Gamified Productivity Web App: TaskDrop

Layla Razvi
lrazvi@oxy.edu
Occidental College

1 Problem Context

Organizing and managing tasks is something that many people struggle with. Studies show that ”54 percent of college students think better organizational skills would improve their performance”[13], yet many of these students still struggle to take the initiative to organize themselves. There is a multitude of methods for task management such as the classic to-do list on a notepad, reminders set up on a phone, calendars, etc. While people often feel a sense of gratification after using these, many people still struggle with getting the motivation to organize, manage, and accomplish their tasks.

My goal is to create a simple productivity web application with a gamified interface with the goal to help engage and motivate users with organizing their tasks and making the process of task management more enjoyable for them. Through this web application, I chose to emulate a genre similar an idle game, which is a genre where the game runs and players have small interactions with the game. I decided on this genre because it would be something easy to digest and wouldn't take up too much time out of the users' schedules and cause extra stress to them. I wanted to focus on the rewarding aspects of gamification like points and badges to give positive motivation to the users and allow them more freedom in defining their goals.

2 Technical Background

The programming languages I utilized in this project included JavaScript, CSS, and JSX, a syntax extension of JavaScript that functions similarly to HTML. The main libraries I used to create the web application were React and Phaser. In the sense of creating my web application, I believed that React would be a great tool to work on the functionality of the overall application while I could utilize Phaser to incorporate the gamified elements into it.

I was able to set up the application as a React app and mostly organized my files where they were either React functions or Phaser scenes.

2.1 React

React is a front-end JavaScript library used for building interfaces. The main algorithmic functions I utilized in react include hooks, arrow functions, and custom events.

Hooks are an addition to React version 16.8 and are basically functions that allow access to state and other React features, also allowing coders to use React without classes. The hooks that I utilized in my web application were the State and Effect hooks. These hooks have to be specifically imported from React along with the React library itself.

I also used methods like `addEventListener`, `CustomEvent`, and `dispatchEvent` to create and fire functions triggered by certain events between React and Phaser.

2.2 Phaser

Phaser is a desktop and mobile HTML game framework for 2D games. The features I made use of from Phaser include WebGL and Canvas, the preloader, sprites, animation, inputs, and sound.

The preloader is a function in every Phaser scene used to load assets such as images, sounds, sprite sheets, tilemaps, JSON data, etc. It allows them to be parsed automatically and available to use in the game stored in a global cache. Phaser uses scenes to go from one screen to another. Each scene is usually created in its own file and typically uses the `preload()` and `create()` functions to add elements like sprites, text, buttons, etc. into the scene [1].

Sprites are computer graphics that can be moved and manipulated on screen. In Phaser, there are special features for sprites that allow you to position, rotate, tween, collide them and much more. Additionally, they have full input support, so you can click, drag, and touch them.

In terms of animation, Phaser supports sprite sheets with fixed frame sizes. For Phaser inputs, it won't matter whether the input came from a touch screen or mouse. Phaser inputs includes functions for touch, mouse, keyboard, gamepad, and more.

Lastly, in terms of sound, Phaser supports web and legacy HTML audio and automatically handles easy audio sprite creation, looping, streaming, volume, playback rates, and detuning. [1]

3 Prior Work

3.1 Productivity Applications

Productivity applications have existed for a while now, but some of the applications I have looked into include the Pomodoro chrome extensions, Todoist, and Habitica.

Todoist is a task manager and to-do list application created to organize work and life. The application is available on the Apple App Store, Chrome Web Store, and Google Play. The application has to-do lists with features such as quick add, recurring due dates, and section and sub-tasks. Additionally, Todoist's task management features include priority levels, favorites, reminders, task collaboration, personalized task views, and progress tracking. [4]

For the Pomodoro applications, I specifically examined "Focus To-Do: Pomodoro Timer and To Do List" and "Otto – Pomodoro timer and website blocker". Focus To-Do includes features such as timing, task organizing, time tracking, schedule planning, and reminders. Its description in the Chrome Web Store claims that " you can capture and organize tasks into your todo lists, start focus timer and focus on work and study, set reminders for important tasks and errands, check the time spent at work." [2] The task management features also include features for recurring tasks, sub tasks to break down tasks into smaller actions, and color coded task priority.

Each of these productivity applications inspired the features I incorporated into my web application. I specifically focused on the task managing aspects although I was not able to incorporate as many of the features other than the to-do list, which I will delve into in my methods section.

3.2 Research on Gamification

Other than looking into productivity applications, I also explored some research to get a stronger understanding of gamification. One of the principal definitions of gamification was explained as "the use of game design elements in non-game contexts" [9]. My intentions with gamification particularly focused on user interaction and engagement to make task management a more enjoyable process for users. These ideas are highlighted in aspects of gamification design, which include sensory motor elements, motivation and persuasion elements, and elements supporting cognitive processes. Goals for sensory motor elements focus on the visual, audio, and tactile elements to increase appeal to the user. Gamification is said to have great potential from a cognitive perspective since effort is measured creating a motivational practice. Lastly, for emotional aims, the user is the center of the design process for gamification, taking into account their motivational configurations [9].

3.3 Gamified Applications

‘ Otto Pomodoro is a timer and website blocker. Unlike Focus To-Do, Otto introduces a point based reward system intended to "make work a bit more fun and also to remind you every action has a price". In the extension, the character Otto is a digital tamagotchi that the user is responsible to look after. Going to blocked websites while the work timer is on depletes Otto's health and avoiding those sites along with completing sessions helps Otto gain health. Other features in Otto Pomodoro consist of insights, notifications, and break mode.[5]

Habitica is another productivity application made for habit building. It includes in-game rewards, punishments, as well as a strong social network. For habits and goals, Habitica allows users to track habits, daily goals, and to-do lists. The reward system in Habitca checks off tasks and lets users level up their avatars and unlock other in-game features. Other gamified aspects include fighting monsters with friends and getting gold to unlock more in-game features. The app is overall meant to help players with health and fitness, schoolwork, and more.[3]

Both of these gamified productivity applications were great inspirations for my project. The tamagotchi and reward systems in Otto Pomodoro was an attribute that I included in my web application with the cat sprite and affection points. Habitica inspired the pixelated aesthetic I went for in the cat sprite, text, and background.

4 Methods

Before starting the code for my application, there was a lot of groundwork I needed to cover to ensure that my project would meet my goals. Prior to the beginning of my project, I did not have much previous experience with web development, so one of my first steps was to get familiar with some of the common web development languages such as JavaScript, CSS, and HTML as well as looking into an ideal framework to use for my application. While I was trying to get familiar with these new topics, I also had to start doing user research to determine specific features potential users of my application would want. I struggled a lot with this beginning process because there was so much unfamiliar material I had to go over, and I wanted to make sure I was doing everything correctly. I ended up going through this process of getting all the information I needed to create my web application in a less organized manner than I intended to and ended up learning a lot more through the process of working on the application.

4.1 Creating the ToDo List

I eventually decided on utilizing React for making the productivity portion of my application and started out by trying out different tutorials on making to-do lists. I initially followed a tutorial by Alberto Montalesi called ”Create a To Do List with JavaScript” which used HTML and CSS along with JavaScript. The tutorial was quite simple and easy to follow for the most part, but I ran into two issues. The first issue had to do with creating the HTML file in React, which did not work due to the way the files were set up in React. I was able to solve this by using JSX, which is very similar to HTML as mentioned previously, in the return section of my React functions. The to-do list started working after this, but it stopped working soon after. I checked in my console while running the application and it gave an error saying the `document.queryselector` was null. I attempted many solutions like changing the CSS selector from an `id` to a `className`, rearranging the function, etc. but none of them made the list function the same as it did before. [10]

After spending too long trying to fix one error and making no progress, I decided to start from scratch and find a tutorial with a different method. The main tutorial that helped me in creating was a YouTube video by Arpan Neupane where he introduced the `useState` hook and the arrow functions to me for the first time, which eventually became very apparent throughout my code [11]. The list stayed functional after following this tutorial and making some of my own adjustments.

One of the major adjustments I wanted to make to the list that I made with React was adding an option for adding more detailed steps or descriptions to the tasks as well as a deadline option for when the user may need to have a task finished by. I began working on the deadline option first by simply duplicating the code I already have for entering an input for the task. However, when I had 2 inputs for the list, categorizing the first input as the task and the second one as the deadline, when I entered both of them, the list that resulted would only print the last input I entered and never both of them. I attempted to research tutorials to see how I could fix this issue, but from the way my algorithm was set up, there were not many options, so I decided to not spend too much time trying to fix this issue and moved on to the next. In terms of adding a description to the tasks, one of the ways I implemented it partially worked where I turned the list item into a button and connected a function to the button's click event to create a new input and add a new list underneath the task. This, however, did not end up working out either because the description list under one task ended up going every other task as well. I had trouble trying to fix this as well, and after working on other sections and coming back to it, I decided to stick with my initial to-do list.

4.2 Integrating React and Phaser

While I was making adjustments to the to-do list, I was also looking into some libraries and CSS designs to help create the gamified aspect of my application. I discovered Phaser and thought it was a great fit since it was also a JavaScript library for creating online games and seemed very straightforward to use. After I decided to use Phaser I looked into tutorials to integrate React and Phaser together.

One of the first tutorials I followed was ”Integrating React and Phaser 3” by Taylor Nodell. From this tutorial I was able to create a very basic integration of React and Phaser in my project where I created a new Phaser game and rendered the React function I used to make the to-do list in the same file. When I ran the application, it appeared to work showing the example image I rendered through Phaser and displaying my to-do list made through React at the bottom of the screen.

The next step was to create a start scene where it transitions from the start screen to a scene containing the list, in other words, I needed to figure out how to display react elements triggered through Phaser events. For this, I found another tutorial by Pablo.GG on how he made a top-down game version of his blog with Phaser and React. In this tutorial, Pablo.GG introduced how to communicate events between React and Phaser through dispatching them [7]. First, I moved the configuration of my game into my main function using the useEffect hook. I had some trouble trying to implement the start screen due to the way the code was presented in Pablo.GG's blog, so I went to his Github repository to understand how it worked in the full context. At this point, I decided to create a separate project folder to test out different algorithms rather than running it all in the same project folder. Creating a start button in Phaser and having it transition to a new scene was quite straightforward in Phaser, but it took me a little while to fully understand how dispatching events worked to integrate React into the event. I was able to figure this out by following another one of Pablo.GG's tutorial, "Creating a Dialog box with React for a Phaser Game" [7]. From this tutorial, and examining Pablo.GG's code from his Github repository, I was able to get a better understanding of dispatching events through creating the dialogue box. Creating the dialogue box itself was also quite a complex process, but I started out by creating a separate file for the dialogue box and then another file for the typewriter animation effect for each message. I utilized React Spring, an animation library, for animating the message, imported the message file into the dialogue box file where the actual messages and box was created, and then imported the dialogue box into my main file and function where I returned it as an element. Now that the dialogue box was working, I figured out how to trigger it through Phaser events by first using the useState hook to create a boolean variable to be changed with a specific event trigger, then creating and adding a window event listener as an arrow function (inside the same useEffect hook I configured the Phaser game in) where the boolean value is changed with the state updater. I used this boolean variable in the return method of my main function in an inline conditional JSX statement where if the value were true, the dialogue box would display, otherwise nothing would be displayed. I then made a custom event with the event listener's event type, and finally dispatched it at the point in the code where the event was triggered (in the case of a button, it would be dispatched in an onClick event). With understanding this process, I not only utilized event listeners and custom events for the dialogue box, but all throughout my project, most importantly the list. Once I was able to trigger the dialogue box, it ended up rendering either above or below the Phaser canvas instead of directly on it, which troubled me for a bit, but I eventually figured out its position just needed to be adjusted with some CSS.

After figuring out this process, I went back to working in my main project folder and was able to get the dialogue box and to-do list to show up after clicking the start button. I had also moved my list function into a separate file and turned it into an arrow function. I imported and rendered it in my main function, similar to how I did with the dialogue box. I followed the same exact process of triggering the display of the list through dispatching its own event by creating another stateHook, event listener, inline conditional in the return method, etc.

My next step was to trigger specific dialogue messages after certain points in the game. At this point I was using a single dialogue message with multiple strings that I created as a list. I wanted to have multiple messages triggered at different events, so I started by creating multiple lists of dialogues assigned to different variables. I then put these variables into a list labeled "dialogues" in the order I wanted them to occur. Once I did this, it was just a matter of adjusting the event listener and the custom events. I added another useState hook to indicate the current message initializing it to 0 and used the variable of this hook in my dialogue event listener where I added the state updater for the current message to set it to a number indicating the current message. I also had a stateHook used for the actual message initialized as an empty string and added the state updater for that in my dialogue event listener with the message being set to my dialogues list at the index of the current message. When creating custom events, I was able to use the same event type for all of my dialogues in multiple places; I just had to add the detail of the current message number along with the event type which resulted in the correct messages being displayed at their assigned events.

After that, I added a point box along with my list to keep track of how many tasks the user completed and also added a section for other thoughts and goals, which was some free space for the user to write whatever they wanted in case some things didn't work with the list format. I created both of these in my list function and created a button under my to-do list to make the other thoughts section appear and disappear.

4.3 Cat Sprite and Sound Effects

After finishing up the interface with React, I began working on my Phaser scenes to add a cat sprite as little companion for the user as they write down their tasks as well as adding a basic setting.

I decided to use a spritesheet I found on itch.io by Pop Shop Packs for the cat sprite. I added the spritesheet into my project folder and imported it into my phaser scenes using the load method the `preload()` function of my scene. I then created a new animation for the cats idle position in the `create()` function using the frames of the cat sitting and wagging its tail, indicated by the start and end numbers in the frames section of the animation function. When I first attempted to run the application with the sprite, a black box showed up instead of the sprite. This occurred because the path to the spritesheet I wrote in the load function was not being recognized. I had some trouble with this for a bit but was able to figure out I just had to import the spritesheet with a name rather than the path directly allowing the sprite to render properly.

I then added a forest tile set I also got from itch.io by MamaNeZakon under the cat sprite so it wouldn't look like it was floating. When I loaded them as images they weren't rendering properly, so I added them as textures instead in the `preload()` function. I had to repeatedly add the same texture at different positions in the scene to get it to extend across the screen appearing as a floor for the cat sprite to walk on.

After that, I worked on different interactions with the cat sprite for the user to make it walk and play with it. I started with making the cat walk and had to do this by creating another animation function, exactly the same as the idle animation but with different frame numbers. I then created a function for when the mouse is clicked where I played the animation of the cat walking along with increasing the x value of the sprite so it simultaneously moves as the walking animation plays. For playing with the cat, I created another animation and played it on the event where the mouse hovers over the cat sprite.

Lastly, for the audios I discovered some free audios online for a button click, ding, and cat meow sound effects. The audios work very similarly to adding sprites and animations in Phaser. I begin by importing and using a specific load method for audios in the scene and then play the audios in the create function. I actually ended up using event listeners for all of them because all of the sounds were triggered from React events. The button click effect was used for almost all of the buttons except for checking off tasks in the list, which was exclusively for the ding sound effect. The meow sound effect was used every 3 tasks the user accomplished, which also added affection points.

While working on the dialogue box, I did have some issues with the Phaser canvas being rendered repeatedly (I would scroll down and there would be multiple canvases rather than having a single one on the screen) when I switched to the second scene. I didn't get an opportunity to figure out this issue, but it stopped happening around the time I finished working on some dispatch events.

4.4 Deploying the App

The last thing I needed to do was deploy my web application, so it would be accessible through a link for my users. I initially attempted to deploy it through Github pages which did not work out because I found out that Github pages only works for static websites. I then tried Netlify where I created a build for my application, added it into the site, and my web application was published in about 30 seconds.

5 Evaluation Metrics

Determining my evaluation metrics was a challenging process since I had to take into account multiple factors such as engagement, usability, productivity, and overall how enjoyable my application was to use.

5.1 User Research

Before I began my project, I conducted some user research, as mentioned in my methods section, to get familiar with how users generally deal with task management. Since I did not have much experience with user research, interviews, and testing, I decided to look into some guides to help me with this process. An article that I especially found useful for learning how to begin this process was ”How to conduct good user research and why it's so important” by Laurent Grima [8]. In the article, Grima begins by discussing the importance of user research, mentioning the potential bias of the designer (me), how things people may say in interviews might be different from what they do, and the importance of making the problem clear. He then provides an introductory guide for good user research that includes the following steps: define your research objective, decide who you want to interview, design an interview guide, conduct the interview, summarise your learnings. Within each of these steps, Grima provides clear information on how to approach each of these steps like figuring out what the designer might want to learn about their users with objectives such as learning their situations, goals or motivations, and behaviors.[8]

Following Grima's instructions, I started out by restating my goal for my application, which is to make task management more engaging and enjoyable for users. I then defined my target audience, and decided to go with fellow college students who may struggle with or are unmotivated with keeping track of and organizing their daily tasks. I chose this demographic not only because they were the most accessible for me but also due to statistical data of students struggling with time management. It is said that 88 percent of students wish to improve their time management skills, and that around 70-75 percent of students tend to procrastinate on their academic tasks [13]. I do not expect my application to completely fix these students time management and organizational skills since that is something that takes time and practice, but instead I want to focus on making the process of task management a more engaging and enjoyable to possibly increase some motivation in my users. After that I came up with some interview questions to figure out how users currently manage their tasks, what their motivators are, how they usually reward themselves, etc. What came next was to interview the users and utilize my findings to aid in the design of my application.

I took some time to find people that fit the criteria (as mentioned earlier, college students who struggle with or wish to improve their current style of task management) for my target audience by asking acquaintances and people they know if they currently struggle with organizing their daily tasks and if they'd be interested in trying out my project. From interviewing them, I found that they tend to use things like planners, Google Calendars, or even a combination of multiple task management systems. Their motivators were also combination of negative and positive things such as fear of failing or wanting to be free from stress, and getting free time to do activities like gaming as the please. As for rewarding themselves for when they complete their tasks, their methods were pretty consistent, which were to play games or to entertain themselves in other ways.

From these interviews, I was able to come to the conclusion that users don't usually have one set way for organizing their tasks and that my application could function as one of the multiple ways they organize themselves. As for motivators and rewards, my initial idea of creating a point and reward system to create more satisfaction and enjoyment for the users seemed to line up well with their responses to my questions. These interviews also helped me define some of my evaluation metrics, especially from figuring out there motivations and ways of rewarding themselves.

5.2 User Testing goals

In terms of defining my user testing goals and evaluation metrics, I followed "User and Usability Testing Questions: Ultimate Guide" by Nick Babich [6] to come up with questions to ask my users that would help me properly evaluate my application. From this guide I was able to develop both general questions about the application like how easy was it to navigate, what features they liked the most and least, etc. as well as questions specific to the gamification aspect of my application like was their engagement in managing their tasks more or less than usual, how enjoyable they found it, etc. I found that for some questions allowing users to use a number (i.e. on a scale of 1-10) to represent their feelings about certain aspects of the application would be a more effective way to measure things like enjoyment or engagement. I was able to develop a lot of these questions from the responses I got from my initial user research and interviews. In addition to these questions, I also put more logistical questions like how many tasks they hoped to complete vs how many they completed, how long they used the application for, etc. These kinds of questions will be used to get more context on the on the organization style of the user and whether or not my application was able to adhere well to it or not.

For the user testing, I decided unmoderated remote testing worked best for my application since it allows the users to use them in their own environments at their own pace. I did one round of user testing and let my users try out my application for about a day. Since I deployed and published my application, my users were able to access it with just a link, so I emailed the link with the instructions on how to use the application along with the final interview questions for them to answer and email back to me once they were finished. While taking into account many students' struggles with time management and procrastination as well as the fact that they would be using the application only for a day, I believe positive user testing results would include the completion of at least half the tasks listed by the user as well as a rating of over 5 out of 10 for engagement and how enjoyable they found it.

6 Results and Discussion

Once I was able to figure out my evaluation metrics and how to approach my user testing, the rest of the process went quite smoothly. I will go through the survey answers I received from my users when they finished testing the application, how these answers show whether or not I reached my project goal, and possible improvements for the application to get better results in the future.

6.1 Survey Responses

I first asked my users the general time they started and finished using the application to get an idea of how long their day is. The average amount of time my users spent using it was about 8.5 hours from around noon till early evening.

The next question asked how many tasks they hoped to complete compared to how many they actually completed. There were some interesting responses here like one of my users stating they added tasks throughout their day rather than having a set amount of tasks to complete. Others either had a set amount of tasks or did not state a specific amount of tasks. It is a bit difficult to calculate a result that averages the task completion by all users since there were some distinct differences in their styles of organizing and completing their tasks (one user listed 14 tasks while another listed 3). I can conclude that the majority of them were able to complete a large portion of their tasks, meaning more than half of them.

With the following question, I wanted to get an idea of how my users felt about their productivity using the application compared to when they're not. Most of the responses stated that they either felt the same or slightly more productive with the idea of gaining points and the cat's affection motivating them. Along with this, I had them rate on a scale of 1-10 how engaged they felt in organizing their tasks as well as how enjoyable they found the gamified interface. In terms of engagement, the average was about 7.5 and the average rating for their enjoyment of the gamified interface was 7.

Lastly, I asked what kind of emotions the gamified aspect invoked and the users' favorite features. The users' responses were consistent since most of them stated they felt satisfaction, happiness, or both when using the gamified interface. For their favorite features, they all actually listed different ones, but the most common ones were related to the cat sprite and sound effects for finishing a task.

As specified in the previous section my evaluation metrics for a successful round of user testing were if they were able to complete more than half of their tasks listed, and a rating of over 5 for engagement and how enjoyable they found the application. In terms of how the users felt about their own productiveness, some procrastination statistics indicate that recognition and reward systems have been helpful to 49 percent of students [13]. While these results do fulfill my goals for my evaluation metrics, some of the responses do indicate that there was little or no improvement in their overall productiveness which shows there is much room for improvement.

6.2 Improvements

User testing is a way for developers and designers to get an understanding of how users feel while using their product, and this information can be used to make improvements to that product. For my application, I am very open to improvements and made sure to include a question on my survey to ask users what improvements they think should be made.

The most common feature to improve amongst the users was the cat sprite and allowing more options for its movement. I also received suggestions like unlocking more cat interactions through accomplishing more tasks which similar to an option I considered while making the application but couldn't go through with it due to lack of time. I completely agree with these responses since I spent more time on the functionality of the application, I would've liked to add more interesting interaction options for the cat sprite.

Along with the cat sprite, I also would add more options in terms of the task list and organization, which happened to be another suggestion from one of my users. As mentioned in my methods section, I wanted to add a deadline option to each task and perhaps provide a time limit for when the user can check off a task. I also think a feature for categorizing and organizing specific tasks would be a useful addition, similar to some of the productivity applications I examined in my prior works section.

Other than the application itself, I would also like to improve my method of user testing. I only had enough time for one round, but ideally I would have done multiple rounds, possibly for longer periods of time with more users.

Overall, my user testing results were eye opening in not only understanding how successful my application was in regards to my goal but also in the improvements and better ways I can create features to help users become more motivated in managing their tasks.

7 Ethical Considerations

While designs with gamified features often have positive intentions, there may be instances where design choices have to be reconsidered in the case of ethically questionable results. There are many instances of gamified systems being used in the workplace, but since my application is designed for the individual handling their personal tasks and not a collective work environment, I will be focusing on ethical issues regarding that.

There is a cognitive limit that an average player has when keeping up with and playing multiple games, so it is more common for them to advance in a few games at a time. These cognitive limits apply to gamified designs, resulting in the question of whether users will be able to keep up with these gamified systems along with their other responsibilites without reaching their cognitive limits [12]. I particularly designed my application so that it wouldn't require too much of the user other than checking in with their tasks while still allowing them to feel the satisfaction of being rewarded. In this way, I have attempted to take my users' cognitive limits into account.

Another issue that may arise with gamified applications has to do with the personal data that is generated from users. The current version of my application doesn't save any data from the users, once the website is closed or refreshed, it starts from the beginning. Despite this, I would still may continue developing this application that may end up utilizing more data from the users than it currently does. What I need to be conscious of in this case is making sure the user is aware of the data they may be feeding to my application and if they consent to it before they lose any control of that data.

References

[1] 2023. URL: https://phaser.io/.

[2] 2023. URL: https://chrome.google.com/webstore/detail/focus-to-do-pomodoro-time/ngceodoilcgpmkijopinlkmohnfifjfb?hl=en.

[3] 2023. URL: https://habitica.com/static/home.

[4] URL: https://todoist.com/features.

[5] URL: https://chrome.google.com/webstore/detail/otto-pomodoro-timer-and/jbojhemhnilgooplglkfoheddemkodld?hl=en.

[6] Babich, Nick. *User and Usability Testing Questions: Ultimate Guide*. 2021. URL: https://xd.adobe.com/ideas/process/user-testing/usability-testing-questions-tips-examples/.

[7] Benmaman, Pablo. *I made a top-down game version of my blog with Phaser and React*. 2021. URL: https://pablo.gg/en/blog/coding/i-made-a-top-down-game-version-of-my-blog-with-phaser-and-react/.

[8] Grima, Laurent. *How to conduct good user research and why it's so important*. 2019. URL: https://uxdesign.cc/how-to-conduct-good-user-research-and-why-its-so-important-55cf5fdda4f3.

[9] Marache-Francisco Cathie, Éric Brangier. *GAMIFICATION AND HUMAN-MACHINE INTERACTION: A SYNTHESIS*. 2015. URL: http://www.jstor.org/stable/43573772.

[10] Montalesi, Alberto. *Tutorial - Create a To Do List with JavaScript*. 2020. URL: https://inspiredwebdev.com/how-to-create-a-to-do-list-with-javascript/.

[11] Neupane, Arpan. *How to Build a Todo-List-App with React — Beginner Project*. 2022. URL: https://www.youtube.com/watch?v=MXId-Ae6k_I&t=664s.

[12] Sami Hyrynsalmi Kai K. Kimppa, Jouni Smed. *Gamification Ethics*. URL: https://www.utupub.fi/bitstream/handle/10024/155567/10.1007_978-3-319-08234-9_138-1(1).pdf?sequence=1.

[13] *Time management statistics everyone should know in 2023 (and beyond)*. 2023. URL: https://clockify.me/time-management-statistics.