WEEK 3 -PRACTICE

UI Architecture - Theme, Widgets, Screens

Learning objectives

- ✓ **Structure Flutter widgets** for extendibility and consistency
- ✓ Comply with **coding conventions**
- ✓ Create a library of **generic widgets** aligned with a **design system**
- ✓ Understand the dart concepts of **static methods**, **attributes** and **part of** syntax

How to submit?

- ✓ Make sure you follow the COMMIT standards (see below)
- ✓ Once finished, submit to MS Team:
  - o GitHub URL
  - o This document

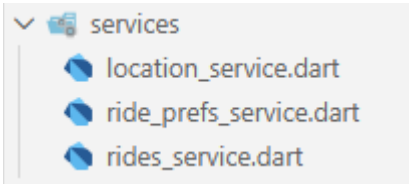
You are **not** allowed to **use AI** to submit this work

This practice focusses more **on clean code structure** rather than Flutter technical skills. We will evaluate how well **you follow the coding conventions**, how your name your class, variables etc.

BLA-001 – Rides Service

Complete the **service layer** and **test it**

The service layer – so far – is composed of 3 services:



All services are static - for now - and access to fake data (dummy data).

Complete the rides service to filter the rides starting from given departure location

```
List<Ride> filterByDeparture(Location departure)
```

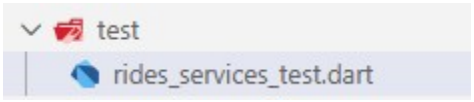
Complete the rides service to filter the rides for the given requested seat number

```
List<Ride> filterBySeatRequested(int seatRequested)
```

Complete the rides service to filter with several optional criteria *(flexible filter options)*

```
List<Ride> filterBy({Location departure, int seatRequested})
```

We use the /test folder to test our service and other layers :



Test your rides service filter methods using mock data: create a main() in the test folder :

```
RidesService.filterBy(
        departure: Location(name: "Dijon", country: Country.france),
        seatsRequested:2
);      // Shall return 1 ride
```

Example of test

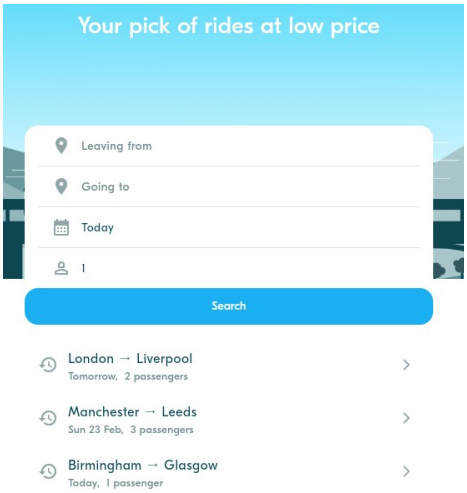
TODAY...

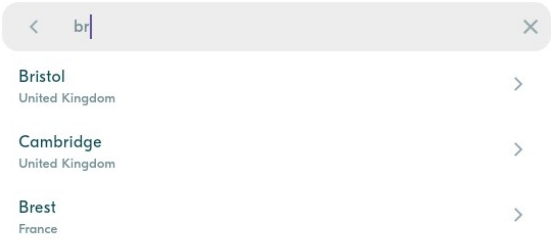
For this practice, you need to implement the **Rides Preferences** screen

- To input the locations, date and seats and press on Search
- Or to click on a past entered **Ride Preference**

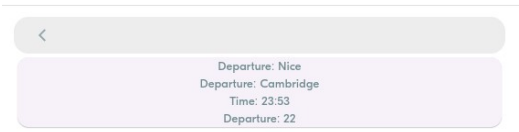
You also need to implement the related input modals:

- Location Picker
- Seat picker
- Date chooser





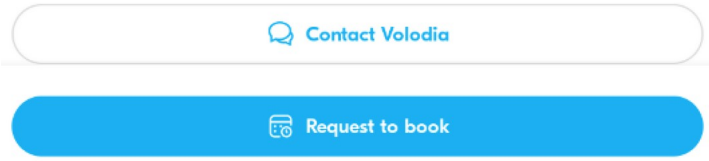
- Once selected, we should navigate to the **Rides screen**, displaying all rides matching preferences (fake view for now):



The Rides screen is just a fake screen for now

BLA-002 – Implement BlaButton

The BlaButton is used in many places in the application.



The BlaButton widget shall handle primary and secondary, without or with icons…

Q1 – First identify the **possible variations** of this button and complete the table

Widget	Screen / Screen Widget /App Widget	Parameters	Callbacks
BlaButton	App Widget	Label Icon Color	onTap

Q2 – Then implement the button and **test** all its variations using a **test screen**.

Q3 – Once validated, commit the code with the proper commit message.

BLA-002 – Implement BlaButton

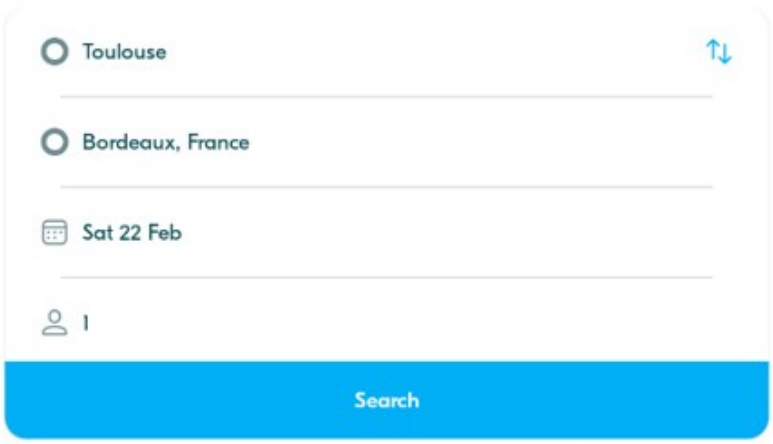
BLA-003 – Implement Form

The ride preferences from can be used either in the **Ride Preferences** screen (*as a screen component)* or in the **Ride screen** (*as a top modal dialog*)

⚠ Its s important to ensure this component **can be used on both screens**

⚠ You need to code the **component exactly as in the real app**.

What are the possible colors in the input fields ? The mandatory and optional icons? Actions?



Q1 – What are the widget parameters? What are the default values?

The widget parameters is final RidePref? InitRidePref. The defualt value is date is initialed today and selecte person is 1.
code:

```
departure = null;
arrival = null;
departureDate = DateTime.now();
requestedSeats = 1;
```

Q2 – What is the condition to validate the Search button? What type of data is popped if valid?

The condition to validate the Search button is check if the like this condition in the code:

```
if (departure != null && arrival != null) {
// Valid - proceed with search
}
```

The Type of data is popped

```
final newRidePref = RidePref(
departure: departure!,
```

```
arrival: arrival!,
departureDate: departureDate,
requestedSeats: requestedSeats,
);
```

Q3 – What are the **form sub widgets** and **app widget** you plan to use for this form?

Widget	Screen / Screen Widget /App Widget	Parameters	Callbacks
BlaButton	App Widget	Icon? Icon; String title; Color color;	VoidCallBack onTap
_BuildFormRow	App Widget(sub widget)	required IconData icon, required String title, Widget? trailing, bool isMandatory = false, bool isFilled = true,	VoidCallback onPressed

Q5 – How to implement the **switch location action** () in a clean way?

This is the condition how to handle with switch location:

I implement it in the setState :

```
final temp = departure;
departure = arrival;
arrival = temp;
```

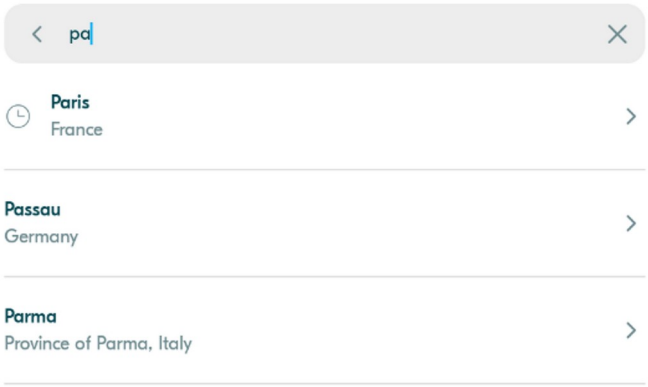
Q6 – Implement the widget and **test** all use cases.

```
void main() {
testWidgets('RidePrefForm renders and interacts', (WidgetTester tester) async {
// Build the widget
await tester.pumpWidget(
const MaterialApp(
home: Scaffold(
body: RidePrefForm(),
),
),
);
```

```dart
// Check that all form fields are present
expect(find.text('Leaving from'), findsOneWidget);
expect(find.text('Going to'), findsOneWidget);
expect(find.byIcon(Icons.calendar_month_outlined), findsOneWidget);
expect(find.byIcon(Icons.person_outline), findsOneWidget);
expect(find.text('Search'), findsOneWidget);

// Tap on the Search button (should not pop since fields are empty)
await tester.tap(find.text('Search'));
await tester.pump();

// Tap on the departure field (should navigate to placeholder)
await tester.tap(find.text('Leaving from'));
await tester.pumpAndSettle();

// Go back to the form
await tester.pageBack();
await tester.pumpAndSettle();

// Tap on the arrival field (should navigate to placeholder)
await tester.tap(find.text('Going to'));
await tester.pumpAndSettle();

// Go back to the form
await tester.pageBack();
await tester.pumpAndSettle();

// Tap on the date field (should open date picker)
await tester.tap(find.byIcon(Icons.calendar_month_outlined));
await tester.pumpAndSettle();

// Tap on the seats field (should open bottom sheet)
await tester.tap(find.byIcon(Icons.person_outline));
await tester.pumpAndSettle();
});
}
```

BLA-004 – Implement the Location Picker

The location picker will be used in many views (as a passenger as well as a driver).



Q1 – Analyze the real app picker behavior

How many actions can be done? When the list of locations is displayed? Etc.

There are 4 actions base on what I analyze:

1. Live search - Type to filter (uses onChange callback)
2. Select location - Tap from filtered/full list to select
3. Clear search - X button appears when search is active
4. Navigate back - Back arrow to cancel selection

The list of location is displayed when the the search input is not empty and the pattern search can math any case in the dummy data. (used contains() method)

Q2 – What are the **picker sub widgets** and **app widget** you plan to use for this form?

Widget	Screen / Screen Widget /App Widget	Parameters	Callbacks
LocationPickerScreen	Screen Widget		

Q3 – Implement the widget and **test** all use cases.