

1. INTRODUCTION

The target of this test is to evaluate your knowledge about existing technologies to develop front-end web applications and evaluate your attention to detail when implementing a design. The exercises are presented in two different parts:

- The first one will focus purely on HTML/CSS stack, designing a responsive website that will contain static elements.
- The second part will focus on dynamic websites, implementing interactive elements in the website that was created in the first stage.

2. RESPONSIVE DESIGN

Objectives

Your first objective is to recreate the responsive layout of a dashboard that is shown in the *figure 1*. You will also find more detailed screenshots in the same folder as this document.

The main layout is composed by the following elements:

- Side navigation menu
- Toolbar on top.
- Grid with panels.

1

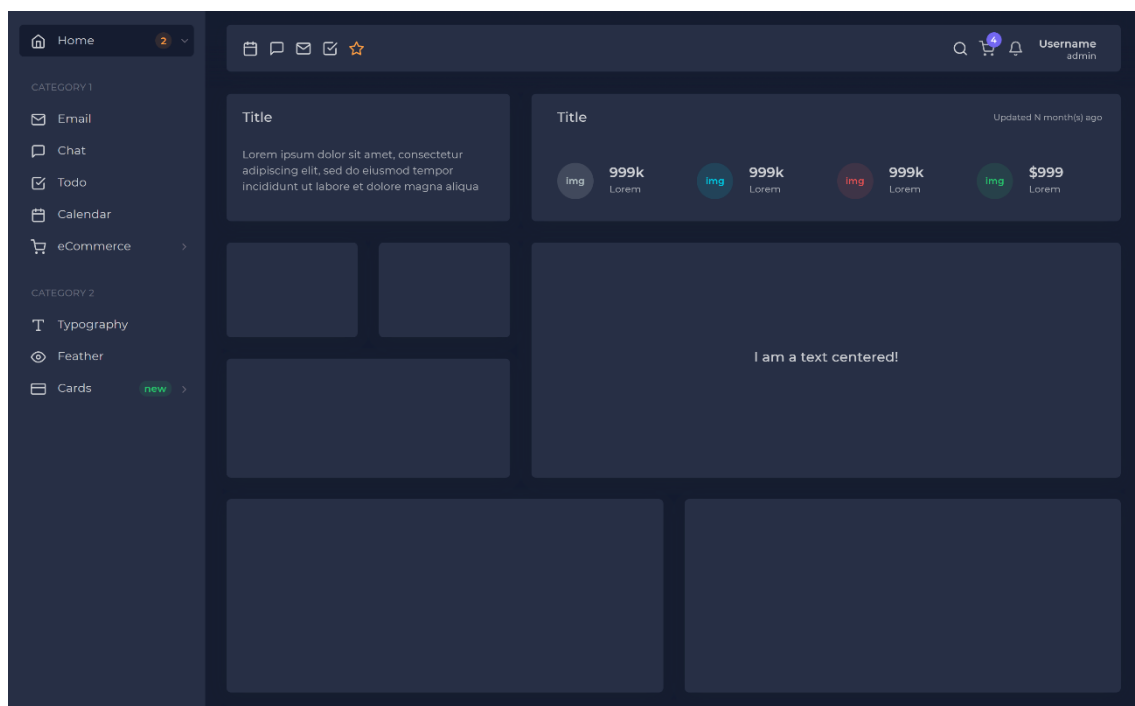


Figure 1 – Dashboard layout showing the desktop size.

The minimum composition must include the following:

- A side menu located on the left side of the web page, containing the following items listed from top to bottom:
 - Drop down menu to choose between different dashboards (note that it does not need to be functional).
 - This is the Home menu on the top left in the example
 - Category 1 containing some items.
 - Category 2 containing more items.
- A top bar that includes:
 - Quick bar buttons on the left.
 - Search, shopping cart and alerts icons on the right side.
 - Username and privileges of the user on the most right.
- Grid panels on the body:
 - One has a header and body text.
 - One has text aligned in the center vertically and horizontally.
 - (Optional) One will show a title and some items inside.

See the reference section at the end of the document to find color and icon definitions.

Responsive

2

The static webpage should be responsive for 3 different screen sizes:

- Desktop size.
- Tablet size (as shown in figure 2).
- Mobile size (as shown in figure 3).

The only difference between these 3 modes is the following:

- The side menu will be hidden when the size is not a desktop size.
- The quick buttons on the left side of the top bar will be hidden too.
- Instead, a menu button is placed (it does not need to have any functionality).
- The username will disappear when showing the webpage for mobile size displays.

The rest of the elements listed will remain the same, except for the distribution of the panels in the grid area.

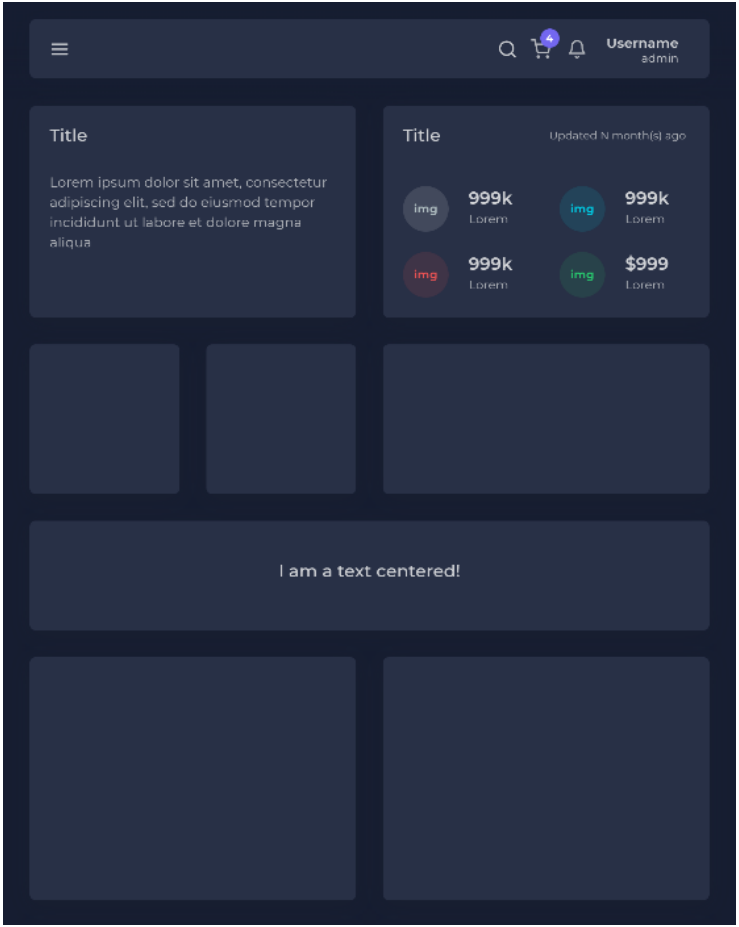


Figure 2 - Tablet size.

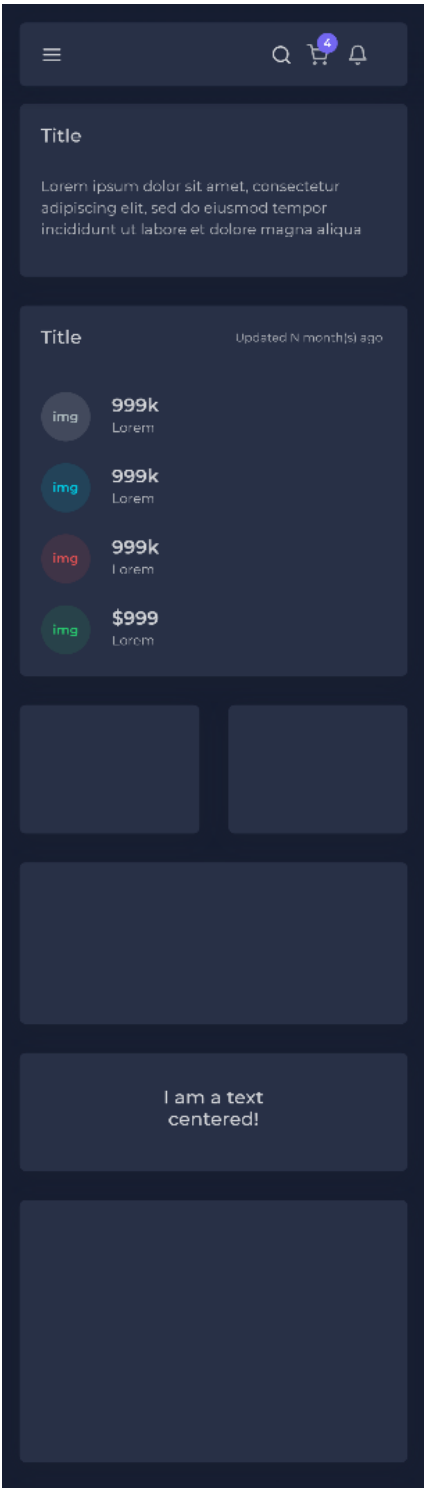


Figure 3 – Mobile size.

3. DYNAMIC WEBSITE - SHOPPING CART

Objectives

Once the static website is done, the next step is to add some interaction.

Using the framework that makes you feel more comfortable, implement a popup in the shopping cart that allows to perform the following actions:

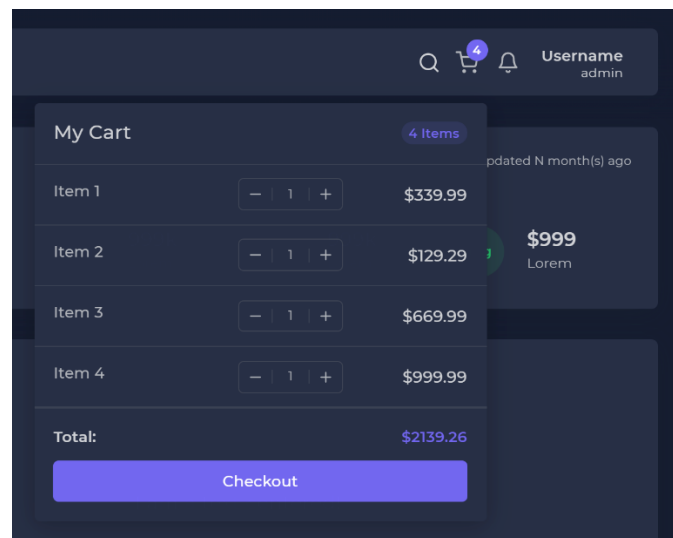


Figure 4 - Shopping cart popup.

5

- List the items you have stored in the shopping cart (*see next section to know how to generate the items*).
- Include a numeric up/down input to store the items count.
- Refresh the price when the numeric up/down input is changed.
- Calculate the total price dynamically when one of the items changed.
- Include a way to remove items from the shopping cart. As a reference, see figure 5 and 6, where a cross icon appears when the mouse hovers the item.
- When the button Checkout is pressed, remove all the items from the shopping cart, and throw a popup (or an alert) to inform the user that the transaction finished successfully.
- Also pay attention to the violet circle on the shopping card icon. It should show the number of items. If there is no item in the cart, the circle should not be shown.

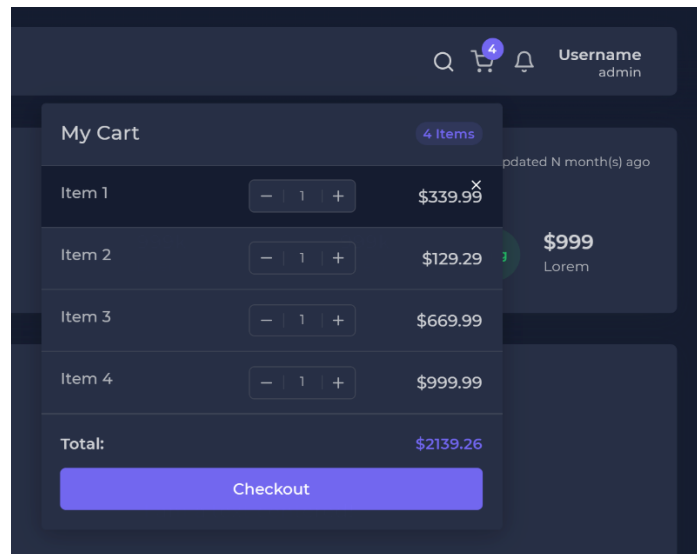


Figure 5 - Delete menu when the mouse hovers the item.

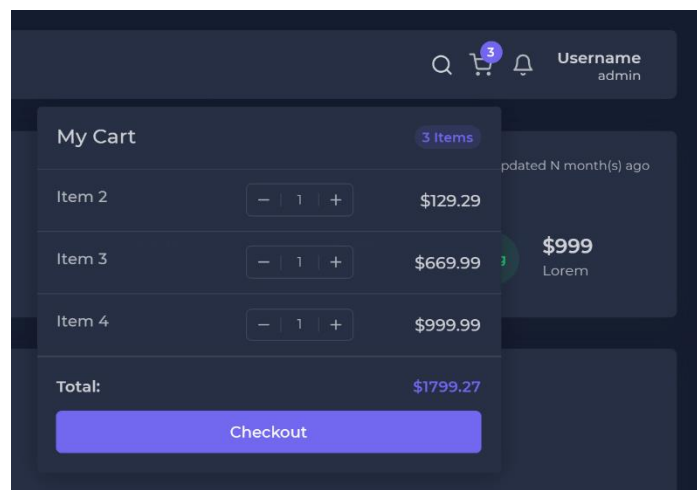


Figure 6 – Item 1 was deleted, showing the total price updated accordingly.

Shopping cart items

A JSON file is provided, and is located in the same folder as this document and detailed in figure 7 as well. The file contains the details of the items of the shopping cart.

Implement a service that simulates a GET request to the server, but instead of performing the GET, returns the data described in the JSON file.

```
{
  "shopping_cart_items": [
    {
      "name": "item 1",
      "count": 1,
      "price": 339.99
    },
    {
      "name": "item 2",
      "count": 1,
      "price": 129.29
    },
    {
      "name": "item 3",
      "count": 1,
      "price": 669.99
    },
    {
      "name": "item 4",
      "count": 1,
      "price": 999.99
    }
  ]
}
```














Figure 7 - JSON file with shopping cart items.

4. REFERENCES

In this subsection you will find references that you can use to speed up the implementation. It contains a list of icons and colors used.

Icons

All the icons are from the Feather icons collection, but they can be easily implemented using <https://iconify.design>.

	feather-check-square		feather-eye
	feather-mail		feather-credit-card
	feather-message-square		feather-search
	feather-calendar		feather-bell
	feather-star		feather-home
	feather-shopping-cart		feather-x
	feather-type		


8


Instructions on how to use them can be found on the page of any of the icons.


Colors


 Background: #161d31


 Foreground: #283046

 Shopping cart counter background, checkout button and item count: #7367f0

 Shopping cart – mouse hover on items: #161D31

 Text: #d0d2d6

 Icons: #b4b7bd

 Star icon and item count text in the dropdown menu: #ff9f43

Item count text background: rgba (255, 159, 67, .12)

 Circle 1 – Text: #b2c0c3

Circle 1 – Background: `rgb(255, 255, 255, .12)`



Circle 2 – Text: `#00cfe8`

Circle 2 – Background: `rgba(0, 207, 232, .12)`



Circle 3 – Text: `#ea5455`

Circle 3 – Background: `rgba(234, 84, 85, .12)`



Circle 4 and “new” – Text: `#28c76f`

Circle 4 and “new” – Background: `rgba(40, 199, 111, .12)`