Web Laboratory works

### **1. Simple landing page**

**Description**: Create a landing page following the given design (Figma).

**Requirements**:

* Use of semantic tags (<header>, <nav>, <ul>, <footer>, etc.) wherever is necessary
* All fonts, colour palette, elements sizes must strictly **match the design.**
* **Pixel perfect** is also **not required,** BUT your website should visually match the design template (i.e. If the element is horizontally centred on the design, it should look centred on the website)
* All **class names** must not be meaningless (i.e. class=”myClass”) and should follow the same **naming convention**, preferably BEM, but you could come up with your own (just be consistent)
* ***Basic cross-browser support*** *(website should look correctly on last versions of modern browsers (Chrome, Safari, Edge)*
* For now, **responsiveness** is not required.

**P.S** If your semester project has a landing website that **satisfies the requirements**, then you could skip this work by bringing your website for an assessment.

Note: The above goes only for those students. who were **actually responsible for the website development**. The whole team cannot use the right to skip this work!

A website example (from Live coding): <https://github.com/NazarGorokhivskiy/LandingForLiveCoding>

**LIVE CODING LINK (YouTube video)**

<https://youtu.be/-X-XTeDaCmo>

|  |  |
| --- | --- |
| **Variant** | **Design template link** |
| **1** | <https://www.figma.com/file/CKE7rSZvwWAckhsFpQLfZW/LabWork1-V1> |
| **2** | <https://www.figma.com/file/XHYmp9yKTxcOQD5zFAbKQH/LabWork1-V2> |
| **3** | <https://www.figma.com/file/vwiwu6GnDphlI25vncvl3F/LabWork1-V3> |
| **4** | <https://www.figma.com/file/pRfvOqsPgu7HrlMPgLKkto/LabWork1-V4> |
| **5** | <https://www.figma.com/file/bW1PzYlXgABwctUobkRAyA/LabWork1-V5> |
| **6** | <https://www.figma.com/file/wmMTQTdBymNHxX8ew0mG45/LabWork1-V6> |
| **7** | <https://www.figma.com/file/iJD99BqtY8vzmZdrXbZOCc/LabWork1-V7> |
| **8** | <https://www.figma.com/file/fJKhikm2WvdbmetEvibfhN/LabWork1-V8> |
| **9** | <https://www.figma.com/file/XT2jhcwmHBGxAjIBJsMnH0/LabWork1-V9> |
| **10** | <https://www.figma.com/file/ZM6o4UFoZUq9Kup8NjV94p/LabWork1-V10> |

### **2. Advanced landing**

**Description**: Improve a landing page following the given design (same as previous) by adding full responsiveness and some animation.

**Requirements**:

* The website should be **partly** **responsive**:   
  Required: for **320px** (iPhone 5s) **2560px** (4K screens).   
  ***Hint:*** *for extra-large screen width, you can just fit website content inside a previously set container for the regular screen width and centre it horizontally.*
* The header should collapse to a **hamburger** on small screens
* The website must contain **animations** (**at least 3**). Can be made in any possible way, preferably with css properties (*animation/transition*).
* Your project should be **logically structured** (all your CSS can’t be in just one-two files)
* All of the **previous** work **requirements** must be kept.

**P.S** If your semester project has a landing website that **satisfies the requirements**, then you could skip this work by bringing your website for an assessment.

Note: The above goes only for those students, who were **actually responsible for the website development**. The whole team cannot use the right to skip this work!

A website example (from Live Coding): <https://github.com/NazarGorokhivskiy/LandingForLiveCoding>

**LIVE CODING LINK (YouTube video)**

<https://youtu.be/-xBxSv51TAw>

### **3. CRUD Javascript App: View Page**

**Description**: In this work, you have to make a simple presentation part of a website - **View Page** (You can find the template of the page by following the link - [**https://wireframepro.mockflow.com/view/lviv-iot-crud-js-app**](https://wireframepro.mockflow.com/view/lviv-iot-crud-js-app#/page/2b044d1d26084553827b6c846eb1db59)For your blocks you must use **data** from your java/python project class.

In case you don’t have a **data** from previous year, you should choose any free task from this link:

<https://docs.google.com/document/d/1RW9PpalOlHn-nVIi8kbCr71vu_XLNsjj4TKpTdBnZ-w/edit?usp=sharing>

Then using JavaScript, you need to implement the following operations on your data *(it is up to you to decide which field should be used for each of the operations)*:

* **Sort** of your items option
* **Search option**
* **Count total amount** of some of the field (e.g total price of all cars)

**Requirements:**

* **Responsiveness** absolutely not required.
* **Styling** is not important at all. Is up to you.

**Our recommendations and tips:**

* use JS Array methods: map(), sort(), filter(), reduce()
* use native JS for any DOM operations (querySelector | findById | insertAdjacentHTML | etc)
* a website [**example**](https://github.com/NazarGorokhivskiy/PureJSDragNDrop) from live coding
* Working with DOM & JS Array methods project  
  <https://github.com/bradtraversy/vanillawebprojects/tree/master/dom-array-methods>
* useful projects [**examples**](https://github.com/bradtraversy/vanillawebprojects)

**LIVE CODING LINK (YouTube video)**

<https://youtu.be/E4qeqHceD-E>

### **4. CRUD Javascript App: Create/Edit Pages**

**Description:** In this work, you need to continue working and add two new parts to your website **- Create & Edit Pages** (You can find the template of these pages by following the [**link**](https://wireframepro.mockflow.com/view/lviv-iot-crud-js-app#/page/2b044d1d26084553827b6c846eb1db59)**)**

**Update/Delete operations are not required for this work!**

Also you must validate the forms using HTML **attributes** (inputs must be configured for your data format).

If incorrect data is entered in the inputs, you must use JavaScript to inform the user with **modal windows** or just plain alert() function.  
  
**Bonus** points are provided for this work, if you implement a styled modal window that will work with JavaScript.

**Requirements:**

* **Responsiveness** absolutely not required.
* **Styling** is not important at all. Is up to you.

**Our recommendations and tips:**

* a website [example](https://github.com/NazarGorokhivskiy/PureJSDragNDrop) from live coding

**LIVE CODING LINK (YouTube video)**

<https://youtu.be/E4qeqHceD-E>

### **5. CRUD Javascript App: Backend**

**Description**: In the last part of working on the website you have to implement all Create/Read/Update/Delete operations which must be made via the corresponding HTTP methods in your REST API.  
  
**Important**. You don't have to make a backend from the beginning - connect an existing one that you worked on in the first year.

If you don’t have a backend server, you should create a new REST API using any preferred technology.

**Our recommendations and tips:**

* [Working with REST API with fetch() link](https://learn.javascript.ru/fetch#post-zaprosy) (POST method example)
* a website [example](https://github.com/NazarGorokhivskiy/PureJSDragNDrop) from live coding

**LIVE CODING LINK (YouTube video)**

<https://youtu.be/jaRmBtmP2TA>

### 