FLUTTER APP PROJECT DIARY

Contents

Write Your First Flutter App, Part 1 1
1. Setup Flutter and Android 1
2. Create New Project 5
3. Run Starter Flutter app 7
4. Test hot Reload 7
5. Use an External Package 8
6. Add a Stateful Widget 11
7. Create an Infinite Scrolling ListView 13
Write Your First Flutter App Part 2 16
1. Add icons to the list 16
2. Add Interactivity 18
6. Navigate to a new screen 19
8. Change the UI using themes 24
Create Task Management App 25
1. Open pre-built app in VSCode 25
2. Create Card Component 26
3. Using ListViews 42
4. Filtering 48

Write Your First Flutter App, Part 1

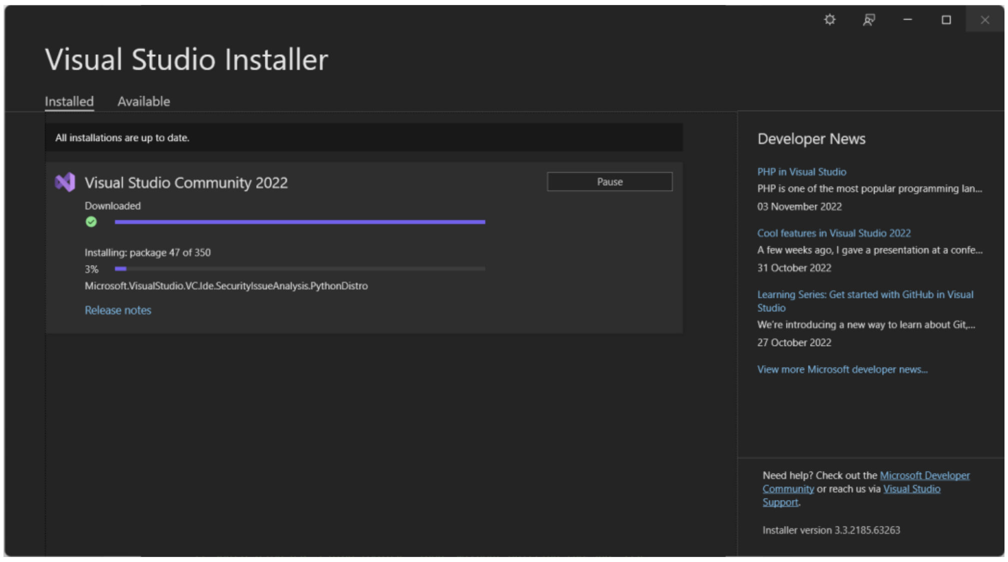
1. Setup Flutter and Android

Invoke **View > Command Palette > Flutter: Run Flutter Doctor** and fix outstanding Flutter issues listed below from the OUTPUT panel to make sure Flutter and its related programs are setup correctly.

```
[flutter] flutter doctor -v
[√] Flutter (Channel stable, 3.3.6, on Microsoft Windows [Version 10.0.22000.1098], locale
en-GB)
    • Flutter version 3.3.6 on channel stable at D:\flutter
    • Upstream repository https://github.com/flutter/flutter.git
    • Framework revision 6928314d50 (2 weeks ago), 2022-10-25 16:34:41 -0400
    • Engine revision 3ad69d7be3
    • Dart version 2.18.2
    • DevTools version 2.15.0

[X] Android toolchain - develop for Android devices
    X Unable to locate Android SDK.
      Install Android Studio from: https://developer.android.com/studio/index.html
      On first launch it will assist you in installing the Android SDK components.
      (or visit https://flutter.dev/docs/get-started/install/windows#android-setup for
detailed instructions).
      If the Android SDK has been installed to a custom location, please use
      `flutter config -android-sdk` to update to that location.
```

```
[√] Chrome - develop for the web
    • Chrome at C:\Program Files\Google\Chrome\Application\chrome.exe

[!] Visual Studio - develop for Windows (Visual Studio Community 2022 17.3.6)
    • Visual Studio at C:\Program Files\Microsoft Visual Studio\2022\Community
    • Visual Studio Community 2022 version 17.3.32929.385
    X Visual Studio is missing necessary components. Please re-run the Visual Studio
installer for the "Desktop development with C++" workload, and include these components:
        MSVC v142 - VS 2019 C++ x64/x86 build tools
         - If there are multiple build tool versions available, install the latest
        C++ Cmake tools for Windows
        Windows 10 SDK

[!] Android Studio (not installed)
    • Android Studio not found; download from
https://developer.android.com/studio/index.html
      (or visit https://flutter.dev/docs/get-started/install/windows#android-setup for
detailed instructions).

[√] Connected device (3 available)
    • Windows (desktop) • windows • windows-x64    • Microsoft Windows [Version
10.0.22000.1098]
    • Chrome (web)      • chrome  • web-javascript • Google Chrome 107.0.5304.88
    • Edge (web)        • edge    • web-javascript • Microsoft Edge 107.0.1418.35

[√] HTTP Host Availability
    • All required HTTP hosts are available

! Doctor found issues in 3 categories.
Exit code 0
```

Used stack overflow to finish outstanding tasks listed above so that there are zero issues detected by Flutter Doctor. Including downloading new packages and accepting licenses etc.



All installations are up to date.

Visual Studio Community 2022
Downloaded

Installing: package 47 of 350
3%
Microsoft.VisualStudio.VC.Ide.SecurityIssueAnalysis.PythonDistro

Developer News

PHP in Visual Studio
PHP is one of the most popular programming lan...
03 November 2022

Cool features in Visual Studio 2022
A few weeks ago, I gave a presentation at a confe...
31 October 2022

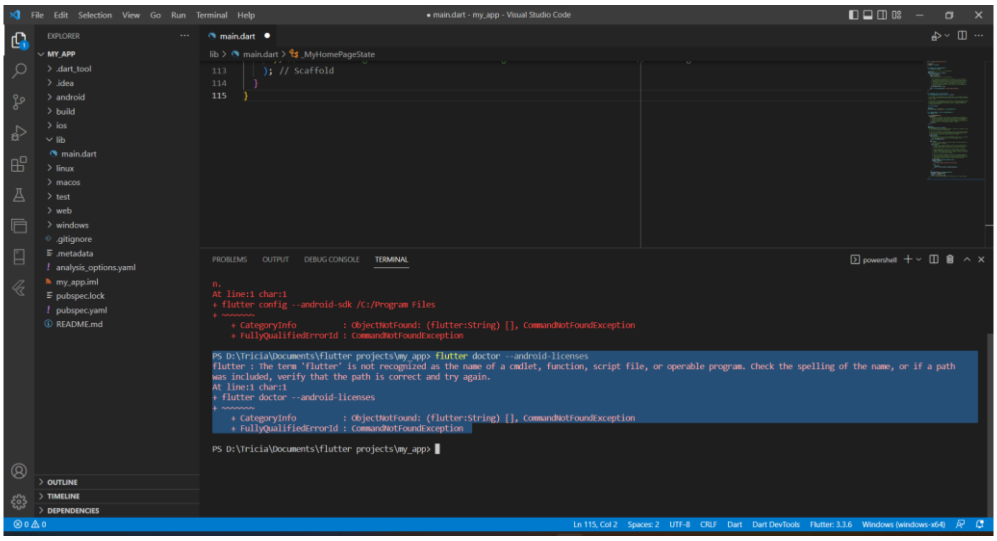
Learning Series: Get started with GitHub in Visual Studio
We're introducing a new way to learn about Git,...
27 October 2022

Need help? Check out the Microsoft Developer Community or reach us via Visual Studio Support.

Installer version 3.3.2185.63263

Downloading Visual Studio Packages

Came across the following error when trying to accept the android licenses using the TERMINAL, due to PATH being set incorrectly for Flutter SDK folder.

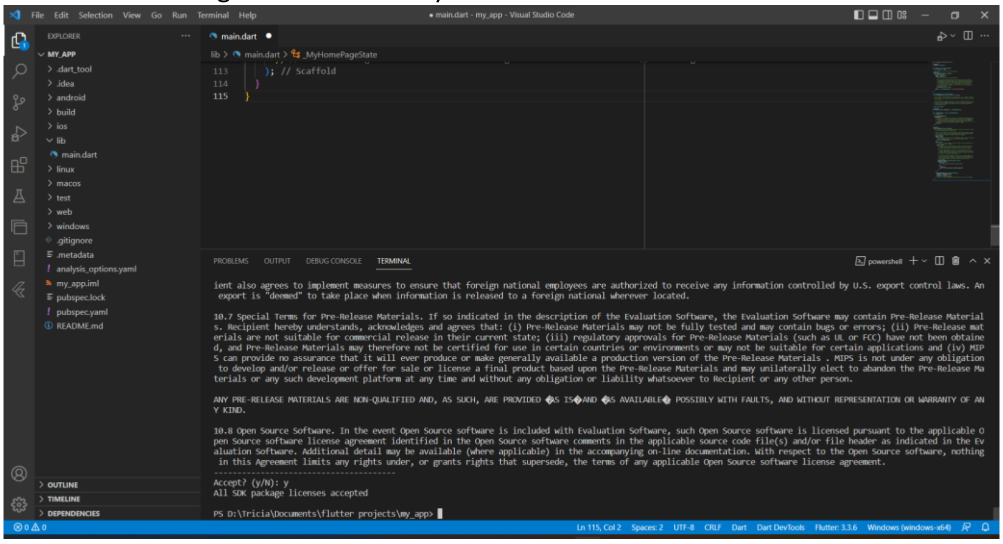


```
    ); // Scaffold
  }
}
```

```
n.
At line:1 char:1
+ flutter config --android-sdk /C:/Program Files
+ ~~~~~~~
    + CategoryInfo          : ObjectNotFound: (flutter:String) [], CommandNotFoundException
    + FullyQualifiedErrorId : CommandNotFoundException

PS D:\Tricia\Documents\flutter projects\my_app> flutter doctor --android-licenses
flutter : The term 'flutter' is not recognized as the name of a cmdlet, function, script file, or operable program. Check the spelling of the name, or if a path
was included, verify that the path is correct and try again.
At line:1 char:1
+ flutter doctor --android-licenses
+ ~~~~~~~
    + CategoryInfo          : ObjectNotFound: (flutter:String) [], CommandNotFoundException
    + FullyQualifiedErrorId : CommandNotFoundException

PS D:\Tricia\Documents\flutter projects\my_app> 
```

Error message for inability to run flutter commands in TERMINAL

Fixed Flutter SDK PATH so terminal commands, including the command to run Android licenses, can be run in the TERMINAL. To fix the last remaining issue detected by Flutter Doctor.



```
ient also agrees to implement measures to ensure that foreign national employees are authorized to receive any information controlled by U.S. export control laws. An
 export is "deemed" to take place when information is released to a foreign national wherever located.

10.7 Special Terms for Pre-Release Materials. If so indicated in the description of the Evaluation Software, the Evaluation Software may contain Pre-Release Material
s. Recipient hereby understands, acknowledges and agrees that: (i) Pre-Release Materials may not be fully tested and may contain bugs or errors; (ii) Pre-Release mat
erials are not suitable for commercial release in their current state; (iii) regulatory approvals for Pre-Release Materials (such as UL or FCC) have not been obtaine
d, and Pre-Release Materials may therefore not be certified for use in certain countries or environments or may not be suitable for certain applications and (iv) MIP
S can provide no assurance that it will ever produce or make generally available a production version of the Pre-Release Materials . MIPS is not under any obligation
 to develop and/or release or offer for sale or license a final product based upon the Pre-Release Materials and may unilaterally elect to abandon the Pre-Release Ma
terials or any such development platform at any time and without any obligation or liability whatsoever to Recipient or any other person.

ANY PRE-RELEASE MATERIALS ARE NON-QUALIFIED AND, AS SUCH, ARE PROVIDED �AS IS�AND �AS AVAILABLE� POSSIBLY WITH FAULTS, AND WITHOUT REPRESENTATION OR WARRANTY OF AN
Y KIND.

10.8 Open Source Software. In the event Open Source software is included with Evaluation Software, such Open Source software is licensed pursuant to the applicable O
pen Source software license agreement identified in the Open Source software comments in the applicable source code file(s) and/or file header as indicated in the Ev
aluation Software. Additional detail may be available (where applicable) in the accompanying on-line documentation. With respect to the Open Source software, nothing
 in this Agreement limits any rights under, or grants rights that supersede, the terms of any applicable Open Source software license agreement.
---------------------------------------
Accept? (y/N): y
All SDK package licenses accepted

PS D:\Tricia\Documents\flutter projects\my_app> 
```

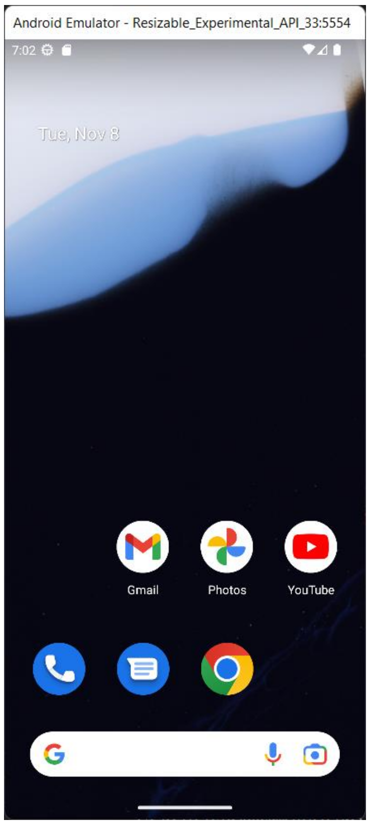
Accepted remaining Android licenses

Invoking **View > Command Palette > Flutter: Run Flutter Doctor** again to double check there are no issues left.

```
[flutter] flutter doctor -v
[√] Flutter (Channel stable, 3.3.6, on Microsoft Windows [Version 10.0.22000.1098], locale
en-GB)
    • Flutter version 3.3.6 on channel stable at D:\flutter
    • Upstream repository https://github.com/flutter/flutter.git
    • Framework revision 6928314d50 (2 weeks ago), 2022-10-25 16:34:41 -0400
    • Engine revision 3ad69d7be3
    • Dart version 2.18.2
    • DevTools version 2.15.0

[√] Android toolchain - develop for Android devices (Android SDK version 33.0.0)
    • Android SDK at C:\Users\Tricia\AppData\Local\Android\sdk
    • Platform android-33, build-tools 33.0.0
    • Java binary at: D:\Tricia\Documents\flutter projects\Android\Android
Studio\jre\bin\java
    • Java version OpenJDK Runtime Environment (build 11.0.13+0-b1751.21-8125866)
    • All Android licenses accepted.
```

```
[√] Chrome - develop for the web
    • Chrome at C:\Program Files\Google\Chrome\Application\chrome.exe

[√] Visual Studio - develop for Windows (Visual Studio Community 2022 17.3.6)
    • Visual Studio at C:\Program Files\Microsoft Visual Studio\2022\Community
    • Visual Studio Community 2022 version 17.3.32929.385
    • Windows 10 SDK version 10.0.19041.0

[√] Android Studio (version 2021.3)
    • Android Studio at D:\Tricia\Documents\flutter projects\Android\Android Studio
    • Flutter plugin can be installed from:
      https://plugins.jetbrains.com/plugin/9212-flutter
    • Dart plugin can be installed from:
      https://plugins.jetbrains.com/plugin/6351-dart
    • Java version OpenJDK Runtime Environment (build 11.0.13+0-b1751.21-8125866)

[√] Connected device (3 available)
    • Windows (desktop) • windows • windows-x64    • Microsoft Windows [Version
10.0.22000.1098]
    • Chrome (web)      • chrome  • web-javascript • Google Chrome 107.0.5304.88
    • Edge (web)        • edge    • web-javascript • Microsoft Edge 107.0.1418.35

[√] HTTP Host Availability
    • All required HTTP hosts are available

• No issues found!
exit code 0
```

No issues detected. I can now move onto setting up the android emulator.



Screenshot of Android Emulator

Now onto creating the starter app!

2. Create New Project

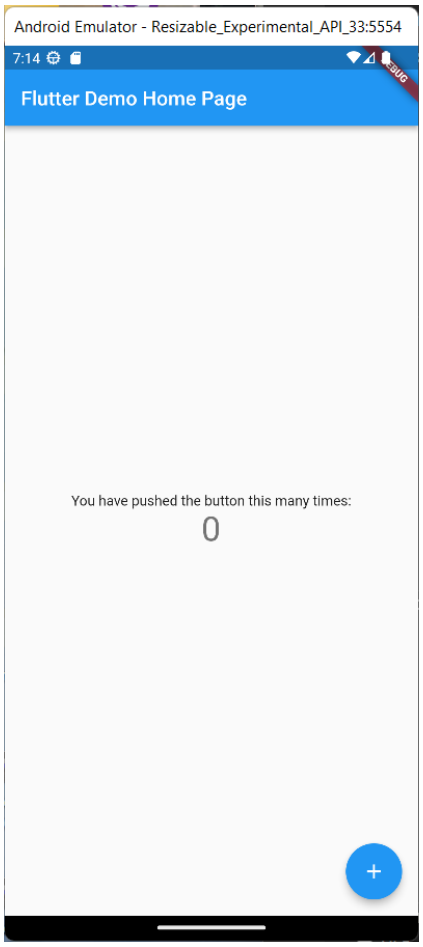
Created new project titled ‘my_app’ by invoking **View > Command Palette** and selecting **Flutter: New Project**. The ‘main.dart’ file was created the contents of which are shown below.

```dart
import 'package:flutter/material.dart';

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        // This is the theme of your application.
        //
        // Try running your application with "flutter run". You'll see the
        // application has a blue toolbar. Then, without quitting the app, try
        // changing the primarySwatch below to Colors.green and then invoke
        // "hot reload" (press "r" in the console where you ran "flutter run",
        // or simply save your changes to "hot reload" in a Flutter IDE).
        // Notice that the counter didn't reset back to zero; the application
        // is not restarted.
        primarySwatch: Colors.blue,
      ),
      home: const MyHomePage(title: 'Flutter Demo Home Page'),
    );
  }
}

class MyHomePage extends StatefulWidget {
  const MyHomePage({super.key, required this.title});

  // This widget is the home page of your application. It is stateful, meaning
  // that it has a State object (defined below) that contains fields that affect
  // how it looks.

  // This class is the configuration for the state. It holds the values (in this
  // case the title) provided by the parent (in this case the App widget) and
  // used by the build method of the State. Fields in a Widget subclass are
  // always marked "final".

  final String title;

  @override
  State<MyHomePage> createState() => _MyHomePageState();
}
```

```dart
class _MyHomePageState extends State<MyHomePage> {
  int _counter = 0;

  void _incrementCounter() {
    setState(() {
      // This call to setState tells the Flutter framework that something has
      // changed in this State, which causes it to rerun the build method below
      // so that the display can reflect the updated values. If we changed
      // _counter without calling setState(), then the build method would not be
      // called again, and so nothing would appear to happen.
      _counter++;
    });
  }

  @override
  Widget build(BuildContext context) {
    // This method is rerun every time setState is called, for instance as done
    // by the _incrementCounter method above.
    //
    // The Flutter framework has been optimized to make rerunning build methods
    // fast, so that you can just rebuild anything that needs updating rather
    // than having to individually change instances of widgets.
    return Scaffold(
      appBar: AppBar(
        // Here we take the value from the MyHomePage object that was created by
        // the App.build method, and use it to set our appbar title.
        title: Text(widget.title),
      ),
      body: Center(
        // Center is a layout widget. It takes a single child and positions it
        // in the middle of the parent.
        child: Column(
          // Column is also a layout widget. It takes a list of children and
          // arranges them vertically. By default, it sizes itself to fit its
          // children horizontally, and tries to be as tall as its parent.
          //
          // Invoke "debug painting" (press "p" in the console, choose the
          // "Toggle Debug Paint" action from the Flutter Inspector in Android
          // Studio, or the "Toggle Debug Paint" command in Visual Studio Code)
          // to see the wireframe for each widget.
          //
          // Column has various properties to control how it sizes itself and
          // how it positions its children. Here we use mainAxisAlignment to
          // center the children vertically; the main axis here is the vertical
          // axis because Columns are vertical (the cross axis would be
          // horizontal).
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            const Text(
              'You have pushed the button this many times:',
            ),
            Text(
              '$_counter',
              style: Theme.of(context).textTheme.headline4,
            ),
          ],
        ),
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: _incrementCounter,
        tooltip: 'Increment',
        child: const Icon(Icons.add),
      ), // This trailing comma makes auto-formatting nicer for build methods.
    );
  }
}
```

3. Run Starter Flutter app

Press F5 to start debugging and display starter Material app on emulator.


Android Emulator - Resizable_Experimental_API_33:5554

7:14

Flutter Demo Home Page

You have pushed the button this many times:

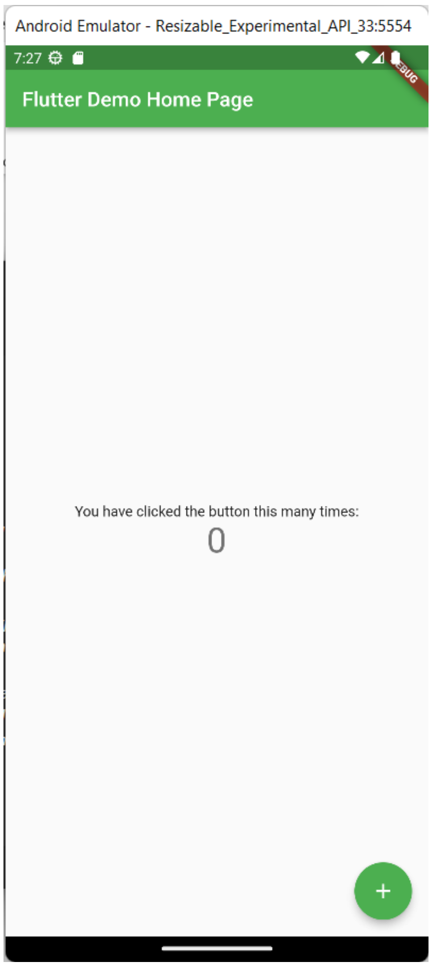
0

First look at starter app on emulator

4. Test hot Reload

Change app colour to green and the centre string from ‘pushed’ to ‘clicked’ then use hot reload (**Ctrl + F5** or lightening symbol at top of VS code) to see the new changes being made quicker.

```
        primarySwatch: Colors.green,
            const Text(
              'You have clicked the button this many times:',
            ),
```

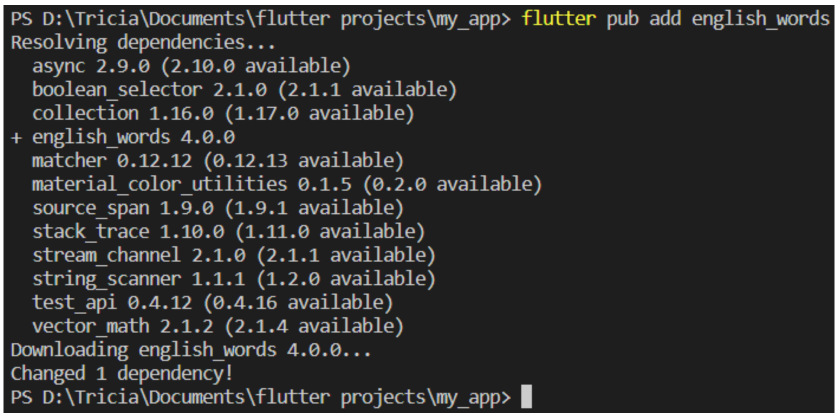

Android Emulator - Resizable_Experimental_API_33:5554
7:27
Flutter Demo Home Page
You have clicked the button this many times:
0
+

New changes to app after hot reload

Hot reload works quicker because it doesn't restart the app. Sometimes need to run debug though some changes don't show up unless you do.

5. Use an External Package

First need to add an open source package containing 1000s of the most-used English words and some utility functions using the TERMINAL and typing in the following line to add the package named 'english_words' as a dependency of the app.



```
PS D:\Tricia\Documents\flutter projects\my_app> flutter pub add english_words
Resolving dependencies...
  async 2.9.0 (2.10.0 available)
  boolean_selector 2.1.0 (2.1.1 available)
  collection 1.16.0 (1.17.0 available)
+ english_words 4.0.0
  matcher 0.12.12 (0.12.13 available)
  material_color_utilities 0.1.5 (0.2.0 available)
  source_span 1.9.0 (1.9.1 available)
  stack_trace 1.10.0 (1.11.0 available)
  stream_channel 2.1.0 (2.1.1 available)
  string_scanner 1.1.1 (1.2.0 available)
  test_api 0.4.12 (0.4.16 available)
  vector_math 2.1.2 (2.1.4 available)
Downloading english_words 4.0.0...
Changed 1 dependency!
PS D:\Tricia\Documents\flutter projects\my_app>
```

Import open source 'english_words' package

Then we need to import the newly added package for use in main.dart – works like importing python packages.

```
import 'package:flutter/material.dart';
import 'package:english_words/english_words.dart';
```

and add the following changes to the main.dart section of code from the before (below)...

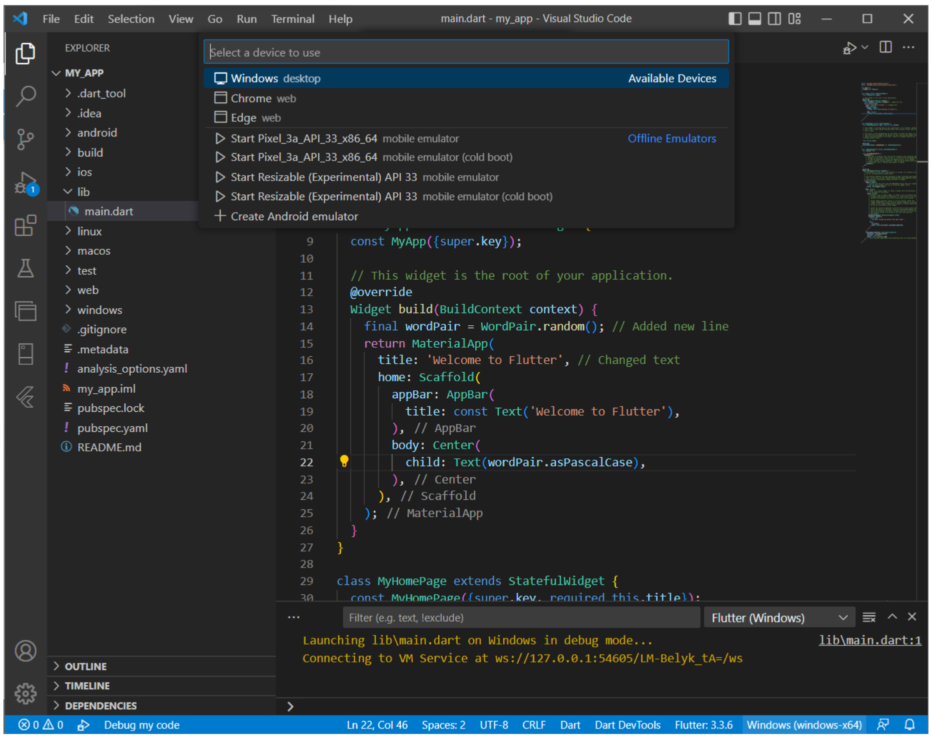
```
void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        // This is the theme of your application.
        primarySwatch: Colors.green,
      ),
      home: const MyHomePage(title: 'Flutter Demo Home Page'),
    );
  }
}
```

…to the after (below). New changes made are explained below.

`asPascalCase` displays each individual word in a string with a capital letter e.g. helloworld would become HelloWord.
`final wordPair` final creates a hardcoded variable called 'wordpair' which combines together two words, a first and a second word or a two part compound.
`WordPair.random()` creates a randomly generated single WordPair or single combination of two words.
`Return MaterialApp` widget that wraps widgets required for material design.
`Class` contained inside of a class is blueprint for an object you're creating.
`Extends` extends properties and objects within one class across another, similar, class.
`@override` overrides superclass member with same name i.e. the overriding class is prioritised.
`Scaffold()` implements basic material design for drawers and bottom sheets.
`appBar()` design for bar at top of app where title is contained.
`body:` main body of app.
`Center(child: Text())` – writes down words generated in wordPair variable created earlier in pascal case also described earlier.

```
void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    final wordPair = WordPair.random(); // Added new line
    return MaterialApp(
      title: 'Welcome to Flutter', // Changed text
      home: Scaffold(
        appBar: AppBar(
          title: const Text('Welcome to Flutter'),
        ),
        body: Center(
          child: Text(wordPair.asPascalCase),
        ),
      ),
    );
  }
}
```

Note that running app emulator on windows device (see below) makes it load quicker than android emulator.

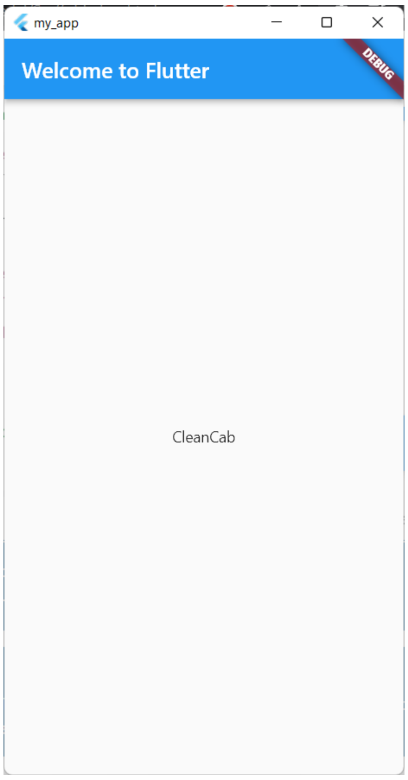

File Edit Selection View Go Run Terminal Help
main.dart - my_app - Visual Studio Code
EXPLORER
MY_APP
.dart_tool
.idea
android
build
ios
lib
main.dart
linux
macos
test
web
windows
.gitignore
.metadata
analysis_options.yaml
my_app.iml
pubspec.lock
pubspec.yaml
README.md
OUTLINE
TIMELINE
DEPENDENCIES
Select a device to use
Windows desktop
Available Devices
Chrome web
Edge web
Start Pixel_3a_API_33_x86_64 mobile emulator
Offline Emulators
Start Pixel_3a_API_33_x86_64 mobile emulator (cold boot)
Start Resizable (Experimental) API 33 mobile emulator
Start Resizable (Experimental) API 33 mobile emulator (cold boot)
Create Android emulator
Filter (e.g. text, !exclude)
Flutter (Windows)
lib\main.dart:1
Debug my code
Ln 22, Col 46 Spaces: 2 UTF-8 CRLF Dart Dart DevTools Flutter: 3.3.6 Windows (windows-x64)

```
  const MyApp({super.key});

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    final wordPair = WordPair.random(); // Added new line
    return MaterialApp(
      title: 'Welcome to Flutter', // Changed text
      home: Scaffold(
        appBar: AppBar(
          title: const Text('Welcome to Flutter'),
        ), // AppBar
        body: Center(
          child: Text(wordPair.asPascalCase),
        ), // Center
      ), // Scaffold
    ); // MaterialApp
  }
}

class MyHomePage extends StatefulWidget {
  const MyHomePage({super.key, required this.title});
```

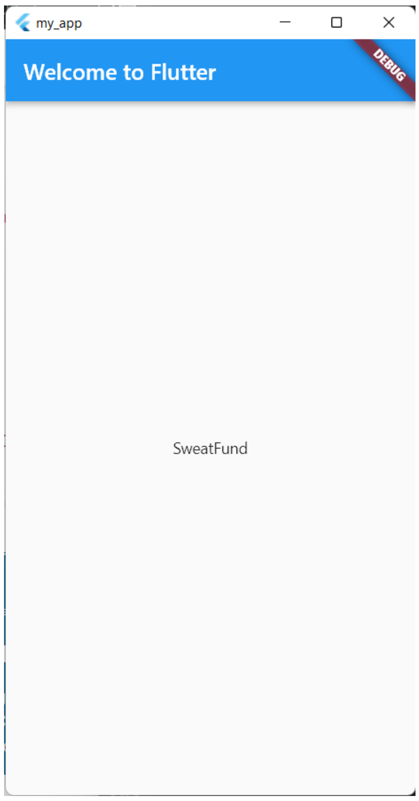
```
Launching lib\main.dart on Windows in debug mode...
Connecting to VM Service at ws://127.0.0.1:54605/LM-Belyk_tA=/ws
```

Windows Device selected to display app

The new code added changed the title of the app at the top blue bar and also generates a new string which is a pair of randomly generated words put together centered at the body of the app as shown below.



App in windows display after changes made to code – a new wordPair is generated with each run



SweatFund

App re-displayed using hot reload in windows again with a new wordPair

6. Add a Stateful Widget

Stateless widgets are widgets that cannot be changed (called immutable) – the only way to implement changes is to throw away the widget and regenerate a new one. The `MyApp` widget is an example of a stateless widget here.
On the other hand, a stateful widget can be changed (mutable).
State objects persist over the lifetime of the widget.
We're going to be creating a stateful widget called `RandomWords` that has a state class `_RandomWordsState`
The widget `RandomWords` will exist as a child inside the already existing `MyApp` stateless widget.

```
class RandomWords extends StatefulWidget {
  const RandomWords({super.key});

  @override
  State<RandomWords> createState() => _RandomWordsState();
}

class _RandomWordsState extends State<RandomWords> {
  @override
  Widget build(BuildContext context) {
    return Container();
  }
}
```

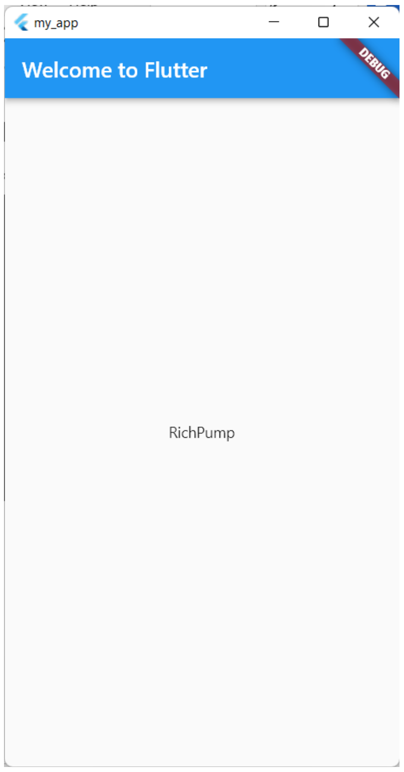
Updated the build() method with the following two new lines.

```
class RandomWords extends StatefulWidget {
  const RandomWords({super.key});

  @override
  State<RandomWords> createState() => _RandomWordsState();
}

class _RandomWordsState extends State<RandomWords> {
  @override
  Widget build(BuildContext context) {
    final wordPair = WordPair.random(); // NEW
    return Text(wordPair.asPascalCase); // NEW
  }
}
```

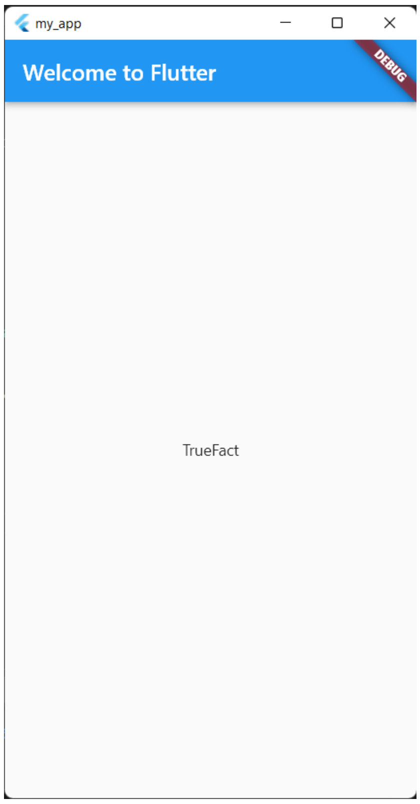
And updated the MyApp class. Essentially moved random word pair generator to the new widget built.

```
class MyApp extends StatelessWidget {
  const MyApp({super.key});

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Welcome to Flutter', // Changed text
      home: Scaffold(
        appBar: AppBar(
          title: const Text('Welcome to Flutter'),
        ),
        body: const Center(
          child: RandomWords(),
        ),
      ),
    );
  }
}
```

Hot reload should generate random word pairs, as it did previously.



App after adding stateful widget – functions the same as before generating a new word.



New word generated.

7. Create an Infinite Scrolling ListView

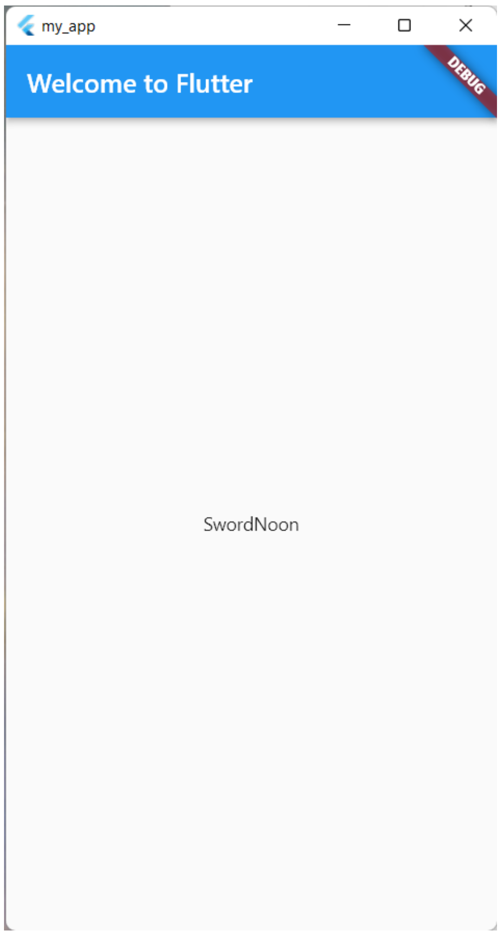
Next task is to generate a list of word pairs that grows and displays more words when the user scrolls. The list of words will grow infinitely longer.

We make some changes to the `_RandomWordsState` class from the before (top) to the after (bottom).

```
class _RandomWordsState extends State<RandomWords> {
  @override
  Widget build(BuildContext context) {
    final wordPair = WordPair.random(); // NEW
    return Text(wordPair.asPascalCase); // NEW
  }
}
```

```
class _RandomWordsState extends State<RandomWords> {
  final _suggestions = <WordPair>[]; // NEW
  final _biggerFont = const TextStyle(fontsize: 18); //NEW

  @override
  Widget build(BuildContext context) {
    final wordPair = WordPair.random();
    return Text(wordPair.asPascalCase);
  }
}
```

The two new added variables `_suggestions` holds a list for saving generated word pairings and `_biggerFont` just makes the font size larger.



No visual changes to app after adding the new code

Added new changes to below `@override` section in the same `_RandomWordsState`.

```
class _RandomWordsState extends State<RandomWords> {
  final _suggestions = <WordPair>[]; // NEW
  final _biggerFont = const TextStyle(fontSize: 18); //NEW

@override
  Widget build(BuildContext context) {
    return ListView.builder(
      padding: const EdgeInsets.all(16.0),
      itemBuilder: (context, i) {
        if (i.isOdd)
return const Divider();

        final index = i ~/ 2;
        if (index >= _suggestions.length) {
          _suggestions.addAll(generateWordPairs().take(10));
        }
        return ListTile(
          title: Text(
            _suggestions[index].asPascalCase,
            style: _biggerFont,
          ),
        );
      },
    );
  }
}
```

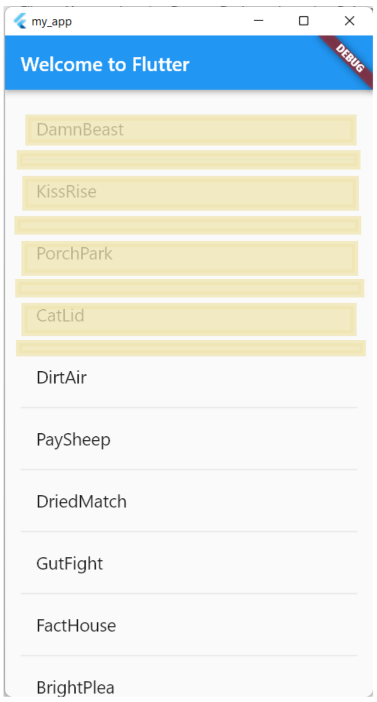
`padding:` determines space between edge of word boxes.

```
@override
  Widget build(BuildContext context) { }
```

The entirety of the above is called the build method.
`itemBuilder:` is used to build list item widgets so itemBuilder callback is called every time a suggested word pairing is created and places each item by returning into a `ListTile` row.
`itemBuilder` runs through index `I` and every odd `i` index a `divider()` thin horizontal line is generated with padding and every even `i` index a word pair is suggested and displayed on the screen and appends to the end of the list with `.addAll.`
so `_suggestions` is a variable that is a list (of word) and `<WordPair>[]` with the blank `[]` to indicate an empty growable list.
Every time the user scrolls the index `i` which I believe is the number of yellow box elements `i` (highlighted below) - gets bigger and every time the index number `i` becomes larger than the list of suggested words generated and displayed on the app a new word pair is created – so a new word pair is generated and added to the existing list and displayed on screen.
Then after each word pair is generated a divider element (thin horizontal grey line) is placed under the word pair to separate between word pairs.
and the `style:` indicates what font style to use which we created a variable `_biggerFont` that is used alongside `style.`
If an identifier starts with _ this indicates private variable.

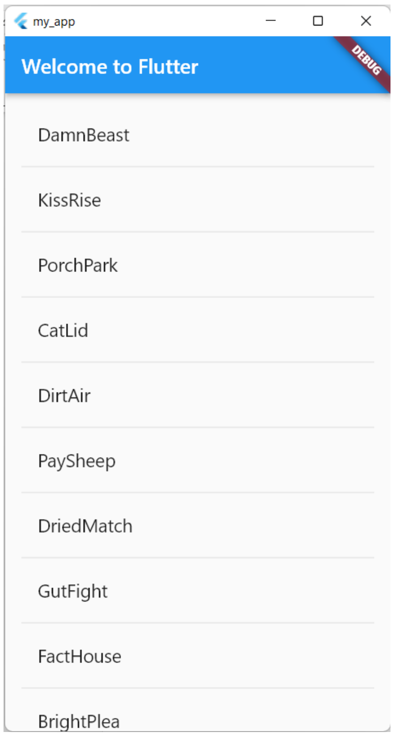
```
        return ListTile(
          title: Text(
            _suggestions[index].asPascalCase,
            style: _biggerFont,
```

The above returns list of suggested word pairs and stores most recently generated word pair in index position `i` of the `_suggestions` variable…



Yellow highlighted boxes shows how elements of app are divided

…which results in the following changes to the app. A new word pairing should be generated no matter how far you scroll.


my_app

Welcome to Flutter

DamnBeast

KissRise

PorchPark

CatLid

DirtAir

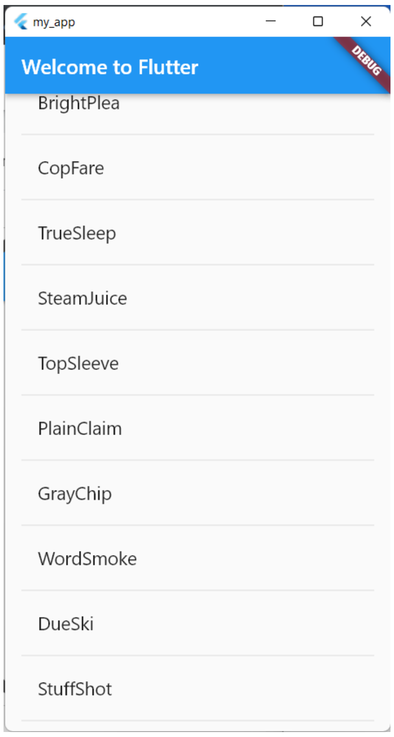
PaySheep

DriedMatch

GutFight

FactHouse

BrightPlea


my_app

Welcome to Flutter

BrightPlea

CopFare

TrueSleep

SteamJuice

TopSleeve

PlainClaim

GrayChip

WordSmoke

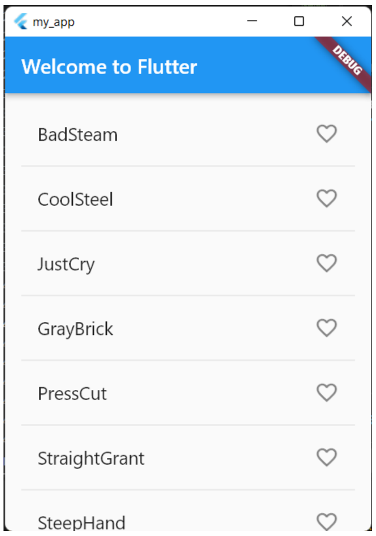
DueSki

StuffShot

New changes to app after change to build() method of app (left) and when scrolling down (right) new words are generated

Write Your First Flutter App Part 2

1. Add icons to the list


my_app

Welcome to Flutter

BadSteam

CoolSteel

JustCry

GrayBrick

PressCut

StraightGrant

SteepHand

App after changes made to add heart icons to list to allow for selection of favourite word pairs later

Open hearts appear on each row of words but aren't interactive yet. We are still working within the `_RandomWordsState` class. New changes are indicated by the `// NEW` below and we'll go through every change added and what the code does.

```
class _RandomWordsState extends State<RandomWords> {
  final _suggestions = <WordPair>[];
  final _biggerFont = const TextStyle(fontSize: 18);
  final _saved = <WordPair>{}; // NEW

  @override
  Widget build(BuildContext context) {
    return ListView.builder(
      padding: const EdgeInsets.all(16.0),
      itemBuilder: (context, i) {
        if (i.isOdd)
       return const Divider();
        final index = i ~/ 2;
        if (index >= _suggestions.length) {
          _suggestions.addAll(generateWordPairs().take(10));
        }
        final alreadySaved = _saved.contains(_suggestions[index]); // NEW
        return ListTile(
            title: Text(
              _suggestions[index].asPascalCase,
              style: _biggerFont,
            ),
            trailing: Icon(
                // NEW from here ...
                alreadySaved ? Icons.favorite : Icons.favorite_border,
                color: alreadySaved ? Colors.red : null,
                semanticLabel: alreadySaved ? 'Remove from saved' : 'Save')
                // to here.
                );
      },
    );
  }
}
```

```
  final _saved = <WordPair>{}; // NEW
```

In the above a new variable called `_saved` was added which is currently an empty list that will store saved words – a function that we'll be adding later.

```
        final alreadySaved = _saved.contains(_suggestions[index]); // NEW
```

Created a new variable called `alreadySaved` which checks if a certain word pairing in the `_suggestions[index]` which `_suggestions` is the full list of generated word pairs and index is the word pair a generated word is contained within.

`.contain` method checks the full list of word pairs in `_saved` variable against the selected word pair in the `_suggestions` variable – the particular word pair being checked is indicated by [index]. _saved is variable of already saved words.

Note that index is actually only every even index number is checked for words because all odd index elements contain the divider line.

```
return ListTile(
            title: Text(
              _suggestions[index].asPascalCase,
              style: _biggerFont,
            ),
            trailing: Icon(
                // NEW from here ...
                alreadySaved ? Icons.favorite : Icons.favorite_border,
                color: alreadySaved ? Colors.red : null,
                semanticLabel: alreadySaved ? 'Remove from saved' : 'Save')
                // to here.
                );
```

Then in `ListTile()` we made it so that every new word pair text item on the list also comes with a heart alongside it. New added heart feature is described in the `trailing: Icon()` section of code. `Title:` stores and prints word pair text `Text()` and `trailing:` is a widget you display after the title and in this widget we want to display an icon `Icon()`.
`?` indicates alreadySaved variable can be null.

♥ – material icon named "favorite".

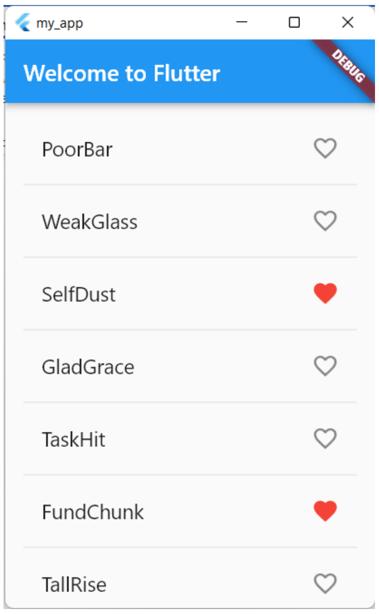
`Icons.Favorite` is the material icon named 'favorite'.

♡ – material icon named "favorite border".

`Icons.Favorite_border` is material icon named 'favorite border'.

`Color: alreadySaved ? Colors.red : null` means to use red.
`semanticLabels` are not shown in UI but announced in accessibility modes e.g. text to speech.

2. Add Interactivity


my_app

Welcome to Flutter

DEBUG

PoorBar ♡

WeakGlass ♡

SelfDust ♥

GladGrace ♡

TaskHit ♡

FundChunk ♥

TallRise ♡

Added new feature where we make the heart icons we added before toggleable

New feature added makes it so when you tap on heart it changes colour when toggled.

Still working in the `_RandomWordStates` class specifically the list output in `_ListTile()` – building on the `trailing: Icon` we built before – we're now going to add a feature that says what to do when the heart icon is tapped in `onTap: () {}`.

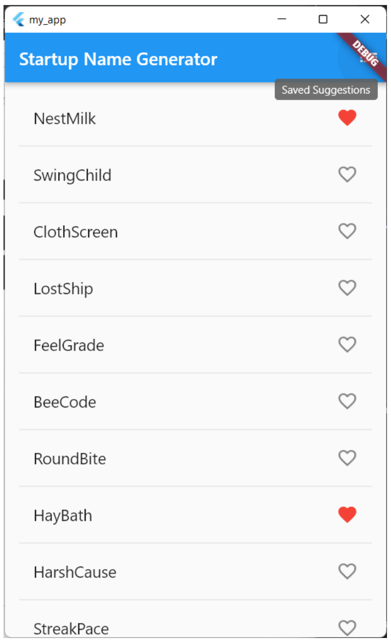
```
class _RandomWordsState extends State<RandomWords> {
  final _suggestions = <WordPair>[];
  final _biggerFont = const TextStyle(fontSize: 18);
  final _saved = <WordPair>{}; // NEW

  @override
  Widget build(BuildContext context) {
    return ListView.builder(
        padding: const EdgeInsets.all(16.0),
        itemBuilder: (context, i) {
          if (i.isOdd) return const Divider();
          final index = i ~/ 2;
          if (index >= _suggestions.length) {
            _suggestions.addAll(generateWordPairs().take(10));
          }
          final alreadySaved = _saved.contains(_suggestions[index]); // NEW
          return ListTile(
            title: Text(
              _suggestions[index].asPascalCase,
              style: _biggerFont,
            ),
            trailing: Icon(
                alreadySaved ? Icons.favorite : Icons.favorite_border,
                color: alreadySaved ? Colors.red : null,
                semanticLabel: alreadySaved ? 'Remove from saved' : 'Save'),
            onTap: () { // NEW from here ...
              setState(
                () {
                if (alreadySaved) {
                  _saved.remove(_suggestions[index]);
                } else {
                  _saved.add(_suggestions[index]);
                }
              }
              ); // to here.
            },
          );
        }
    );
  }
}
```

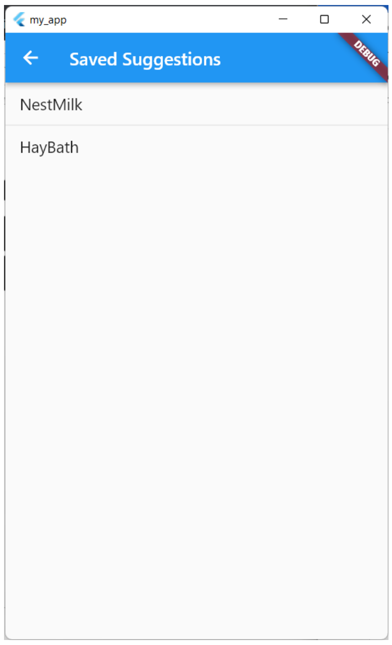
```
            onTap: () { // NEW from here ...
              setState(
                () {
                if (alreadySaved) {
                  _saved.remove(_suggestions[index]);
                } else {
                  _saved.add(_suggestions[index]);
                }
              }
              ); // to here.
```

`Setstate()` indicates change in internal state of this object.
Created `if` and `else` statement.
`if (alreadySaved) {}` means if the word pair corresponding to the heart icon is in the `_saved` list (remember `alreadySaved` checks word pair against the `_saved` list then we remove the word from that particular index in `suggestion[index]` from the `_saved` list using `.remove`.

In the `else {}` statement states what happens if there is no match between `_saved` and `_suggestions` word pairs. It adds using `.add` the selected wordPair in `_suggestions[index]`.

6. Navigate to a new screen





Adding code to navigate to a new page called 'Saved Suggestions' (or route as it's called in Flutter)

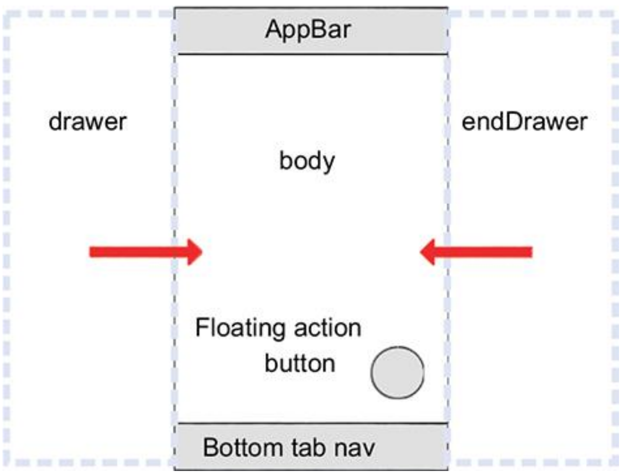
In this section I created a new page or route on the app which directs you to page showing list of saved suggested word pairs.

Starting with `MyApp`class below. This is the before code inside the `MyApp` class.

```
class MyApp extends StatelessWidget {
  const MyApp({super.key});

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Welcome to Flutter', // Changed text
      home: Scaffold(
        appBar: AppBar(
          title: const Text('Welcome to Flutter'),
        ),
        body: const Center(
          child: RandomWords(),
        ),
      ),
    );
  }
}
```

Then after applying changes where we removed the `Scaffold()` from `home: Scaffold()` and changed it to `RandomWords().` We will later move `Scaffold()` to `_RandomWordsState` class so we can add an icon button to the scaffold's `AppBar` that can direct to these new saved suggestions page.

`Home:` is the default route of the app.
`Scaffold()` class implements visual layout structure of page e.g.



Scaffold area spans across entire screen basically including nav tab, body, floating action button & appbar

Below is the changes to the `MyApp` class, with the `Scaffold()` class removed and the `RandomWords()` class added.

```
class MyApp extends StatelessWidget {
  const MyApp({super.key});

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return const MaterialApp(
      title: 'Startup Name Generator',
      home: RandomWords(), // NEW
    );
  }
}
```

Below is the before code on `_RandomWordsStates` class. We are going to remove the `ListView.builder()` from the return and swap it for the `Scaffold()` class, the brackets of which will encompass the `ListView.builder()`. Then move `ListView.builder()` to `body:` which we will add inside the newly added `Scaffold()` class.

```
class _RandomWordsState extends State<RandomWords> {
  final _suggestions = <WordPair>[];
  final _biggerFont = const TextStyle(fontSize: 18);
  final _saved = <WordPair>{}; // NEW

  @override
  Widget build(BuildContext context) {
    return ListView.builder(
        padding: const EdgeInsets.all(16.0),
        itemBuilder: (context, i) {
          if (i.isOdd) return const Divider();
          final index = i ~/ 2;
          if (index >= _suggestions.length) {
            _suggestions.addAll(generateWordPairs().take(10));
          }
          final alreadySaved = _saved.contains(_suggestions[index]);
          return ListTile(
            title: Text(
              _suggestions[index].asPascalCase,
              style: _biggerFont,
            ),
            trailing: Icon(
                alreadySaved ? Icons.favorite : Icons.favorite_border,
                color: alreadySaved ? Colors.red : null,
                semanticLabel: alreadySaved ? 'Remove from saved' : 'Save'),
            onTap: () {

              setState(() {
                if (alreadySaved) {
                  _saved.remove(_suggestions[index]);
                } else {
                  _saved.add(_suggestions[index]);
                }
              });
            },
          );
        });
  }
}
```

Below is the build method code section after changes have been made. We added the `Scaffold()` after return and made it so that it encompasses the newly moved `ListView.builder()` which we placed next to `body:`

`Body:` is the class for body of app (see diagram above).
`IconButton()` to add an icon button
`Tooltip:` provides text labels for functions of buttons when long pressing widget
`onPressed:` to execute navigation to `_pushSaved` route we create later in the upper half of the `_RandomWordsState` class.

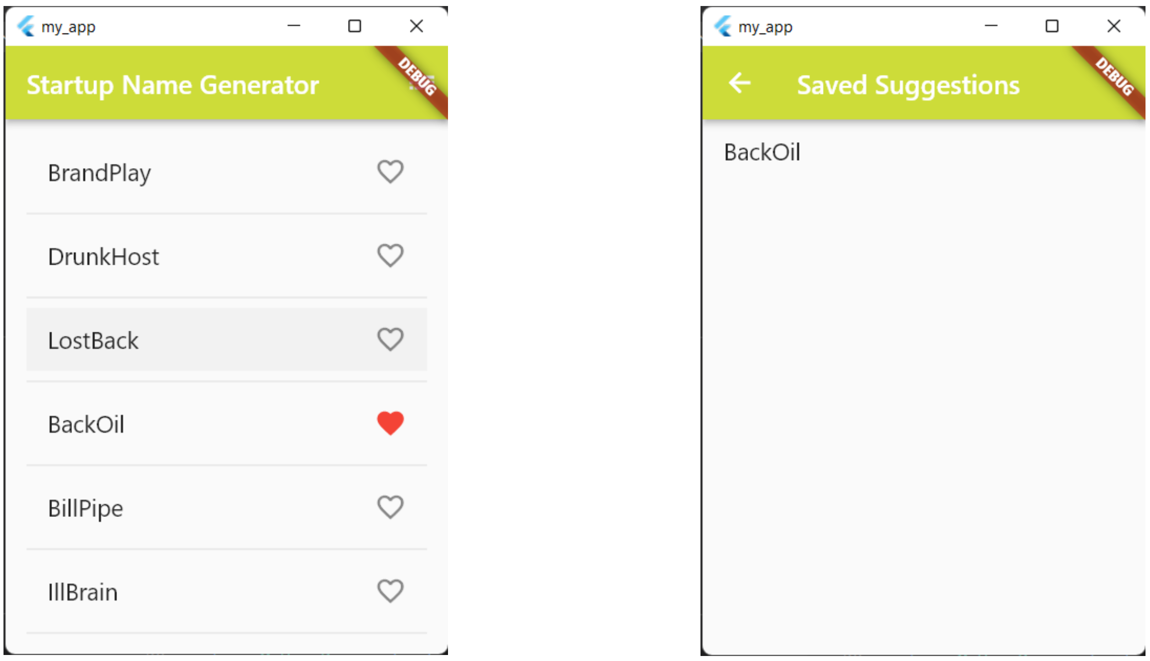
```
  @override
  Widget build(BuildContext context) {
    // NEW from here ...
    return Scaffold(
      appBar: AppBar(
        title: const Text('Startup Name Generator'),
        actions: [
          IconButton(
            icon: const Icon(Icons.list),
            onPressed: _pushSaved,
            tooltip: 'Saved Suggestions',
          ),
        ],
      ),
      body: ListView.builder(
          // to here.
          padding: const EdgeInsets.all(16.0),
          itemBuilder: (context, i) {
            if (i.isOdd) return const Divider();
            final index = i ~/ 2;
            if (index >= _suggestions.length) {
              _suggestions.addAll(generateWordPairs().take(10));
            }
            final alreadySaved = _saved.contains(_suggestions[index]);
            return ListTile(
              title: Text(
                _suggestions[index].asPascalCase,
                style: _biggerFont,
              ),
              trailing: Icon(
                  alreadySaved ? Icons.favorite : Icons.favorite_border,
                  color: alreadySaved ? Colors.red : null,
                  semanticLabel: alreadySaved ? 'Remove from saved' : 'Save'),
              onTap: () {
                setState(() {
                  if (alreadySaved) {
                    _saved.remove(_suggestions[index]);
                  } else {
                    _saved.add(_suggestions[index]);
                  }
                });
              },
            );
          }),
    ); // NEW end of Scaffold
  }
```

Then still in the `_RandomWordsState` we add a `_pushSaved()` function outside and above the `Scaffolding().`
`Void` is just a marker doesn't actually manifest as anything in the app.
`Navigator` manages the app's routes, or the app's pages.
`Navigator.push` means to push a route to the Navigator's stack – which means to change the screen to display the new route.
Context is build context of a widget.
`MaterialPageRoute` is visuals of new route or page we're creating.
Return `ListTile()` dictates how list of saved words appear on page font, letters etc. contents of brackets determine how to display list.
`isNotEmpty` checks if string is not empty.
and `divideTiles()` puts divider between words and contexts `.toList()` collects elements into a list.
We have two `Scaffold()`'s and `appBar`'s one for the main page 'Startup Name Generator' and another for the 'Saved Suggestions' page.

```
class _RandomWordsState extends State<RandomWords> {
  final _suggestions = <WordPair>[];
  final _biggerFont = const TextStyle(fontSize: 18);
  final _saved = <WordPair>{};

  void _pushSaved() {
    // NEW from here ...
    Navigator.of(context).push(
      MaterialPageRoute<void>(
        builder: (context) {
          final tiles = _saved.map(
            (pair) {
              return ListTile(
                title: Text(
                  pair.asPascalCase,
                  style: _biggerFont,
                ),
              );
            },
          );
          final divided = tiles.isNotEmpty
```

```
       ? ListTile.divideTiles(
         context: context,
         tiles: tiles,
        ).toList()
       : <Widget>[];

     return Scaffold(
       appBar: AppBar(
        title: const Text('Saved Suggestions'),
       ),
       body: ListView(children: divided));
    },
   ), // to here.
  );
 }
```

8. Change the UI using themes



Changed UI theme to black (foreground) and white (background)

Working in the `MyApp` class this time, below is the code prior to changes.

```
class MyApp extends StatelessWidget {
 const MyApp({super.key});

 // This widget is the root of your application.
 @override
 Widget build(BuildContext context) {
  return const MaterialApp(
   title: 'Startup Name Generator',
   home: RandomWords(),
  );
 }
}
```

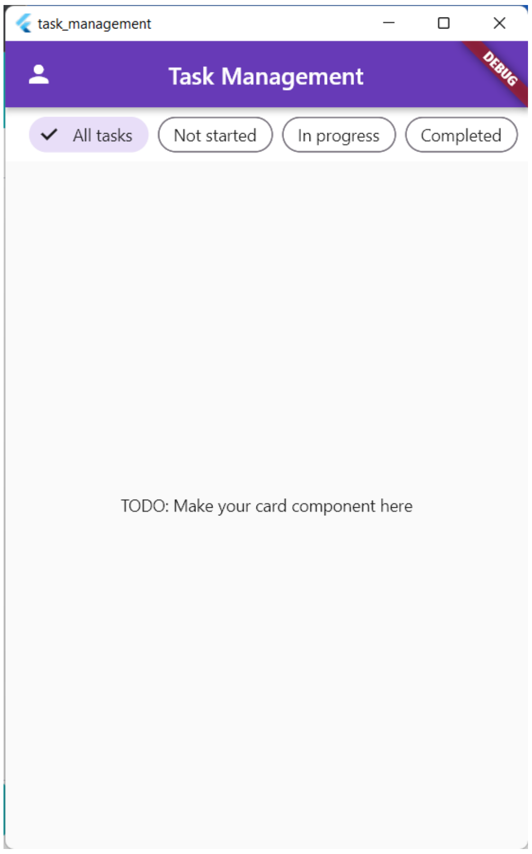
Below is the code after changes were made. Added lime top bar colour and white font to titles and icons in top bar.

```
class MyApp extends StatelessWidget {
  const MyApp({super.key});

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      // NEW
      title: 'Startup Name Generator',
      theme: ThemeData(
        // NEW from here ...
        appBarTheme: const AppBarTheme(
          backgroundColor: Colors.lime,
          foregroundColor: Colors.white,
        ), // to here.
      ),
      home: const RandomWords(),
    );
  }
}
```

Create Task Management App

Screenshots of app along with corresponding code underneath each image.

1. Open pre-built app in VSCode



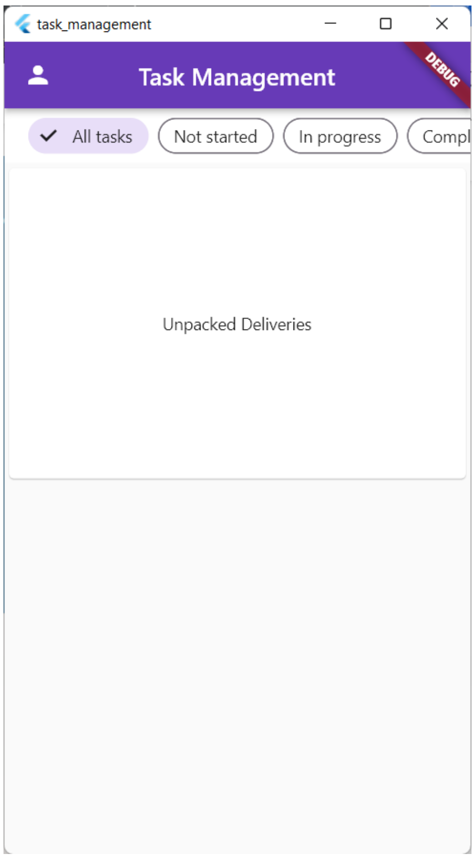
TODO: Make your card component here

App opened using windows emulator

Open app .dart file in lib folder.

2. Create Card Component

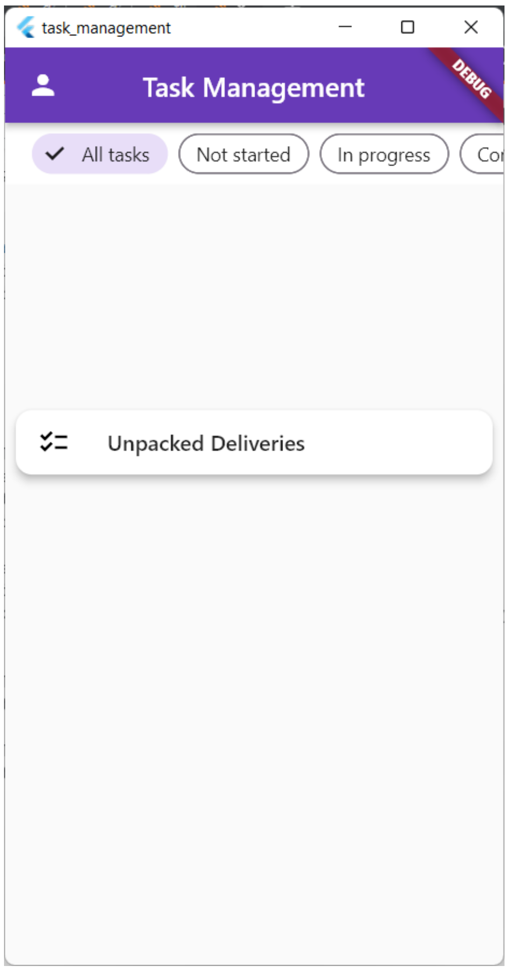
Created a card component to display summary of tasks the user needs to complete using a card widget and rows and columns to organise text and icons on the page.



Unpacked Deliveries

Card component centered on screen

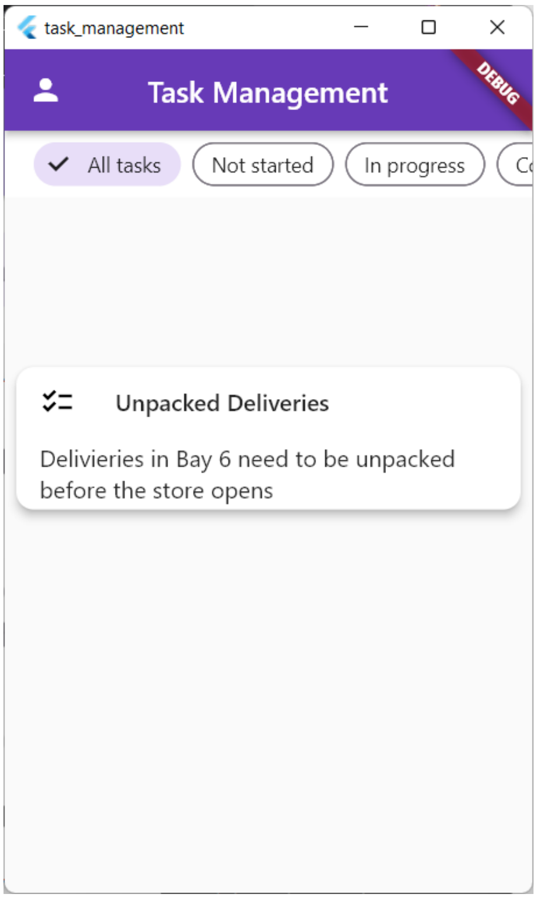
```
class TaskCard extends StatelessWidget {
  final Task task;

  const TaskCard(this.task, {Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    // NEW from here ...
    return const Center(
      child: Align(
        alignment: Alignment.topCenter,
        child: Card(
          child: SizedBox(
            height: 261,
            child: Center(child: Text('Unpacked Deliveries')),
          ),
        ),
      ),
    );
  }
}
```



Unpacked Deliveries

Added task icon and task title as string for now (later changed)

```
class TaskCard extends StatelessWidget {
  final Task task;

  const TaskCard(this.task, {Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    // NEW from here ...
    return Container(
      padding: EdgeInsets.all(8.0),
      height: 261,
      child: Align(
        alignment: Alignment.topCenter,
        child: Card(
          elevation: 5,
          margin: EdgeInsets.fromLTRB(0.0, 0.0, 0.0, 16.0),
          shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(12.0),
          ),
          child: Column(
            mainAxisSize: MainAxisSize.min,
            children: const [
              ListTile(
                title: Text(
                  'Unpacked Deliveries',
                  overflow: TextOverflow.ellipsis,
                  style: TextStyle(fontWeight: FontWeight.w500),
                ),
                leading: Icon(
                  Icons.checklist,
                  color: Colors.black,
                ),
              ),
            ],
          ),
        ),
      ),
    );
  }
}
```


task_management

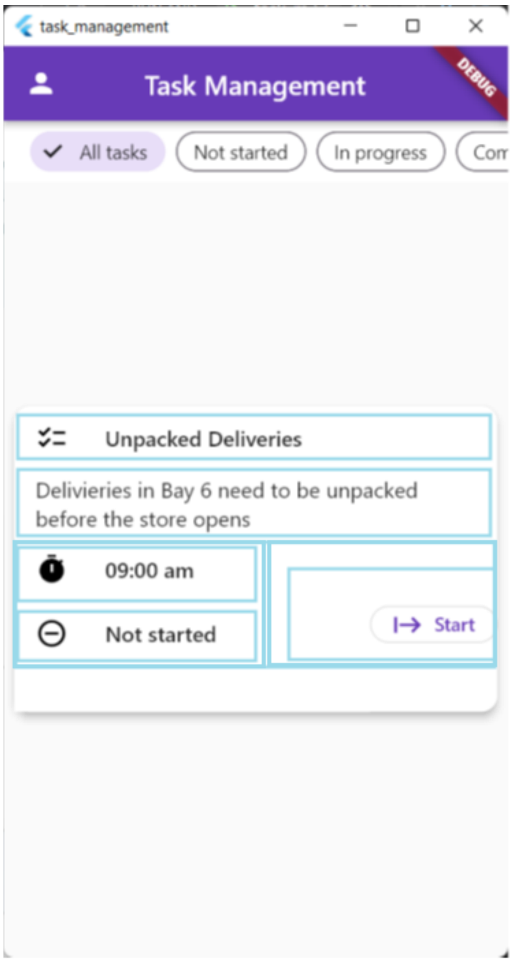
Task Management

Unpacked Deliveries

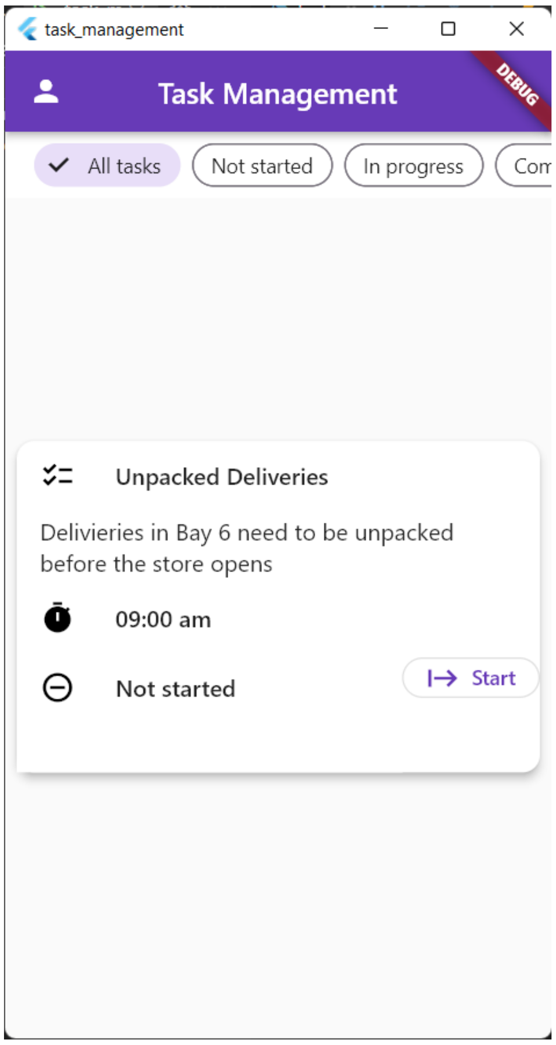
Delivieries in Bay 6 need to be unpacked before the store opens

Added task description as string (for now)

```
class TaskCard extends StatelessWidget {
  final Task task;

  const TaskCard(this.task, {Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    // NEW from here ...
    return Container(
      padding: EdgeInsets.all(8.0),
      height: 261,
      child: Align(
        alignment: Alignment.topCenter,
        child: Card(
          elevation: 5,
          margin: EdgeInsets.fromLTRB(0.0, 0.0, 0.0, 16.0),
          shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(12.0),
          ),
          child: Column(
            mainAxisSize: MainAxisSize.min,
            children: const [
              ListTile(
                title: Text(
                  'Unpacked Deliveries',
                  overflow: TextOverflow.ellipsis,
                  style: TextStyle(fontWeight: FontWeight.w500),
                ),
                leading: Icon(
                  Icons.checklist,
                  color: Colors.black,
                ),
              ),
              ListTile(
                title: Text(
                    'Delivieries in Bay 6 need to be unpacked before the store opens',
                    overflow: TextOverflow.visible),
              ),
            ],
          ),
        ),
      ),
    );
  }
}
```



Visual layout of rows and columns drawn on and main components of each row and column so I know how to set up the code


task_management

Task Management

All tasks | Not started | In progress | Com

Unpacked Deliveries

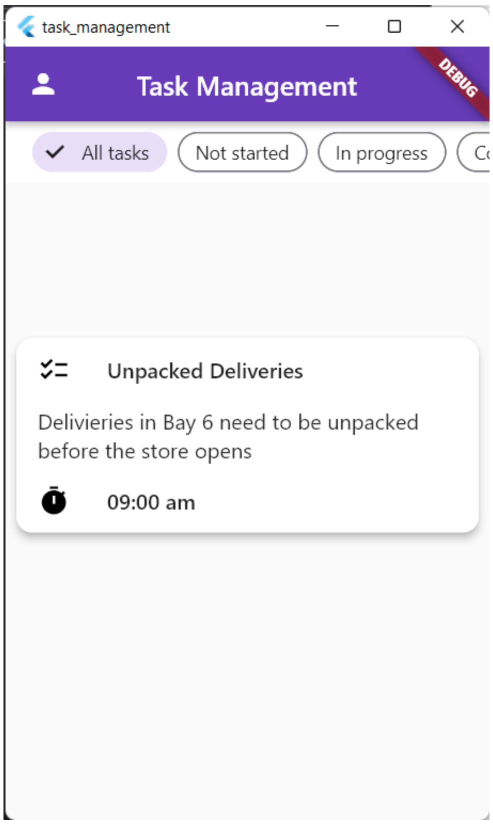
Delivieries in Bay 6 need to be unpacked before the store opens

09:00 am

Not started

Code to attempt to adjust formatting of objects inside card

```dart
class TaskCard extends StatelessWidget {
  final Task task;

  const TaskCard(this.task, {Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    // NEW from here ...
    return Container(
      padding: EdgeInsets.all(8.0),
      height: 261,
      child: Card(
        elevation: 5,
        margin: EdgeInsets.fromLTRB(0.0, 0.0, 0.0, 16.0),
        shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(12.0),
        ),
        child: Column(
          mainAxisSize: MainAxisSize.min,
          mainAxisAlignment: MainAxisAlignment.start,
          children: <Widget>[
            ListTile(
              title: Text(
                'Unpacked Deliveries',
                overflow: TextOverflow.ellipsis,
                style: TextStyle(fontWeight: FontWeight.w500),
              ),
              leading: Icon(
                Icons.checklist,
                color: Colors.black,
              ),
            ),
            ListTile(
              title: Text(
                  'Delivieries in Bay 6 need to be unpacked before the store opens',
                  overflow: TextOverflow.visible),
            ),
            // ignore: avoid_unnecessary_containers
            Expanded(
              child: Row(
                mainAxisSize: MainAxisSize.min,
                children: <Widget>[
                  Expanded(
                    child: Container(
                      color: Colors.white,
                      child: Column(
                        children: const <Widget>[
                          ListTile(
                            title: Text(
                              '09:00 am',
                              overflow: TextOverflow.ellipsis,
                              style: TextStyle(fontWeight: FontWeight.w500),
                            ),
                            leading: Icon(
                              Icons.timer,
                              color: Colors.black,
                            ),
                          ), // listtile
                          ListTile(
                            title: Text(
                              'Not started',
                              overflow: TextOverflow.ellipsis,
                              style: TextStyle(fontWeight: FontWeight.w500),
                            ),
                            leading: Icon(
                              Icons.remove_circle_outline,
                              color: Colors.black,
                            ),
                          ), // listtile
                        ], // widget
                      ), // column
                    ),
                  ), // expanded
                  OutlinedButton.icon(
                    onPressed: () {},
                    icon: Icon(
                      Icons.start,
                      size: 24.0,
                    ),
                    label: Text('Start'),
                    style: ButtonStyle(
                      shape: MaterialStateProperty.all<RoundedRectangleBorder>(
                        RoundedRectangleBorder(
                          borderRadius: BorderRadius.circular(18.0),
                          side: BorderSide(color: Colors.black),
                        ), // roundedRectangleBorder
                      ), // roundedRectangleBorder
                    ), // buttonstyle
                  ),
                ],
              ),
            ), // outlinedbutton
          ], // widget
        ), // column
      ), // card
    );
  }
}
```



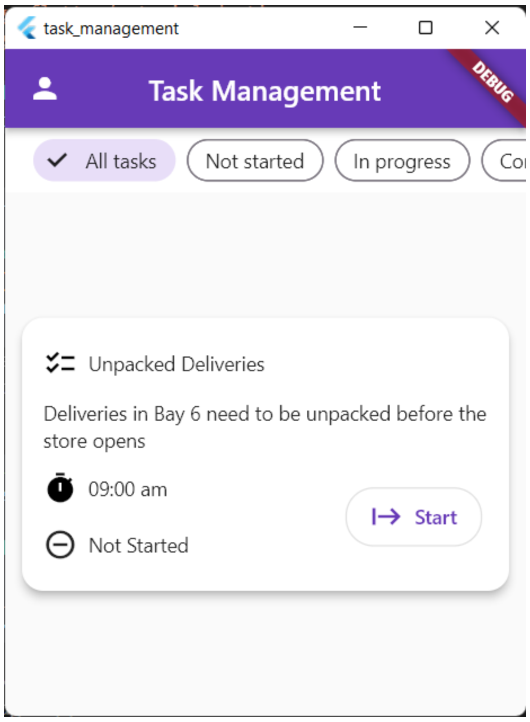
Unpacked Deliveries

Delivieries in Bay 6 need to be unpacked before the store opens

09:00 am

More attempts to fix formatting inside card

```
class TaskCard extends StatelessWidget {
  final Task task;

  const TaskCard(this.task, {Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    // NEW from here ...
    return Container(
      padding: EdgeInsets.all(8.0),
      height: 261,
      child: Align(
        alignment: Alignment.topCenter,
        child: Card(
          elevation: 5,
          margin: EdgeInsets.fromLTRB(0.0, 0.0, 0.0, 16.0),
          shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(12.0),
          ),
          child: Column(
            mainAxisSize: MainAxisSize.min,
            children: const [
              ListTile(
                title: Text(
                  'Unpacked Deliveries',
                  overflow: TextOverflow.ellipsis,
                  style: TextStyle(fontWeight: FontWeight.w500),
                ),
                leading: Icon(
                  Icons.checklist,
                  color: Colors.black,
                ),
              ),
              ListTile(
                title: Text(
                    'Delivieries in Bay 6 need to be unpacked before the store opens',
                    overflow: TextOverflow.visible),
              ),
              ListTile(
                title: Text(
                  '09:00 am',
                  overflow: TextOverflow.ellipsis,
                  style: TextStyle(fontWeight: FontWeight.w500),
                ),
                leading: Icon(
                  Icons.timer,
                  color: Colors.black,
                ),
              ),
            ],
          ),
        ),
      ),
    );
  }
}
```



Unpacked Deliveries

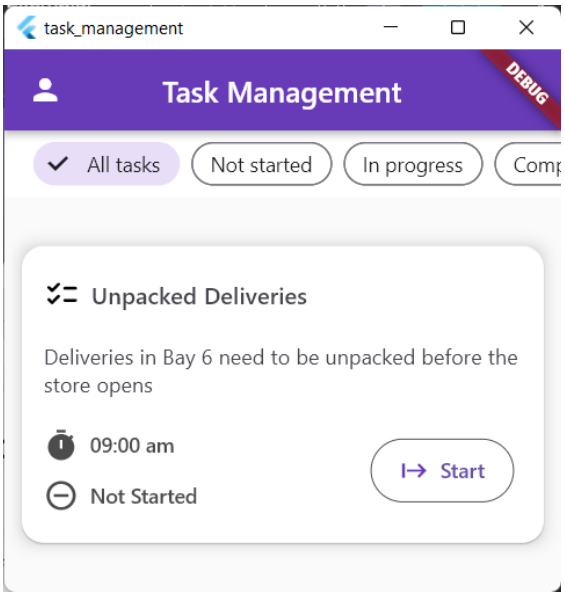
Deliveries in Bay 6 need to be unpacked before the store opens

09:00 am

Not Started

Closer to what figma images look like of the app

```
class TaskCard extends StatelessWidget {
  final Task task;

  // title
  final tasktitle = Row(
    children: const <Widget>[
      Icon(Icons.checklist, color: Colors.black),
      Padding(
        padding: EdgeInsets.all(8.0),
        child: Text('Unpacked Deliveries', overflow: TextOverflow.ellipsis),
      ),
    ],
  );

  // description
  final taskdescription = Row(
    children: const <Widget>[
      Expanded(
        child: Text(
          'Deliveries in Bay 6 need to be unpacked before the store opens',
          overflow: TextOverflow.visible,
        ),
      ),
    ],
  );

  // time, status, button
  final taskstatus = Row(
    children: <Widget>[
      Expanded(
        flex: 2,
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.center,
    children: <Widget>[
      // time
      Row(
        children: const <Widget>[
          Icon(Icons.timer, color: Colors.black),
          Padding(
            padding: EdgeInsets.all(8.0),
            child: Text(
              '09:00 am',
              overflow: TextOverflow.ellipsis,
            ),
          ),
        ],
      ),

      // blank divider
      SizedBox(height: 5),

      // status
      Row(
        children: const <Widget>[
          Icon(Icons.remove_circle_outline),
          Padding(
            padding: EdgeInsets.all(8.0),
            child: Text(
              'Not Started',
              overflow: TextOverflow.ellipsis,
            ),
          ),
        ],
      ),
    ],
  ),
),

// task button
Expanded(
  child: Column(
    children: [
      // task button
      OutlinedButton.icon(
        onPressed: () {},
        icon: Icon(Icons.start),
        style: OutlinedButton.styleFrom(
          padding: EdgeInsets.all(17),
          shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(20.0),
          ),
        ),
        label: Text(
          'Start',
          overflow: TextOverflow.ellipsis,
        ),
      ),
          ],
        ),
      ),
    ],
  );

  TaskCard(this.task, {Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    // NEW from here ...

    // main body of task card
    return Container(
      padding: EdgeInsets.all(8.0),
      child: Card(
        elevation: 5,
        shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(15.0),
        ),
        child: Container(
          padding: EdgeInsets.all(15),
          child: Column(
            mainAxisAlignment: MainAxisAlignment.spaceEvenly,
            mainAxisSize: MainAxisSize.min,
            children: [
              tasktitle,
              SizedBox(height: 8), // insert task title
              taskdescription,
              SizedBox(height: 8), // insert task description
              taskstatus, // insert time, condition, button
            ],
          ),
        ),
      ),
    );
  }
}
```



Unpacked Deliveries

Deliveries in Bay 6 need to be unpacked before the store opens

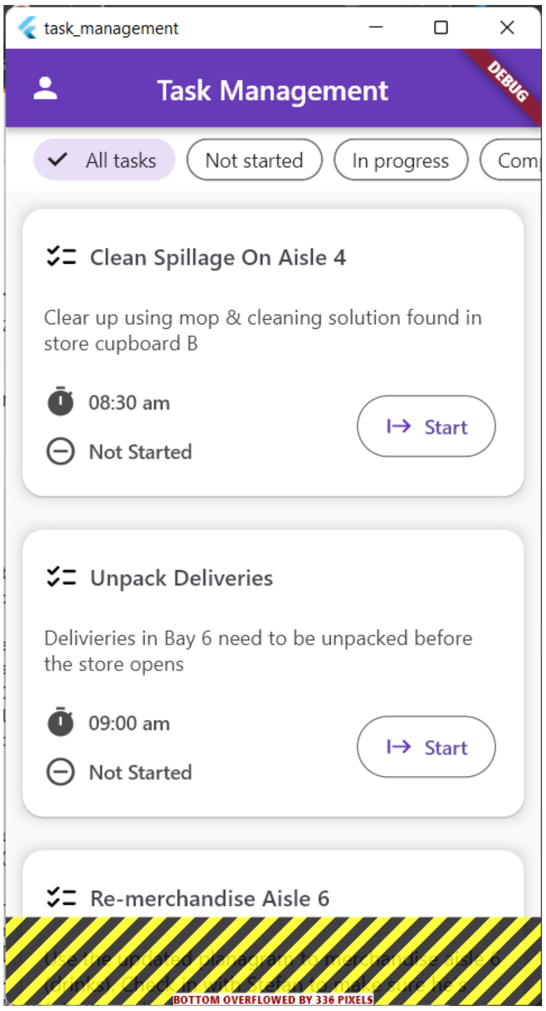
09:00 am

Not Started

Managed to fix object sizes and separate icons and button in last row of card

```
class TaskCard extends StatelessWidget {
  final Task task;

  const TaskCard(this.task, {Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    // ADD CODE BELOW

    // title font
    var titlefont = Theme.of(context).textTheme.titleMedium?.copyWith(
          color: Color.fromRGBO(73, 69, 79, 1),
          fontFamily: 'Roboto',
          fontWeight: FontWeight.w500,
          letterSpacing: 0.1,
          fontSize: 16,
        );

    // description font
    var descriptionfont = Theme.of(context).textTheme.bodyMedium?.copyWith(
          color: Color.fromRGBO(73, 69, 79, 1),
          fontFamily: 'Roboto',
          fontWeight: FontWeight.w400,
          letterSpacing: 0.25,
          fontSize: 14,
        );

    // status font
    var statusfont = Theme.of(context).textTheme.bodyMedium?.copyWith(
          color: Color.fromRGBO(76, 76, 76, 1),
          fontFamily: 'Roboto',
          fontWeight: FontWeight.w500,
          fontSize: 14,
        );

    // button font
    var buttonfont = Theme.of(context).textTheme.labelLarge?.copyWith(
          color: Color.fromRGBO(103, 80, 164, 1),
          fontSize: 14,
          fontFamily: 'Roboto',
          letterSpacing: 0.1,
        );

    // button style
    var statusbuttonstyle = OutlinedButton.styleFrom(
      padding: EdgeInsets.all(20),
      side: BorderSide(
          color: Color.fromRGBO(73, 69, 79, 1),
          width: 0.75,
          style: BorderStyle.solid),
      shape: RoundedRectangleBorder(
        borderRadius: BorderRadius.circular(40.0),
      ),
    );

    // TITLE
    final tasktitle = Row(
      children: <Widget>[
        Icon(
          Icons.checklist_rounded,
          color: Color.fromRGBO(0, 0, 0, 1),
          size: 25,
        ),
        Padding(
            padding: EdgeInsets.all(8.0),
            child: Text(
              'Unpacked Deliveries',
              overflow: TextOverflow.ellipsis,
              style: titlefont,
            )),
      ],
    );

    // DESCRIPTION
    final taskdescription = Row(
      children: <Widget>[
        Expanded(
          child: Text(
            'Deliveries in Bay 6 need to be unpacked before the store opens',
            overflow: TextOverflow.visible,
            style: descriptionfont,
          ),
        ),
      ],
    );

    // TIME, STATUS, BUTTON
    final taskstatus = Row(
      children: <Widget>[
        Expanded(
          flex: 2,
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.center,
            children: <Widget>[
              // time
              Row(
                children: <Widget>[
                  Icon(
                    Icons.timer,
                    color: Color.fromRGBO(76, 76, 76, 1),
                  ),
                  Padding(
                    padding: EdgeInsets.all(8.0),
                    child: Text(
                      '09:00 am',
                      overflow: TextOverflow.ellipsis,
                      style: statusfont,
                    ),
                  ),
                ],
              ),

              // status
              Row(
                children: <Widget>[
                  Icon(
                    Icons.remove_circle_outline,
                    color: Color.fromRGBO(76, 76, 76, 1),
                  ),
                  Padding(
                    padding: EdgeInsets.all(8.0),
                    child: Text(
                      'Not Started',
                      overflow: TextOverflow.ellipsis,
                      style: statusfont,
                    ),
                  ),
                ],
              ),
            ],
          ),
        ),

        // task button
        Expanded(
          child: Column(
            children: [
              // task button
              OutlinedButton.icon(
                onPressed: () {},
                icon: Icon(
                  Icons.start,
                  size: 20,
                ),
                style: statusbuttonstyle,
                label: Text(
                  'Start',
                  overflow: TextOverflow.ellipsis,
                  style: buttonfont,
                ),
              ),
            ],
          ),
        ),
      ],
    );

    // main body of task card
    return Container(
      padding: EdgeInsets.all(8.0),
      decoration: BoxDecoration(
        boxShadow: const [
          BoxShadow(
            color: Color.fromRGBO(0, 0, 0, 0.3),
            blurRadius: 10,
            spreadRadius: -15.0,
          ),
        ],
      ),
      child: Card(
        shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(15.0),
        ),
        child: Container(
          padding: EdgeInsets.all(15),
          child: Column(
            mainAxisAlignment: MainAxisAlignment.spaceEvenly,
            mainAxisSize: MainAxisSize.min,
            children: [
              tasktitle,
              SizedBox(height: 15),
              taskdescription,
              SizedBox(height: 15),
              taskstatus,
            ],
          ),
        ),
      ),
    );
  }
}
```

3. Using ListViews



Clean Spillage On Aisle 4

Clear up using mop & cleaning solution found in store cupboard B

08:30 am

Not Started

Unpack Deliveries

Delivieries in Bay 6 need to be unpacked before the store opens

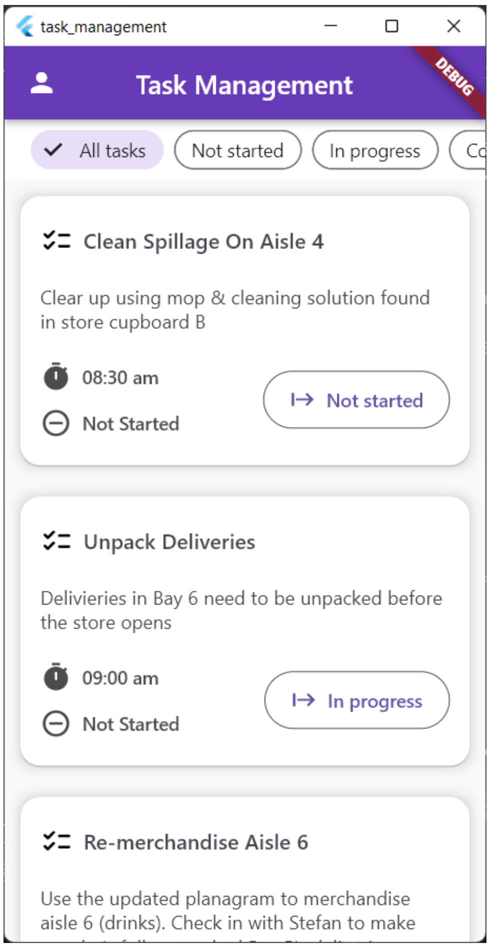
09:00 am

Not Started

Re-merchandise Aisle 6

Used `listView` to display all list of task cards replacing string text I had before and instead calling the list of `Tasks`

```dart
import 'package:flutter/material.dart';
import 'package:task_management/components/task_card.dart';
import 'package:task_management/data/task_model.dart';
import 'package:task_management/data/tasks.dart';

import '../components/filter_pills.dart';

class MyHomePage extends StatefulWidget {
  static const routeName = '/home';

  const MyHomePage({Key? key}) : super(key: key);

  @override
  State<MyHomePage> createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Task Management'),
        centerTitle: true,
        leading: const Icon(Icons.person),
      ),
      body: Column(
        mainAxisSize: MainAxisSize.min,
        children: [
          FilterPills(
            onSelected: (value) {
              // TODO: Add filter code here
            },
          ),

          // TODO: Later this needs to be replaced with a list of cards
          ListView.builder(
              scrollDirection: Axis.vertical,
              shrinkWrap: true,
              physics: const AlwaysScrollableScrollPhysics(),
              itemCount: taskList.length,
              prototypeItem: TaskCard(taskList[0]),
              itemBuilder: (context, index) {
                return Expanded(child: TaskCard(taskList[index]));
              }),
        ],
      ),
    );
  }
}
```


task_management

Task Management

All tasks | Not started | In progress | Co

Clean Spillage On Aisle 4

Clear up using mop & cleaning solution found in store cupboard B

08:30 am

Not Started

Not started

Unpack Deliveries

Delivieries in Bay 6 need to be unpacked before the store opens

09:00 am

Not Started

In progress

Re-merchandise Aisle 6

Use the updated planagram to merchandise aisle 6 (drinks). Check in with Stefan to make

Finally fixed overflow issues inside and outside of card

Just had to change prototype from `ListView` builder. `PrototypeItem` sizes all cards based on first widget which is why all the subsequent larger cards would overflow, because they took on the same size as the first card.

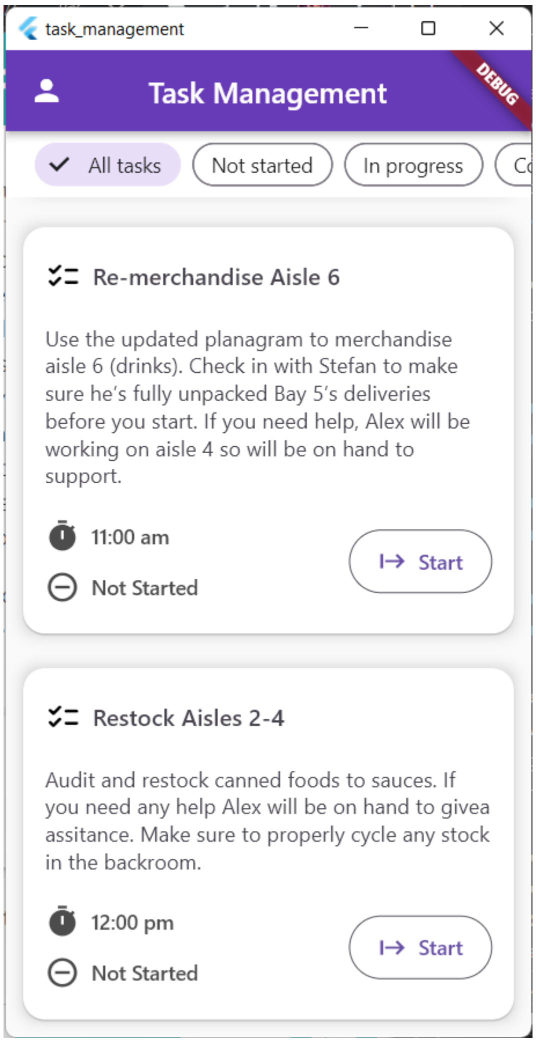
```
Expanded(
            child: ListView.builder(
                physics: const AlwaysScrollableScrollPhysics(),
                itemCount: taskList.length,
                prototypeItem: null,
                itemBuilder: (context, index) {
                  return TaskCard(taskList[index]);
                }),
          ),
```

```
// ignore_for_file: prefer_const_constructors

import 'package:flutter/material.dart';
import 'package:task_management/data/task_model.dart';

class TaskCard extends StatelessWidget {
  final Task task;

  const TaskCard(this.task, {Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    // ADD CODE BELOW

    // title font
    var titlefont = Theme.of(context).textTheme.titleMedium?.copyWith(
          color: Color.fromRGBO(73, 69, 79, 1),
          fontFamily: 'Roboto',
          fontWeight: FontWeight.w500,
          letterSpacing: 0.1,
          fontSize: 16,
        );

    // description font
    var descriptionfont = Theme.of(context).textTheme.bodyMedium?.copyWith(
          color: Color.fromRGBO(73, 69, 79, 1),
          fontFamily: 'Roboto',
          fontWeight: FontWeight.w400,
          letterSpacing: 0.25,
          fontSize: 14,
        );

    // status font
    var statusfont = Theme.of(context).textTheme.bodyMedium?.copyWith(
          color: Color.fromRGBO(76, 76, 76, 1),
          fontFamily: 'Roboto',
          fontWeight: FontWeight.w500,
          fontSize: 14,
        );

    // button font
    var buttonfont = Theme.of(context).textTheme.labelLarge?.copyWith(
```

```dart
      color: Color.fromRGBO(103, 80, 164, 1),
      fontSize: 14,
      fontFamily: 'Roboto',
      letterSpacing: 0.1,
    );

var buttoncolor = Color.fromRGBO(103, 80, 164, 1);

// button style
var statusbuttonstyle = OutlinedButton.styleFrom(
  foregroundColor: Color.fromRGBO(103, 80, 164, 1),
  backgroundColor: Color.fromARGB(255, 255, 255, 255),
  padding: EdgeInsets.all(20),
  side: BorderSide(
      color: Color.fromRGBO(73, 69, 79, 1),
      width: 0.75,
      style: BorderStyle.solid),
  shape: RoundedRectangleBorder(
    borderRadius: BorderRadius.circular(40.0),
  ),
);

// TITLE
final tasktitle = Row(
  children: <Widget>[
    Icon(
      Icons.checklist_rounded,
      color: Color.fromRGBO(0, 0, 0, 1),
      size: 25,
    ),
    Padding(
        padding: EdgeInsets.all(8.0),
        child: Text(
          task.title,
          overflow: TextOverflow.ellipsis,
          style: titlefont,
        )),
  ],
);

// task description
final taskdescription = Row(
  children: <Widget>[
    Expanded(
      child: Text(
        task.description,
        overflow: TextOverflow.visible,
        style: descriptionfont,
        maxLines: 6,
      ),
    ),
  ],
);
```

```
    // time, status, button
    final taskstatus = Row(
      children: <Widget>[
        Expanded(
          flex: 2,
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.center,
            children: <Widget>[
              // time with icon i.e. '09:00 am'
              Row(
                children: <Widget>[
                  Icon(
                    Icons.timer,
                    color: Color.fromRGBO(76, 76, 76, 1),
                  ),
                  Padding(
                    padding: EdgeInsets.all(8.0),
                    child: Text(
                      task.completeBy,
                      overflow: TextOverflow.visible,
                      style: statusfont,
                    ),
                  ),
                ],
              ),

              // status with icon i.e. 'Not Started'
              Row(
                children: <Widget>[
                  Icon(
                    Icons.remove_circle_outline,
                    color: Color.fromRGBO(76, 76, 76, 1),
                  ),
                  Padding(
                    padding: EdgeInsets.all(8.0),
                    child: Text(
                      'Not Started',
                      overflow: TextOverflow.visible,
                      style: statusfont,
                    ),
                  ),
                ],
              ),
            ],
          ),
        ),

        // button i.e. 'Started'
        Column(
          children: [
            OutlinedButton.icon(
              icon: Icon(
                Icons.start,
                size: 20,
              ),
              style: statusbuttonstyle,
              label: Text(
                task.getCompletionStatusText(),
                overflow: TextOverflow.ellipsis,
                style: buttonfont,
              ),
              onPressed: () {},
            ),
          ],
        ),
      ],
    );

    // task card
    return Container(
      padding: EdgeInsets.all(8.0),
      decoration: BoxDecoration(
        boxShadow: const [
          BoxShadow(
            color: Color.fromRGBO(0, 0, 0, 0.3),
            blurRadius: 10,
            spreadRadius: -15.0,
          ),
        ],
      ),
      child: Card(
        shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(15.0),
        ),
        child: Container(
          padding: EdgeInsets.all(15),
          child: Column(
            mainAxisAlignment: MainAxisAlignment.spaceEvenly,
            mainAxisSize: MainAxisSize.min,
            children: [
              tasktitle,
              SizedBox(height: 15),
              taskdescription, // task description
              SizedBox(height: 15),
              taskstatus, // time, status, button
            ],
          ),
        ),
      ),
    );
  }
}
```

4. Filtering


task_management

Task Management

DEBUG

All tasks | Not started | In progress | Co

Re-merchandise Aisle 6

Use the updated planagram to merchandise aisle 6 (drinks). Check in with Stefan to make sure he's fully unpacked Bay 5's deliveries before you start. If you need help, Alex will be working on aisle 4 so will be on hand to support.

11:00 am

Not Started

Start

Restock Aisles 2-4

Audit and restock canned foods to sauces. If you need any help Alex will be on hand to givea assitance. Make sure to properly cycle any stock in the backroom.

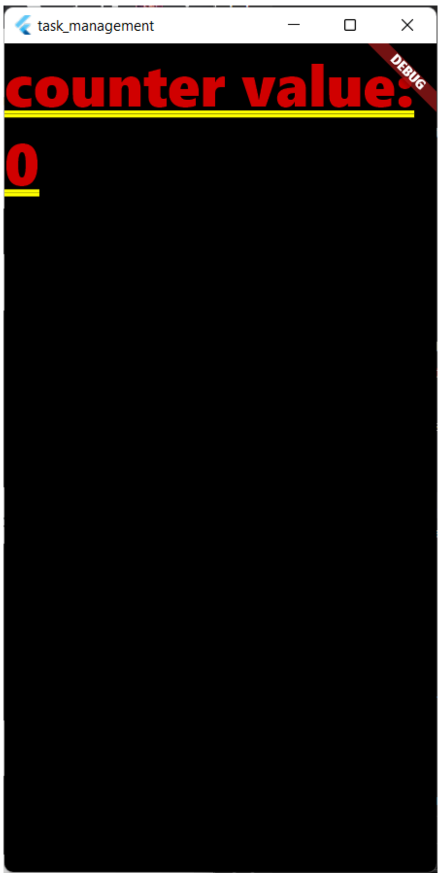
12:00 pm

Not Started

Start

Attempted to create filter for filter pills

Tried counter demo linked to see if it would help.

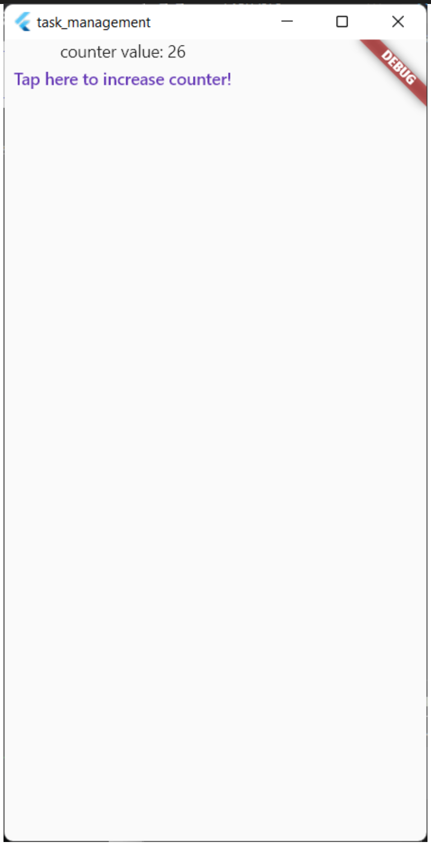


counter value:

0

Counter demo app (link: https://dev.to/nicks101/when-to-use-setstate-in-flutter-380)

```dart
import 'package:flutter/material.dart';
import 'package:task_management/components/task_card.dart';
import 'package:task_management/data/task_model.dart';
import 'package:task_management/data/tasks.dart';

import '../components/filter_pills.dart';

class MyHomePage extends StatefulWidget {
  static const routeName = '/home';

  const MyHomePage({Key? key}) : super(key: key);

  @override
  State<MyHomePage> createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  @override
  void setState(VoidCallback fn) {}

  int counter = 0;

  @override
  Widget build(BuildContext context) {
    /*
    return Scaffold(
      appBar: AppBar(
        title: const Text('Task Management'),
        centerTitle: true,
        leading: const Icon(Icons.person),
      ),
      body: Column(
        mainAxisSize: MainAxisSize.min,
        children: [
          FilterPills(
            onSelected: (value) {
              // TODO: Add filter code here
            },
          ),
          Expanded(
            child: ListView.builder(
                physics: const AlwaysScrollableScrollPhysics(),
                itemCount: taskList.length,
                prototypeItem: null,
                itemBuilder: (context, index) {
                  return TaskCard(taskList[index]);
                }),
          ),
        ],
      ),
    );
    */

    // counter
    return Text('counter value: $counter');
  }
}
```



counter value: 26
Tap here to increase counter!

Implemented `setState` *and UI change in counter demo app*

```
import 'package:flutter/material.dart';
import 'package:task_management/components/task_card.dart';
```

```dart
import 'package:task_management/data/task_model.dart';
import 'package:task_management/data/tasks.dart';

import '../components/filter_pills.dart';

class MyHomePage extends StatefulWidget {
  static const routeName = '/home';

  const MyHomePage({Key? key}) : super(key: key);

  @override
  State<MyHomePage> createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  int counter = 0;

  @override
  Widget build(BuildContext context) {
    /*
    return Scaffold(
      appBar: AppBar(
        title: const Text('Task Management'),
        centerTitle: true,
        leading: const Icon(Icons.person),
      ),
      body: Column(
        mainAxisSize: MainAxisSize.min,
        children: [
          FilterPills(
            onSelected: (value) {
              // TODO: Add filter code here
            },
          ),
          Expanded(
            child: ListView.builder(
                physics: const AlwaysScrollableScrollPhysics(),
                itemCount: taskList.length,
                prototypeItem: null,
                itemBuilder: (context, index) {
                  return TaskCard(taskList[index]);
                }),
          ),
        ],
      ),
    );
    */

    // counter
    return Scaffold(
      body: Column(
        children: [
          Text('counter value: $counter'),
          TextButton(
            onPressed: () {
              counter++; // increases counter by +1
              setState(() {
                // updates the UI upon increase of counter by +1
              });
            },
            child: const Text('Tap here to increase counter!'),
          ),
        ],
      ),
    );
  }
}
```

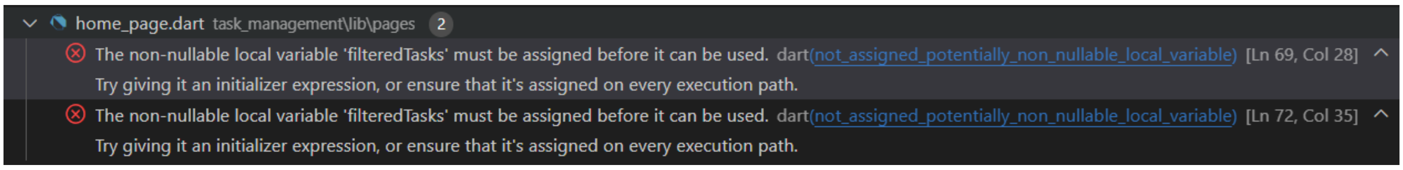
```
class _MyHomePageState extends State<MyHomePage> {
  String filtertab = 'All tasks';
```

Not sure how to do the same for task management app e.g. not sure how to link string from `FilterPills()` to the filter_pills.dart file and how and where to set variables used for filtering so they can be called upon inside and outside of e.g. functions.

I have watched and read through all the links and I do not know which parts of those examples and demos I can implement into this particular app. Below is what I have so far.

```
import 'package:flutter/material.dart';
import 'package:task_management/components/task_card.dart';
import 'package:task_management/data/task_model.dart';
import 'package:task_management/data/tasks.dart';

import '../components/filter_pills.dart';

class MyHomePage extends StatefulWidget {
  static const routeName = '/home';

  const MyHomePage({Key? key}) : super(key: key);

  @override
  State<MyHomePage> createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  // empty string for filterPills value input
  var selected = '';

  @override
  Widget build(BuildContext context) {
    // variable for full list of tasks
    var filteredList = taskList; // NEW
    List<Task> filteredTasks;

    return Scaffold(
      appBar: AppBar(
        title: const Text('Task Management'),
        centerTitle: true,
        leading: const Icon(Icons.person),
      ),
      body: Column(
        mainAxisSize: MainAxisSize.min,
        children: [
          // 1) tasks card filters
          FilterPills(onSelected: (value) {
            selected = 'All tasks';

            if (selected != 'All tasks') selected = value;

            // get enum type of value input
            testing(String value) {
              switch (value) {
                case 'Not started':
                  return CompletionStatus.notStarted;
                case 'In progress':
                  return CompletionStatus.inProgress;
                case 'Completed':
                  return CompletionStatus.completed;
              }
            }

            // keep enum type of value input
            var category = testing(value);

            // change list variable to include only filter cards of specific enum type
            filteredTasks =
                filteredList.where((task) => task.status == category).toList();

            // refresh UI to match change
            setState(() {});
          }),

          // 2) display task cards
          Expanded(
            child: ListView.builder(
                physics: const AlwaysScrollableScrollPhysics(),
                itemCount: filteredTasks.length,
                prototypeItem: null,
                itemBuilder: (context, index) {
                  return TaskCard(filteredTasks[index]);
                }),
          ),
        ],
      ),
    );
  }
}
```

which returns the following errors.



```
home_page.dart task_management\lib\pages 2
The non-nullable local variable 'filteredTasks' must be assigned before it can be used. dart(not_assigned_potentially_non_nullable_local_variable) [Ln 69, Col 28]
Try giving it an initializer expression, or ensure that it's assigned on every execution path.
The non-nullable local variable 'filteredTasks' must be assigned before it can be used. dart(not_assigned_potentially_non_nullable_local_variable) [Ln 72, Col 35]
Try giving it an initializer expression, or ensure that it's assigned on every execution path.
```

Error for filtering attempt