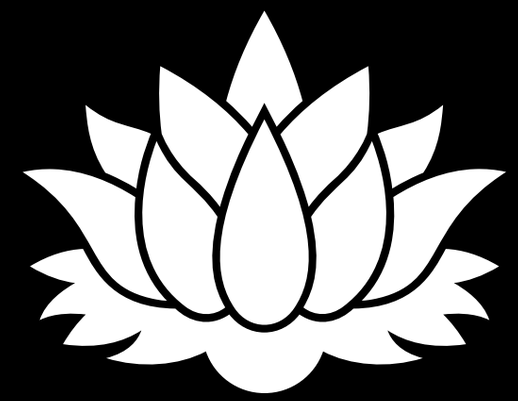
Team Lotus

Programming Manual



Authors

Sailesh Rajanala

- Team Lead
- UI/UX Developer (Front End & Back End)

Priyanka Limbu

- Timekeeper & Project Management
- Front End Developer

Subash Acharya

- Scribe
- Back End Developer

Shiva Karki

- Facilitator
- Front End Developer

Table of Contents

Prerequisites

Before we begin understanding various tools that we can use to develop the system, we need to get familiar with the technologies we will be using for the system maintenance.

Technologies

We will use five technologies for this project.

HTML 5	Front End
CSS 3	Front End
JavaScript	Front End
PHP 7.4	Back End
MySQL	Back End

Getting Started

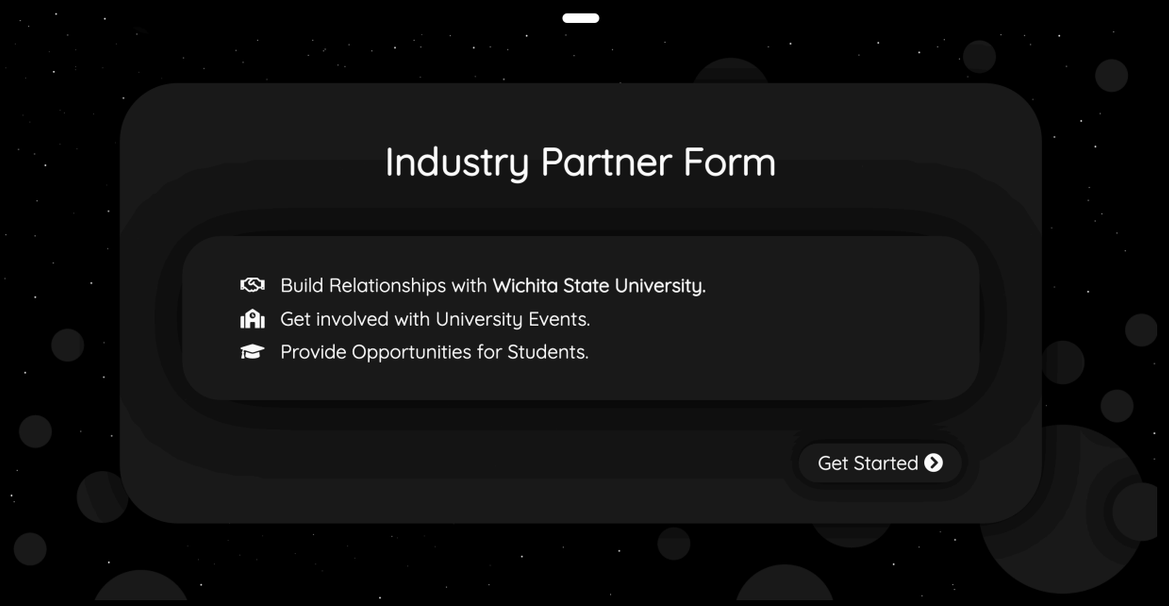
Form

For the web form, we will be using only HTML, CSS, and JavaScript.

The form is built by using only the Front End technologies.

The form contains three basic parts

- Personal & Professional Information
- Education Experience
- Involvement Opportunities



Purpose

The web the purpose of the web for me to populate the Industry Partner Database Table.

Database

For the Database we will be using only MySQL.

There are two important tables in the database.

- Industry Partner Database
- main

Industry Partner Database table is the primary table that contains information about all form submissions.

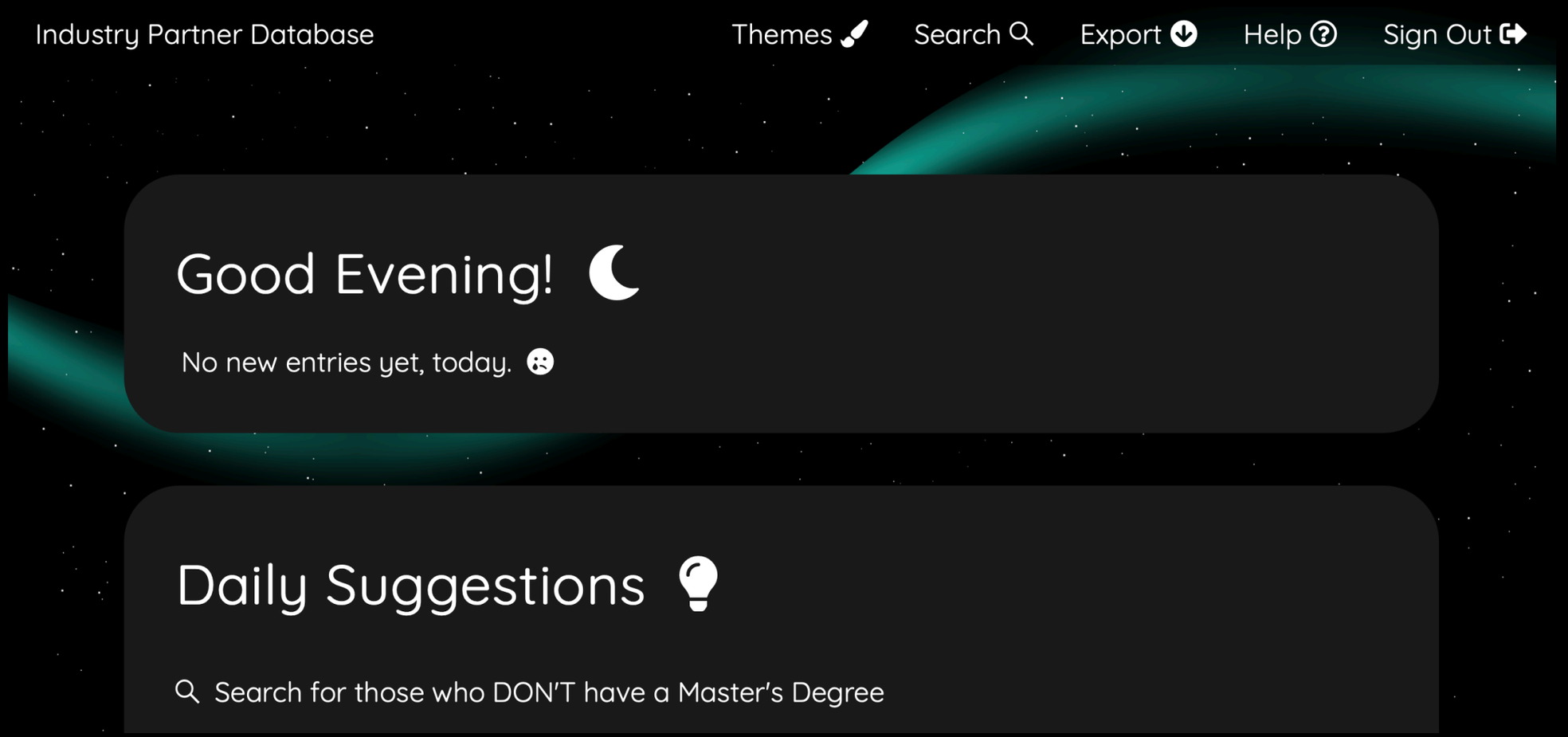
The main table contains information about users and login data such as email, password, themes, and type of user.

Dynamic webpage

We i'll be using all the five technologies for the Dynamic Webpage.

Purpose

The purpose of the Dynamic Webpage to retrieve information from the Industry Partner Database table, and also to update information in the main table of the database.



Understanding Tools

Front End

There are lots of tools that you can use for managing the Web Form.

These tools include JavaScript functions and CSS rules.

Let's go over the CSS rules.

The web form contains three major divisions

- Personal & Professional Information
- Education Experience
- Involvement Opportunities

For each form division, we will use the CSS class `.formDiv` and its CSS rules can be found in the file style_dark.css

Inside `.formDiv` Form Division, we can have multiple form sections and for the form sections, we will use the class `.formSection`

A form section represents an individual or an independent section of the form that asked the user specific questions.

Inside each form section, you can multiple `.sideByside` elements, which will contain two `.normalBlock` elements.

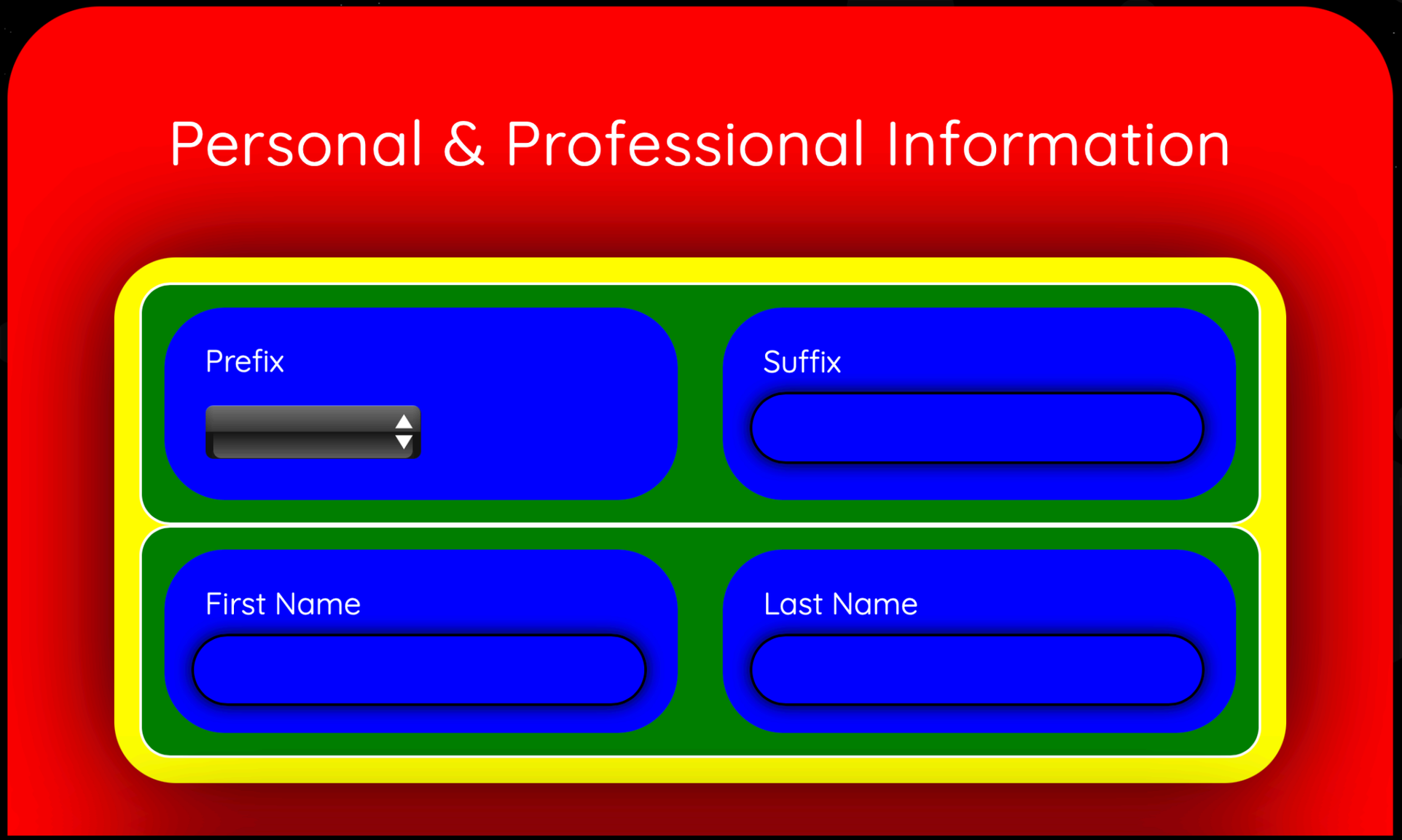
There are lots of CSS classes that are used throughout the web form. We will discuss the significance, pros, and cons of each major CSS class that brings the web form to life.

Also, keep in mind that all these classes may look different depending on the stylesheet being used. We use same classes but with different rules in different stylesheets.

In this way, we can rapidly and dynamically change the overall theme of the given webpage.

This is an important programming practice and we use this in the Dynamic Webpage.

Next, you will see the entire layout of a `.formDiv` to get a better idea of the structure of these elements.



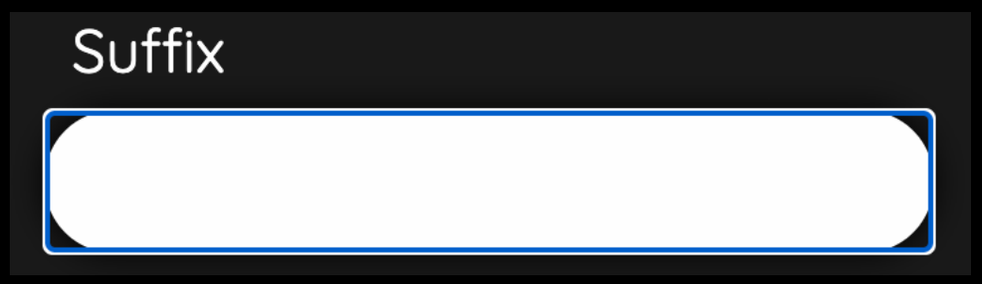
Form Division Layout

`.formDiv` `.formSection` `.sideByside` `.normalBlock`

Now, let's talk about CSS rules for various classes of web form.

```
input:focus, textarea:focus, select:focus
{
  outline: none;
}
```

We do this to avoid the below outer blue line.

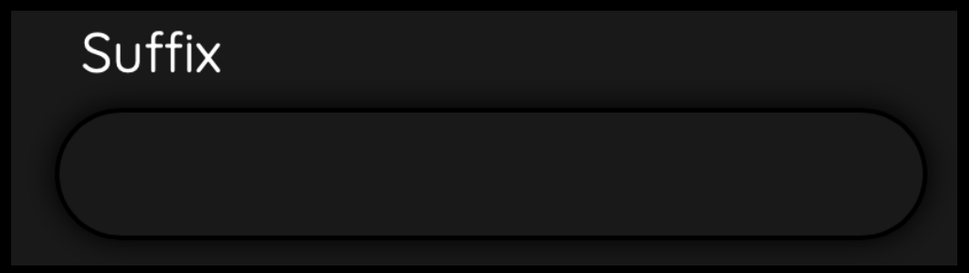


`.mainTitle`

We use the class `.mainTitle` to represent headings.

```css
box-shadow: 0em 0em 2em black;
```

We use the above code to get the element's shadow.



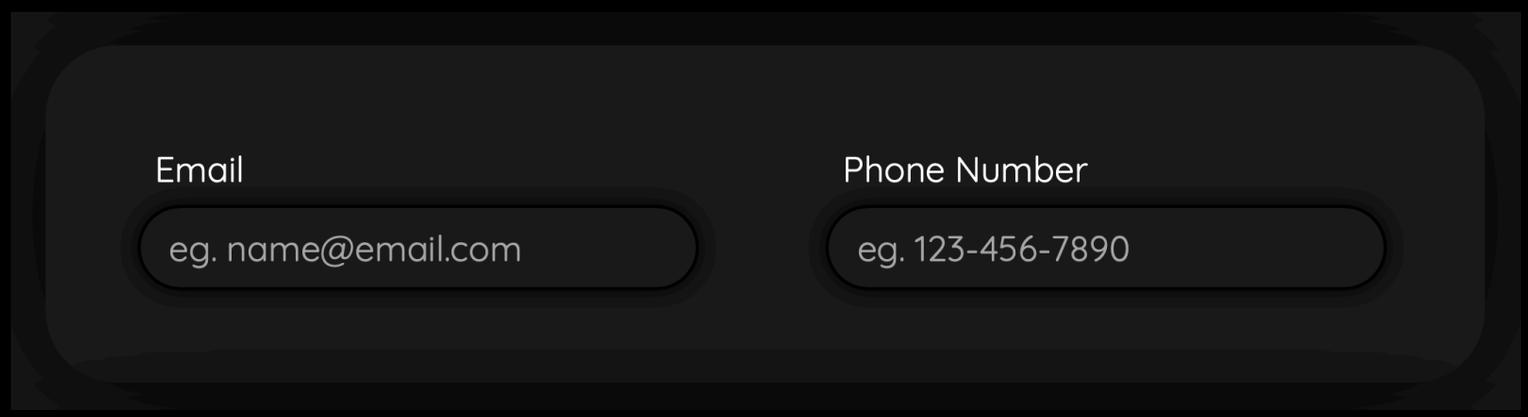

```css
border-radius: 2em;
```

We use the above code to curve the border radius of various form elements.

We use the code below to keep two elements side by side.

```css
.sideByside
{
  display: flex;

  margin: 0;
  padding: 0;
}
```



We use this code to style the selection of the text.


```
::selection
{
  background-color: white;
  color: black;
}
```

We animate elements in the webpage by using @keyframes and by storing these keyframes in classes like shown below.

```
@keyframes messagePop
{
  0%
  {
    transform: scale(0.4);
    opacity: 0;
  }
}
```

```
.msgPopAnimation
{
  animation-name: messagePop;
  animation-duration: 0.25s;
  animation-fill-mode: both;
}
```

We use the class below to make sure that the elements do not scale when user hovers the cursor over them.

```
.noHover:hover
{
    transform: none;
}
```

Checkboxes & Radiobuttons

We will use our own custom designed radio boxes and check boxes in the form.

Do you have some form of college education?

- No, I have not taken any college classes
- Yes, I have taken some college classes
- Yes, I have an Associate's degree

What is your preferred level of involvement?

(Select all that apply)

- A one-time event lasting 1 to 2 hours
- A day long event
- A recurring relationship over a semester

Motivation

Checkboxes and Radio Buttons have two different purposes.

Check Boxes = to select ONE or MORE options

Radio Buttons = to select ONLY ONE option

Now, if we pay close attention, one thing they both have in common is SELECT

This is the key idea for innovation. This is the idea behind the redesign or merging both input elements. With this, the user has only one goal : TO SELECT

This is why we merge radio buttons and check boxes and it will be very easy later to add an option to create an entire list of radio buttons or checkboxes.

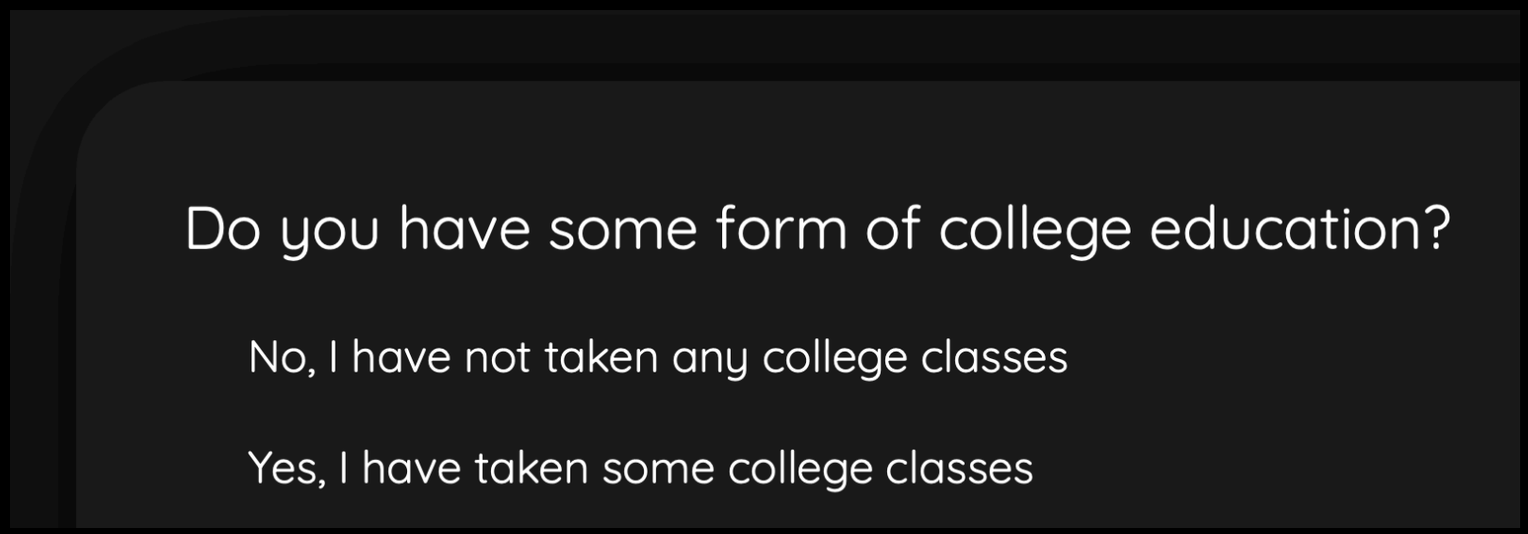
Radio Buttons

Now let's see how we can create a list of radio buttons.

For this, we need to follow a basic structural rule by using HTML technology. The rule is to have different elements in the right order and inside each other. The right ordering of elements is shown below.

- Create a `.formSection` element
- Create a `.radioBlock` element and a label for the radio buttons
- Create a `.radioList` that contains list items which are pairs of inputs and labels

```
<div class="formSection" id="college_education_div">
<div class="radioBlock">
  <label>Do you have some form of college education?</label>
  <ul type="none" class="radioList">
  <li>
    <input type="radio" id="college_education1" name="college_education"
    value="1" required>
    <label for="college_education1">No, I have not taken any college classes</label>
  </li>
```



Checkboxes

Now let's see how we can create a list of Checkboxes.

Similar to the creation of radio buttons with very few changes.

The right ordering of elements for creating checkboxes is shown below.

- Create a `.formSection` element
- Create a `.checkBlock` element and a label for the checkboxes
- Create a `.checkList` that contains list items which are pairs of inputs and labels

In the next page, there is the list of checkboxes created and the code snippet is showcased below along with the output.

The functionality of the radio buttons and checkboxes will be discussed in detail when we learn about JavaScript functions.

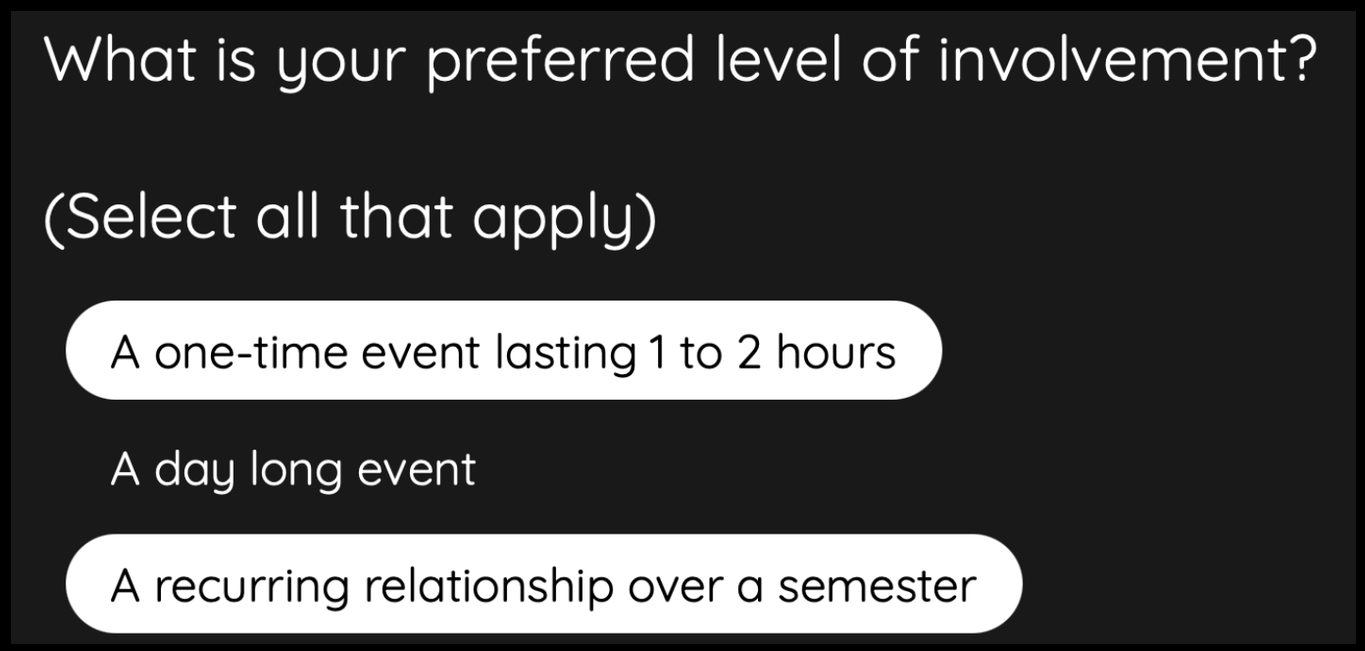
```
<div class="formSection" id="Involvement_Level_div">
<div class="checkBlock">

<label>What is your preferred level of involvement?
  <br><br>(Select all that apply)<br></label>

<ul type="none" class="checkList">

<li>
<input type="checkbox" id="Involvement_Level1" name="Involvement_Level[]"
value="1">

<label for="Involvement_Level1">
  A one-time event lasting 1 to 2 hours
</label>
</li>
```



JavaScript Functions

In this module, we will be learning about various JavaScript tools that could be used to enhance the functionality of the web form.

All these functions can be found in the file validation.js

The basic functions are listed below and will be used to simplify the code.

```
function tag_(_tag)
{
  return document.getElementsByTagName(_tag);
}
```

This function returns all the elements matching the given _tag.

```
function id_(_id)
{
  return document.getElementById(_id);
}
```

This function returns all the elements matching the given _id.

```
function class_(_class)
{
  return document.getElementsByClassName(_class);
}
```

This function returns all the elements matching the given _class.

```
function message(msg, icon = '<i class="fas fa-comment"></i>')
```

This function displays message on top of the screen (like a notification).

Function takes a message msg and also takes an icon but the default icon value is a font-awesome icon that represents a comment.

Calling the function as below

```
message(“Industry Partner Form”, “”)
```

gives you the output below

function animate(_id='message', _animationClass=‘msgPopAnimation’)

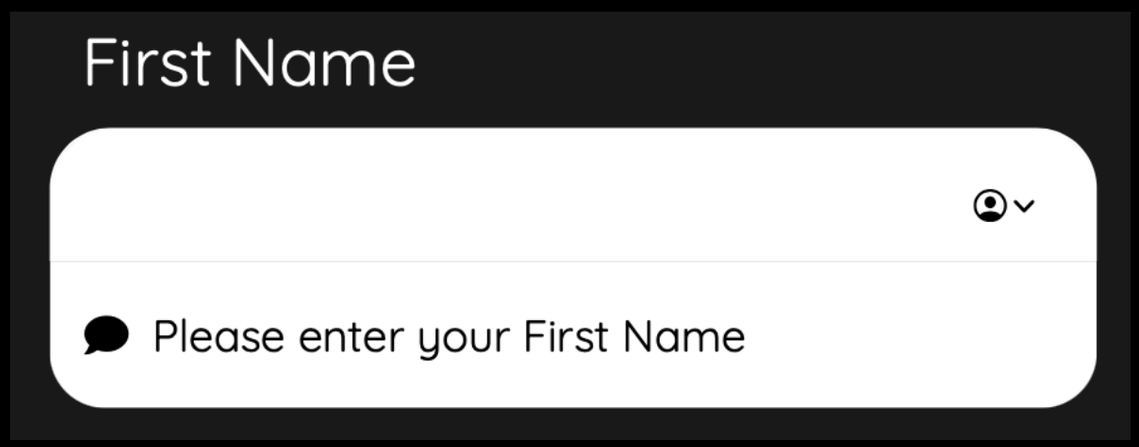
This function intelligently animates the element of given id once. This function will take the id of the element as well as the animation which is a CSS class that contains the key frames of an animation.

function id_message(_id, msg, icon = '<i class="fas fa-comment”></i>’)

This function is similar to the message() except it has the ability to animate any element of given id

```
id_message(guideID, ‘Please enter your First Name)
```

gives you the output below

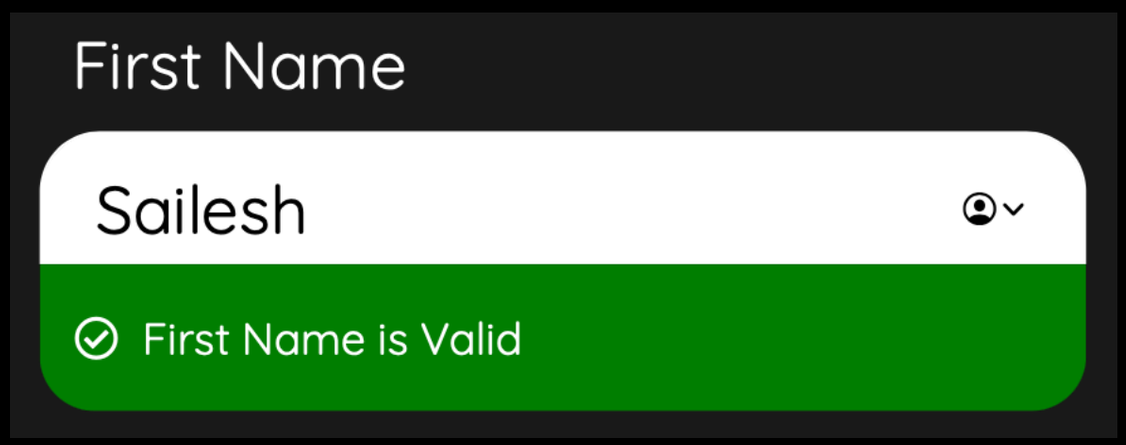


`function id_error_message(_id, msg)`

This function is similar to the id_message() except it displays an error message with red background.

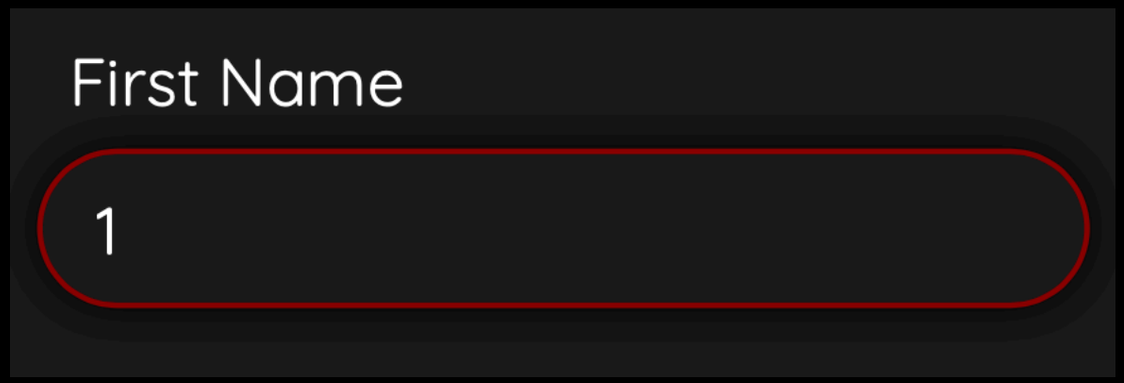
`function id_success_message(_id, msg)`

This function is similar to the id_message() except it displays a success message with green background.



`function checkInput(_id, _regex)`

This function checks whether the value of the HTML element of the given id matches the given regular expression. If the value does not match the regular expression, then the function will highlight the input element with the red border.



`function validate(_id, _regex, _msg, _errorMsg, _successMsg, _inOutAnimation = true)`

This function validates any input element of given id and displays the respective error or success message depending on the value of the element of given id when matched to a given regular expression. The _inOutAnimaiton parameter enables or disables the animation.

To validate a 10 digit phone number, we call the function as below.

```
validate('phone_number', /^(\d{10}|\d{0})$/, 'Enter your 10-digit Phone Number',
  'Enter a valid U.S. Phone Number', 'Phone Number is Valid');
```

`function sanitize(_id)`

This function capitalizes the very first letter of all the words of the string when the user types their input.

`function other(_name, _form = “Industry_Partner_Database")`

The function has the following features

Feature 1 : Automatic detection of the given HTML element.

(Determines whether the given id belongs to a radio button or to a checkbox)

Feature 2 : Element specific form handling.

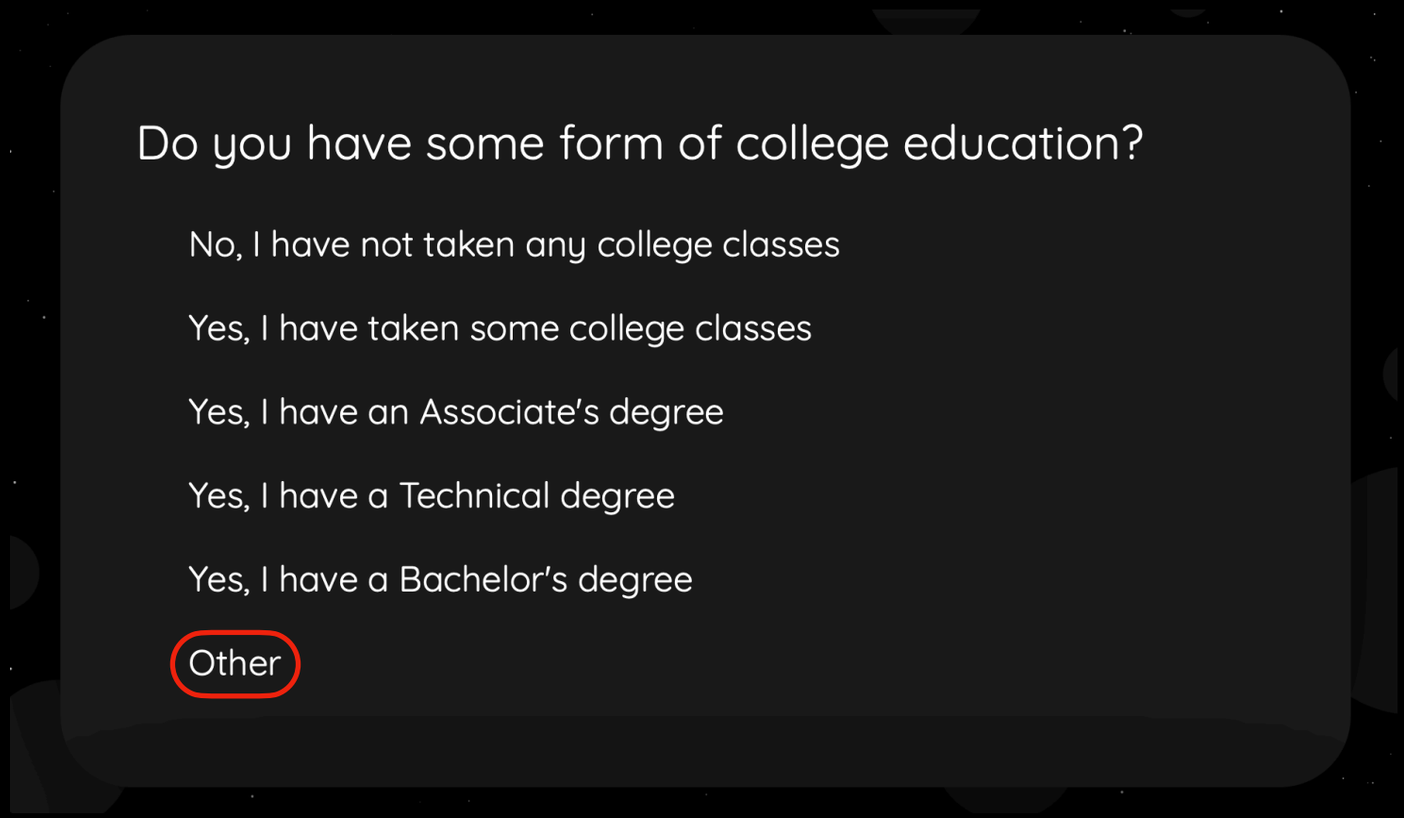
(Makes the resulting text box act and behave either like a radio button or a checkbox)

Feature 3 : Highly Cohesive with Minimal Coupling.

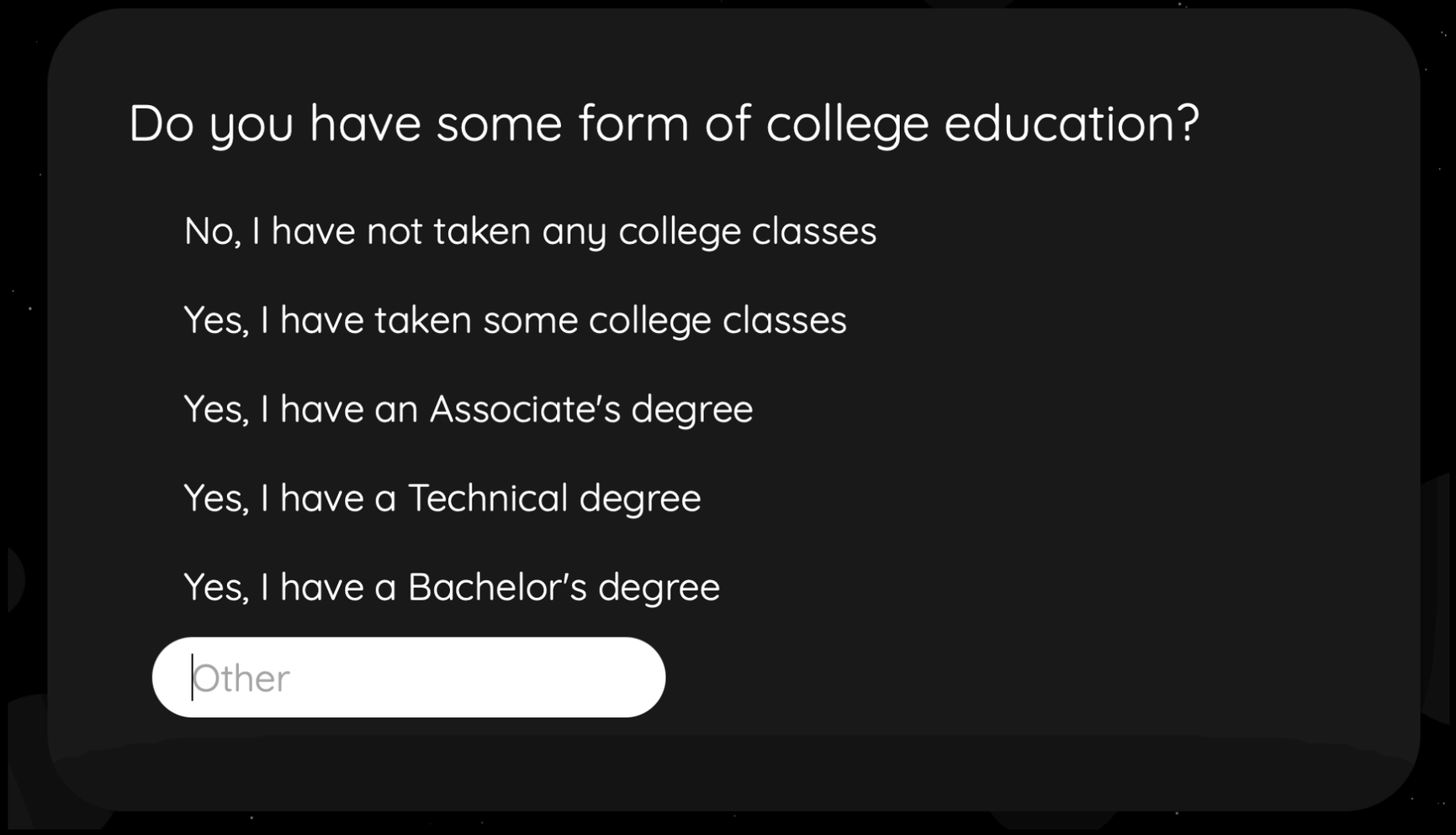
(Programmer can use this function to create the “other” option just by a function call)

```
other("college_education");
```

The creation of “Other” option that is a text box but behaves like a radio button.



When the user selects the “Other” option, it transforms into a text-box.

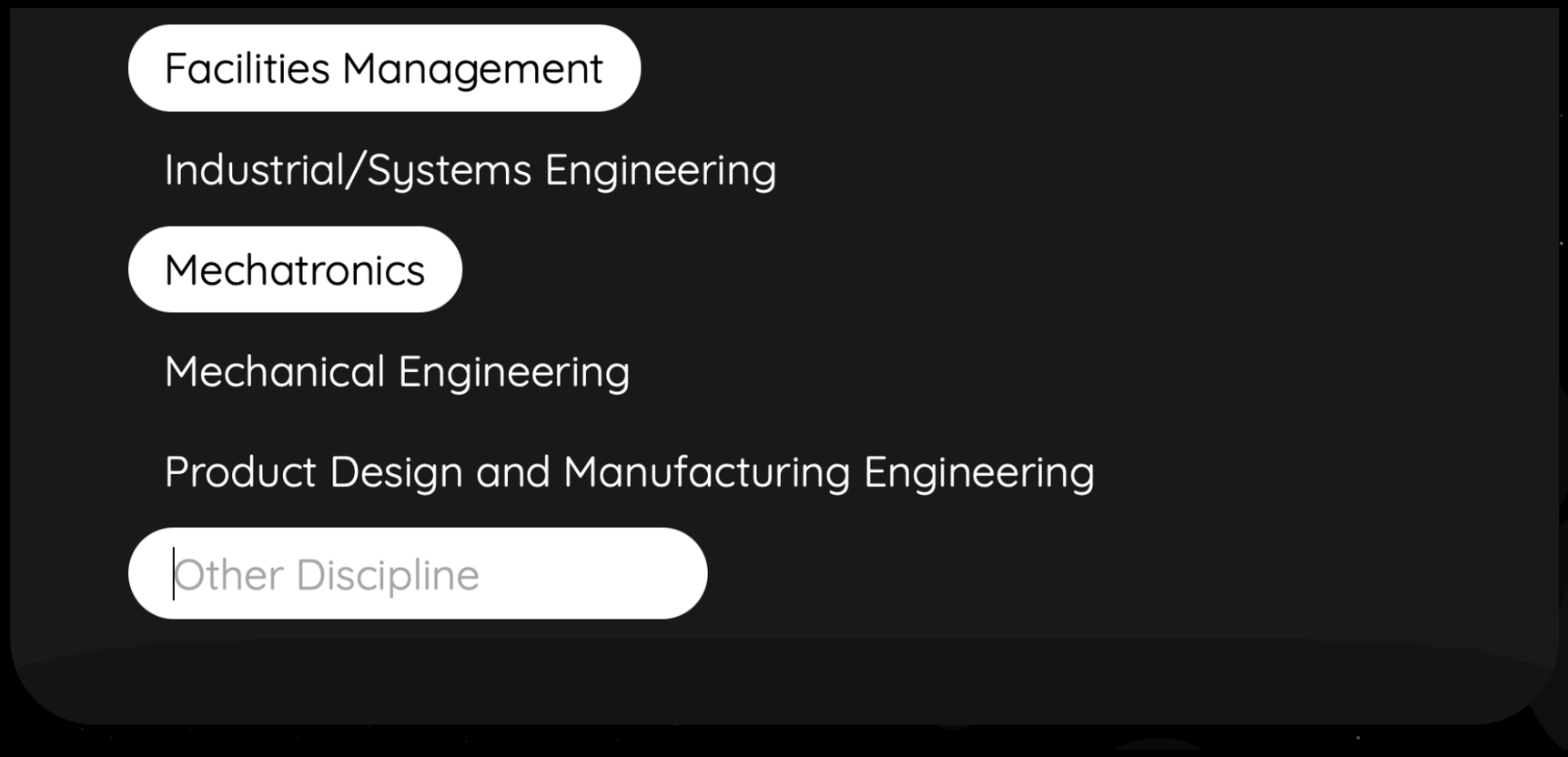


Now, the user is given the freedom of creating their own option among the list of options. This is the power of calling the `other()`

On the other hand, we can also use this functionality for checkboxes as well.

Now, the “Other” in this case, is a text box but behaves like a checkbox.

Therefore, `other()` can make a text-box behave as a radio button or as a checkbox.



```
function isSelected(_name, _otherExists = false, _form =
“Industry_Partner_Database”)
```

This function checks whether in option is selected from either a radio button group or a checkbox list.

```
function isFilled_childTextBox(_checkedID, _target, _regex = “”)
```

This function and make sure that the resulting text-box is filled upon selection of either a radio button or a checkbox.

For example, when the user selects that they have an Associates Degree, then they have to fill the resulting Associates Degree text box as well as the Year text-box.

```
isFilled_childTextBox('college_education3', 'associates_degree')  &&
isFilled_childTextBox('college_education3', 'college_degree_year') &&
```

The code above will make sure that required text fields are filled upon selecting an option.

```
function resetSelectionGroup(_name, _otherExists = false, _form =
“Industry_Partner_Database”)
```

This functions resets all options in either a group of radio buttons or a group of checkboxes.

By resetting, we uncheck all the checked checkboxes and radio buttons and also clear the value of any text field that is associated with the selection group.

```
function isSelected_childRadioGroup(_checkedID, _target, _otherExists =
false, _regex = “”)
```

This function is similar to isFilled_childTextBox() that ensures that the user has selected an option in the resulting required radio buttons group.

```
function isSelected_childCheckBoxGroup(_checkedID, _target, _otherExists =
false, _form = “Industry_Partner_Database")
```

This function is similar to isSelected_childRadioGroup() that ensures that the user has selected one or more options in the resulting required checkboxes group.

```
function switchDiv(targetDiv, currentDiv, _button = “Next”)
```

This function changes between the form divisions. This function gets inputs when the user clicks a previous button or next button in the form division and thereby switches between divisions. The function call for this function is inside the HTML file and is in the element of class `.prevNextDiv`

```
function displayOnSelect(_targetID, _selectID, _form = “Industry_Partner_Database”)
```

This function displays the resulting form section of inputs when they use that selects the target option.

```
function higherOrderDisplayConstraint(_targetDivID, _higherID, _form =
“Industry_Partner_Database")
```

This function applies constrains that check for correct form functionality. For example, if a person checks an the Bachelor’s Degree option, then the form will display additional questions but if the user decides to change their option to “I prefer not to answer”, then we have to not display all the additional questions we displayed when the user selected that they have Bachelor’s Degree.

Using this function, we do exactly that. We will simply call this function with the specific parameters to create constrains (restrictive relations) or rules that controls displaying relevant information basing on the user’s input.

Dynamic Webpage

The dynamic webpage is built using all the five technologies.

- HTML 5
- CSS 3
- JavaScript
- PHP 7.4
- MySQL

The dynamic webpage retains information from the database. The dynamic webpage execute MySQL queries dynamically and uses PHP 7.4 as a server that moves between the database and the webpage to retrieve and update data of the database.

This dynamic webpage is built on the foundation of different Front End technologies like HTML, CSS, and JavaScript. There are many innovative features that bring the dynamic webpage to life like Search, Export, Search Suggestions, and Themes.

Fonts

When you look at the dynamic webpage, you will first notice that there are many external links to various fonts across the internet. All of these links from Google fonts, which is a simple and free store for web fonts.

Using Google fonts is really easy as it only requires you to copy the external link of the front and put it in the <head> tag of the webpage and use the foreign family in the CSS rule of the element of the webpage.

Icons

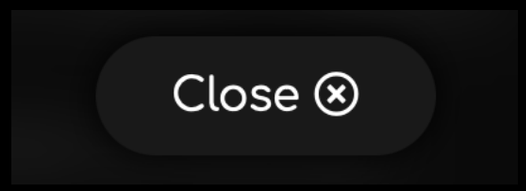
It is very obvious and also became an obsession for us to include a lot of icons that will make it very easy across the entire system for navigation and interaction. For using various icons that represent or share an idea, we use an external link for the entire set of icons from fontawesome.com

```
<i class="far fa-smile-wink”></i>
```

represents a smiley icon like this.

Functional Elements

```
<button class="uiButtonOff" id="close" onclick="closePreview()">
  Close <i id="closeLinkIcon" class='far fa-times-circle'></i>
</button>
```



The HTML element with id `#close` will be used as a button to close any record preview that is active.

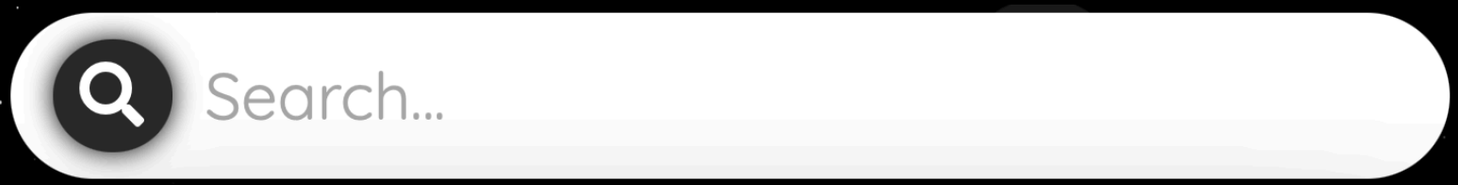
```
<div id="layer"></div>
```

The HTML element with id `#layer` will be used as a blur background for the record preview.

```
<div id="searchSuggestionsDiv"></div>
```

The HTML element with id `#searchSuggestionsDiv` will be used to display all Search Suggestions upon loading the webpage.

The above is the `#topBar` that contains all the **links** or **buttons** to various **features** of the webpage.



The above is the `#searchBarDiv` and this **contains** two elements the **Search Button** and the **Search Text-box**.

These are the two primary **functional** elements of the **webpage** and then these elements **carry out** various functions which are important **features** of the webpage like **changing themes**, **searching for a record**, and even **exporting data** from the **database**.

Dynamic Webpage's JavaScript Modules

```
<script type="text/javascript" src="searchBar_Graphic.js"></script>
<script type="text/javascript" src="searchBar_Functionality.js"></script>
```

The above two scripts determine the entire functionality of the Search Div.

The searchBar_Graphic.js controls the dynamics of the Search Bar. This script basically controls the behavior of the `#searchBarDiv` when the search text-box is focused. The script changes the css rules of the `#searchBarDiv` by adding and removing its CSS classes.

The searchBar_Functionality.js controls the functionality of the `#searchBarDiv`

For example, the script controls everything that happens when the users clicks the search feature button in the `#topBar`

The animations as well as displaying and hiding the `#searchBarDiv` are controlled by this script.

Search Suggestions

```
function precisionSearchSuggestion(_label, _searchConditions, _action = "advancedSearch.php")
```

The above function creates a search suggestion by using the given label and searchConditions.

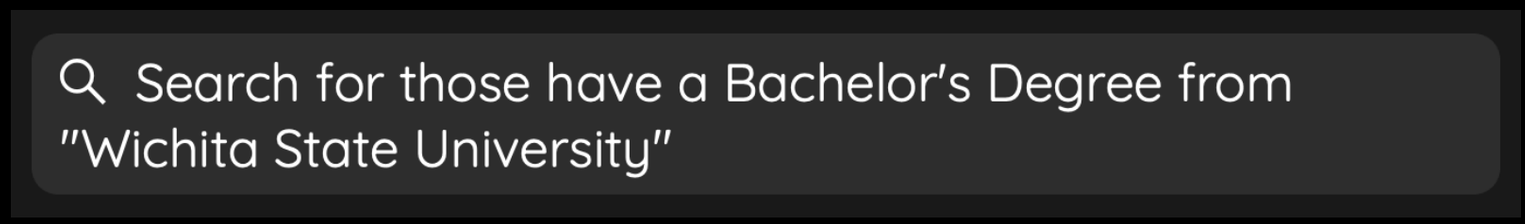
Label is the label that the user sees or what represents the search suggestion.

searchConditions are ONLY the MySQL WHERE statements that contains one or more OR, AND conditions.

```
suggestions.push(precisionSearchSuggestion(
  "Search for those have a Bachelor's Degree from \"Wichita State University\"",

  " WHERE UPPER( BS_school ) LIKE UPPER( '1' ) " +
  " OR UPPER( BS_school ) LIKE UPPER ( '1,%') " +
  " OR UPPER( BS_school ) LIKE UPPER ( '%, 1,%') " +
  " OR UPPER( BS_school ) LIKE UPPER ( '%, 1')"
));
```

The above function call creates this Search Suggestion.



Themes

One of the important features of the webpage is to provide the user with the ability to change the look on the web page. This is a feature we called as Themes. With Themes, the user can change the entire theme of the dynamic webpage just by clicking a button.

This is done by changing the entire CSS stylesheet the webpage with the click of a button.

Therefore, there are custom built themes available to you to explore and even create your own theme.

Some of these themes include the following.

- Bright * Dark * Midnight — Extremes
- Summer * Abyss * Citrus * Bubbles — Celebration of Color
- WSU — University Theme

There are 8 themes in total.

To understand the Themes feature even better, let us understand how they came to life.

Firstly, since there are many HTML elements in the dynamic webpage and we will see all the CSS rules for all these elements in a single CSS.

Since there are 8 themes, we will then have 8 different CSS style sheets, which are unique.

This means, we can change the theme of the webpage just by changing one line of code.

The only line which we have to change is the external link to this team CSS file.

```
<link href="dark.css" id="themeCSS" rel="stylesheet" type="text/css">
```

Changing the above line of code to the below changes the theme from dark to citrus.

```
<link href="citrus.css" id="themeCSS" rel="stylesheet" type="text/css">
```

Notice that all the css files of the themes are named after them. For instance,

- citrus.css is the CSS file for the Theme Citrus
- abyss.css is the CSS file for the Theme Abyss

and so on, and please make use of the `#themeCSS` id to identify the external link when chaining its attribute in the future.

Precision Search

Precision Search enables you to search precisely by creating your own Search Conditions.

Precision Search in technical terms, is a webpage that contains one or more number of searchConditions.

Each searchCondition is an HTML division that contains 4 inputs.

- A Conditional Selector [AND, OR]
- Table Column Selector { all Database table columns }
- A Operation Selector [IS, CONTAINS, STARTS WITH, ENDS WITH]
- Either a Text-Box or a Value Selector (contains all the possible values for a table column)

Here, the user gets to add / remove searchConditions and ultimately create their own search rules and search for the records in the database that satisfies the search rules.

Now, let's go over some basic JavaScript functions that bring Precision Search to life.

Precision Search's JavaScript Modules

```
function addRule(targetDivId = "searchConditionsDiv")
```

The addRule() function adds a searchCondition whenever the user clicks the add button

Every Search condition that is added comes with a remove button that can remove that particular such condition.

```
function replaceTextFieldWithSelectorIn(sourceArray, searchCondition)
```

The above function is very useful and is an important function of the Precision Search feature. This function replaces the text-field with a selector, which is not any ordinary selector but a selector that contains all options which are all the possible values of the selected table column.

```
function inputCalibration(tableColumnSelector)
```

This function enhances the dynamics of the Precision Search. It will detect whenever the user selects a table column and automatically calibrates or changes the input to either a text box or a selector depending on the selected table column.

Back End

For the Back End Development, we use the following technologies.

- PHP 7.4
- MySQL

We use PHP to submit all the entries provided by the user into the Database. We also use PHP to retrieve information from the database and displayed in the Dynamic Webpage.

We will focus on prerequisites.php file. This file contains a lot of PHP functions that we could use across all other PHP scripts in the system.

Next, we will discuss or walk through all the important variables and functions that are defined already for us to make our development easy.

Prerequisites.php

The first thing we will notice in the file prerequisites.php is a block of code that represents various array declarations. These array declarations store a lot of useful information regarding the form and the database.

`$htmlFields` is an array that contains all the names of HTML input fields in the form.

`$tableColumns` is an array that contains all the table column names stored in the database.

There are other arrays the store information about all the options of either a radio buttons group or a list of checkboxes like `$collegeEducation`, `$bsSchool`, `$Fields`, `$engDisciplines` etc.

In this file, there are also some important shortcuts for database schema like `$insertSchema` which is dynamically created by using the `$tableColumns` array.

```
function printRecords($sql)
```

This function prints all the records (not record previews) basing on the given MySQL from the database.

```
function printRecordPreviews($sql)
```

This function prints all the records previews basing on the given MySQL from the database.

A Record Preview is actually an element in the webpage that displays the entire information about a particular record.

```
function searchArrayForKeyword($arr, $keyword)
```

This function searches all elements of the given array that contain the given keyword and it will return the array of indices of the matching elements.

```
function optionDetectionRules($columnName, $option)
```

This function returns the rules or Search Conditions to find the encoded option in the database.

This function will be used for the search feature of the dynamic webpage.

```
function getSearchConditionsFor($keyword)
```

This function returns all these Search Conditions basing on the given search keyword.

This function generates all search conditions basing on MySQL syntax.

```
function exportSearchLink($keyword)
```

This function returns a button that acts like an export button for any search results in the form that downloads all the search results as an Excel File.

Contact Support

Sailesh Rajanala

◉ sailesh777@live.com

Priyanka Limbu

◉ limbupriyanka8@gmail.com

Subash Acharya

◉ hubert3158@gmail.com

Shiva Karki

◉ cvakarki17@gmail.com

References

- https://fonts.google.com/
- https://www.javascript.com
- https://fontawesome.com/
- https://www.htmldog.com/
- https://www.tutorialspoint.com/
- https://www.javascript-coder.com/
- https://css-tricks.com/
- https://www.w3schools.com/
- https://www.php.net/
- https://www.mysql.com/