

KodeGo – Hands on Activities









HTML and CSS Hands On 1

Instruction:

1. Create html table with the following data:

First Name	Last Name	Address	City	Age
Mickey	Mouse	123 Fantasy Way	Anaheim	73
Bat	Man	321 Cavern Ave	Gotham	54
Wonder	Woman	987 Truth Way	Paradise	39
Donald	Duck	555 Quack Street	Mallard	65
Bugs	Bunny	567 Carrot Street	Rascal	58
Wiley	Coyote	999 Acme Way	Canyon	61
Cat	Woman	234 Purrfect Street	Hairball	32
Tweety	Bird	543	Itotltaw	28

2. Create another table reference to the first one, but this time add table header "Image" and insert image in table data, images will have same width and height.

First Name	Last Name	Address	City	Age	Image
Mickey	Mouse	123 Fantasy Way	Anaheim	73	
Bat	Man	321 Cavern Ave	Gotham	54	
Wonder	Woman	987 Truth Way	Paradise	39	
Donald	Duck	555 Quack Street	Mallard	65	
Bugs	Bunny	567 Carrot Street	Rascal	58	
Wiley	Coyote	999 Acme Way	Canyon	61	
Cat	Woman	234 Purrfect Street	Hairball	32	
Tweety	Bird	543	Itotltaw	28	

3. Make an ordered list of your top 3 restaurant and link their website address to your list items.
4. Make an unordered list of 5 colors.
5. Create an article with your own content.
6. Get the embed google map of your town address use this [link.Links to an external site.](#)
7. Define a video content that specify video control such as play and paused, height and width should be 300px, video will start all over again as it finish and muted as default as the page loaded.

Note: Upload the git link and the screen shoot of your hands on here at canvas.

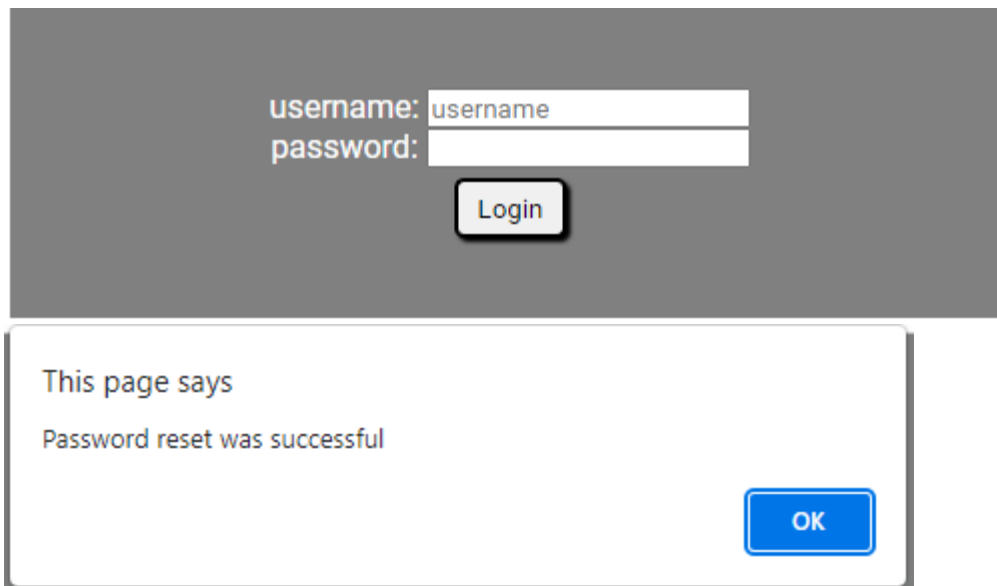
Objective: Defining and Differentiating HTML elements

Tools: Visual Studio Code, [HTML Tutorial \(w3schools.com\)](https://www.w3schools.com/html/)[Links to an external site.](#)

HTML Hands On 2

Instruction:

1. Define Login Form with userName , password input field and login button
upon Login button click it should show an alert Login successful.



The image shows a login form on a dark gray background. It consists of two input fields: one labeled 'username:' with the placeholder text 'username', and another labeled 'password:'. Below these fields is a button labeled 'Login'. Below the form, there is a white alert dialog box with a gray border. The dialog box contains the text 'This page says' followed by 'Password reset was successful' on the next line. In the bottom right corner of the dialog box is a blue button labeled 'OK'.

2. Define Forgot Password form with old Password inputfield, new password inputfield and submit button Upon Submit button click an alert should be displayed “Password reset was successful”.

3. Define Registration Form with the following fields:

Field	Type	Required
firstName	text	yes
middleName	text	no
lastName	text	yes
birtDate	date	yes
Email	email	yes
contactNumber	text	yes
City	Dropdown list(Cities of your choice)	no
preferredContactTime	Checkboxes(Morning, Afternoon, Evening)	yes
Register	submit	

then upon clicking Register button alert "Hi! firstName your account is Successfully Registered".

Objective: Defining and Familiarizing form elements, attributes and javascript function implementation.

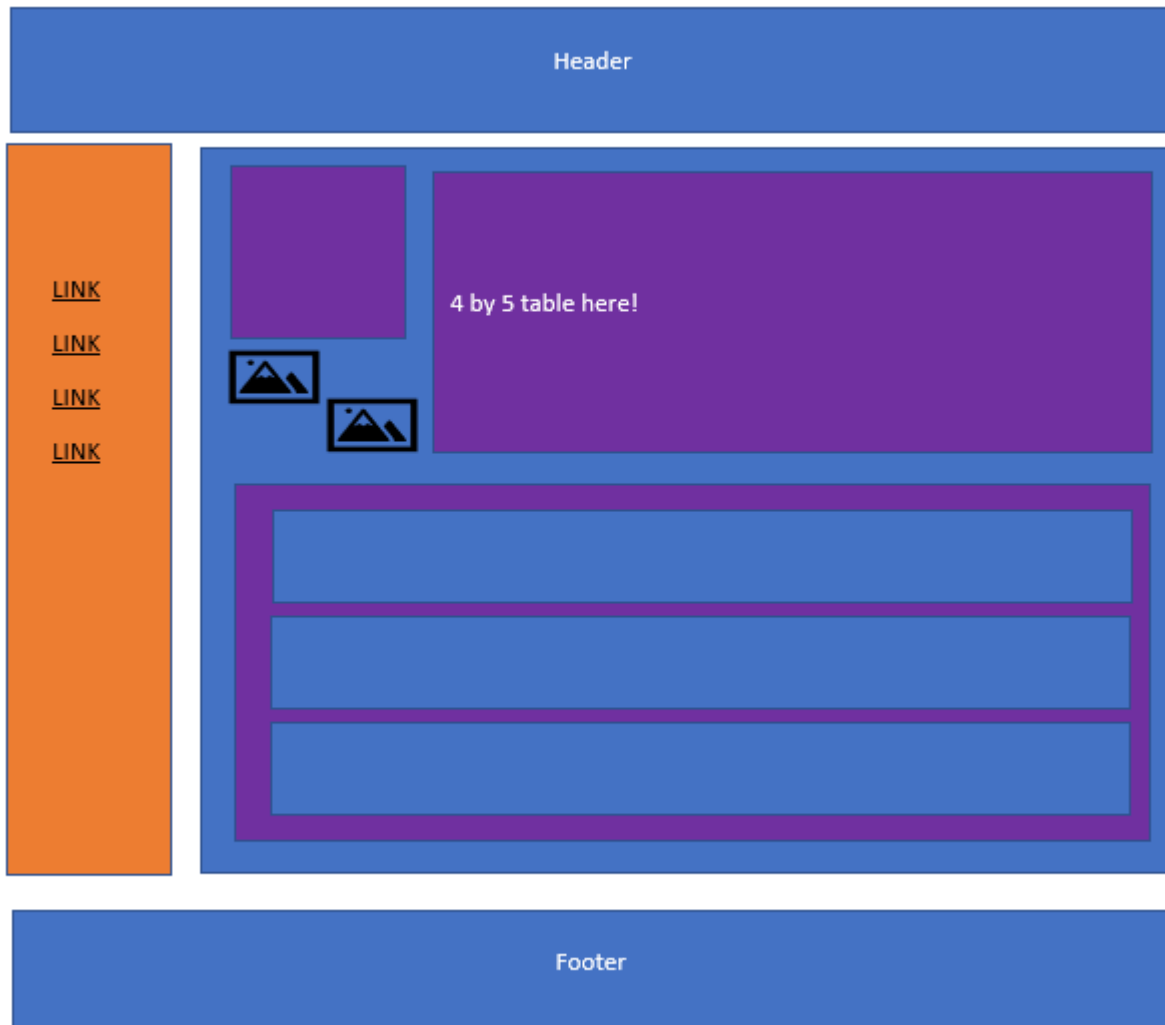
Note: Upload your git link and screen shoots of your output here at canvas.

Tools: Visual Studio Code, [HTML Tutorial \(w3schools.com\)](https://www.w3schools.com/html/)[Links to an external site.](#)

HTML and CSS Hands On 3

Instruction:

1. Define the content of web browser as per requirement of image below.



Objective: Familiarized with the positioning attribute of elements and how margin, padding and border different to each other.

Note: Upload screen shot of your out and git hub link here at canvas

Tools: Visual Studio Code.

HTML and CSS Hands On 4

Instruction:

1. Create a copy of webpage as per the requirement of the image below.



2. Create your box layout.
3. Image Resources [here](#). [Links to an external site.](#)

Note: Upload your git link and screen shoot of your output here at Canvas

Objective: With this hands on you will experienced technique on how to duplicate a layout from existing webpage.

Tool: Visual Studio Code

HTML and CSS Hands On 5

Instruction:

1. Define the content of web page as per the requirement of the image below.

Group	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Period																		
1	1 H 1.008																	2 He 4.0026
2	3 Li 6.94	4 Be 9.0122											5 B [277.15]	6 C [276.15]	7 N [281.16]	8 O [290.16]	9 F [285.17]	10 Ne [20.18]
3	11 Na 22.990	12 Mg 24.305											13 Al [293.18]	14 Si [296.17]	15 P [303.99]	16 S [3205.12]	17 Cl [3205.12]	18 Ar 39.948
4	19 K 39.098	20 Ca 40.078	21 Sc 44.956	22 Ti 47.867	23 V 50.078	24 Cr 51.996	25 Mn 54.938	26 Fe 55.845	27 Co 58.933	28 Ni 58.693	29 Cu 63.546	30 Zn 65.38	31 Ga 69.723	32 Ge 72.64	33 As 74.9216	34 Se 78.96	35 Br 79.904	36 Kr 83.798
5	37 Rb 85.468	38 Sr 87.62	39 Y 88.906	40 Zr 91.224	41 Nb 92.906	42 Mo 95.94	43 Tc [98.906]	44 Ru 101.07	45 Rh 102.9055	46 Pd 106.3277	47 Ag 107.8682	48 Cd 112.4118	49 In 114.818	50 Sn 118.710	51 Sb 121.757	52 Te 127.6	53 I 126.905	54 Xe 131.29
6	55 Cs 132.905	56 Ba 137.327	57 La [138.905]	58 Ce 140.12	59 Pr 140.90765	60 Nd 144.242	61 Pm [144.9126]	62 Sm 150.36	63 Eu 151.964	64 Gd 157.25	65 Tb 158.92534	66 Dy 162.5001	67 Ho 164.93032	68 Er 167.259	69 Tm 168.93274	70 Yb 173.044	71 Lu 174.967	72 Hf 178.49
7	87 Fr [223.018]	88 Ra [226.025]	89 Ac [227.033]	90 Th [232.0377]	91 Pa [231.036]	92 U [238.02891]	93 Np [237.048173]	94 Pu [244.06422]	95 Am [243.061381]	96 Cm [247.070351]	97 Bk [247.070351]	98 Cf [251.079589]	99 Es [252.083312]	100 Fm [257.10371]	101 Md [258.103868]	102 No [259.108888]	103 Lr [262.10536]	104 Uuo [289]
*Lanthanoids			57 La 138.905	58 Ce 140.12	59 Pr 140.90765	60 Nd 144.242	61 Pm [144.9126]	62 Sm 150.36	63 Eu 151.964	64 Gd 157.25	65 Tb 158.92534	66 Dy 162.5001	67 Ho 164.93032	68 Er 167.259	69 Tm 168.93274	70 Yb 173.044		
*Actinoids			89 Ac [227.033]	90 Th [232.0377]	91 Pa [231.036]	92 U [238.02891]	93 Np [237.048173]	94 Pu [244.06422]	95 Am [243.061381]	96 Cm [247.070351]	97 Bk [247.070351]	98 Cf [251.079589]	99 Es [252.083312]	100 Fm [257.10371]	101 Md [258.103868]	102 No [259.108888]		

Objective: With this hands on you can enhanced efficient construction of your CSS and minimized the number of declaration of classes.

Note: Upload your HTML and CSS file at your git hub account send a the link and screen shoot of your output

Tool: Visual Studio Code

HTML, CSS, and JavaScript

Creative Developer Assessment

Do you have Photoshop?

If you have a copy of Photoshop, please carry on and use it.

However, if you do not have a copy of Photoshop, please use this free online alternative instead: <https://www.photopea.com/>Links to an external site.

Materials: <https://drive.google.com/drive/folders/1HazCSsr4CDG6SVCAARM3YliQslPM-3BV?usp=sharing>Links to an external site.

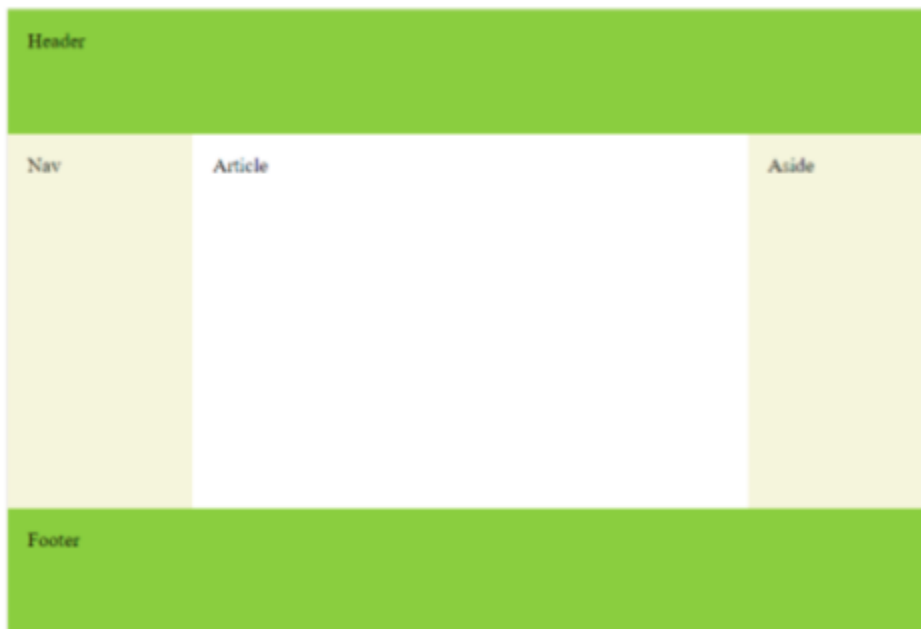
Animation Guide:

[Watch my Powtoon: untitled-1](#)Links to an external site.

CSS Layout Hands On 1

Instruction: Design the webpages as per the requirements of the image below. Use Flex css properties to layout the webpages.

1. Create a file holygrail1.html



2. Create a file holygrail2.html



3. Create a file holygrail3.html

Header

Name

Phone

Email

Pickup Date/Time ☐

Pickup Place

Dropoff Place

Submit Booking

Which vehicle do you require?

☐ Van

☐ Jeepney

☐ Tricycle

Extras

☐ Baby Seat

☐ Wheelchair Access

☐ Stock Tip

Aside

Nav

Footer

4. Create link tag in every Nav elements to connect each pages.

Use this [linkLinks to an external site.](#) to be familiarized with html semantics.

Objective: Creating css layout with the use of display flexbox properties.

Note: Upload screen shot of your output and git hub links here at canvas.

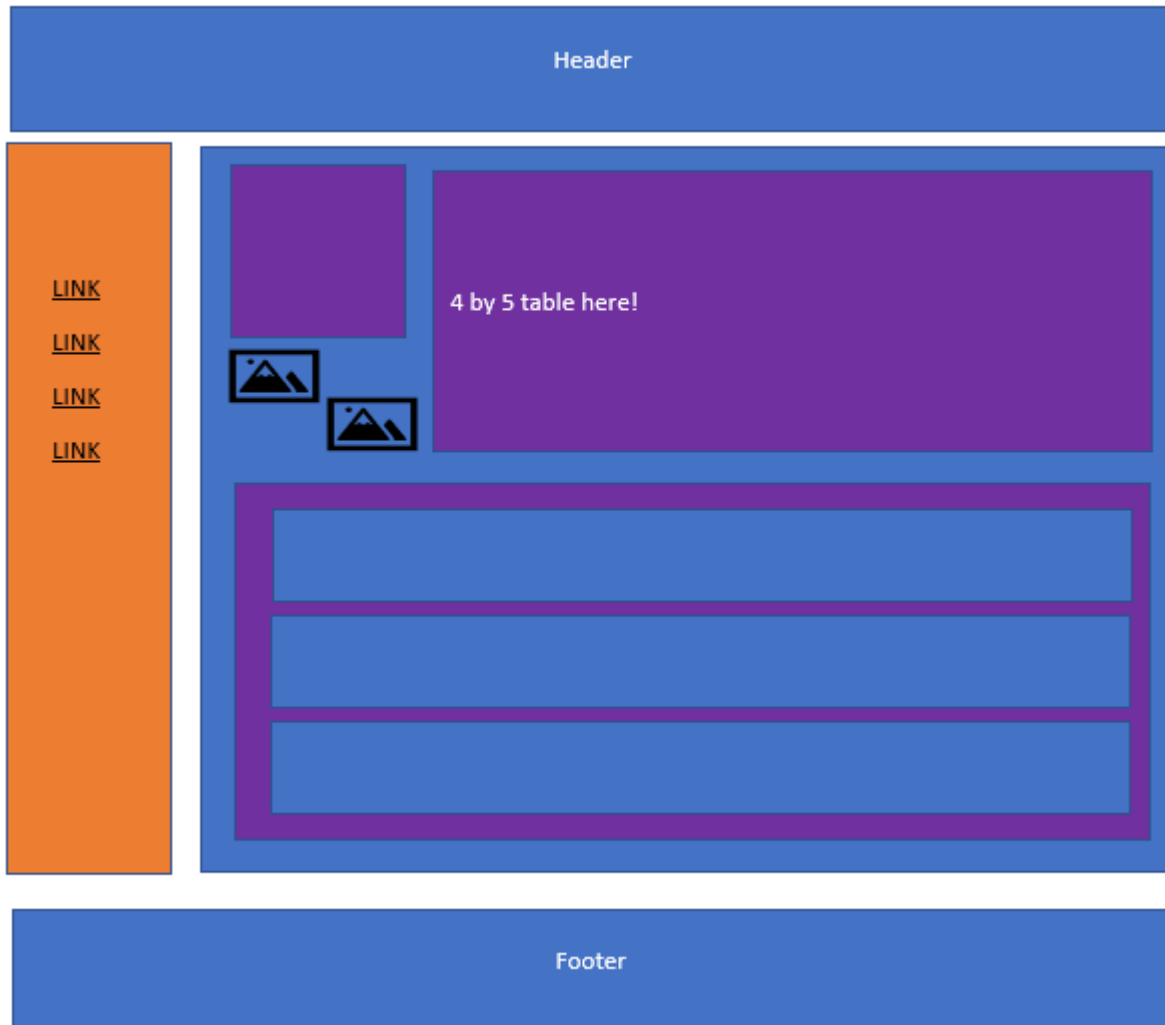
Tool: Visual Studio Code

CSS Responsive Hands On 2

[Start Assignment](#)

Instruction:

1. Make your [WD37]HTML and CSS Hands On 3 Responsive by adding media query on your css file.



2. create layout design for phone, tablet and desktop view.

Objective: Familiarized on how to used media query and design a responsive layout

Note: Upload screen shot of your output and git hub link here at canvas

Tools: Visual Studio Code.

Bootstrap Hands On 1

Instruction: Create the following requirement using bootstrap framework no additional CSS is needed.

1. Design a table shown below.

Requirement 1

First Name	Last Name	Email
John	Doe	john@example.com
Mary	Moe	mary@example.com
July	Dooley	july@example.com

2. Define the button inside the table as shown below.

Requirement 2

Button	Link	class
default	default	btn
Primary	Primary	btn btn-primary
Info	Info	btn btn-info
Success	Success	btn btn-success
Warning	Warning	btn btn-warning
Danger	Danger	btn btn-danger
Inverse	Inverse	btn btn-danger
Link	Link	btn btn-link
Large	Large	btn btn-primary btn-lg
Small	Small	btn btn-primary btn-sm
Mini	Mini	btn btn-primary btn-mini
Block	Block	btn btn-primary btn-block
Disabled	Disabled	btn btn-primary disabled

3. Create a login page show below

Please sign in

Email address

fsf

Password

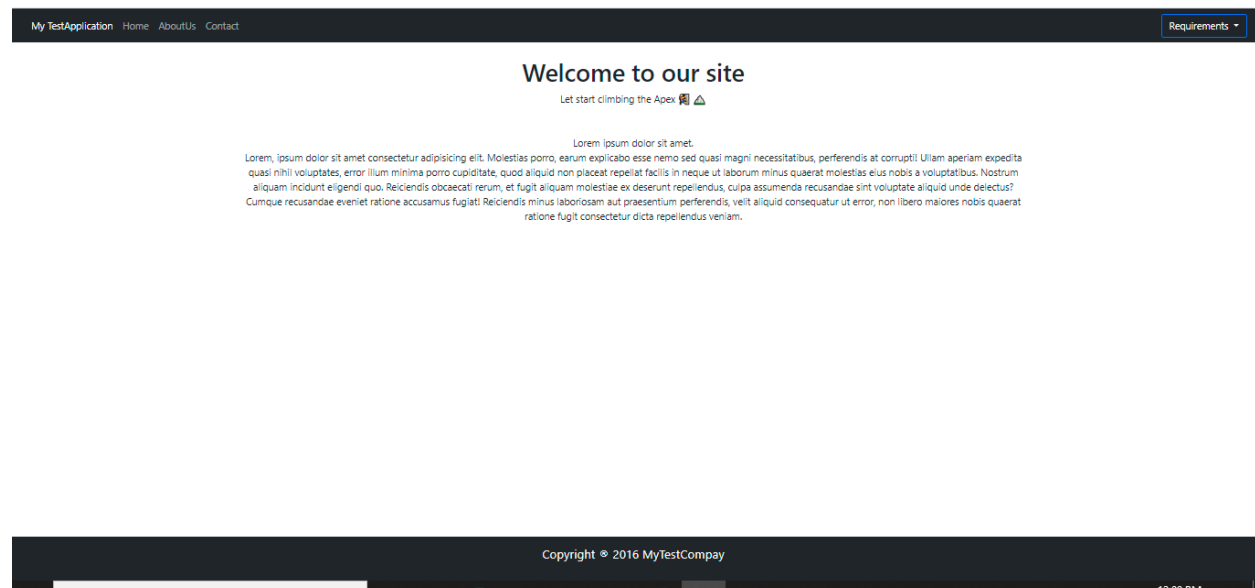
....

☐ Remember me

Sign in

Please include an '@' in the email address. 'fsf' is missing an '@'.

4. Design the page as per the requirement of the image below.



5. Design The blog without using header tag, use only predefined bootstrap classes.

Angular Blog

Build more accessible Angular apps Build more accessible Anular apps

This blog post shows a few different types of content that's supported and styled with Bootstrap. Basic typography, images, and code are supported.

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper. Aenean ultricies mi vitae est. Mauris placerat eleifend leo.

Pellentesque Habitant Morbi Tristique Senectus Et Netus Et Malesuada Fames Ac Turpis Egestas.

MALESUADA FAMES AC TURPIS EGESTAS.

lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer nec odio. praesent libero. sed cursus ante dapibus diam. sed nisi. nulla quis sem at nibh elementum imperdiet. Duis sagittis ipsum. praesent mauris. fusce nec tellus sed augue semper porta. mauris massa. vestibulum lacinia arcu eget nulla.

Heading

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer nec odio. Praesent libero. Sed cursus ante dapibus diam. Sed nisi. Nulla quis sem at nibh elementum imperdiet. Duis sagittis ipsum. Praesent mauris. Fusce nec tellus sed augue semper porta. Mauris massa. Vestibulum lacinia arcu eget nulla. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Curabitur sodales ligula in libero. Sed dignissim lacinia nunc. Curabitur tortor. Pellentesque nibh. Aenean quam. In scelerisque sem at dolor. Maecenas mattis. Sed convallis tristique sem. Proin ut ligula vel nunc egestas porttitor. Morbi lectus risus, iaculis vel, suscipit quis, luctus non, massa. Fusce ac turpis quis ligula lacinia aliquet. Mauris ipsum. Nulla metus metus, ullamcorper vel, tincidunt sed, euismod in, nibh. Quisque volutpat condimentum velit. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos.

Sub-heading

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer nec odio. Praesent libero. Sed cursus ante dapibus diam. Sed nisi. Nulla quis sem at nibh elementum imperdiet. Duis sagittis ipsum. Praesent mauris. Fusce nec tellus sed augue semper porta. Mauris massa. Vestibulum lacinia arcu eget nulla. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Curabitur sodales ligula in libero.

Example Code Block: `console.log("Testing Code Element in HTML");`

6. Design the buttons using predefined bootstrap classes

Button Styles can be applied to anchor tag, input button, button

Link Input Button

Options

Default Primary Success Info Warning Danger Link

Sizes

Large Button

Default Button

Small Button

Extra Small Button

State

Primary Active Button

Primary Disabled Button

7. Create the layout of element as per requirement below.

Three equal columns

Get Three equal-width columns with space between starting at desktop and scaling large desktop. On mobile devices, tables and below, columns will automatically stack.

Three equal-width columns

Three equal-width columns

Three equal-width columns

Three unequal columns

Get Three equal-width columns starting at desktop and scaling large desktop. On mobile devices, tables and below, columns will automatically stack.

.col md-3

.col md-6

.col md-12

Two columns

Get two different columns and scaling to large desktops with in the same row

.col md-8

.col md-4

Full width, single, column

no grid classes are necessary for full-width elements

8. Define dropdown elements in different ways as per image below.

Single Dropdown

Select Course

Dropdown with anchor tag

JavaScript Courses ▾

Dropdown with Header

Programming Courses ▾

Dropdown with Divider

Programming Courses ▾

Dropdown with Disabled

Programming Courses ▾

Dropdown with Left Right

◀ Programming Courses

Programming Courses ▶

DropUp

Programming Courses ▲

9. Using requirement number 4 connect all of the webpages.

Objective: Familiarized with Bootstrap classed and different layout technique.

Note: Upload screen shots of your output and send your git link here at canvas

Tool: Visual Studio Code, [Bootstrap Documentation](#)Links to an external site.

Bootstrap Hands On 2

Instruction:

1. Design using bootstrap as per the requirement of the image below.

Background Utilities

This is default primary background
This is 75% opacity secondary background
This is 50% opacity info background
This is 25% danger success background
This is 10% opacity success background

2. Design the border using bootstrap as per the image below.

Borders Utilities



3. Create the table using bootstrap defining different color utilities as per image below.

Colors Utilities

Result	Class Utilities
Primary Default Text	<code>text-primary</code>
Danger with 50% opacity	<code>text-danger text-opacity-50</code>
Success with 25% opacity	<code>text-success text-opacity-25</code>
Light with 75% opacity	<code>text-light text-opacity-75</code>
This is muted Text	<code>text-muted</code>

4. Create the following responsive display layout as per requirement of image below.

Display Utilities

Display Inline Block vs Block



- A. Hide First 2 images in large and larger Screens.
- B. Hide 3rd image in medium Screen.
- C. 2nd Image is only visible in small Screen.

Objective: Familiarize in different bootstrap utilities

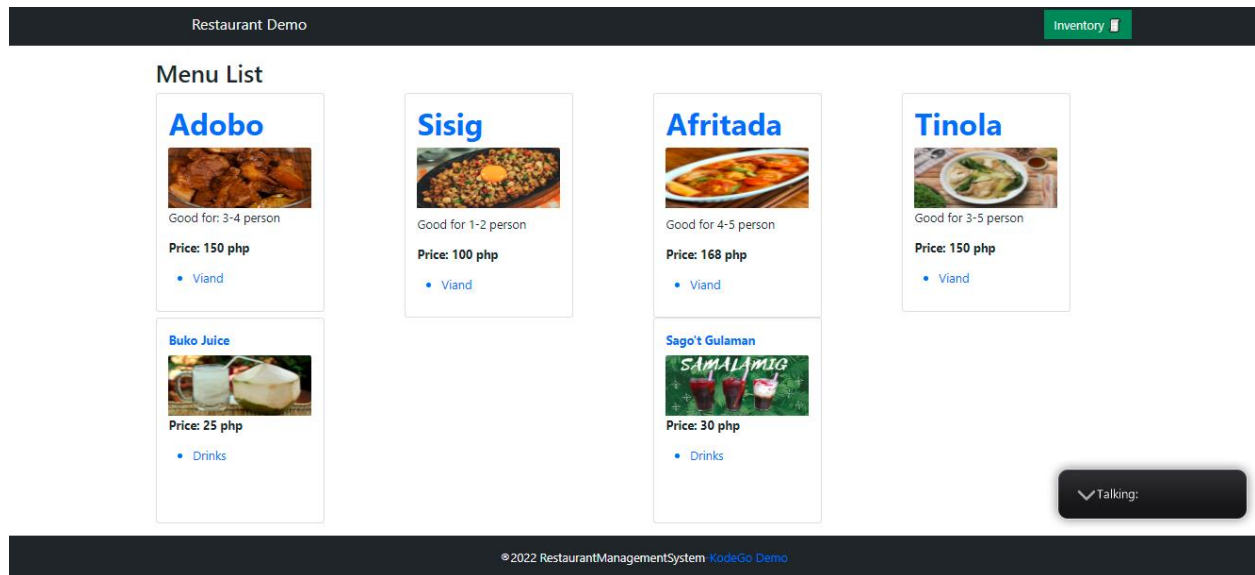
Note: Upload Screen shoot of your output and the gitlink here at canvas.

Tool: Visual Studio Code, [Bootstrap Documentation](#)Links to an external site.

Bootstrap Hands On 3

Instruction:

1. Create the webpage elements using bootstrap as per requirement of image below



2. Upon clicking inventory button modular component will appear. Create the modal component as per image below.

RestoDemo Records ×

List of Available items

Item ID	Item Name	Item Price	Item Category	Action
F01	Adobo	150	Viand	<div>Edit</div> <div>Delete</div>
F02	Sisig	100	Viand	<div>Edit</div> <div>Delete</div>
F03	Afritada	168	Viand	<div>Edit</div> <div>Delete</div>
F04	Tinola	150	Viand	<div>Edit</div> <div>Delete</div>
D01	Buko Juice	25	Drinks	<div>Edit</div> <div>Delete</div>
D02	Sago't Gulaman	30	Drinks	<div>Edit</div> <div>Delete</div>

click add button if you want to add more item

Add

Objective: Familiarized to the basic components of bootstrap such Card and Modal.

Note: Upload the screen shot of your output and git link here at canvas".

Tool; Visual Studio Code, [Bootstrap DocumentationLinks to an external site.](#)

Bootstrap Hands on 4

Instruction:

1. Used the existing project code from [WD37] Bootstrap Hands on 3 and add the following requirement.
2. Create login.html as per the requirement of the image below.

Login

Email

Password

3. Create a modal for signup as per the requirement of the image below.

SignUp

ApplicationUser

Email:

Enter Email

FullName

Enter Full Name

Password

Enter Password

Sign Up

Sing Up Sucessful, Thanks for registering with us! Go to [Login](#)

4. Create create.html as per the requirement of the image below.

note: Type of food and Choose city are dropdowns.

Create

Restaurant

Name

Type of Food

☐ Open for Online Orders?

LuanDate

Choose City:

[Create](#)

[Back to List](#)

5. Create edit.html as per the requirement of the image below.

Edit

Restaurant

Name

Type of Food

☐ Open for Online Orders?

LuanDate

Choose City:

[Update](#)

[Back to List](#)

6. Add links on your nav to connect every web pages.

Objective: Familiarized and defined the difference of form-group to form-control classes using bootstrap framework.

Note: Upload the screen shoot of your output and the git link here at canvas.

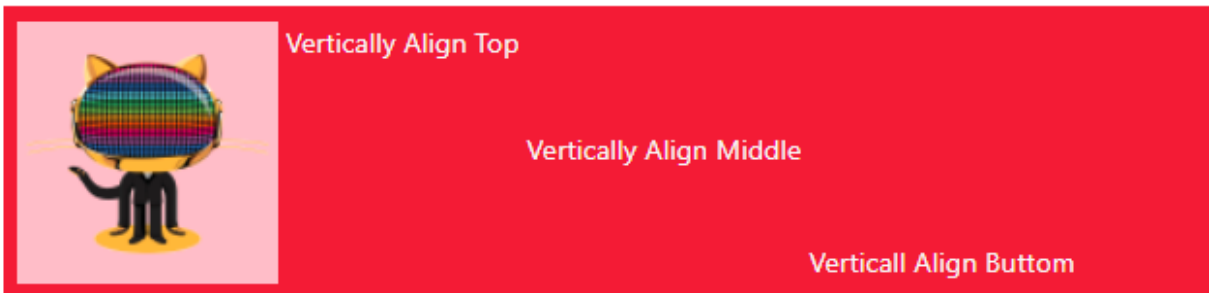
Tools: Visual Studio Code.

Bootstrap Hands On 5

Instruction:

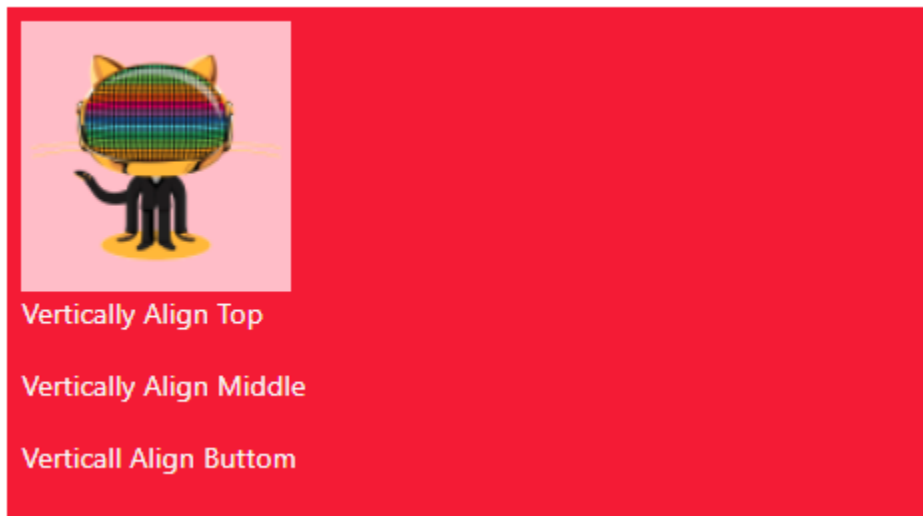
1. Design a webpage as per requirement of the image below.

Vertical Align Utilities



Responsive design Requirement: When the screen is small label will be displayed blocked

Vertical Align Utilities



2. Design a web page with the requirement shown in the image below.

Content Start



Content Middle



Content Between



Content Evenly



Content End



Responsive Design Requirement: When screen is small elements in all of the division will be centered and the first division element will have reverse order.

Flex Utilities

Content Start



Content Middle



Content Between

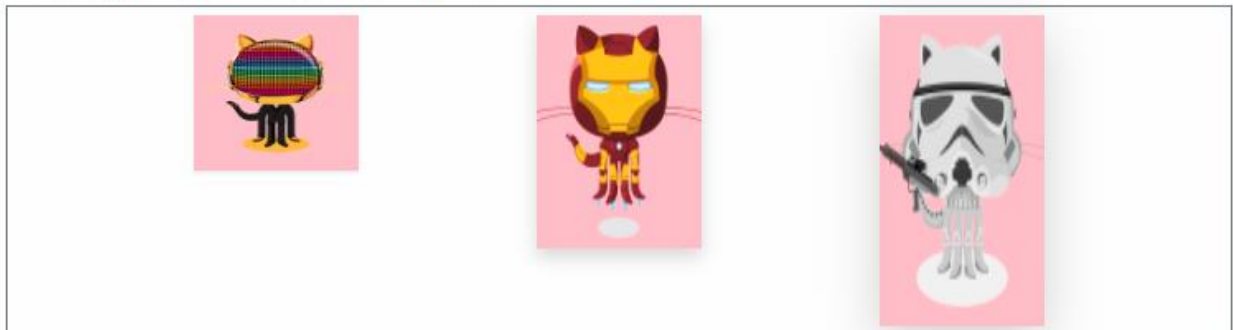


Content Evenly



3. Design a webpage as per requirement of the image below.

Sizing and Shadow Utilities



Objective: Familiarize in different bootstrap utilities

Note: Upload your output screen shoot and git link for the source here at canvas .

Tools: Visual Studio Code, [Bootstrap Documentation](#)Links to an external site.

JavaScript Algorithm 1

Instruction:

1. Create your [Account here.Links to an external site.](#) This website will help you to understand the javascript algorithm for predicting the output of every provided codes.
2. Complete the Level 1.
3. Create an Observation report in Pdf Format. Upload your observation report here at canvas.

Tools: [Algorithm WebsiteLinks to an external site.](#), Microsoft Word.

JavaScript Algorithm 2

Instruction:

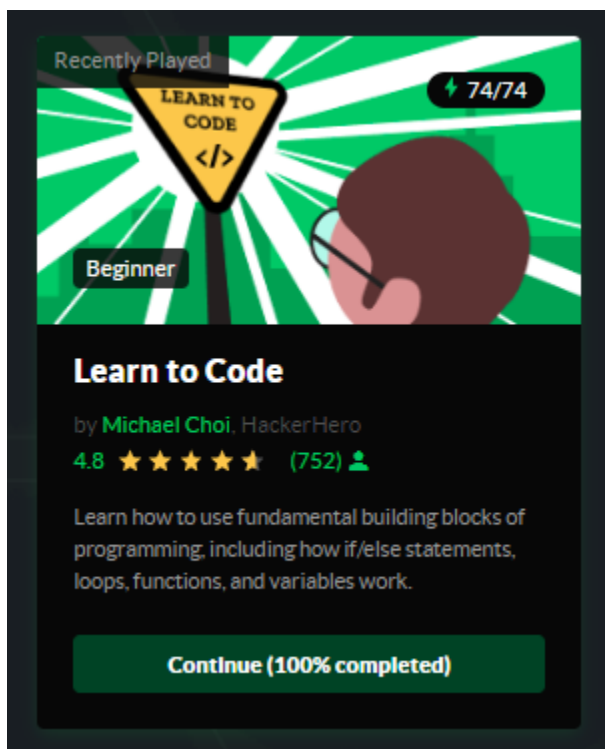
1. Create your [Account here. \(Links to an external site.\)](#) This website will help you to understand the javascript algorithm for predicting the output of every provided codes.
2. Complete the Level 2.
3. Create an Observation report in Pdf Format. Upload your observation report here at canvas.

Tools: [Algorithm Website \(Links to an external site.\)](#), Microsoft Word.

JavaScript Algoritihm 3

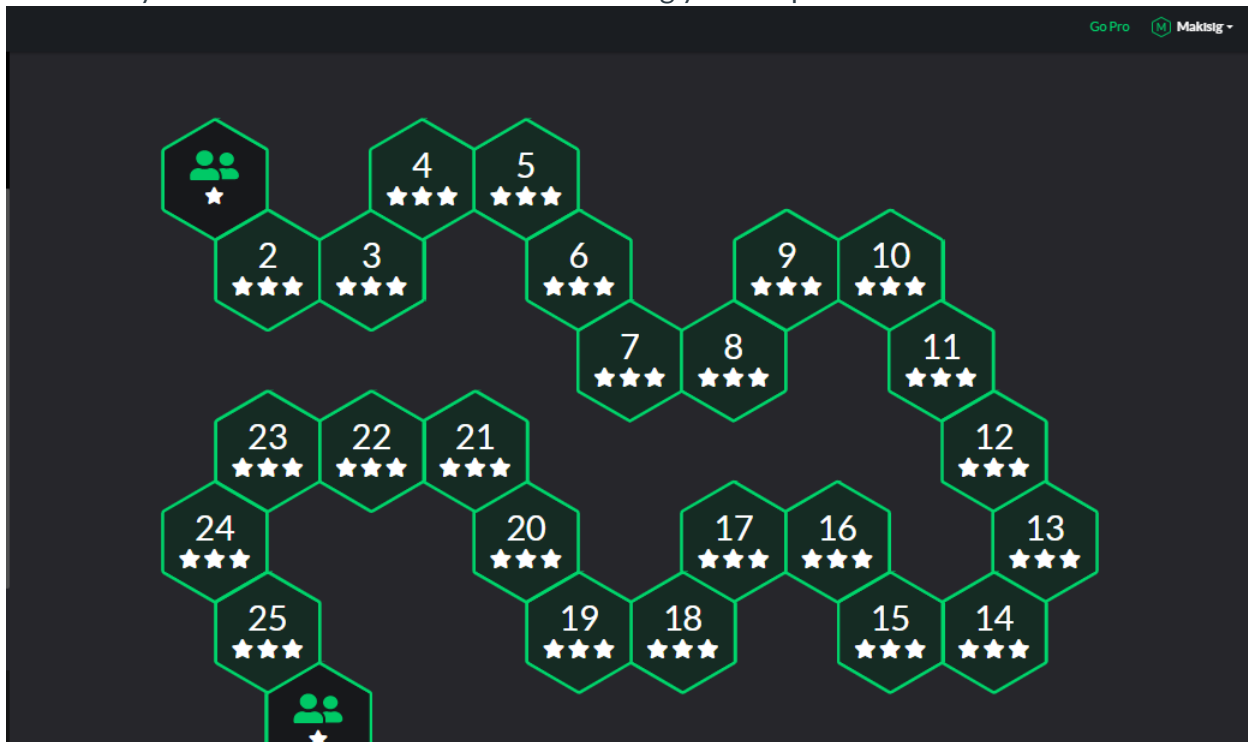
Instruction:

1. Create your [Account here. \(Links to an external site.\)](#) This website will help you to learn and understand javascript concept. It focuses on basic building blocks of computer programming such as variables, if/else statement, loops, and functions.
2. Complete the Learn to Code Challenge.



3. Send your screen shoot here at canvas showing you complete all the level.

3. Send your screen shoot here at canvas showing you complete all the level.



JavaScript HandsOn 1

1. Create a Js file that will show different Data Types(Number, String, Object). Used typeof keyword to show the data types.

example:

```
console.log("Number: ", numberVar, typeof numberVar);  
console.log("String: ", stringVar, typeof stringVar);  
console.log("Boolean: ", isTest , typeof isTest);  
console.log("Array: ", arrayOfColors , typeof arrayOfColor);  
console.log("Null Value: ", nullValue , typeof nullValue);
```

2. Create Js file to show different Mathematical Operations(+, -, *, /, **, %, ++, --).

3. Create Js File to show different Assignment Operators (=, +=, -=, *=, /=, %=, **=).

4. Create Js file to show different Logical Operators (&& , || , !).

5. Create Js file to program an if else statement and convert this to ternary operator.

6. Create Js file the has blank array, then push or assigned 5 elements inside the array, after you add items in the array loop and show all the elements of the array.

7. Create Js file to program if, else if , else statement.

8. Create Js file to program switch case statement.

9. Create an observation report for the every output save it in pdf format and upload here at canvas.

Tools: Visual Studio Code, Microsoft Office.

JavaScript HandsOn 2

Instruction: Answer the following Questions/Problems using JavaScript syntax. Set of possible output is provided for your better understanding of the questions. Compile and run your answer one js file per question upload your solution here at Canvas and in Git for your Backup upload your screen shoot pdf of your output here at canvas.

Tool: Visual Studio Code

Objective: Improve basic algorithm skills. It focuses on basic building blocks of computer programming such as variables, if/else statement, loops, and functions.

1. Create a function that prints/logs all the integers from 1 to 20.

Test Cases (0/1)

- print1to20() to log 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

2. Create a function that prints/logs all the odd numbers from 3 to 20.

Test Cases (0/1)

- printOdd3to20() to log 3 5 7 9 11 13 15 17 19

3. Create a function that prints/logs all the even numbers from 4 to 22.

Test Cases (0/1)

- printEven4to22() to log 4 6 8 10 12 14 16 18 20 22

4. Print/log all the multiples of 7 between the numbers 7 to 62.

Test Cases (0/1)

- multiplesOf7() to log 7 14 21 28 35 42 49 56

5. Log positive numbers starting at 50, counting down by fives (exclude 0).

Test Cases (0/1)

- countdownByFives() to log 50 45 40 35 30 25 20 15 10 5

6. Given an array, print/log the sum of the first value in the array, plus the array's length. Assume that the array is composed of numbers.

Test Cases (0/4)

- `firstPlusLength([1,2,5])` to log 4
- `firstPlusLength([3,0,2,5])` to log 7
- `firstPlusLength([-5,0,2,5])` to log -1
- `firstPlusLength([1])` to log 2

7. Given an array, write a function that prints the index value of its positive values.

For example, `printPositiveIndex([1, 3, -10])`, have it print/log 0, 1 (as the 0th index had a positive value and index 1 also had a positive value).

`printPositiveIndex([10, 5, -5, 15])` should print/log 0, 1, and 3. In other words, it prints the index of each positive number in the array.

Test Cases (0/3)

- `printPositiveIndex([1, 3, -10])` to log 0 1
- `printPositiveIndex([10, 5, -5, 15])` to log 0 1 3
- `printPositiveIndex([10, 5, 5, 15])` to log 0 1 2 3

JavaScript Hands On 3

Instruction:

1. Write a function `displayStudentName()` that will get the user input(Student Name) and display it as a paragraph element in the browser.
2. Add 8 Input field for the 8 subjects.
3. Write function `getAverage()` that will compute the average of the given 8 subjects. Display the average in the console log.
4. Write function `getFailedGrade()` that will Get the grade below 75
5. Display the name of the subjects and grade below 75 print the result as console log.
6. Write function `getPassedGrade()` that will get grade equal and above 75.
7. Display the name of the subjects and grade above and equal 75 print the result as console log .
8. Check the Average grade if the 68 below Display on log "Retake/Failed" , 69 - 74 Display on log "Remedial", 75 - 100 Display on log "Passed", 50 below and 100 above Display on log "Invalid".
9. All of this function will be invoke in one button "Compute Grade". Alert the average grade and name of the student.

Note: Upload pdf file that will capture your output here at Canvas. Save your file in your git folder.

Objective: Working with this hands on will give you an experienced working with any dynamic language requires the ability to read, process and output user data. JavaScript is especially useful when you want to take user information and process it without sending the data back to the server.

Tools: Visual Studio Code

SASS Hands on 1

1. Use your html css code in [HTML and CSS Hands On 5](#) and rewrite your code using SASS.

Note: Upload your file at your git hub account send the link and screen shoot of your output.

Tool: Visual Studio Code

Node Js Hands On 1

Instruction: Create a Javascript functions to using Object DataType with the following Module Requirements:

1. Create a getInfo.js for the module of creating an object with "name:StringDataType" and "age:NumberDataType" as "key: value" pairs.
2. Create a checkAge.js for the module of checking "age" morethan 18 then key:value "isLegalAge:True" will be added hence "isLegalAge:False".
3. Create an index.js to execute modules as per requirements.

Tool: Visual Studio Code.

Note: Upload a screen shoot of your output and the git link for source code.

Node JS Hands On 2

Instruction:

1. Create a get function with the given json Array.

```
let data = [  
  { id: 1, title: 'Create a project', order: 1, completed: true, author: 'Diego Laura' },  
  { id: 2, title: 'Take a coff   ', order: 2, completed: true, author: 'Laong-Laan' },  
  { id: 3, title: 'Write new article', order: 3, completed: true, author: 'Agap-ito  
Bagumbayan' },  
  { id: 4, title: 'Walk toward home', order: 4, completed: false, author: 'Taga-Ilog' },  
  { id: 5, title: 'Have some dinner', order: 5, completed: false, author: 'Dimas-Illaw' },  
];
```

2. Create a get function using id
3. Create a post function using params and body (id, title, order, author)
4. Create a put function using id send status 404/message if not existing id
5. Create a delete function using id send status 404/message if not existing id

Objective: Understanding the Basic Concept of NodeJs and Express

Note: Upload pdf observation of Screen Shoots of Postman request and link of git hub for source code

Tools: VisualStudioCode, NodeJs, PostMan

Node JS Hands on 3

Instruction:

1. Create FrontEnd for Node Js hands on 2.

*add linkBookImage:'url' properties on data array of objects

example:

```
let data = [  
  { id: 1, title: 'Create a project', order: 1, completed: true, author: 'Diego  
  Laura' ,linkBookImage:'url' }  
];
```

2. Home page has hero section.

3. About page.

4. Gallery minimum 10 image design it using bootstrap or css.

5. Create Nav links to connect every webpages.

6. Create Functional CRUD(With this requirement you need to create page to show all the records and add new record, page to view one item by id, page to edit the record).

Note: Upload Screen Shoots of your output and git link for the source code.

Tools: Visual Studio Code

NodeJs Hands on 4

Instruction:

1. Create FrontEnd for Node Js hands on 2.
2. Home page has hero section.
3. About page .
4. Gallery minimum 10 image design it using bootstrap or css.
5. Create Nav links to connect every webpages.
6. Create Functional CRUD using MongoDB and MVC pattern

Note: Upload Screen Shoots of your output and git link for the source code.

Tools: Visual Studio Code

NodeJs Hands on 5

Instruction:

1. Create Forms for Login and Registration of Users.
2. Build Validation and Authentication for these forms.
3. Use the source code on Hands on 4 to show the pages upon successful login.

This are the module you might need to build this app:

```
"dependencies": {  
  "bcryptjs": "^2.4.3",  
  "dotenv": "^16.0.2",  
  "ejs": "^3.1.8",  
  "express": "^4.18.1",  
  "joi": "^17.6.0",  
  "jsonwebtoken": "^8.5.1",  
  "mongoose": "^6.6.0",  
  "nodemon": "^2.0.19"  
}
```

Note: Upload screenshot of your output and git link for the source code.

Tools: Visual Studio Code

Node Js Hands On 6

Instruction:

1. Used NodeJs Hands On 5 for your source code
2. Add more items for your books min of 20 items.
3. Filter the items per page.
4. Create functionality by searching name of the books.

This are the module you might need to build this app:

```
"dependencies": {  
  "bcryptjs": "^2.4.3",  
  "dotenv": "^16.0.2",  
  "cors": "^2.8.5",  
  "ejs": "^3.1.8",  
  "express": "^4.18.1",  
  "joi": "^17.6.0",  
  "jsonwebtoken": "^8.5.1",  
  "mongoose": "^6.6.0",  
  "nodemon": "^2.0.19"  
}
```

Note: Upload screenshot of your output and git link for the source code.

Tools: Visual Studio Code

React Js Hands on 1

Instruction:

1. Create a new React Application with the name “my-first-react”, Run the application to print “welcome to the first session of React” as heading of that page.

Check Documentation: [Create React App \(create-react-app.dev\)](https://create-react-app.dev/)[Links to an external site.](#)
[Links to an external site.](#)

2. To create a React Application with the name of “my-first-react”, type the following command:

```
npx create-react-app my-first-react  
cd my-first-react
```

code .

3. Open the folder of my-first-react in Visual Studio Code

4. Open the App.js file in Src Folder of my-first-react. Remove the current content of “App.js”

Replace it with the following:

```
> JS App.js > ...  
import './App.css';  
function App() {  
  return (  
    <div className="App">  
      <h1>Welcome the First Session of React</h1>  
    </div>  
  );  
}  
export default App;
```

Run the following command to execute the React application:

```
npm start
```

10. Open a new browser window and type “localhost:3000” in the address bar.



Welcome the First Session of React

Objective: In this hands-on you will learn how to:

- Set up a react environment.
- Use create-react-app.

The following is required to complete this hands-on:

- Node.js
- NPM
- Visual Studio Code

Note: Zip your project to your git repository folder, upload a screen shot of your output and the link here at canvas.

React Js Hands on 2

Instruction:

Using the source code of your React Js Hands on 1 .

1. Create a Component RandomUser that handle a function that will generate five Random User.
2. Create a Component RandomAge that handle a function that will generate five Random Age.
3. Create a Component that for header
4. Create a Component that will handle the footer.
5. Import the Components in App.js
6. Create a function that will check if user's age is under 18 or not. If user's age is greater and equal to 18 display "Age: <Age> is Adult!" if not display "Age: <Age> too Young!"
7. Random User Name and Age will change every time page is refresh.

hint:

```
const randomName = ()=>{  
  const nameArr = ['Jhon Paul', 'Jon Rick', 'Jan Ferdie', 'Jan Joseph', 'Jan']  
  const index = Math.floor(Math.random() * 5);  
  return nameArr[index];  
}
```

Sample Output:

Jhon Paul
Age: 14 is too Young

Objectives: In this hands-on lab, you will learn how to:

- Understand JSX
- reusable Components

Note: Zip your project to your git repository folder, upload a screen shot of your output and the link here at canvas.

React Js Hands on 3

Instruction:

Continue working with your React Js Hands on 2 use it as your starting code.

1. Define UseState to handle the state of RandomName and RandomAge
2. Invoke the Function of RandomName And RandomAge using onclick event. RandomName and RandomAge will change every time change value button is click.
3. Using UseState change the theme of the page into Dark Mode, Light Mode and your Desired Color Pallete.

Objective: In this hands-on, you will learn how to:

- Create reusable components
- Apply style to components
- Used useState to render changes

Note: Zip your project to your git repository folder, upload a screen shot of your output and the link here at canvas.

React Js Hands on 4

Instruction:

1. Create a React App “counter-app” which will have a component named “CountPeople” which will have 2 methods.
2. LoginIncrement() → which will display the number of people who entered the mall.
3. ExitIncrement() → which will display the number of people who exited the mall.

```
const LoginIncrement = () => {  
  |   setLoginCount(LoginCount + 1);  
};  
  
const ExitIncrement = () => {  
  |   setExitCount(ExitCount + 1);  
};
```

4. Use useState to initialize states of number of login and exit.

```
const [LoginCount, setLoginCount] = useState(0);  
const [ExitCount, setExitCount] = useState(0);
```

5. The component has 2 buttons
Login → when clicked, the entrycount should get incremented by 1
Exit → when clicked, the exitcount should get incremented by 1
6. Output should look like this:

Login
11 People Entered!!!

Exit
13 People Left!!!

Objectives: In this hands-on lab, you will learn how to:

- Use React useState
- Rendering function using onclick event

Note: Zip your project to your git repository folder, upload a screen shot of your output and the link here at canvas.

React Js Hands on 5

Instruction:

Create a React Application “event-examples-app” to handle various events of the form elements in HTML.

1. Create “Increment” button to increase the value of the counter and “Decrement” button to decrease the value of the counter. The “Increase” button should invoke multiple methods(choose any event). "Say Welcome" button invoke function that will return " welcome" as an argument. "Click on me and Press Any in keyboard" will trigger the function in pressing any keyboard button.

Counter: 12

Increment
Decrement
Say Welcome
Click on me and Press Any in keyboard

Amount:

Currency:

Submit

Code Hint:

```
<button className="btn btn-info me-3 h3 col-3" onClick={increment}>  
  Increment  
</button>
```

```
<button className="btn btn-info me-3 h3 col-3" onClick={decrement}>  
  Decrement  
</button>
```

```
<button  
  className="btn btn-info me-3 h3 col-3"  
  onClick={() => sayWelcome("welcome")}  
>  
  Say Welcome  
</button>
```

```

<input
  className="btn btn-info me-3 h3 col-3"
  type="submit"
  value="Click on me and Press Any in keyboard"
  onKeyDown={() => clickMe("I was Invoked by pressing any keys")}
></input>

```

2. Create a “CurrencyConvertor” component which will convert the PH to USD when the Convert button is clicked.

Handle the Click event of the button to invoke the handleSubmit event and handle the conversion of the ph to usd.

Code Hint:

```

import React, {useState} from "react";
function ConvertCurrency() {
  const [val, setName] = useState(0);
  var usd = 0.018;
  const handleSubmit = event => {
    setName(event.target.value);
  };
  const logValue = (e) => {
    e.preventDefault();
    // test if val isCaptured
    console.log('val:', val);
    alert("Converting to USD Amount is "+(val*usd));
  };
  return (
    <div>
      Amount: <input type="number" onChange={handleSubmit}/><br/>
      Currency: <textarea value="Ph to USD"/><br/>
      <button className="btn" onClick={logValue}>Submit</button>
    </div>
  );
}
export default ConvertCurrency;

```

Objectives: In this hands-on lab, you will learn how to:

- Implement Event handling concept in React applications
- Use useState and from react library
- Use synthetic event

Note: Zip your project to your git repository folder, upload a screen shot of your output and the link here at canvas.

React Js Hands on 6

Instruction:

1. Create a React Application named "office-space-rental-app" which uses React JSX to create elements, attributes and renders DOM to display the page.
2. Create an element to display the heading of the page. Attribute to display the image of the office space
3. Create an object of office to display the details like Name, Rent and Address.

```
import React from "react";
import img1 from './Img/image0.jpg';
import img2 from './Img/image1.jpg';
import img3 from './Img/image2.jpg';
import DisplayDetails from './DisplayDetails';
const Items=(props)=>{
  const rentInfo = [
    {name:"Office Space 1", rentPrice:5000, address:"Manila",scr:img1},
    {name:"Office Space 2", rentPrice:6000, address:"New Delhi",scr:img2},
    {name:"Office Space 1", rentPrice:7000, address:"San Francisco",scr:img3}
  ];
  return(
    <section>
      <h1>Office Space, at Affordable Range</h1>
      {rentInfo.map((e)=>{
        <DisplayDetails image={e.scr} name={e.name} price={e.rentPrice} add={e.address}/>
      })}
    </section>
  );
};
export default Items;
```

4. Create a list of Object and loop through the office space item to display more data.
5. To apply Css, Display the color of the Rent in Red if it's below 60000 and in Green if it's above 60000.

```

.red {
  color: ■ red;
  font-family: monospace;
}

.green {
  color: ■ green;
  font-family: monospace;
}

body {
  padding-left: 40px;
}

.position {
  display: inline-block;
  margin: 10px;
}

img {
  border: 3px □ black solid;
}

```

```

const DisplayDetails=(props)=>(
  <div className="position">
    <img src={props.image} alt="imageItem"/>
    <p>Name:{props.name}</p>
    <p className={props.price<=6000? "red":"green"}>Rent:{props.price}</p>
    <p>Address:{props.add}</p>
  </div>
)
export default DisplayDetails;

```

6. Output look like this.

Office Space, at Affordable Range



Name:Office Space 1

Rent:5000

Address:Manila



Name:Office Space 2

Rent:6000

Address:New Delhi



Name:Office Space 1

Rent:7000

Address:San Francisco

Objectives: In this hands-on lab, you will learn how to:

- Use JSX syntax in React applications
- Use inline CSS in JSX

Note: Zip your project to your git repository folder, upload a screen shot of your output and the link here at canvas.

React Js Hands on 7

Instruction:

Create a react app for Student Management Portal named score-calculator-app and create a function component named "CalculateScore" which will accept Name, School, Total and goal in order to calculate the average score of a student and display this information.

1. Create a React project named "score-calculator-app" type the following command in terminal:

```
npx create-react-app my-first-react
cd my-first-react
```

code

2. Create a new folder under src folder with the name "Components". Add a new file named "CalculateScore.js"

```
import '../css/mystyle.css'
const percentToDecimal=(decimal)=>
{
  return(decimal.toFixed(3)+'%')
}
const calcScore=(total, goal)=>
{
  return percentToDecimal(total/goal)
}
const markScore=(total, goal)=>
{
  if((total/goal)<75){
    return "failed";
  }else return "passed";
}
```

```

export const CalculateScore=({Name, School, total, goal})=>
{
  return(
    <div className="formatstyle">
      <h1>Student Details</h1>
      <div className="Name">
        <b>Name:</b>
        <span>{Name}</span>
      </div>
      <div className="School">
        <b>School:</b>
        <span>{School}</span>
      </div>
      <div className="Total">
        <b>Total:</b>
        <span>{total}</span>
      </div>
      <div className="Score">
        <b>Score:</b>
        <span>{calcScore(total,goal)}</span>
        <span>{markScore(total,goal)}</span>
      </div>
    </div>
  )
}
export default CalculateScore;

```

3. Create a Folder named css and add a file named "mystyle.css" in order to add some styles to the components:

```

v src
  v Components
    JS CalculatorScore.js
  v css
    # mystyle.css

```

```

.Name {
  font-weight: 300;
  color: blue;
}

.School {
  color: crimson;
}

.Total {
  color: darkmagenta;
}

.formatstyle {
  text-align: center;
  font-size: large;
  font-family: Impact, Haettenschweiler, 'Arial Narrow Bold', sans-serif;
}

.Score {
  color: forestgreen;
}

span {
  font-weight: 200;
  font-family: cursive;
}

```

4. Edit the App.js to invoke the CalculateScore functional component as follows:

```

import CalculateScore from './Components/CalculatorScore';

function App() {
  return (
    <div className="App">
      <CalculateScore Name={"Alfren James"}
        School={"KodeGO"}
        total={284}
        goal={3}
      />
    </div>
  );
}
export default App;

```

5. Run your Project, Output would look like this:

Student Details

Name:Alfren James

School:KodeGO

Total:284

Score:94.667%passed

6. Create DisplayCalculateScore components and import CalculateScore Components and Functionality to get the input for "Name", "School", "Total" and "Goal".

7. Edit App.js and import DisplayCalculateScore instead of CalculateScore Components. Sample output:

Student Details

Name:User

School:KodeGO

Total:300

Score:100.000%passed

User Name:

School:

Total Score:

Goal:

Objective: In this hands-on, you will learn how to:

- Create a function component
- Apply style to components
- Render a component
- Apply Properties of components

Note: Zip your project to your git repository folder, upload screen shot of your output and the link here at canvas.

React Js Hands on 8

Instruction:

Create a React Application named "cricket-app"

Declare an array with 11 players and store details of their names and scores using the map feature of ES6.

Create Components for the Following:

```
const players = [  
  {name : "Maharlika", score : 50},  
  {name : "Kidlat", score : 70},  
  {name : "Shawn", score : 40},  
  {name : "Adbul", score : 61},  
  {name : "Dakila", score : 61},  
  {name : "Urduja", score : 50},  
  {name : "Makisig", score : 39},  
  {name : "Malakas", score : 84},  
  {name : "Alab", score : 64},  
  {name : "Amihan", score : 75},  
  {name : "Datu", score : 80}  
];
```

Create Components for the Following:

1. ListOfPlayers

```
const ListOfPlayers = ({players}) => {  
  return(  
    players.map((item) =>  
      <div>  
        <li>Player.{item.name} <span>{item.score}</span></li>  
      </div>  
    )  
  );  
}  
export default ListOfPlayers
```

2. PlayerScoreBelow70 . Which Filter the players with scores below 70 using arrow functions of ES6.

```
const PlayerScoreBelow70 = ({players}) => {  
  var playersBelow70 = players.filter(player => player.score <= 70);  
  return(  
    playersBelow70.map((player) =>  
      <div>  
        <li>Player.{player.name} <span>{player.score}</span></li>  
      </div>  
    )  
  );  
}  
export default PlayerScoreBelow70
```

3. ListOfPinoyPlayers

```
const ListOfPinoyPlayers = ({players}) => {  
  return(  
    players.map((item)=>{  
      return(  
        <div>  
          <li>Player Place: {item}</li>  
        </div>  
      )  
    })  
  )  
}  
export default ListOfPinoyPlayers
```

4. Create MainDisplay component to call to render the other components functionality. Display the Odd Team and Even Team players using the Destructuring features of ES6.

```
export function OddPlayers ([first, , third, , fifth]){
  return(
    <div>
      <li>1st: {first.name} {first.score}</li>
      <li>5th: {third.name} {third.score}</li>
      <li>7th: {fifth.name} {fifth.score}</li>
    </div>
  )
}
export function EvenPlayers ([, second, , fourth, , sixth]){
  return(
    <div>
      <li>2nd: {second.name} {second.score}</li>
      <li>6th: {fourth.name} {fourth.score}</li>
      <li>8th: {sixth.name} {sixth.score}</li>
    </div>
  )
}
```

5. Declare two arrays GetOddFilipinoPlayer and GetEvenFilipinoPlayer and merge the two arrays and display them using the Merge feature of ES6

```
//Destructuring
let a,b,c,d,e,rest;
[a,b,c,d,e,...rest] = players;

const FilipinoTeam = rest;

const GetOddFilipinoPlayer = ['1st Player','2nd Player',' 5thPlayer'];
const GetEvenFilipinoPlayer = ['6th Player',' 7thPlayer','8th Player'];
export const FilipinoPlayers = [...GetOddFilipinoPlayer,...GetEvenFilipinoPlayer]
```

Display these two components in the same home page using a simple if else by using useState to toggle the value of flag (changing the value to true - false and vice versa) by using clickEvent of button.

```

function MainDisplay() {
  const [flag, setFlag] = useState(true);
  if (flag === true) {
    return (
      <div>
        <button onClick={() => setFlag(false)}>Flag: True</button>
        <h1>List Of Players</h1>
        <ListOfPlayers players={players} />
        <h1>List of Player Below 70</h1>
        <PlayerScoreBelow70 players={players} />
      </div>
    );
  } else {
    return (
      <div>
        <button onClick={() => setFlag(true)}>Flag: False</button>
        <h1>Filipino Team</h1>
        <h1>Odd Players</h1>
        {OddPlayers(FilipinoTeam)}
        <h1>Even Players</h1>
        {EvenPlayers(FilipinoTeam)}
        <h1>List of Filipino Player Merge</h1>
        <ListOfPinoyPlayers players={FilipinoPlayers} />
      </div>
    );
  }
}
export default MainDisplay;

```

When flag === true
 Display All Players and Score
 Display All Players Score Below 70

Flag: True

List Of Players

- Player.Maharlika 50
- Player.Kidlat 70
- Player.Shawn 40
- Player.Adbul 61
- Player.Dakila 61
- Player.Urduja 50
- Player.Makisig 39
- Player.Malakas 84
- Player.Alab 64
- Player.Amihan 75
- Player.Datu 80

List of Player Below 70

- Player.Maharlika 50
- Player.Kidlat 70
- Player.Shawn 40
- Player.Adbul 61
- Player.Dakila 61
- Player.Urduja 50
- Player.Makisig 39
- Player.Alab 64

When Flag is false

Display Filipino Players(Name and Score) In Odd and Even Position

Display All Filipino Players and their Respective Positions

Flag: False

Filipino Team

Odd Players

- 1st: Urduja 50
- 5th: Malakas 84
- 7th: Amihan 75

Even Players

- 2nd: Makisig 39
- 6th: Alab 64
- 8th: Datu 80

List of Filipino Player Merge

- Player Place: 1st Player
- Player Place: 2nd Player
- Player Place: 5thPlayer
- Player Place: 6th Player
- Player Place: 7thPlayer
- Player Place: 8th Player

6. Import MainDisplay component to main App.js.

7. Start your Project and check the requirements if its qualifies the requirements.

Objective: In this hands-on lab, you will learn how to:

- Use map() method of ES6
- Apply arrow functions of ES6
- Implement Destructuring features of ES6

Note: Zip your project to your git repository folder, upload a screen shot of your output and the link here at canvas.

React Js Hands on 9

Instruction:

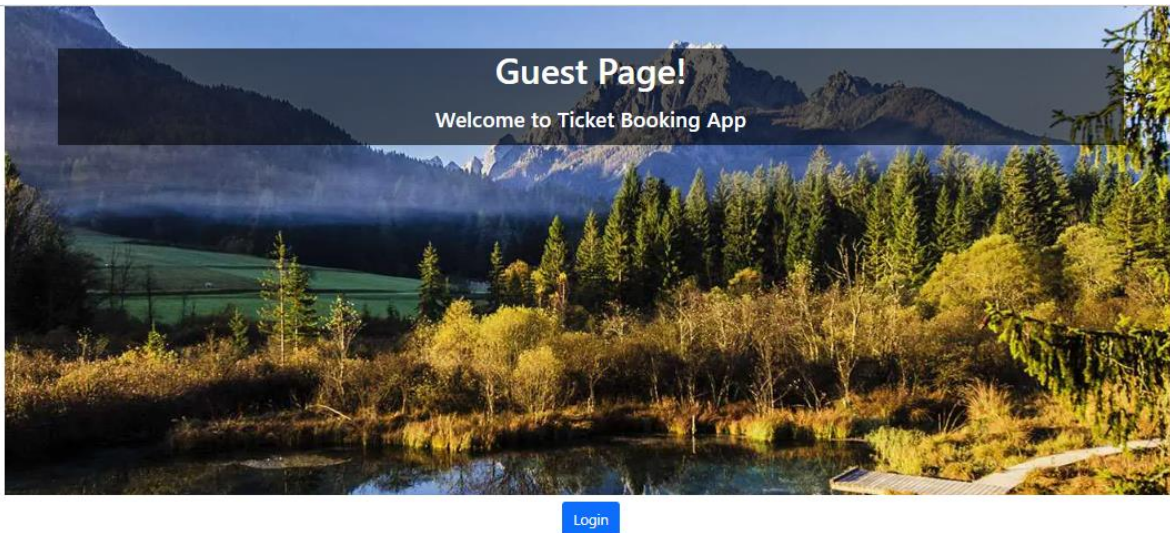
Create a React Application named “ticket-booking-app” where the user can browse the page where the flight details are displayed.

The Login and Logout buttons should accordingly display different pages. Once the user is logged in the User page should be displayed. When the user clicks on Logout, the Guest page should be displayed.

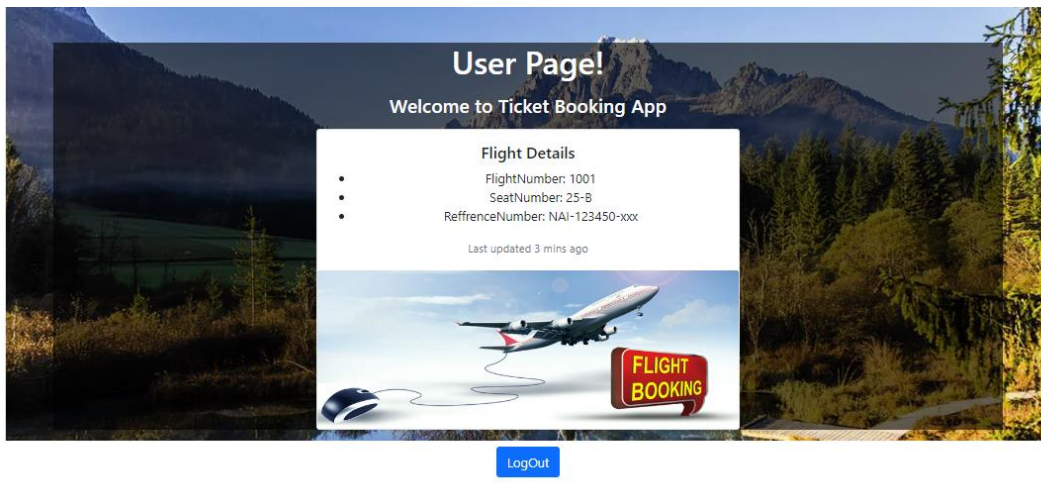
Note: No need to use React Router/Links.

Sample Output:

Guest Page



User Page



Hint:

1. Create Components for Greetings, LoggedInPage, LoggedOutPage,

Greeting Component will render Login and Logout Buttons to Toggle the Greeting
Greeting Property will depends if LoggedInPage or LoggedOutPage is true

```
const Greeting = (props) => {
  const isLoggedIn = props.isLoggedIn;
  if (isLoggedIn) {
    return <LoggedInPage />;
  }
  return <LoggedOutPage />;
};

const LoginButton = (props) => {
  return (
    <div className="container mt-2 d-flex justify-content-center">
      <button className="btn btn-primary" onClick={props.onClick}>
        LogIn
      </button>
    </div>
  );
};

const LogoutButton = (props) => {
  return (
    <div className="container mt-2 d-flex justify-content-center">
      <button className="btn btn-primary" onClick={props.onClick}>
        LogOut
      </button>
    </div>
  );
};
```

```
export default function LoggedOutPage() {
  return (
    <section>
      <div className="display-5">Guest Page!</div>
      <div className="display-6">Welcome to Ticket
        Booking App</div>
    </section>
  );
}
```

2. Create LoginControl component to render the value of the Buttons and set the values using useState and return the Greetings based on the state of button

```
import { LoginButton, LogoutButton, Greeting } from "../Greeting";
import { useState } from "react";

const LoginControl = () => {
  const [userLoggedIn, setUserLoggedIn] = useState(false);
  const isLoggedIn = userLoggedIn;

  let button = null;
  if (isLoggedIn) {
    button = <LogoutButton onClick={() => setUserLoggedIn(false)} />;
  } else {
    button = <LoginButton onClick={() => setUserLoggedIn(true)} />;
  }
  return (
    <main>
      <Greeting title={"HELLO"} isLoggedIn={isLoggedIn} />
      {button}
    </main>
  );
};

export default LoginControl;
```

3. Edit App.js to import and render the LoginControl components

Objectives: In this hands-on, you will learn how to:

- Implement conditional rendering in React applications
- React Property

- `useState`
- Used reusable React Functions

Note: Zip your project to your git repository folder, upload a screen shot of your output and the link here at canvas.

React Js Hands on 10

Show

Hide

Course Details

Title:Vue

4/5/2022

Title:React

6/3/2022

Book Details

Title:React Learning

670

Title:Deep Dive Vuex

800

Title:Mongo Essentials

450

Blog Details

Title:React Learning

Stephen Biz

Welcome to learning React

Title:Installation

Schewzdenier

You can Install React from npm

1. Create Components for CourseDetails, BookDetails and BlogDetails to map their props. You can designed different approached in showing the data like cards, table, list etc..
2. Create Page Component to render the CourseDetails, BookDetails and BlogDetails Components.

Objectives: In this hands-on, you will learn how to:

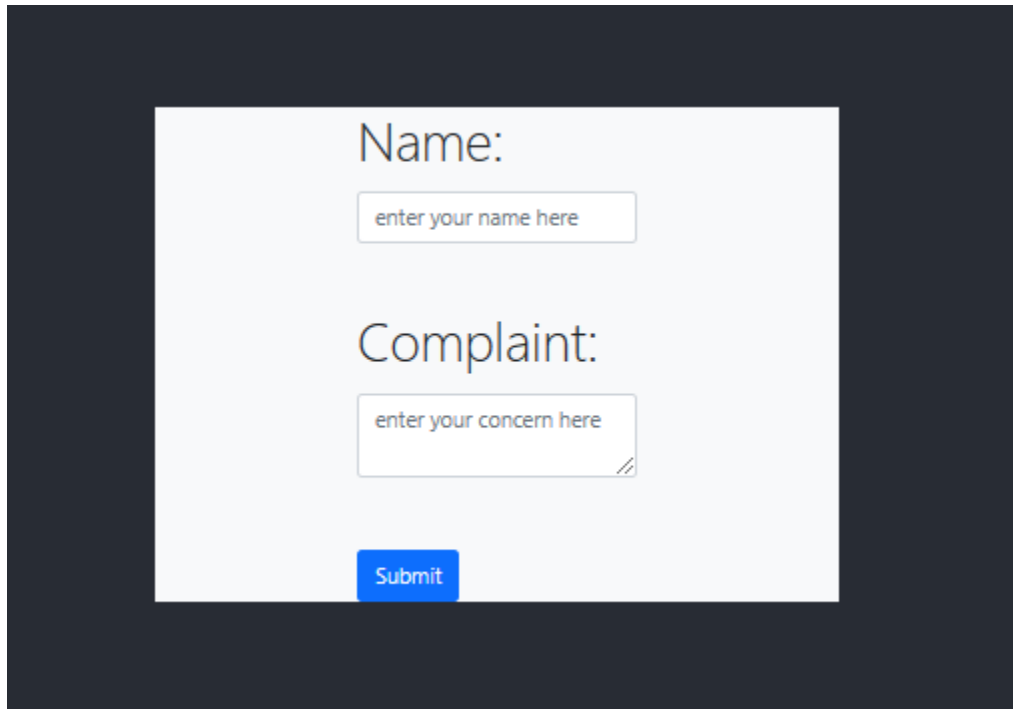
- Implement conditional rendering in React applications
- Use ReactDOM
- Use constructor
- Use this keyword
- React components and map() function

Note: Zip your project to your git repository folder, upload a screen shot of your output and the link here at canvas.

React Js Hands on 11

Instruction:

1 .Create a React App named “ticket-raising-app” which will help to raise a complaint and get it resolved.



Name:

Complaint:

Submit

The image shows a web application interface. At the top, there is a white alert box with a dark border. The text inside the alert box reads: "localhost:3000 says", "Thank You, John Paul", "Your Complaint Was Submitted.", and "Transaction ID is: 01676015168406". There is a blue "OK" button in the bottom right corner of the alert box. Below the alert box, there is a light gray form box. The form has two sections: "Name:" with a text input field containing "John Paul", and "Complaint:" with a text area containing "Slow Connection". At the bottom of the form box is a blue "Submit" button.

2. Create a component named "ComplaintRegister" with a form containing a textbox to enter the Customer name and a textarea to enter the complaint. Use "handleSubmit" event of the button to submit the complaint and generate a Reference number(Random Number) for further follow ups in the alert box.

handleSubmit Function invokes upon form submit

handleSubmit Function invokes upon form submit

```
function handleSubmit() {  
  alert(  
    "Thank You, " +inputName+ "\nYour Complaint  
    Was Submitted." +  
    "\nTransaction ID is: 0"+transactionId  
  );  
}
```

Function to generate random number using min max values

```
function getRandomNumberBetween(min, max) {  
  min = Math.ceil(min);  
  max = Math.floor(max);  
  return Math.floor(Math.random() * (max - min) +  
    min); // The maximum is exclusive and the minimum  
    is inclusive  
}
```

Function to get value of input name and set new value base on inputted string

```
function getName(e){  
  setInputName(e.target.value)  
}
```

Function to generate transaction Id based on date and random number this function will invoke upon clicking submit

```
function generateTransID(){  
  const randomNumber = getRandomNumberBetween(1,  
    100);  
  setTransactionId(Date.now() + randomNumber);  
}
```

2. Create form element that will that has Event handling for capture of change/value in the

input field and invoking the function via Submit form.

```
<div className="container mt-4 bg-light text-dark  
w-25 d-flex justify-content-center">  
  <form  
    onSubmit={handleSubmit}  
    className="text-start col-lg-auto g-3  
align-items-center"  
  >  
    <label>  
      <div className="mb-3">  
        <div className="display-6 mb-3">Name:</div>  
        <input  
          className="form-control mb-5"  
          type="text"  
          placeholder="enter your name here"  
          id="name"  
          onChange={getName}</input>  
      </div>  
      <div className="mb-3">  
        <div className="display-6  
mb-3">Complaint:</div>  
        <textarea  
          className="form-control mb-5"  
          placeholder="enter your concern here"  
        ></textarea>  
      </div>  
      <input onClick={generateTransID}  
        type="submit" value="Submit" className="btn  
btn-primary" />  
    </label>  
  </form>  
</div>
```

Note: You can Create your own function and refactor the structure of code as long the requirements were accomplished.

Objectives: In this hands-on, you will learn how to:

- Implement React forms
- Use various input controls like textbox, button and textarea.
- Implement logic and better user experienced.
- Use useState

Note: Zip your project to your git repository folder, upload a screen shot of your output and the link here at canvas.

React Js Hands on 12

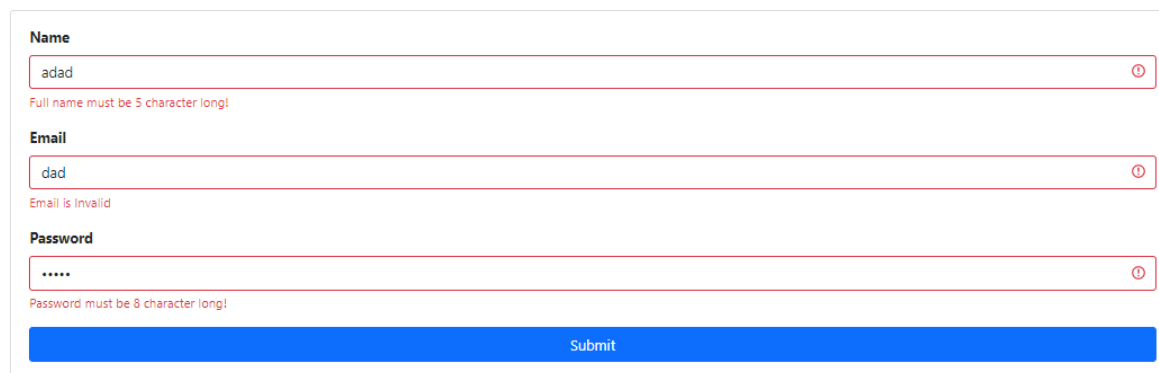
Instruction:

Create a React App named “mail-register-app” which will have a component named “Register.js”. Create a form which accepts the name, email and password and validate the fields as per the following:

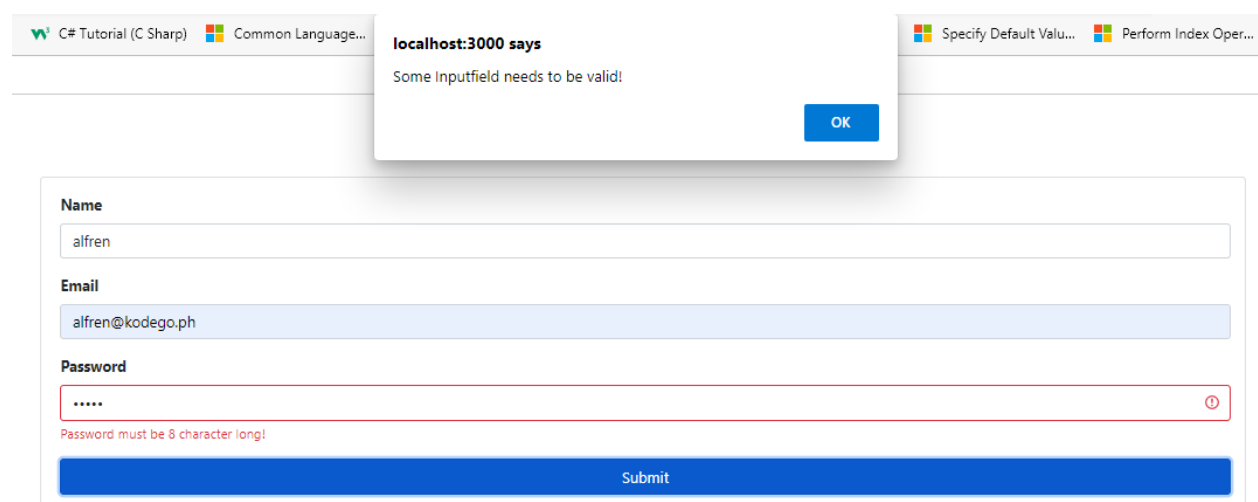
- 1.Name should have at least 5 characters
- 2.Email should have "@" and "."
3. Password should have at least 8 characters.

After clicking the Submit button alert will display if your have a valid form or some fields need to change as per the requirements.

Output:



The screenshot shows a registration form with three input fields: Name, Email, and Password. The Name field contains 'adad' and has a red border with a red error message 'Full name must be 5 character long!'. The Email field contains 'dad' and has a red border with a red error message 'Email is Invalid'. The Password field contains '*****' and has a red border with a red error message 'Password must be 8 character long!'. A blue 'Submit' button is at the bottom.



The screenshot shows the registration form with a validation alert. The alert box, titled 'localhost:3000 says', contains the message 'Some Inputfield needs to be valid!' and an 'OK' button. The form fields are: Name 'alfren', Email 'alfren@kodego.ph', and Password '*****'. The Password field has a red error message 'Password must be 8 character long!'. A blue 'Submit' button is at the bottom.

C# Tutorial (C Sharp)
Common Language...
localhost:3000 says
Some Inputfield needs to be valid!
OK
Specify Default Valu...
Perform Index Oper...

Name
alfren
Email
alfren@kodego.ph
Password

Password must be 8 character long!
Submit

C# Tutorial (C Sharp)
Common Language...
localhost:3000 says
Valid Form
OK
Specify Default Valu...
Perform Index Oper...

Name
alfren
Email
alfren@kodego.ph
Password

Submit

Code Hint:

Create useState for Valid form and capturing error message

```
//state of error
const [errorName, setErrName] =useState('')
const [errorEmail, setErrEmail] =useState('')
const [errorPass, setErrPass] =useState('')
const [valid, setValid] = useState(false)
//validate function
```

Validation Function:

```
const validateForm = (event) => {
  event.preventDefault();
  const { name, value } = event.target;
  const emailValid = RegExp(/^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$/);
  switch (name) {
    case 'name':
      if (value.length <= 4) {
        //add error message if length of name does not meet
        setErrName('Full name must be 5 character long!')
        setValid(false)
        //checking value if was able to get by onchange event, err capture
        and valid
        console.log(value, errorName, valid)
      } else {
        //set no error message if name length meet
        setErrName('')
        //checking value if was able to get by onchange event, err capture
        and valid
        console.log(value, errorName, valid)
      }
      break;
```

```
      break;
    case 'email':
      if (emailValid.test(value) === false) {
        setErrEmail('Email is Invalid');
        setValid(false)
      } else {
        setErrEmail('')
      }
      break;
    case 'password':
      if (value.length < 8 ) {
        setErrPass('Password must be 8 character long!')
        setValid(false)
      } else {
        setErrPass('')
      }
      break;
    default:
      break;
  }
  //if no error valid state will be true
  if(errorName===' ' && errorEmail===' ' &&errorPass===' '){
    setValid(true)
  }
};
```

onFormSubmit Function to check if the form inputs are valid

```
function onFormSubmit(){
  console.log(valid);
  if (!valid) {
    alert("Some Inputfeild needs to be valid!");
  } else {
    alert("Valid form");
  }
}
```

return form and take advantage of bootstrap class by using invalid-form

```
return (
  <div className="container">
    <div className="card mt-5">
      <form className="card-body" onSubmit={onFormSubmit}>
        <div className="form-group mb-3">
          <label className="mb-2">
            <strong>Name</strong>
          </label>
          <input
            required
            type="text"
            name="name"
            onChange={validateForm}
            className={
              errorName.length > 0
                ? "is-invalid form-control"
                : "form-control"
            }
          />
          {errorName.length > 0 && (
            <span className="invalid-feedback">{errorName}</span>
          )}
        </div>
      </form>
    </div>
  </div>
)
```

```

</div>
<div className="form-group mb-3">
  <label className="mb-2">
    <strong>Email</strong>
  </label>
  <input
    required
    type="text"
    name="email"
    onChange={validateForm}
    className={
      errorEmail.length > 0
        ? "is-invalid form-control"
        : "form-control"
    }
  />
  {errorEmail.length > 0 && (
    <span className="invalid-feedback">{errorEmail}</span>
  )}
</div>

```

```

    <div className="form-group mb-3">
      <label className="mb-2">
        <strong>Password</strong>
      </label>
      <input
        required
        type="text"
        name="password"
        onChange={validateForm}
        className={
          errorPass.length > 0
            ? "is-invalid form-control"
            : "form-control"
          }
      />
      {errorPass.length > 0 && (
        <span className="invalid-feedback">{errorPass}</span>
      )}
    </div>
    <button type="submit" class="btn btn-primary">
      Submit
    </button>
  </form>
</div>
</div>
);

```

Objectives: In this hands-on, you will learn how to:

- Implement React forms validation
- Use Regular Expression
- Use various input controls using bootstrap or other css framework

Note: Zip your project to your git repository folder, upload a screen shot of your output and the link here at canvas.

React Js Hands on 13

Instruction:

1. Create a React Application “fetch-user-app” which will retrieve the user details from <https://random-data-api.com/api/v2/users> Links to an external site. or use this <https://randomuser.me/api/>

and display the title, firstname ,image, mobile, email and birthday of a user.



Tip: need to install date-fns for date format.

2.Create a component named “GetUser” and in the asynchronous method “getUser()” invoke the URL using fetch method. You may also test the response of 3rd party api url using postman or other api testing software.

3. Render or return the html elements using bootstrap or other css framework in react

```

const url = "https://random-data-api.com/api/v2/users"
const [user, setUser] = useState({})

const getUser = async () => {
  try {
    const response = await fetch(url, { mode: "cors" })
    const data = await response.json()

    setUser(data)
    console.log(user)
  } catch (error) {
    console.log(error)
  }
}

useEffect(() => {
  getUser()
}, [])

const { first_name, last_name, avatar, email, date_of_birth, phone_number } =
  user

```

Code Snippet in GetUser Component:

3, Create a button that will invoke function to get random user.

Objectives: In this hands-on, you will learn how to:

- Construct a React application that invokes the REST API and fetch data from the API
- Display the fetch api to the elements.
- Used date-fns format

Note: Zip your project to your git repository folder, upload a screen shot of your output and the link here at canvas.

SQL Hands On 1

Instruction:

1. Create a table as per the requirement of the image below:

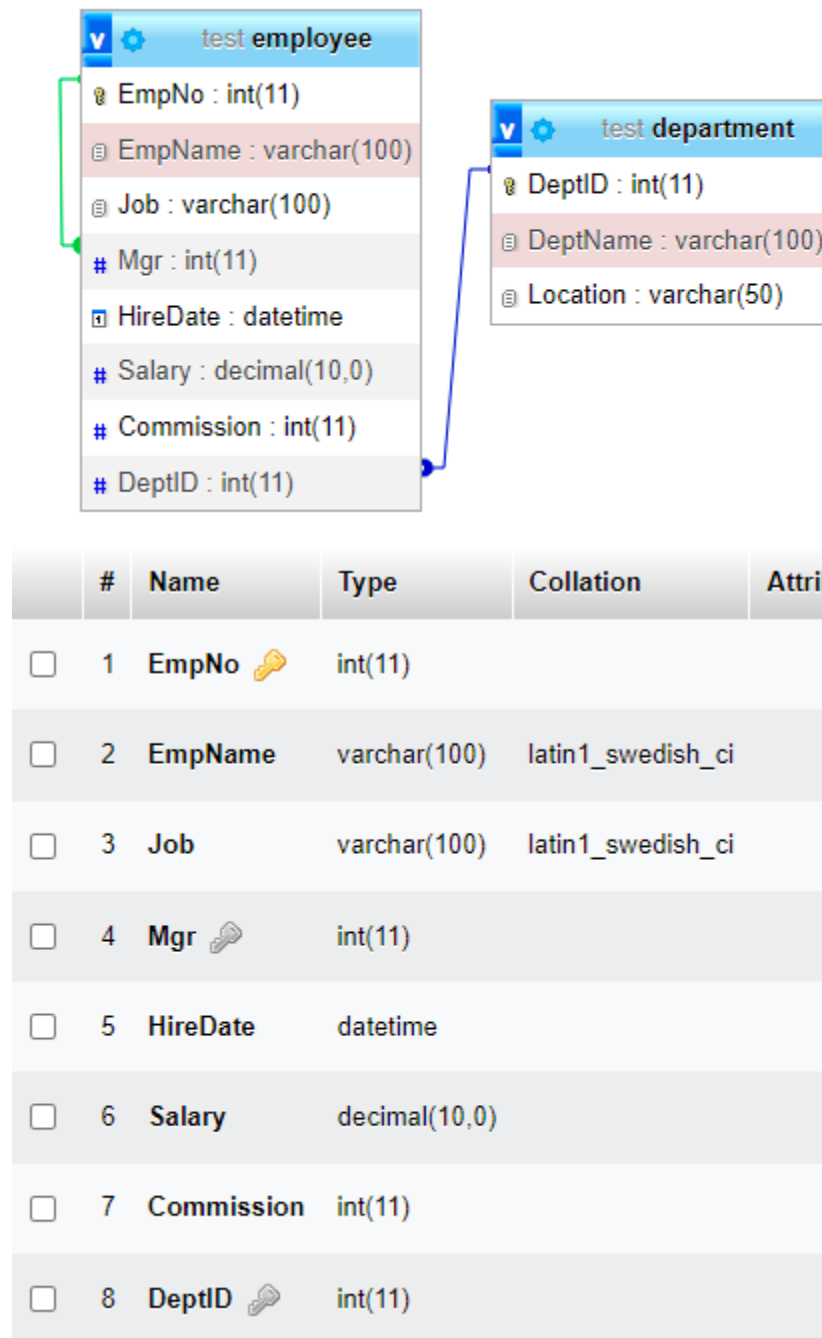
employee table

EmpNo	EmpName	Job	Mgr	HireDate	Salary	Commission	DeptID
1001	Scott	President	NULL	1978-01-01 00:00:00	5000	NULL	1001
1058	Clark	Manager	1001	1978-01-01 00:00:00	4000	NULL	1001
1059	Jeff	Manager	1001	1978-01-01 00:00:00	3500	500	1002
1060	Marko	Manager	1001	1978-01-01 00:00:00	4000	NULL	1003
1065	Bryan	Salesman	1060	1979-02-04 00:00:00	3000	300	1002
1066	Frauline	Salesman	1060	1978-02-08 00:00:00	4000	600	1002
1067	Phylip	Salesman	1060	1978-04-15 00:00:00	3300	0	1002
1068	Ejerson	Clerk	1060	1978-01-15 00:00:00	2400	NULL	1002
1069	Julie	Clerk	1065	1983-05-12 00:00:00	2000	NULL	1003
1070	Russel	Analyst	1065	1979-03-01 00:00:00	3400	NULL	1003
1071	Buboy	Analyst	1065	1979-03-01 00:00:00	3600	NULL	1003
1072	Joseph	Analyst	1065	1981-01-09 00:00:00	3100	NULL	1003

department table

▼	DeptID	DeptName	Location
e	1001	Executive	BGC
e	1002	Hr	Mandaluyong
e	1003	IT	Bulacan
e	1004	Sales	Y Ami
e	1005	Marketing	Jolo

Table requirement and constraint



Note: Upload screen shoot of your output and git hub link of sql commands

Tool: Xamp PhpMyAdmin app

Instruction: Using the Records from Hands on 1 create queries as per the requirement of the given output.

1. Display all employee working with the president.

EmpNo	EmpName	Job	Mgr	HireDate	Salary	Commission	DeptID
1058	Clark	Manager	1001	1978-01-01 00:00:00	4000	NULL	1001
1059	Jeff	Manager	1001	1978-01-01 00:00:00	3500	500	1002
1060	Marko	Manager	1001	1978-01-01 00:00:00	4000	NULL	1003

2. Create a query to Show all Employees working as Manager

EmpNo	EmpName	Job	Mgr	HireDate	Salary	Commission	DeptID
1058	Clark	Manager	1001	1978-01-01 00:00:00	4000	NULL	1001
1059	Jeff	Manager	1001	1978-01-01 00:00:00	3500	500	1002
1060	Marko	Manager	1001	1978-01-01 00:00:00	4000	NULL	1003

3. Create a query to Show all Employees Earning more than 2500.

EmpNo	EmpName	Job	Mgr	HireDate	Salary	Commission	DeptID
1001	Scott	President	NULL	1978-01-01 00:00:00	5000	NULL	1001
1058	Clark	Manager	1001	1978-01-01 00:00:00	4000	NULL	1001
1059	Jeff	Manager	1001	1978-01-01 00:00:00	3500	500	1002
1060	Marko	Manager	1001	1978-01-01 00:00:00	4000	NULL	1003
1065	Bryan	Salesman	1060	1979-02-04 00:00:00	3000	300	1002
1066	Frauline	Salesman	1060	1978-02-08 00:00:00	4000	600	1002
1067	Phylip	Salesman	1060	1978-04-15 00:00:00	3300	0	1002
1070	Russel	Analyst	1065	1979-03-01 00:00:00	3400	NULL	1003
1071	Buboy	Analyst	1065	1979-03-01 00:00:00	3600	NULL	1003
1072	Joseph	Analyst	1065	1981-01-09 00:00:00	3100	NULL	1003

4. Create a query to display employees without manager.

EmpNo	EmpName	Job	Mgr	HireDate	Salary	Commission	DeptID
1001	Scott	President	NULL	1978-01-01 00:00:00	5000	NULL	1001

5. Create a query to display employees working on Hr Department.

EmpNo	EmpName	Job	Mgr	HireDate	Salary	Commission	DeptID
1059	Jeff	Manager	1001	1978-01-01 00:00:00	3500	500	1002
1065	Bryan	Salesman	1060	1979-02-04 00:00:00	3000	300	1002
1066	Frauline	Salesman	1060	1978-02-08 00:00:00	4000	600	1002
1067	Phylip	Salesman	1060	1978-04-15 00:00:00	3300	0	1002
1068	Ejerson	Clerk	1060	1978-01-15 00:00:00	2400	NULL	1002

6. Create a query to display employees working on IT Department.

EmpNo	EmpName	Job	Mgr	HireDate	Salary	Commission	DeptID
1060	Marko	Manager	1001	1978-01-01 00:00:00	4000	NULL	1003
1069	Julie	Clerk	1065	1983-05-12 00:00:00	2000	NULL	1003
1070	Russel	Analyst	1065	1979-03-01 00:00:00	3400	NULL	1003
1071	Buboy	Analyst	1065	1979-03-01 00:00:00	3600	NULL	1003
1072	Joseph	Analyst	1065	1981-01-09 00:00:00	3100	NULL	1003

7. Create a query to display employees earning more than 2500 and less than 5000.

EmpNo	EmpName	Job	Mgr	HireDate	Salary	Commission	DeptID
1058	Clark	Manager	1001	1978-01-01 00:00:00	4000	NULL	1001
1059	Jeff	Manager	1001	1978-01-01 00:00:00	3500	500	1002
1060	Marko	Manager	1001	1978-01-01 00:00:00	4000	NULL	1003
1065	Bryan	Salesman	1060	1979-02-04 00:00:00	3000	300	1002
1066	Frauline	Salesman	1060	1978-02-08 00:00:00	4000	600	1002
1067	Phylip	Salesman	1060	1978-04-15 00:00:00	3300	0	1002
1070	Russel	Analyst	1065	1979-03-01 00:00:00	3400	NULL	1003
1071	Buboy	Analyst	1065	1979-03-01 00:00:00	3600	NULL	1003
1072	Joseph	Analyst	1065	1981-01-09 00:00:00	3100	NULL	1003

8. Create a query to display employees whose jobs is Sales or Clerk.

EmpNo	EmpName	Job	Mgr	HireDate	Salary	Commission	DeptID
1065	Bryan	Salesman	1060	1979-02-04 00:00:00	3000	300	1002
1066	Frauline	Salesman	1060	1978-02-08 00:00:00	4000	600	1002
1067	Phylip	Salesman	1060	1978-04-15 00:00:00	3300	0	1002
1068	Ejerson	Clerk	1060	1978-01-15 00:00:00	2400	NULL	1002
1069	Julie	Clerk	1065	1983-05-12 00:00:00	2000	NULL	1003

9. Create a query to display employees joined in 1979-03-01.

EmpNo	EmpName	Job	Mgr	HireDate	Salary	Commission	DeptID
1070	Russel	Analyst	1065	1979-03-01 00:00:00	3400	NULL	1003
1071	Buboy	Analyst	1065	1979-03-01 00:00:00	3600	NULL	1003

10. Create a query to display employees joined after 1979-03-01.

EmpNo	EmpName	Job	Mgr	HireDate	Salary	Commission	DeptID
1069	Julie	Clerk	1065	1983-05-12 00:00:00	2000	NULL	1003
1072	Joseph	Analyst	1065	1981-01-09 00:00:00	3100	NULL	1003

Note: Upload the screenshot of your output and the git link of SQL commands

SQL Hands On 3

Instruction:

1. Create a query to get all employees who have more than 25 years of experience.
2. Create a query to get employees whose names are more than 5 characters long.
3. Create a query to display total salary combined with commission of all employees.
4. Create a query to display annual salary of all employees.
5. Create a query to display current Date.
6. Create a query to get all employees who don't get commission.
7. Create a query to get all employees whose name contains character 'a'
8. Create a query to Get All employees working with the president whose name ends with 'o'.
9. Create a query to Get the Min , Max and Average Salary of all the employees.
10. Create a query to get the department details of employees working at BGC.
11. Create a query to Get the Min , Max and Average Salary of all the employees working in Marketing Department.
12. Create a query to get the department wise maximum, minimum and average salary on employee table(Hint: use Inner Join).
13. Create a query to count total number of employees.
14. Create a query to display distinct salaries in employee table.
15. Create a query to get all employees whose names length is greater than 3 characters and less than 5.
16. Create a query to get all employees who are getting salary 2000, 3000, 4000, 5000.
17. Create a query to get the top 5 highest paid employee.
18. Create a query to get the least 5 salaries of employee table.
19. Create a query to get the total no departments in the organization.
20. Create a query to get the Highest paid department with the average salary.

Note: Upload screenshot of your output and Git link for your source code.

SQL Hands On 4

Instruction:

1. Write a query to get employee name, department name(Hint: join).
2. Write a query to get the highest paid employee details in organization.(Hint: sub query <select inside select>)
3. Write a query to get all department details whose average is more than 2500(Hint: Join, group by, having).
4. Write a query to display department name and count number of employees per department. (Hint: inner join, group by, count).
5. Write a query to display employee name, department and location.(Hint: join, alias)
6. Write a query to display employee name and his/her manager.(Hint: join, alias)
7. Write a query to display total no of employee joined on 1978-01-01.
8. Write a query to get the list of department name and its total salary in the organization display it in highest to least salary.
9. Write a query to get the employee name, department name including employees who are not assigned to any department(Hint: outer join).
10. Write a query to get employee name, department name including departments where no employee is working yet(outer join).

Note: Upload screenshot of your output and send the git link here at canvas

SQL Hands On 5

Instruction:

1. Create a stored procedure for employee table.

- select all employee records.
- select employee by id.(With Parameter)
- insert new employee.(With Parameter)
- update employee records.(With Parameter)
- delete employee.(With Parameter)

2. Create a stored procedure for department table

- select all department records.
- select employee by department name.(With Parameter)
- insert new department.(With Parameter)
- update department records.(With Parameter)
- delete department.(With Parameter)

3. Create a stored procedure for the following requirement.

- get the list of employees by searching department name.(With Parameter)
- get the list of distinct salaries per department name.
- get the name of employee and their assigned location. (With Parameter)

Note: Upload the screenshot of output and the git link of source code.

SQL Group Activity 1

Instruction:

1. Design Relational table as per the planned project of the group.
2. Apply Normalization upon creating the tables.
3. Create an Entity Relationship Diagram.
4. Create a minimum of 4 tables that will represent your main transaction.
5. Create CRUD stored procedure in every table created.
6. Insert at least ten records per table.
7. Create a Slide/Presentation.
8. Present your work to the whole team. One member of the group will upload materials/links used on canvas. [Links to an external site.](#)

SQL Group Activity 2

Instruction:

1. With the Relational Database Created in SQL Group Activity 1 apply the following Requirements:
 - Create BackEnd using NodeJs
 - Create FrontEnd using React
2. Used CRUD stored procedure in every table created.
3. Insert at least ten records per table.
4. Create a Slide/Presentation.
5. Present your work to the whole team. One member of the group will upload materials/links used on canvas. [Links to an external site.](#)

PHP Group Exercise 1

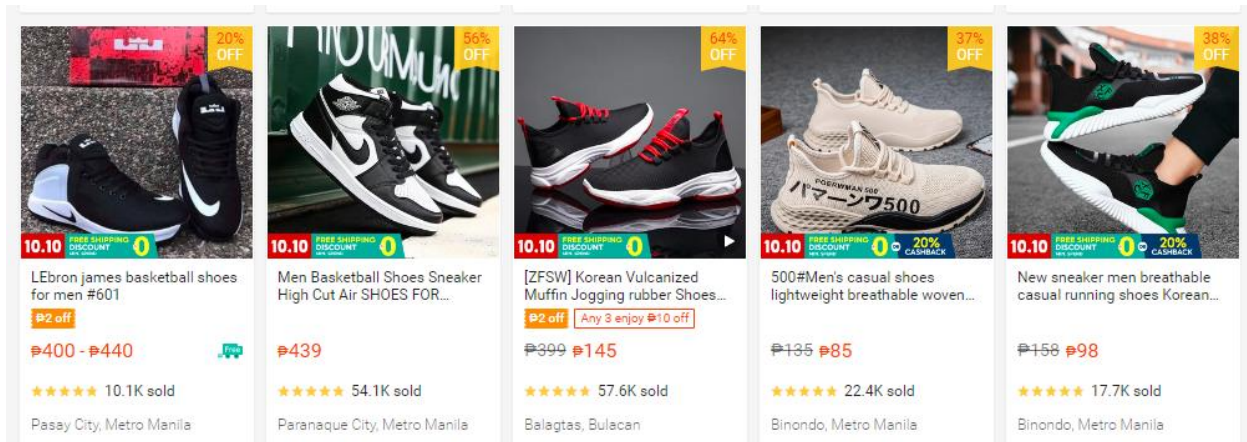
Instruction:

1. Go to this website [10,000+ Coding Practice Challenges // Edabit](#)Links to an external site. and create an account.
2. Complete at least 10 very easy, 5 easy and 3 medium challenges.
3. Make an observation report on how your group were able to solve the problems.
4. Present your work to the whole team.
5. upload the pdf file here at canvas. (Indicate your group members on observation report. Only one member will Upload the file).

PHP Hands on 1

Instruction:

1. Using Class in PHP create a static file that will create a web page as per the requirement of the image below.



Properties:

- Discount
- Item Description
- Price
- Item Image
- Number of Sold
- Sellers Location

2. You can Create your own design with css/any css framework as long as the given requirements was followed.

3. Make your design responsive.

4. Display min of ten items.

Note: Upload the screen shoot of your output and the git link for the source code.

PHP Hands on 2

Instruction:

1. Using your PHP hands on 1 Restructure and create folder management as follows:

public (will contain public sources)

->img

->css

->js

includes (will contain reusable contents)

->partials

classes(folder that holds all of the created classes)

index.php(main php file)

2, Create a new class that will add Sellers Info:

- Name:
- Location:
- Contact Number:
- Photo:

3. Create seller.php file to render sellers list make at least 10 sellers. Make this page responsive and add you own css design try to use diff css framewok.

Note: Upload screen shoot of your output and the git link for the source code.

PHP Hands on 3

Instruction:

1. Create a Form Validation as per requirement:

- productName (Required min of 5 char but less than 100 char).
- price(Required must be int).
- discount(Not Required must be int).
- numberOfSold(Required must be int)
- sellerLocation(Required min of 10 char)
- productImage(Not Required)

2. Create a validation for file upload that will only accept min of 4 image extension.

Note: Upload your screenshot and gitlink for the source code.

PHP Hands on 4

Instruction:

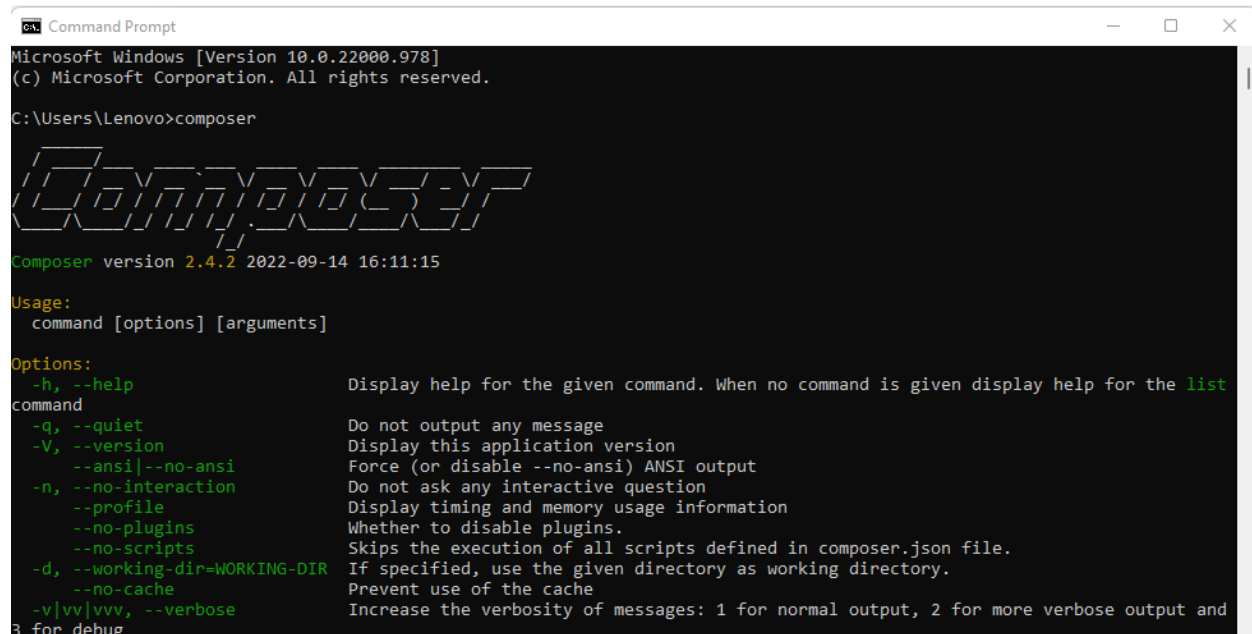
1. Used your code from the previous PHP activities.
2. Create MySql table for Products and Sellers and CRUD Stored Procedures
3. Create Class for to initialize your properties and objects
4. Create php file to Perform CRUD for Products and Sellers table
5. Create Form Validations.
6. Create Folder Management for reusable components and functionalities.
7. Create Searching Methods Using Sql commands.
8. Make your design Responsive.

Note: Upload your output and send the screenshot of your website here at canvas.

Laravel Hands On 1

Instruction:

1. Install composer [Composer \(getcomposer.org\)](https://getcomposer.org/)[Links to an external site.](#)
2. Check if composer is working. Open Command prompt and type "composer".



```
Command Prompt
Microsoft Windows [Version 10.0.22000.978]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Lenovo>composer

Composer
Composer version 2.4.2 2022-09-14 16:11:15

Usage:
  command [options] [arguments]

Options:
  -h, --help                Display help for the given command. When no command is given display help for the list
command
  -q, --quiet               Do not output any message
  -V, --version              Display this application version
  --ansi|--no-ansi          Force (or disable --no-ansi) ANSI output
  -n, --no-interaction       Do not ask any interactive question
  --profile                 Display timing and memory usage information
  --no-plugins              Whether to disable plugins.
  --no-scripts              Skips the execution of all scripts defined in composer.json file.
  -d, --working-dir=WORKING-DIR If specified, use the given directory as working directory.
  --no-cache                Prevent use of the cache
  -v|vv|vvv, --verbose      Increase the verbosity of messages: 1 for normal output, 2 for more verbose output and
3 for debug
```

3.Check PHP Version.

```
C:\Users\Lenovo>php --version
PHP 8.1.10 (cli) (built: Aug 30 2022 18:05:49) (ZTS Visual C++ 2019 x64)
Copyright (c) The PHP Group
Zend Engine v4.1.10, Copyright (c) Zend Technologies
```

Note: as of this date of activity we need PHP version 8 minimum to be compatible to Laravel 10 current version as of this date.

4.If we already have composer and Latest version of PHP installed were good to go. The next step is to check laravel documentation [Installation - Laravel - The PHP Framework For Web Artisans](#)[Links to an external site.](#).

5. Create laravel Project using composer command

```
composer create-project laravel/laravel example-app
```

Or

```
composer global require laravel/installer

laravel new example-app
```

6. Change directory to your app folder in this example "cd example-app".

7.Open it with VSCode and run the project to test if it's working.

```
cd example-app

php artisan serve
```

8. grab your code in MiniProject 1 and start implementing it to Laravel views and routes.

Laravel Hands On 2









Instruction:

1. Create a class model in your laravel project that will create object for the following requirements:

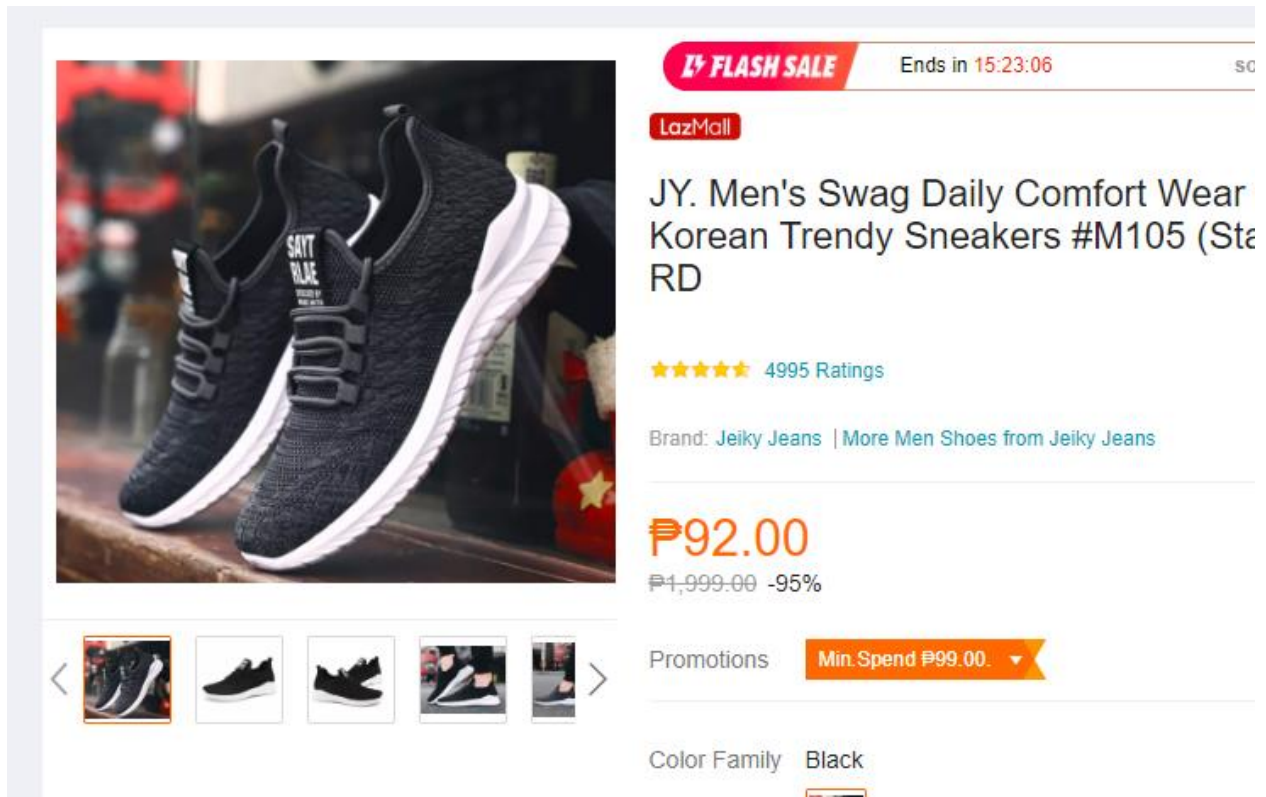
productId, productName, productDescription, productDiscount, productImage, numberOfSold, sellerLocation

2.Create a route to display min. of 10 records.

3.Make a design similar with this:

			
			
LazMall JY. Men's Leather Canvas w/ 6CM Sole Anti-Skid Shoes for Me...	LazMall JY. Men's Swaggy Classic Slip- On w/ Massage Paded Casual...	LazMall JY. Men's Real Air Cushion Running Sneakers XBO Canvas Sho...	LazMall JY. Couple's Kanye West Slides Summer 4CM Thick Soles...
P183.14 P299 -39%	P89 P599 -85%	P177.39 P299 -41%	P68.01 P199 -66%
★★★★★ (931)	★★★★★ (6585)	★★★★★ (630)	★★★★★ (2576)
Bulacan	Bulacan	Bulacan	Bulacan

4. Create a route that will display specific Item.



5. Make your web design responsive.

6. Upload Screenshot of your output here at canvas and git link for the source code.

Laravel Hands On 3

Instruction:

1. Create a new Page for List of Sellers that will show the following attributes:
 - Name:
 - Location:
 - Contact Number:
 - Photo:
2. Create Seller.php file to Models folder. Create static function to return at least 10 sellers and a function to get Seller's id.
3. Create Search method and use php functions to search or filter values of the object. Apply also search function to Products page.
4. Make the pages responsive and add you own css design try to use diff css framewok.

Note: Upload screen shoot of your output and the git link for the source code.