Please make as per wireframe shared.

You are expected to make only first 4 Pages, it includes dashboard, login, Signup pages.

You can use Javascript for functionality.

Wireframe Wireframe : <https://xd.adobe.com/view/0f53194f-36d1-42b6-87de-d0fbe6eb05b1-5a00/>

Test Cases-

1) Store the User Registration data in Json file

2) Click on Login-> Validate the user data from Json for login

3) If User found in Json file then Redirect to the home page

**Coding Standards:**

**1. Declare a DOCTYPE**

The DOCTYPE declaration should be in the first line of HTML. Actually, it activates the standard mode in all browser. It is recommended that you use the HTML doctype:

**2. Close your tag**

Leaving some tags open is simply a bad practice. Only self-closing tags (  
,

,,,, etc)are valid. Normal elements can never have self-closing tags.

**3. Explain which div you are closing**

If we view most of the website source at the very bottom of the page an almost endless list of closing

tags. Without proper code organization, it can be messy. Using indentation and comment for every div is a good practice.

…

<! – #header →

**4. Use a CSS reset**

CSS reset will ensure that all elements have no particular style so you can define your own.

html, body, div, span, applet, object, iframe,

h1, h2, h3, h4, h5, h6, p, blockquote, pre,

a, abbr, acronym, address, big, cite, code,

del, dfn, em, font, img, ins, kbd, q, s, samp,

small, strike, strong, sub, sup, tt, var,

b, u, i, center,

dl, dt, dd, ol, ul, li,

fieldset, form, label, legend,

table, caption, tbody, tfoot, thead, tr, th, td {

margin: 0;

padding: 0;

border: 0;

outline: 0;

font-size: 100%;

vertical-align: baseline;

}

**5. Never use inline styles**

When creating your markup, do not use inline styling because it would be very hard to override these styles in case you need to.

**6. Include external CSS inside the HEAD tag**

Style sheets can be placed anywhere but the HTML specification recommends that they be placed within the document HEAD tag. The primary benefit is that pages will load faster.

**7. Consider placing JavaScript at the bottom**

When loading a script, the browser cannot continue until the entire file has been loaded. If we have JavaScript files in order to add functionality, we should place those files at the bottom, just before the closing body tag. This is a good performance practice and the results are quite noticeable.

**8. Use lowercase in your tags**

It is a good practice to keep markup lower-case. The capitalizing markup will work and will probably not affect how your web pages are rendered, but it does affect code readability. We should keep it simple.

**9. Keep the syntax organized**

When pages will grow, managing HTML can be hard. There are some quick rules that can help us to keep our syntax clean and organized. These include the following:

* Indent nested elements
* Use double quotes, not single or completely omitted quotes
* Omit the values on Boolean attributes

**10. Use practical ID and classes names and values**

We should only give elements an ID attribute if they are unique. Classes can be applied to multiple elements that share the same style properties. It is always preferable to name something ID or class, by the nature of what it is rather than by what it looks like.Class names should be all lowercase and use hyphen delimiters.

**11. Organize CSS with comments**

Let’s keep our styles organized in logical groups and provide a comment noting what the following styles pertain to. Otherwise we will regret it later.

**12. Write CSS using multiple lines and spaces**

It is important to place each selector and declaration on a new line. That will make the code easy to read and edit.

**vs13. Use shorthand**

One feature of CSS is the ability to use shorthand properties and values. Using that allows us to quickly set and identify styles. But when we’re only setting one value, shorthand alternatives should not be used, because it will be hard to identify which CSS attr is being applied.

**14. Avoid units on zero values**

One way to easily cut down on the amount of CSS we write is to remove the unit from any zero value. A zero will always be a zero.

**15. Modularize styles for reuse**

CSS is built to allow styles to be reused, specifically with the use of classes. For this reason, styles assigned to a class should be modular and available to share across elements as necessary.

**16. Use multiple stylesheets, but be aware of them expanding beyond control**

Depending on the complexity of the design and the size of the site, sometimes it’s easier to make smaller, multiple stylesheets instead of a giant one.

**17. Use conditional comments**

Sometimes some clients want even more by requiring to create web pages which are compatible with this obsolete browser. To target specific versions of IE, we can use the well known IE hacks, as shown below:

height: 200px; /\* normal browsers \*/

\_height: 300px; /\* IE6 \*/

.height: 250px; /\* IE7 \*/

\*height: 350px; /\* All IEs \*/

Those hacks are extremely useful sometimes, but they are not the best way to target a specific version of IE, and it can cause CSS validation to fail.  
Instead, we can use the conditional comment shown below to target IE6.

**18. Checking in cross-browser while developing**

One of the biggest mistake most of the beginners made when developing HTML, CSS, and JavaScript, was not to test pages on multiple browsers while developing them. Generally, we used to write all code and just view in Firefox to see how it was rendered.

In theory, this should be good. But cross-browser issues are a major problem for front-end developers, especially due to IE. If we test our documents on Firefox/IE/Chrome while writing it, cross-browser rendering problems will be much easier to fix.