Bioinformatics

Lefteris Koumakis,
Maria Chatzimina

Lecture 6

Use Mobile Camera

- Package:
  - Image_picker
    - url: https://pub.dev/packages/image_picker/install
- imageSource can be either one:
  - Gallery
  - Camera
- If in the emulator camera is not working we are running in folder “emulator (in Android SDK path) the command line:
  - emulator -camera-front webcam0 -avd το_ονομα_του_emulator

```
//ειναι ασυγχρονη συναρτηση που επιστρεφει ενα future γιαυτο δηλων
Future<void> _takePicture() async {
  final picker = ImagePicker();
  final imageFile = await picker.pickImage( //ImageSource.camera (
      source: ImageSource.camera, maxWidth: 600, maxHeight: 600);

}
```

Use Mobile Camera

- In packages that are using native parts of the mobile such as microphone, camera etc. we read the “Read me” section to find the permissions needed
- Camera permission for example :
  - Android:
    - No configuration required - the plugin should work out of the box.
  - iOS:
    - Add the following keys to your Info.plist file, located in <project root>/ios/Runner/Info.plist:
      - NSPhotoLibraryUsageDescription - describe why your app needs permission for the photo library. This is called Privacy - Photo Library Usage Description in the visual editor.
      - NSCameraUsageDescription - describe why your app needs access to the camera. This is called Privacy - Camera Usage Description in the visual editor.
      - NSMicrophoneUsageDescription - describe why your app needs access to the microphone, if you intend to record videos. This is called Privacy - Microphone Usage Description in the visual editor.

Use Mobile Camera - iOS

- In info.plist we add information such as:
  - App version
  - permissions needed
  - The text to be shown when the user is informed about the permissions needed

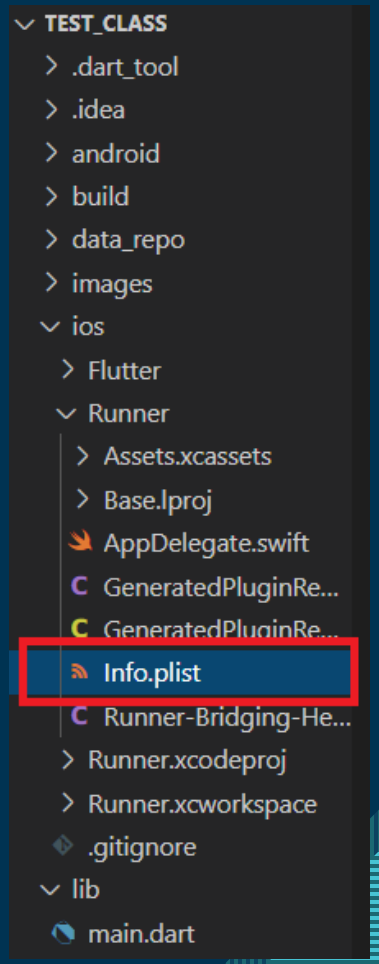
```
<key>LSRequiresIPhoneOS</key>
<true/>
<key>NSPhotoLibraryUsageDescription</key>
<string>We need access to your Photo Library!</string>
<key>NSCameraUsageDescription</key>
<string>We need access to your camera</string>
<key>UILaunchStoryboardName</key>
<string>LaunchScreen</string>
<key>UIMainStoryboardFile</key>
<string>Main</string>
```

Permissions needed

Displayed text to the user

Permissions needed

Displayed text to the user



Save Image to Mobile

- Packages:
  - path_provider : provides mobile paths
    - https://pub.dev/packages/path_provider
  - path : creates paths
    - https://pub.dev/packages/path

```
class _MyAppState extends State<MyApp> {
  late File _storedImage ;
  var image;
  bool imagefile = false;

  //ειναι ασυγχρονη συναρτηση που επιστρεφει ενα future γιαυτο δηλωνουμε οτι ειναι και async
  Future<void> _takePicture() async {
    final picker = ImagePicker();
    final imageFile = await picker.pickImage( //ImageSource.camera δηλωνει οτι θελουμε να παρουμε φωτογραφια απο την καμερα
        source: ImageSource.camera, maxWidth: 600, maxHeight: 600);
    final appDir = await syspath.getApplicationDocumentsDirectory(); //ο φακελος που αποθηκευονται οι εικονες(οχι προσωρινα)
    if (imageFile != null){
      print("inside");
    final fileName = path.basename(imageFile.path); //το path που αποθηκευεται η εικονα προσωρινα
    File file = File(imageFile.path);
    final savedImage = await file.copy('${appDir.path}/$fileName'); //ετσι περναμε σαν string τις τιμες των μεταβλητων ( με το $)
    print(savedImage.path); //τυπωνω το path που αποθηκευτηκε στην συσκευη
    setState(() {
      _storedImage = File(imageFile.path);
      imagefile = true;
    });
    }
  }
```

Save Image to Mobile

- FlatButton.icon
  - Creates a button with an icon
- Expanded:
  - Using an expanded widget on the child of a Row or Column causes it to expand to fill the available space along the main axis (eg horizontally for Row or vertically for Column).

```
return Scaffold(
  appBar: AppBar(
    title: Text("Image"),
  ), // AppBar
  body: Row( //παρομοιο με το column απλα η στοιχιση ειναι το ενα widget διπλα στο αλλο
    children: <Widget>[
      Container(
        width: 150,
        height: 100,
        decoration: BoxDecoration(
          border: Border.all(width: 1, color: Colors.grey),
        ), // BoxDecoration
        child: imagefile != false //αν η εικονα απο την καμερα εχει δημιουργηθει
            ? Image.file(  //εμφανισε τη στο κουτι
                _storedImage,
                fit: BoxFit.cover,
                width: double.infinity,
              ) // Image.file
            : Text(  //διαφορετικα εμφανισε ενα κειμενο
                'No Image Taken',
                textAlign: TextAlign.center,
              ), // Text
        alignment: Alignment.center,
      ), // Container
      SizedBox(
        width: 10,
      ), // SizedBox
      Expanded(
        child: TextButton.icon( //χρησιμοποιουμε ενα κουμπι με εικονιδιο
          icon: Icon(Icons.camera), //το εικονιδιο της καμερας
          label: Text('Take Picture'), // το κειμενο που θα φαινεται στο κουμπι
          onPressed: _takePicture, //η future συναρτηση _takePicture που δηλωσαμε παραπανω
        ), // TextButton.icon
      ), // Expanded
    ], // <Widget>[]
  ), // Row
); // Scaffold
```

Useful Links

- Icons list: https://api.flutter.dev/flutter/material/Icons-class.html
- Charts package: https://pub.dev/packages/charts_flutter#-readme-tab-
- Available charts: https://google.github.io/charts/flutter/gallery.html

UI Design

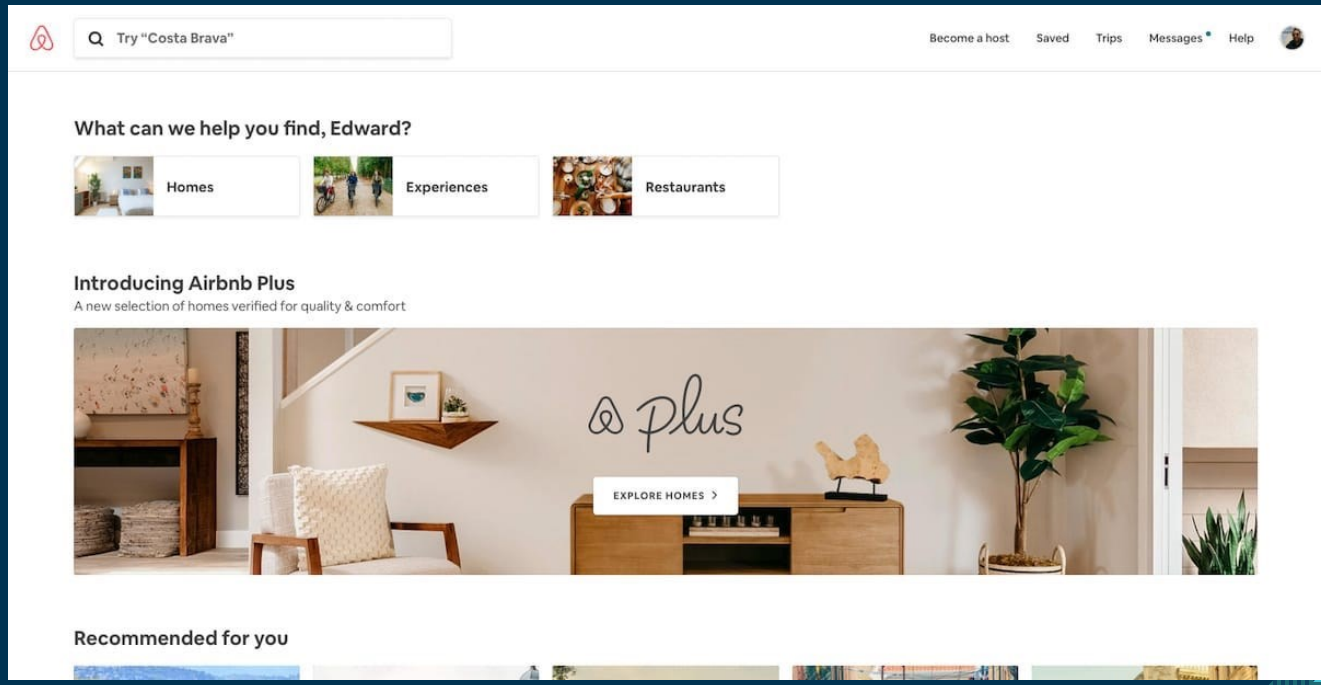
- How do we know that a UI Design is good?
  - **Learnability**: How easily can users complete their tasks?
  - **Efficiency:** The interface allows the user to complete tasks promptly/on time?
  - **Memorability**: After leaving the website, how likely are users to remember how to use it when they return?
  - **Errors**: What steps does the interface take to minimize user errors, and how does it support users in correcting errors?
  - **Satisfaction**: Do users enjoy interacting with the website?

Successful Examples: Medium

- Online reading and publishing platform → Medium
  - Minimal use of color
  - Generous line spacing
  - Well chosen combination of typography
  - The articles are in a single column format which makes it easy to read
  - Estimated reading time
  - Direct commenting on specific parts of the articles

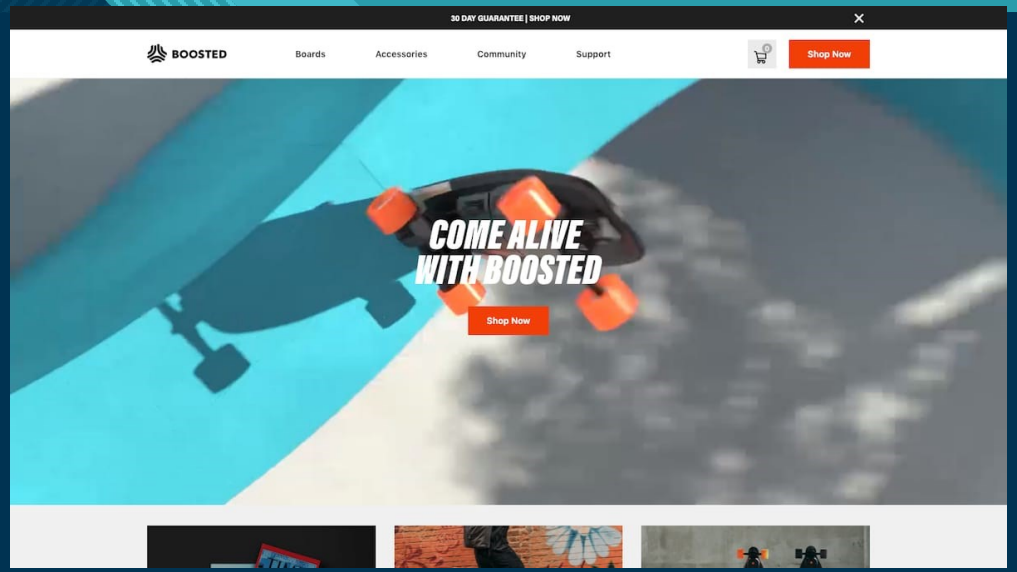
Successful Examples: Airbnb

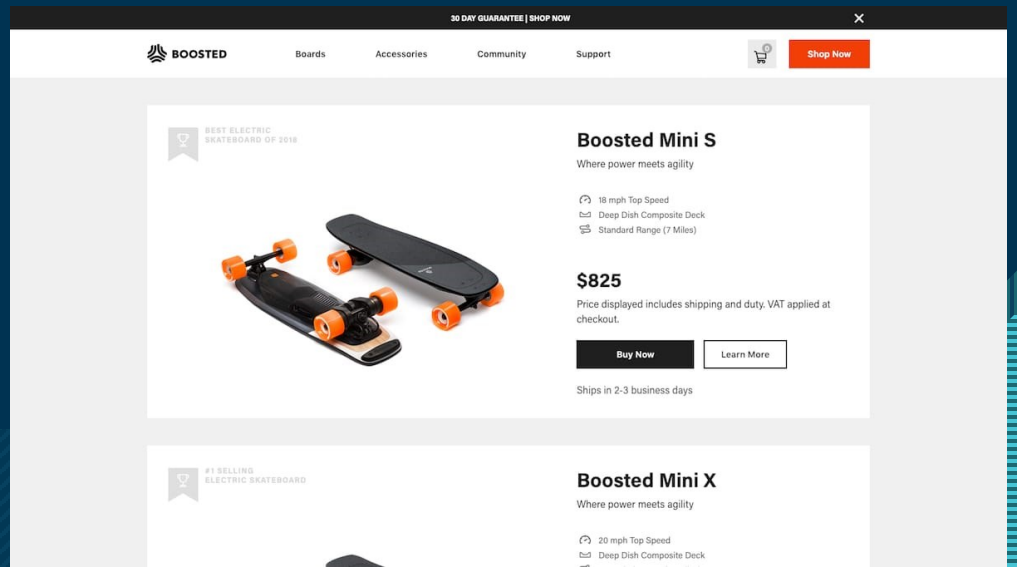
- It accomplishes two things
  - Book a place to stay
  - Building trust between two complete strangers.
- Simple process of finding accommodation
- Clean design
- Helpful hint text
  - "What can we help you find, Edward?"



Successful Examples: Boosted Boards

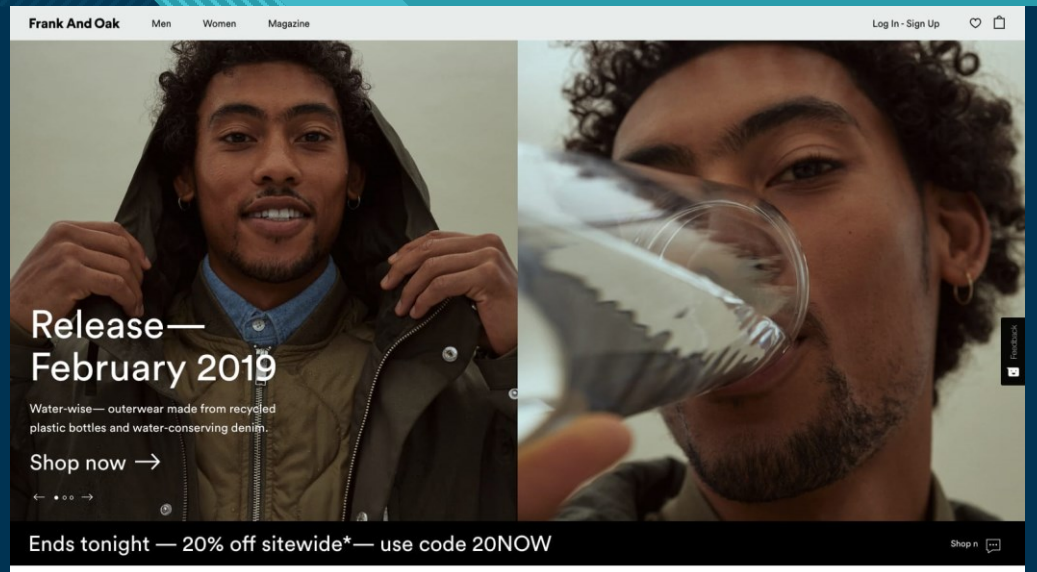
- Users are immediately welcomed with a dynamic product video.
- The interface is free of color
- Only the most relevant information is provided: image, maximum speed, range, price, shipping time and, of course, "Buy Now".

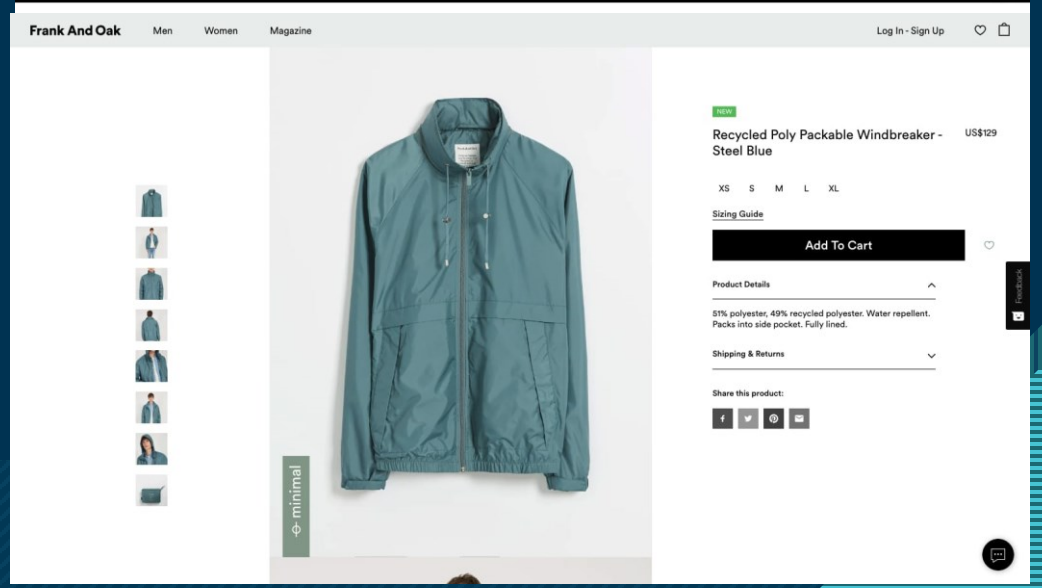




Successful Examples : Frank and Oak

- It uses the user interface to grab users' attention
- The company uses bold photography and clean, sans-serif typography
- After clicking on an item, the product page focuses on two things: the product photo and the buy button, as they are essentially the only colored elements on the page.





Best Practices for Designing an Interface

- Keep the interface simple. The best interfaces are almost invisible to the user. They avoid unnecessary elements and are clear in the language they use in labels and messages.
- Use color and texture strategically. You can direct attention or redirect attention away from objects by using color, light, contrast and texture to your advantage.
- Use typography to create hierarchy and clarity. Consider carefully how you use the font. Different sizes, fonts and layout of text to help increase legibility and readability.

Best Practices for Designing an Interface

- Be intentional about your page layout. Consider the spatial relationships between elements on the page and structure the page based on their importance. Careful placement of items can help draw attention to the most important information and can aid readability.
- Make sure the system communicates what is happening. Always keep your users informed of actions, status changes, or errors. Using various user interface elements to communicate the status and, if necessary, next steps can reduce user frustration.

Best Practices for Medical and Healthcare Apps

- Consider the importance of color combinations
- The medical color palette is quite limited. It’s difficult to imagine a healthcare organization where neon yellow is the leading color. This is due to the influence of color on the mood and wellbeing of patients.
- Flashy and vibrant colors can lead to feelings of aggression. A medical application should convey a friendly attitude and be therapeutic and calm. Therefore, it’s best to use cold or pastel shades.

Colors

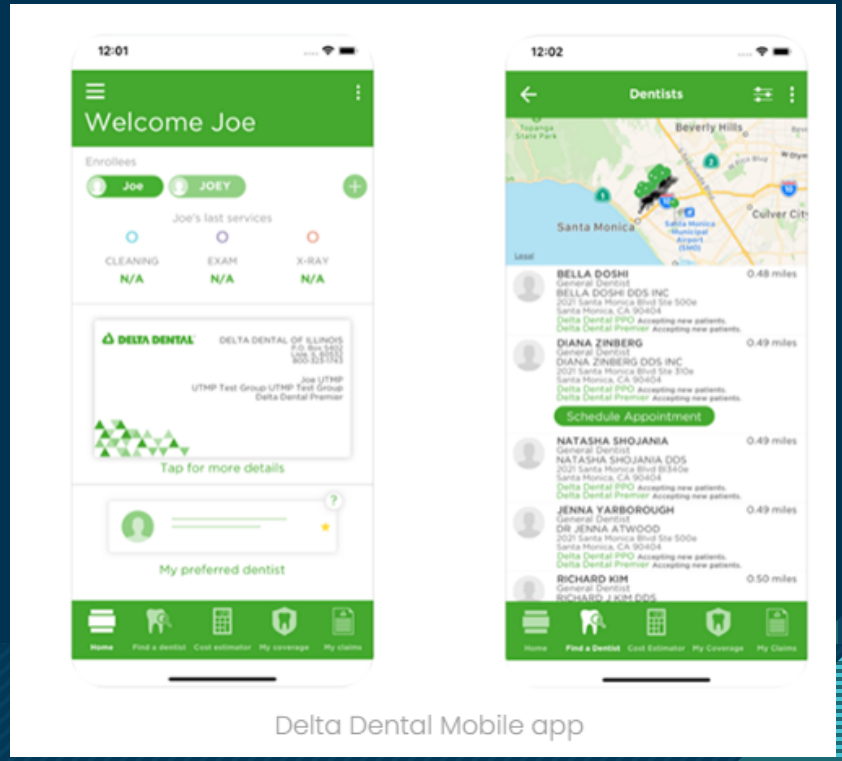
Blue

- Blue is a natural color, and you can often find it in the sky and in water. Perhaps that's why blue is associated with calm and serenity.

eClinicalMobile app

Green

- Green symbolizes the world of nature. It's believed that green removes stress and improves wellbeing.

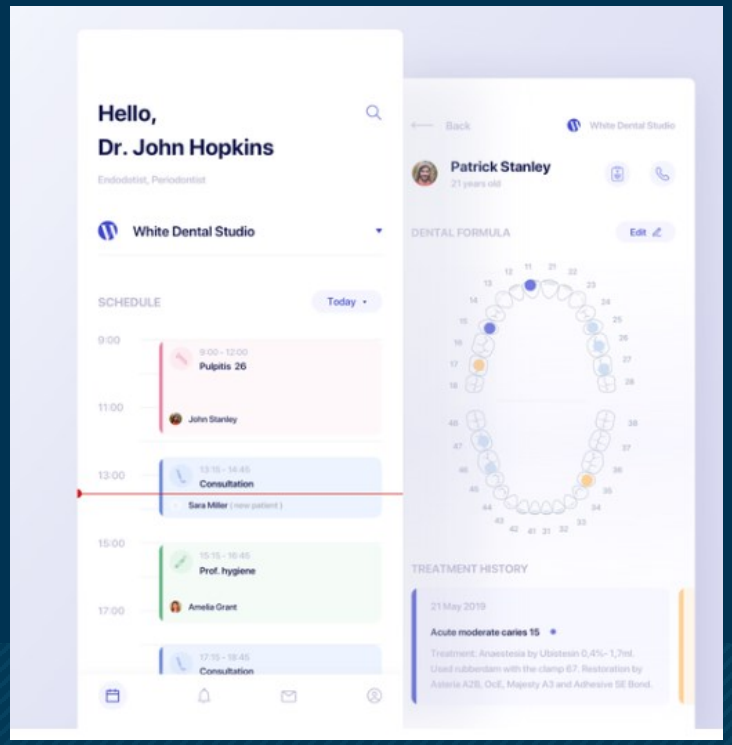


Delta Dental Mobile app

Colors

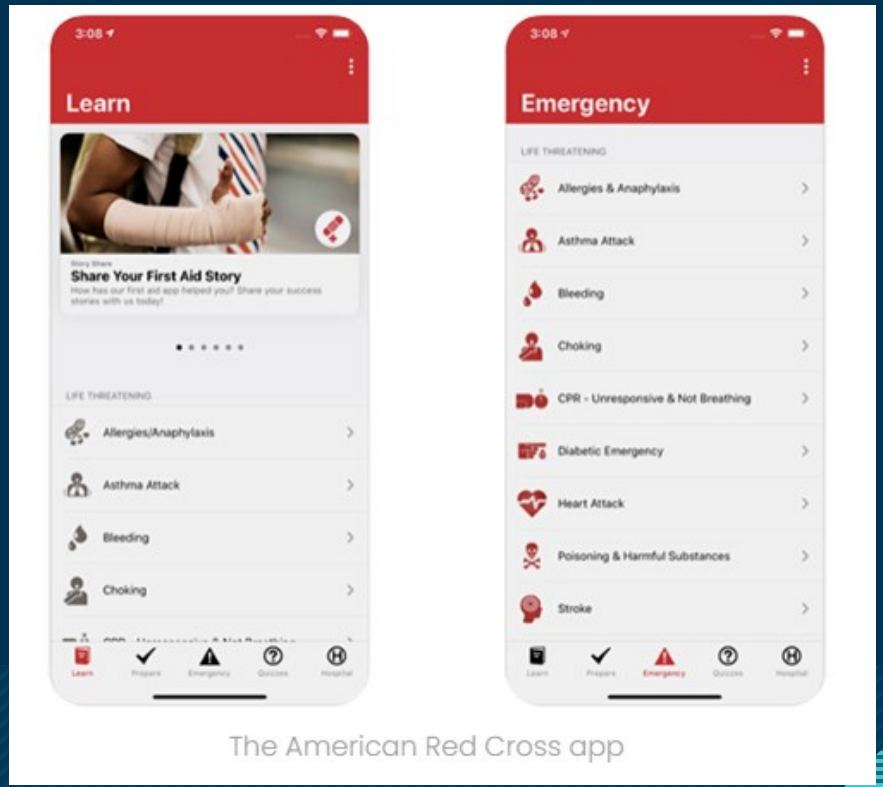
White

- White is often used to convey a sense of purity and peace. It evokes positive emotions and calms users.



Red

- Red is risky because this color may evoke an aggressive mood. But in some cases, red can serve as a great tool.

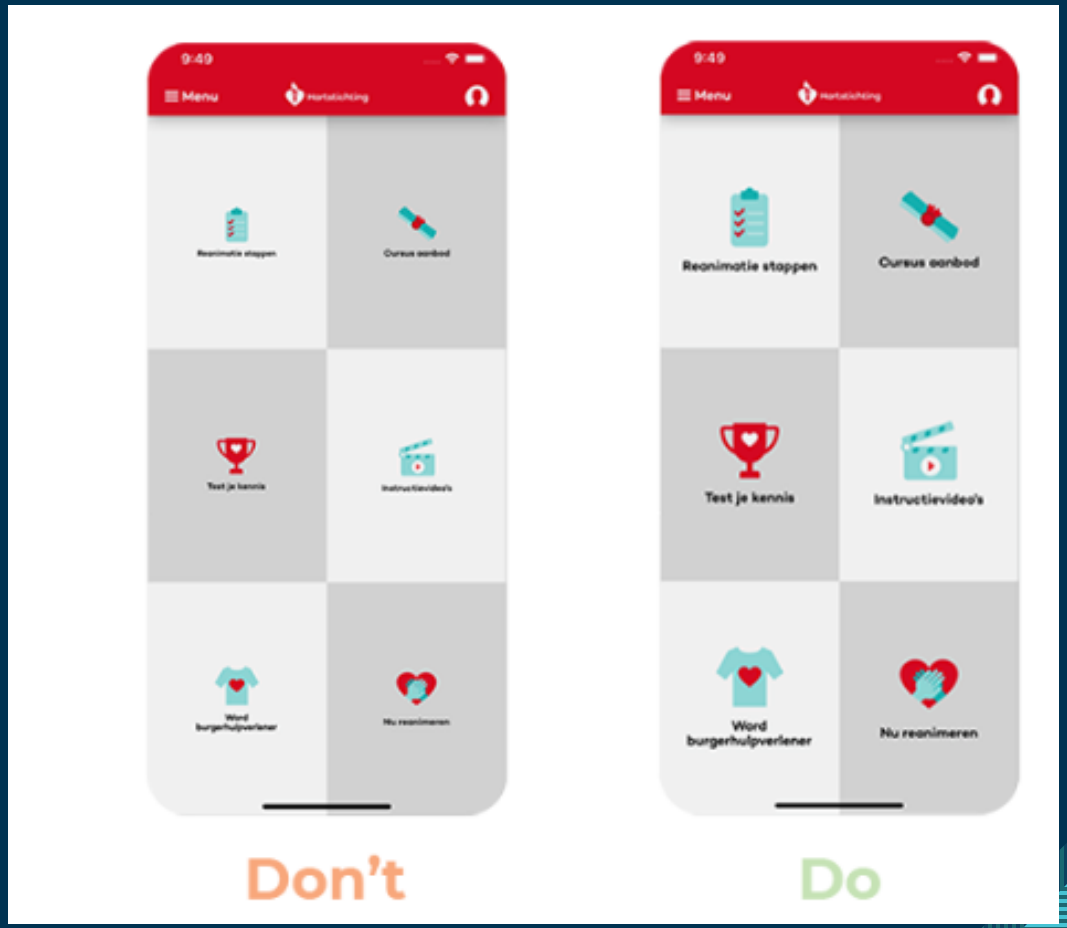


The American Red Cross app

Buttons

Be attentive to buttons

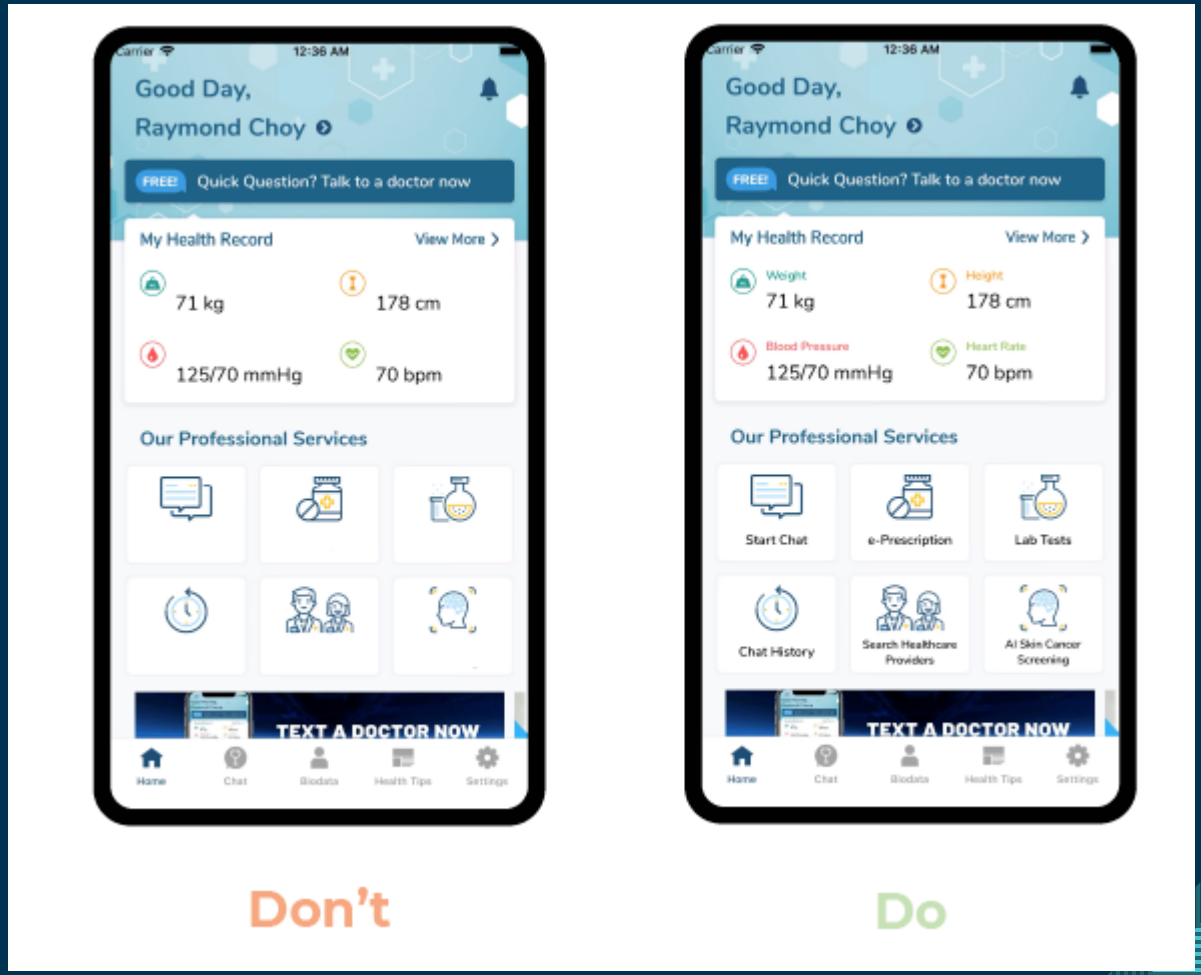
- The size and placement of buttons is an important part of a medical app's UI design since it helps users navigate and perform their desired actions. All buttons should be large enough and located at such a distance that users can interact with them without accidental taps. But avoid making buttons huge, as this will violate the integrity of the layout.



Icons

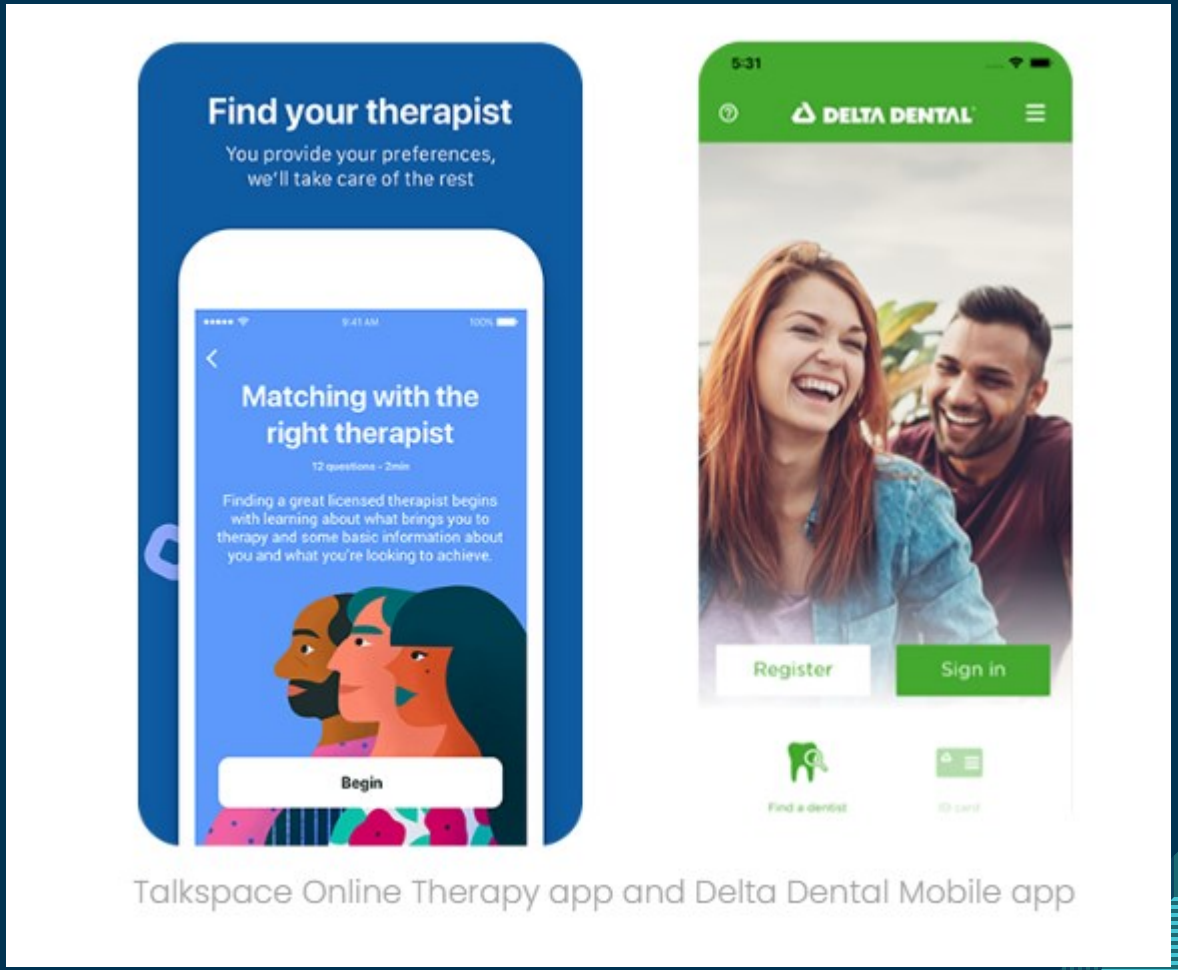
Use responsive icons

- Icons should be intuitive so that users can understand what stands behind them. Although icons are a great graphical element, you shouldn’t use them on each screen, however. Don’t mislead users, and accompany each icon with text.



Images and photos

- Illustrations and photos can also create a pleasant impression when using a medical application. Use high-resolution images to be visually pleasing. Flat illustrations and photos of people can also create a positive vibe.

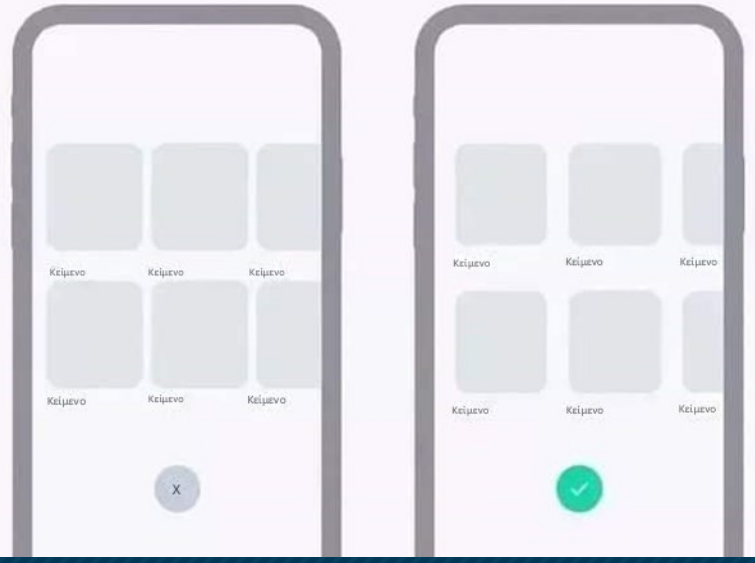


Talkspace Online Therapy app and Delta Dental Mobile app

User Interface Design

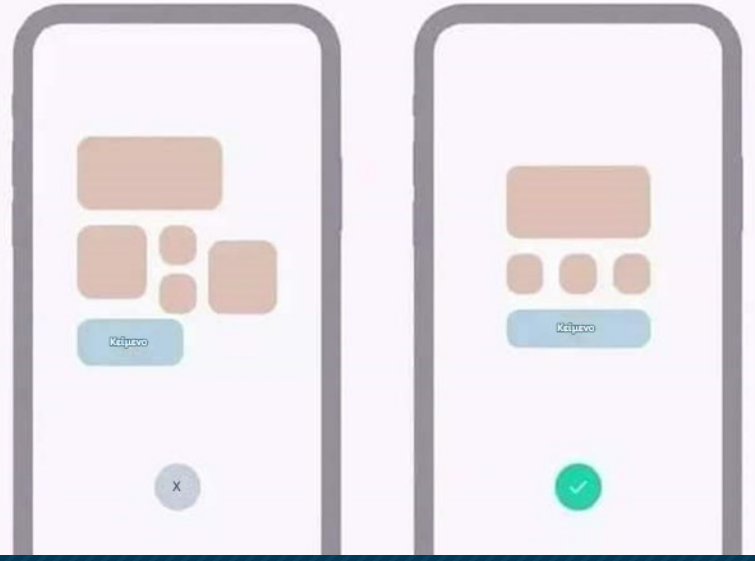
Whitespace

Giving your design space to breathe is essential for easier navigation.



Limiting Choices

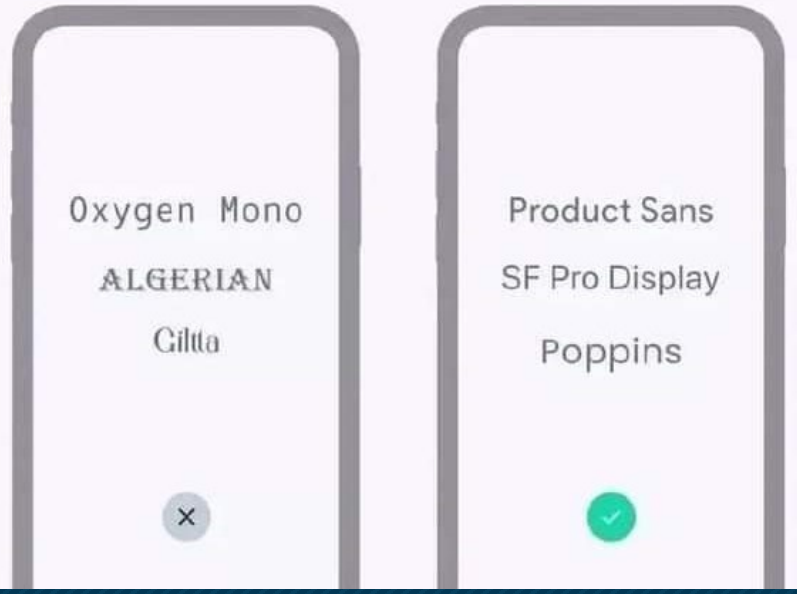
One of the strongest aspects of minimalism in interfaces is improved user focus.



User Interface Design

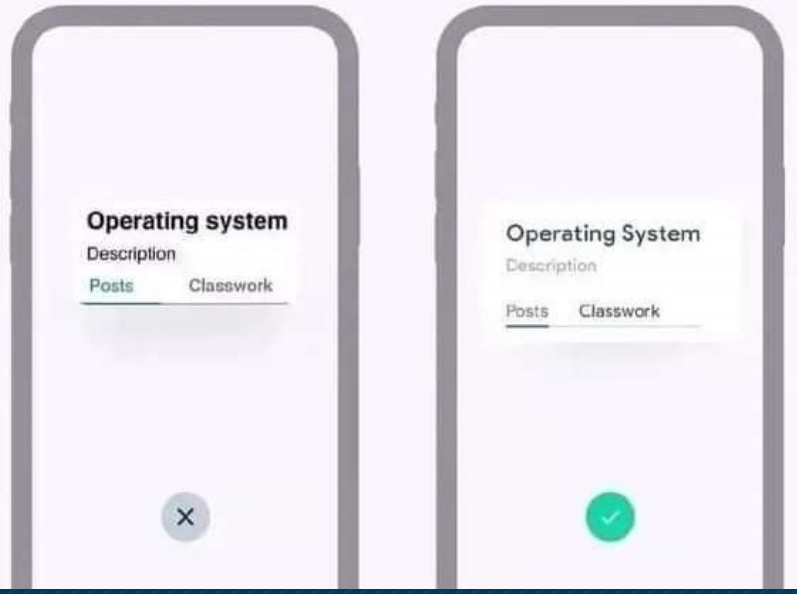
Simple Fonts

Using simple and minimal fonts will make your text — and your overall design — more readable.



Small Details Matter

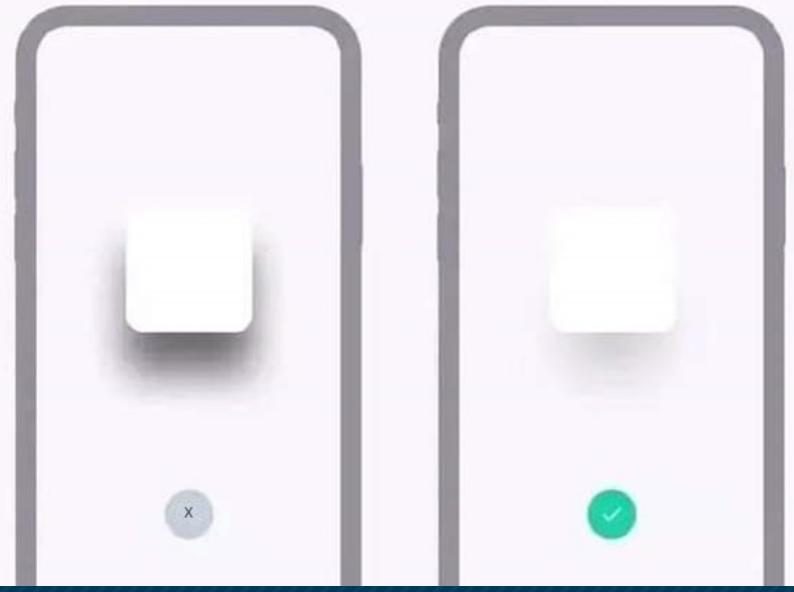
Designers often believe they only need to learn how to use colors and place elements, but it goes much deeper than that.



User Interface Design

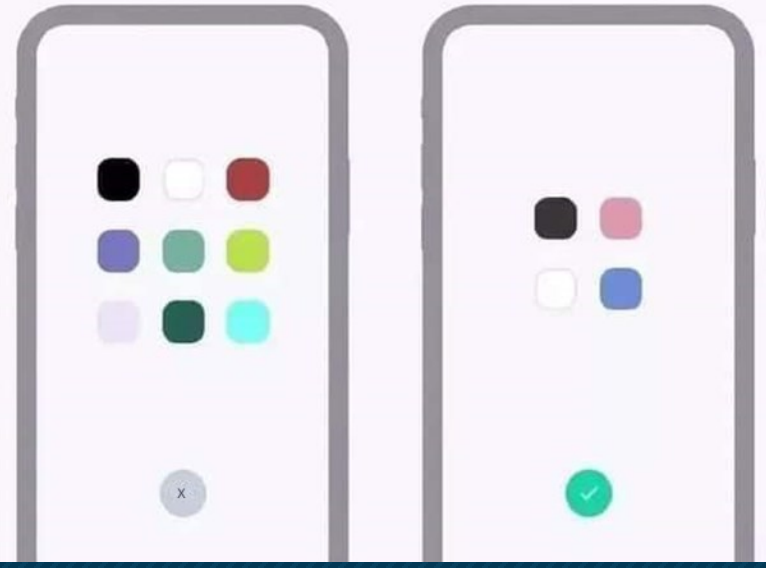
Heavy Shadows

This is a common mistake in UI design, but it often looks better if you use lighter blacks to reduce eye strain and excessive contrast.



Limited Color Palette

This strengthens the selected colors and prevents distracting users with excessive variety.

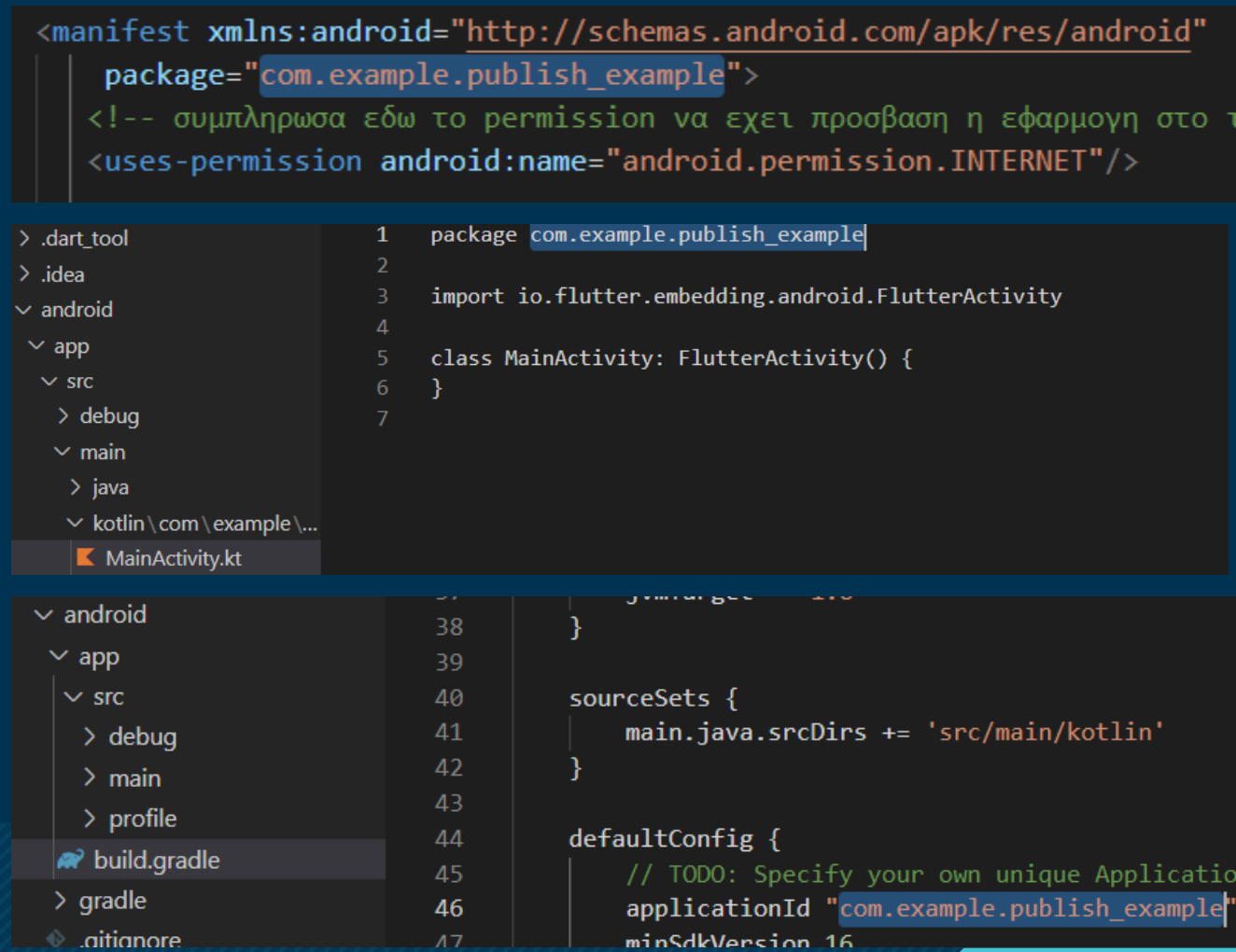


Publish app to Android και iOS store

- Android:
  - Φάκελος android → app → src→ main → AndroidManifest.xml
- Android:label : apps title
- Be sure that you have declared correctly the permissions
- There is also an AndroidManifest.xml in debug folder but this is a different file

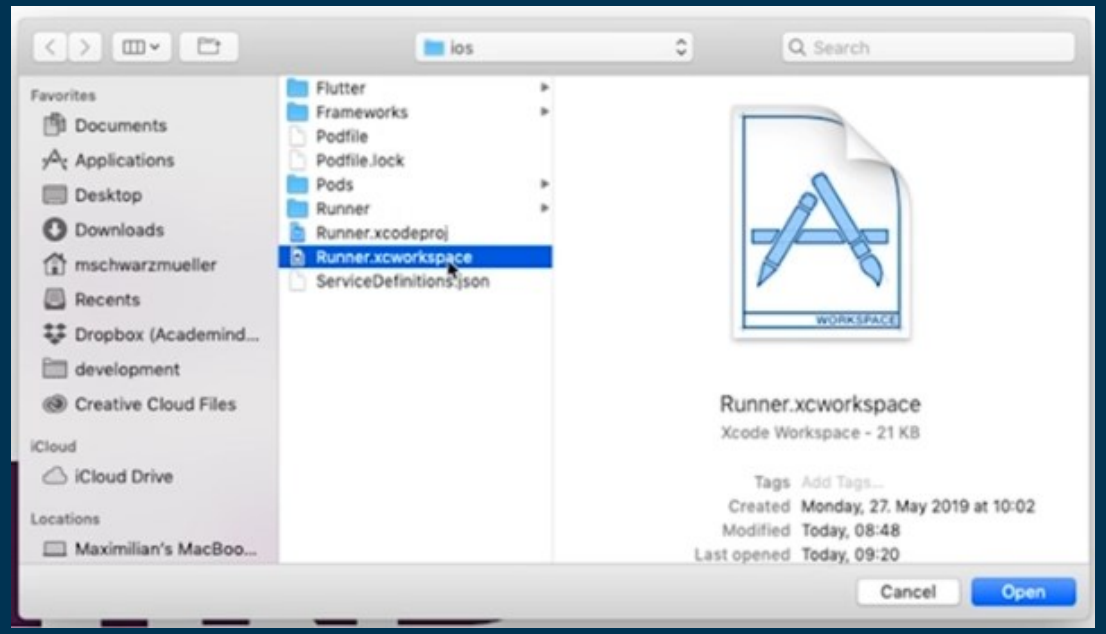
Android

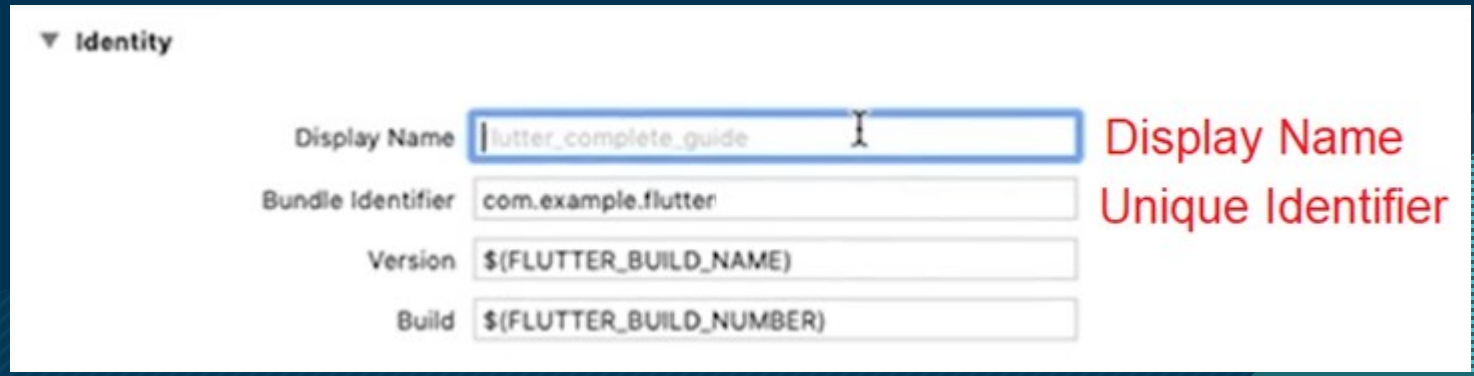
- Package identifier: it should be unique in App Store
- If you change it in AndroidManifest.xml you should change it in all other files such as:
  - android→src→debug→ AndroidManifest.xml
  - android→app→main→kotlin→com→example→publish_example→ MainActivity.kt
  - android→app→build.grandle



iOS

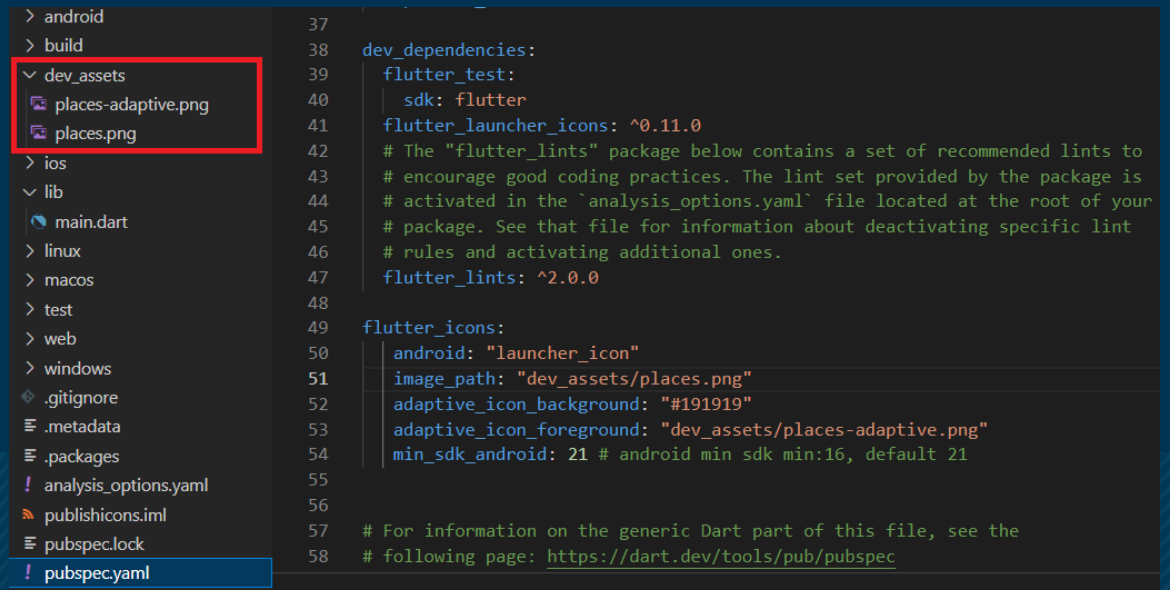
- ios→Runner→Info.plist : we check if all the permissions the app needs are correct
- Package identifier and app title are inserted in XCode
  - open folder: ios→Runner.xcworkspace

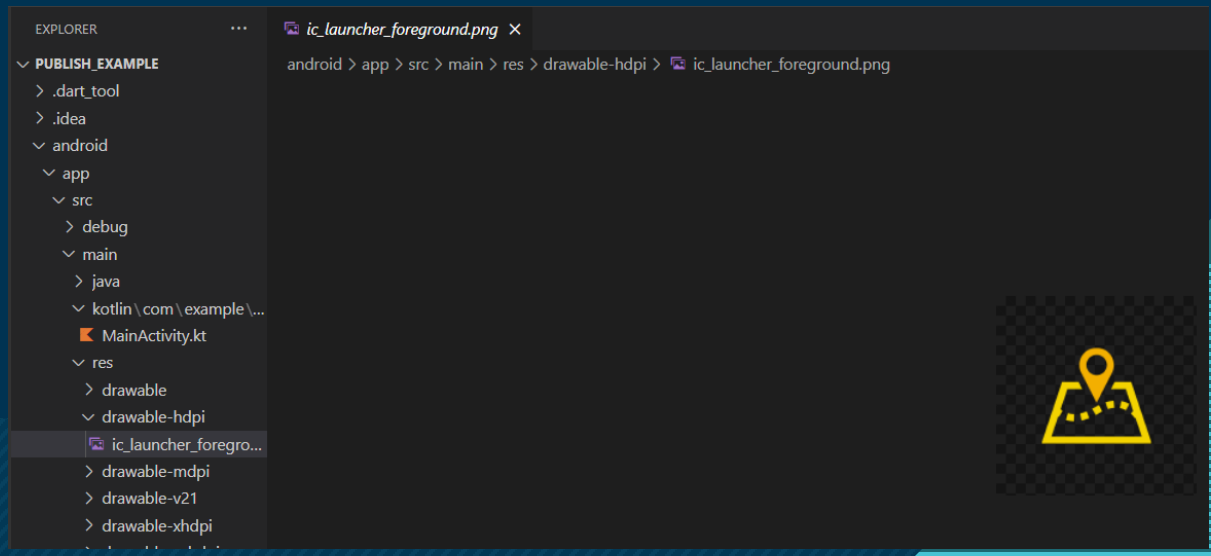




Adding Icons

- flutter_launcher_icons
  - We use this package to automatically generate all the images needed
  - We add it in “dev_dependencies” in pubspec.yaml because we don't want it to be part of the final application
- Image_path: icons path
- Adaptive icon : round icons used in apps
  - Adaptive background : color
  - Adaptive foreground : icon
- Creates icons for each different mobile version the user is using for both Android and iOS
- Run the command that the package has in the Readme file : flutter pub run flutter_launcher_icons:main

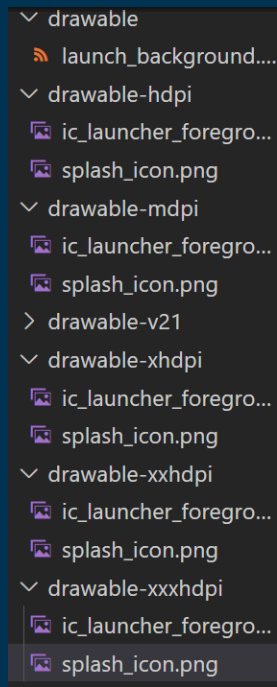




Adding Splash Screen - Android

- Android→app→res→launch_background.xml: loading screen
- 1) In all drawable_hdpi etc. folders we add the icons and splash screen icons such as splash_icon.png
- 2) values→colors.xml add the splash color
- 3) drawable→launch_background.xml and drawable-v21→launch_background.xml add color title add in colors.xml for example @color/splash

1)



2)

android > app > src > main > res > values > colors.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="ic_launcher_background">#191919</color>
    <color name="splash">#191919</color>
</resources>
```

3)

android > app > src > main > res > drawable-v21 > launch_background.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<!-- Modify this file to customize your launch splash screen -->
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
    <!--<item android:drawable="?android:colorBackground" />-->
    <item>
        <color android:color="@color/splash" />
    </item>
    <!-- You can insert your own image assets here -->
    <item>
        <bitmap
            android:gravity="center"
            android:src="@drawable/splash_icon" />
    </item>
</layer-list>
```

Adding SplashScreen - iOS