Exercise 3
Navigation and Routing

Overview

- This exercise is to be conducted **outside of the class.**
- You will be adopting a **Pair Programming** strategy in doing this exercise.
  What is pair programming? (https://youtu.be/oBraLLybGDA)
- You can maintain your partner from the previous exercise or change to a new one. You may also choose your partner from the other section.
- You and your partner will be coding collaboratively online using VS Code and Live Share.
  Using Live Share for online collaborative coding (https://youtu.be/s9hfONtUcR8)
- You will communicate to each other using Google Meet, Webex or any online meeting tool.
- Record your pair programming session.

Plagiarism Warning

You may discuss with others and refer to any resources. However, any kind of plagiarism will lead to your submission being dismissed. No appeal will be entertained at all..

Late Submission and Penalties

- The submission must be done via elearning. Other than that (such as telegram, email, google drive, etc), it will not be entertained at all.
- Programs that CANNOT COMPILE will get a 50% penalty.
- Programs that are submitted late will get a 10% penalty for every 10 minutes.

Pair Programming and Collaborative Coding

- Pick any time worth **THREE (3) hours** (maximum) within the given date to conduct the pair programing session with your partner.
- You may also split your pair programming into several sub-sessions provided the total time is still within 3 hours.
- Log the date and time for every pair programming session conducted. Write them in the program source code.
- Record the meeting about your pair programming session. If you do your programming in multiple sessions, record all of them. You do not have to edit the video.
- Code submissions without the video at all or the video was too short, will be declined.

Notes:

- You are advised to explore the exercise on your own first before doing the pair programming session with your partner. This should make yourself be more prepared for the pair programming session.

How To Record the Session

- You can choose Google meet as your online meeting tools, and use the feature **"record meeting"** to record your pair programming session.
- Note that, a free personal google account does not support the recording feature.
- Try using your student or graduate account from UTM to be able accessing the "record meeting" feature.
- Alternatively, you can also record your par programming session locally using softwares like OBS, PowerPoint, etc. Additional Resources: Video recording and editing

Exercise Material

This exercise comes with the following materials:

- A codebase that has been organized into a structure as shown in Figure 1, 2 and 3
- A video of the expected result
- The question

Additional Materials – Guide Video

I had made a video about some guides to the exercise. Check it out on elearning.

https://youtu.be/iT1e2qDCuMc

Please take note that, in the video I mentioned "Exercise 2" and "Exercise 1". It is supposed to mean "Exercise 3" and "Exercise 2", instead.

Also, please do make changes on the file name "router.dart" to "routes.dart". The new file (routes.dart) should consist a class called `Routes`

File Structure

```
[exercise]
        |
        +---[lib]
        |       |
        |       + ---main.dart
        |       + ---routes.dart (routes class)
        |       |
        |       + ---[models]
        |       |       |
        |       |       + --todo.dart  (model class for Todo)
        |       |       + --task.dart  (model class for Task)
        |       |       + --mock_todos.dart  (mock data)
        |       |
        |       + ---[screens]
        |       |       |
        |       |       + --[todo_list]
        |       |       |       + ---todo_list_screen.dart
        |       |       |       + ---bar.dart
        |       |       |       + ---body.dart
        |       |       |
        |       |       + --[task_list]
        |       |       |       + ---task_list_screen.dart
        |       |       |       + ---bar.dart
        |       |       |       + ---body.dart
        |       |       |       + ---float.dart
```

Figure 1: Project File Structure

TodoListScreen

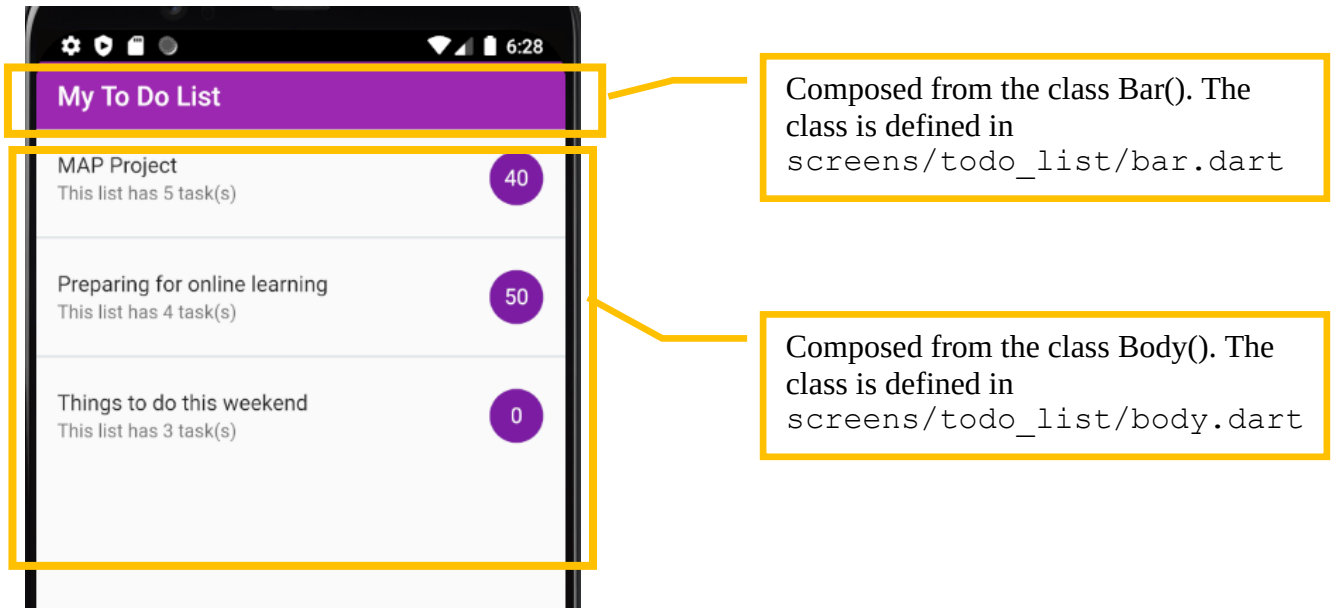


Figure 2: Components of TodoListScreen

TaskListScreen

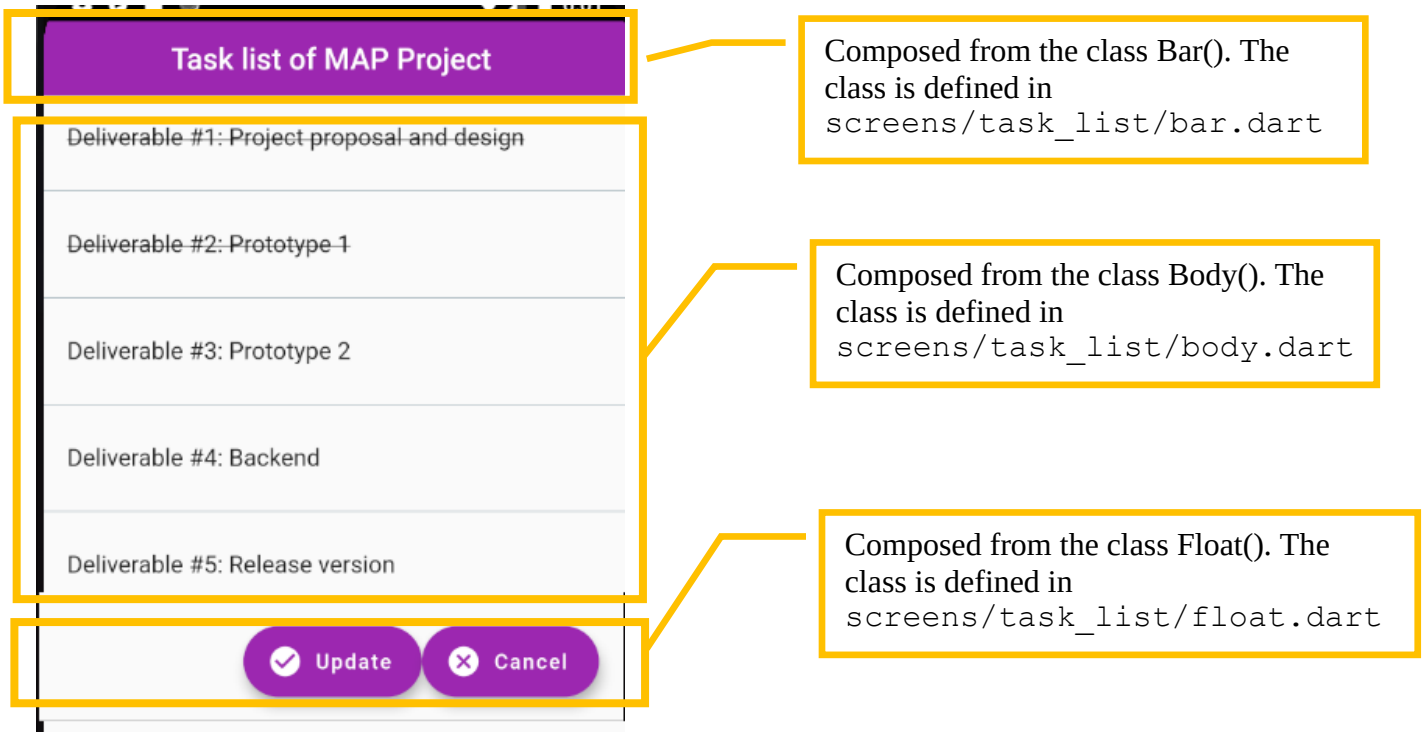


Figure 3: Components of TaskListScreen

Question

In this exercise you will be implementing navigation and routing using the class Navigator. Develop a simple Todo application consisting of two screens, the first screen (Todo List) and the second screen (Task List).

Todo List Screen

This screen shows the list of list of todo items along with the following information:

- The number tasks that each todo item has.
- The progress of each todo item shown by the percentage of tasks completed for the todo item.

Task List Screen

This screen will show up when the user taps on a todo item in the first screen. The Task List screen shows the list of tasks for the selected todo has. The user can perform the following operations on the task items in this screen:

- Toggle the status a task from "in-progress" to "completed" or vice-versa by tapping on the task. A completed task is shown in Strike-through (or line-through), whereas an in-progress task in normal text.
- Remove a task item by long-pressing on the task.

Navigation

Use named routes approach to implement the navigation.

Model Classes and Mock Data

Both screens must be built dynamically (do not use hard-coded to build the screens). In this regard, you need to define the following classes and create some mock data by using these classes:

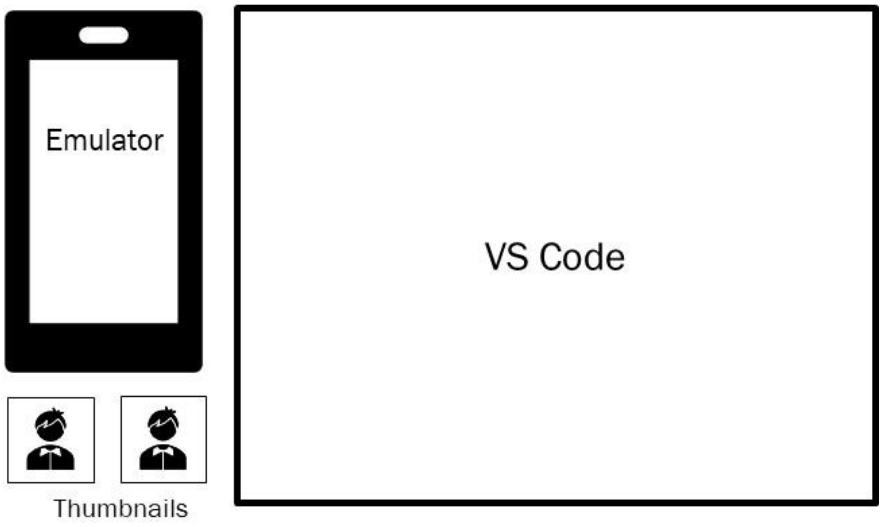
- class Task. Each task has a title and the status indicating whether it has been completed or not. This class is fully given in the base program.
- class Todo: Each todo has a title and a list of tasks associated to it. Add a convenience method with a getter to calculate the percent of tasks completed. You may also need to define a copy constructor for this class.
- Mock data: create your own mock data for a list of todo items. Each todo item then needs to have a list of tasks.

Additional Requirements

- You must log your code changes with Git. Read the guide below about the git commits.
- Organize your code and files properly. In this regard, do the followings:
    - Refactor your code into several classes rather than put them in a big class.
    - Write each class in a separate file.
    - Follow proper naming convention in the code and file names.

About the Video

- The video is not meant for presentation purposes, but for recording your pair programming session.
- The video must show that you are coding, communicating and collaborating with your partner. In this regard, **speak only in English**.
- In the video, show your VS Code and the emulator side-by-side for example as follows. Turning on the web cam is optional (but recommended). As for the emulator, you may also use a web emulator instead of a mobile emulator.



- Besides, at the end of the video, do include a walkhthrough about the outcome of your work. It should follow the order the tasks or questions of the exercise (see the Assessement).
- You can record the session in a single or multiple videos. If you use multiple videos, put them in a folder, and share only the folder's link. Submit the link to elearning.

- Set the video file (or folder) permissions with “Anyone can view”.
- Make sure the video is available until the end of the semester.
- Submit the raw videos, i.e., you don’t have to do post-editing.

Notes:

- Please make the font-size of your VS Code a little bit larger so that it easy for me to see your code in the video. You can do this by pressing the key **Ctrl** and **+** in VS Code.

About the Git Commits

- Only one user needs to issue the commits, for example by the host's user.
- As for the first commit, include both of your name and your partner's into the commit description.
- For each commit, include information about who was the driver and navigator for the related task.
- The commits you need to issue are based on the "Code criteria" stated in the Assessment section, i.e., File and code structure, custom widget for action buttons, etc.
- Write your commit messages based on the guidelines presented in this article: How to Write a Git Commit Message (https://chris.beams.io/posts/git-commit/)

Git Commands

The followings are some git commands you may need to use in this exercise.

Initializing Git

1. Go to the exercise folder (using Git Bash)
2. Type the following command to initialize the folder with git

```
git init
```

Committing Changes

1. Type the following command to commit (or to record) any changes you have made. This will add new commit to the git repo.

```
git add -A
git commit -a -m "a good commit message"
```

2. To update the recent commit, type

```
git add -A
git commit --amend
```

Creating A Git Bundle File

1. Once you have done with your exercise, prepare a git bundle for submission.

   ```
   git bundle create ../your_filename.git HEAD master
   ```

   Perform the above command in the `master` branch and make sure you create a bundle for your latest code. You may want to perform the "`git commit`" before creating the git bundle file.

2. If you want to test out the git bundle file (exercise1.git) before you submit, clone the repo in another folder. First, copy the git file (using Window explorer) to the other folder. Then clone the repo (using Git bash)

   ```
   git clone your_filename.git test.git
   ```

Assessment

This exercise carries **5%** weightage for the final grade of this course. The breakdown weightage is as follows (out of 100 points):

Criteria	Points
1. The code	
a. Model classes and mock data	
i. class Todo	5
ii. Mock data	5
b. Building the first screen	
i. TodoListScreen	5
ii. Body	10
c. Building the second screen	
i. TaskListScreen	5
ii. Bar	5
iii. Body	10
iv. Float	5

d. Navigation	
i. Navigate between screens	5
ii. Passing data and result between screens	5
iii. Update and cancel operations	10
e.	
2. Git commits	
a. Performing commits for all requirements or tasks	5
b. Writing good commit messages	5
3. Pair Programming	
a. Video and overall (including outcome walkhthrough)	10
b. Active collaboration	5
c. Both members play both roles Driver and Navigator.	5

Submission

- Deadline: **Sunday, 12 Dec 2021, 11:59 PM**
- Only one member from each pair needs to do the submission.
- Submission must be done on elearning. Any other means such as email, telegram, google drive will not be accepted at all.
- You will need to submit TWO (2) items:
  a. The source code in a **git bundle** file.
  b. The video link (submit later, once the video is ready). If you recorded in multiple video files, put the videos in a folder and share only the folder link. Label each file in order, for example, part 1, part 2, etc.

FAQs

1. **Who will be my partner?**

   In this exercise you must rotate your partner. That means, you will partner with someone that you have not paired with before in prevous exercises.

2. **Can I pair up with someone from a different section?**

   You can pair with anyone from any section provided you fulfill the requirement from FAQ (1) above.

3. **Can I do the exercise alone?**

   You must do the exercise in pair..

4. **What do we need to show in the video?**

   You should show that you are **doing pair programming** rather than explaining about your code. The video is not meant for presentation.

5. **Do we need to switch roles between Driver and Navigator?**

   Yes. Your video should show that you and your partner keep switching between these two roles. No one should be dominant or play only one particular role.

6. **What if we do pair programming physically (face-to-face)?**

   You and your partner should use only one computer and sit side-by-side. You do not have to open LiveShare and online meetings. You can record the video locally using software like OBS. Again, you still need to talk and discuss with your partner in the video. It is also recommended to turn on the web camera. Keep in mind that you keep following the SOP about COVID-19 when working in a face-to-face environment.

7. **Do we need to upload our work to Github?**

   No. In this exercise, you will use Git. However, you will package your work locally using git bundle, instead of pushing it to Github.