



# Kitchen Assistant



Vishnu Prasad Vishwanathan- 793782749

## **Intent:**

To develop a framework for an interactive Restaurant Database. An app for the effective maintenance of the restaurant kitchen.

It should be a simple interactive app as the target consumers are Restaurant managers, chefs and workers.

Improve the overall quality of the kitchen by reducing the human errors.

Look at almost everything stored, to be bought, to be discarded, the punching time of the workers,

automated alert on the existing items if they are going to reach the expiration date etc.

## **Idea:**

Maintaining a restaurant is a challenging task as it deals with important aspects like cleanliness, food storage, restaurant maintenance etc.

The food storage is the primary concern in any kitchen.

It requires considerable efforts to make a kitchen look good.

An app that would notify the users about almost everything thereby helping them to buy or discard or avoid placing things wrongly could be of great help to them.

## **Functionality:**

Every restaurant has an account.

The app should be able to get the inputs from the user.

The inputs include the structure of the storage containers.

The top level inputs are the containers and types. For example: number of freezers, number of coolers etc.

The next level of inputs include the items in each of these containers, their expiry dates, the cleaning date, items checkout, required items etc.

The app should be able to store the data obtained from the user.

The app should be able to notify the user prior to the date of expiration along with the product name, the cleaning date deadline and items to be bought.

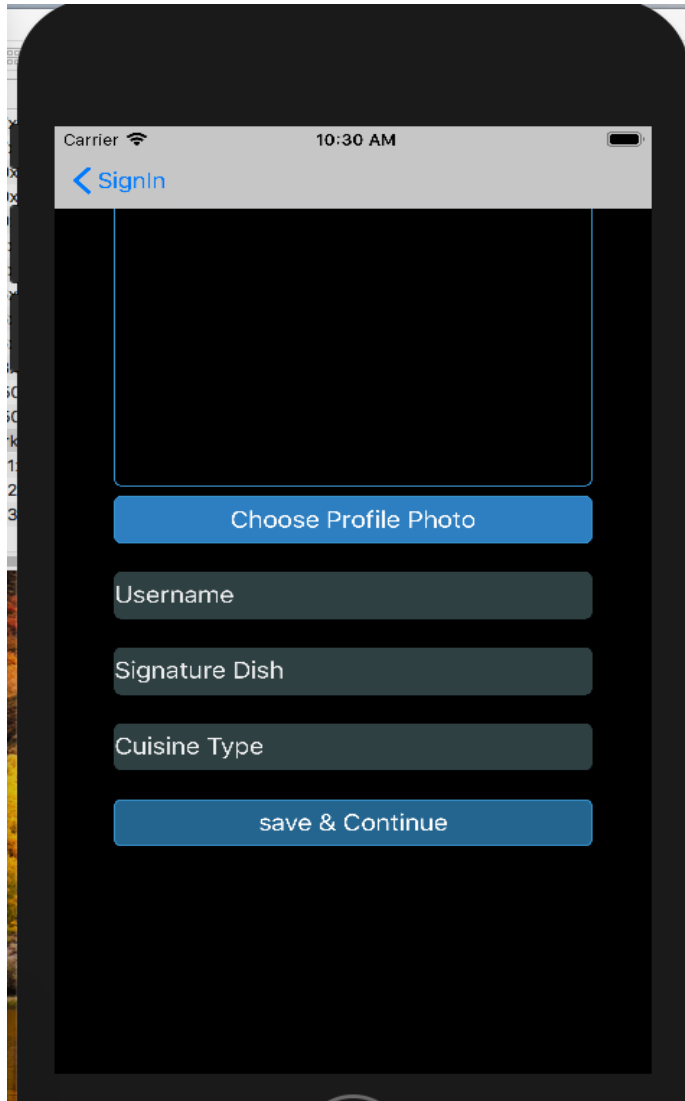
The app needs to be updated everyday once(at least) to keep track of the items.

**The App doesn't have a storyboard.**

## Login Page:

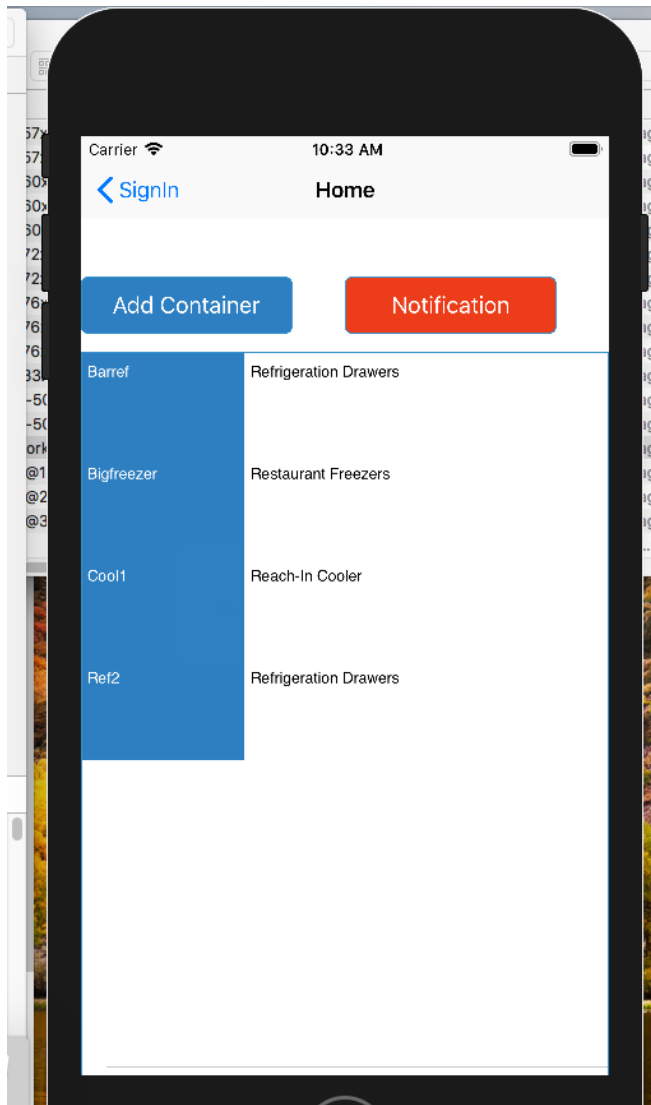


Home page takes us to the login, Sign Up, if there is no network then it takes to our dice game.



The Signup takes us to the profile page which helps us to give the information and profile picture for the restaurant.

## Home Page :



The Home page shows the Containers that are added to the restaurant page and we can add more by using the Add Container Button.

Notification shows the expired food items in the containers.

## Code Snippet for finding the expiration:

```
let dateFormatter = DateFormatter()
dateFormatter.dateStyle = .short
dateFormatter.timeStyle = .short
dateFormatter.dateFormat = "yyyy-MM-dd hh:mm"
let currdatestr = dateFormatter.string(from: Date())
let convdate = dateFormatter.date(from: self.oneitem.isexpdt!)
let checkdate = dateFormatter.string(from: convdate!)

if checkdate.compare(currdatestr) == .orderedAscending
{
    print("expired", checkdate, currdatestr)

self.itemarr.append(self.oneitem)
print("name , count :",self.itemarr[0].iname!,self.itemarr.count)

self.tableView1.reloadData()
}
```

## Code Snippet for finding the Date extraction:

```
func createdategetter()
{
    let toolbar = UIToolbar()
    toolbar.sizeToFit()
    datepicker.datePickerMode = .dateAndTime

    let finishButton = UIBarButtonItem(barButtonItemSystemItem: .done, target: nil, action:
        #selector(finishButtonPress))
    toolbar.setItems([finishButton], animated: true)

    self.itemexpirydttxt.inputAccessoryView = toolbar
    self.itemexpirydttxt.inputView = datepicker
}
@objc func finishButtonPress()
{
    let dateFormatter = DateFormatter()
    dateFormatter.dateStyle = .short
    dateFormatter.timeStyle = .short
    dateFormatter.dateFormat = "yyyy-MM-dd hh:mm"
    let datefinalstr = dateFormatter.string(from: datepicker.date)
    itemexpirydttxt.text = "\(datefinalstr)"
    self.view.endEditing(true)
}
```

## Code Snippet for Image Picker:

```
extension RestaurantProfileViewController: UIImagePickerControllerDelegate, UINavigationControllerDelegate{
    func imagePickerController(_ picker: UIImagePickerController, didFinishPickingMediaWithInfo info: [String : Any]){
        if let imagefile = info[UIImagePickerControllerOriginalImage] as? UIImage{
            profileimage.image = imagefile
            putimg = imagefile
        }
        print("activated picker")
        print(info)
        self.dismiss(animated: true, completion: nil)
        let userID = Auth.auth().currentUser?.uid
        let firestore = Storage.storage().reference().child("ProfilePics").child(userID!)
        if let profilepic = putimg , let picdata = UIImageJPEGRepresentation(profilepic, 0.2){
            firestore.putData(picdata, metadata: nil, completion: { (metadata, error) in
                if error != nil{
                    return
                }
            })
        }
    }
}
```

`self.imagestr = (metadata?.downloadURL()?.absoluteString)!` ⚠️ 'downloadURL()' is deprecated: Use 'Sto.

## Code Snippet for Image Picker:

```
    }
    if(self.oneitem.isexpdt != nil){
        print("date check")
        let dateFormatter = DateFormatter()
        dateFormatter.dateFormat = "yyyy-MM-dd hh:mm"
        let currdatestr = dateFormatter.string(from: Date())
        let convdate = dateFormatter.date(from: self.oneitem.isexpdt!)
        let checkdate = dateFormatter.string(from: convdate!)

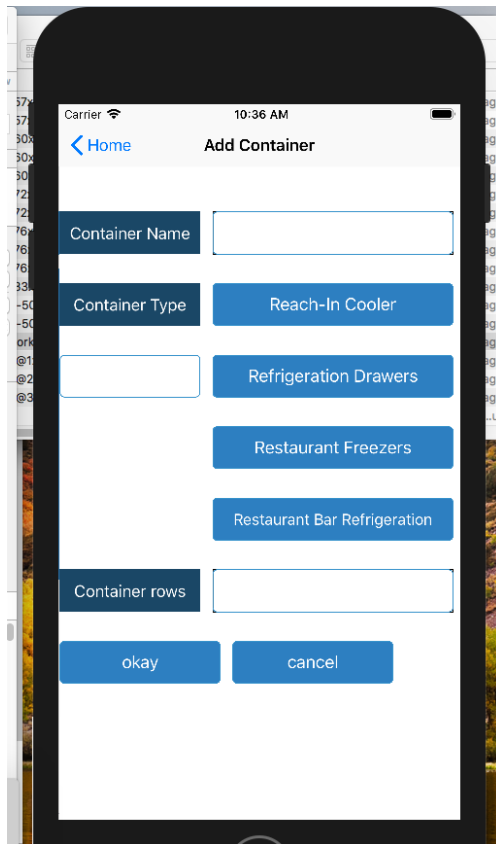
        if checkdate.compare(currdatestr) == .orderedAscending
        {
            print("expired", checkdate, currdatestr)
        }

        self.itemarr.append(self.oneitem)
        print("name , count :",self.itemarr[0].iname!,self.itemarr.count)

        self.tableView1.reloadData()
    }
}
```



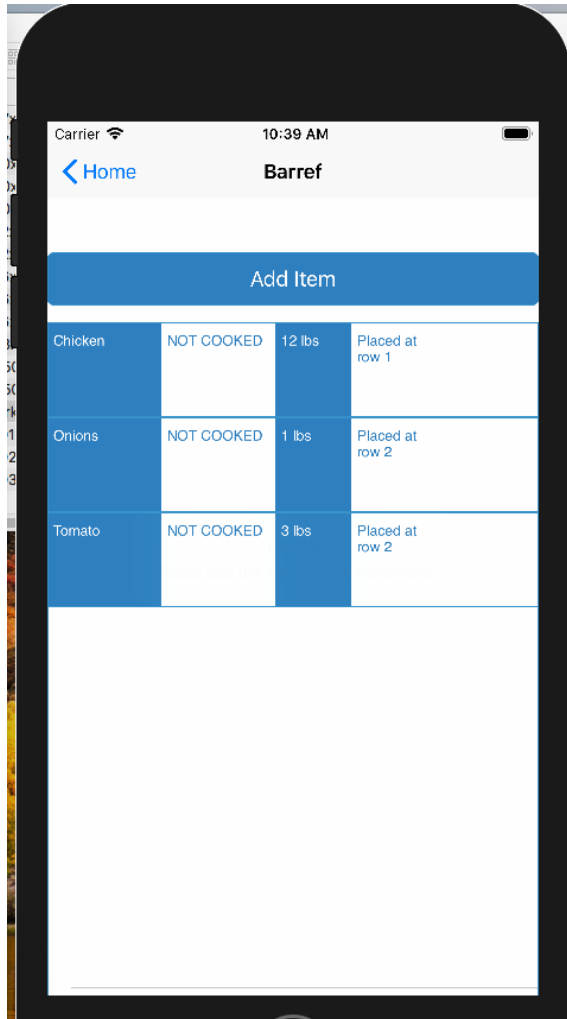
## Add Container:



The screenshot shows a mobile application interface for adding a container. The status bar at the top indicates 'Carrier', signal strength, and the time '10:36 AM'. The app's navigation bar includes a blue back arrow and the text '< Home', followed by the title 'Add Container'. The form consists of several input fields and buttons: a 'Container Name' label followed by a text input field; a 'Container Type' label followed by a list of four blue buttons: 'Reach-In Cooler', 'Refrigeration Drawers', 'Restaurant Freezers', and 'Restaurant Bar Refrigeration'; and a 'Container rows' label followed by a text input field. At the bottom of the form are two blue buttons labeled 'okay' and 'cancel'.

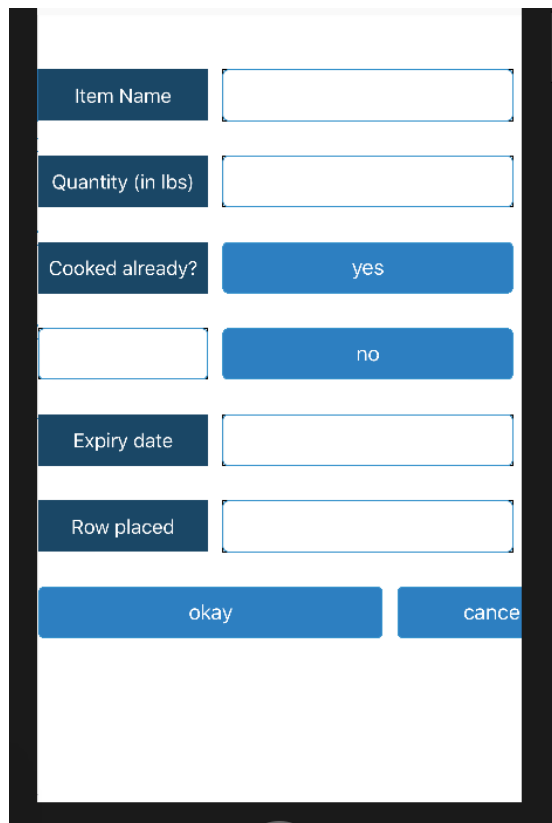
Container information.

## Container MainPage:



We can see the items added to the particular container here.

## Add Items to the Container:

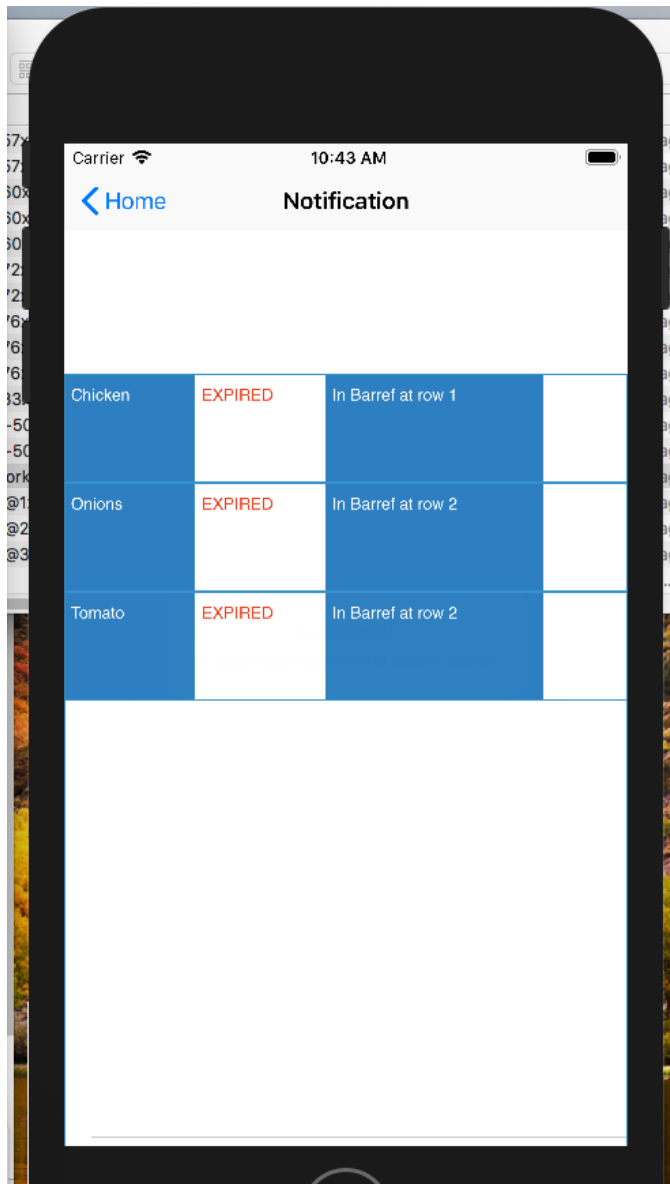


Item Name	<input type="text"/>
Quantity (in lbs)	<input type="text"/>
Cooked already?	<input type="button" value="yes"/>
	<input type="button" value="no"/>
Expiry date	<input type="text"/>
Row placed	<input type="text"/>
<input type="button" value="okay"/>	<input type="button" value="cancel"/>

The item information should be provided in this page.

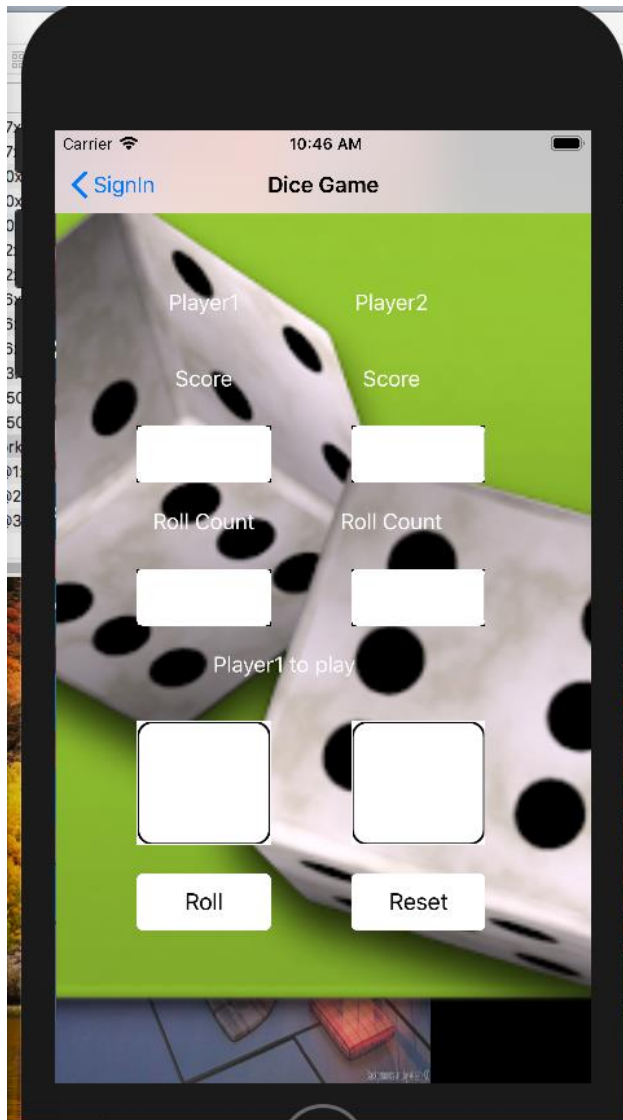
Which we use for storing them.

## Notification Page:



The Expired food items are shown here and if we keep cant keep the cooked and uncooked food on the same cooler at the same row.

## DiceGame Page:



The game gives some offline experience to the user by letting them roll the dice.

## **Conclusion:**

The primary ambitious feature is to make the app accessible for the workers to share their working hours, time in, time out records, work update, manager adding daily activities etc.

The above mentioned feature makes the manager the server and the workers as the client and the DB stands as the server for both of them. In that case the workers would have only parts of the app visible to them.

The app can help us get the best places to get the items and the tracker for the items to be bought every month.

We can add the dining features to the app where the app can manage the orders.