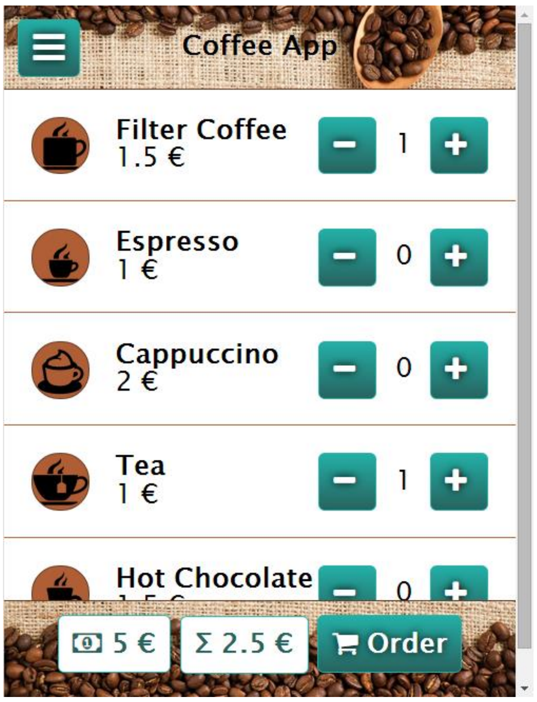
Final Exam

During this exam, you'll create a coffee app prototype, which will allow users to order hot beverages.

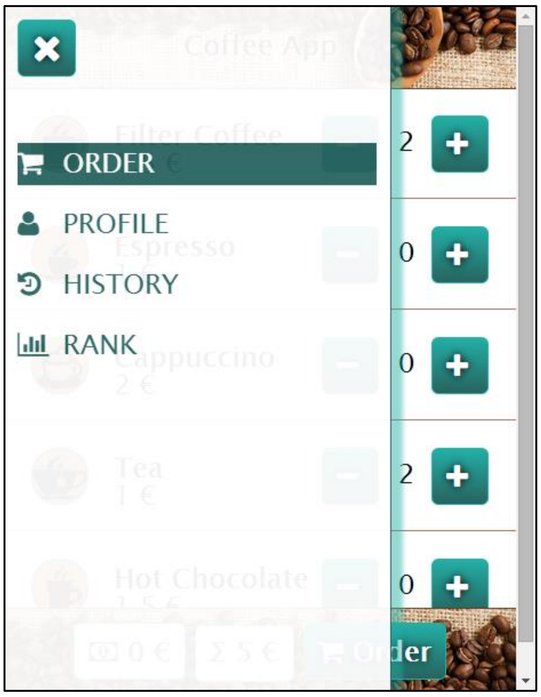
> **Begin**: 14:15h. Download the file "exam.zip" from Moodle, which contains a document structure, images and JS frameworks: https://www.moodle.tum.de/mod/resource/view.php?id=277272
>
> **End**: 16:15h. Upload your solution to Moodle at the end of the session: https://www.moodle.tum.de/mod/assign/view.php?id=277273
>
> **Total Points**: 102 P.

Please, make sure that the layout of your prototype corresponds to this sample below:

View 1: Order drinks



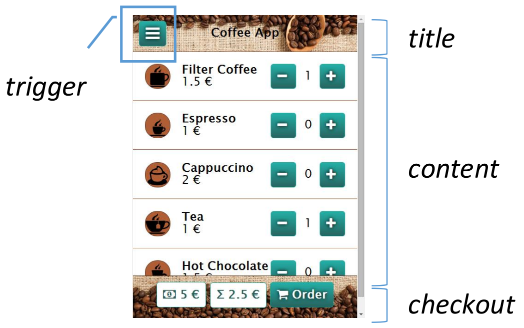
View 2: Menu



Please fulfil the following requirements.

HTML Structure (20 Points)

Requirement 1: Start with the provided "index.html" template. Create the four structural block level elements *trigger*, *title*, *content* and *checkout* with their corresponding ids. Use semantic elements where possible. (2 P)



Requirement 2: Please, insert the "Coffee App" headline in the *title* section. (1 P)

Requirement 3: Create placeholders for the two outputs 5 € Σ 0 € and the button Order inside the corresponding section. Those need to be block level elements and have an id. Assign them, as well as the *trigger* element, the class *output* or *button* respectively. (2 P)

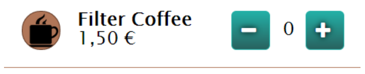
Requirement 4: Add the Σ sign as text. For adding the other icons, proceed in the following way: (3 P)

Add elements of the following Font-Awesome classes (present in the included Font-Awesome library, http://fortawesome.github.io/Font-Awesome/):

- *fa-money* (for the available credit)
- *fa-shopping-cart* (for the "Order" button)
- *fa-bars* (for the *trigger* icon)

Increase the size of the *trigger* icon by adding the designated Font-Awesome class.

Requirement 5: Create a block level element of the class *item* (later containing the drink information) inside the *content* section: (1 P)



Requirement 6: Create six separate elements for *image*, *name*, *price, remove, count* and *add* inside the *item* box. Use dummy text and one of the coffee images in the folder "images". Each of the elements need to be addressable via CSS/JS (keep in mind, that there will be multiple automatically generated *item* boxes). Let the and elements appear as buttons. Use *fa-plus* and *fa-minus* of Font-Awesome. (6 P)

Requirement 7: Your app has to consist of only one HTML document (containing the order view 1 and the menu view 2). Add a *menu* section (with corresponding id) in your body after all previous sections (1 P)

Requirement 8: Add a list containing the four items "order", "profile", "history", "rank" (all written lower-case). Add the corresponding Font-Awesome icons (*fa-shopping-cart, fa-user, fa-history, fa-bar-chart*) in extra elements. (4 P)

CSS Layout & Style (50 Points)

Colors

Use the following complementary colour theme:

light café brown	*dark café brown*	*light café turquoise*	*dark café turquoise*
RGB: 176, 94, 53	RGB: 100,60,40	RGB 35, 176, 167	RGB: 40, 100, 96
#B05E35	#643C28	#23B0A7	#286460

Sizes

For all sizes and distances, use *multiples of 24 pixels!*

Requirement 9: Modify the *title* and *checkout* sections at the same time in CSS (using one selector for both sections): (4 P)

- They cover the full screen width
- They are 72 pixels high.
- Text is centred

Requirement 10: Position the *title* and *checkout* section: (4 P)

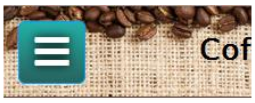
- *Both* sections do not move while scrolling the browser
- *Both* sections are layered in front of other elements
- *Title* is placed at the topmost part of the browser
- *Checkout* is placed at the very bottom part of the browser

Requirement 11: Style the *title* and *checkout* section: (5 P)

- Below the *title* and above the *checkout* is a "*dark café brown"* 1 pixel thick line
- The image "coffee-beans.jpg" in the folder "images" is used as a background for both
- The background image is scaled to be fully displayed in its width
- The *title* displays mainly the textile and spoon part of the image
- The *checkout* displays mainly coffee beans.

Requirement 12: Move the "Coffee App" headline down by 24 pixels. Move the *checkout* section's contents down by 12 pixels. (2 P)

Requirement 13: Place the *trigger* element to be fixed, bring it above all other layers and centre it in front of the left part of the *title* as shown: (3 P)



Requirement 14: Style all *buttons* and *outputs* for the following requirements: (3 P)

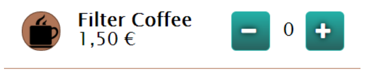
- Text is bold
- Border is "light café turquoise"
- Border is rounded

Requirement 15: Style all *buttons* to meet the following requirements: (3 P)

- The background is a gradient from light to dark "café turquoise"
- The text and icons are coloured white
- There is a black shadow centred behind the text, it blurs for 10 pixels.

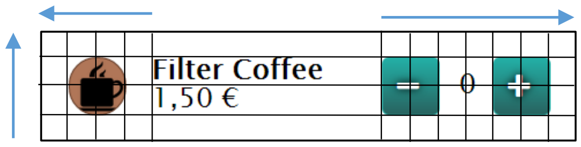
Requirement 16: Modify the *content* element (e.g., the inner distance), so that its contained items are never hidden by the *title* and *checkout* bars. (1 P)

Requirement 17: Style the *item* box to match the following requirements: (2 P)



- Height: 96 pixels
- Solid line below in "light café brown"

Requirement 18: Align and style all elements in the *item* box according to the 24 pixels raster given in the image below: (7 P)

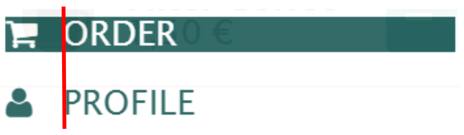


- Directly place all elements (*image*, *name* and *price* are oriented left, *remove*, *count* and *add* are oriented right).
- Size all elements (where necessary)
- The product *image* background is "light café brown", its border "dark café brown"
- The *image* is placed in a circle frame
- The product *name* is bold
- The product *count* is centred

Requirement 19: Format the *menu*:

- The *menu* fills the left part (70%) of the screen and its full height (3 P)
- It is placed in front of all other elements, but behind the *trigger* (1 P)
- It has a white background, which has a transparency of 5% (1 P)
- The right border is "dark café turquoise" (1 P)
- On the right side, it has a "light café turquoise" shadow, which is 50% transparent (2 P)
- Padding: top 96 pixels, otherwise 12 pixels (1 P)

Requirement 20: Format the *menu* entries:



- Remove the bullet points and the list indentation (2 P)
- Text is written in capital letters and "dark café turquoise" (1 P)
- Increase the distance between the menu entries by 24 pixels (1 P)
- Invert the colours of the first list element ("order", without using an id or a class): (1 P)
- Let the text start at the same position after the icons (indicated by the red line above) (1 P)

Requirement 21: Modify the *menu's* CSS so it's no longer displayed. (1 P)

JS/jQuery Interaction (32 Points)

For the interaction part, you may generally use JS and/or jQuery.

Requirement 22: The overlay can be opened and closed

- Add an event handler to the button to show/hide the *menu* section when clicked. (2 P)
- Also change the symbol of the button to when the *menu* is open (toggle between class *fa-times* and *fa-bars*) (2 P)

Requirement 23: Products are written programmatically into the *content* area

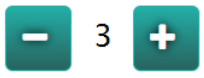
- Create a `function updateProducts()` (1 P)
- Inside this function, loop through all items in the given array `products` (2 P)
- In this loop, access the particular item and add its name to a local variable `productsHtml` (3 P)
- Modify the html of the *content* section to display this resulting list of coffee products (1 P)
- Call `updateProducts()` so it is executed (1 P)

Requirement 24: Ensure, that `updateProducts()` is called **after** the DOM has loaded (2 P)

Requirement 25: Create the whole item section programmatically

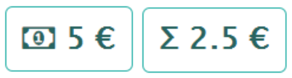
- Write a `function createProductHtml(id)` which accepts a number as an input and returns the whole *item* section of the product with the corresponding id (the id is its position in the `products` array) in HTML (4 P)
- Call the `createProductHtml()` in `updateProducts()` in order to display all products(1 P)

Requirement 26: Implement the *add*, *remove* and *count* functionality:



- Implement a function `function modifyProduct(id, amount)`. When called, it increases/decreases the `count` attribute of the product with the `id` corresponding in the `products` array. Increasing/decreasing is controlled by the argument `amount` (can be +1 or -1). (2 P)
- The corresponding event handlers are created programmatically in `createProductHtml()` (2 P)
- No amount may be negative (1 P)
- The content is updated during each change (1 P)

Requirement 27: Implement the *credit* and *sum* functionality:



- The *credit* (left) starts with a balance of 5 € (1 P)
- The *sum* (right) starts with a total of 0 € and displays the price for all selected products. (1 P)
- The *sum* is updated whenever the selection changes (1 P)
- Trying to select too many products triggers an alert message "You have insufficient credit!"(1 P)
- Clicking "Order" decreases/updates the *credit* by the *sum*, shows a message and resets the item list. (3 P)