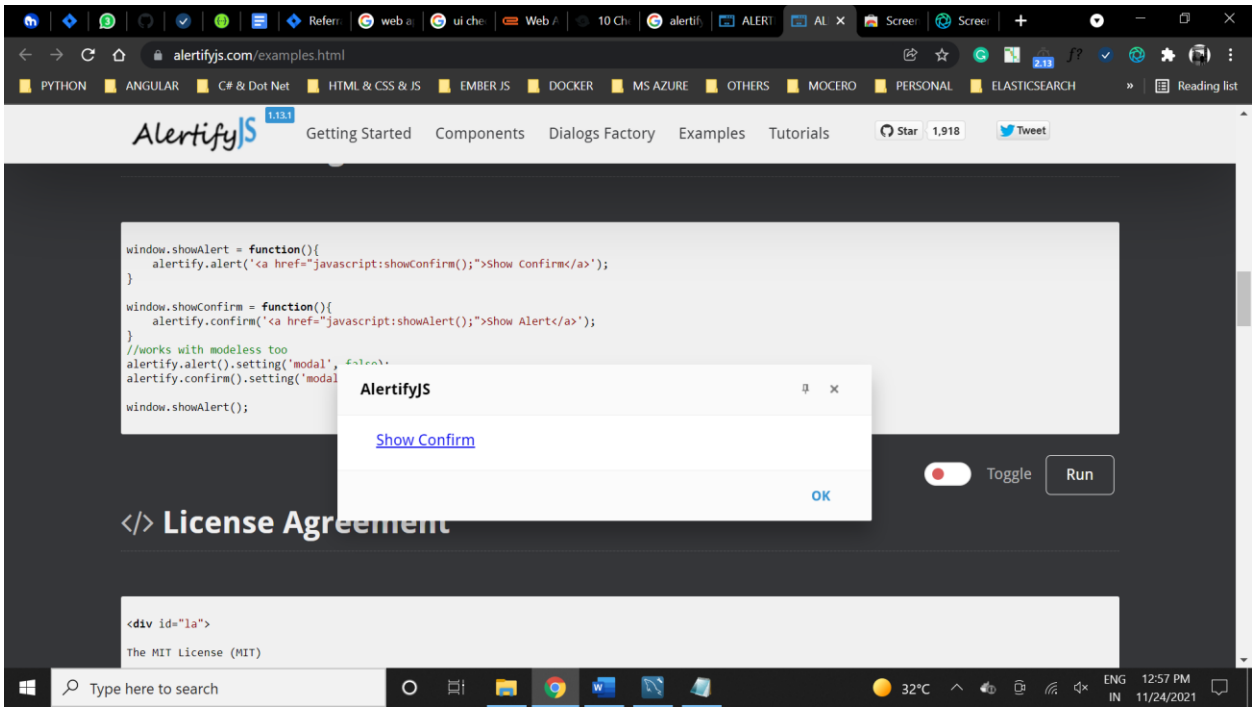
Web Application Development and Testing Check List

Steps:

1. Frontend Check List:
    a. Fix broken links
    b. Spelling and grammar
    c. Check website in all web browser
    d. Check page speed
    e. Comment needed
    f. Use optimizes images
    g. HTML5/CSS3 compatibility check
    h. Custom 404 page
    i. Page URL – valid name
    j. Mobile browser responsiveness
    k. Validation needed for all input boxes
    l. Color contrast
    m. Use reusable style classes
    n. Label name should be meaningful
    o. Check if the horizontal and vertical scrollbars are appearing wherever necessary
    p. Check if autofill function is working as expected and that there are no UI issues
    q. Confirmation on successful action on each step completion (use proper alert msg not default alert pop up)



r. Help/Suggestions
   - Description
   - Max length limit

s. Handle the error messages and notify proper message.
t. Reuse the JavaScript functions.
u. Avoid declaring more variables.
v. Use proper data types for sending request or validation
w. In CRUD operation, use binding function call for all the operation.
x. In JavaScript page, every function top comment the function description and mandatory fields.
   Example:

```
// Create a "close" button and append it to each list item
var myNodelist = document.getElementsByTagName("LI");
var i;
for (i = 0; i < myNodelist.length; i++) {
  var span = document.createElement("SPAN");
  var txt = document.createTextNode("\u00D7");
  span.className = "close";
  span.appendChild(txt);
  myNodelist[i].appendChild(span);
}
```

y. Use plugin script file instead of calling direct URL.
z. Avoid using inline style class.
aa. Avoid unwanted code.
bb. All file name should be lowercase.

2. Middleware Check List:

a. Use proper MVC controller name and reuse the controller for same module.
b. JWT integration needed.
c. Use log4net for saving error messages.
d. Modal validation needed.
e. In all MVC controller, comment needed.
f. Use server-side input validation controls to constrain inputs.
g. Validate length, range, format and type for every input to system.
h. Use strong data typing.
i. Defined session timeout properly.

3. API Development Check List:

a. Use proper API controller name and method name.

b. Authentication needed.
c. In all API controller, comment needed.
d. Avoid creating multiple controllers for same module. Reuse the existing controller.
    i. One module – One Controller and if you want create multiple methods.
e. Use proper payloads and API methods (Get, POST)
f. Modal validation needed.
g. Use log4net for saving every request and error messages both API Controller and Data Access layer.
h. Use proper return data. (Data table or Dataset)
i. Avoid creating multiple entity class, to reuse the existing class.
j. API Response should be unique for all API request. (Both web and mobile) Example:

- ```
  {
   "status": true,
   "data": {
    /* Application-specific data would go here. */
   },
   "message": “success” /* Or optional success message */
  }
  ```
- ```
  {
   "status": false,
   "data": null, /* or optional error payload */
   "message": "Error xyz has occurred"
  }
  ```

4. Backend Check List:

a. Use proper table name, store procedure name and function name.
b. Constraints are needed for each table. (Foreign key, Not Null, Unique Key and etc.)
c. In every table have one unique primary key. (Integer data type)
d. Avoid varchar data type as a primary key.
e. In all table have, created by, updated by, created on and updated on fields.
f. In every query to be executed less than one second.
g. Use view for large query.
h. Use index if you have big table like Medicine Prescription.

5. Testing Check List:

1. **Title & Metadata**
2. **Favicon**

3. **Proofread** - Read everything on the page and get it read from someone else as well. Even if you've already read it. There is always something you will identify and have to change. Check if you can reduce the amount of text by keeping it more specific. Break down large text blocks into smaller paragraphs. Add individual headings throughout the content, and use lists so that users can scan through the text easily. Don't forget to check validation content such as alert boxes, inline messages, etc
4. **Cross Browser Checks -** There is a usual case with web developers that the site looks great in chrome but it is completely broken in IE. Well, It is important that your website works across browsers. It doesn't have to be pixel perfect, but everything should ideally work, and the user should not face any problems with usability and visibility of the website. The most popular browsers to check are Internet Explorer 8 and above, Firefox 3+, Safari 3+, Chrome, Opera and the iPhone
5. **Link -** Please don't assume that all your links will work. Click on them and check them. You may often forget to add "http://" for links to external websites. Also make sure your logo links to the home page. make sure external links open in a new tab/window and the links should be easily identifiable when compared to the rest of the text on the page
6. **Usability/Functionality Checks -** Test everything in-depth. Get others to test your website especially your target market. Sit back and watch how he/she uses the website. It's amazing what you'll pick up on when others use your website differently than how you assume they'd use it. Common things to check for are contact forms, search functions, shopping baskets and log-in areas
7. **Validation -** You should aim for a 100% valid website. This doesn't mean that things will not work if it isn't, but it's important to know the reasons why it does not so that you can fix any critical errors. Common issues include no "alt" tags, no closing tags and using "&" instead of "&" for ampersands. There are various CSS and HTML validators available online to validate your website
8. **RSS -** If your website has a blog or a newsreel, you should have an RSS feed that users can subscribe to. Users should be able to easily find your RSS feed: the common convention is to put a small RSS icon in the browser's address bar. Put this code between your <head> tags
9. **Error Handling/Defensive Design -** The most commonly overlooked defensive design element is the 404 page. If a user requests a page that doesn't exist, your **404 page** is displayed. This may happen for a variety of reasons, including another website linking to a page that doesn't exist. Get your users back on track by providing a useful 404 page that directs them to the home page or suggests other pages they may be interested in. Another defensive design technique is **checking your forms for validation**. Try submitting unusual information in your form fields (e.g. lots of characters, letters in number fields, etc.) and make sure that if there is an error, the user is provided with enough feedback to be able to fix the issue
10. **Optimization -** Like every other developer you'll want to configure your website for **optimal performance**. You should do this on an ongoing basis after launch, but you can take a few simple steps before launch, too. Reducing HTTP requests, using CSS sprites wherever possible, optimizing images for the Web, compressing JavaScript and CSS files and so on can all help load your pages more quickly and use less server resources

6. Testing Basic Check List:

Basic Testing Checklist		
Type of Field	Check For	Pass / Fail
1. Text Field	1. Minimum Input Length	
	2. Maximum Input Length	
	3. Accepts Special Characters	
	4. Accepts Numeric/ Alpha-numeric/ Character Input respect to type of field	
	5. Can user Paste into the field	
	6. Can user can Copy the field.	
	7. Can Delete / Edit the Text	
2. Numeric	1. Accepts Positive Input	
	2. Accepts Negative Input Values	
	3. Accepts Alphabets Character Input	
	4. Accepts Special Symbols	
	5. Max input Length	
	6. Min input Length	
3. Date Field	1. Is valid Date Pre-format shown	
	2. Can user manually enter date	
	3. Does the field accepts invalid format for the date	
	4. Does the software handles the invalid input into the field properly	
	5. Can User copy paste into the date field	
	6. Is calendar shown to user to select the date	
	7. Can user edit/ delete the selected date	
4. List (Drop Down)	1. Is user able to view all the entities in the list	
	2. Is user able to scroll down the list	
	3. Is user able to scroll up the list	
	4. Is user able to select the entity from the list	
	5. Is user able to deselect the entity from the list	

	6. Is user able to delete the text after selection	
	7. Is matching text (auto – suggestion) shown to user when the user types into the field	
	8. Is user able to select the matching text	
	9. Can user Copy / Paste into the field	
	10. Can user Copy / Paste from the field	
	11. User Select / Deselect Multiple List Options	
5. Search Box	1. Can user type into the search box	
	2. Is user shown matching text (auto – Suggestion) on typing into search field	
	3. Is user able to select the matching text shown	
	4. Is user able to Copy/ Paste into the field	
	5. Is user able to delete the text from search box	
	6. Does the field handles special character input properly	
	7. Does the field handles invalid data properly	
	8. Matching Result should be shown correctly	
6. Checkbox	1. Is user able to select the checkbox	
	2. Is user able to deselect the checkbox	
7. Button	1. Is the user able to click on the button	
	2. Is the user able to navigate to specified path on clicking on the button options	
	3. Is the user able to right click/ left click on the button	
	4. Label Check for all Buttons	
	5. Button Alignment	
8. Computed Field (Sum/Calculated)	1. Is the value computed/Calculated right	
	2. Is user able to edit the computed value	
	3. Is user able to delete the computed value	
	4. Is the user able to copy / paste into the field	
9. File Upload	1. Is the File Explorer Window opening up on clicking the File Name button	
	2. Is the user able to view all the files in the file explorer window	

	3. Is the user able to select the file	
	4. Is the user able to upload the file	
	5. Is the uploaded file viewable	
	6. Does the software handles Invalid file format uploaded properly	