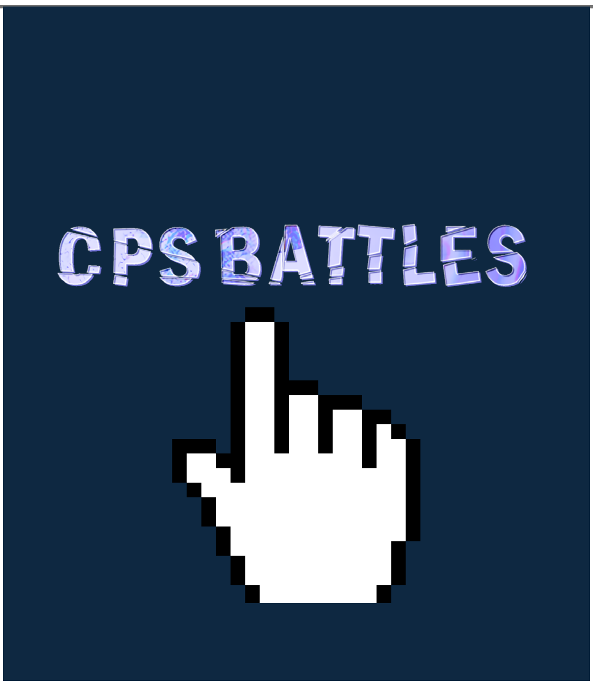

CPS Battles Project Report

Personal Project

Daniel Prizginas
Daniel.Prizginas@gmail.com
Github Profile
Github Repo

TABLE OF CONTENTS .... PAGE:

1 Introduction

CPS Battles is website where users can compete against each other to see who has the highest mouse click speed. Users can either directly compete in a one against one or compete indirectly by seeing who gets highest on the leaderboard. This website is a full stack project developed by me, Daniel Prizginas as a showcase of my skills as a second-year computer science university student.

The project uses the Django framework, thus the languages used are: HTML, CSS, JavaScript, Python and indirectly, SQL. This report outlines the processes and rules used while developing the project, as well as the information considered, such as user testing information.

2 Prototyping

CPS Battles was prototyped using the Figma platform, the link to it is here. Please note that that the prototype is not clickable and should be viewed as a series of images. This Figma link showcases the final version of the prototype, the prototype was changed over time based on user testing. Please view the user testing chapter for specifics.

3 User Groups

NAME of group	**Standard User**
WHO background	**Age:** Any
	Gender: Any
	Education: Any
	Abilities/Disabilities: Not blind and not an amputee
	Computer skills: Basic knowledge
	Number: Tens of users
WHY main goals	Check click speed and compare click speed against friends
WHAT equipment	Desktop computer or laptop
WHERE environment	At home
WHEN usage of system	**How often:** a few times on the day of discovery, then likely never again
	For how long each time: 10 minutes
HOW Important	Of paramount importance. This is the only user group.

4 Requirements List

Number	Requirement description (User story)	Priority	Additional Info
1.	As a user, I want to check my clicking speed	Core	Use Case #1
2.	As a user, I want to join a lobby and invite my friend, so that I can compare my clicking speed against theirs	Core	Use Case #2
3.	As a user, I want to see a leaderboard of all other users, so that I can see which users click the fastest and compare myself to them	Core	Use Case #3
4.	As a user, I want to log in or create an account	Core	Use Case #4
5.	As a user, I want to see and edit my own profile	Core	Use Case #5
6.	As a user, I want to see the history of all my played games	Core	Use Case #6

5 Use Cases

Name:	**As a user, I want to check my clicking speed**
Number:	1
Priority:	Core
Precondition:	None
Base flow:	1. The user starts clicking on the big “Click me!” button 2. The user either looks at the record reached at the bottom or the current speed at the top
Alternative flow:	None
Post condition:	The record is saved to the database by the system IF the user is logged in.
Actor(s):	User, System
Author(s):	Daniel Prizginas

Name:	**As a user, I want to join a lobby and invite my friend, so that I can compare my clicking speed against theirs**
Number:	2
Priority:	Core
Precondition:	The user is logged in
Base flow:	1. The user starts clicking on the big “Play with friends” button 2. The user clicks on the copy button to copy the game link 3. The user opens a 3rd party application and shares the link with a friend and waits for them to join 4. The user chooses the game mode as they please and presses start when the friend has joined 5. Once started, the user will click on the big “Click me!” button while looking at the right hand side of the screen to see progress compared to the other user.
Alternative flow:	• The user is sent to the login page since the user is not logged in. ○ The user follows use case 4 ○ The user restarts the use case.
Post condition:	The game is saved to the database, as well as any record set while playing.
Actor(s):	User, System
Author(s):	Group 34

Name:	**As a user, I want to see a leaderboard of all other users, so that I can see which users click the fastest and compare myself to them**
Number:	3
Priority:	Core
Precondition:	None
Base flow:	1. The user clicks on the leaderboards button at the top 2. The user flicks through the pages as they please 3. The user clicks on a username to view their profile if they please
Alternative flow:	None
Post condition:	None
Actor(s):	User, System
Author(s):	Group 34

Name:	**As a user, I want to log in or create an account**
Number:	4
Priority:	Core
Precondition:	The user is logged out
Base flow:	1. The user clicks on the login button at the top right corner 2. The user either enters their credentials or chooses to create an account by clicking the “create account” hyperlink and doing the same thing 3. The user has either logged in or created their account
Alternative flow:	• The user’s credentials were incorrect ○ The user is thrown back to the login page with an error message ○ The user re-enters their details and tries again
Post condition:	The users account exists in the system and the user is logged in
Actor(s):	User, System
Author(s):	Group 34

Name:	**As a user, I want to see and edit my own profile**
Number:	5
Priority:	Core
Precondition:	The user is logged in
Base flow:	1. The user clicks on the profile button at the top right corner 2. The user views their own profile or clicks the edit button 3. If the edit button was clicked, the user enters the information they wish to change and submit the form 4. The system updates the user information
Alternative flow:	• The user entered incompatible information into the form ○ The user is thrown back to their profile page with an appropriate error message ○ The user retries • The user is sent to the login page since the user is not logged in. ○ The user follows use case 4 ○ The user restarts the use case.
Post condition:	The users profile is updated if the user chose to edit it.
Actor(s):	User, System
Author(s):	Group 34

Name:	**As a user, I want to see the history of all my played games**
Number:	6
Priority:	Core
Precondition:	The user is logged in
Base flow:	5. The user clicks on the profile button at the top right corner 6. The user scrolls down and has a view of their previous played games.
Alternative flow:	• The user is sent to the login page since the user is not logged in. ○ The user follows use case 4 ○ The user restarts the use case.
Post condition:	None
Actor(s):	User, System
Author(s):	Group 34

6 User Testing

6.1 Prototype Interview Summary

When the user first had a look at the website they immediately noted that rounded corners on everything would work much better than such a sharp design. The user liked the layout of the homepage, it was obvious for the user how the click speed button worked and the user could tell pretty easily what all of the buttons were meant to do.

When the user was tasked with creating a lobby, they were told that only the user who created the lobby would be able to edit the lobby, whereas the second player to join the game would only be able to wait, after hearing the explanation, the user explained that it would be useful for both players to be able to see who the leader of the "party" was.

It was clear that the user knew exactly how the whole lobby menu functioned but they were still concerned with a couple more visual aspects; the shareable link bar/background was too short and that "<value>" should actually be the value selected by the value selector. The user was reassured that the second point was only a visual indicator as it was a prototype.

Moving on to the leaderboard, the user was pretty happy with everything but noted that having some sort of thin lines between the leaderboard entries would help, also the user noted that the little arrows for the pagination system would have too small of a "hitbox" and suggested giving them a background like all of the other buttons to make them effectively bigger.

Lastly on the profile page, the user strongly believed that the game history should be expanded to cover the page from left to right with sections instead of just being a bit of text on the left-hand side, the user also suggested having it be a bit more fun with the win and loss text having a pixel font.

Features implemented based on user prototype testing:

Rounded corner design

Golden crown to indicate leader

Longer black bar behind the link on the lobby screen

Thin black lines between leaderboard entries

Button background added behind pagination arrows on the leaderboard

Expanded game history view on the profile

Comparison to similar services

6.2 Type Racer

When making the prototype, I was still wondering how best it would be to “create a lobby”, the first website/app I thought of that does this sort of this was type racer, a website where you compare typing speed against other players. Typeracer has a big button “Create a race” as well as a “Copy link” button to share with others so that they may join the lobby.

6.3 Other random click speed websites

On CPS Battles, the cps is constantly shown, the problem with this is that the CPS spikes at the beginning and then hardly changes later on because the average is either calculated on too few or too many references, I went to other click speed test websites to see how exactly their constant cps looked like, did it spike? Did it stagnate? After a while I realised that not a single website had constant cps showing, they only showed the average cps after a certain amount of time had passed.

I likely could have found a reference after more searching but I gave up there and went with my implementation; biasing the first few clicks to count for less to prevent a spike and resetting the cps counter every few seconds so it wouldn’t stagnate.

7 Programming Rules

7.1 Naming conventions

PascalCase is used for classes and *snake_case* for functions/variables/filenames since that is what the linters will be set to by default.

7.2 Code organization

CSS, Images, JS and the base html template are organized into the static folder, while the rest of the code (Python backend and html) for pages are organized into their own Django apps.

7.3 Commenting/documentation

Normal commenting will be performed on especially difficult code, nothing special here. The readme file explains the requirements for the website as well as any additional information needed to get full functionality. This very document is then used to document development.

7.4 Use of version control

Git is used throughout development for this project, meaningful changes and bugfixes are committed to the repository on a regular basis with descriptive messages, this allows changes to be reversed very easily by the developers if an issue arises.

7.5 Coding language/style guide adherence

The main languages used for this project are HTML, JavaScript, CSS, Python. I will be using the Ruff linter for Python, ESlint for JavaScript and for HTML and CSS, I will use the W3C validator. All HTTP urls for API calls adhere to the REST standard.

8 Security Concerns

There are two types of security concerns:

Gameplay Integrity and User/System Integrity.

I believe gameplay integrity to have very little worth focusing on since for a game like this, cheating will always be possible with auto clickers, this is why all of the gameplay code is run client side, cheating will always be possible anyway, so might as well offload the code onto the client to at least take advantage of performance and server-load relief. Of course the BARE minimum was implemented server side, like only allowing users associated with a game to edit it and such.

However user and system integrity is extremely important, I want users to feel safe that their password and personal information won't be stolen via CPS Battles and for them to have peace of mind that their profile avatar won't be vandalised by the next time they log on.

Of course Django uses SHA-256 for password encryption, but even so I'd still prefer that user's can't see other users password, even if it is encrypted, this is why whenever a user is fetched via the DataLayer API, the password is always set to blank before sending it to any client.

Then, to prevent a user's account being edited without permission, the backend always checks if the user trying to edit a user is the same as the user being edited.

For system integrity, Django handles 99% of the issues like users uploading malicious files or injecting code. This is something I don't have to worry about too much for a non-enterprise project like CPS Battles.

9 Deployment

CPS Battles is deployed on Microsoft Azure. This means that the database, the web app and all of the images are served through Azure.

Deploying the database was a very simple process, especially so because I had experience. But deploying the web app itself brought great challenges, I had a lot to learn. The issues encountered were not limited to the live website being outdated compared to github commits, or staticfiles not being found, but this was all solved in due time by reading up on documentation for services like whitenoise which simplify the static files for Azure.

Then was the media (user uploaded files). For this I had to create a storage account on Azure and modify Django settings.py very specifically to get it to work, this process was not aided by the fact that to see changes on the website I had to commit a change and then wait up to 5 minutes for the whole web app to be rebuilt. This is why I decided to streamline the process by splitting up settings.py into two, a local version and a cloud version. This way I could choose to run the site locally and debug and test code on my machine without having to commit every single change and waiting minutes to see changes. I would then only commit the big changes. Settings.py will use the correct settings automatically with no input from the developer.

10 Conclusion

CPS Battles is a fully developed full-stack web application, created both to provide a fun and competitive environment, but also to showcase my skills as a developer.

Thorough feedback was collected and acted upon during the development process, as well as references from other services.

Security and programming best practices were applied throughout the project, I prioritized user data protection and responsible system architecture. Cheating was considered near impossible to combat so most backend protections were focused on user and system integrity instead.

CPS Battles is a demonstration of thoughtful design, secure implementation, and iterative development that fulfils all core user requirements and stands as a robust example of what can be accomplished through a user-first approach to software development.