**Environment**

Welcome to the pre-certification practice task! This task is designed to be a playground that lets you get acquainted with the testing environment and practice the platform's functionalities before your certification.

Here are a few tips to help you get started in the CodeSignal IDE:

* Before clicking the "Run tests" button for the first time, please make sure all the libraries are installed and the application has started in the preview pane.
* If you would like to work in a different front-end framework, you can click theicon above the code window to change the current environment. Currently available are Angular, React and Vue. Note that this will **reset your environment** and all changes will be lost.
* If the web server stops for any reason, you can restart it by running this command in the Terminal: **npm start**

**IDE**

To customize the editor settings and see the editor hotkeys, check out the **Settings** tabon the left sidebar.

See the **README** tab for more information about the environment and test run verdicts.

The IDE will automatically install the development environment and required packages from **package.json**. Feel free to modify the **package.json** file to include additional packages as needed. You can use the console to stop or restart your application whenever you want. The live preview pane will update as you work to reflect the current state of your application. If you close the console and want to restart the application, run the command **npm start**.

If needed, you can hard reset the environment by clicking on the circular reset code icon () on the top right of the IDE. Note that your progress will not be saved, so please be careful when using this.

**Prettier**

Please note that **prettier** is already included in **package.json**. For formatting the code, run the command **npm run prettier** in the console.

**Debugging**

If you add debugging output to the console in your components, please check your browser's debug console for output messages.

Feel free to browse the project workspace. All necessary component files have been created for you.

**Requirements**

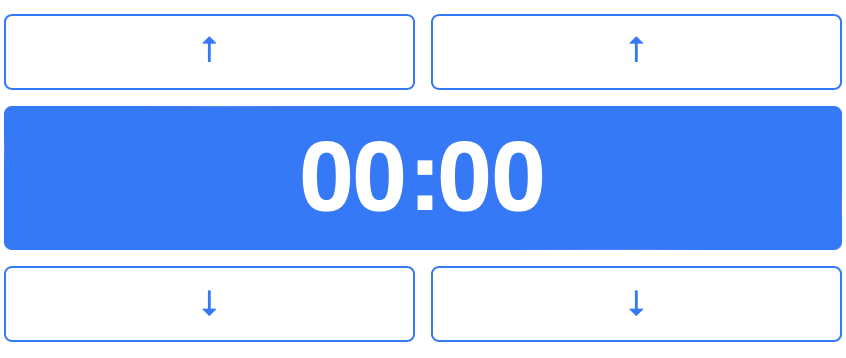
Your task is to implement a page with a clock. Plan your design according to the level specifications below (the current level is in bold):

* **Level 1: The clock should be rendered with all the functional elements required (buttons, clock blocks, etc.).**
* Level 2: The rendered clock should support increasing and decreasing the hours and minutes.

To move to the next level, you need to pass all the tests at this level when submitting the solution.

**Level 1**

You are given a page with a partial implementation of the clock. Your task is to render the missing buttons. The result should look as follows:



This is the HTML template (note that it's pure HTML, and you might need to convert it to your framework's template syntax):

**<div id="ClockUpdater" class="container">**

**<div class="row">**

**<button id="hours-up-button">&uarr;</button>**

**<button id="minutes-up-button">&uarr;</button>**

**</div>**

**<div class="row">**

**<div id="clock"></div>**

**</div>**

**<div class="row">**

**<button id="hours-down-button">&darr;</button>**

**<button id="minutes-down-button">&darr;</button>**

**</div>**

**</div>**

**Note:** HTML element ids are used for the testing. Please, make sure you are using the correct ones. Otherwise, your test won't run correctly.

**Tests**

* Unit tests are provided in the **test/level<i>.test.js** files for each level. To run the tests, click the blue button 'Run'. You can choose to run the tests **In Terminal** or in the **Structured** manner.
* You may use **test/sample.test.js** to write your own tests, which will also be included in the test runs.
  + If you would like to include debugging output to the console in your tests, use the **In Terminal** option to receive the raw test output.
  + Note, that debugging output from your application code you can find in your browser's console.

When working on a scored certification, partial credit will be granted for each unit test passed, so press **Submit** often to run tests and receive partial credits for passed tests.

* **[execution time limit] 55 seconds**
* Please note that if the question needs to run Front-End unit tests, a headless browser might be initialized for every test run, taking an average time of ~20 seconds.
* **[memory limit] 4g**