CCS3351 Mobile Application Development

Mid term project (100 marks)

Instructions:

- Submit your answers as a single file (**.ZIP**) on or before the deadline provided in the LMS.
- Submission must include this document explaining your code, UI and lessons learnt
- Late submission will not be considered for the marking.
- Make sure to include this document in your submission

Full name: Chandrasekara Arachchige Pasindu Kavishka Widushan

Student index: 22UG1-0729

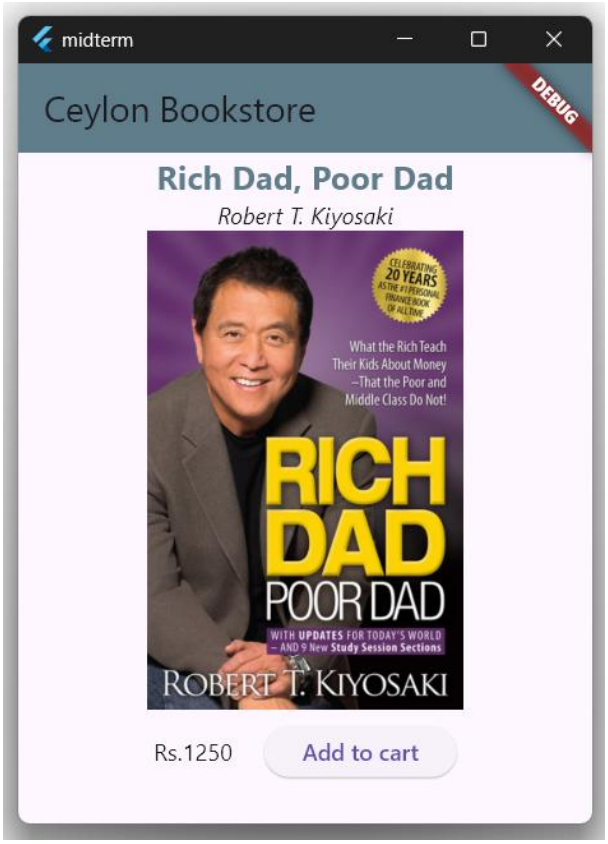
Date of submission: 24 / 01 / 2025

Understanding layout

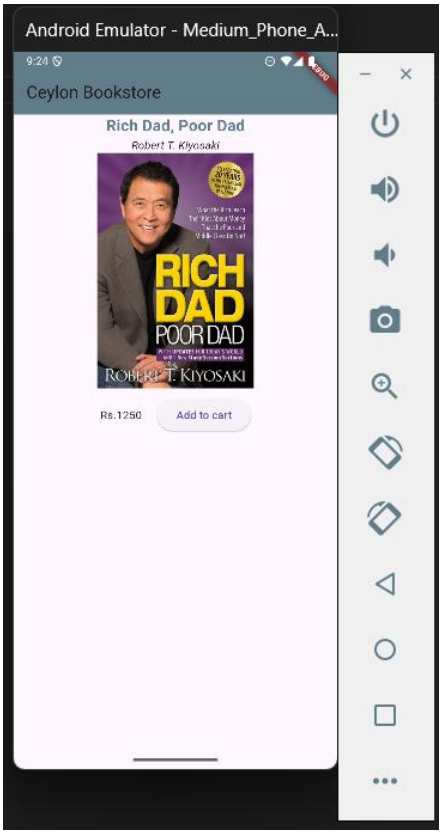
The core of Flutter's layout mechanism is widgets. In Flutter, almost everything is a widget—even layout models are widgets. The images, icons, and text that you see in a Flutter app are all widgets. But things you don't see are also widgets, such as the rows, columns, and grids that arrange, constrain, and align the visible widgets. You create a layout by composing widgets to build more complex widgets.

Here's an example of a book store app, showing a list of books. Observe the completed UI (browser and emulator) and understand the breakdown. Create the Book layout as a custom widget, which can be reused.

Single book

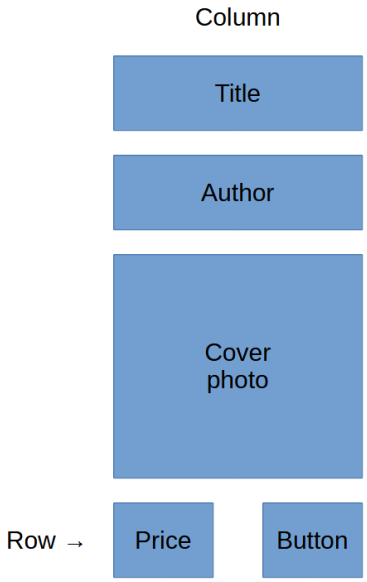


Browser



Widget layout

You may or may not use a Card to enclose the elements. The layout uses 4 elements inside a Column and 2 elements in the last Row. Text widget is used for Title, Author and Price.



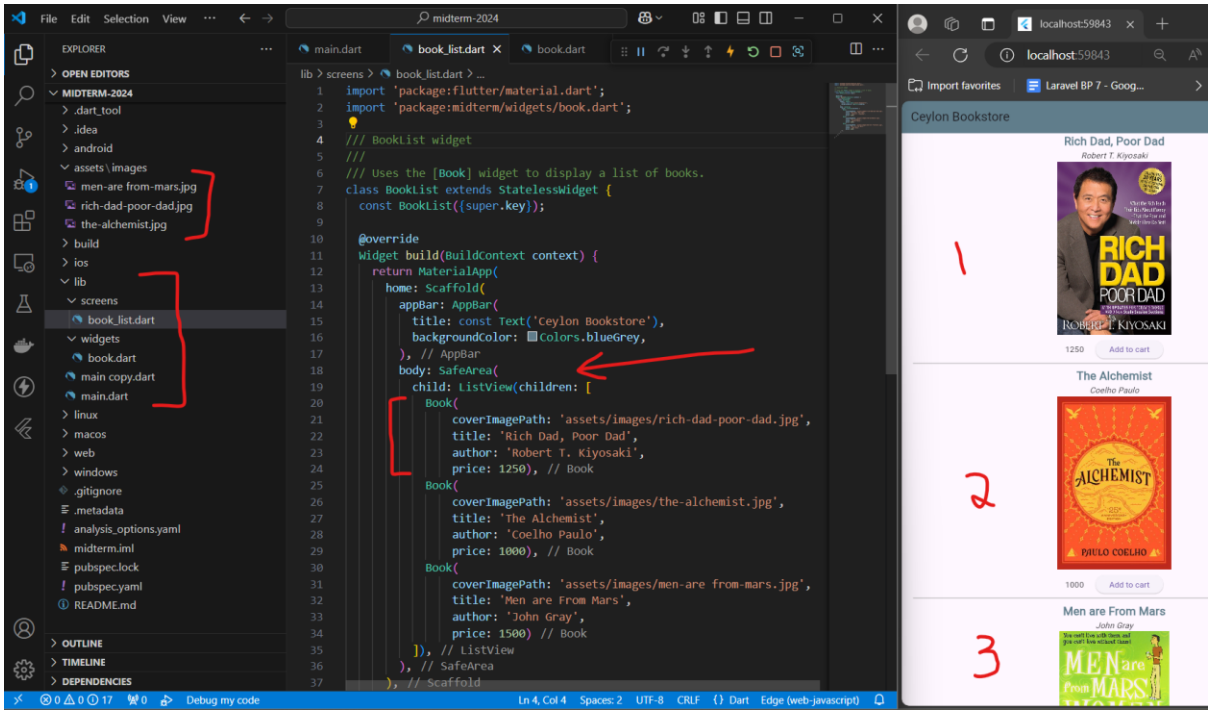
Exercise: Basic Flutter UI (total marks: 100)

1. Draw the widget hierarchy diagram (20 marks)
2. Create the widget as per the UIs given (20 marks)
3. Create 3 child elements (objects) from the widget you created (20 marks)
4. Copy paste the code to your document (10 marks) or include a repo link
5. Copy paste the UI to your document (10 marks)
6. Write a note on lessons learned and your rationale for selecting components from the library (20 marks)
7. ZIP the whole project folder along with this document and upload to LMS

Sample code with a list of 3 books

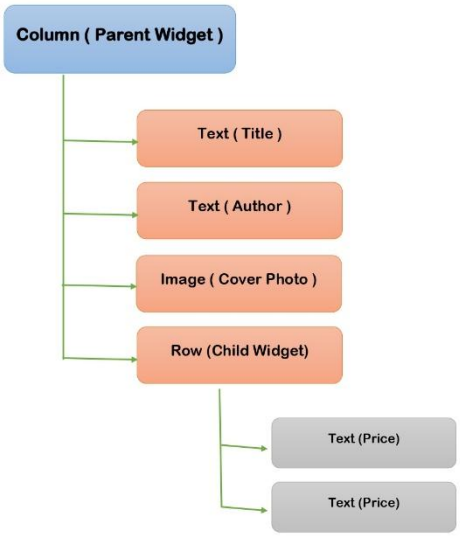
Read the documentation here:

- https://docs.flutter.dev/ui/layout
- https://docs.flutter.dev/ui/layout/tutorial
- https://medium.com/flutter-community/flutter-layout-cheat-sheet-5363348d037e
- https://www.flutterbeads.com/set-background-image-in-flutter/
- https://www.flutterbeads.com/flutter-position-widget-in-stack/



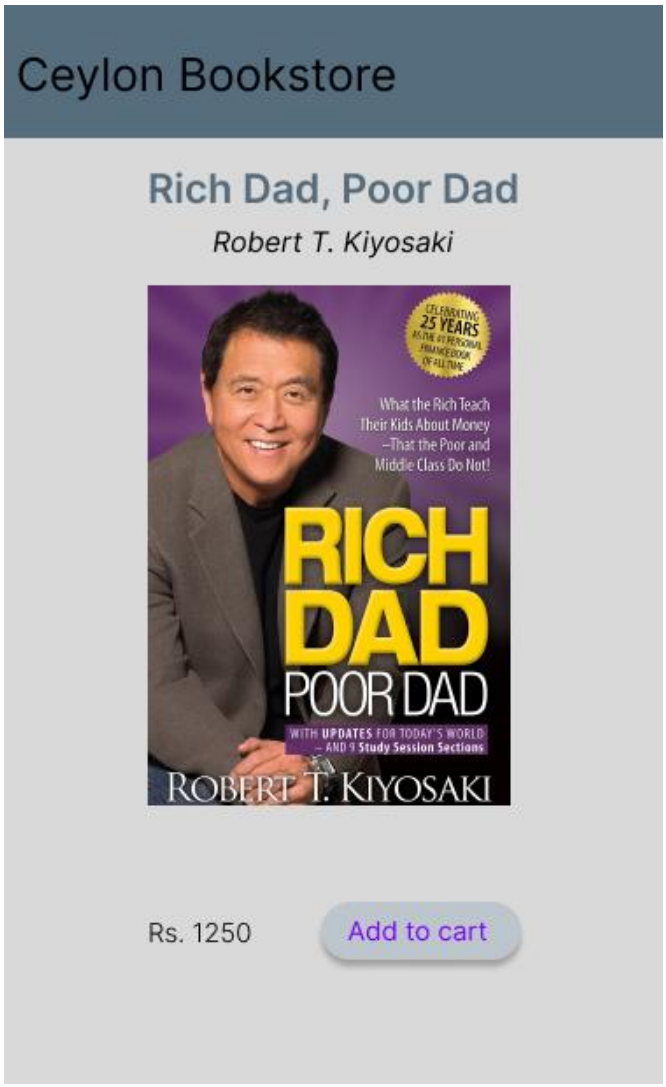
Paste the diagram & repo link here

You may post the architecture diagram here with a link to GitHub repo.



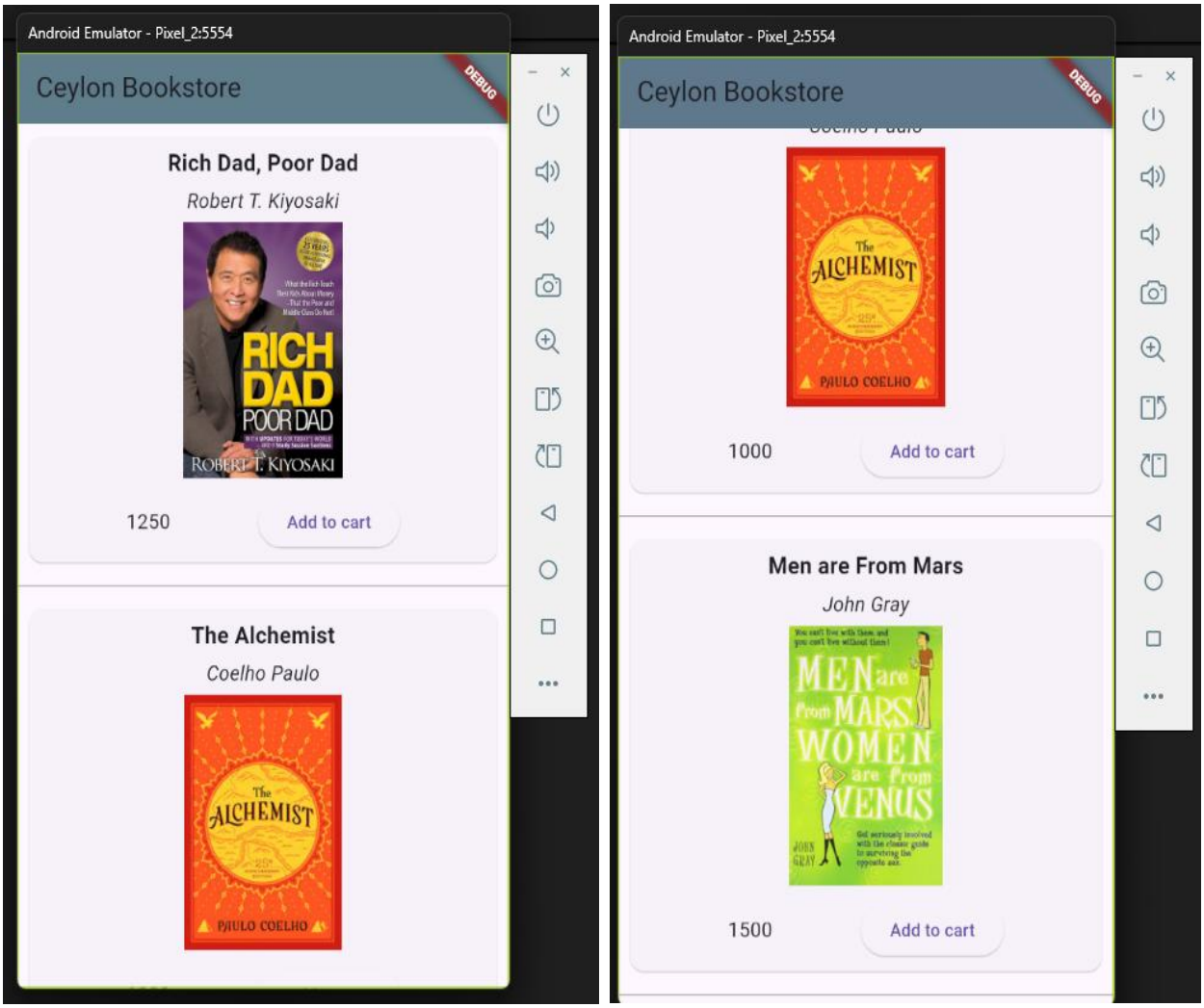
Paste the UI here

a. Figma prototype



Link - https://www.figma.com/design/WnU2Mvc30If2AAF2QiE4q6/Untitled?node-id=0-1&p=f&t=ik6ZtcycxTkA9ZCj-0

b. Final Flutter UIs



GitHub Repo Link - [widushan/Mid-Term-Project: CCS3351 Mobile Application Development]

Lessons learnt

Notes on what you learnt in doing this activity.

1. Flutter Widget Hierarchy and Composition

- **Custom Widgets**: Creating reusable custom widgets like the Book widget makes your code modular, cleaner, and easier to manage.
- **Separation of Concerns**: Dividing the UI into smaller components allows better reusability and maintainability.

---

2. Layout Management

- **Row and Column Usage**: You understood how to structure the layout effectively using Flutter's Row and Column widgets.
- **Spacing Adjustments**: Managing spacing between elements (SizedBox, padding) allowed you to fine-tune the appearance of your app.

---

3. Styling and Customization

- **UI Refinements**: Styling texts with properties like fontSize, fontWeight, and fontStyle helped improve the visual appeal.
- **Material Design Widgets**: Leveraging widgets like ElevatedButton for interactive elements aligned with Flutter’s Material Design.

---

4. Simplifying UI with Dividers

- **Separating Elements**: Adding Divider widgets gave a clear visual distinction between items without additional complexity.
- **Alternative Approaches**: Instead of using backgrounds (Card), you explored simpler ways to structure your layout.

---

5. Debugging and Iterative Design

- **Trial and Error**: Adjusting margins, spacing, and alignment to match the design taught the value of testing and iterating.
- **Attention to Detail**: Fine-tuning gaps, alignments, and scaling images for a polished UI emphasized the importance of small design details.

---

6. Flutter Asset Management

- **Image Loading**: Learned how to load local images using the assets folder and manage paths in the pubspec.yaml file.
- **Importance of Organization**: Keeping assets organized (e.g., /images) simplifies future development and maintenance.

---

7. Problem-Solving Skills

- **Design Refinements**: Identifying problems (e.g., background removal or spacing adjustments) and implementing efficient solutions helped develop problem-solving abilities.
- **Exploring Alternatives**: Comparing and choosing between widgets like Card and Container improved decision-making.

---

8. Understanding Flutter Basics

- **State Management**: Even though this app was primarily static, working with widgets is the first step toward understanding dynamic state management.
- **Widget Tree Principles**: Learned how parent-child relationships affect the structure and behavior of Flutter apps.