Documentation Page

Bootstrap components used:

1. **Navbar**

   The Bootstraps' responsive navigation bar was used: *.navbar-expand-lg* was used for responsive collapsing, *navbar-dark* for white text on dark backgrounds, *navbar-brand* for the name of the organisation and also two navigation buttons(*nav-btn*), one that opens a modal with information about the charity and the other that opens a modal with the shopping cart.

   It is a standard Bootstrap navigation bar that it is controlled by my own JavaScript, not the Bootstraps' one.

2. **Grid System and Gutters**

   It is the Bootstraps' flexbox grid that I used to build my pages' layout, especially to lay out the donation items. I used the responsive breaking points like *col-md*, *col-lg* etc., to automatically adapt to screen size of the device. Also used with gutters, that helped with the padding between columns and rows.

3. **Buttons**

   I used the Bootstraps' custom button styles for different actions in my website. For example, *btn-success* was used for the Add to Cart button, *btn-primary* for the Checkout and Confirm buttons, *btn-secondary* for the Back and Cancel buttons and *btn-danger* for the Remove item button. And also style them accordingly to my design, for example size: *btn-sm* etc.

4. **Cards**

   The Bootstraps' cards were used as containers for every item and its information on the donation page. I didn't especially use class = "card", but I named the class *item-card*, so it would be easier to distinguish them in the code. Each of the item card contains the items name, image, price and add to card button. Also, I used padding and shading for them: *p-2* and *shadow-sm*, their class also has some css style and also their components like image and price.

5. **List Group**

   The List Group component was used for shopping cart items and the order confirmation summary at the end. The reason I used them is so I had a clean vertical layout with the items clearly separated, so it would be easier to read the quantities and prices.

6. Forms

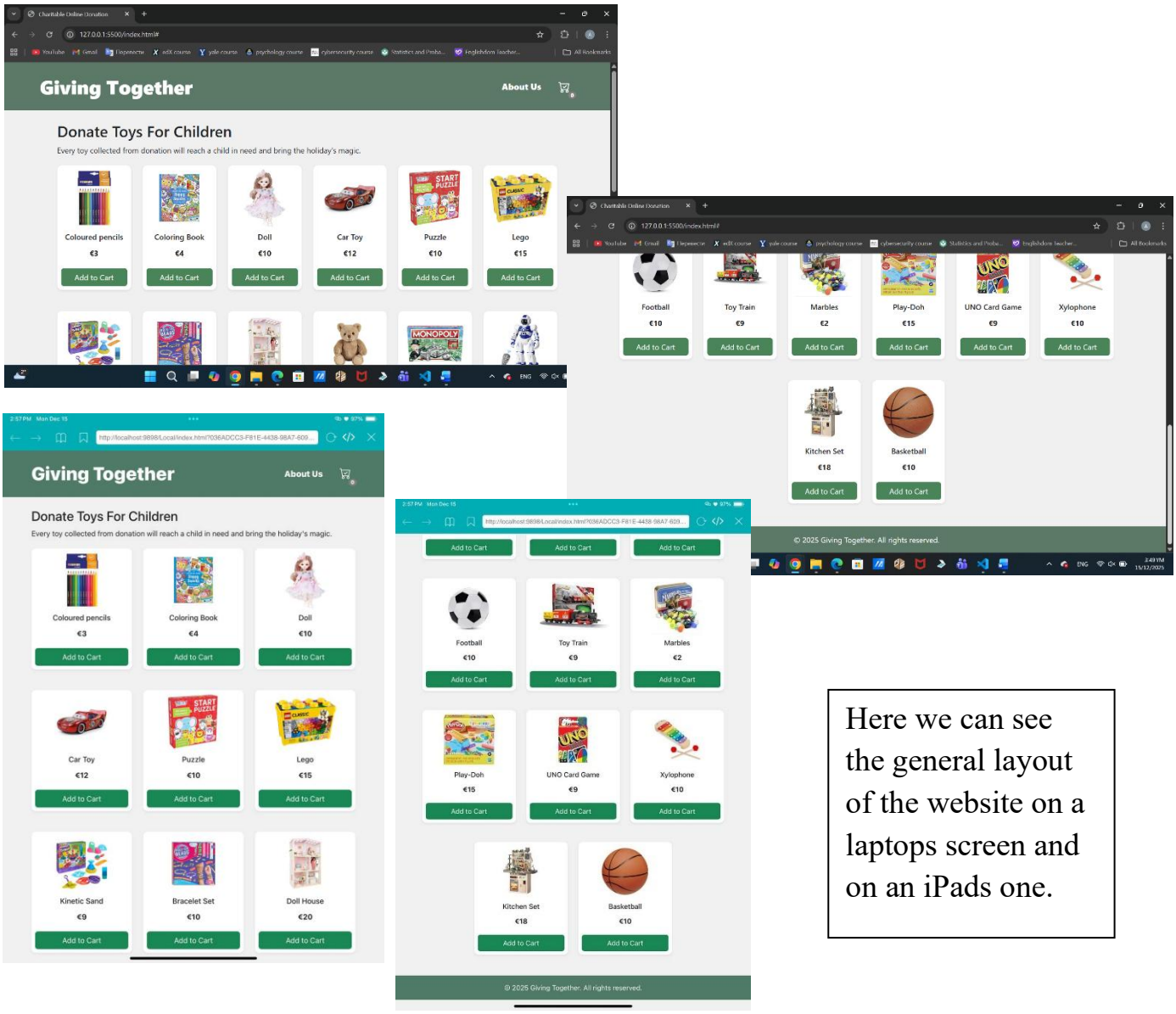
Forms were used at the checkout user information, shipping selection and at the payment details. I used the standard HTML <form> element with multiple <input> elements, for the user to enter. Also, the *.form-control*, *.form-check-input*, *.form-check-label*, *.text-danger*, *.small*, *.mt-1* Bootstrap classes were used for the styling of the forms (input/radio/error styling).

7. Utilities

Flexbox and spacing utilities were used for the layout of the page. From flexbox utilities I used: *d-flex*, *align-items-center* etc. and from spacing utilities *mt-3*, *mb-3*, *p-2* etc. were used. These utilities help in keeping the layout consistent across screen sizes and improve readability.

8. Icon

I also used the Bootstrap cart icon in the navigation bar, that I implemented using a special link in the <head> tag, under the link to the Bootstrap CSS library.



Here we can see the general layout of the website on a laptops screen and on an iPads one.

How does the JavaScript code work:

1. **About Us Modal**

In the navigation bar we have an About Us button, that functions based on 2 functions in the JavaScript file, the toggleAbout() function and closeAbout() one. What do the functions do: - the toggleAbout() opens the About Us modal when the button from the navbar is pressed and also moves the screen focus on the modal

- the closeAbout() closes the About Us modal when you press anywhere on the screen outside it.

2. **Display of Items**

The items displayed on the screen appear every time the page is loaded using the *window.onload = function()*. How does it work? :

1) Firstly, items were stored in an array *items* that contains their names, prices, images and a unique id.
2) Secondly, the function loops through every item in the array and writes an html code for every item, putting him in a card with its name, image, price and also an Add to Cart button. And than just lists this html code in the actual html page where the id *item-list* is mentioned.

3. **Adding Items to Cart**

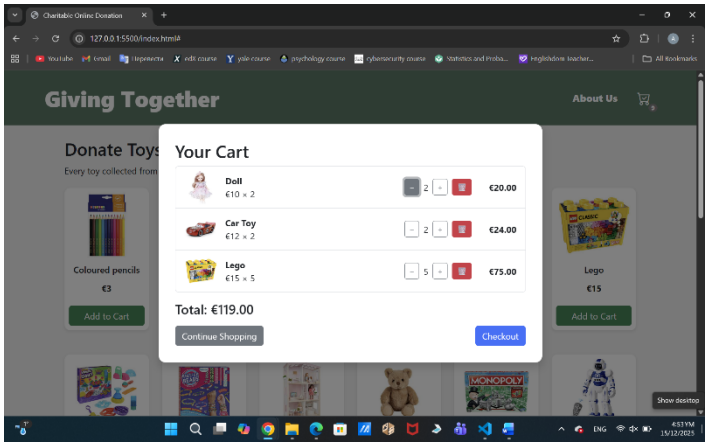
Items are added to cart by pressing the Add to Cart button under each item, for this I used the *addToCart()* function, that adds the items to the cart based on their id and updates the cart count after(the little number in the right lower corner of the cart icon). Also all the items in the cart are stored in a *cart* array.

4. **Opening and Closing Cart**

You can open the cart by pressing on the cart icon in the navigation bar and a modal with the carts' contents and total price appear. This was done using the *openCart()*, that works the following way:

1) Selects the cart modal and list container.

2) Clears previous cart content by setting *innerHTML* to an empty string.
3) Groups duplicate items and counts their quantities.
4) Sets total to be zero.
5) Loops through all the items and generates html for each one, containing name, image, price, quantity, subtotal, buttons to increase, decrease or remove the item.
6) Calculates and displays total cost.
7) Shows cart modal.
8) Traps focus inside the modal and sets it on first button.



You can close the cart by clicking on the Continue Shopping button or anywhere on the screen. This was done by using the *closeCart()* function that removes the "show" for the cart modal.

5. Increasing/ Decreasing Quantity, Remove Items Buttons

The functions of these buttons where also done using functions in JavaScript:

1)*increaseQty(id)* : - finds item in the array of items using its id, adds another instance of it to the cart array and refreshes the cart view and cart count

2)*decreaseQty(id)* : - works exactly like the function for increasing, the difference is that instead of adding it removes an item from the cart array

3)*removeItemCompletely(id)* : - removes all the items with the given id in the cart array and refreshes the cart view and cart count

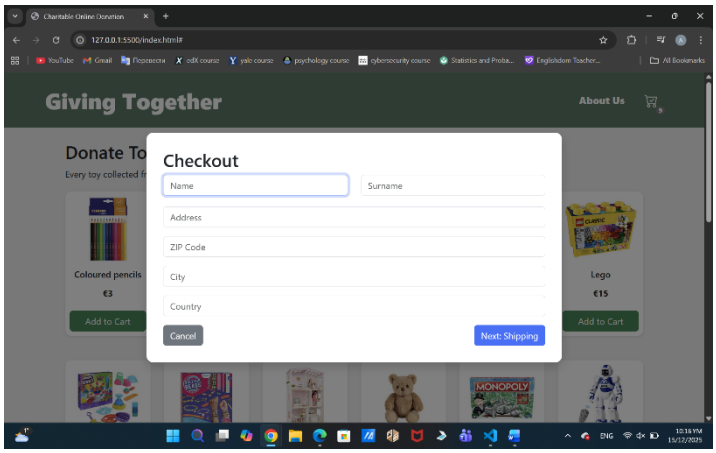
6. Cart Count

Is calculated using the *updateCartCount()* function that updates the number displayed near the cart icon every time something is added or removed, by setting the text to the length of the cart array.

7. Opening and Closing Checkout

After adding and removing all the wanted and unwanted items in the cart, there is a button that goes to checkout. The functionality of this button was done using the *goToCheckout()* function, that closes the cart and shows the checkout step 1, because the variable currentStep is set to be 1 every time before iterating through the next steps.

Also there is a function for closing the checkout, *closeCheckout()*, that closes the modal with the id *checkout-overlay* when you press anywhere on the screen beside the modal. And a *cancelCheckout()* function the closes the checkout modal and opens the cart one.



8. Going Through the Checkout Steps

To go through all the checkout steps I used the function *showCheckoutStep(step)*, that works the following way:

1) Stores the current step in a variable.
2) Using the *show* class makes the checkout modal visible.
3) Loops through all the steps, displaying them one at a time, based on the current step number.
4) If step 3 is reached, the checkout total is updated.
5) Like the rest of the functions of the modals, it traps the focus inside and moves it to the first input/button the model for accessibility.

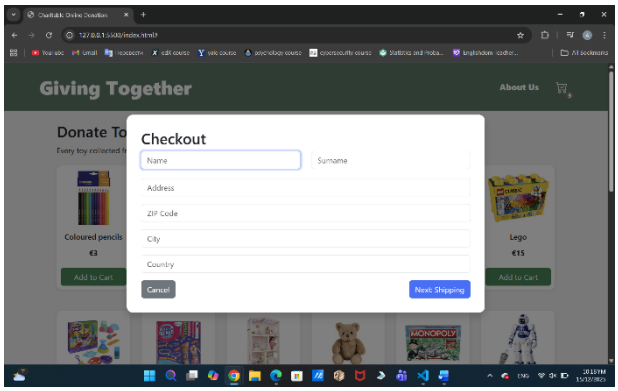
Then going to step 2, I used a function called *proceedToShipping()* that goes to the second step only if the information introduced in the form on step 1 passed the validation check and also one that gets cart subtotal (just prices of the items) called *getCartSubtotal()*.

At step 2, you have to choose the type of delivery, that was done in the html using a form, without any JavaScript.

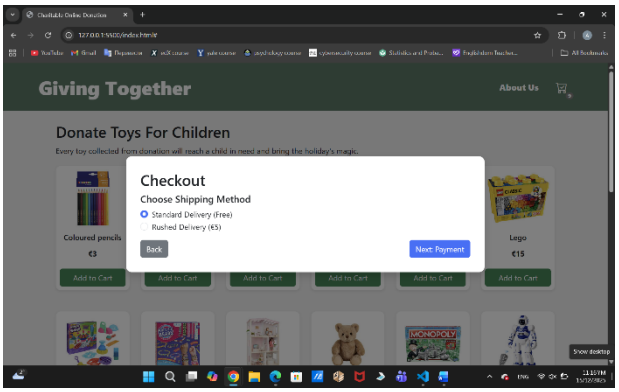
After completing step 2, automatically the function *updateCheckoutTotal()* calculates the total amount, including delivery (based on what was selected) and a discount of 10% (written as a constant in the code) if subtotal is greater than 30 Euros.

Going to the final step, step 3, I used the function *proceedToFinal()* that works only if a delivery service was selected.

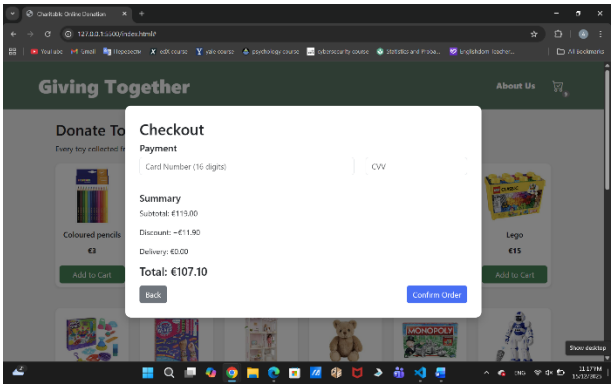
Step 1:



Step 2:



Step 3:



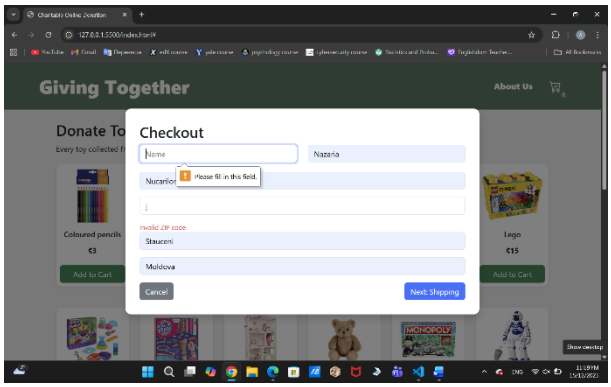
9. Validation of information

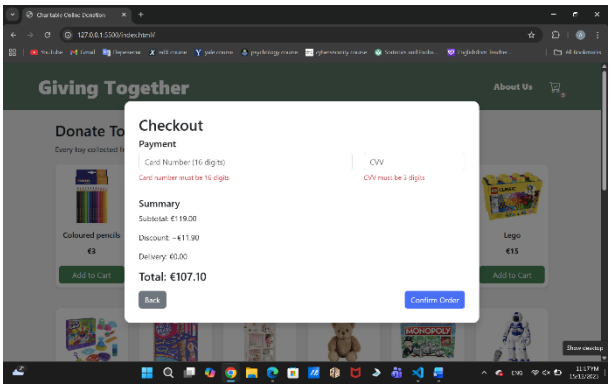
The validation of all the information introduced in the forms in the step 1 and step 3 of the checkout, was done using 2 functions:

1. *validateInfoStep()* : function that first takes and trims all the values introduced in the name, surname, address, zip code, city and country written in the input areas, than checks if the spaces for this information have been completed, if they weren't it shows an error and also you can't go further to next step. And it

also checks if the zip code introduced is formed of numbers, with the length in the range of 1 and 6, if it isn't it also shows an error and you can not go to the next step.

2. *validatePayment()* : takes the input for the card number and cvv and trims it, if no input was written it shows an error when trying to go to the next step, and also if the card length isn't 16 or the cvv length isn't 3 (in both cases numbers have to be inputted) also an error is showed and you can not continue.

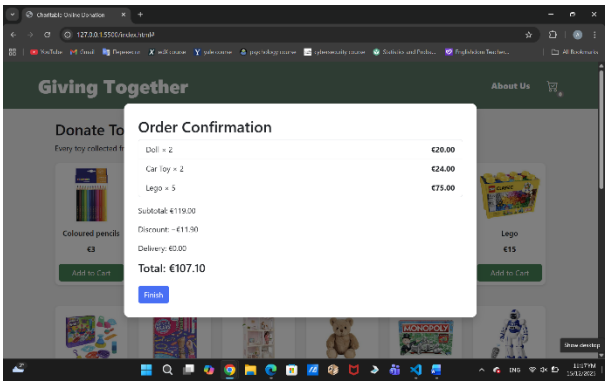




10. Confirmation of Order

For the confirmation of order 3 functions were used:

1. *submitCheckout(event)* : it prevents the page from reloading, validates payment information and only after validation shows the order confirmation modal.
2. *showConfirmation()* : selects confirmation-overlay from the html page, closes checkout, so only the confirmation modal is shown, it stores in variables the selected delivery, subtotal, discount and total and writes the html code using all this information, and then display the confirmation overlay. Also traps focus in the modal and focuses on the first button in the modal for accessibility. Than resets the cart array, and updates the cart icon to say 0.
3. *closeConfirmation()* : closes the confirmation modal.



11. Trap Focus Function (*trapFocus()*)

1) Keeps focus inside open modal.
2) Detects all elements inside that can receive focus.
3) Prevents Tab Key from going outside of modal.

4) Loops focus from first element to last and vice vers.
5) Allows to navigate the modal using Tab Key and Shift +Tab
6) Closes modal when Esc Key is pressed.

The main majority of this functions are used when buttons are pressed, you can see in the html file which function is called with the pressing of every button.

Sources I learned JavaScript from in order to be able to do this project:

- I want to mention I had a little bit of knowledge in JavaScript before this course so, some of the things I knew to do on my own but I also used google and different articles.
- Mainly used MDN webpage and all its information about JavaScript: https://developer.mozilla.org/en-US/docs/Web/JavaScript
- Also I used W3C for DOM and JavaScript Events: https://www.w3schools.com/js/js_htmldom.asp https://www.w3schools.com/js/js_events.asp