with a group of people. I gained a lot of insight on the various processes and components that are required to build a successful project. There were a lot of lessons learned, weather it was from making mistakes or creating something that worked well.

Everybody in our group had different set of skills to work with. Some people were quite good at front end development and designing user interface and experience, others were strong in database management and researching. For myself, I have discovered that I am able to adapt to any work environment that the project needs. Previously, I would only write code in Java and complete various challenges on website such as Leetcode and Hackerrank, and I was not very familiar with Javascript, HTML and CSS. This project changed my view and perspective on other sides of coding. Personally, it was not difficult to switch from Java to Javascript, a lot of fundamentals carried over and I was able to get comfortable with new programming language quickly.

During the development of CombatClicker, I discovered a Javascript library called React. React was very interesting to me because after using it for a bit, I realised that it would be incredibly beneficial to our project and it would make the development process a lot more smooth. React would allow as to replace certain components of our user interface effortlessly, and it would not destroy the structure of our previously written classes and functions. Unfortunately, our team decided to not switch to ReactJS, because not everybody in a team wanted to spend time and learn another library and re-develop what we already had done. It was too late to switch our development to React, as we already had hundreds of lines of code that worked perfectly. What I learned from this, is that at the beginning of each project, we, as a team should discuss better which frameworks and libraries to use so that everybody is on the same page during the development. Myself, and a couple of other people heard about React from other projects and CS353 module documentation, but we did not realise how beneficial it is and how well it would work for the project that we all worked with. Perhaps in the future, I will find some time to redevelop or create a new game based on ReactJS, this time with a lot more experience.

A huge part of the project was the use of GitLab. Gitlab is an open source code repository and software development platform for DevOps projects. It allowed our team to store the code online for us to track and use update. This was a totally new experience to me as I was previously not familiar with GitLab and how it works. I had a lot of issues with it in the beginning as I did not understand how to connect my VSCode compiler with GitLab and commit my code into a branch. After many failures, I was able to upload my code into our project repository, and start tracking all the processes and game patches. I was pleased to see other group members using GitLab, and seeing new commits and branches appearing in our work environment. It was very intuitive to use and download the work of my team members or downloading the most recent version of our game to continue the development. Our team had everything sorted neatly, we had a section for Core game mechanics, Menus, Database contents, Game sounds and many more. I noticed that not all team members were using GitLab. I talked to these people and turns out they were having similar issues as me in the beginning, and they just preferred to upload their work into a drop box or similar service. Our team should have communicated this better and I wish some of our team members could come up to me and ask me questions on how to use GitLab, I would be more than happy to help them. I believe the work we did through GitLab will be very beneficial for my future software engineering career and I'm glad that this module provided us unlimited access to GitLab and allowed us to get familiar with it before we start our internships in 2023.

Another huge problem that our team encountered during the development process was that we did not write enough comments for the code that we would write, both in the main game and menus. There were many times where a team member would deploy a new build to GitLab, I would then download the build and would not be able to find what exactly was changed since the last build. This was a huge problem and it slowed down our development process. We did discuss this issue with the team in a couple of our SCRUM meeting and we started to add more comments at that time. Sometimes, I wouldn't even recognise my own code because I forgot to write a comment for what a certain function would do. I learned from my mistakes and I will comment my code much more often for future projects.

Overall I'm quite happy with how the end product has turned out, and I am very proud of all the work and my contributions to the project. I am extremely interested in continuing working on this game even after the project is submitted and the module has ended. I am planning to implement an adventure mode for the game and add various enemies and player progression for CombatClicker. It would be very interesting to implement multiplayer to this game and host it online, so other people could try it out and play it from their homes. I have many more ideas for this project and I am looking forward to converting these ideas into a fully functional online game. Perhaps, one day you will be able to find "CombatClicker" on the app store and try it out yourself.

References

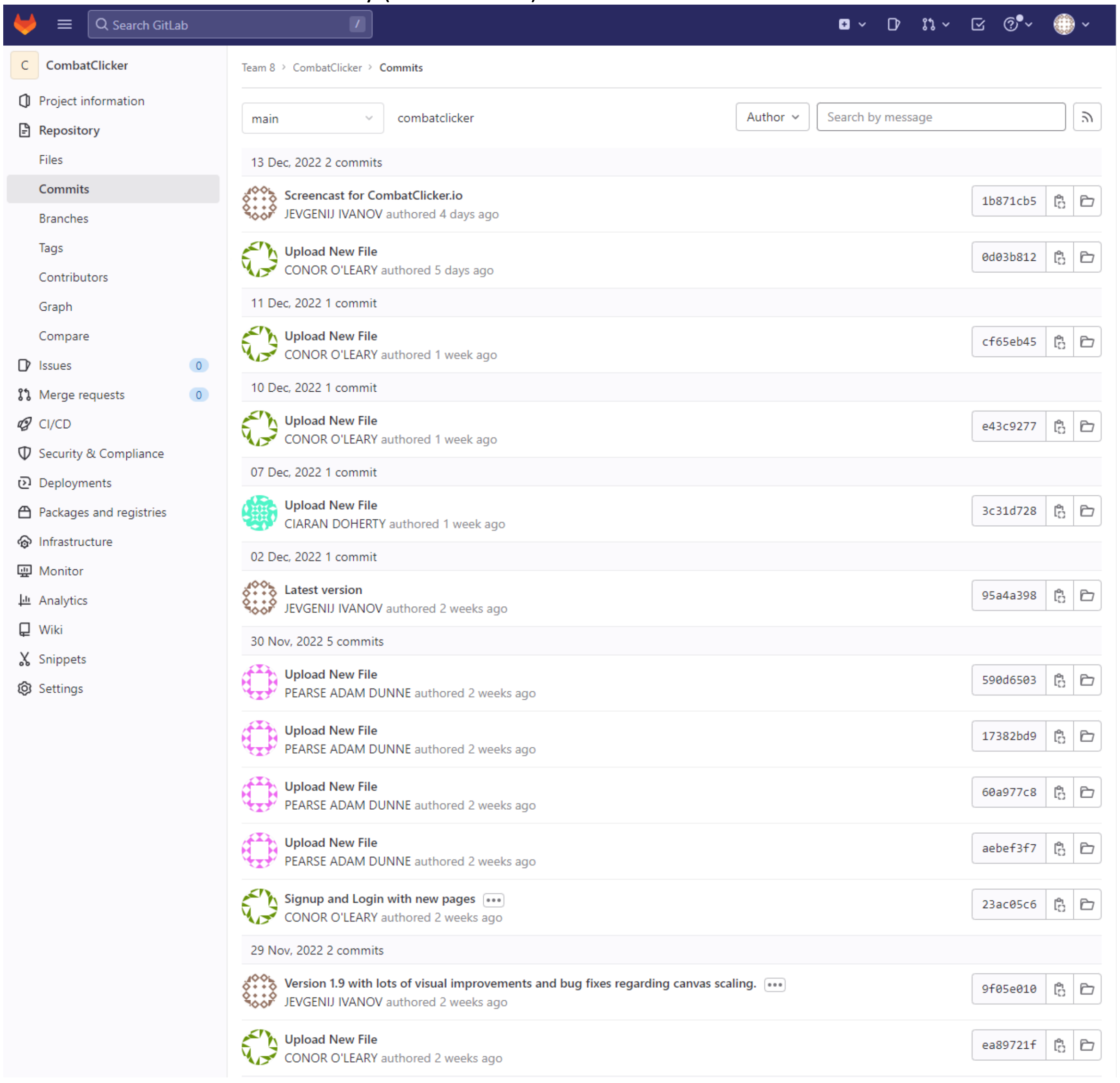
(1) Nevan Scott, https://codepen.io/nevan/pen/kVBmom
(2) Image for silhouette of a character, taken from website https://carnage.ru
(3) HTML checkboxes https://www.w3schools.com/tags/att_input_type_checkbox.asp
(4) Background image of medieval combat https://wallpapercave.com/medieval-battle-wallpaper
(5) Image of a rat warrior https://shipoffools.fandom.com/wiki/The_Great_Rat_Army
(6) "Bang" image for damage numbers http://kanrojyouhou.co.jp/bang-icon-k.html

Scrum/Agile information and images:
https://agilemanifesto.org/principles.html
https://scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf
https://blog.scrumstudy.com/blending-agile-frameworks-for-project-success/

Appendix

GitLab project commit history

Evidence of Gitlab commit history (Main branch):



13 Dec, 2022 2 commits

Screencast for CombatClicker.io
JEVGENIJ IVANOV authored 4 days ago — 1b871cb5

Upload New File
CONOR O'LEARY authored 5 days ago — 0d03b812

11 Dec, 2022 1 commit

Upload New File
CONOR O'LEARY authored 1 week ago — cf65eb45

10 Dec, 2022 1 commit

Upload New File
CONOR O'LEARY authored 1 week ago — e43c9277

07 Dec, 2022 1 commit

Upload New File
CIARAN DOHERTY authored 1 week ago — 3c31d728

02 Dec, 2022 1 commit

Latest version
JEVGENIJ IVANOV authored 2 weeks ago — 95a4a398

30 Nov, 2022 5 commits

Upload New File
PEARSE ADAM DUNNE authored 2 weeks ago — 590d6503

Upload New File
PEARSE ADAM DUNNE authored 2 weeks ago — 17382bd9

Upload New File
PEARSE ADAM DUNNE authored 2 weeks ago — 60a977c8

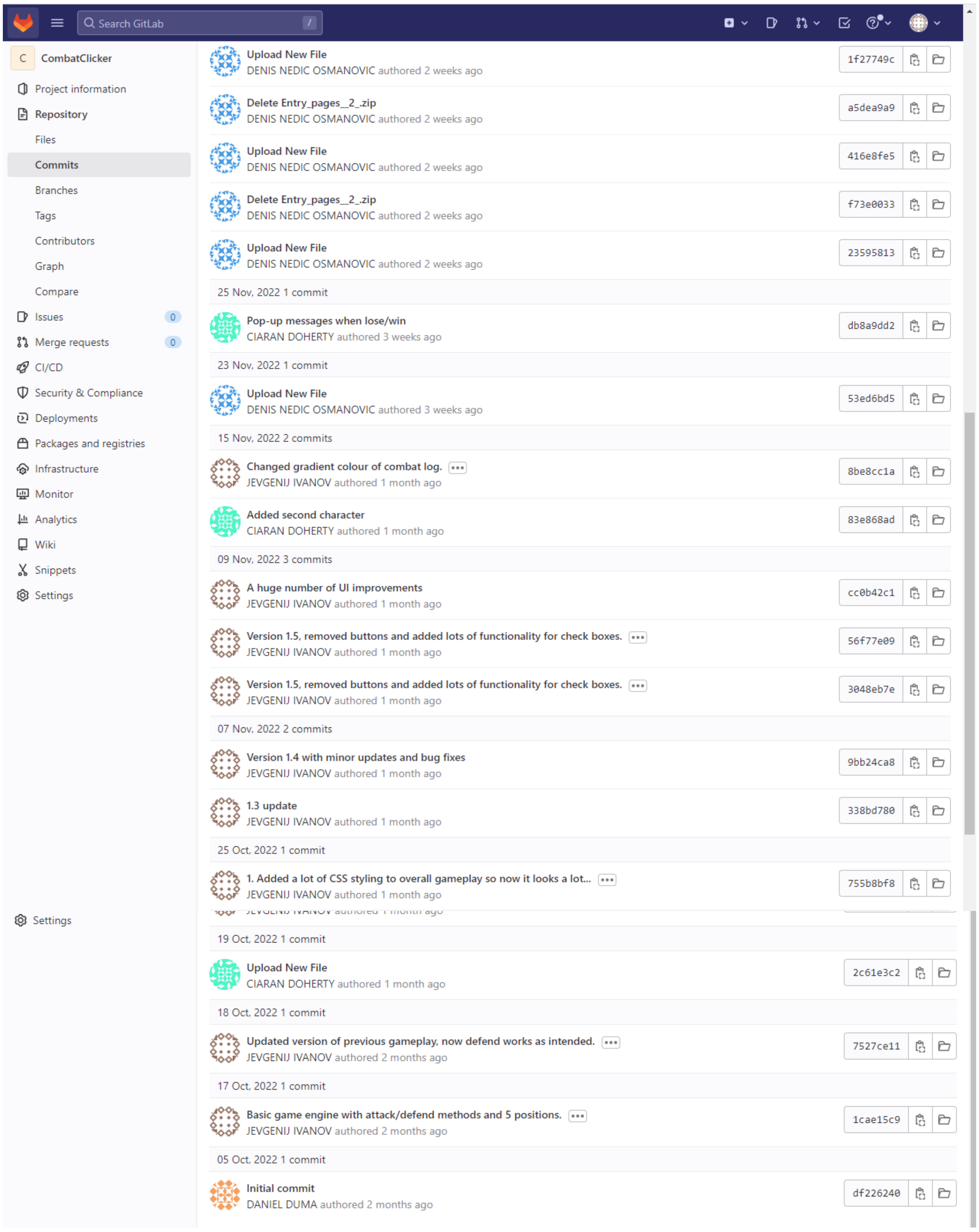
Upload New File
PEARSE ADAM DUNNE authored 2 weeks ago — aebef3f7

Signup and Login with new pages
CONOR O'LEARY authored 2 weeks ago — 23ac05c6

29 Nov, 2022 2 commits

Version 1.9 with lots of visual improvements and bug fixes regarding canvas scaling.
JEVGENIJ IVANOV authored 2 weeks ago — 9f05e010

Upload New File
CONOR O'LEARY authored 2 weeks ago — ea89721f



Search GitLab

C CombatClicker

Project information
Repository
Files
Commits
Branches
Tags
Contributors
Graph
Compare
Issues 0
Merge requests 0
CI/CD
Security & Compliance
Deployments
Packages and registries
Infrastructure
Monitor
Analytics
Wiki
Snippets
Settings

Settings

Upload New File
DENIS NEDIC OSMANOVIC authored 2 weeks ago — 1f27749c

Delete Entry_pages__2_.zip
DENIS NEDIC OSMANOVIC authored 2 weeks ago — a5dea9a9

Upload New File
DENIS NEDIC OSMANOVIC authored 2 weeks ago — 416e8fe5

Delete Entry_pages__2_.zip
DENIS NEDIC OSMANOVIC authored 2 weeks ago — f73e0033

Upload New File
DENIS NEDIC OSMANOVIC authored 2 weeks ago — 23595813

25 Nov, 2022 1 commit

Pop-up messages when lose/win
CIARAN DOHERTY authored 3 weeks ago — db8a9dd2

23 Nov, 2022 1 commit

Upload New File
DENIS NEDIC OSMANOVIC authored 3 weeks ago — 53ed6bd5

15 Nov, 2022 2 commits

Changed gradient colour of combat log. ...
JEVGENIJ IVANOV authored 1 month ago — 8be8cc1a

Added second character
CIARAN DOHERTY authored 1 month ago — 83e868ad

09 Nov, 2022 3 commits

A huge number of UI improvements
JEVGENIJ IVANOV authored 1 month ago — cc0b42c1

Version 1.5, removed buttons and added lots of functionality for check boxes. ...
JEVGENIJ IVANOV authored 1 month ago — 56f77e09

Version 1.5, removed buttons and added lots of functionality for check boxes. ...
JEVGENIJ IVANOV authored 1 month ago — 3048eb7e

07 Nov, 2022 2 commits

Version 1.4 with minor updates and bug fixes
JEVGENIJ IVANOV authored 1 month ago — 9bb24ca8

1.3 update
JEVGENIJ IVANOV authored 1 month ago — 338bd780

25 Oct, 2022 1 commit

1. Added a lot of CSS styling to overall gameplay so now it looks a lot... ...
JEVGENIJ IVANOV authored 1 month ago — 755b8bf8

JEVGENIJ IVANOV authored 1 month ago

19 Oct, 2022 1 commit

Upload New File
CIARAN DOHERTY authored 1 month ago — 2c61e3c2

18 Oct, 2022 1 commit

Updated version of previous gameplay, now defend works as intended. ...
JEVGENIJ IVANOV authored 2 months ago — 7527ce11

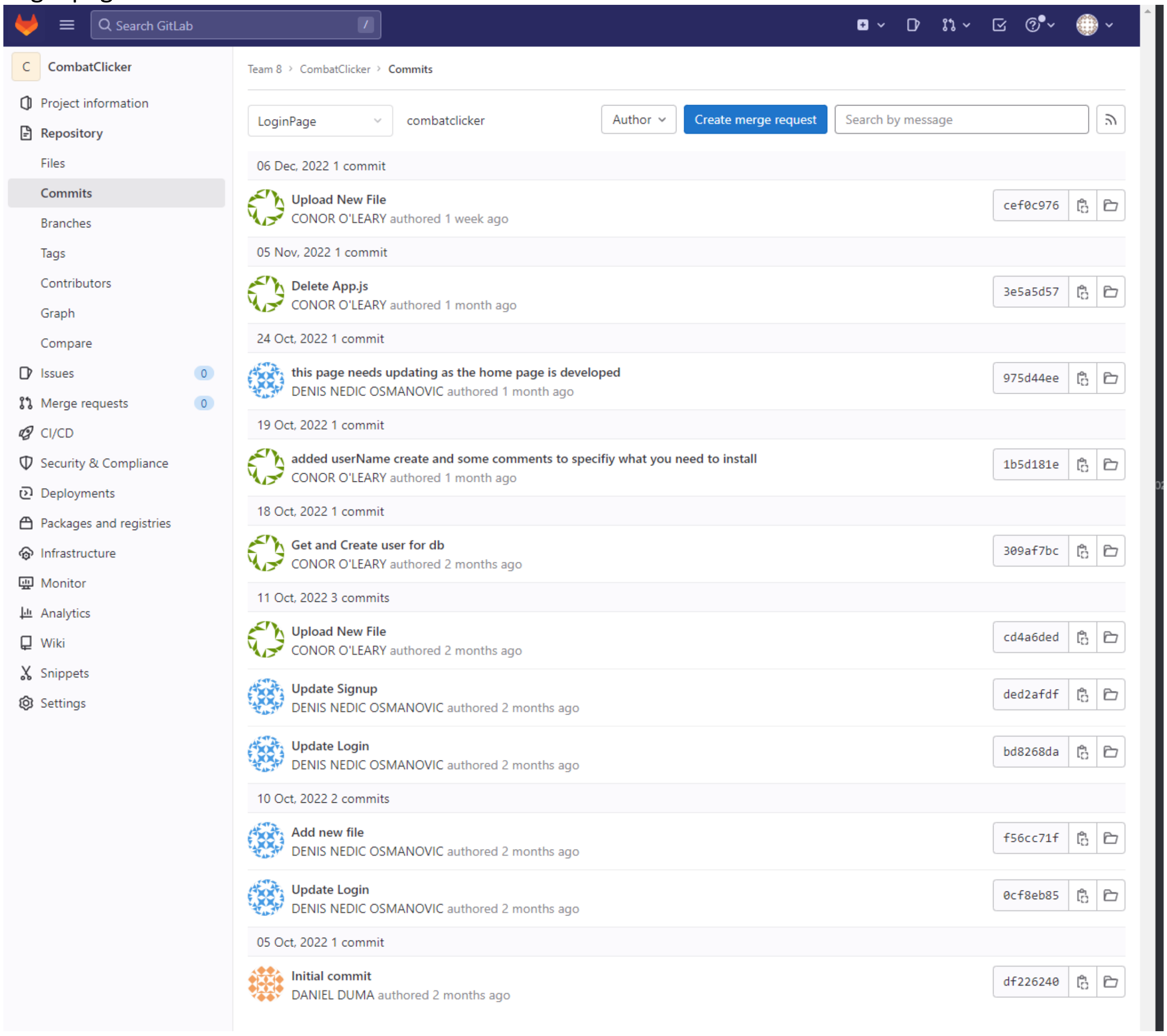
17 Oct, 2022 1 commit

Basic game engine with attack/defend methods and 5 positions. ...
JEVGENIJ IVANOV authored 2 months ago — 1cae15c9

05 Oct, 2022 1 commit

Initial commit
DANIEL DUMA authored 2 months ago — df226240

Login page branch:



Search GitLab

CombatClicker

Project information
Repository
Files
Commits
Branches
Tags
Contributors
Graph
Compare
Issues 0
Merge requests 0
CI/CD
Security & Compliance
Deployments
Packages and registries
Infrastructure
Monitor
Analytics
Wiki
Snippets
Settings

Team 8 › CombatClicker › Commits

LoginPage combatclicker Author Create merge request Search by message

06 Dec, 2022 1 commit

Upload New File
CONOR O'LEARY authored 1 week ago — cef0c976

05 Nov, 2022 1 commit

Delete App.js
CONOR O'LEARY authored 1 month ago — 3e5a5d57

24 Oct, 2022 1 commit

this page needs updating as the home page is developed
DENIS NEDIC OSMANOVIC authored 1 month ago — 975d44ee

19 Oct, 2022 1 commit

added userName create and some comments to specifiy what you need to install
CONOR O'LEARY authored 1 month ago — 1b5d181e

18 Oct, 2022 1 commit

Get and Create user for db
CONOR O'LEARY authored 2 months ago — 309af7bc

11 Oct, 2022 3 commits

Upload New File
CONOR O'LEARY authored 2 months ago — cd4a6ded

Update Signup
DENIS NEDIC OSMANOVIC authored 2 months ago — ded2afdf

Update Login
DENIS NEDIC OSMANOVIC authored 2 months ago — bd8268da

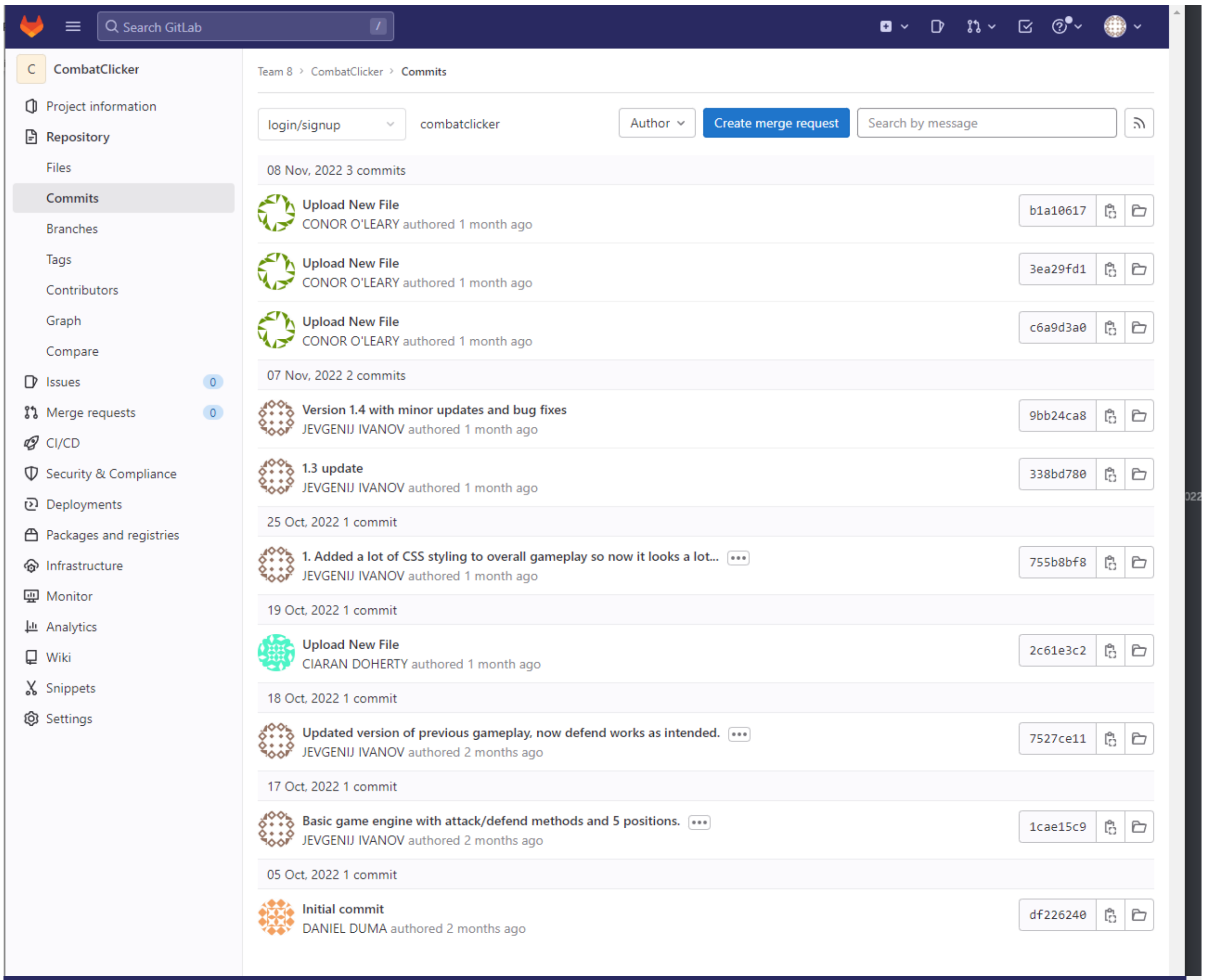
10 Oct, 2022 2 commits

Add new file
DENIS NEDIC OSMANOVIC authored 2 months ago — f56cc71f

Update Login
DENIS NEDIC OSMANOVIC authored 2 months ago — 0cf8eb85

05 Oct, 2022 1 commit

Initial commit
DANIEL DUMA authored 2 months ago — df226240


Team 8 › CombatClicker › Commits

login/signup | combatclicker | Author | Create merge request | Search by message

08 Nov, 2022 3 commits

Upload New File
CONOR O'LEARY authored 1 month ago — b1a10617

Upload New File
CONOR O'LEARY authored 1 month ago — 3ea29fd1

Upload New File
CONOR O'LEARY authored 1 month ago — c6a9d3a0

07 Nov, 2022 2 commits

Version 1.4 with minor updates and bug fixes
JEVGENIJ IVANOV authored 1 month ago — 9bb24ca8

1.3 update
JEVGENIJ IVANOV authored 1 month ago — 338bd780

25 Oct, 2022 1 commit

1. Added a lot of CSS styling to overall gameplay so now it looks a lot...
JEVGENIJ IVANOV authored 1 month ago — 755b8bf8

19 Oct, 2022 1 commit

Upload New File
CIARAN DOHERTY authored 1 month ago — 2c61e3c2

18 Oct, 2022 1 commit

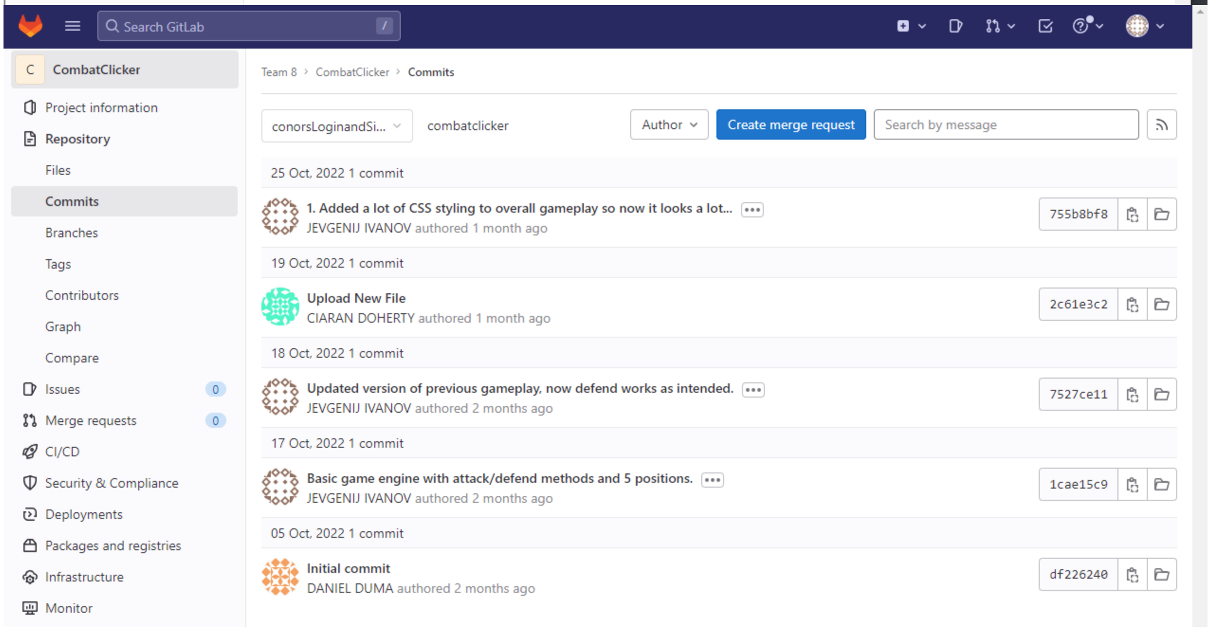
Updated version of previous gameplay, now defend works as intended.
JEVGENIJ IVANOV authored 2 months ago — 7527ce11

17 Oct, 2022 1 commit

Basic game engine with attack/defend methods and 5 positions.
JEVGENIJ IVANOV authored 2 months ago — 1cae15c9

05 Oct, 2022 1 commit

Initial commit
DANIEL DUMA authored 2 months ago — df226240


Team 8 › CombatClicker › Commits

conorsLoginandSi... | combatclicker | Author | Create merge request | Search by message

25 Oct, 2022 1 commit

1. Added a lot of CSS styling to overall gameplay so now it looks a lot...
JEVGENIJ IVANOV authored 1 month ago — 755b8bf8

19 Oct, 2022 1 commit

Upload New File
CIARAN DOHERTY authored 1 month ago — 2c61e3c2

18 Oct, 2022 1 commit

Updated version of previous gameplay, now defend works as intended.
JEVGENIJ IVANOV authored 2 months ago — 7527ce11

17 Oct, 2022 1 commit

Basic game engine with attack/defend methods and 5 positions.
JEVGENIJ IVANOV authored 2 months ago — 1cae15c9

05 Oct, 2022 1 commit

Initial commit
DANIEL DUMA authored 2 months ago — df226240