

# Navigation and Routing

# Basic Navigations

Part 1 - Introduction

Jumail Bin Taliba

School of Computing, UTM  
April 2020

# Agenda

- Introduction to Navigation and Routes
- Navigating to another screen
- Navigating back to the previous screen
- Passing data between screens

# Introduction

## Navigation:

- Move between screens

## Routes

- Screens or Pages

# Navigations in Flutter are handled in Stack

**Push**

Show screens



# Navigations in Flutter are handled in Stack

**Pop**

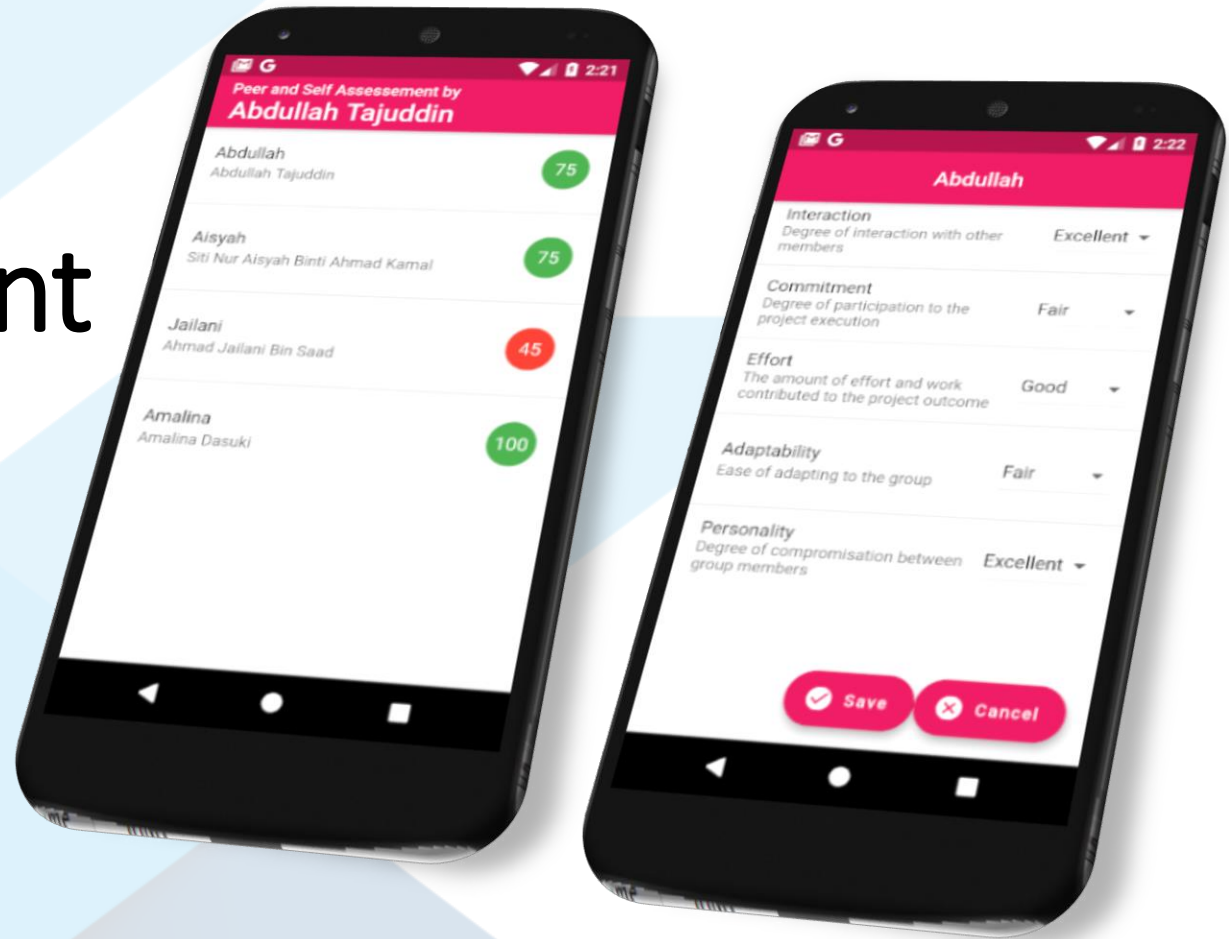
Close screens

Last In First Out (LIFO)

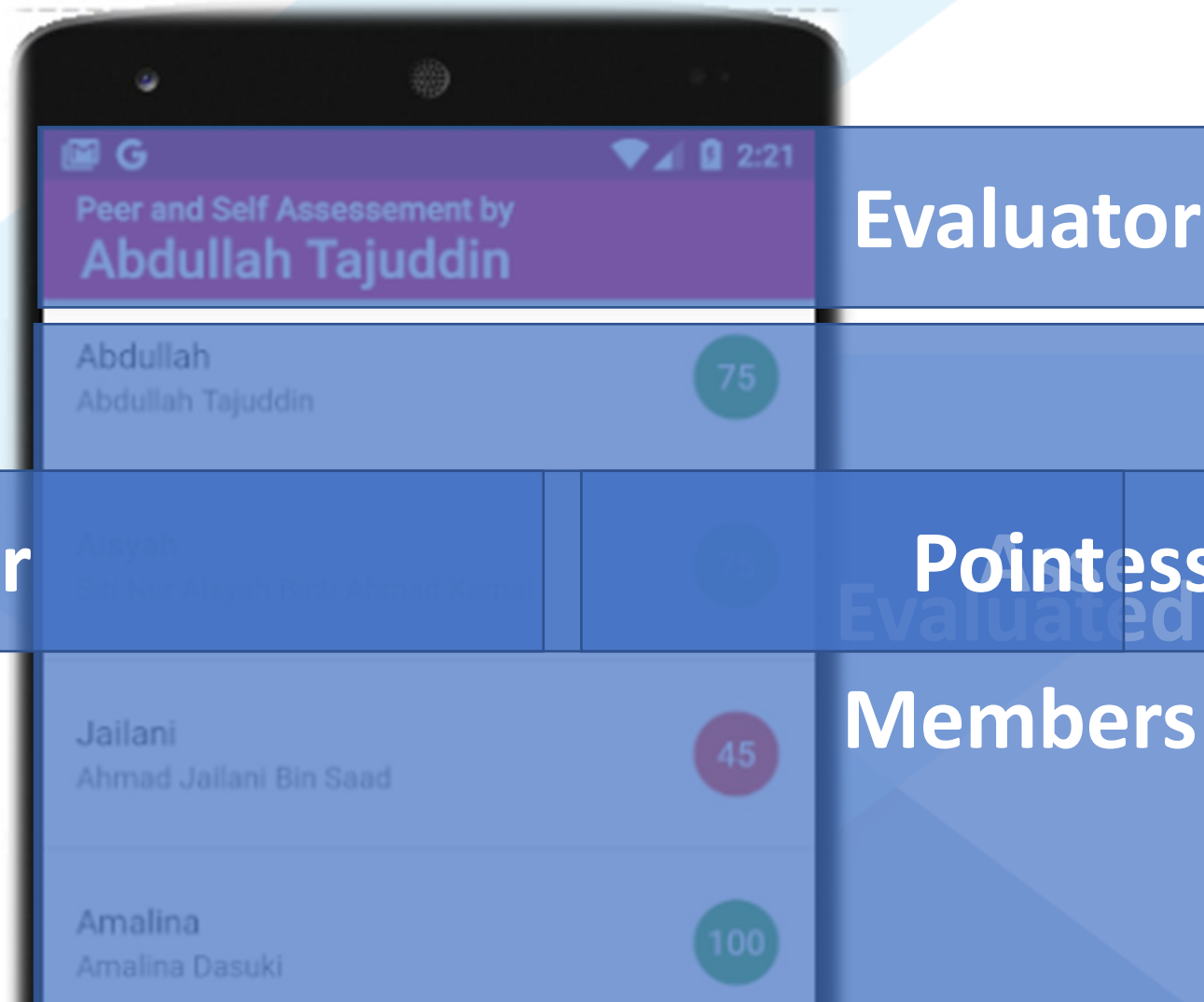


# Demo App

## Peer and Self Assessment



# About the app



Evaluator

Group member

Pointessment

Members

# About the app

Abdullah

Interaction  
Degree of interaction with other members

Excellent

Commitment  
Degree of participation to the project execution

Fair

Effort  
The amount of effort and work contributed to the project outcome

Good

Adaptability  
Ease of adapting to the group

Fair

Personality  
Degree of compromisation between group members

Excellent

Member being evaluated

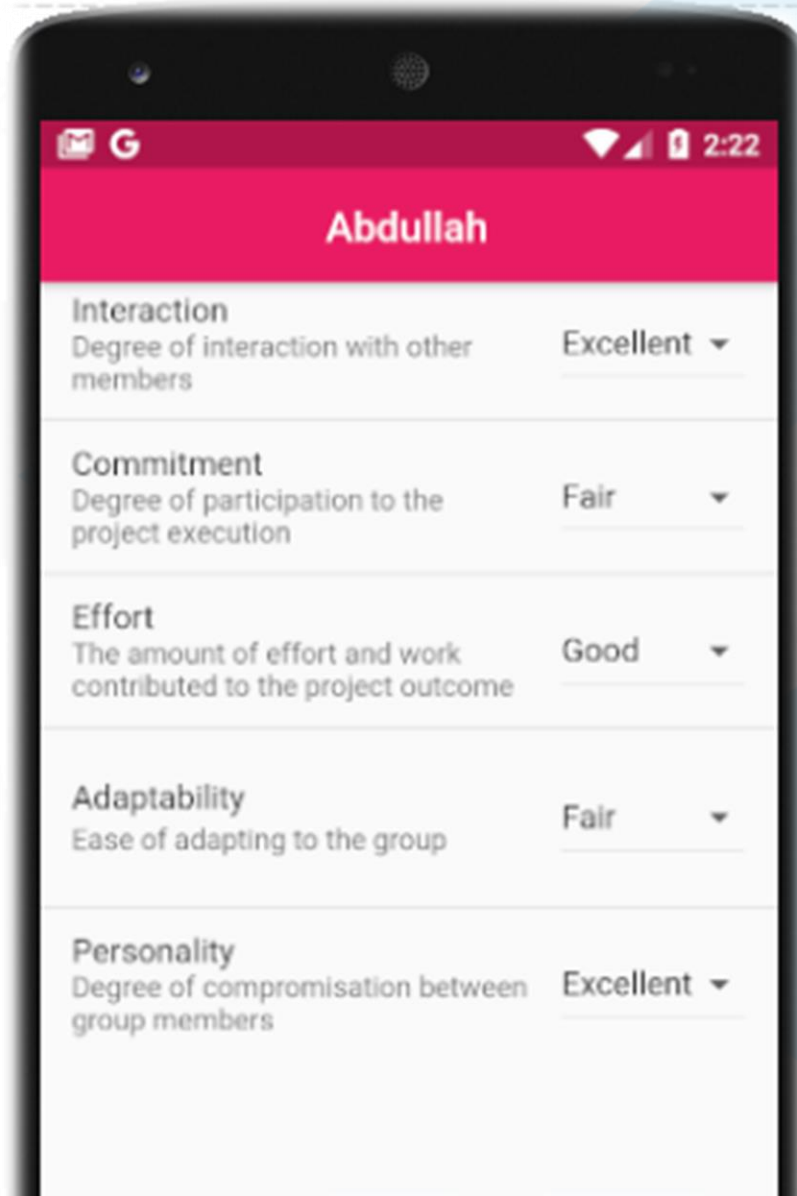
Scale

Value	Scale description
4	Excellent
3	Good
2	Fair
1	Poor
0	Not at all

Criteria



# About the app

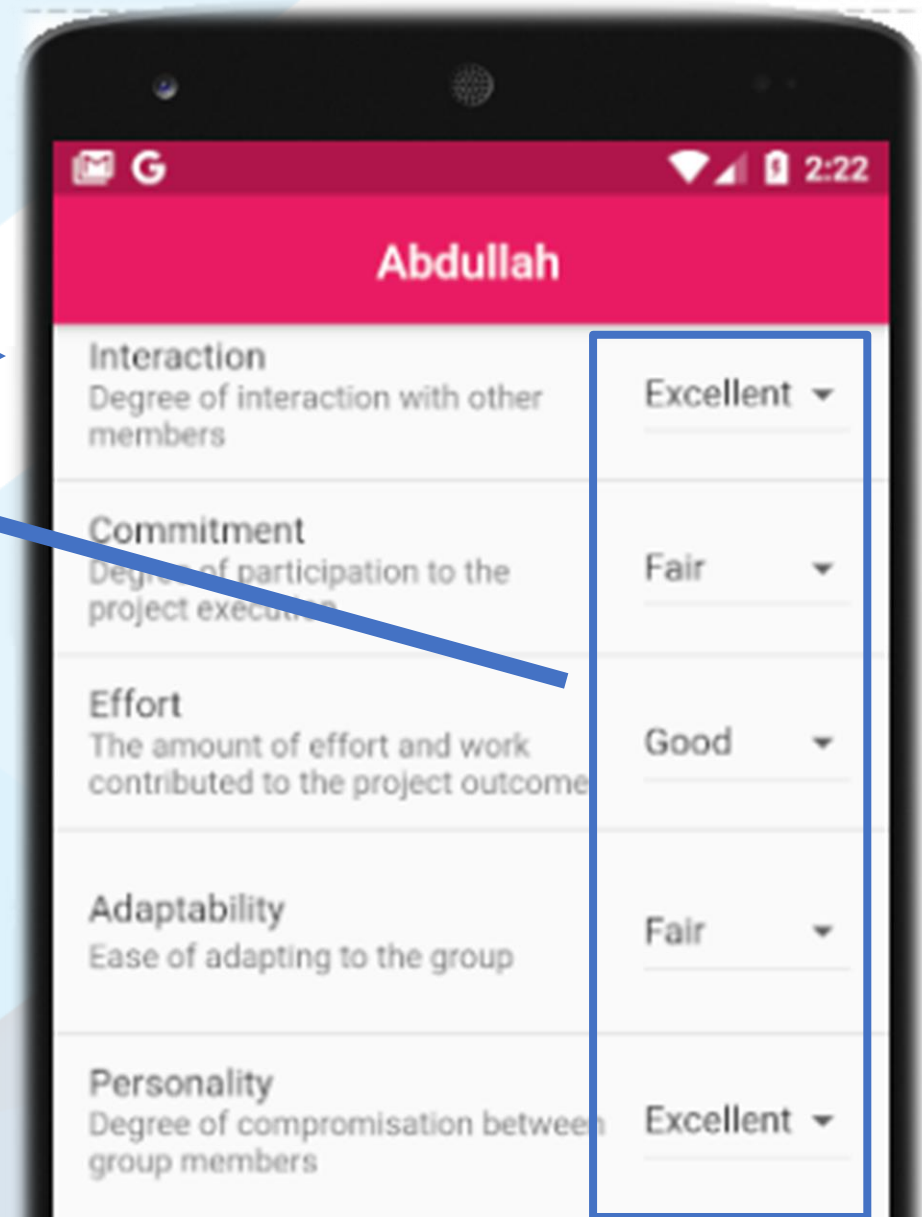
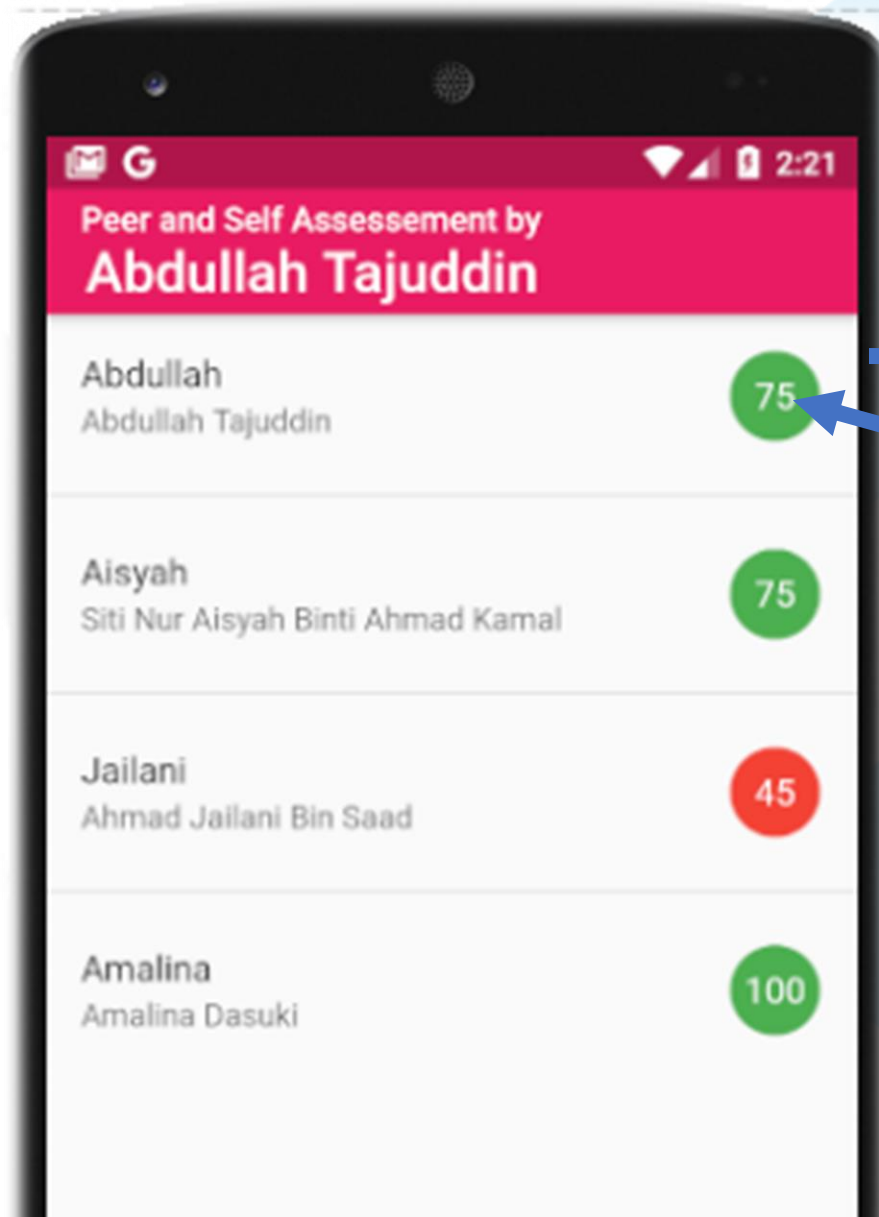


## Overall Performance Example Calculation

Criterion	Scale Grade	Scale Point
Interaction	Excellent	4
Commitment	Fair	2
Effort	Good	3
Adaptability	Fair	2
Personality	Excellent	4
Total		15
Percentage		$(15 / 20) \times 100 =$ <b>75%</b>

# Summary (main) screen

# Details screen



# Code

[https://github.com/jumail-utm/navigation\\_simple](https://github.com/jumail-utm/navigation_simple)

Attendance

# SINGLE ANSWER

# Program's file structure

```
[navigation_simple]
|
+---[lib]
|
|   + ---main.dart
|   |
|   + ---[models]
|   |   + ---assessment.dart
|   |   + ---group_member.dart
|   |   + ---mock_data.dart
|   |   + ---form.dart
|   |
|   + ---[screens]
|       + ---summary.dart
|       + ---details.dart
```



Remaining parts of this lesson  
will be available soon ....

# Navigation and Routing

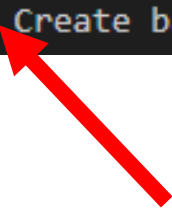
# Basic Navigations

Part 2 - Demo

Jumail Bin Taliba  
School of Computing, UTM  
April 2020

# Getting started with part 2

```
$ git log --oneline
df7c991 (HEAD -> master, origin/master) Disable AppBar and phone back buttons in DetailsScreen
d71ac69 Fix cancel feature problem on DetailsScreen by sending a copy data
78ea142 Add copy constructor to model classes (Assesment and GroupMember)- to achieve passing objects by value
d9b9086 Add cancel feature on DetailsScreen. But got error, data is updated even the cancel button is pressed
bd04606 Catch return data from sent via pop-refactor code to async/await code
16f1f91 Return data from DetailsScreen via pop parameter-example how it works
cf0c459 Back from DetailsScreen to SummaryScreen programatically (via button)
15a040a Open DetailsScreen from SummaryScreen (with passed data)
4868d1f Update SummaryScreen - dynamic build
c1e97dc Update DetailsScreen - dynamic build
7785365 Define model classes and create mock data
77187c0 Add DetailsScreen with hard coded data
cb5d6ea Add SummaryScreen with hard coded data
51ed06e Create base dir structure and empty files for model classes and screens
```



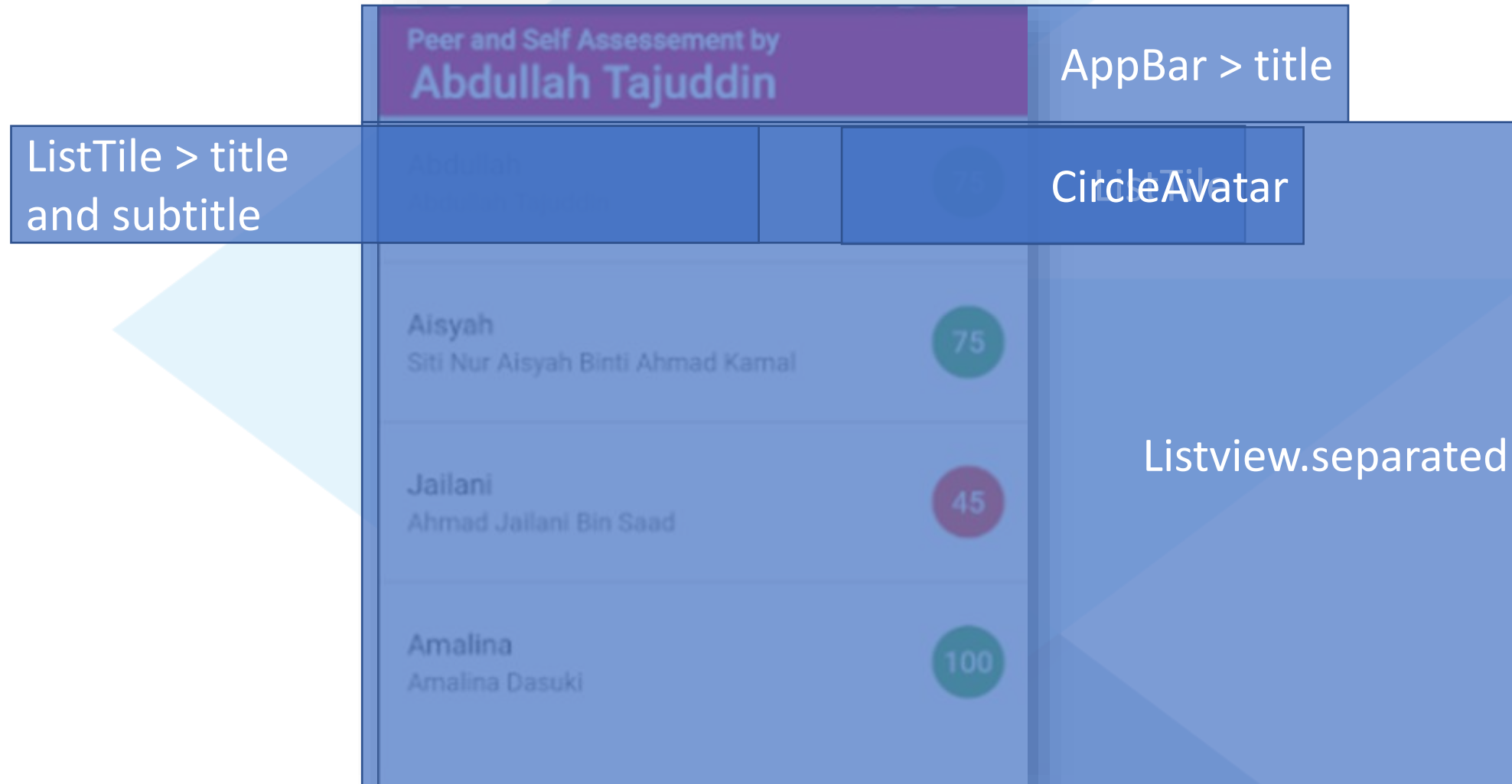
# Program's file structure

```
[navigation_simple]
|
+---[lib]
|
|   + ---main.dart
|   |
|   + ---[models]
|   |   + ---assessment.dart
|   |   + ---group_member.dart
|   |   + ---mock_data.dart
|   |   + ---form.dart
|   |
|   + ---[screens]
|       + ---summary.dart
|       + ---details.dart
```



## Task 1

# Create the screen skeletons



Abdullah	
<b>Interaction</b> Degree of interaction with other members	Excellent ▼
<b>Commitment</b> Degree of participation to the project execution	Fair ▼
<b>Effort</b> The amount of effort and work contributed to the project outcome	Good ▼
<b>Adaptability</b> Ease of adapting to the group	Fair ▼
<b>Personality</b> Degree of compromisation between group members	Excellent ▼

DropDownButton

Task 2

# Define model classes and mock data

class

GroupMember

Peer and Self Assesement by  
Abdullah Tajuddin

Abdullah

75

Aisyah

Siti Nur Aisyah Binti Ahmad Kamal

75

Jailani

Ahmad Jailani Bin Saad

45

Amalina

Amalina Dasuki

100

class

Assessment

list

mockData

class

Criterion

Abdullah

Interaction

Degree of interaction with other members

Excellent

Commitment

Degree of participation to the project execution

Fair

Effort

The amount of effort and work contributed to the project outcome

Good

Adaptability

Ease of adapting to the group

Fair

Personality

Degree of compromisation between group members

Excellent

class

Scale

list

criteria

## Task 3

# Build the screens with dynamic content

mockData  
(mock\_data.dart)

Peer and Self Assesement by Abdullah Tajuddin		
Abdullah	Abdullah Tajuddin	75
Aisyah	Siti Nur Aisyah Binti Ahmad Kamal	75
Jailani	Ahmad Jailani Bin Saad	45
Amalina	Amalina Dasuki	100

Abdullah	
Interaction Degree of interaction with other members	Excellent ▾
Commitment Degree of participation to the project execution	Fair ▾
Effort The amount of effort and work contributed to the project outcome	Good ▾
Adaptability Ease of adapting to the group	Fair ▾
Personality Degree of compromisation between group members	Excellent ▾

criteria  
scales  
(form.dart)

## Task 4

# Navigate and pass data to the second screen

Peer and Self Assessment by Abdullah Tajuddin		
Abdullah	Abdullah Tajuddin	75
Aisyah	Siti Nur Aisyah Binti Ahmad Kamal	75
Jailani	Ahmad Jailani Bin Saad	45
Amalina	Amalina Dasuki	100

Abdullah		
Interaction	Degree of interaction with other members	Excellent ▾
Commitment	Degree of participation to the project execution	Fair ▾
Effort	The amount of effort and work contributed to the project outcome	Good ▾
Adaptability	Ease of adapting to the group	Fair ▾
Personality	Degree of compromisation between group members	Excellent ▾

## Task 5

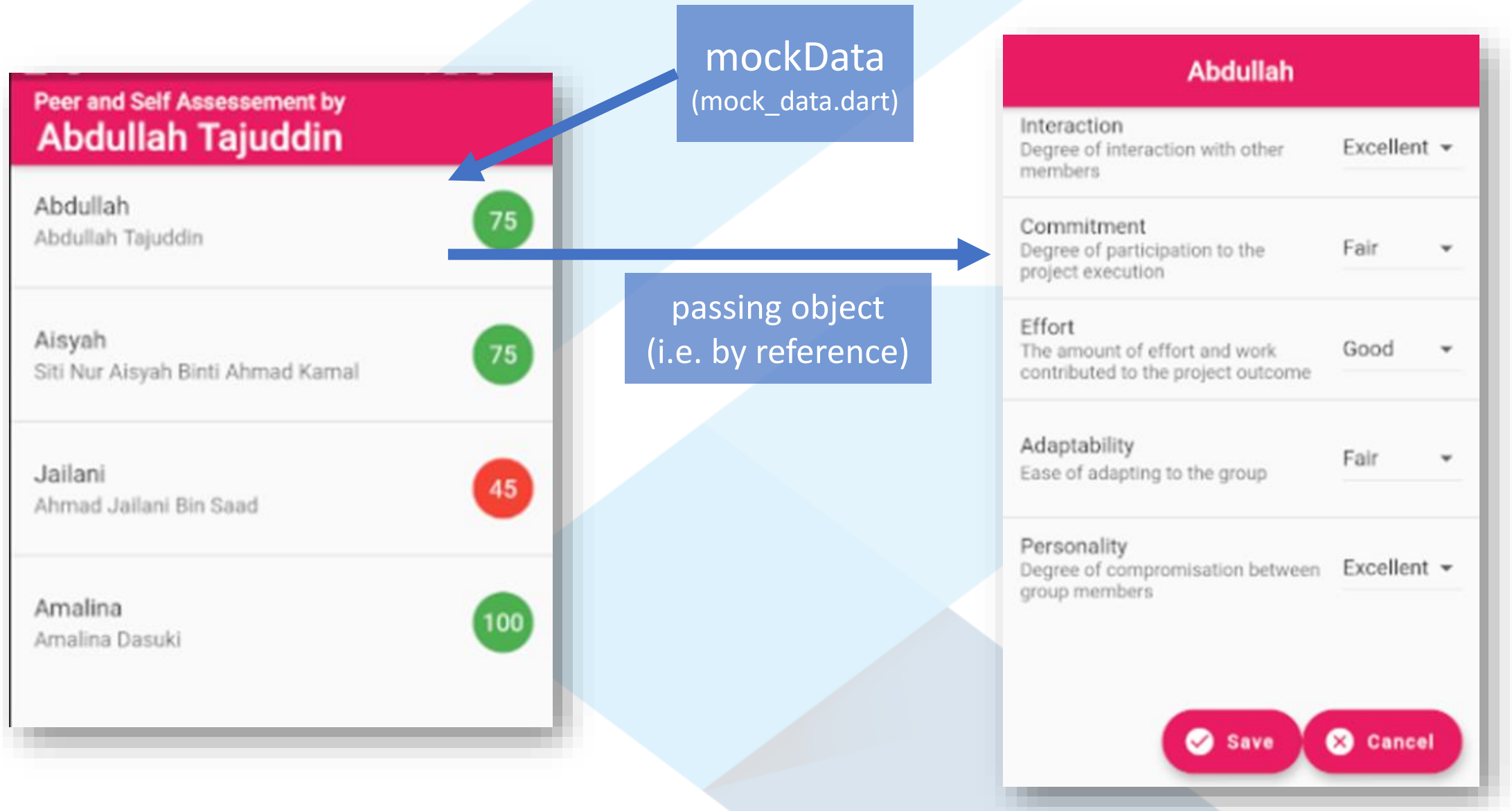
# Return from the second screen with result

Peer and Self Assesement by Abdullah Tajuddin	
Abdullah Abdullah Tajuddin	75
Aisyah Siti Nur Aisyah Binti Ahmad Kamal	75
Jailani Ahmad Jailani Bin Saad	45
Amalina Amalina Dasuki	100

Abdullah	
Interaction Degree of interaction with other members	Excellent ▾
Commitment Degree of participation to the project execution	Fair ▾
Effort The amount of effort and work contributed to the project outcome	Good ▾
Adaptability Ease of adapting to the group	Fair ▾
Personality Degree of compromisation between group members	Excellent ▾
<div><div>✓ Save</div><div>✕ Cancel</div></div>	

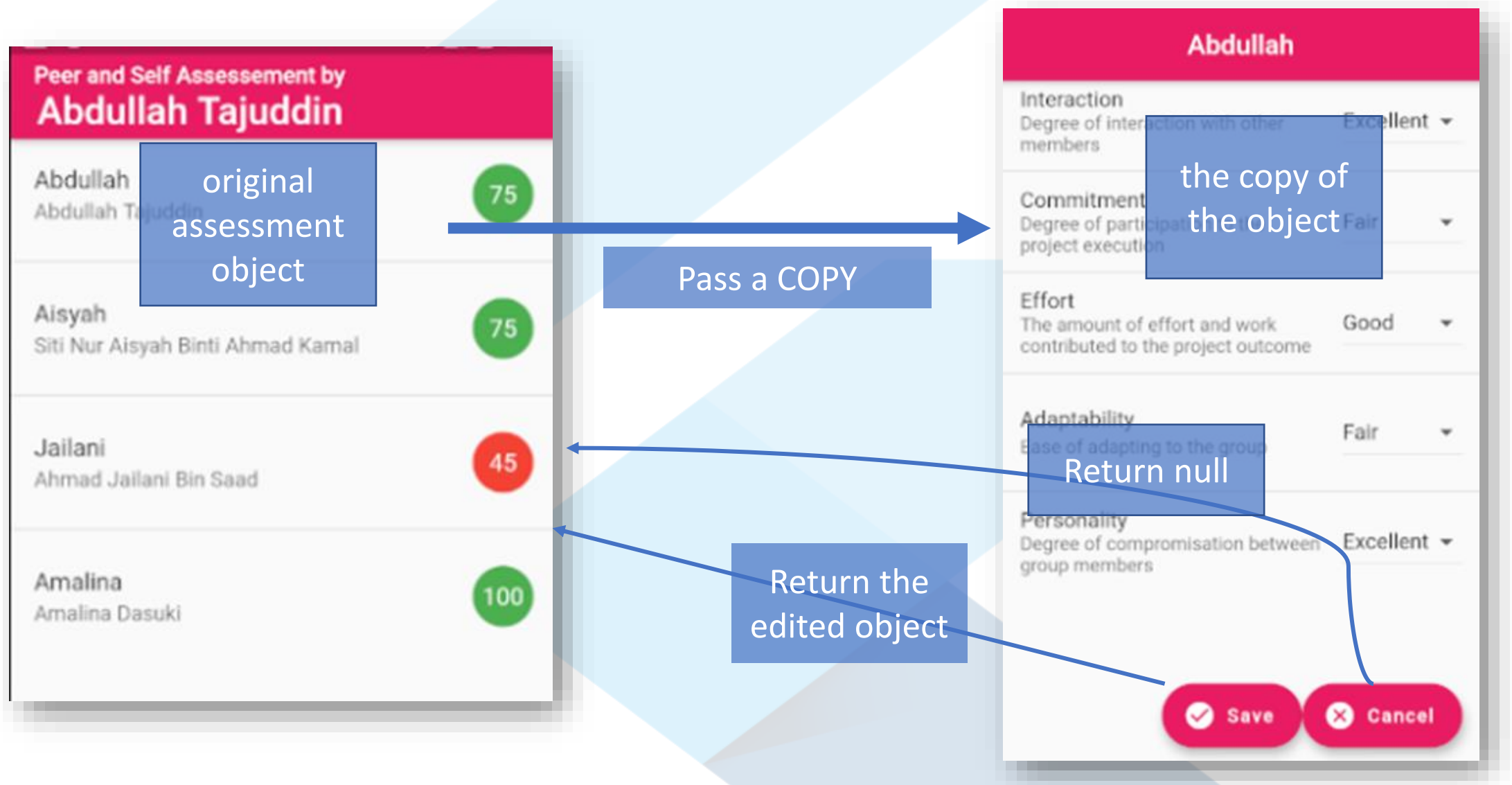
## Task 6

# Figure out the problem with the “Cancel” operation



## Task 6

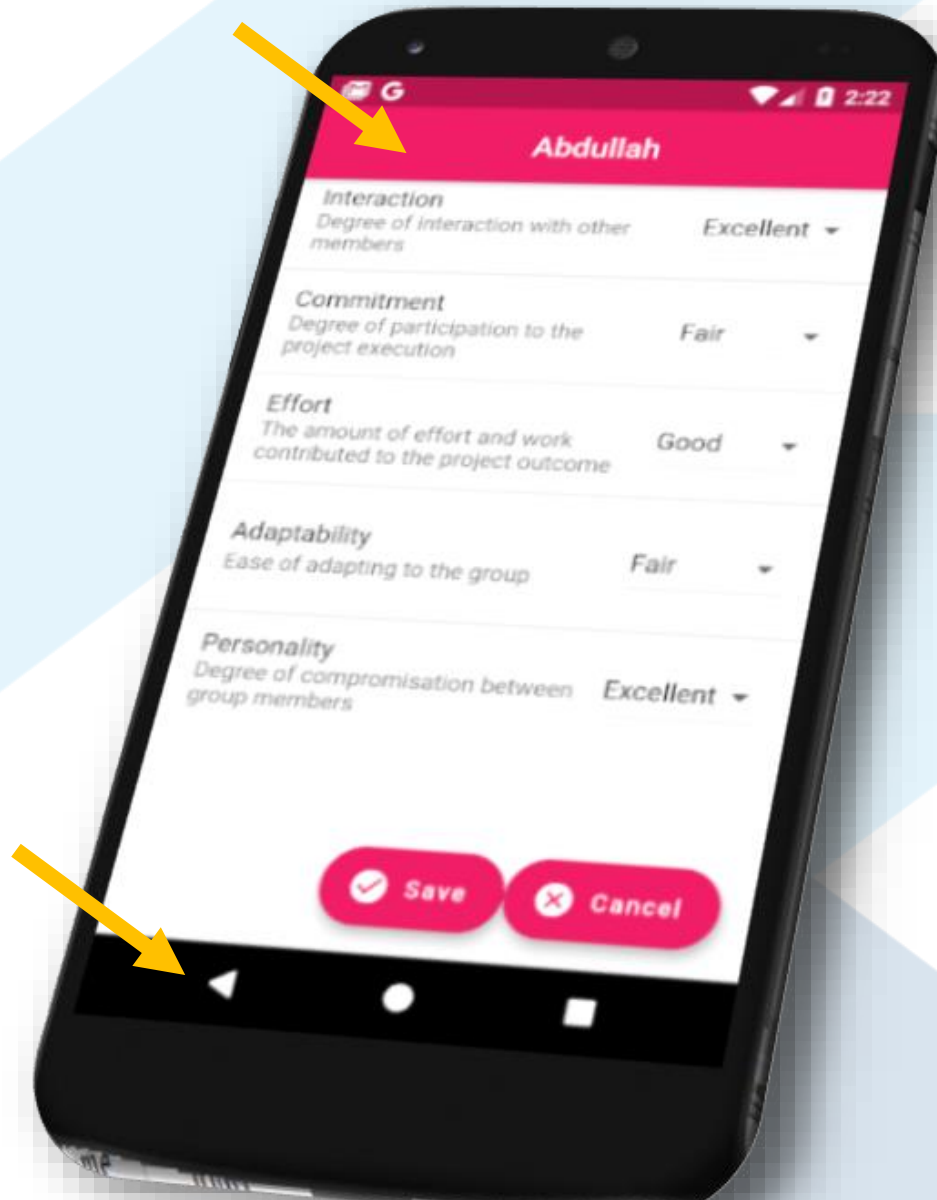
# Solution to the “Cancel” operation problem





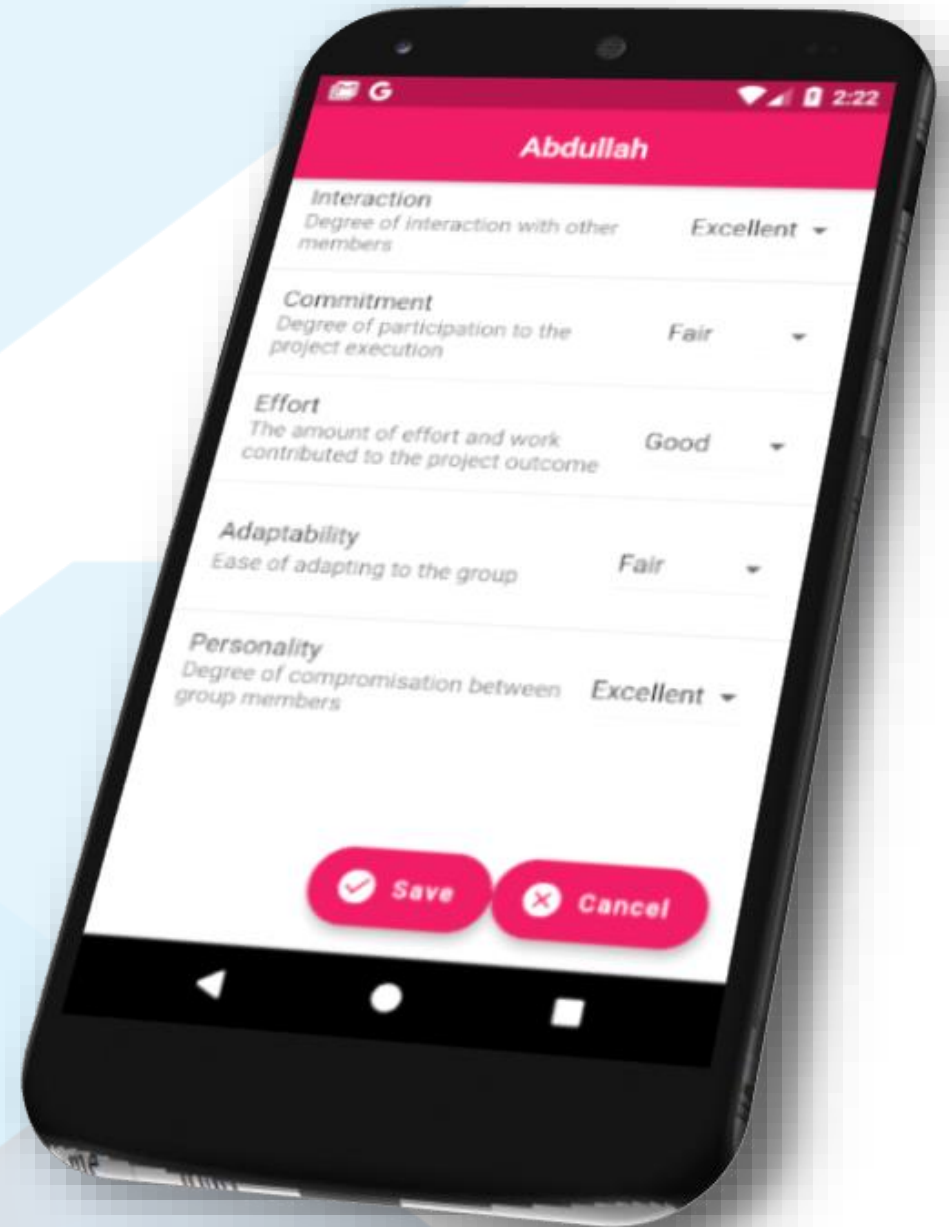
## Task 7

# Disable AppBar's and phone's back buttons



# Summary

- Define model classes
- Navigate to another screens
- Passing data from a screen to another
- Return result to the previous screen
- Asynchronous programming
- Passing a copy of object
- Disable default back buttons



# Navigation and Routing

# Named Routes

Jumail Bin Taliba

School of Computing, UTM

April 2020

# Why named routes?

- Reference a screen with name
- Create aliases

*e.g.* the home screen can be referenced with different names

`/`, `/home`, `/main`
- Define explicit names

*e.g.* constructing different screens from the same class

  - `/adminProfile` refers to `widget ProfileScreen('admin')`
  - `/userProfile` refers to `widget ProfileScreen('guest')`
- Centralize the routing code

# How to implement named routes?

`MaterialApp()` provides two ways:

- `onGenerateRoute` recommended
- `routes`

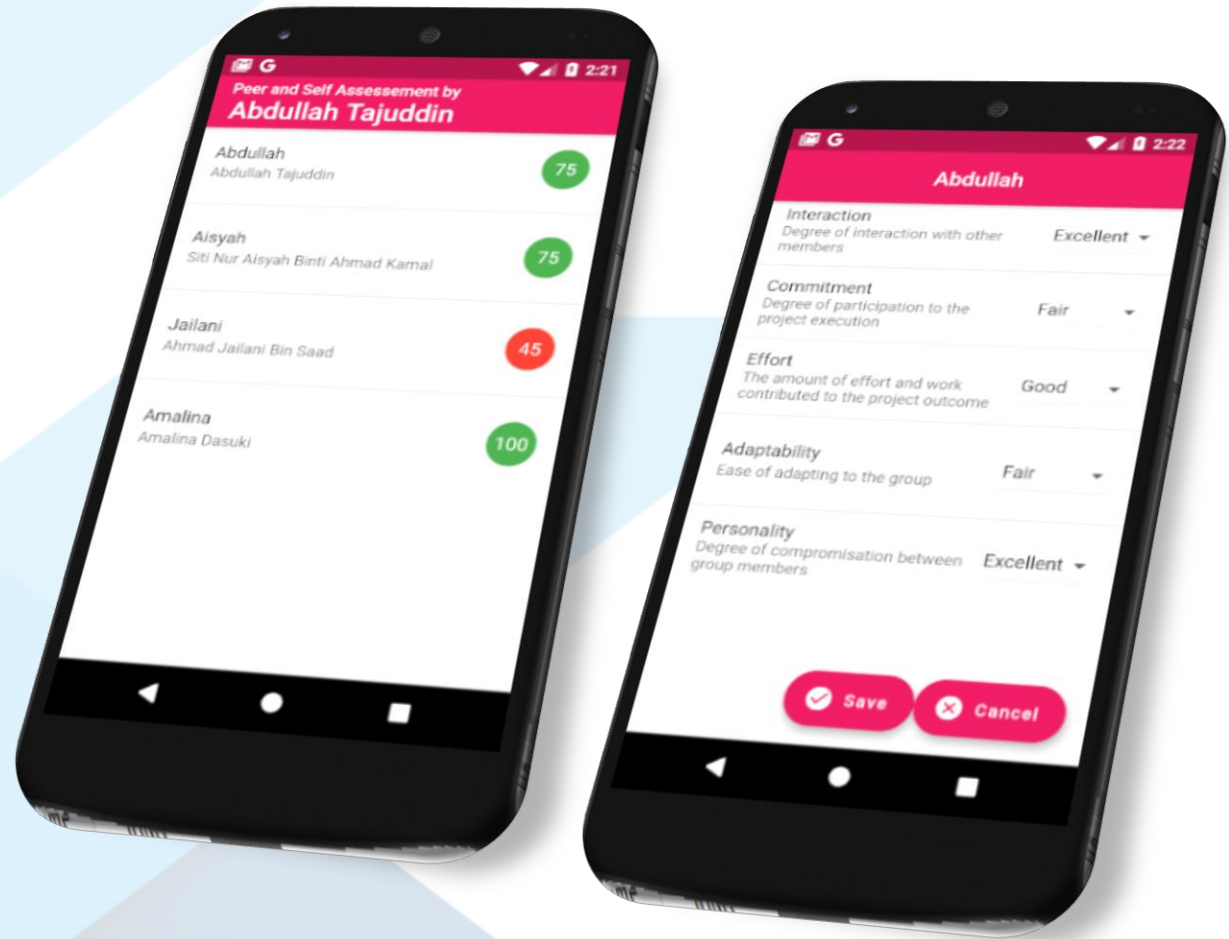
The demo will focus only on the “onGenerateRoute” approach

To learn about the “routes” approach, see the sample code (in the `using_routes` branch)

`git checkout using_routes`

# Demo App

Continue from the previous app



## Task 1

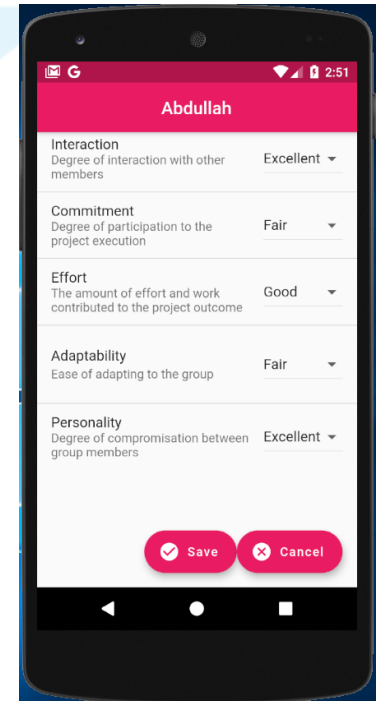
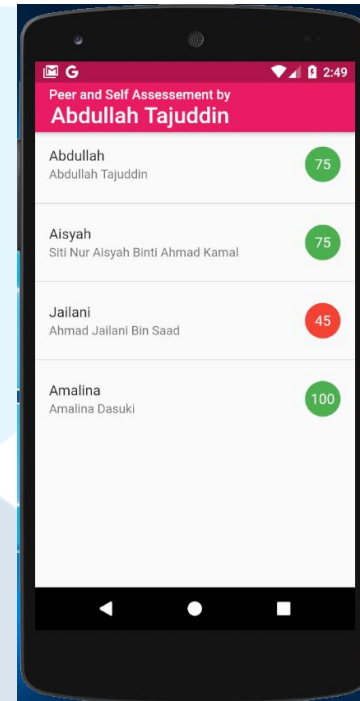
# Prepare the base code

# Clone the source code

```
git clone https://github.com/jumail-utm/navigation_named_routes
```

# Start from the base code

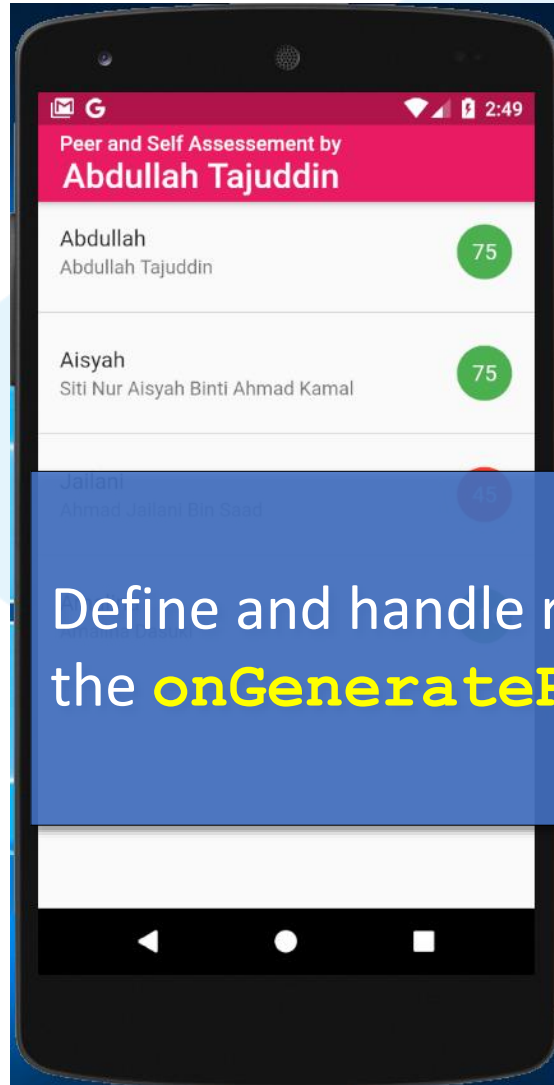
```
git checkout <the_base_commit>  
git branch playground
```



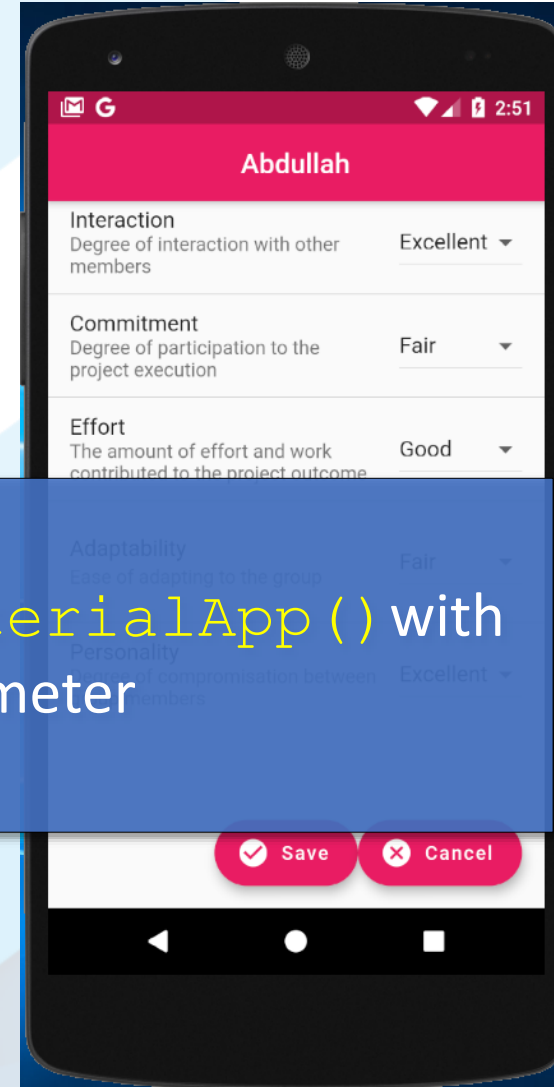
## Task 2

# Define and handle the named routes

"/summary"



"/details"

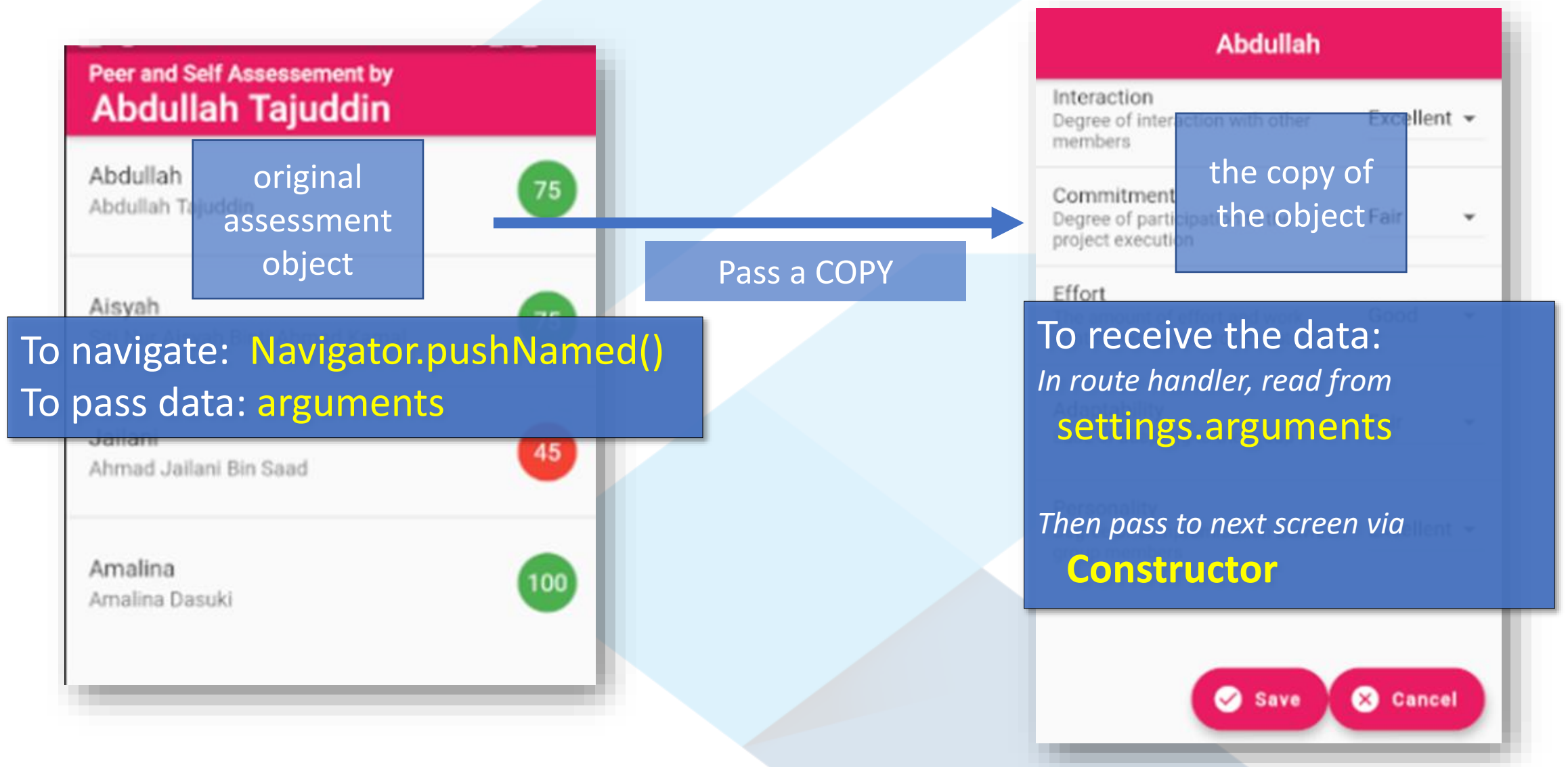


Define and handle routes in `MaterialApp()` with the `onGenerateRoute` parameter



## Task 3

# Navigate and pass data to the second screen



# How to pass multiple data

## Wrap the data in an object

```
.pushNamed(..., arguments: Arithmetic(op: '+',  
                                         numbers: [1, 2]));
```

## Use collections such as map or list

```
.pushNamed(..., arguments: ['+', 10, 5] );
```

```
.pushNamed(..., arguments: { 'op': '+',  
                             'num1': 10,  
                             'num2': 5 } );
```

recommended

## Task 4

# Return from the second screen with result

Peer and Self Assessment by Abdullah Tajuddin	
Abdullah Abdullah Tajuddin	75
Aisyah Siti Nur Aisyah Binti Ahmad Kamal	75
Jailani Ahmad Jailani Bin Saad	45
Amalina Amalina Dasuki	100

Abdullah	
Interaction Degree of interaction with other members	Excellent
Commitment Degree of participation in project execution	Fair
Effort The amount of effort and work contributed to the project outcome	Good
Adaptability Ease of adapting to the group	Fair
Personality Degree of compromise between group members	Excellent

the copy of the object

Return null

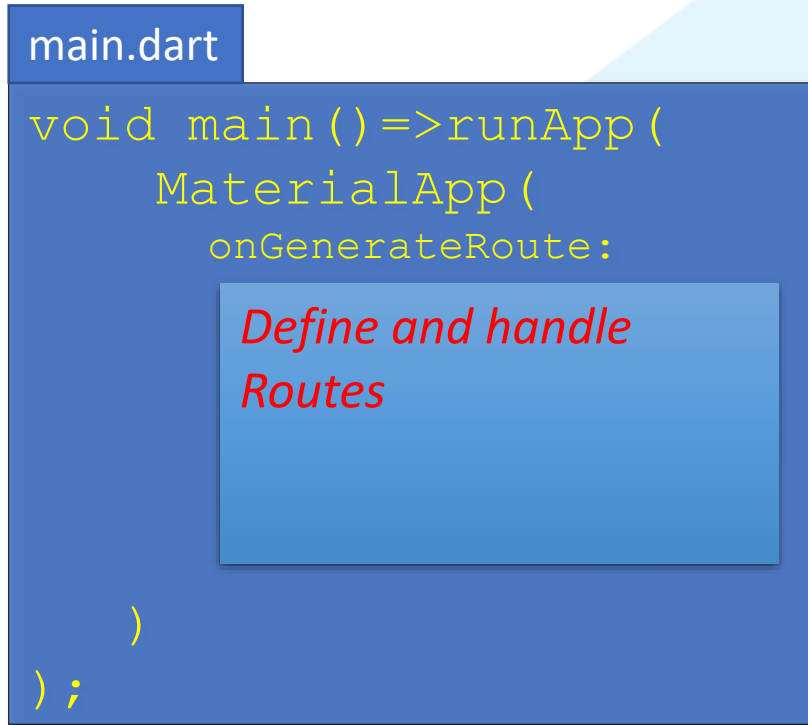
Return the edited object

To return: `Navigator.pop()`  
To pass result: via parameter

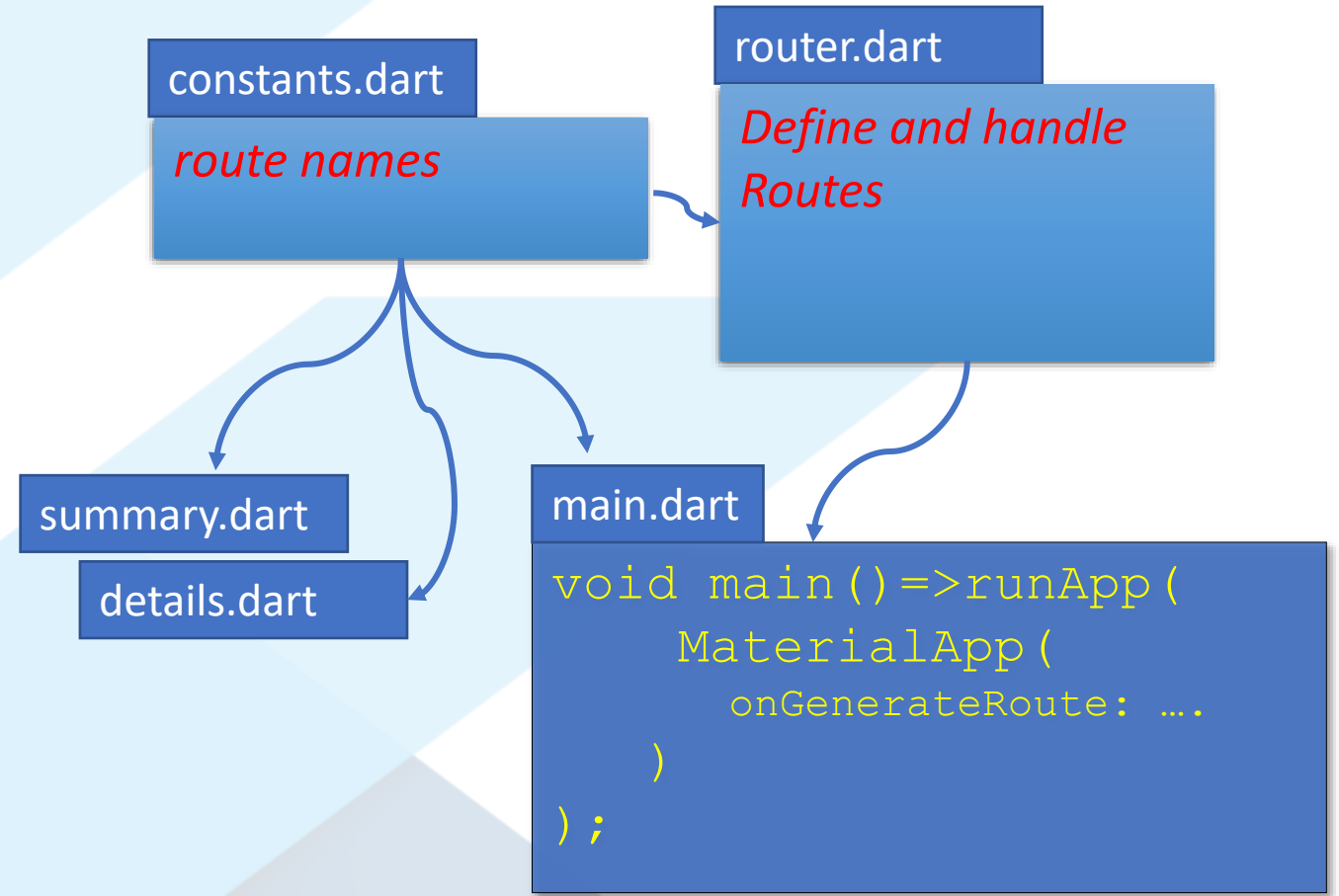
## Task 5

# Refactor routing code into separate files

Before



After

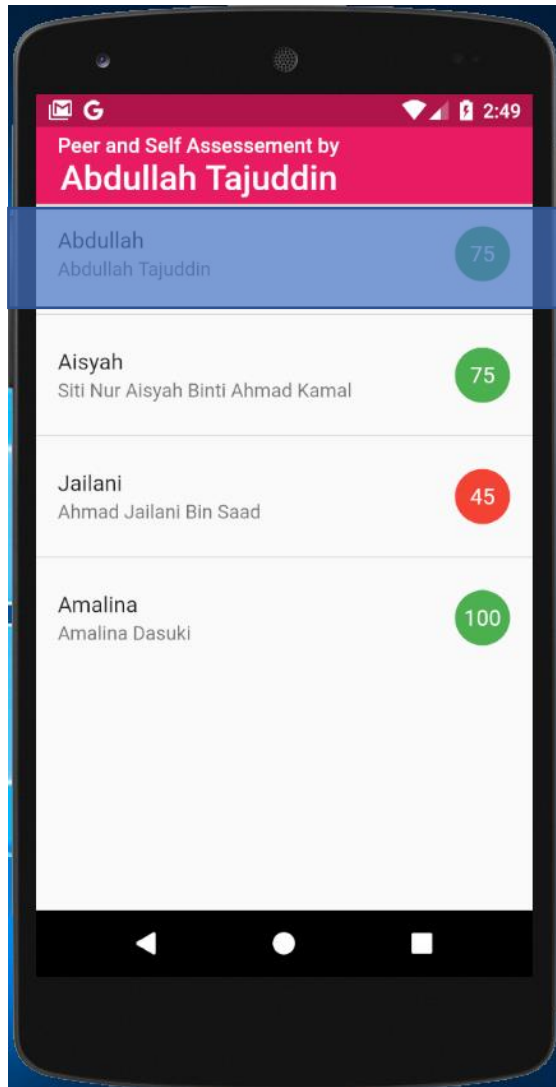


# Program's file structure

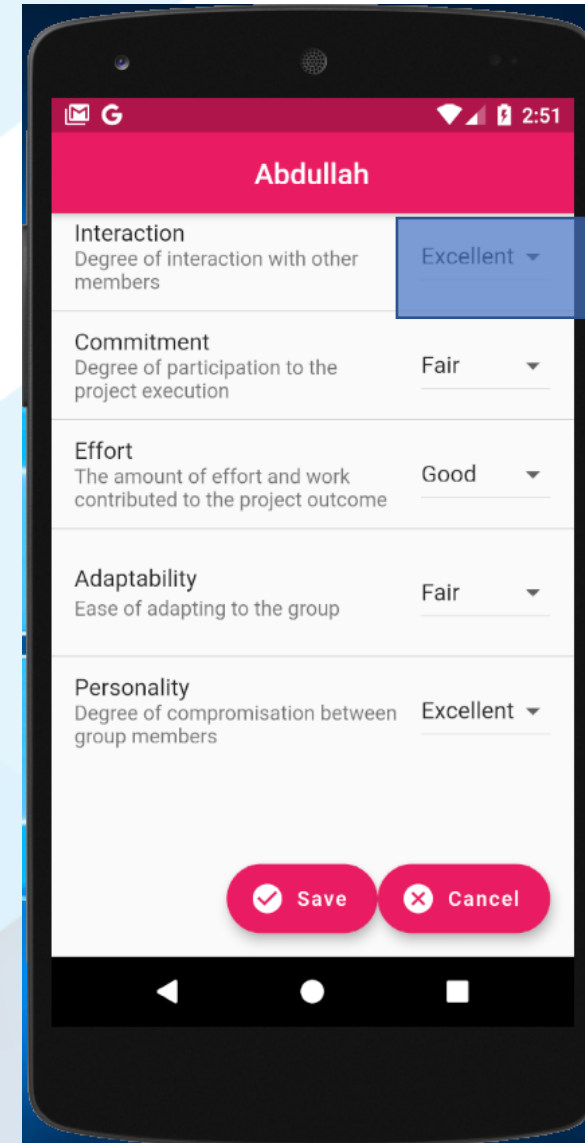
```
[navigation_named_routes]
|
+---[lib]
|
|   + ---main.dart
|   + ---router.dart
|   + ---constants.dart
|
|   + ---[models]
|       + ---assessment.dart
|       + ---group_member.dart
|       + ---mock_data.dart
|       + ---form.dart
|
|   + ---[screens]
|       + ---summary.dart
|       + ---details.dart
```

## Task 6

Reorganize the UI built, make only selected child widget stateful rather than the whole screen



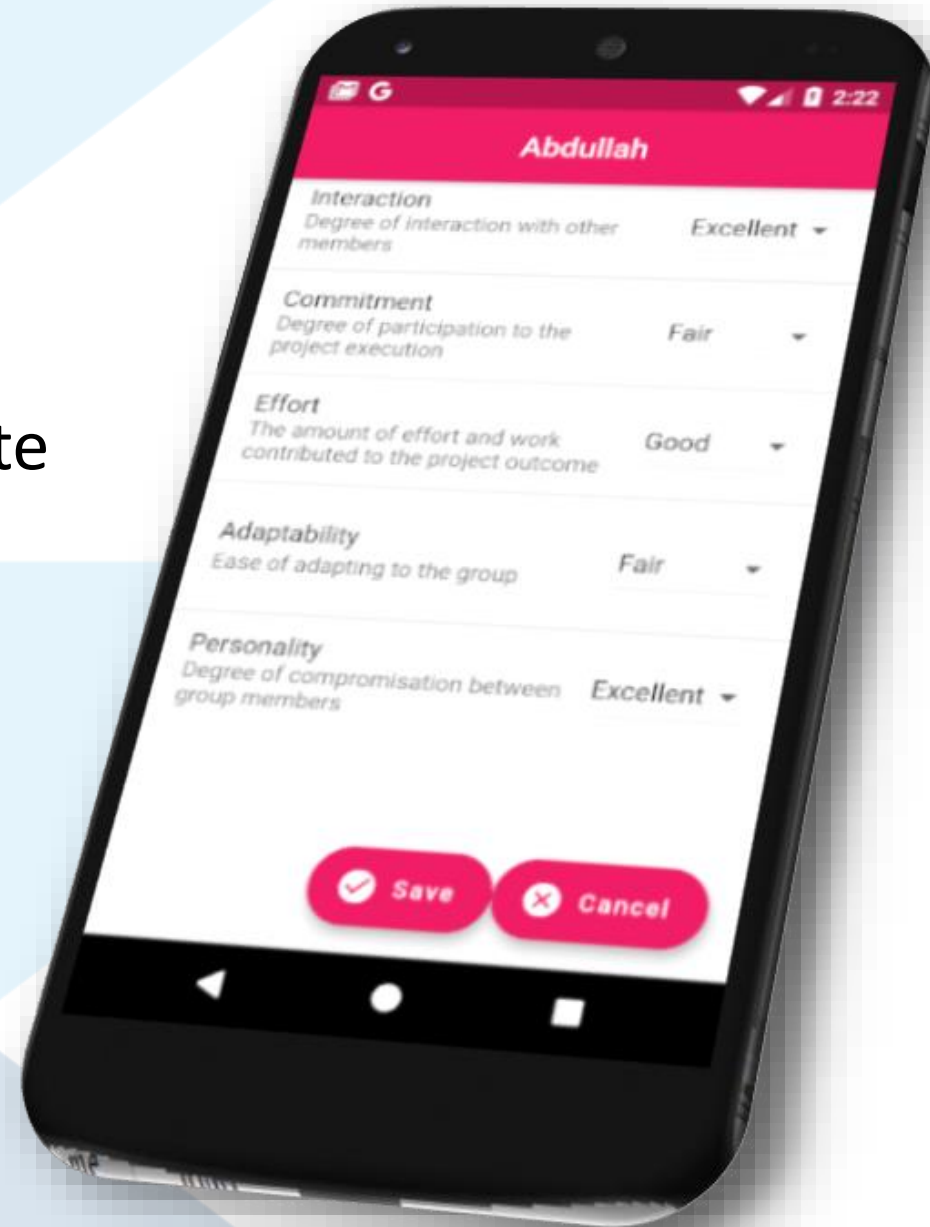
ListTile  
will be stateful



DropDownButton  
will be stateful

# Summary

- Why use named routes
- Implementation: routes and onGenerateRoute
- Navigate and pass data to another screen
- Back to main screen and pass results
- Refactor routes



# Navigation and Routing

# More Push and Pop

# Operations

Jumail Bin Taliba  
School of Computing, UTM  
April 2020



# Agenda

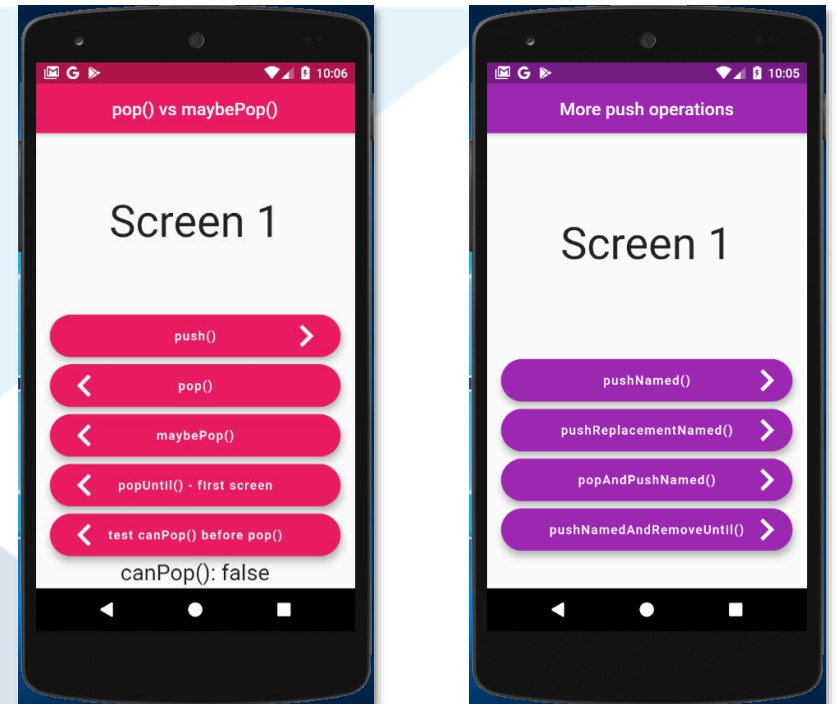
- More pop operations:
  - `maybePop()`
  - `canPop()`
  - `popUntil()`
- More push operations:
  - `pushNamed()`
  - `pushReplacementNamed()` ,
  - `popAndPush()`
  - `pushNamedAndRemoveUntil()`
- How they work and some use cases
- Passing data and return results between routes

# Download the source code

```
git clone https://github.com/jumail-utm/navigation_push_pop
```

There are two branches (besides master):

```
git branch -a (to check branch list)  
git checkout more_pop (Example 1)  
git checkout more_push (Example 2)
```



# About the codebase

- Only one file, main.dart
- Only one Screen widget class
- Multiple routes from the same class

# maybePop ()

- It's like `pop ()` but doesn't work on the last screen.
- Example use case: to prevent user from closing the app accidentally

In Screen 3, invoking

**Navigator**

pops the screen

In Screen 2, invoking

**Navigator.maybePop (context)**

pops the screen from the stack

In Screen 1, invoking

**Navigator.maybePop (context)**

**has no effect**



# canPop ()

Returns **true** if a screen can be popped off from the stack

Example use case: to override phone back button

Example:

- Invoking this method in Screen 3 and 2 returns **true**
- Invoking this method in Screen 1 returns **false**



# popUntil ( )

- pop off all screens from the current one till reaching the specified screen.
- Must explicitly name the route with **settings** parameter
- Example use case: in a shopping app, to return to the home screen after completing payment process

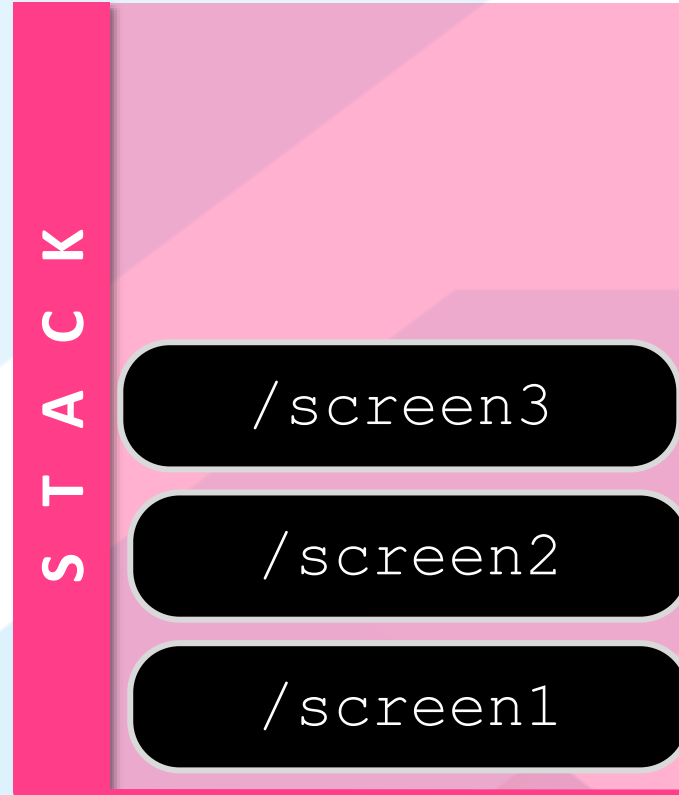
In Screen 3, invoking

```
popUntil( /screen1 )
```

pops Screen 3 and Screen 2



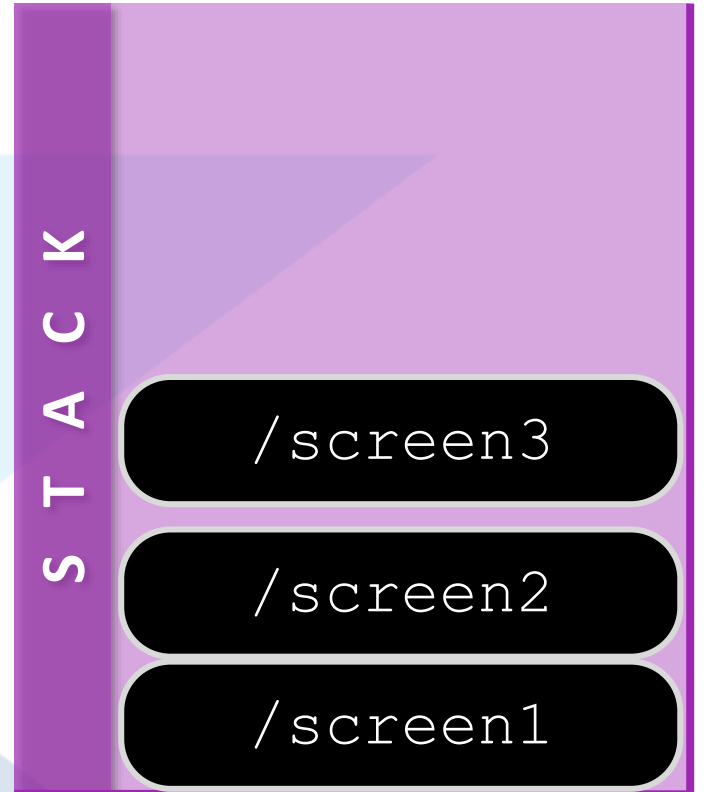
# More pop operations



# `pushReplacementNamed()` `popAndPushNamed()`

- Replace the current screen with a specified one
- To prevent going back to the previous screen
- The difference between these two methods: screen transition animation

`/replacement`

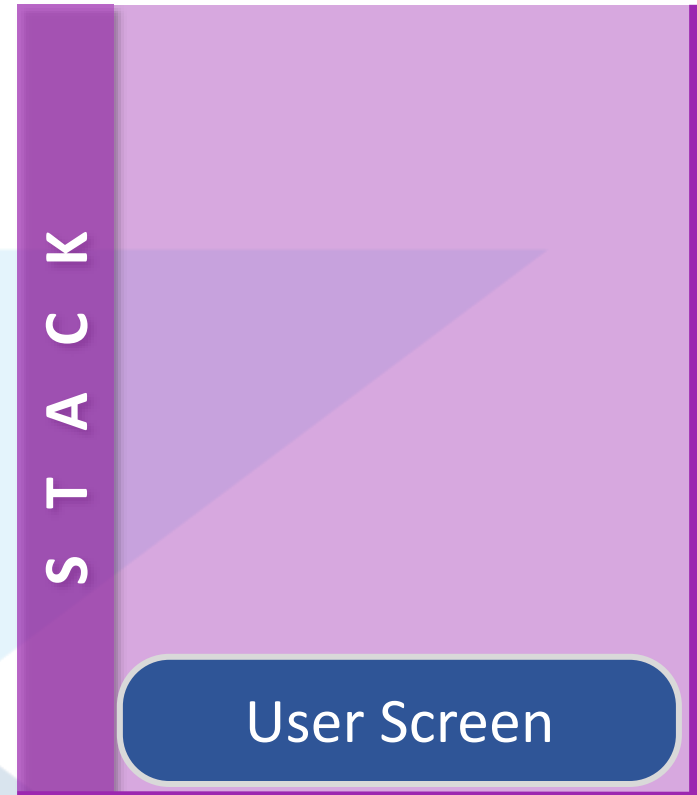




# `pushReplacementNamed()` `popAndPushNamed()`

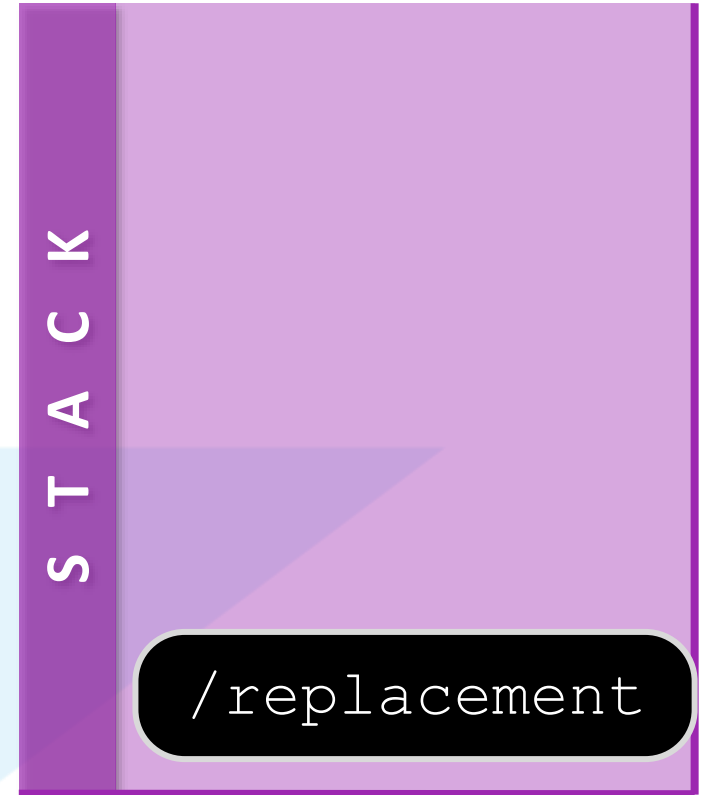
Example use cases:

- Show Main screen after a splash screen
- Open User screen after logging in



# pushNamedAndRemoveUntil()

- Pops all screen till reaching the specified screen and push a new screen



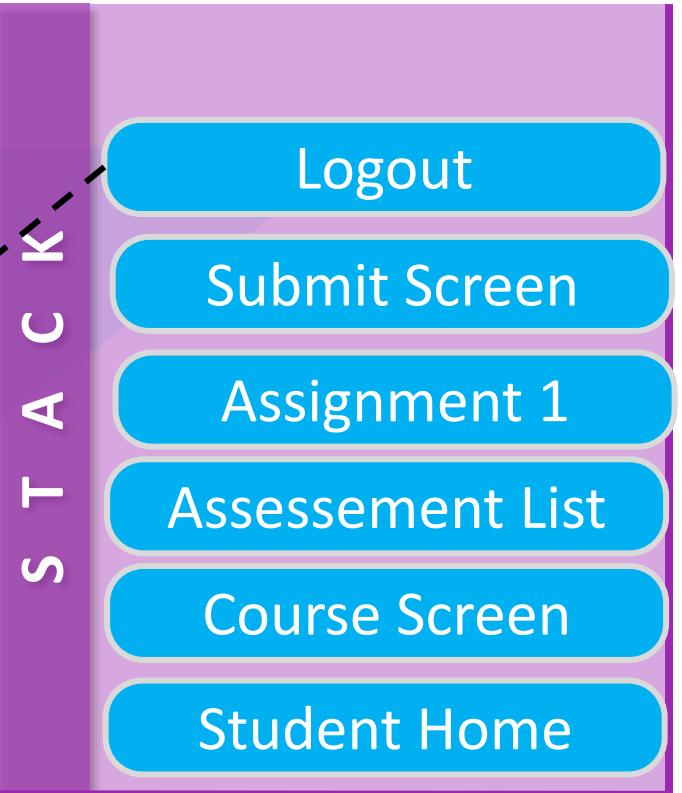
# pushNamedAndRemoveUntil ()

Example use cases:

- Logout after going through a series of screens:  
e.g. Logging out after submitting an assignment on elearning, bring the user to Elearning Home Screen

*in Logout*

```
Navigator.pushNamedAndRemoveUntil (  
  context,  
  '/elearningHome',  
  ModalRoute.withName('/studentHome')  
)
```

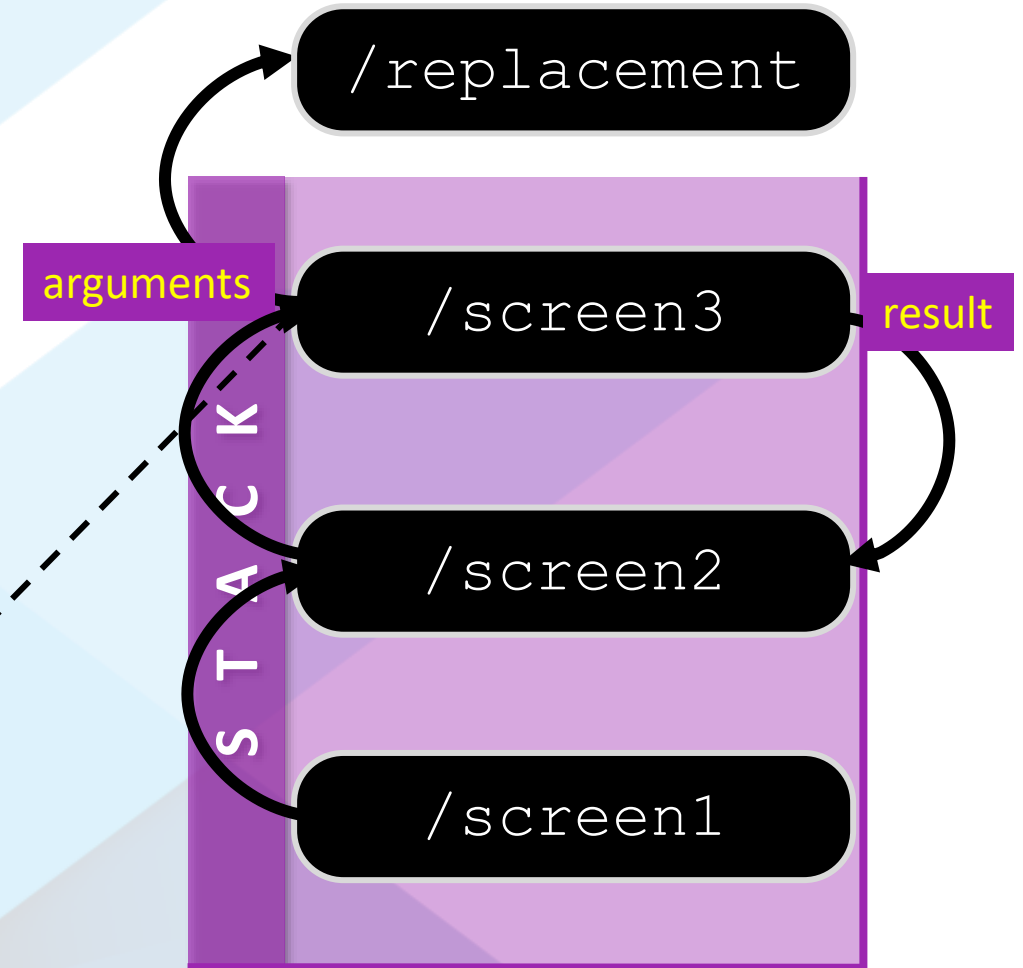


# Passing data and results between routes (1)

- A route may pass data to the next route via the **arguments** parameter
- A route may return results to its **creator** via the **result** parameter (or the second parameter in case of pop() method)

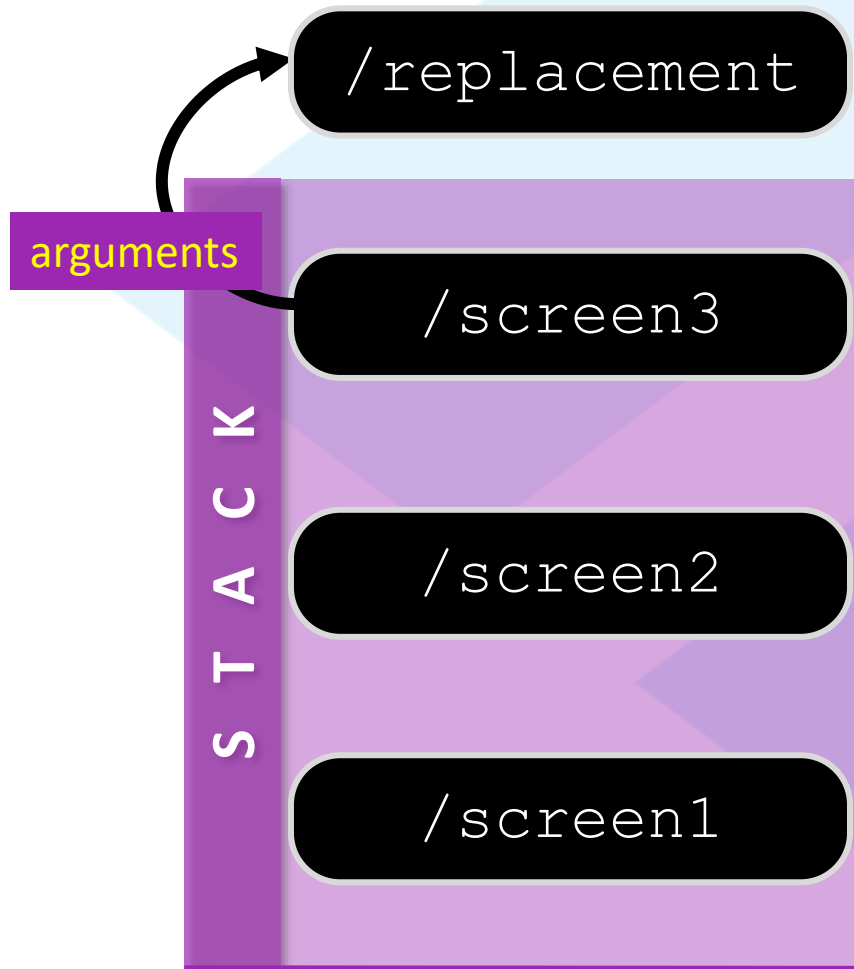
*in Screen 3*

```
Navigator.pushReplacementNamed  
(context, '/replacement',  
  arguments: ....., result: .... )
```



# Passing data and results between routes (2)

- The **arguments** is sent to onGenerateRoute callback function
- Then, passed to the route via the screen **constructor**



```
Route<dynamic> generateRoute(settings) {  
  final data = settings.arguments; // The value of t  
  
  switch (settings.name) {  
    case '/':  
    case '/screen1':  
      return MaterialPageRoute(  
        builder: (_) => Screen( // Screen  
      );  
    case '/screen2':  
      return MaterialPageRoute(  
        builder: (_) => Screen( // Screen  
      );  
    case '/screen3':  
      return MaterialPageRoute(  
        builder: (_) => Screen(  
          title: 'Replacement',  
          nextRoute: null,  
        ), // Screen  
      ); // MaterialPageRoute  
  }  
  
  return null;  
}
```

# Passing data and results between routes (3)

- The **creator** route receives the result from the function **return**

*in Screen 2*

```
theResult = await Navigator.pushNamed  
(context, '/screen3', arguments: .... )
```

