Recipes.io App -> The App allows users to store recipes, browse them, organize them by categories and view the details of a certain recipe. The user can also add a new recipe to our collection. The App contains a basic tab bar, data model, grid layout, and AsyncImage. Besides a tab user interface that switches between the main screens of the App, the home screen also presents a list of recipes in the form of cards with images stored remotely. Each card can be redirected to a specific recipe outlining its details. The recipes can also be accessed from the categories where the recipes are filtered by type of meal. All that was for browsing purposes. The App was built from Scratch.

Functionalities/Demo:

- Allowing users to store their favorite recipes from various online resources like Pinterest.
- Allowing users to organize their saved recipes by different categories and view details.
- Allowing users to create new recipes and add all details and descriptions.

Developing Languages, Tools, and Techniques Needed:

SwiftUI https://developer.apple.com/xcode/swiftui/

Xcode 14.0.1 iPhone 13 Simulator iOS 15 compatibility Any version < iOS 16
https://developer.apple.com/documentation/xcode-release-notes/xcode-14_0_1-release-notes

Scratch https://en.wikipedia.org/wiki/Scratch_(programming_language)

SF Symbols https://developer.apple.com/sf-symbols/

Swift https://developer.apple.com/swift/

Forks over Knives Recipes https://www.forksoverknives.com

PostImage(JPG/PNG Direct Link Generator) https://postimages.org

Prerequisites & Setups:

Create a new Xcode project.

Rearrange all files into different folders properly.

Set up Automatic Preview for SwiftUI files: Adjust Editor Options -> Enable Canvas on-side.

Synchronous Developing Notes:

Start by create several View files under `Main` and define their navigation view titles:

In `HomeView.swift:`

```
struct HomeView: View {
    var body: some View {
        NavigationView {
            Text("My Recipes")
                .navigationTitle("My Recipes")
        } } }
```

In `CategoriesView.swift:`

```
 struct CategoriesView: View {
    var body: some View {
        NavigationView {
            Text("Categories")
                .navigationTitle("Categories")
```

In `NewRecipeView.swift:`

```
 struct NewRecipeView: View {
    var body: some View {
        NavigationView {
            Text("New Recipe")
                .navigationTitle("New Recipe")
```
In `FavoritesView.swift`, initialize no favorites currently saved for the user:
```
 struct FavoritesView: View {
    var body: some View {
        NavigationView {
          Text("You haven't saved any recipe to your favorites yet.")
                .padding()
                .navigationTitle("Favorites")
```
Initialize version in `SettingsView.swift:`
```
 struct SettingsView: View {
    var body: some View {
        NavigationView {
            Text("v1.0.0")
                .navigationTitle("Settings")
```
<u>Tab Bar Implementation:</u>

Create a new SwiftUIView file named `TabBar.swift:`

Add tab views and tab items for each sections with icons from SF Symbol such as:
```
  TabView{
            HomeView()
                .tabItem {
                    Label("Home", systemImage: "house") }
...
```
all tab bars displayed.PNG

Add Tab Bars into `ContentView.swift:`

`TabBar()`

And in Simulator, we can go to different screen sections by clicking on different tab bars icons.
Add Stack navigation view style to all views:

`.navigationViewStyle(.stack)`

<u>Data Model:</u>

Create `RecipeModel.swift` under `ViewModel:`

Conform to the identifiable protocol since each recipe is unique with all its attributes:
```
  struct Recipe: Identifiable {
    let id = UUID()
    let name: String
    let image: String
    let description: String
    let ingredients: String
    let directions: String
    let category: Category
    let datePublished: String
    let url: String}
```
Create a enum to enumerate all the preset meal courses:
```
 enum Category:  String {
    case breakfast = "Breakfast"
    case soup = "Soup"
    case salad = "Salad"
    case appetizer = "Appetizer"
    case main = "Main"
    case side = "Side"
    case dessert = "Dessert"
    case snack = "Snack"
    case drink = "Drink"}
```
Prepare some recipes in the form of static data by adding some extensions to the recipe.
Test in `HomeView.swift` to see if recipe names display:
```
    NavigationView {
            List(Recipe.all){ recipe in
                Text(recipe.name)
                    .navigationTitle("My Recipes")
```
all recipe names displayed in HomeView.PNG

Recipe Card-AsyncImage:

Create a SwiftUI file `RecipeCard.swift` in `Components.`

Use VStack to define the width and height properties of recipe photo and background:
```
    VStack {
            AsyncImage(url: URL(string: recipe.image)) { image in
                image
            } placeholder: {
                Image(systemName: "photo")
                    .resizable()
                    .scaledToFit()
                    .frame(width: 40, height: 40,alignment: .center)
                    .foregroundColor(.white.opacity(0.7))
                    .frame(maxWidth: .infinity, maxHeight: .infinity)
```
creamy carrot soup image shows up.PNG

To make the recipe image fit better with aspect ratio:
```
   .resizable()
   .aspectRatio(contentMode: .fill)
```
Make the recipe name overlay properly display:
```
    Text(recipe.name)
        .font(.headline)
        .foregroundColor(.white)
        .frame(maxWidth: 136)
        .padding()
```
recipe name displayed.PNG

Recipe List:

Create a new file `RecipeList.swift` in `ViewModels.`

Use string literals to represent the recipe lists:

```
VStack {
HStack {
  Text("\(recipes.count) \(recipes.count > 1 ? "recipes" : "recipe")")
             .font(.headline)
             .fontWeight(.medium)
             .opacity(0.7)
                  Spacer()
```

Use LazyGrid to make all recipe images display and layout properly:

```
LazyVGrid(columns:
[GridItem(.adaptive(minimum: 160), spacing: 15)], spacing: 15) {
           ForEach(recipes) { recipe in
           RecipeCard(recipe: recipe)
```

all recipe images displayed.PNG

Also pass recipe list to HomeView:

```
RecipeList(recipes: Recipe.all)
```

HomeView screen final look.PNG

Recipe View:

Create a new `RecipeView.swift` file in `Details.`

Error: The vertical images were overflowing into the recipe title and description. DEBUGGING: Change `.frame(height: 300)` to `.scaledToFill() ..frame(height: 300, alignment: .center) ..clipped()` to manually crop the image into aligned height to fit the screen.

Import recipe ingredients and directions:

```
      VStack(alignment: .leading, spacing: 20){
                        Text("Ingredients")
                             .font(.headline)
                        Text(recipe.ingredients)}
                   VStack(alignment: .leading, spacing: 20){
                        Text("Directions")
                             .font(.headline)
                        Text(recipe.directions)
```

Adjust the framework of text alignment:

```
       .frame(maxWidth: .infinity, alignment: .leading)
       .padding(.horizontal)
       .ignoresSafeArea(.container, edges: .top)
```

Navigate to ViewList in `Recipe.List.swift:`

```
  NavigationLink(destination: RecipeView(recipe: recipe)){
            RecipeCard(recipe: recipe)
```

Now run the simulator, click on image on HomeView will be redirected to Recipe Page:

RecipeView from HomeView redirect works-1.PNG

RecipeView from HomeView redirect works-2.PNG

Recipe Categories:

Identify category list in `CategoriesView.swift:`

```
 List {
    ForEach(Category.allCases){ category in
    Text(category.rawValue + "s")
```

Now all categories displayed as a list:

categories displayed as a list.PNG

And if click certain category, will be navigated to its related recipes:

categories navigation works-1.PNG

categories navigation works-2.PNG

Add Recipe Form:

Create `AddRecipeView,swift` to add recipe forms:

Set Categories a selectable view:

```
   Section(header: Text("Category")){
         Picker("Category", selection: $selectedCategory) {
                 ForEach(Category.allCases) { category in
                 Text(category.rawValue)
```

selected category works.PNG

Text Editor:

Use `TextEditor(text: $...))` to bind text editing sections.

Toolbar Item:

Obtain a x mark for cancellation and a check mark for done:

```
   ToolbarItem(placement: .navigationBarLeading) {
                    Button {
                        } label:{
                        Label("Cancel", systemImage: "xmark")
                            .labelStyle(.iconOnly)
     ToolbarItem(placement: .navigationBarLeading) {
                    Button {
                        } label:{
                        Label("Done", systemImage: "checkmark")
                            .labelStyle(.iconOnly)
```

xmark and checkmark.PNG

Present Form:

In `NewRecipeView,swift,` manually add the form and make it present:

```
  @State private var showAddRecipe = false
...
            Button("Add recipe manually"){
                showAddRecipe = true
        ...
        .sheet(isPresented: $showAddRecipe){
            AddRecipeView()
```

manually add new recipe.PNG

MVVM Design Pattern:

Create a `RecipesViewModel.swift`:

```
 class RecipesViewModel: ObservableObject {
    @Published private(set) var recipes: [Recipe] = []
    init() {
        recipes = Recipe.all }
```

Configure the state object and its environment in `Recipes.io AppApp.swift`:

```
  @StateObject var recipesViewModel = RecipesViewModel()
```

Save Recipe:

Create a new add recipe function in `RecipesViewModel.swift`:

```
 func addRecipe(recipe: Recipe){
        recipes.append(recipe } }
```

Also add an extension to configure the newly added recipes in `AddRecipeView.swift`:

```
extension AddRecipeView{
    private func saveRecipe(){
        let now = Date()
        let dateFormatter = DateFormatter()
        dateFormatter.dateFormat = "yyyy-mm-dd"
        let datePublished = dateFormatter.string(from: now)
        print(datePublished)
        let recipe = Recipe(name: name, image: "", description:
description, ingredients: ingredients, directions: directions,
category: selectedCategory.rawValue , datePublished: datePublished,
url: "")
        recipesVM.addRecipe(recipe: recipe)
    } }
```

Test, and here, new input recipe is added:

add a new recipe.PNG

new recipe is added-1.PNG

new recipe is added-2.PNG