Assignment 1

Evaluation Objectives

1. Programming skills using SwiftUI
2. App UI design capabilities
3. Ability to select most suitable data structures for storing values
4. Problem solving abilities
5. Logical and Analytical skills
6. Modular code organization

Technical Requirements

1. The app must be created in **XCode** using **SwiftUI**
2. The app name must be **YourFirstName_Pizza** such as John_Pizza
3. Besides implementing the required functionality, submissions are required to use the correct coding conventions used in class, professional organization of the code, alignment, clarity of names is all going to be part of the evaluation.

Academic Integrity

What is allowed:

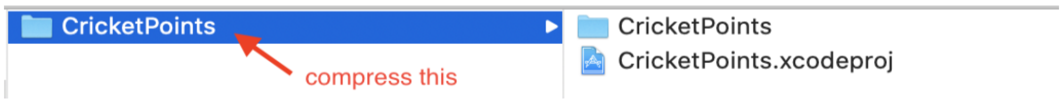
- Looking up **syntax** related to SwiftUI

What is NOT allowed:

- Searching for partial or full solutions to the main problem description
- Communication with others, either inside or outside the class
- Sharing of resources, including but not limited to links, computers, accounts, etc.

Submission Checklist

- A **screen recording** demonstrating your application.
- A zip file containing your entire project. Here is an example of what the entire project means.



- Ensure you perform Build menu -> Clean on the project before zipping it:
    - Name your zip file: **YourFirstName_Pizza.zip.** For example: **John_Pizza.zip**
    - ***.rar and *.7zip files are NOT accepted and will automatically be graded 0.**

Task:

You have been hired by a local pizza store to build a mobile application that enables customers to order pizza. Create an iOS app using ***SwiftUI*** named **YourFirstName_Pizza** (such as John_Pizza) that enables the user to enter their order details and generate a receipt indicating the total cost of their order.

In your app present pizza order form that allow the user to input following details with provided restrictions:

1. the name of the application
2. selection of vegetarian or non-veg pizza. By default, non-vegetarian should be selected.
3. selection of the size of the pizza (small, medium, large or extra-large). By default, medium size should be selected.
4. the quantity of pizzas. The minimum allowed quantity is 1, and maximum allowed quantity is 5. The default quantity is 1.
5. user's name and phone number. These are ***mandatory fields***. Order cannot be placed if these values are not provided.
6. a coupon code (e.g., OFFER284, 23OFFER, SALE123). The only valid coupons are coupons that begin with the word OFFER. By default, there is no coupon code entered.
    - If the provided coupon code starts with "OFFER" , then the customer will get 20% discount on the total *pre-tax* order price.
    - If the user attempts to place an order with an invalid coupon code, the order should not be processed, the coupon text box cleared, and an error message displayed.
7. a Navigation Bar Button labelled "DAILY SPECIAL". Tapping this button automatically populates the form field with the following values:
    - Pizza size: Large
    - Quantity: 2
    - Non vegetarian
    - Coupon code: OFFERSPECIAL
8. a Navigation Bar Button labelled "RESET". Tapping this button automatically resets the form to its default values.
9. a Button labelled "PLACE ORDER". When the user taps the button, calculate the total order price as per user's selection and the criteria given below, and display Alert showing user's selection of pizza and total price.
    - The prices given are for one pizza. Make sure you consider the quantity of pizzas while calculating the final cost. The final cost must include a 13% sales tax.

Size	Veg	Non-veg
Small	5.99	6.99
Medium	7.99	8.99

Large	10.99	12.99
Extra Large	13.99	15.00

Technical Evaluations:

1. The UI must be using Stacks and/or Grids to better organize the views on screen by creating appropriate layout.
2. You must choose appropriate input controls corresponding to the input expected. For example, when you expect user to input one of the two options, you should use Switch, when you expect user to select from a limited options you should use picker or segmented picker, etc.
3. The text fields must have placeholders that indicate the purpose of the textboxes.
4. You must use the appropriate keyboard type for each of the text field corresponding to the input type.
5. Highlight the important details by using different font color and/or weight.
6. Use suitable OOP concepts to make your code more modular, reusable, and dynamic.
7. You should consider how to represent any data in an appropriate data structure using Swift best practices.

Rubric:

Input, validation & output	50%
Use of OOP	30%
App look & feel, behavior	20%
Total	**100%**