# PMH

## LAB SESSION 1 Flutter installation & UI design using flutter

To: Mr Shiam Beeharry
By: Kritika Bissessur

## Contents

# 1 Task 2

Task-2

| Using an appropriate online tool create a wireframe for a Flutter App |
|---|
| Provide a documentation with a short description and screenshots for all the steps of the lab. |

## 1.1 Brief Overview

To create a wireframe for a Flutter travel app, the key tasks involve designing the interface for sign-in, login, destination search, booking, payment, messaging, and settings features. These elements should be strategically positioned within the wireframe to ensure a seamless user experience. The wireframe must focus on the logical flow of the app, emphasizing a user-friendly navigation system and the intuitive placement of essential components. This approach will lay the foundation for a visually appealing and highly functional travel app within the Flutter framework.

App usedLucid Chart

## 1.2 Steps

### 1.2.1 Login Page

The login page is the initial interface in the wireframe, serving as the entry point for users to access the travel app. Here, users are prompted to enter their email and password to gain access to their account. Additionally, if users do not have an existing account, a clear and intuitive option for sign-up and registration should be provided. The design should focus on simplicity and usability to ensure a seamless login process for users.

### 1.2.2 Destination Selection Page

Following the login page, the wireframe leads users to the destination selection page. This interface offers users the ability to browse and select their desired travel destination. The design should prioritize showcasing enticing visuals and detailed information about each destination, ultimately enabling users to make informed decisions when selecting their travel location.

### 1.2.3 Origin Selection Page

The wireframe progresses to the origin selection page, providing users with the option to choose their current location or origin, usually by selecting a country or city. The interface should be designed to streamline the process of inputting the user's starting point, ensuring convenience and accuracy in travel planning.

#### 1.2.3.1 Feed or Homepage

The feed or homepage segment of the wireframe acts as a central hub within the app, displaying personalized travel recommendations, popular destinations, and other relevant content. This section serves as a gateway to accessing recent searches, saved destinations, and notifications, ultimately enhancing the user's overall experience and engagement with the app.

#### 1.2.3.2 Booking Page

Upon selecting a destination, users are directed to the booking page, where they can choose their desired travel dates and months. This page offers a user-friendly interface for selecting travel dates and initiating the booking process. Subsequently, users are led to the payment page, where they can securely provide their card details, including the card number, expiry date, and country for payment processing.

#### 1.2.3.3 Payment Page

The payment page serves as the interface for users to enter their payment details. It prioritizes security and ease of use, culminating in a seamless transaction process for users. The design should instill confidence in users' payment interactions, fostering trust and satisfaction in the app's payment processing capabilities.

### 1.2.4 Messages Page

The wireframe includes a dedicated messages page, offering users the ability to communicate and discuss their travel plans with other users or customer support. This section should feature messaging functionalities such as real-time chat and message history, contributing to a collaborative and supportive environment for users within the app.

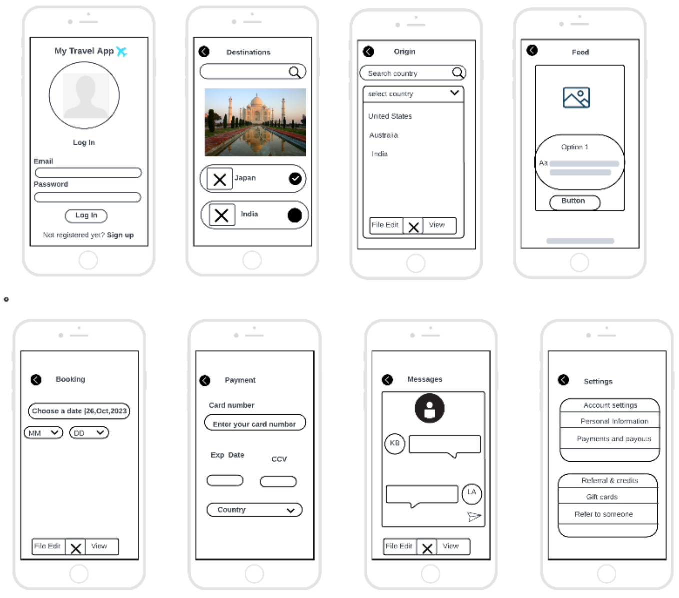
### 1.2.5 Settings Page

The wireframe concludes with the settings page, encompassing a range of features such as account settings, personal information management, payment and payout options, referral credits, and the ability to redeem gift cards. This page provides users with control over their account and preferences within the travel app, enhancing their overall experience and satisfaction with the platform.

Through the meticulous construction of these wireframe segments, the Flutter travel app aims to deliver a user-centric and feature-rich interface, ultimately shaping a compelling and seamless travel booking experience.

## 1.3 Final Output

The wireframe serves as a blueprint for the Flutter travel app, systematically guiding users through each stage of the travel booking process with a focus on user-friendly design and intuitive navigation. Beginning with the login page, it offers a streamlined entry point, seamlessly transitioning users to the destination and origin selection pages to efficiently input their travel preferences. The feed or homepage segment acts as a central hub, presenting tailored travel recommendations and essential user interactions. As users proceed to the booking and payment pages, the interface prioritizes simplicity and security, ensuring a hassle-free booking experience. Additionally, the inclusion of a dedicated messages page and settings page fosters seamless communication and personalized control over account preferences. Overall, the wireframe orchestrates a cohesive and engaging user journey, ensuring a seamless travel booking experience within the Flutter travel app.



## 2 Task 3

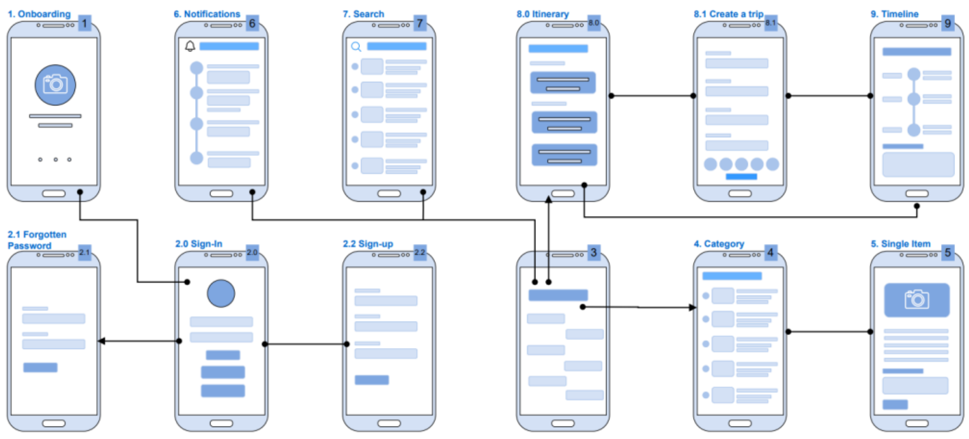
**Task-3**

| |
|---|
| **Using an appropriate online tool create a wireflow for a Flutter App** |
| **Provide a documentation with a short description and screenshots for all the steps of the lab.** |

### 2.1 Brief Overview

I crafted a wireflow for an innovative travel app, seamlessly guiding users through onboarding, authentication, and profile customization. The app features category exploration, detailed item views, and a centralized Notification Center for real-time updates. With efficient search functionality, users can easily discover content. The Itinerary feature facilitates trip planning, while the Timeline offers a visual journey through past engagements. This travel app promises a user-centric experience, combining functionality and aesthetics for a seamless and memorable exploration.

App usedLucid Chart



## 2.2 Steps

### 2.2.1 Onboarding

- This initial step introduces users to the app, guiding them through the setup process. Users provide essential information to personalize their experience, laying the foundation for a tailored journey within the platform.

### 2.2.2 Authentication

#### 2.2.2.1 Forgotten Password

In case users forget their password, this step facilitates recovery by guiding them through a secure process to reset their password and regain access to their account.

#### 2.2.2.2 Sign-up

New users embark on their journey by creating an account, providing necessary details to establish a unique profile within the platform.

#### 2.2.2.3 User Profile

This section empowers users to manage their personal information, preferences, and settings, fostering a sense of ownership and customization over their experience.

### 2.2.3 Category Selection

Users choose from a selection of broad categories, setting the tone for their exploration within the app. This step serves as a pivotal point in tailoring content to individual interests.

### 2.2.4 Single Item View

- Delving into a specific item within the chosen category, users experience a detailed view. This immersive exploration allows users to focus on and engage with content at a granular level.

### 2.2.5 Notification Center

- Acting as a centralized hub, the Notification Center keeps users informed about updates, interactions, and relevant activities within the app, enhancing the overall user experience.

### 2.2.6 Search Functionality

- Empowering users to efficiently explore and find specific content, the search feature provides a streamlined experience for those seeking particular information or items.

### 2.2.7 Itinerary

#### 2.2.7.1 Create a Trip

Users can plan and organize their journeys with ease, utilizing this feature to create itineraries that align with their preferences, ensuring a seamless and organized travel experience.

#### 2.2.7.2 Timeline

The timeline visually represents users' activity and history within the app, offering a chronological overview of their engagement. This feature serves as a dynamic record, allowing users to reflect on their journey and interactions over time.

# 3 Task 4

Task-4

| Using an appropriate online tool create a prototype for a Flutter App |
|---|
| Provide a documentation with a short description and screenshots for all the steps of the lab. |

## 3.1 Brief Overview

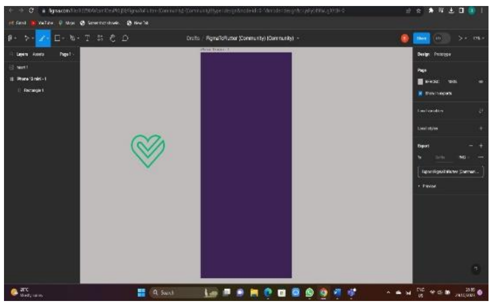
The prototype for the gym equipment shop encompasses essential features such as login and sign-up screens, along with seamless access to a diverse range of gym equipment categories upon successful login. The login and sign-up screens provide users with a secure entry point to the app, fostering a personalized and secure experience. Upon logging in, users are granted access to a comprehensive array of gym equipment categories, facilitating convenient browsing and purchasing. The prototype effectively prioritizes user authentication and intuitive navigation, instilling confidence and ease-of-use within the gym equipment shopping experience.

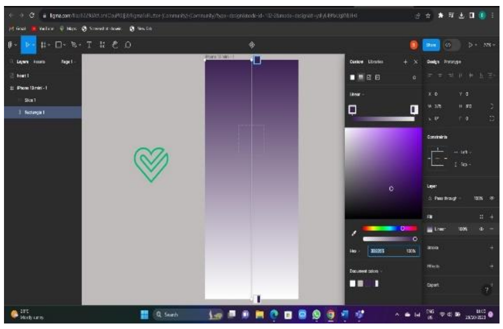
App UsedFigma

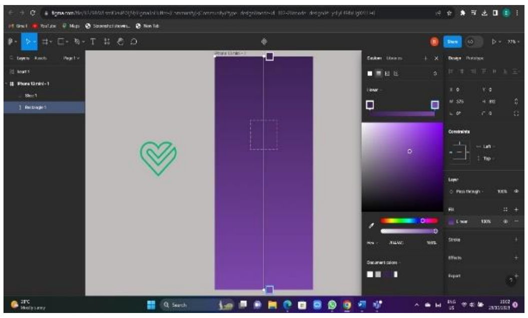
## 3.2 Steps

### 3.2.1 Designing the Screens

To commence the app development process, I initiated by outlining the foundational framework of the main screen. Incorporating a distinct company logo, I meticulously positioned it within the interface to establish a strong brand presence. The logo, prominently featuring the phrase "Let’s go to the gym," was meticulously integrated to resonate with the app's fitness-oriented theme. This deliberate design element aims to captivate users and convey the app's core essence



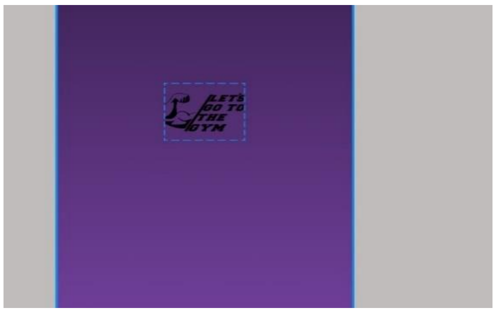




### 3.2.1.1 Logo

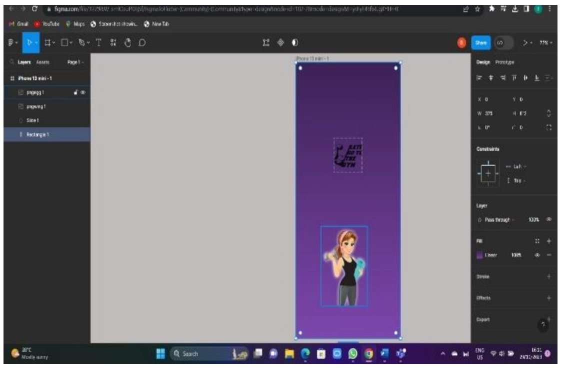
As you can see,I have written 'Let's go to the gym'

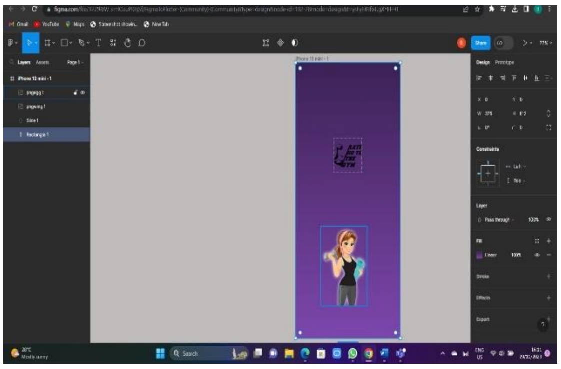
A pivotal component of the app's visual identity, the inserted logo serves as a compelling representation of the brand. Featuring the empowering tagline "Let's go to the gym," the logo stands as a charismatic focal point, exuding motivation and allure. This carefully curated visual element endeavors to resonate with fitness enthusiasts, instilling a sense of excitement and anticipation for engaging with the app's functionalities.

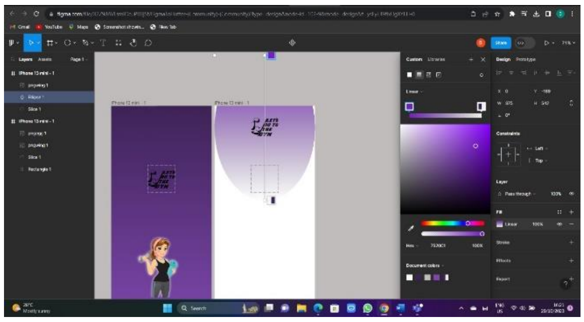


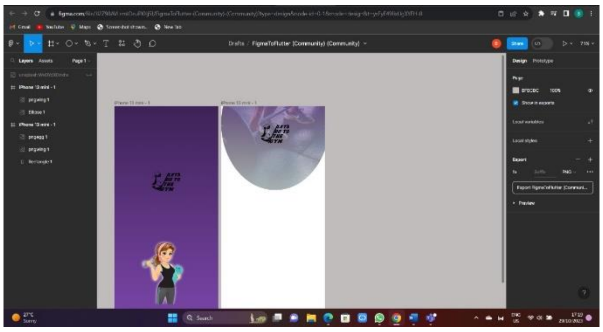
### 3.2.1.2 Decoration

In an effort to elevate the aesthetic appeal of the main screen, deliberate efforts were made to embellish the interface with visually captivating decorations. This involved thoughtfully incorporating relevant and engaging imagery to foster an immersive and visually enriching user experience. The integration of these decorative elements was purposefully executed to enhance user engagement and create an inviting and dynamic interface.

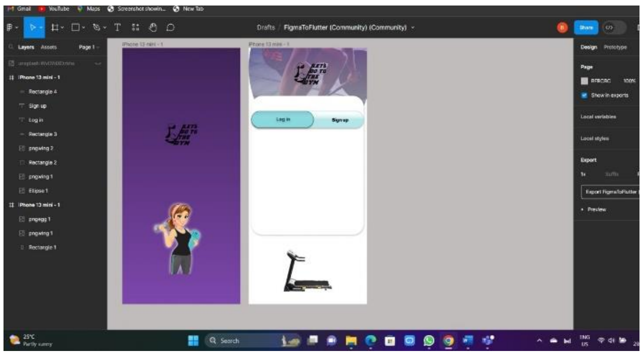






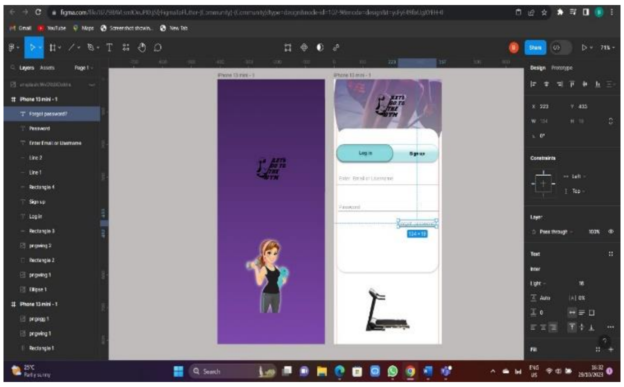


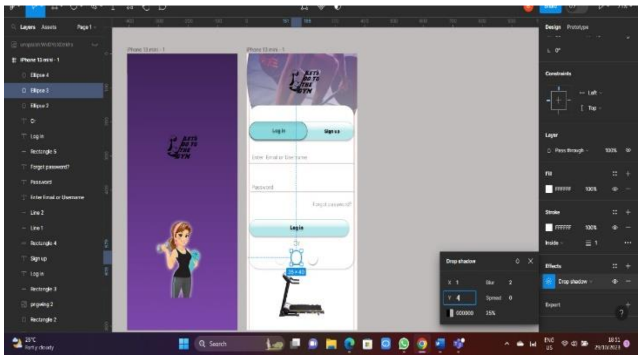
Here,I start by drawing a rectangle in which the log in and sign up form will be implemented

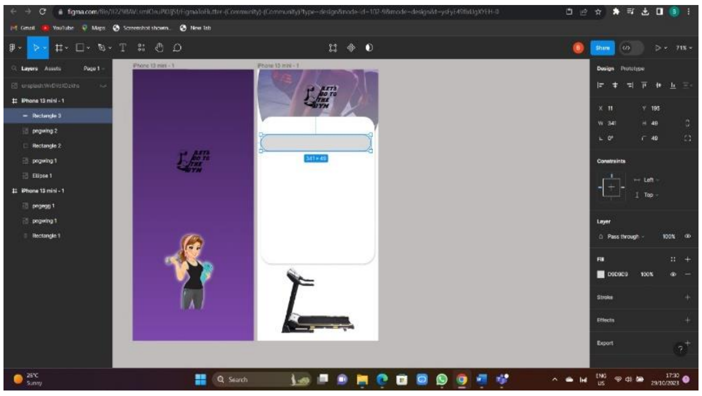


### 3.2.2 User Authentication Implementation

Implementing user authentication involves incorporating secure mechanisms for users to verify their identity and access relevant features within an application or platform. This process typically includes creating login and signup interfaces

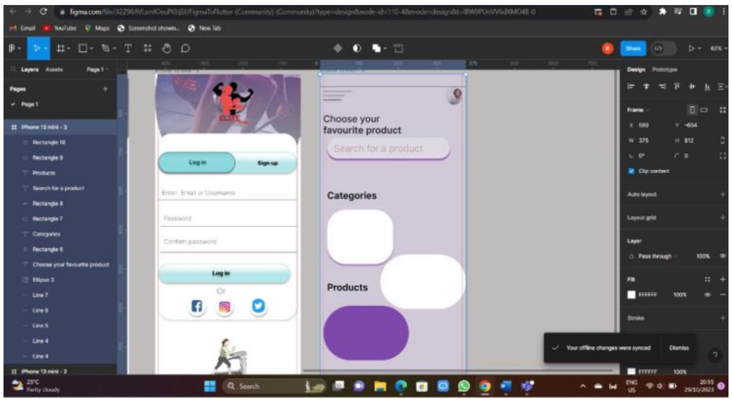


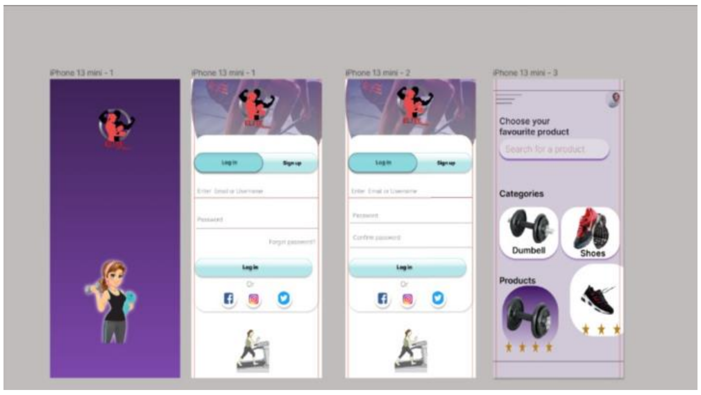




### 3.2.3 Creating Product Categories

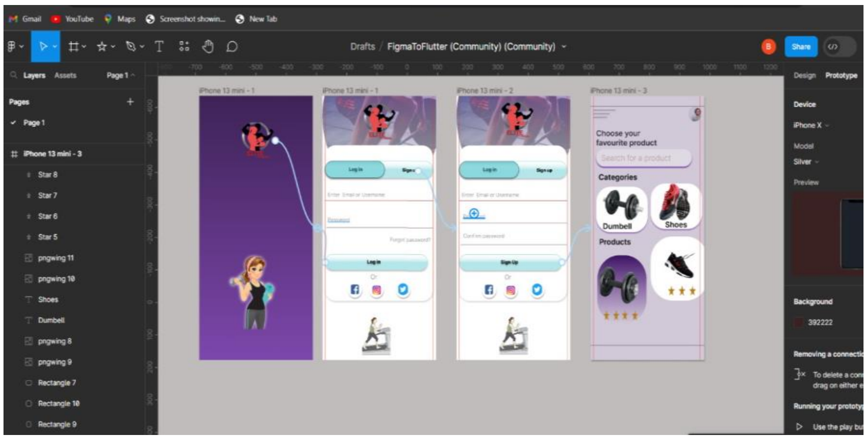
A fundamental aspect of the app's interface involves meticulously designing and configuring the segment dedicated to product categories. This section is meticulously crafted to enable users to seamlessly explore and select from a diverse array of gym equipment categories. Through intuitive navigation and visually compelling categorization, users are empowered to effortlessly browse and engage with various gym equipment options, enhancing their overall shopping experience within the app.



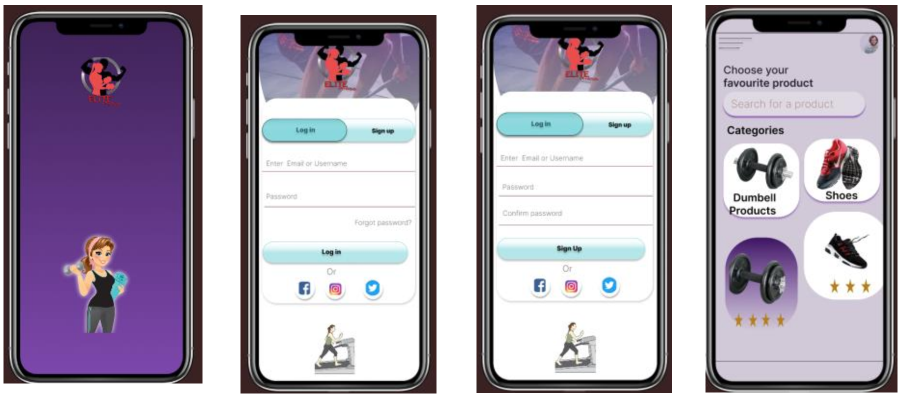


### 3.2.4 Prototype

My prototype app for the gym equipment shop is very user-friendly and intuitive. Upon clicking the logo icon, users are directed to the login page with options for email, password, and username. For new users or those not yet registered, there is an accessible signup process as well as an option for password recovery. After successful login or signup, users are seamlessly navigated to the product categories page where they can easily select the desired gym equipment categories. This streamlined process ensures a smooth and efficient user experience, from initial access to navigating the diverse product offerings.



## 3.3 Final Output



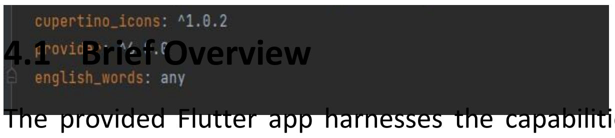
Below you can find the link in order to explore how this prototype works

https//www.figma.com/proto/JI2Z98AVLsmIOeuPI0JjSf/FigmaToFlutter-(Community)-(Community)?type=design&node-id=102-2&t=7UksfXn49Qj0cARP-1&scaling=scale-down&page-id=0%3A1

# 4 Task 5

Task-5

| Create your first Flutter App in VSCode or Android Studio |
|---|
| Provide a documentation with a short description and screenshots for all the steps of the lab. |



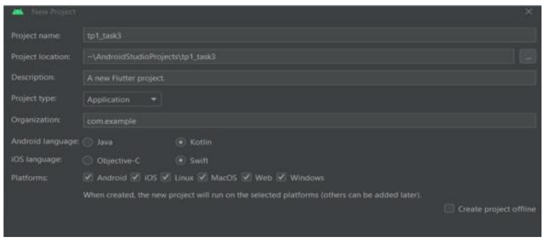
```
cupertino_icons: ^1.0.2
provider: ^6.0
english_words: any
```

## 4.1 Brief Overview

The provided Flutter app harnesses the capabilities of the english_words package to dynamically generate random word pairs, offering users a diverse and engaging experience. Leveraging the power of the provider package for efficient state management, the app ensures seamless interactions and smooth data handling. Furthermore, by integrating the flutter/material.dart library, the app capitalizes on a comprehensive array of UI components to craft visually compelling and user-friendly interfaces. Overall, this combination of packages and libraries empowers the app to deliver a rich and interactive user experience, characterized by dynamic content generation and robust state management.

## 4.2 Steps

1. We need to create a new Flutter project and replace the contents of the main.dart file with the provided code.

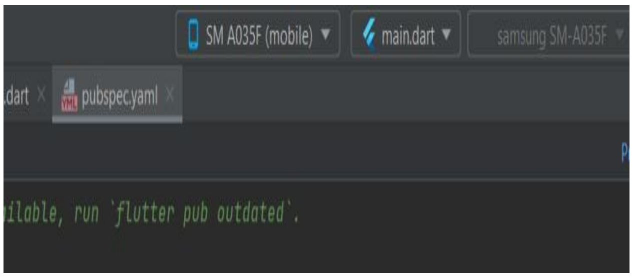


Project name: tp1_task3
Project location: ~\AndroidStudioProjects\tp1_task3
Description: A new Flutter project.
Project type: Application
Organization: com.example
Android language: Java [✓ Kotlin]
iOS language: Objective-C [✓ Swift]
Platforms: ✓ Android ✓ iOS ✓ Linux ✓ MacOS ✓ Web ✓ Windows
When created, the new project will run on the selected platforms (others can be added later).
☐ Create project offline

2. We need to add the required dependencies in the pubspec.yaml file. In this case, the required dependencies are english_words, provider, and flutter/material.dart.

3. Run the app using a simulator or a physical device connected to the computer. The app should open in the selected device.

**Method Used Physical Device Or Chrome**

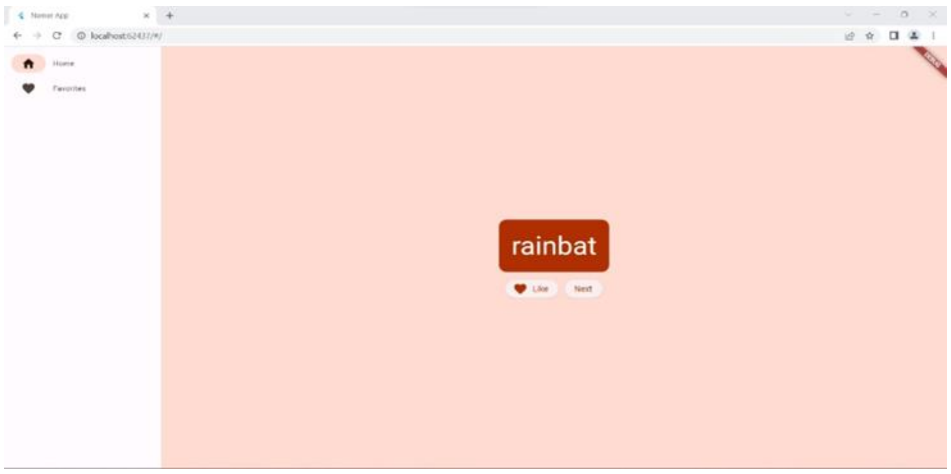


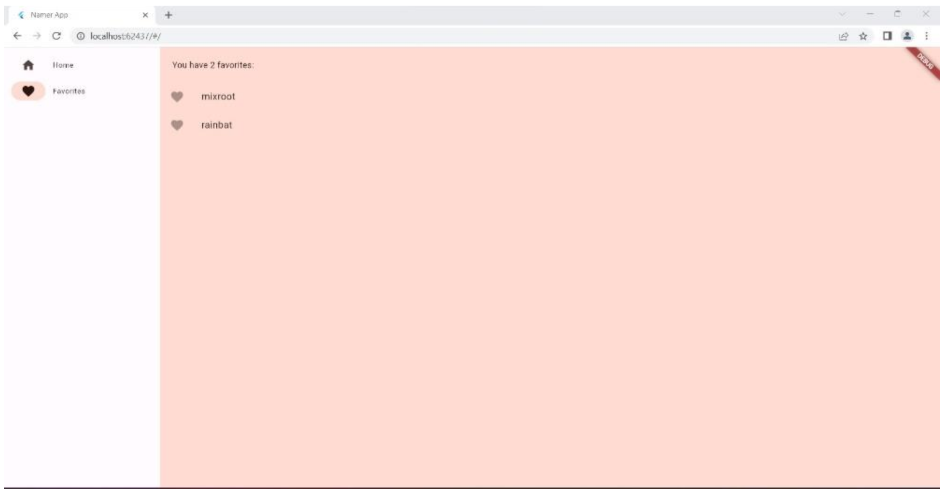
```
ilable, run `flutter pub outdated`.
```

## 4.3 How the app works?

- The app has two main screens the generator page and the favorites page. The generator page generates a random word pair and allows the user to mark it as a favorite or generate a new pair. The favorites page shows a list of all the marked favorite pairs.
- To switch between the generator page and the favorites page, we use the navigation rail at the left side of the screen. The generator page is the default page that opens when the app is launched.
- To mark a word pair as a favorite, click on the heart icon labeled "Like" on the generator page. The icon will change to a filled heart if the word pair is marked as a favorite.
- To generate a new word pair, we need to click on the "Next" button on the generator page.
- To view all the marked favorite pairs, go to the favorites page using the navigation rail.
- If there are no marked favorite pairs, a message "No favorites yet." will be displayed on the favorites page.

## 4.4 Final Output





You have 2 favorites:

mixroot

rainbat

In summary, the provided code is a simple Flutter app that uses provider for state management to generate random word pairs and allows the user to mark their favorite pairs. The app uses the english_words package to generate random word pairs and flutter/material.dart to build UI components.

# 5 Task 6

**Task-6**

| **Visualizing dynamic color in your app - Codelab** |
|---|
| https://codelabs.developers.google.com/visualize-dynamic-color#0<br><br>**Provide a documentation with a short description and screenshots for all the steps of the lab.** |

## 5.1 Brief Overview

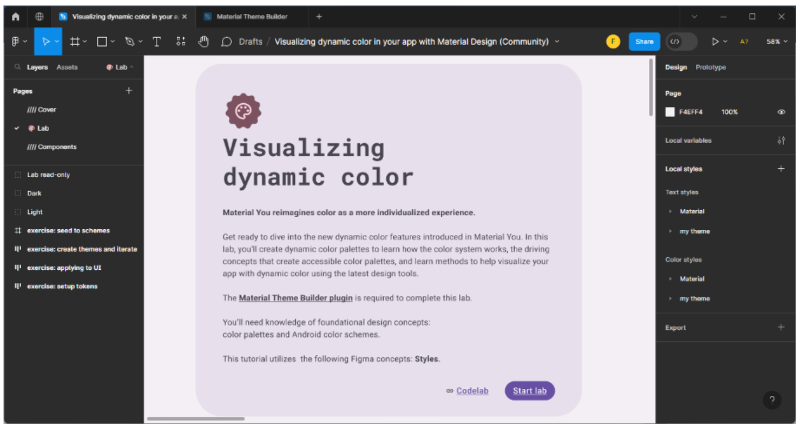
During the codlab session, we explored the new dynamic color features introduced with Material You. The focus was on creating dynamic color palettes and understanding the underlying concepts that contribute to accessible color schemes. Additionally, we learned how to apply user-generated colors to our app and utilized the latest design tools to visualize our app with dynamic colors.

In this session, we covered the following topics

- Understanding the updates in Material Design color
- Applying user-generated colors to our app
- Utilizing tools such as Figma and the Material Theme Builder plugin
- Building upon foundational design concepts, including color palettes and Android color schemes

To participate in the codlab, we needed the following

- A Figma Account
- The Figma Dynamic color Designlab file
- The Figma plugin Material Theme Builder



Visualizing dynamic color

**Material You reimagines color as a more individualized experience.**

Get ready to dive into the new dynamic color features introduced in Material You. In this lab, you'll create dynamic color palettes to learn how the color system works, the driving concepts that create accessible color palettes, and learn methods to help visualize your app with dynamic color using the latest design tools.

The **Material Theme Builder plugin** is required to complete this lab.

You'll need knowledge of foundational design concepts:
color palettes and Android color schemes.

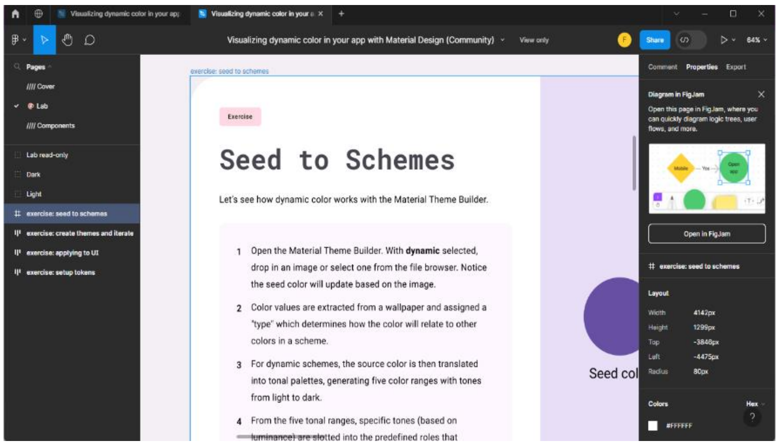
This tutorial utilizes  the following Figma concepts: **Styles**.

## 5.2 Steps

After the introduction, we accessed the 'Visualizing dynamic color in your app with Material Design' Figma file. This file served as a visual resource for understanding and implementing dynamic color concepts in our app. We explored the concept of dynamic color, which allows colors to adapt and change based on various factors. Additionally, we delved into color concepts such as luminance, which refers to the brightness of a color, and tonal palettes, which provide a range of colors within a specific hue. These discussions helped us gain a deeper understanding of color principles and how they can be applied effectively in our app design.

### 5.2.1 Extracting colours

Next, the codlab introduced us to material theme design. We learned about the process of extracting colors and creating color schemes using the Material Theme Builder.



Visualizing dynamic color in your app with Material Design (Community)

Exercise

Seed to Schemes

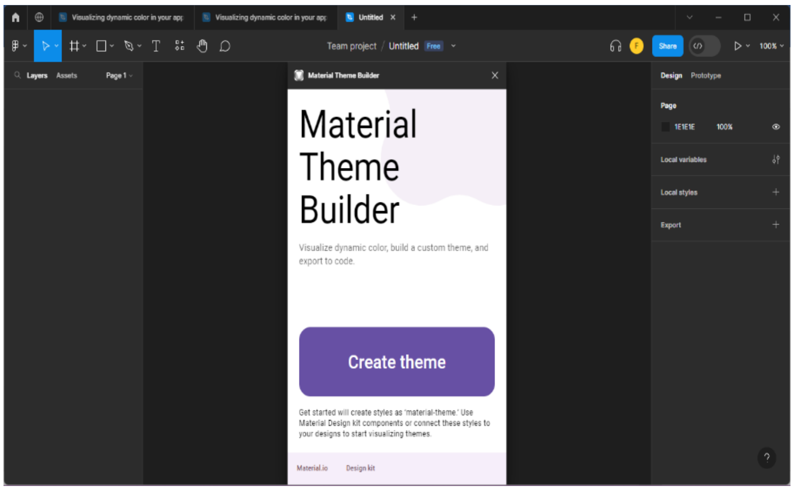
Let's see how dynamic color works with the Material Theme Builder.

1 Open the Material Theme Builder. With **dynamic** selected, drop in an image or select one from the file browser. Notice the seed color will update based on the image.

2 Color values are extracted from a wallpaper and assigned a "type" which determines how the color will relate to other colors in a scheme.

3 For dynamic schemes, the source color is then translated into tonal palettes, generating five color ranges with tones from light to dark.

4 From the five tonal ranges, specific tones (based on luminance) are slotted into the predefined roles that

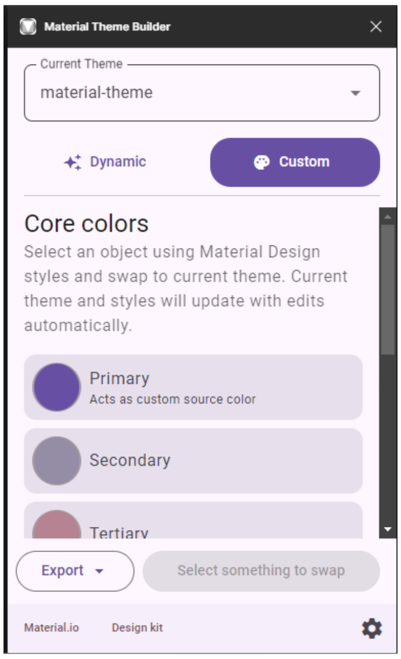


Material Theme Builder

Visualize dynamic color, build a custom theme, and export to code.

Create theme

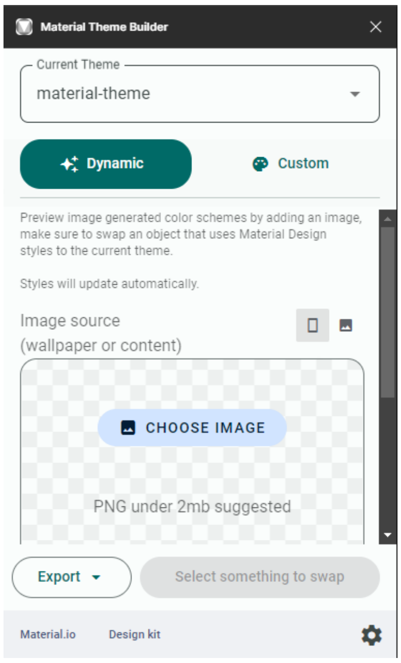
Get started will create styles as 'material-theme.' Use Material Design kit components or connect these styles to your designs to start visualizing themes.



Core colors

Select an object using Material Design styles and swap to current theme. Current theme and styles will update with edits automatically.

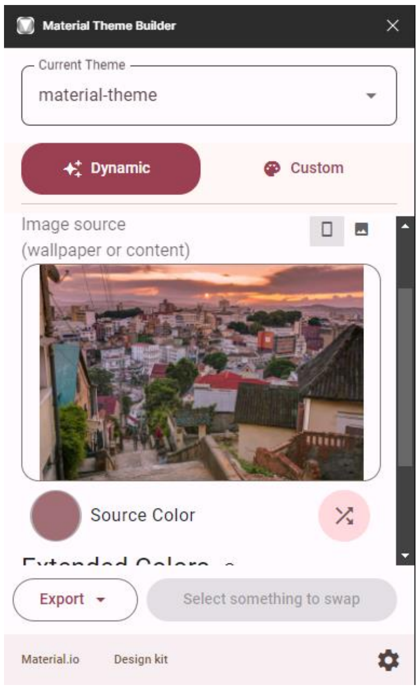
In this step, we utilized the Material Theme Builder tool. With the dynamic color option selected, we either dropped in an image or selected one from the file browser. We observed that the seed color automatically updated based on the image chosen.



Preview image generated color schemes by adding an image, make sure to swap an object that uses Material Design styles to the current theme.

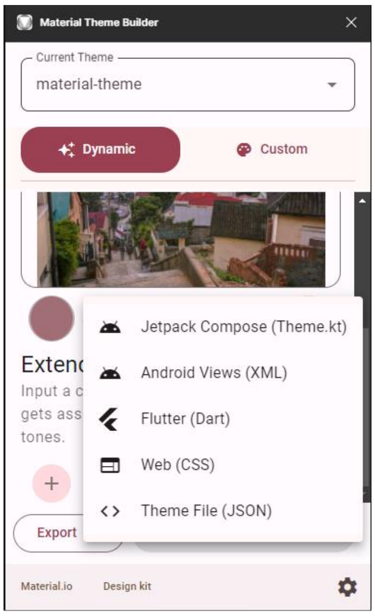
Styles will update automatically.

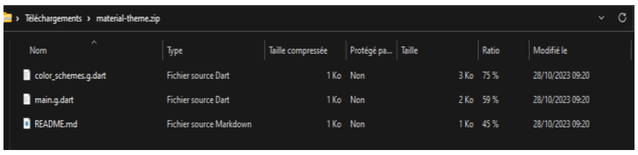
During this process, color values were extracted from the wallpaper or image, and each color was assigned a "type" that determined its relationship with other colors in the scheme. The "key colors" on the right-hand side of the interface were updated to reflect these extracted color values.



By exploring this step, we gained insights into how dynamic colors can be derived from images and how they are categorized to create harmonious color schemes within the Material Theme Builder.

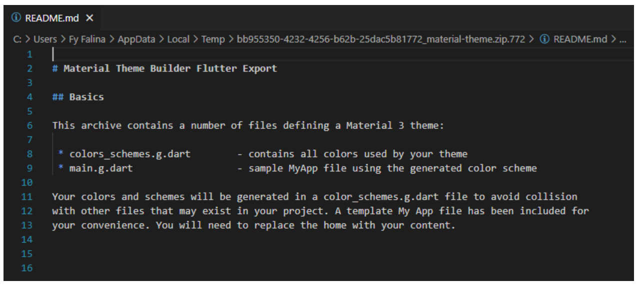
Furthermore, we discovered that the Material Theme Builder offers the functionality to directly download the generated color scheme as a Dart file. This feature enables us to obtain a ready-to-use theme for our apps seamlessly. By downloading the color scheme as a Dart file, we can easily integrate it into our Flutter projects, saving time and effort in manually configuring the theme colors. This capability enhances the efficiency and convenience of implementing consistent and visually appealing themes across our applications.





› Téléchargements › material-theme.zip

| Nom | Type | Taille compressée | Protégé pa... | Taille | Ratio | Modifié le |
|---|---|---|---|---|---|---|
| color_schemes.g.dart | Fichier source Dart | 1 Ko | Non | 3 Ko | 75 % | 28/10/2023 09:20 |
| main.g.dart | Fichier source Dart | 1 Ko | Non | 2 Ko | 59 % | 28/10/2023 09:20 |
| README.md | Fichier source Markdown | 1 Ko | Non | 1 Ko | 45 % | 28/10/2023 09:20 |



README.md ×

C: > Users > Fy Falina > AppData > Local > Temp > bb955350-4232-4256-b62b-25dac5b81772_material-theme.zip.772 > README.md > ...

```

# Material Theme Builder Flutter Export

## Basics

This archive contains a number of files defining a Material 3 theme:

 * colors_schemes.g.dart        - contains all colors used by your theme
 * main.g.dart                  - sample MyApp file using the generated color scheme

Your colors and schemes will be generated in a color_schemes.g.dart file to avoid collision
with other files that may exist in your project. A template My App file has been included for
your convenience. You will need to replace the home with your content.



```

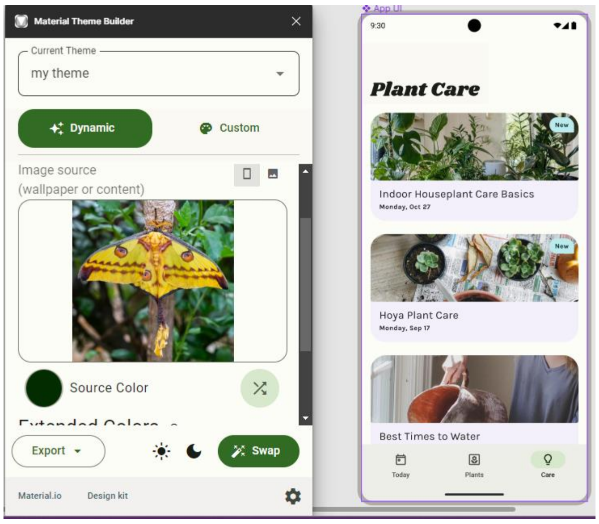
### 5.2.2 Themes and tokens

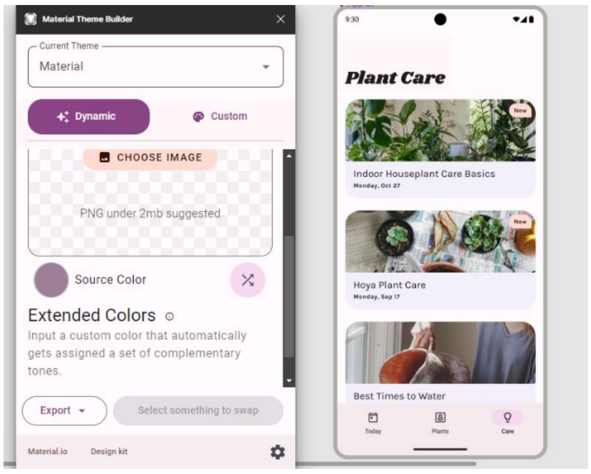
Afterward, we proceeded to set up and utilize tokens in our designs. Design tokens play a crucial role in achieving flexibility and consistency across a product. They allow designers to assign an element's color role in a user interface (UI) rather than a specific color value. Tokens act as a connection between an element's assigned role and the chosen color value for that role. By designing based on color roles rather than specific colors, we embrace a more fundamental approach, especially with the introduction of dynamic color.

Themes encompass Material Design tokens for both color and typography, providing a single source of truth to represent the baseline design, including user-generated palettes and custom values.

In Figma, these tokens are generated as styles by the plugin. This means that if we utilize the generated styles, we will be employing the Material Design tokens.

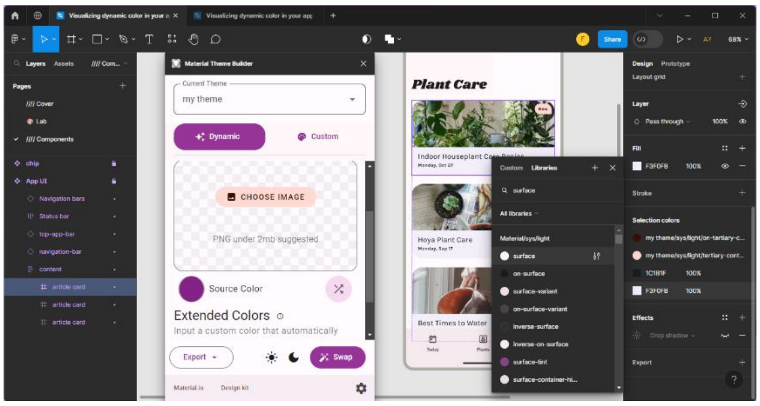
Colors in a tonal palette are mapped to either a light or dark scheme through design tokens. The mapping system assigns a specific tone to each element within a component.



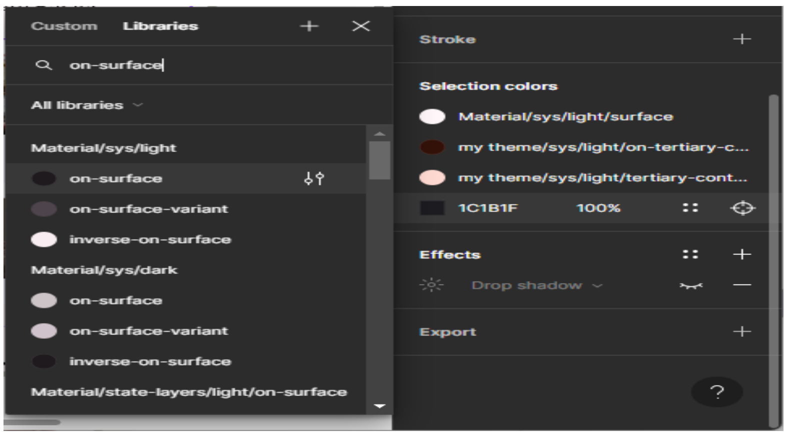


To set up the tokens, we needed to apply them to our designs. We accomplished this by selecting the frame of the layout and using the "swap" function to apply all the tokens (Figma styles) on the right side. As a result, we observed the style prefix updating in the selection colors. This step ensured that our layout utilized the designated theme with dynamic color applied through the tokens.

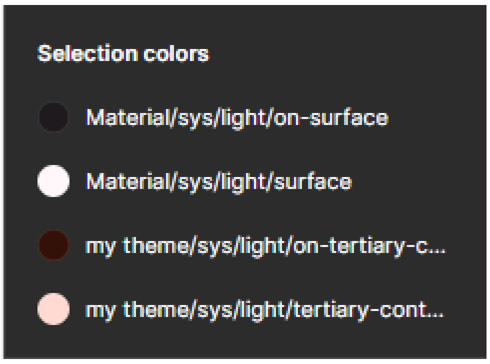
Moving on to applying the tokens to the user interface (UI), we encountered layouts that were initially created using the Material Design Kit, which already utilized Material Design tokens. However, there were a few custom elements that were not mapped to tokens.



To address this, we selected the article cards and adjusted their fill. By clicking on the style icon (depicted as four dots), we set the fill style to "material-theme/surface." Alternatively, we could search for "surface" to locate the appropriate style.

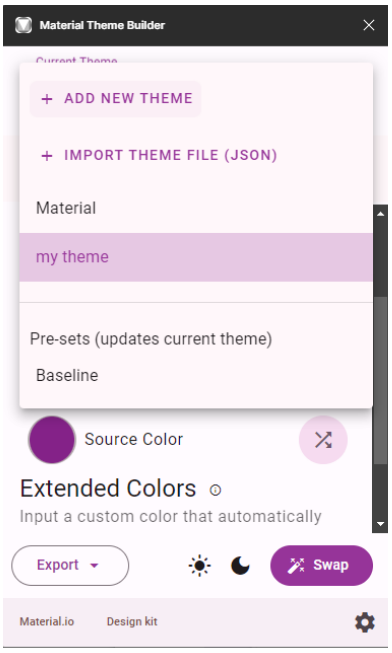


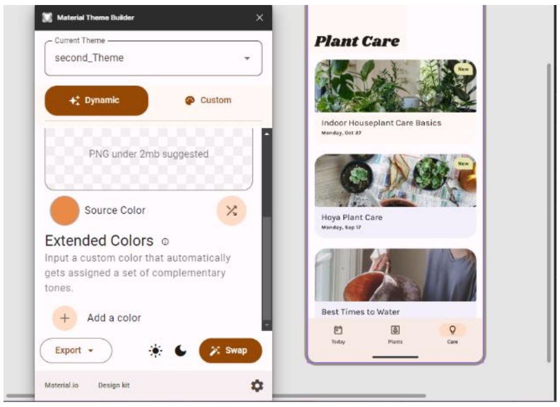
Next, we proceeded to set the type in the cards. Similar to the previous step, we selected the type elements within the cards and set them to "on-surface." Additionally, we adjusted the checkboxes and set them to "primary" to align with the desired token mappings.


Selection colors
Material/sys/light/on-surface
Material/sys/light/surface
my theme/sys/light/on-tertiary-c...
my theme/sys/light/tertiary-cont...

By following these steps, we ensured that the UI elements, including the article cards' fill, type, and checkboxes, were appropriately mapped to the corresponding Material Design tokens. This helped maintain consistency and coherence throughout the design.

Utilizing Material Design, we can create multiple themes simultaneously and apply them to different duplicates of our design. This allows us to view and compare each theme side by side, facilitating the process of theme selection and evaluation.


Material Theme Builder
Current Theme
+ ADD NEW THEME
+ IMPORT THEME FILE (JSON)
Material
my theme
Pre-sets (updates current theme)
Baseline
Source Color
Extended Colors
Input a custom color that automatically
Export
Swap
Material.io
Design kit


Material Theme Builder
Current Theme
second_Theme
Dynamic
Custom
PNG under 2mb suggested
Source Color
Extended Colors
Input a custom color that automatically gets assigned a set of complementary tones.
Add a color
Export
Swap
Material.io
Design kit
Plant Care
Indoor Houseplant Care Basics
Hoya Plant Care
Best Times to Water
Today
Plants
Care

App UI
App UI

# 6 Task 7

**Task-7**

| **Create your first App using MIT AppInventor** |
|---|
| https://appinventor.mit.edu/ <br><br> **Provide a documentation with a short description and screenshots for all the steps of the lab.** |

## 6.1 Brief Overview

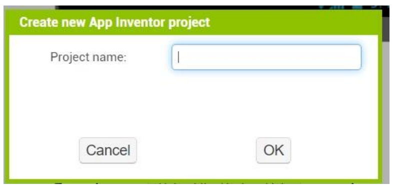
The task at hand involves implementing a program flow that enables a user to engage in a conversational interaction with a chatbot through a graphical interface. The program responds dynamically to user input and chatbot responses, utilizing speech recognition and synthesis functionalities to enhance the interactive experience. In summary, the objective is to create an interactive system whereby users can initiate and engage in chatbot conversations while incorporating speech-based interactions through MIT App Inventor or a similar platform.

App UsedMIT App Inventor

## 6.2 Steps

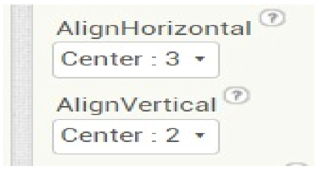
### 6.2.1 Creating a project

I first start by creating my project, that is chatbot


Create new App Inventor project
Project name:

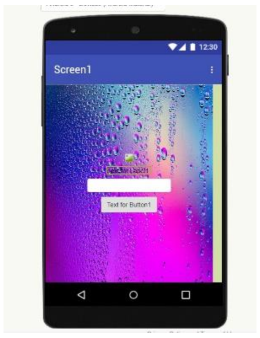
### 6.2.2 Screen background

I then choose a proper background for my main screen and sets the AlignVertical and AlignHorizontal with numbers 3 and 2


AlignHorizontal
Center : 3
AlignVertical
Center : 2

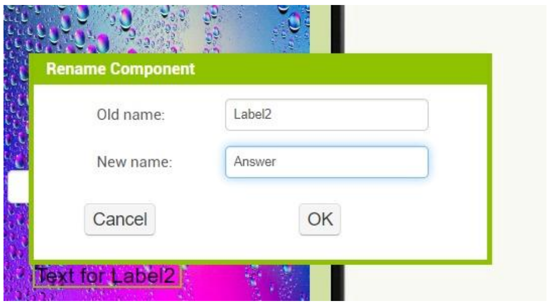
### 6.2.3 Labels

I chose to have 2 labels;one to type my question and the other for the answer



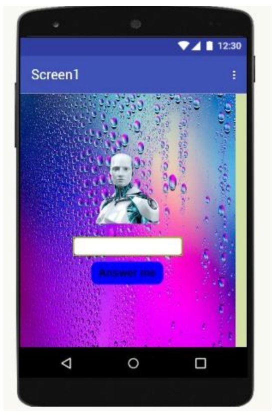
### 6.2.4 Answer button

I will then rename the label 2 button to answer



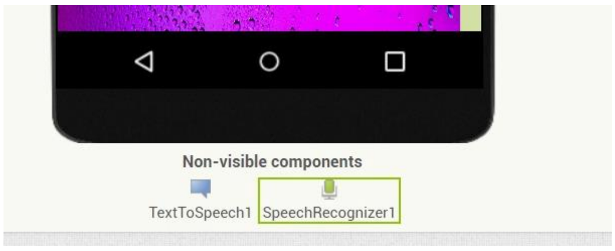
### 6.2.5 Interface

As you can see,I have uploaded a robot picture and I have modified the answer me button in order to look more attractive



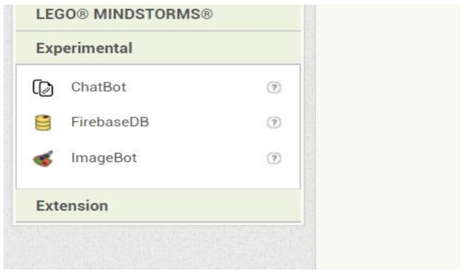
### 6.2.6 Speech recognition

I also integrated speech recognizer and texttospeech since I wanted to integrate vocal in my app



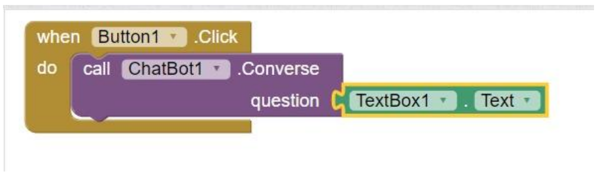
### 6.2.7 ChatBot

In order for the app to answer my questions correctly,it is important to add chatbot,which I discovered from the ChatBot



### 6.2.8 Initiating Chatbot Interaction with User Input

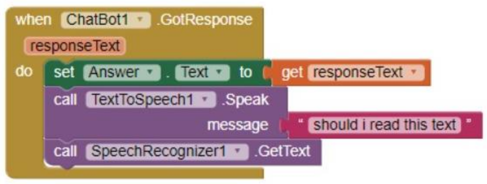
The provided code snippet is a sequence of actions related to a chatbot interaction. It involves triggering a chatbot conversation upon clicking the button ("Button1.Click"), initiating the conversation by calling the chatbot functionality ("ChatBot1.Converse"), and potentially using the text from a text box as an input or question for the chatbot ("TextBox1.Text"). The code outlines a user interface interaction that prompts the chatbot to engage with the user based on the input provided in the text box.



```
when Button1 .Click
do  call ChatBot1 .Converse
                   question  TextBox1 . Text
```

### 6.2.9 Improving user engagement and interaction using voice-based features

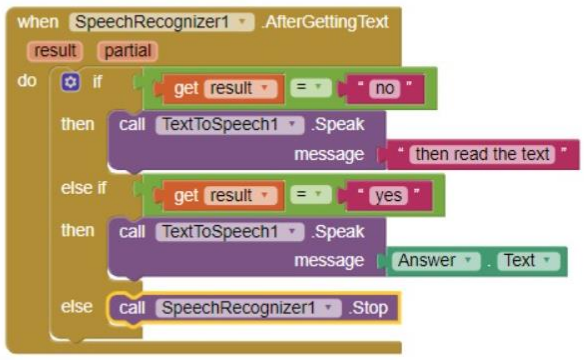
It defines a condition or event related to receiving a response from the chatbot ("ChatBot1.GotResponse") and then setting the answer text to be spoken using a text-to-speech functionality ("set AnswerText to call TextToSpeech1.Speak message get responseText") followed by a prompt for potentially asking if a certain text should be read ("should I read this text") and subsequently using a speech recognizer to potentially capture a response ("call SpeechRecognizer1.GetText").



```
when ChatBot1 .GotResponse
  responseText
do  set Answer . Text to  get responseText
    call TextToSpeech1 .Speak
                  message  " should i read this text "
    call SpeechRecognizer1 .GetText
```

### 6.2.10 Interactive Speech-Enabled System

In the provided code, upon receiving a partial result from the SpeechRecognizer1 component, the program is structured to take specific actions based on the recognized input. If the received result is "no," it will activate the TextToSpeech1 component to orally deliver a message, which is then followed by reading the text aloud. Conversely, if the recognized result is "yes," the program would proceed to utilize the TextToSpeech1 component to audibly convey a message and also set the "Answer" text. Additionally, in the event of an unidentifiable response, the program is designed to initiate the stop action for the SpeechRecognizer1 component. This delineates a dynamic sequence of actions tailored to the recognized input, shaping a responsive and interactive system.



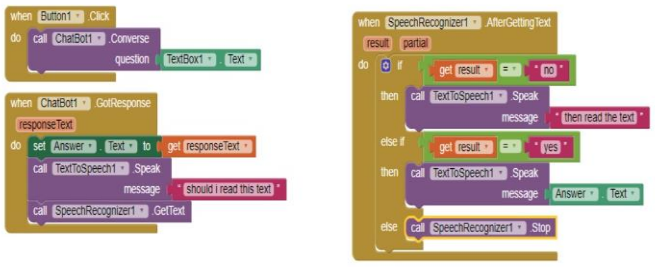
```
when SpeechRecognizer1 .AfterGettingText
  result  partial
do  if      get result = " no "
    then    call TextToSpeech1 .Speak
                         message  " then read the text "
    else if get result = " yes "
    then    call TextToSpeech1 .Speak
                         message  Answer . Text
    else    call SpeechRecognizer1 .Stop
```

### 6.2.11 Enhancing Chatbot Interactions with Dynamic Speech Control in MIT App Inventor

In this program flow, the Button1 Click Event initiates a dialogue with ChatBot1 by utilizing the inquiry provided in TextBox1.Text as its input. Subsequently, when ChatBot1 responds, the program captures and stores the response in the Answer Text before using the TextToSpeech1 component to audibly convey the received message.

Furthermore, the SpeechRecognizer1 AfterGettingText Event intricately handles potential recognition scenarios. If the recognition result is partial and the text obtained is "no," the program orchestrates the TextToSpeech1 component to verbalize the outcome and proceed to read the text aloud. Conversely, if the recognized result is "yes," the program prompts the TextToSpeech1 component to inquire "should I read this text." Lastly, in the absence of a discernible response, the program takes measures to terminate the SpeechRecognizer1 component.

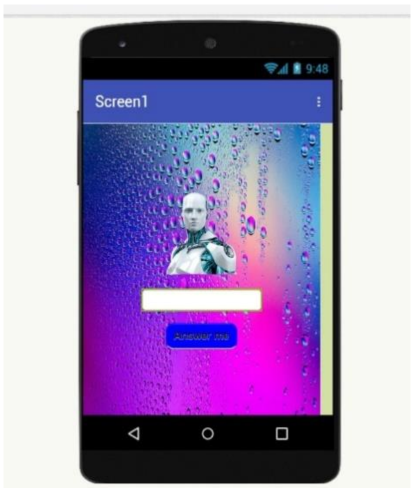
This comprehensive interaction management encompasses the process of initiating, handling, and responding to chatbot interactions, while effectively incorporating speech synthesis and recognition functionalities to enhance user engagement in the MIT App Inventor environment.

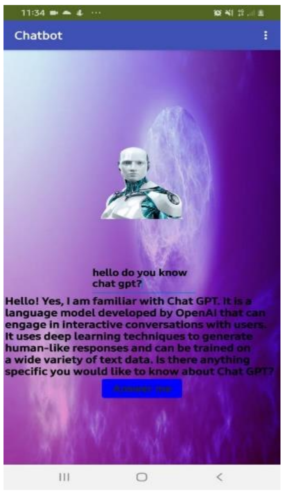


```
when Button1 .Click
do  call ChatBot1 .Converse
                 question  TextBox1 . Text

when ChatBot1 .GotResponse
 responseText
do  set Answer . Text to  get responseText
    call TextToSpeech1 .Speak
                 message  " should i read this text "
    call SpeechRecognizer1 .GetText

when SpeechRecognizer1 .AfterGettingText
 result  partial
do  if  get result = " no "
    then  call TextToSpeech1 .Speak
                       message  " then read the text "
    else if  get result = " yes "
    then  call TextToSpeech1 .Speak
                       message  Answer . Text
    else  call SpeechRecognizer1 .Stop
```

## 6.3 Final Output

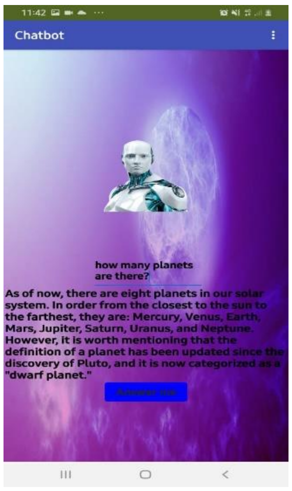
In order to validate the functionality of the app, I downloaded MIT App Inventor on my phone and subsequently utilized the QR code scanning feature to access the app. I systematically engaged with the chatbot by posing various inquiries, thereby assessing the responsiveness and effectiveness of the vocal capabilities. This meticulous process served as a comprehensive validation method, ensuring the proper functioning of the app across diverse operational aspects.





hello do you know chat gpt?

Hello! Yes, I am familiar with Chat GPT. It is a language model developed by OpenAI that can engage in interactive conversations with users. It uses deep learning techniques to generate human-like responses and can be trained on a wide variety of text data. Is there anything specific you would like to know about Chat GPT?



how many planets are there?

As of now, there are eight planets in our solar system. In order from the closest to the sun to the farthest, they are: Mercury, Venus, Earth, Mars, Jupiter, Saturn, Uranus, and Neptune. However, it is worth mentioning that the definition of a planet has been updated since the discovery of Pluto, and it is now categorized as a "dwarf planet."