# User Interface

## Tools

### Dart

Dart is an object-oriented programming language by Google. It is best suited for web-, mobile- and desktop applications. Dart forms the foundation of Flutter.

### Flutter

Flutter is a UI toolkit, for creating native applications for mobile, web and desktop. Flutter uses its own rendering engine for drawing customisable widgets, instead of default widgets of the target device or web browser technology.

Flutter uses Dart for the implementation for its widgets, animations, gestures, etcetera...

### Android Studio

Android Studio is an IDE for the development of Android Apps in addition Android Studio Provides an Android Emulator for testing mobile Applications.

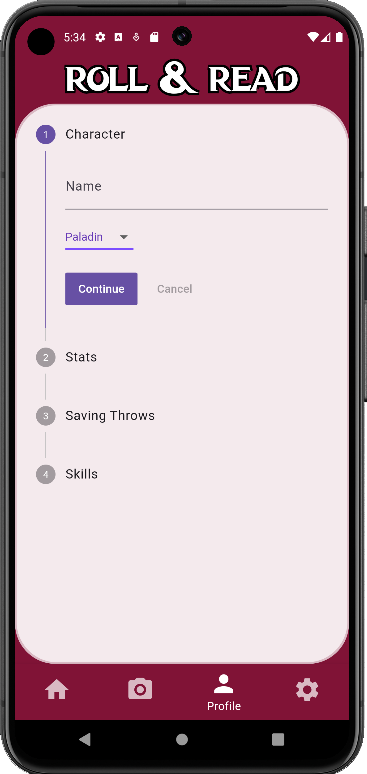
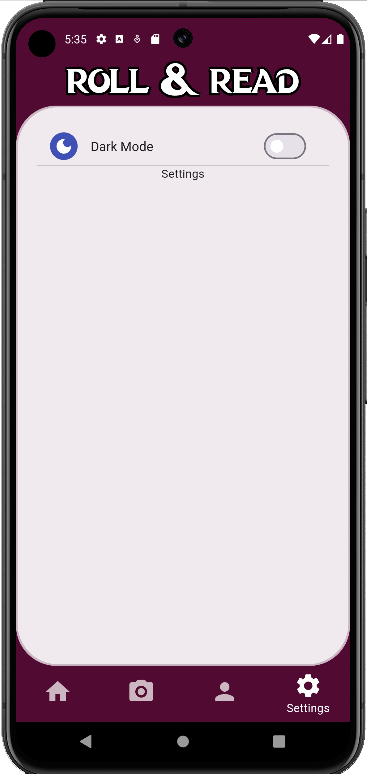
## Design

The user interface consists of four Pages, which can be navigated on a bottom-navigation-bar



Home Page

Scan Page



Settings Page

Profile Page

### Main-Scaffold

The Main-Scaffold consists of an AppBar that holds a centered custom Text Widget textWithOutline.

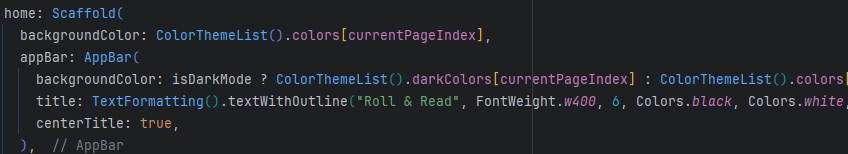


The textWithOutline Widget returns a Stack Widget in order to create the desired Outline effect on a Text, the effect is create by stacking two Text Widgets on top of each other.



Furthermore does the Main-Scaffold uses the Body: argument to draw all the other pages inside a Rounded Container with an outline.

The colours theme used is stored in a custom colour theme list class.  
I used the iterability of the List to change the colour theme of the body.

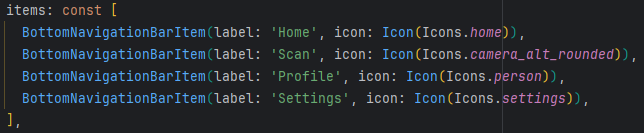


### Bottom-Navigation-Bar

For the purpose of Navigating the App and its different pages a Navigation is needed, I choose for this case the BottomNavigationBar Widget.

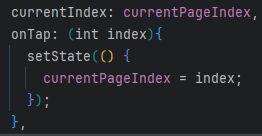
This widget uses the required Argument items, which is a List full of BottomNavigationBarItems.

This Widget uses a label (String) and an Icon.



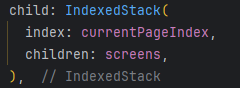
currentIndex: tells the Application which page to load.

The navigation bar, additionally has an argument onTap: which returns an index of the pressed BottomNavigationBarItem



The setState Argument tells the main-app if a Navigation Button was pressed and that the app should reload in order to display the new Page.

The pages themselves are loaded in the Body: of the App, via IndexedStack.



The indexed stack ensures that at all times, all pages are loaded in the Stack, so app values will persist even after navigating to another page.

### Home-Page

The Home-Page of the App consists only of the centered Audio Button and a Background consisting of an Image

### Scan-Page

The Scan-Page will be used to scan a dice and displays the rolled amount.

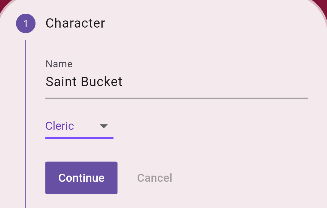
### Profile-Page

The profile Page, can show either a character creator or character display

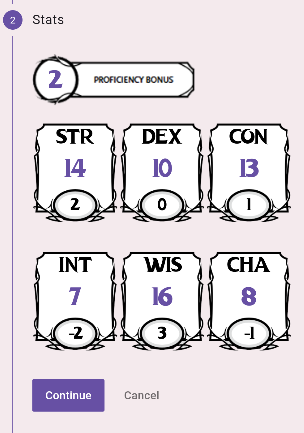
The CharCreator Widget is a Custom Widget which includes a Stepper Widget.

The Stepper Widget similarly to the Bottom-Navigation-Bar uses a List of Steps to display a character creation process. The List consists of four Steps:

**Character**: Where the User can choose a Name and a Class for creation

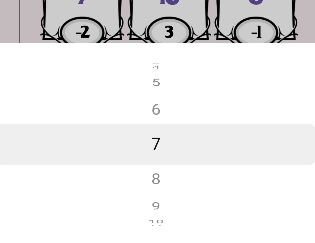


**Stats**: Where the User can add his Stats and the Proficiency Bonus to the Character.



The background Images displayed on Screen are from the official character sheet from Dungeons and Dragons fifth edition, the Images represent the background of the 7 Containers.

In order to choose the stats, a CupertinoButton is used.

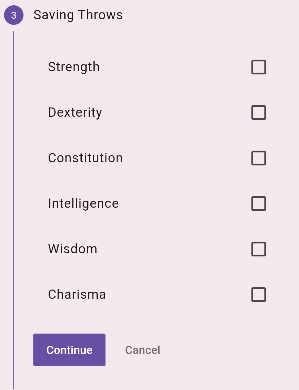


The second number is the Abillity Bonus which is usually calculated by the following formula

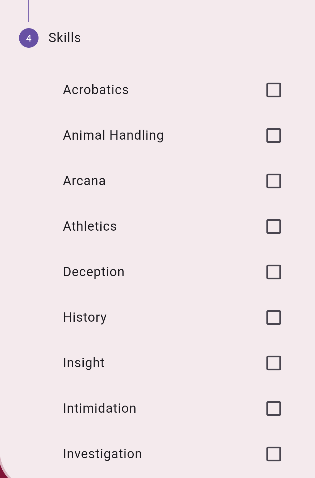
then rounded down.

But the default flutter round() function rounds to the nearest number so in order to calculate the correct values the following is used:

**Saving Throws:** On this Step the user can choose his Saving Throws via checkboxes.



**Skills**: On the last Step the User can choose his character skills similarly to the previous Step.



With the completion of the last Step the character creation is completed and a Boolean flag is set to true in order to load the character display, instead of the creator.

To accomplish this a ValueChangeObserver from the flutter\_settings\_screen plugin is used. With the call of

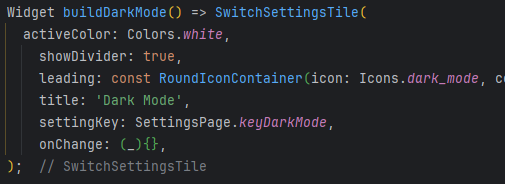


The first value is the key to access the variable, the second value is the value to set the variable and notify true is to notify the App-Builder to reload the Page in order to display the new Page.

### Settings-Page:

The Settings-Page consists of the Dark-Mode Setting.

A SwitchSettingsTile is used to check if the dark mode is activated or not, this Widget from the flutter\_settings\_screen plugin uses the settingsKey argument to set the observed value to the switch value.

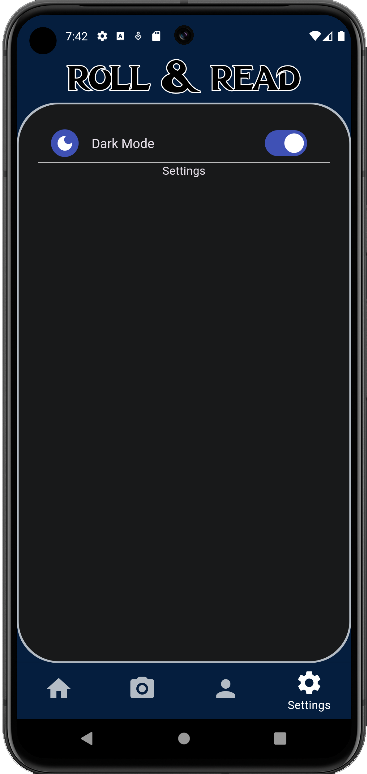


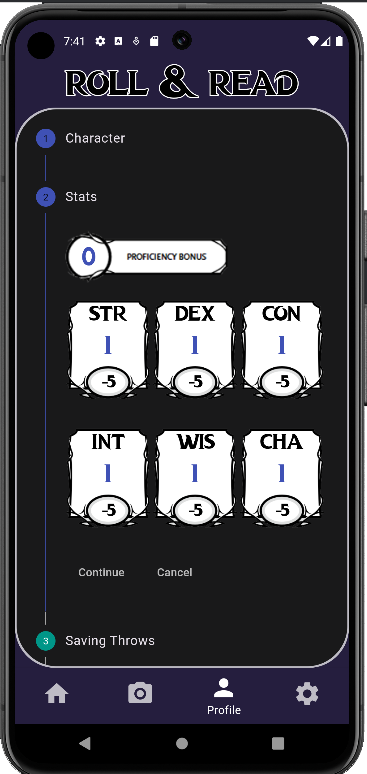
The value of keyDarkMode can now be used to enable the dark-mode setting, in the rest of the app.

For example the AppBar:



### Dark Mode





The dark-mode uses a different Theme for the entire app and a different entry in the ColorThemeList and a different background Image for the Home page.