

Criterion E: Product development

Complex techniques used to address the client's requirements:

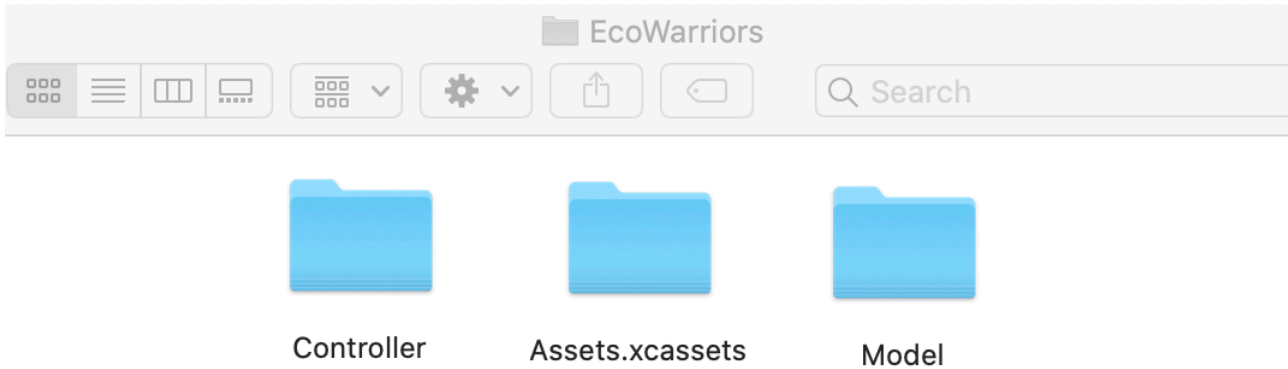
1. Use of Swift to customise pages and improve functionality
2. Graphical user interface (GUI)
3. Functions and Sub-routines
4. If-then
5. Arrays and Loops and Exit Conditions
6. Manipulation of graphics involving multiple techniques using Adobe Photoshop¹
7. Integration of components using advanced features from Procreate for digital art to create logo ²

¹ "Photo, image & design editing software" <https://www.adobe.com/products/photoshop.html>. Accessed 21 Mar. 2021.

² "Procreate." <https://procreate.art/>. Accessed 21 Mar. 2021.

Organisation of the product

Application folder structure:

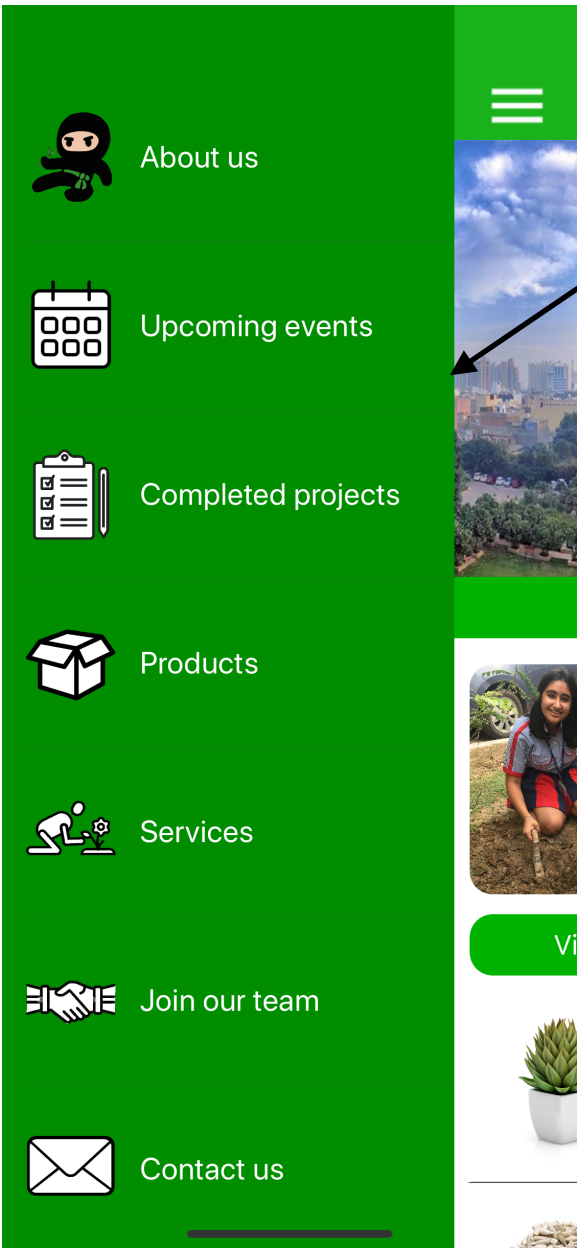


Content is placed in appropriate folders with logical folder names:

- **Assets.xcassets:** Holds images, icons for buttons etc
- **Controller:** Contains code related to UIElements (eg. UIButton's reference+ clicked action) for all pages.
- **Model:** Contains protocols/menu layout code.

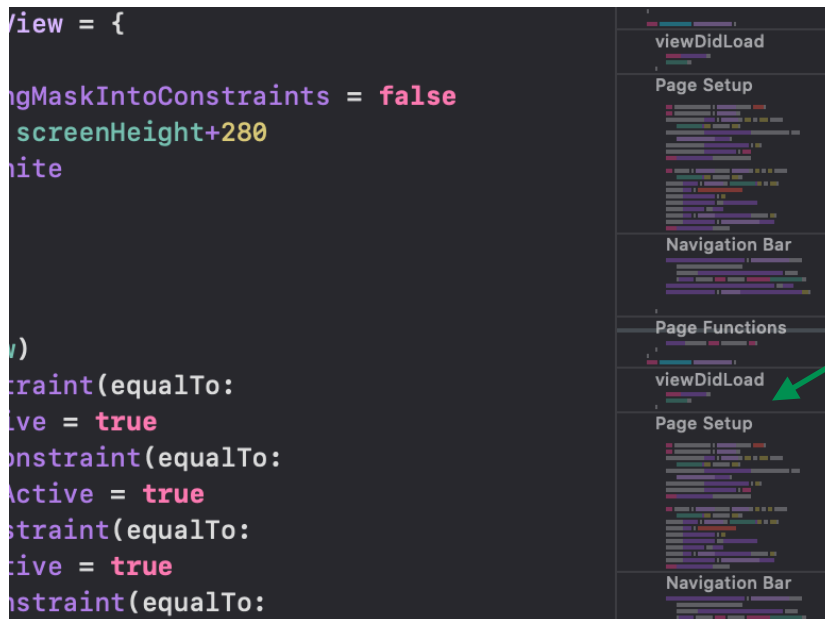
Such folders make it easy to locate a particular code or picture.

Application structure:



A menu bar breaks up content in smaller sections making specific content easier to find.

Application code structure



Marks have been created in between the code. They are displayed on the side bar of the coding area on Xcode. It organises and helps one jump to a particular part of the code

Class Page itself

viewDidLoad() Function called when view (UI) loads on screen

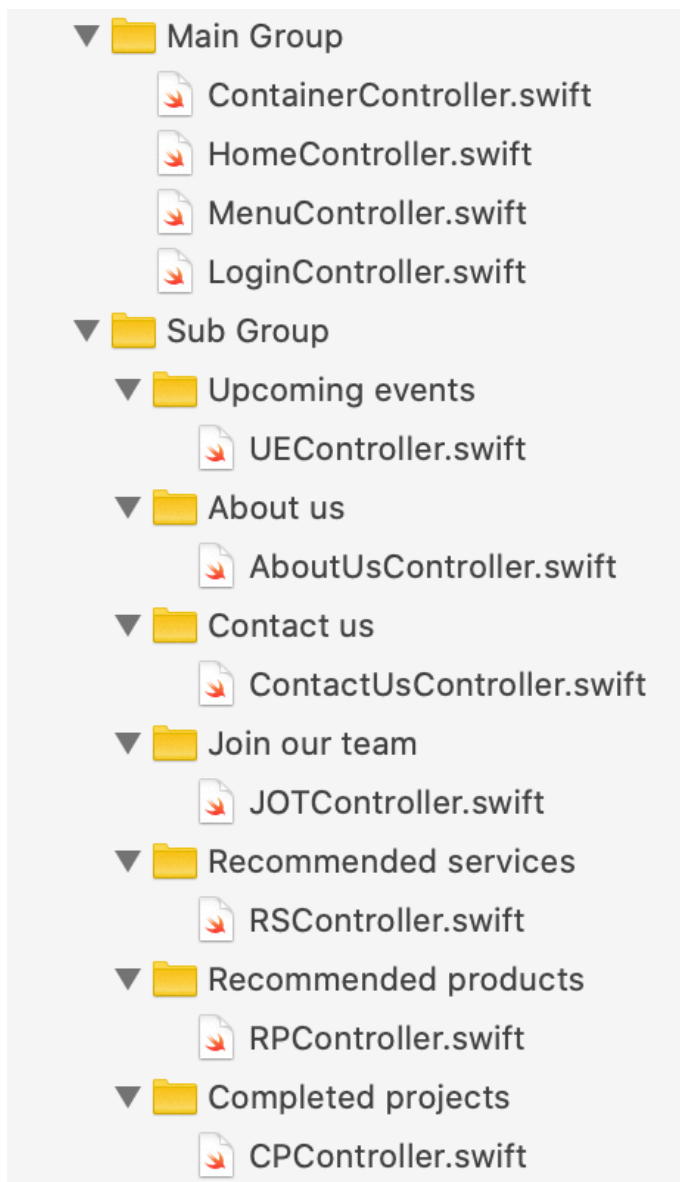
Page Setup Buttons, fonts, text colour, labels etcetera

Navigation Bar Sets up top bar of the page.

Page Functions Allows buttons to perform actions like linking pages, opening pages on safari etcetera

Scroll View Amount of scroll area given to the user.

Application code file structure:



The code is distributed in two main folders:

- **Main Group:** contains the code for the home page, menu bar, container controller and code for login.
- **Sub Group:** Contains the code for all screens in the menu bar.

This keeps code organised and is easy to locate.

Techniques

1. Use of Swift to customise pages and improve functionality

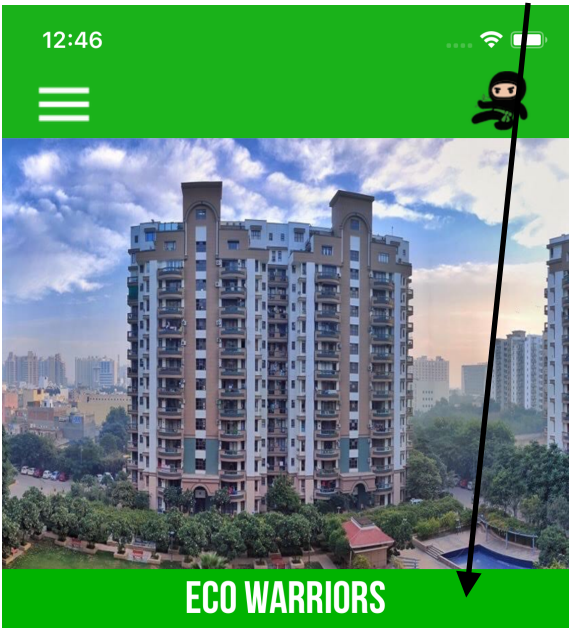
Customised colours can be created using swift.

```
extension UIColor {  
    static func rgb(red: CGFloat, green: CGFloat, blue: CGFloat) ->  
        UIColor {  
        return UIColor(red: red/255, green: green/255, blue: blue/255,  
            alpha: 1)  
    }  
  
    static func mainBlue() -> UIColor {  
        return UIColor.rgb(red: 0, green: 179, blue: 0)  
    }  
  
    static func googleRed() -> UIColor {  
        return UIColor.rgb(red: 0, green: 141, blue: 0)  
    }  
  
    static func lightGreen() -> UIColor {  
        return UIColor.rgb(red: 74, green: 196, blue: 47)  
    }  
  
    static func UltraLightGreen() -> UIColor {  
        return UIColor.rgb(red: 160, green: 255, blue: 160)  
    }  
  
    static func lightGrey() -> UIColor {  
        return UIColor.rgb(red: 230, green: 230, blue: 230)  
    }  
}
```

These colours are applied to different sections of the app for example:

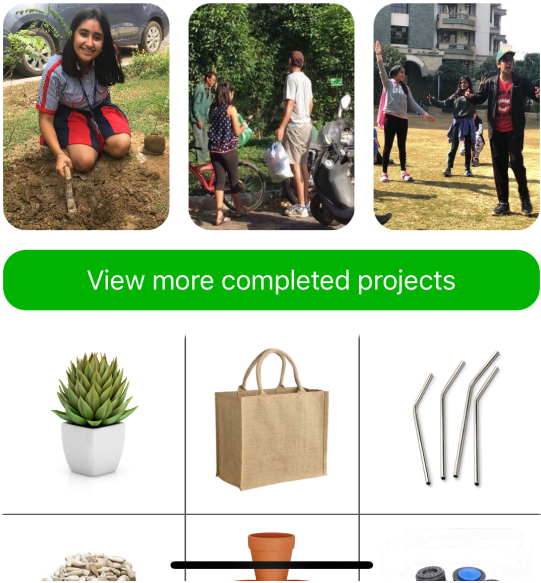
```
let label = UILabel(frame: CGRect(x: 0, y: 0, width:  
    screenWidth, height: 40))  
label.center = CGPoint(x: screenWidth/2, y: 305)  
label.textAlignment = NSTextAlignment.center  
label.text = "ECO WARRIORS"  
label.backgroundColor = .mainBlue()  
label.textColor = .white  
label.font = UIFont(name: "Bebas Neue", size: 30)  
self.scrollView.addSubview(label)
```

As seen in the above screenshot, swift also allows one to customise the overall look and feel of the app, example the below label



Font style, size, color, positioning size etc. of the label was customised.

Swift was used to customise the whole app in a similar manner.

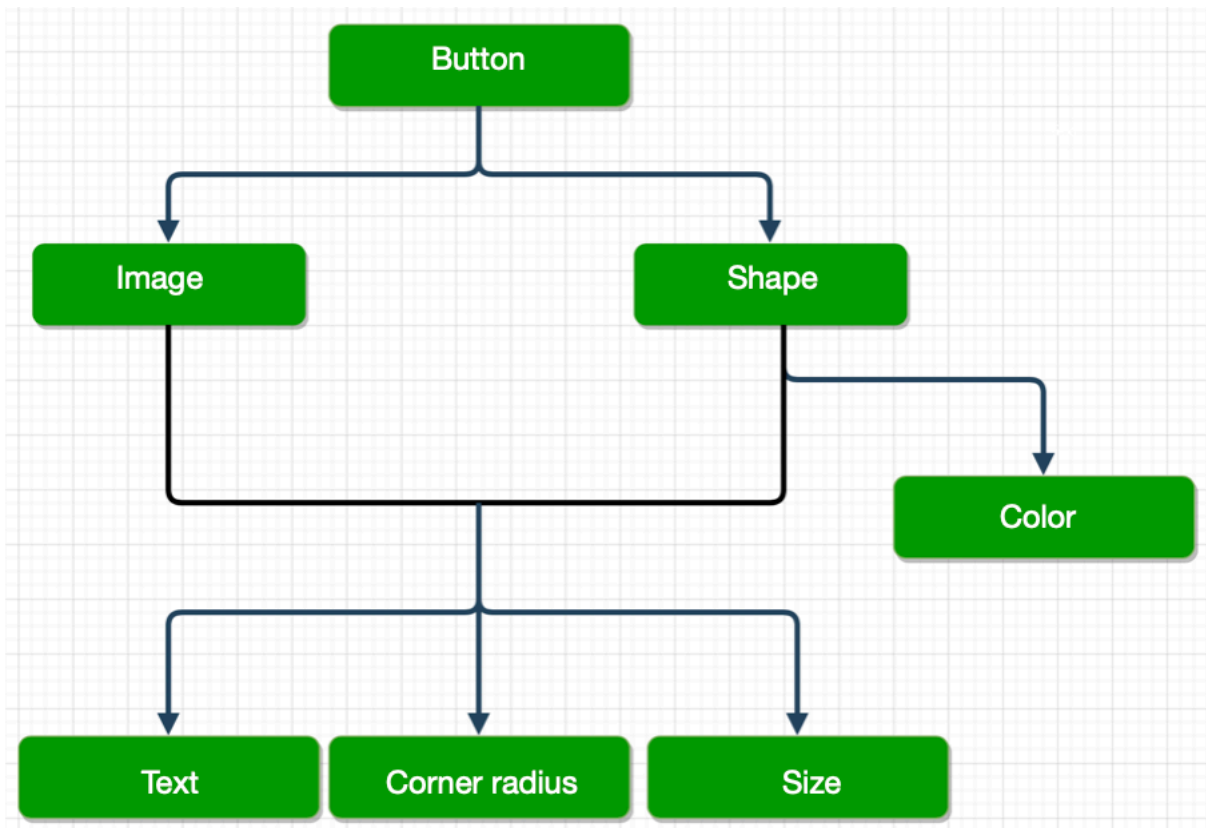


Swift was further used to create buttons and functions to improve functionality. (next 2 techniques)

2. Graphical user interface (GUI)

Buttons display objects that convey information, and represent actions that can be taken by the user. It is an essential element of an interactive IT solution like an app.

Buttons were represented in the following forms:



A button was either a shape or an image:



Types of buttons
in the app

View more products

Code for “view more products” button:

```
let moreButton = UIButton()
moreButton.frame = CGRect(x: 0, y: 0, width: screenWidth-20,
    height: 40)
moreButton.center = CGPoint(x: screenWidth/2, y: 525)
moreButton.backgroundColor = .mainBlue()
moreButton.layer.cornerRadius = 15
moreButton.setTitle("View more completed projects", for:
    .normal)
```

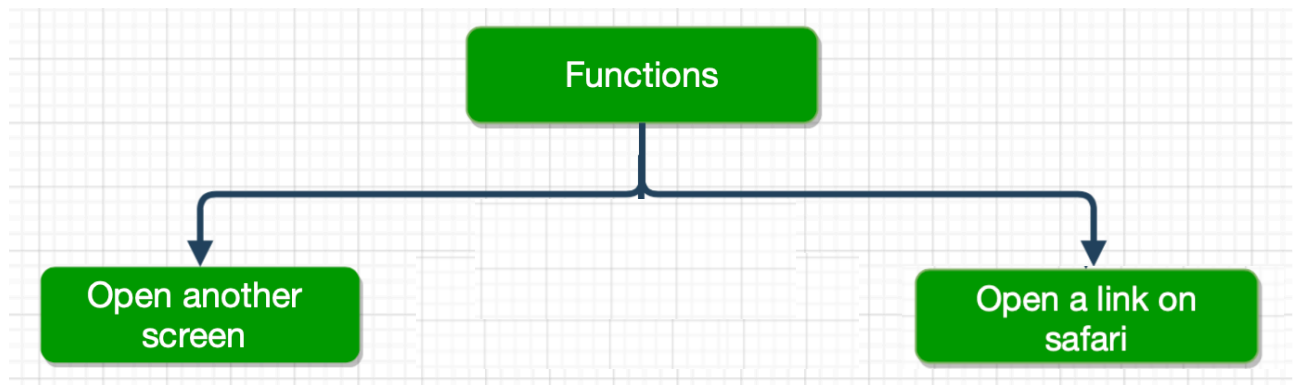
- The size (“frame”), the colour (“backgroundColor”) and the corner radius (“layer.cornerRadius”) was specified to make the button look attractive.
- The text (setTitle) was put to tell the user what function the button is going to perform.

Similar technique was used for the above picture button:

```
let achievementImg1 = UIImage(named: "A1")
let achievementBut1 = UIButton()
achievementBut1.frame = CGRect(x: 10, y: 342, width: (screenWidth/3)-15, height: 150)
achievementBut1.setBackgroundImage(achievementImg1, for: UIControl.State.normal)
achievementBut1.addTarget(self, action:#selector(handlePress1), for: .touchUpInside)
achievementBut1.layer.cornerRadius = 15;
achievementBut1.clipsToBounds = true
self.scrollView.addSubview(achievementBut1)
```

3. Functions and Sub-routines

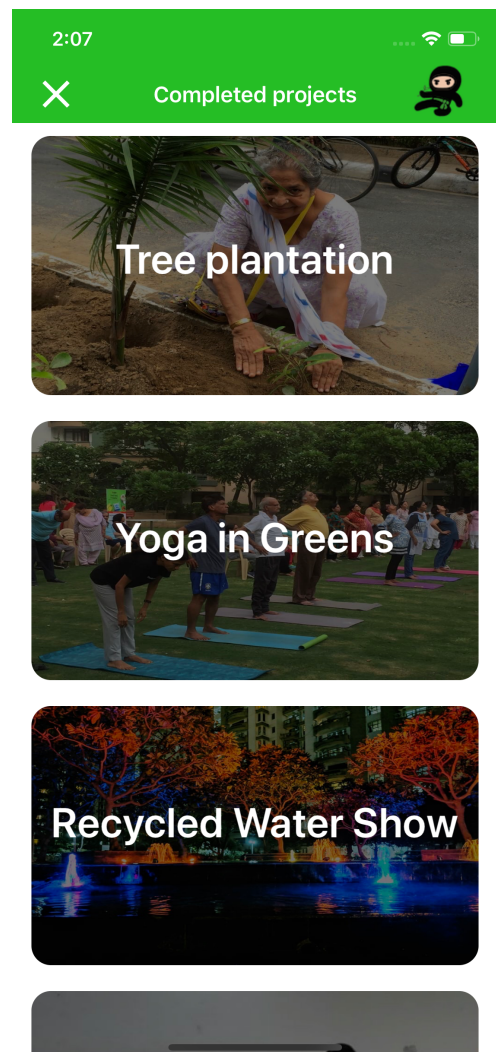
Buttons had two types of functions using sub routines:



Example:

View more completed projects

opens another screen
("completed projects") from the
menu bar.



This is done by creating a handle press (“handlePress4”) which is given the command to open “CPController”(Completed projects screen).

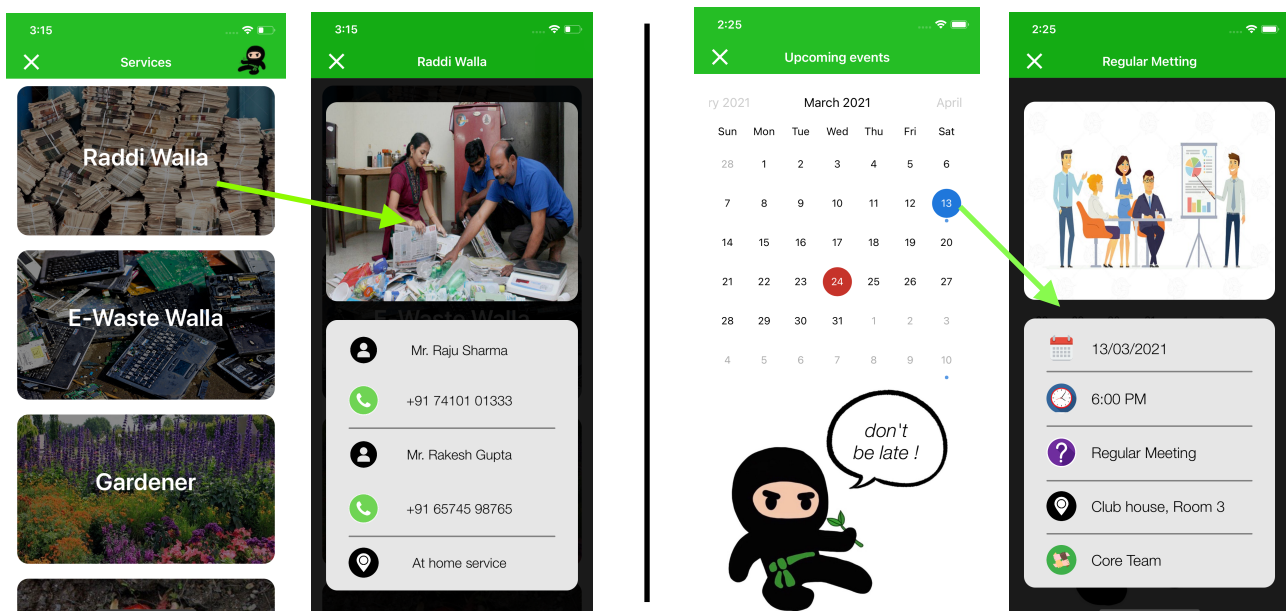
```
@objc func handlePress4() {  
    let controller = UINavigationController(rootViewController:  
        CPController())  
    controller.modalPresentationStyle =  
        UIModalPresentationStyle.overCurrentContext  
    present(controller, animated: true, completion: nil)  
}
```

“handlePress4” is put into to the code written for the “view more completed projects” button.

```
let moreButton = UIButton()  
moreButton.frame = CGRect(x: 0, y: 0, width: screenWidth-20,  
    height: 40)  
moreButton.center = CGPoint(x: screenWidth/2, y: 525)  
moreButton.backgroundColor = .mainBlue()  
moreButton.layer.cornerRadius = 15  
moreButton.setTitle("View more completed projects", for:  
    .normal)  
moreButton.addTarget(self, action: #selector(handlePress4),  
    for: .touchUpInside)  
moreButton.clipsToBounds = true  
self.scrollView.addSubview(moreButton)
```

These types of buttons are used in most of the screens like “upcoming events”, “Services”

etc.

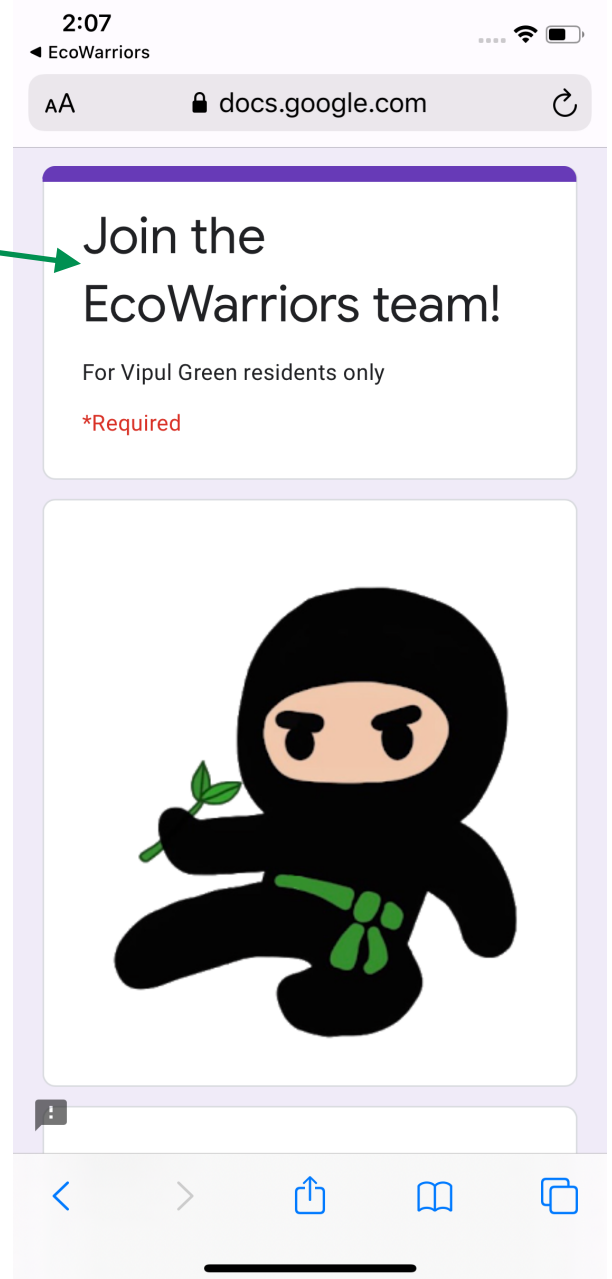


Join the EcoWarriors now!

opens a link of a google form in
Safari

This is done by creating an object function
“joinNow” and the exact url is inserted.

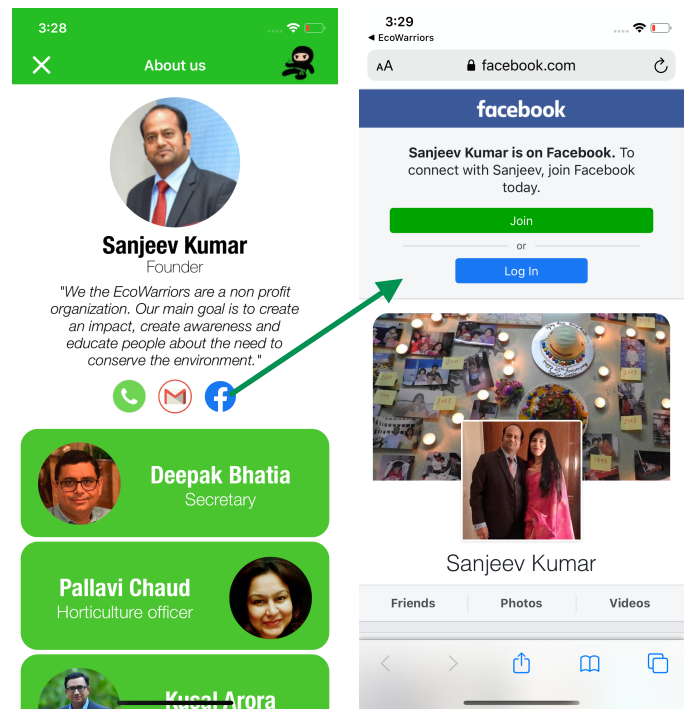
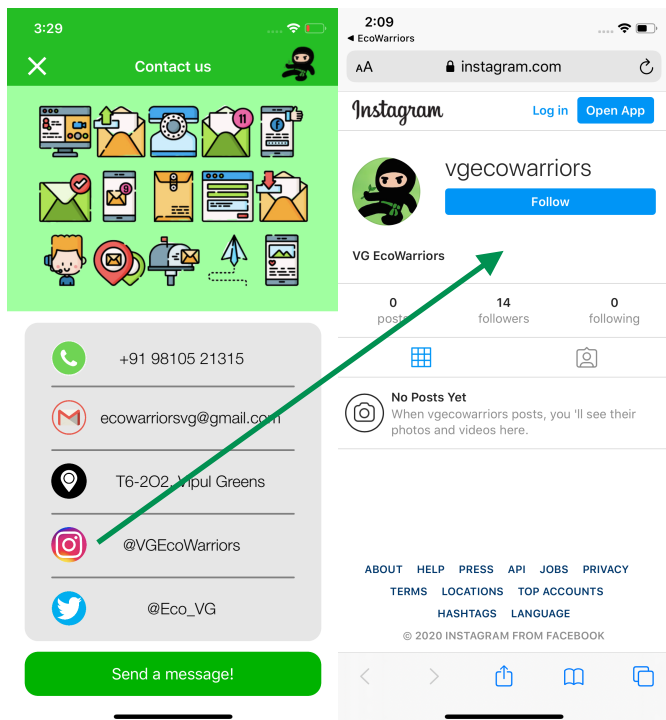
```
@objc func joinNow() {  
    if let url = URL(string:  
        "https://forms.gle/SH8iPC6P7AdC5s898") {  
        UIApplication.shared.open(url)  
    }  
}
```



“joinNow” is added as a action to the code of the “join the EcoWarrior now!” button.

```
let moreButton = UIButton()  
    moreButton.frame = CGRect(x: 0, y: 0, width:  
        screenWidth-20, height: 40)  
    moreButton.center = CGPoint(x: screenWidth/2, y: 640)  
    moreButton.backgroundColor = .mainBlue()  
    moreButton.layer.cornerRadius = 15  
    moreButton.setTitle("Join the EcoWarriors now!", for:  
        .normal)  
    moreButton.clipsToBounds = true  
    moreButton.addTarget(self, action: #selector(joinNow),  
        for: .touchUpInside)  
    self.scrollView.addSubview(moreButton)
```

These types of buttons are used in some of the screens of my app like “about us”, “contact us” etc.

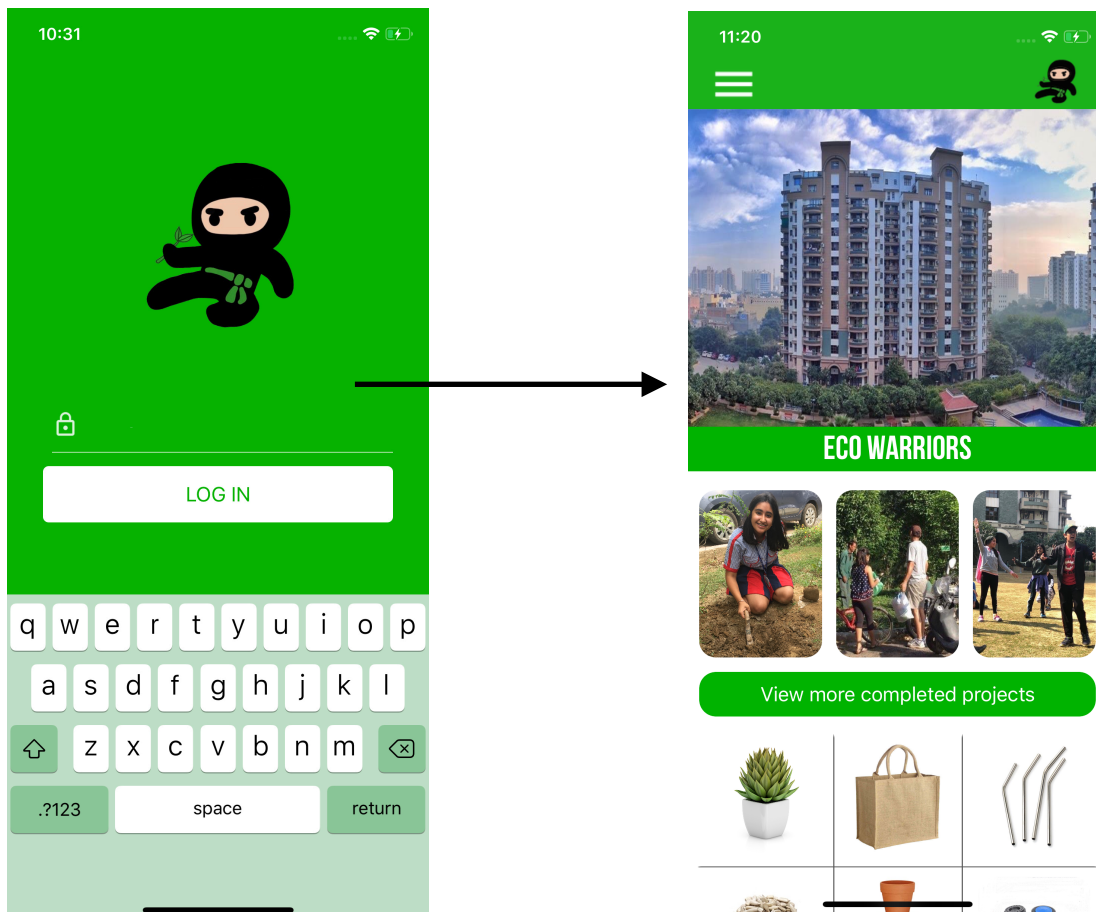


4. If-then

A password system was created to prevent unauthorised users from accessing the content of the app.

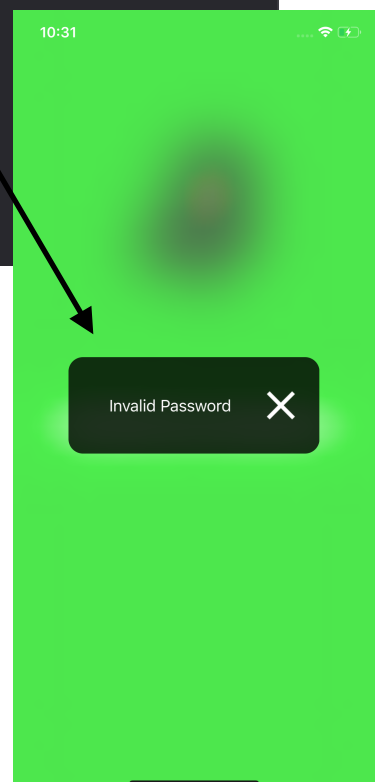
```
override func viewDidLoad() {  
    super.viewDidLoad()  
    configureViewComponents()  
}  
  
@objc func handleLogin() {  
    guard let password = passwordTextField.text else { return }  
  
    if password == "vgeco" {  
        self.presentingViewController?.presentingViewController?  
            .dismiss(animated: true, completion: nil)  
        self.dismiss(animated: true, completion: nil)  
    }  
}
```

The above code states that if the right password is entered, then user would be able to have access to the rest of the app and the home page would open.

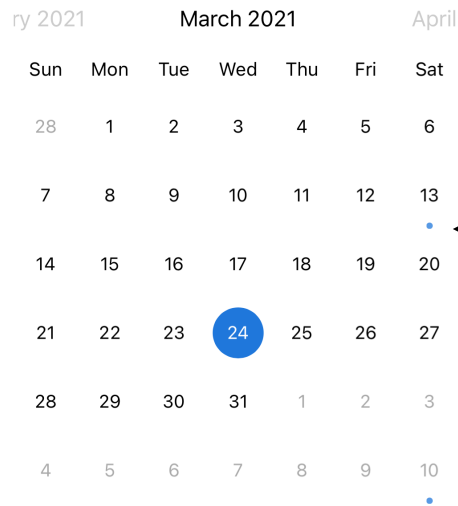
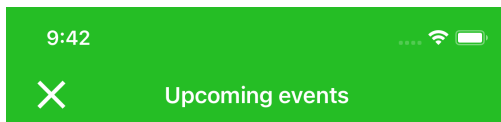


But if the wrong password is entered then, a pop up box with the text “ invalid password” will appear

```
49     else {
50         errorPopup()
51     }
52 }
53
54 var addButton = UIButton()
55 let blurEffectView = UIVisualEffectView(effect: UIBlurEffect(style:
    UIBlurEffect.Style.light))
56 var removeButton = UIButton()
57 let addedLabel = UILabel()
58 let addedImage = UIImageView()
59
60 func errorPopup() {
61     blurEffectView.frame = view.bounds
62     blurEffectView.autoresizingMask = [.flexibleWidth, .flexibleHeight]
63
64     addButton.frame = CGRect(x: (screenWidth/2)-130, y: 360, width: 260, height:
        100)
65     addButton.layer.cornerRadius = 15;
66     addButton.clipsToBounds = true
67     addButton.backgroundColor = UIColor.black.withAlphaComponent(0.8)
68     addButton.setTitle("", for: .normal)
69
70     addedLabel.textColor = .white
71     addedLabel.text = "    Invalid Password"
72     addedLabel.frame = CGRect(x: (screenWidth/2)-110, y: 360, width: 230, height:
        100)
73
74     addedImage.frame = CGRect(x: (screenWidth/2)+65, y: 385, width: 50, height: 50)
75     addedImage.image = UIImage(named:"Cross")
76
77     removeButton.addTarget(self, action: #selector(popupRemove), for:
        .touchUpInside)
78     removeButton.frame = CGRect(x: 0, y: 0, width: screenWidth, height:
        screenHeight)
79
80     UIView.transition(with: self.view, duration: 0.25, options:
        [.transitionCrossDissolve], animations: {
81         self.view.addSubview(self.blurEffectView)
82         self.view.addSubview(self.addButton)
83         self.view.addSubview(self.removeButton)
84         self.view.addSubview(self.addedLabel)
85         self.view.addSubview(self.addedImage)
86     }, completion: nil)
87 }
```



5. Arrays, Loops and Exit Conditions



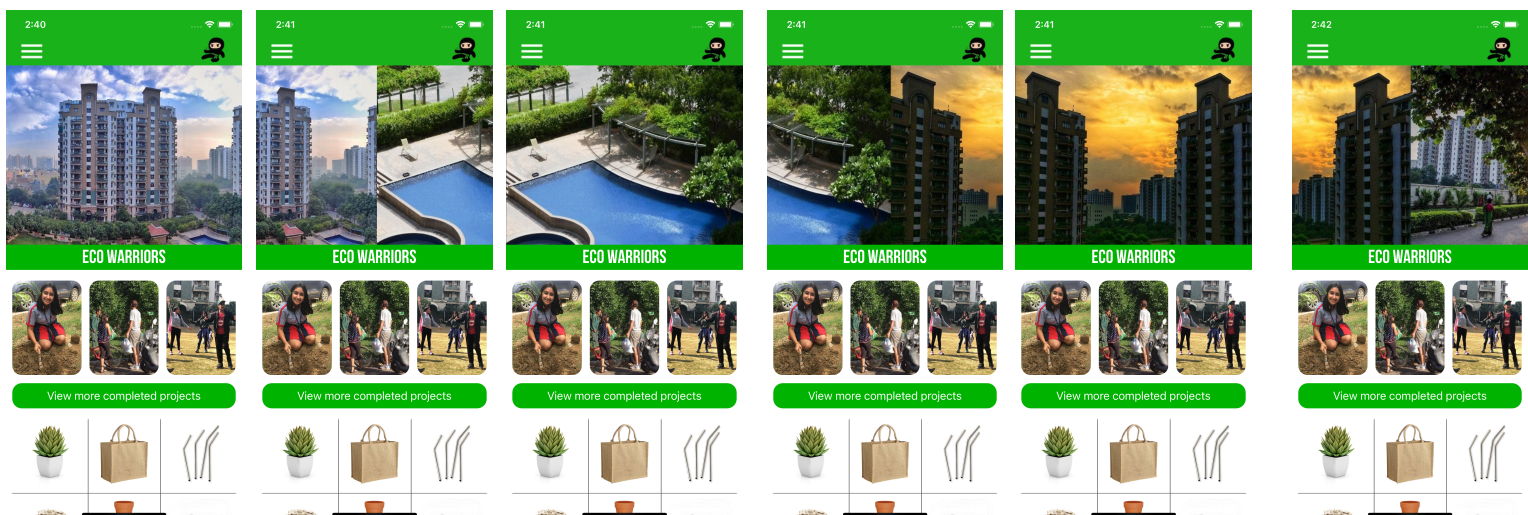
For the FSC calendar, dates that had upcoming events were marked with a dot.

Dates that have to be marked were put into an array

```
var datesWithEvent =  
  ["2021-01-09", "2021-02-13", "2021-03-13", "2021-04-10",  
   "2021-05-08", "2021-06-12"]
```



Another application of arrays was in the image gallery on the home page. It is a series of images put together that can be viewed by swiping on it from right to left.



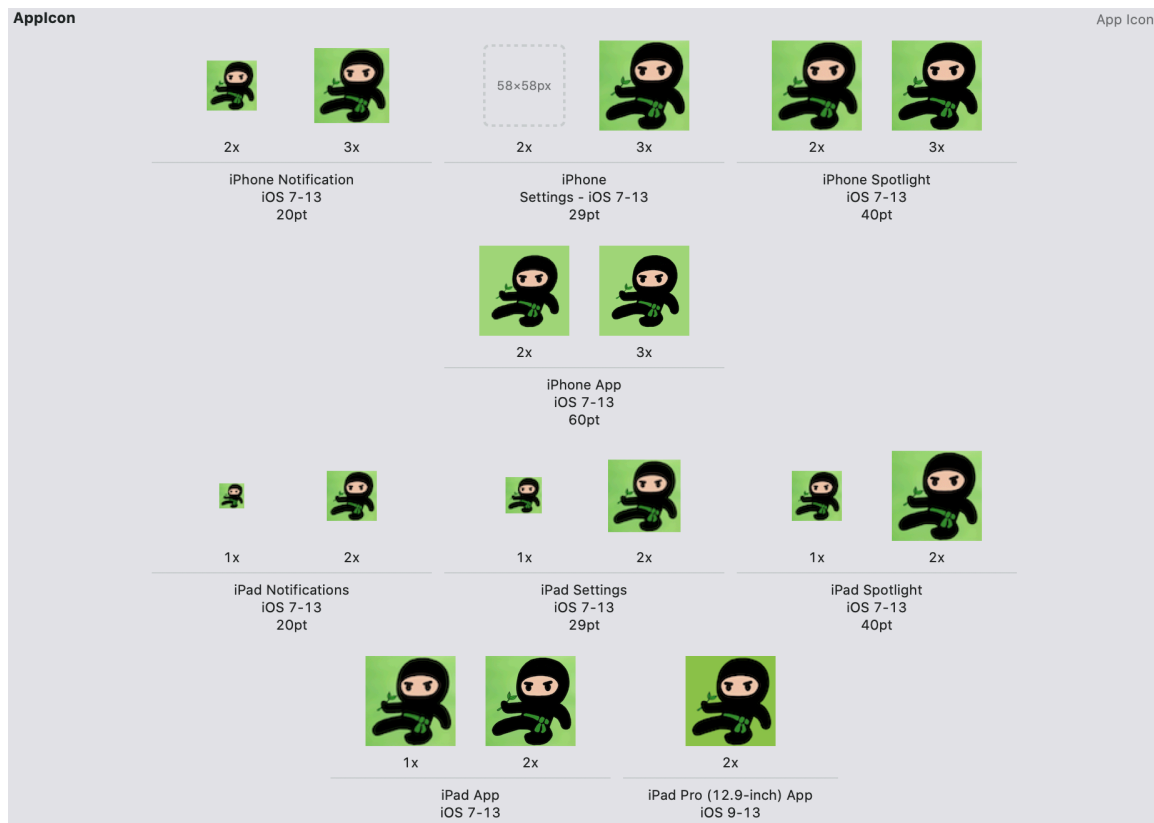
This also follows the concept of arrays. The below code puts all images one by one in a loop.

```
let mainScreenView = UIScrollView()
    imageArray = [🐼, 🐼, 🐼, 🐼]
    mainScreenView.frame = (CGRect(x: 0, y: 0, width:
        screenWidth, height: 325))
```

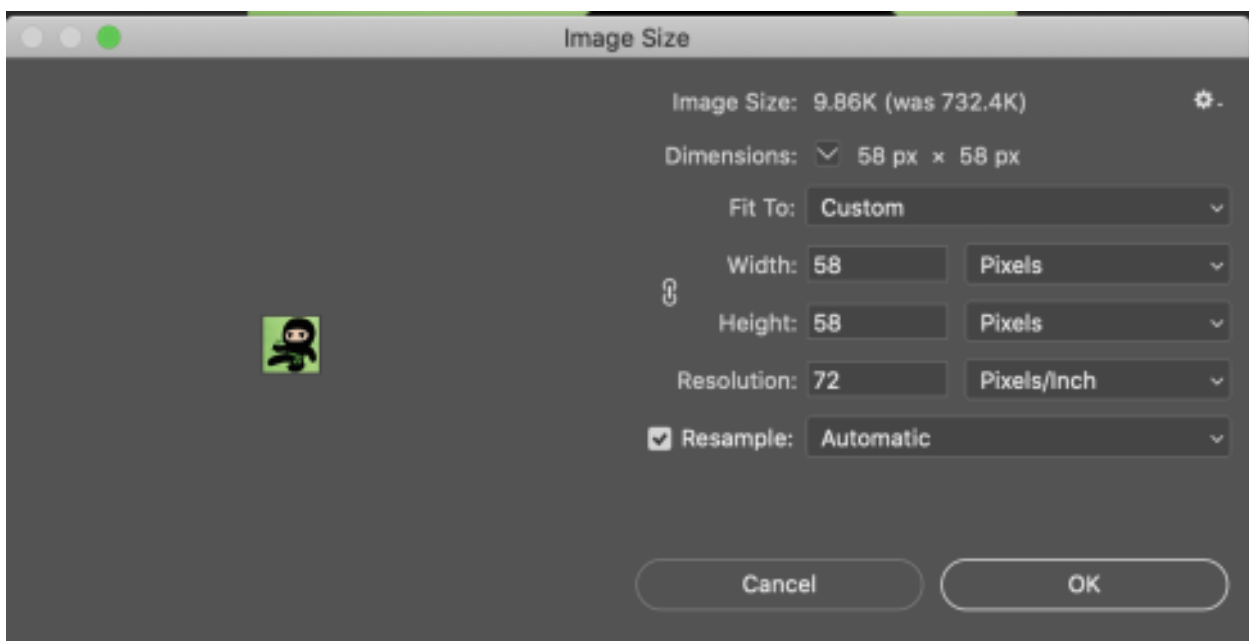
The exit condition for the loop is that, pictures will loop while images are left in the array.

```
for i in 0..
```

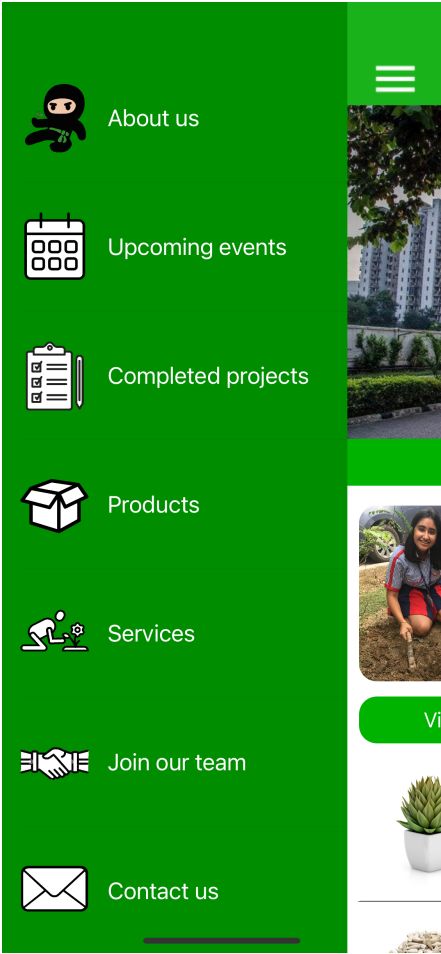
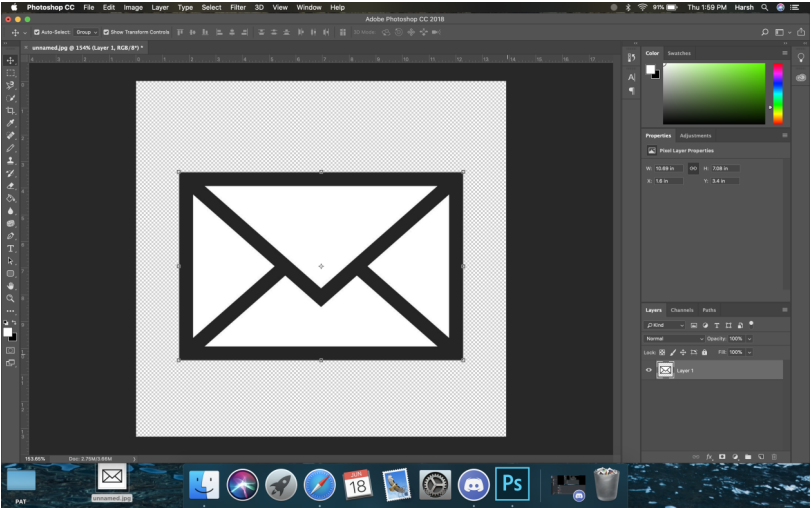
6. Manipulation of graphics involving multiple techniques using Adobe Photoshop



The pixel size of the logos were reduced according to apple device to ensure that the application icon looks apt in all apple devices.

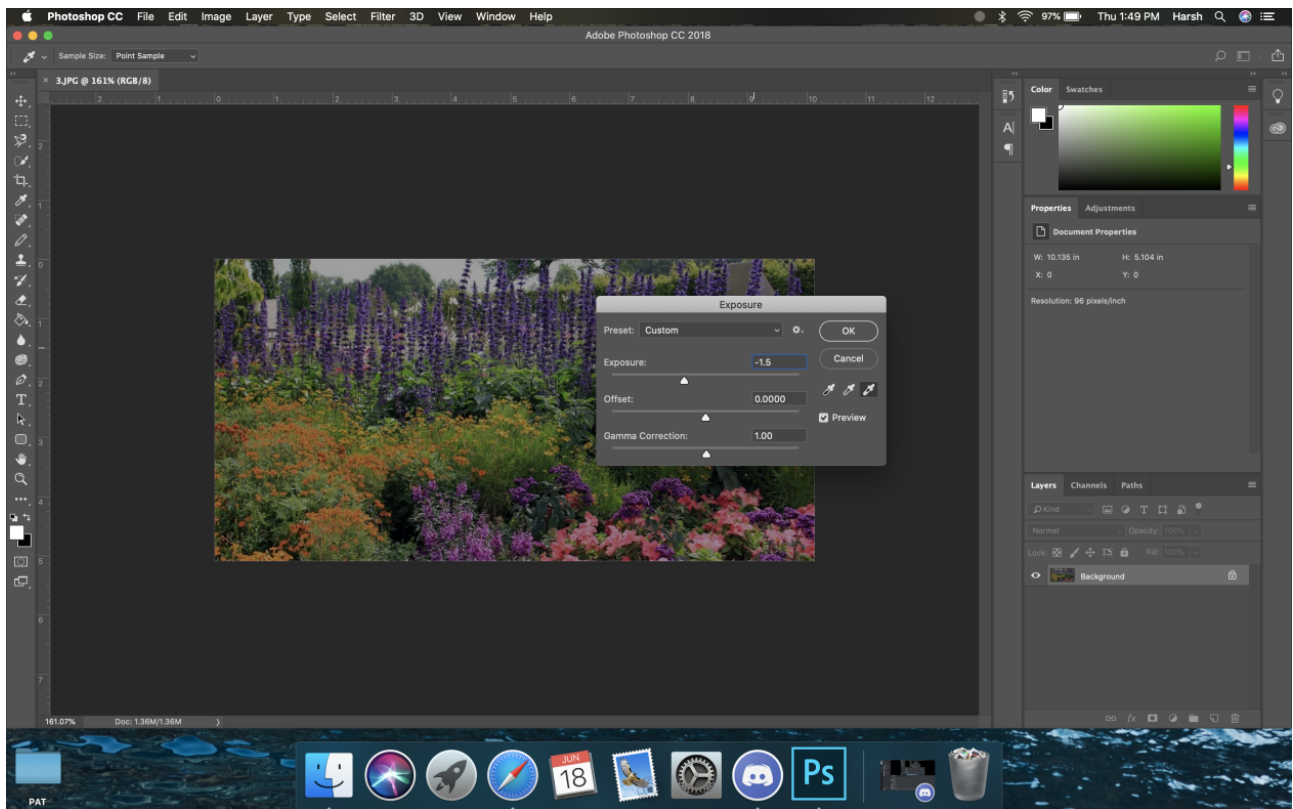


Icons were converted to PNG (background was removed) so they look professional when displayed on the app.

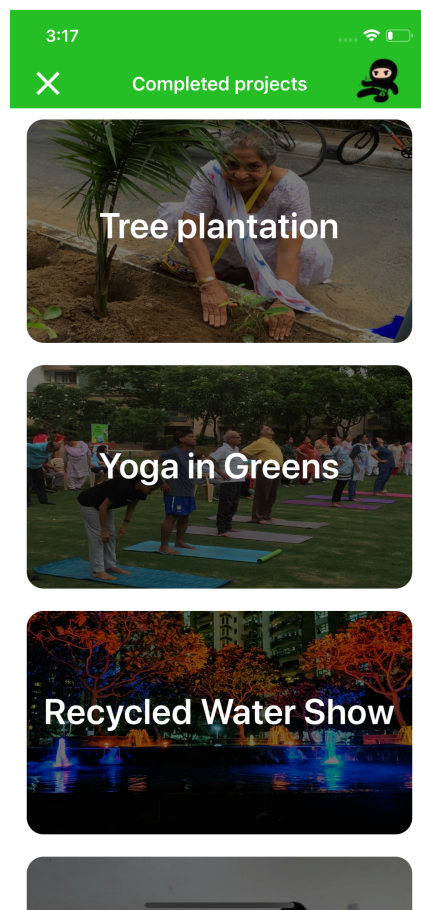
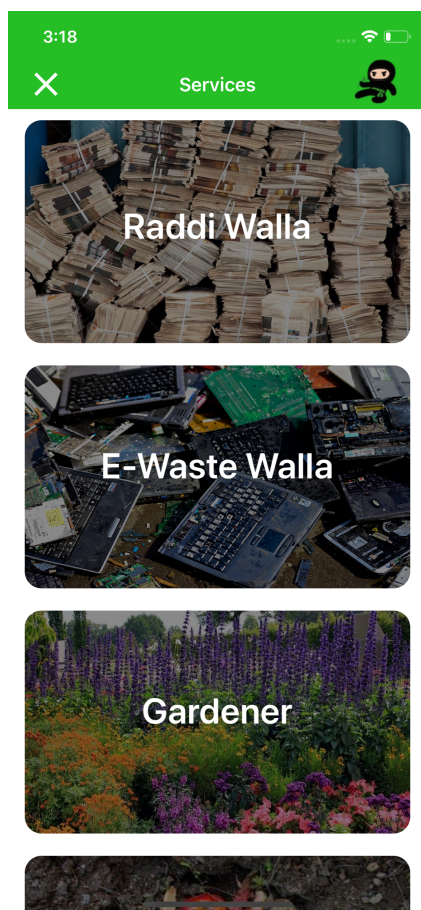


The exposure of the pictures of the “completed projects” and “services” main screen was reduced to -1.5.





This was done so that the focus of the user goes on the text which explains what the project / service is.





The above images show the before and after results of using the adjustments of brightness, contrast, vibrance and saturation .

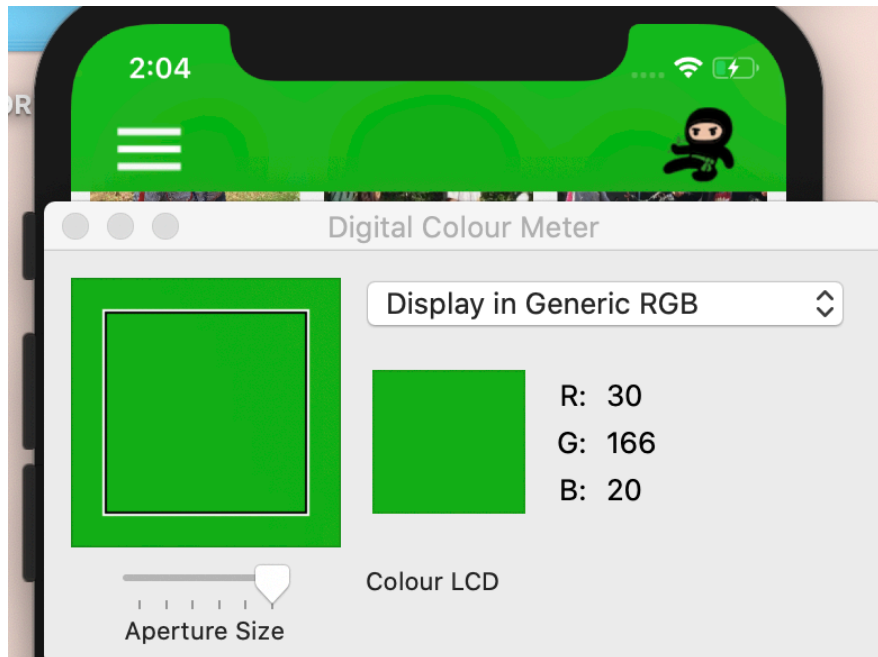
To make these changes the following techniques were used:

- Brightness was increased by 20%
- Contrast was increased by 20%
- Vibrance was incensed by 30%
- Saturation was increased by 15%

This was done to multiple pictures in the app to make it more appealing and attractive so the user of the app has a better experience

7. Integration of components using advanced features from Procreate for digital art to create logo

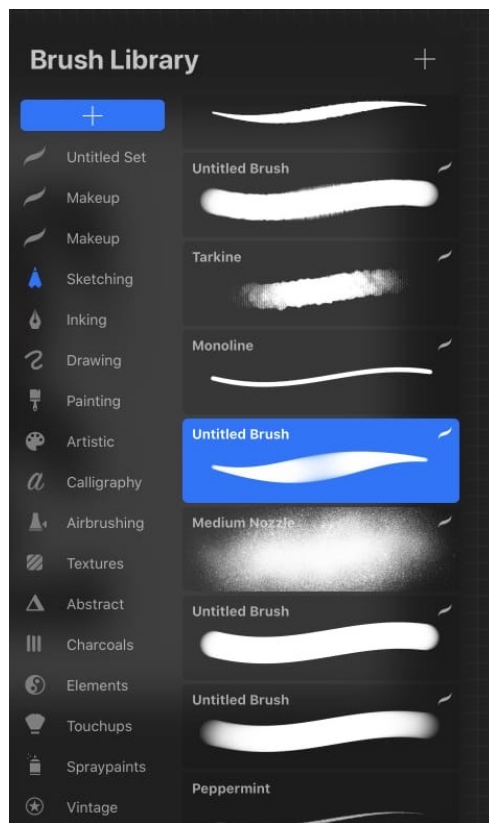
The RGB of the green used in the apps bars, buttons etc. was mainly R:30, G:166, B:20



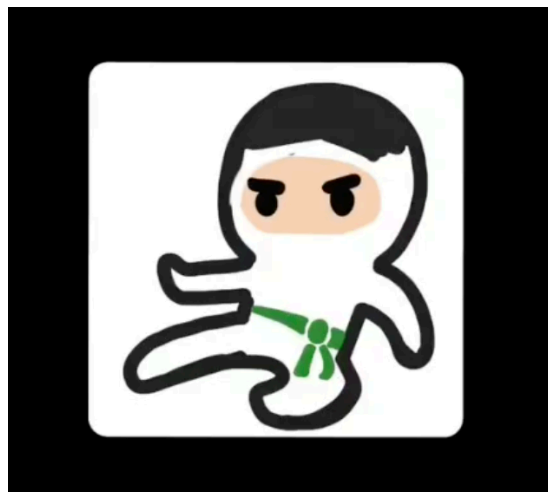
So for the logo I chose R:49, G:126, B:48 mainly using the color disk. The shades of green are kept differently so the logo is visual on the top bar of the app.



I then decided on the highlighted brush stork

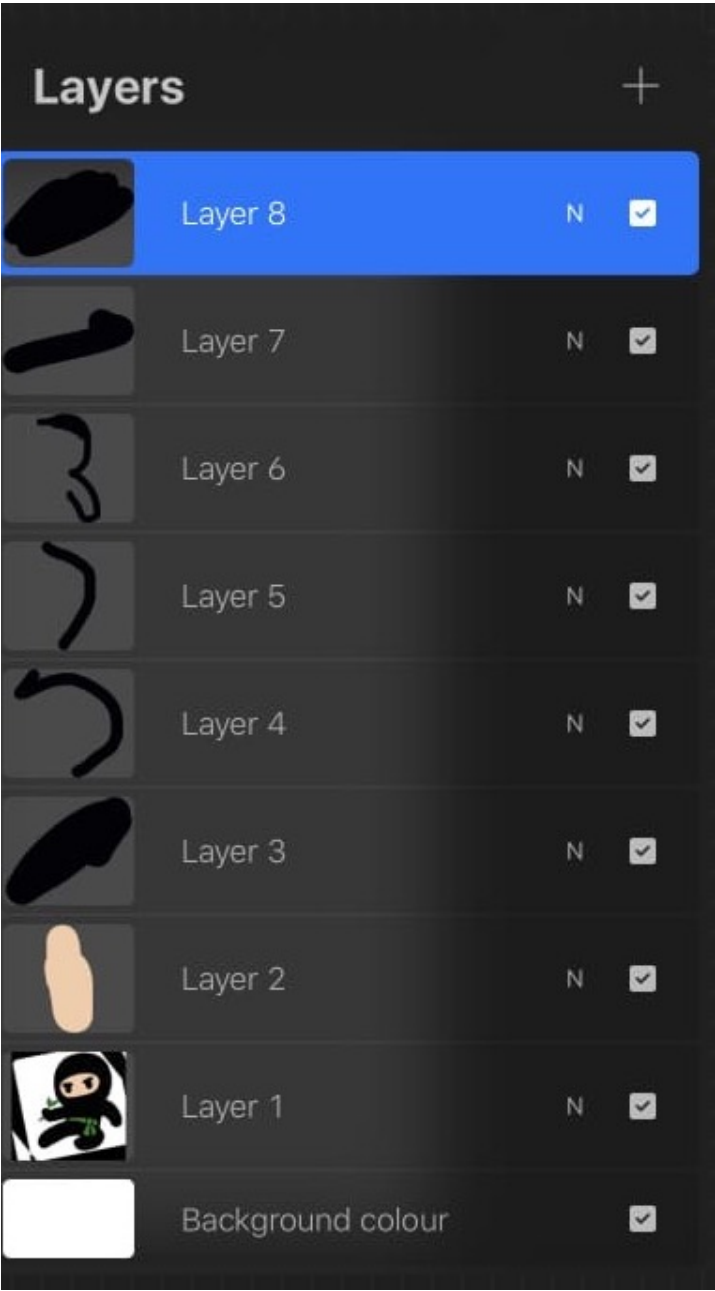


I sketched out the drawing using a stylus.





Appropriate layers were created to move the objects around (bring to front/ bring to back) while sketching out the drawing.



Word count: 975