MSIMBO

Designing the User Interface

# Description

This is the first of several assignments that will guide you through the creation of a Website that you will be working on throughout the course. The skills you practice on the assignments are meant to be used on a project of your own. Commit your work frequently throughout the day and push your changes at the end of the day to deploy and restart your server. When you are ready to submit this assignment, ***tag*** your last commit you wish to be graded on as ***v1.0*** on GitHub. You can keep working and committing and pushing. The instructor can always view your assignment using the ***v1.0*** tag.

The Website you will be building throughout the assignments will allow users to create and use online, mobile friendly, Websites. We will be building a Website to create Websites. There are two types of users. We will refer to those users creating the Websites as ***developers***. Users that visit the Websites created by the developers will be referred to as ***end users***. Only developers can create and modify the Websites. End users can not modify the Websites, they can only interact with them. The Website will be built using the MEAN stack using the four underpinning technologies MongoDB, Express, React, and Node.js.

In this assignment you will get started creating the Website by first focusing on creating a set of static pages that will serve as a prototype. The prototype will allow evaluating user interface aspects of the application such as navigation, information layout, page dependencies, modalities, and authentication. Although the pages will have links, forms and buttons to interact with, the Website won't actually do anything yet. It will only be an empty shell for now. We will add the functionality as we progress through the rest of the assignments. All pages and content must be responsive and mobile friendly.

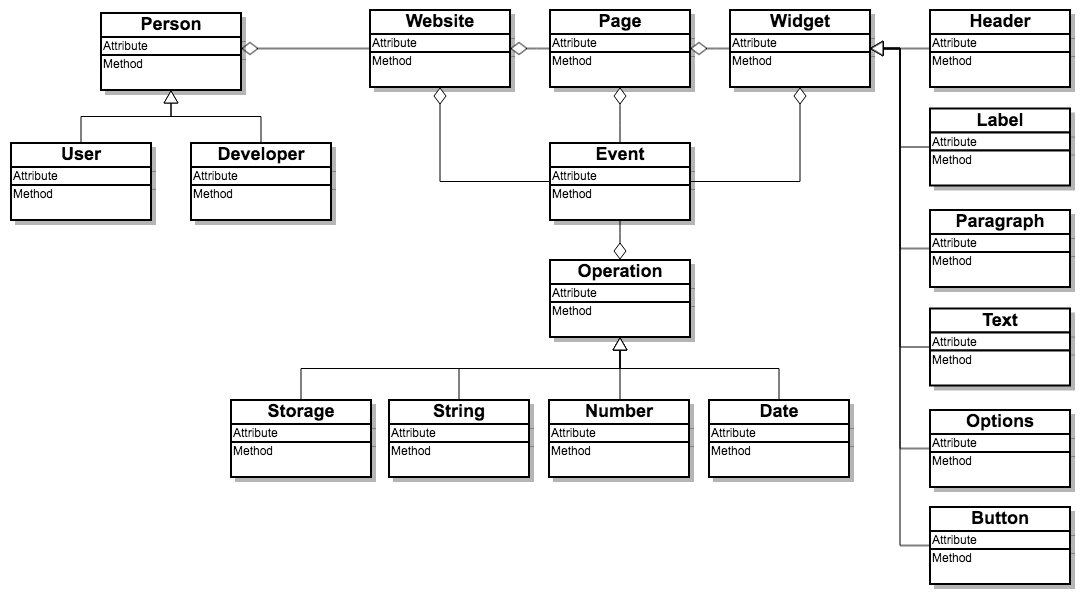
## Use Cases

The following use cases describe the functionality of the Web application in terms of a set of use cases

|  |  |
| --- | --- |
| Developer Use Cases  1. Developer registers 2. Developer logs in 3. Developer views their profile 4. Developer updates their profile 5. Developer logs out | Website Use Cases  1. Developer views a list of her websites 2. Developer creates a website 3. Developer updates a website 4. Developer deletes a website |
| Page Use Cases  1. Developer views a list of pages in a website 2. Developer adds a page to the website 3. Developer updates a page 4. Developer deletes a page | Widget Use Cases  1. Developer views a list of widgets in a page 2. Developer adds a widget to a page 3. Developer reorders widgets in a page 4. Developer updates a widget 5. Developer deletes a widget |

## Data Model

A data model describes the various types of objects that are used in a Website such as pages, widgets, and users. A ***type of object*** is also referred to as a ***class*** and the diagram below is called a ***class diagram***. Data models also capture the relationships between the different types of objects or classes. For instance the class diagram below states that a person might be related to, or have, several Websites, that a Website in turn can have many pages, a page can have several widgets.



## Wireframes

Given the wireframes discussed in class, implement the following HTML pages using HTML, CSS, and Bootstrap. The HTML pages should responsive and mobile first. The wireframes just show the functionality and layout of the page, the do not show the look and feel of the page. You can use the default look and feel provided by bootstrap stylesheet, but you can improve on the style. Apply the styles using bootstrap classes. The instructor will demonstrate how to use bootstrap styles and code samples to build the Web pages. These are the Web pages you will be creating for this assignment:

|  |  |
| --- | --- |
| User Pages  1. Login.html 2. Register.html 3. Profile.html | Website Pages  1. WebsiteList.html 2. WebsiteNew.html 3. WebsiteEdit.html |
| Page Pages  1. PageList.html 2. PageNew.html 3. PageEdit.html | Widget Pages  1. WidgetList.html 2. WidgetChoose.html 3. WidgetHeading.html 4. WidgetImage.html 5. WidgetYoutube.html |

## Creating the html pages

The first step will be to create static versions of the pages we will be working with. The static pages will live in the assets folder of the angular application. In the next assignment we will refactor these pages to be dynamic components. For now they will be static pages. Do all your work in the local repository you created for this course

## Viewport

Put these meta tags in your head, this will help to make your website mobile friendly.

***<!-- Meta tags for making website for mobile-->***

**<meta name="viewport" content="width=device-width,initial-scale=1.0">**

## User Pages

The pages created must look like

|  |  |  |
| --- | --- | --- |
| Login.html  login.png | Register.html  register.png | Profile.html  profile-portrait.png |

## 

## Website Pages

|  |  |
| --- | --- |
| WebsiteList.html Portrait  website-list-portrait.png | WebsiteList.html Landscape  website-list-landscape.png |
| WebsiteNew.html Portrait  website-new-portrait.png | website-new.html Landscape  website-new-landscape.png |
| website-edit.html Portrait  website-edit-portrait.png | website-edit.html Landscape  website-edit-landscape.png |

## 

## Page Pages

|  |  |  |
| --- | --- | --- |
| page-list.html Portrait  pageList.png | page-new.html Portrait  pageNew.png | page-edit.html Portrait  pageEdit.png |

## Widget Pages

|  |  |  |
| --- | --- | --- |
| widget-list.html  widget-list-1.png  Note: The image, paragraphs, and YouTube video shown above are only for illustration purposes. You can use your image, paragraphs, and YouTube video | widget-chooser.html  widget-choose.png | widget-heading.html  header.png |
| widget-image.html  image-edit.png | widget-youtube.html  youtube-edit.png |

## Website Navigation

Implement navigation as shown in the page flow diagram and table below. Ignore links and buttons not listed here. Other links and buttons will be addressed in subsequent assignments.

### Page Flow Diagram

### Page Flow Tableflow.png

|  |  |  |
| --- | --- | --- |
| **From Page** | **Action/Button/Link** | **To Page** |
| Login Page - login.html | Login | Profile Page - profile.html |
| Register | Register Page - register.html |
| Register Page - register.html | Register | Profile Page - profile.html |
| Cancel | Login Page - login.html |
| Profile Page - profile.html | Logout | Login Page - login.html |
| Websites | Website List Page - website-list.html |
| Ok (Check) | Profile Page - profile.html |
| Website List Page - website-list.html | New Website (Plus) | New Website Page - website-new.html |
| Edit Website (Cog) | Edit Website Page - website-edit.html |
| Website Name in List | Page List Page - page-list.html |
| Profile (Person)/  Back (Left Arrow) | Profile Page - profile.html |
| New Website Page - website-new.html | Ok (Check) | Website List Page - website-list.html |
| Cancel (Left Arrow) |
| Profile (Person) | Profile Page - profile.html |
| Edit Website Page - website-edit.html | Ok (Check) | Website List Page - website-list.html |
| Delete |
| Cancel (Left Arrow) |
| Profile (Person) | Profile Page - profile.html |
| Page List Page - page-list.html | Add Page (Plus) | New Page Page - page-new.html |
| Edit Page (Cog) | Edit Page Page - page-edit.html |
| Page Name in List | Widget List Page - widget-list.html |
| Back (Left Arrow) | Website List Page - website-list.html |
| Profile (Person) | Profile Page - profile.html |
| New Page Page - page-new.html | Ok (Check) | Page List Page - page-list.html |
| Cancel (Left Arrow) |
| Profile (Person) | Profile Page - profile.html |
| Edit Page Page - page-edit.html | Ok (Check) | Page List Page - page-list.html |
| Delete |
| Cancel (Left Arrow) |
| Profile (Person) | Profile Page - profile.html |
| Widget List - widget-list.html | Add Widget (Plus) | Choose Widget - widget-choose.html |
| Back (Left Arrow) | Page List - page list |
| Edit Header (Cog) | Edit Header Widget - widget-heading.html |
| Edit Image (Cog) | Edit Image Widget - widget-image.html |
| Edit YouTube (Cog) | Edit YouTube - widget-youtube.html |
| Profile (Person) | Profile page - profile.html |
| Choose Widget - widget-choose.html | Header | Edit Header Widget - widget-heading.html |
| Image | Edit Image Widget - widget-image.html |
| YouTube | Edit YouTube Widget - widget-youtube.html |
| Back (Left Arrow) | Widget List - widget-list.html |
| Edit Header Widget - widget-heading.html | Ok (Check) | Widget List - widget-list.html |
| Cancel (Left Arrow) |
| Delete |
| Edit Image Widget - widget-image.html | Ok (Check) | Widget List - widget-list.html |
| Cancel (Left Arrow) |
| Delete |
| Edit YouTube Widget - widget-youtube.html | Ok (Check) | Widget List - widget-list.html |
| Cancel (Left Arrow) |
| Delete |

## Styling

Implement styling as shown in the wireframes. Apply the styles where appropriate. Not all styles are applicable for this particular assignment. In particular, maker sure to

1. Use [Bootstrap](http://getbootstrap.com/) to style the website. Alternatively, you may use [Angular Material](https://material.angularjs.org), or Foundation
2. All **text** input fields must be of type **text**
3. All **password** input fields must be of type **password** to hide passwords from prying eyes
4. All **email** input fields must be of type **email** to support simple validation
5. All **date** input fields must be of type **date** to provide a simple date picker
6. All **textarea** input fields must be **at least 3 rows high** as shown in the wireframes. Text Areas should not have default white space
7. All **file** input fields must be of type **file** to allow browsing files
8. Configure **input field default text** or values as shown in the wireframes
9. All input fields must use Bootstrap's **form-control** class or equivalent
10. All input fields must have a **placeholder** that describes the type of input. Use the placeholders shown in the wireframes
11. **Buttons and links** that look like buttons, must use bootstrap classes **btn** and **btn-block** classes to render them as buttons where appropriate. Use other button related bootstrap classes to color the buttons as shown in the wireframe, e.g., **btn-primary**, **btn-danger**, **btn-success**
12. **Tables must be styled** with bootstrap classes **table** and **table-responsive** to ensure they are responsive. Use other table classes as appropriate
13. All links should be styled with a shade of blue, but no underline when hovering
14. Use **FontAwesome** to illustrate actions such as ok, cancel, back, done, add, remove, config, as shown in the wireframe.Make proper use of **whitespace**, e.g., use padding, margins, wrapping and text justification to style the content as shown in the wireframes
15. Pages should not allow **pinching-to-zoom** on mobile devices, unless otherwise stated
16. Headers and footers should be **statically positioned** at the top and bottom respectively. They should not scroll with the rest of the content
17. All **CSS and JavaScript libraries** must use CDN and declared in the **meta** element. Alternatively, libraries can be declared at the bottom of the **body** element for performance considerations
18. Pages must **not use style element or attributes**. All additional custom styling must be done in a separate stylesheet file, e.g., **public/assignment/css/styles.css**
19. Custom stylesheets overriding CSS library styles, must be declared after the library styles they override
20. Pages must **not use inline JavaScript**. All additional custom scripting should be done in a separate JavaScript file, e.g., **public/assignment/js/app.js**

# Deliverables

## GitHub Deliverables

To allow instructor to see your changes, please frequently commit and push your work to GitHub repository. Below is an example of the commands you will use.

**> git add .**

**> git commit -m 'A comment describing your work'**

**> git push**

### Tagging a Release

We will be using code repository tags (or releases) to "submit" assignments. When you consider your work complete and ready for evaluation (ready for release), go to your code repository in GitHub and generate a release by navigating to "releases". Then click on "Create a new release" and type the name of the tag in the input field labeled "Tag version". We will be using the following tags for the various assignments:

**v1.0(this assignment)**

If you need to resubmit the assignment then create a new tag by adding a version number, e.g.,

v1.1, v1.2, etc...

I will grade the very last release. The date/time you create the tag will be considered the date/time of submission.