Technical Requirements – Circular Stock Tracker

Challenge is accessible on Memberspot: https://pl-coding.mymemberspot.io/library/jx3b7Qik9ip5qpNI8IF2/jmJLYnGZ2RljR6im5zVR/JFWndhOnbRP7Ph3Xtamx/details

🎭 Scenario

This mini-app simulates a product card screen featuring a circular indicator that shows how many discounted items remain in stock. Each time the user presses the Buy button, the remaining quantity decreases, and the circular indicator smoothly updates to reflect the change.

🎨 Figma Mockups

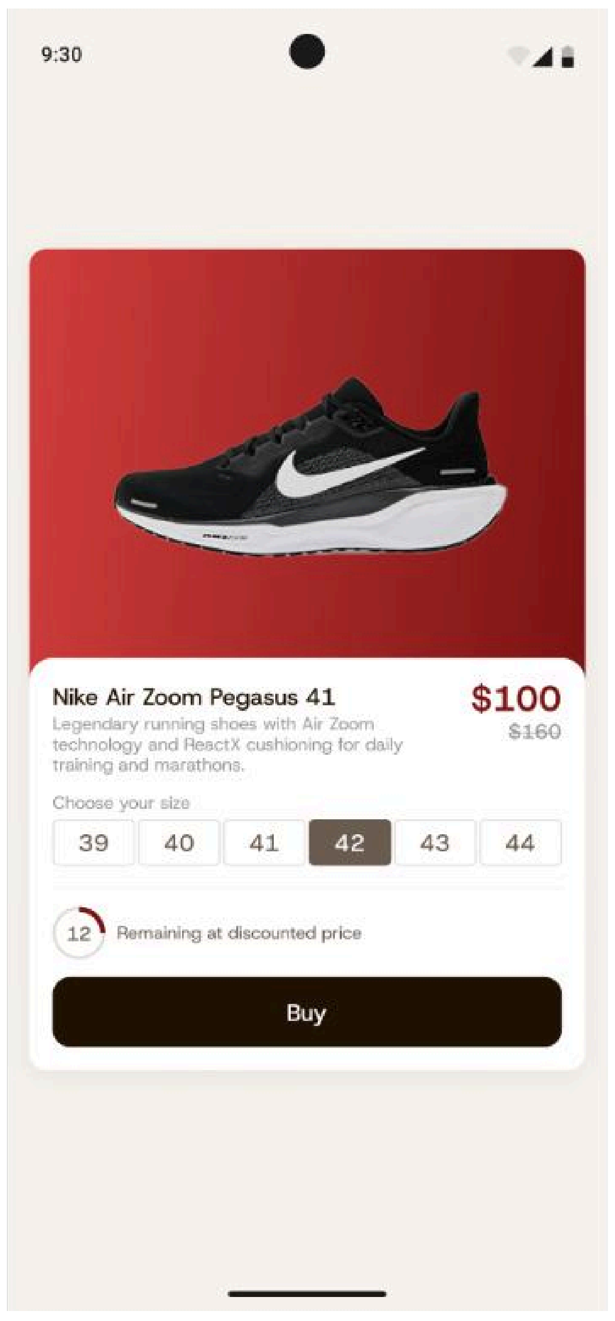
> https://www.figma.com/design/OUKoX7XUafdcBm3mqWdqiu/Black-Friday-Madness?node-id=2008-2186

🔤 Fonts - Host Grotesk

🎯 Feature Goal

Implement an interactive product card where a circular progress indicator displays the remaining discounted stock.

📌 Requirements



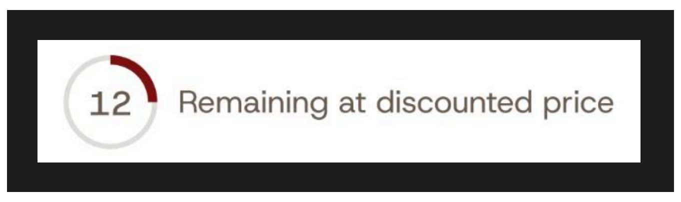
Legendary running shoes with Air Zoom technology and ReactX cushioning for daily training and marathons.

Product Card

- Includes:
  - Product image (Nike Air Zoom Pegasus 41).
  - Short description: *"Legendary running shoes with Air Zoom technology and ReactX cushioning for daily training and marathons."*
  - Price:
    - **Discounted:** highlighted in red ($100)
    - **Original:** crossed out ($160)
  - Size selection block (decorative, **non-functional**): 39, 40, 41, 42 (selected), 43, 44 — selection between sizes does not need to be implemented.
  - The product card has two visual states:
    - **Active state** — regular appearance with full-color product image.

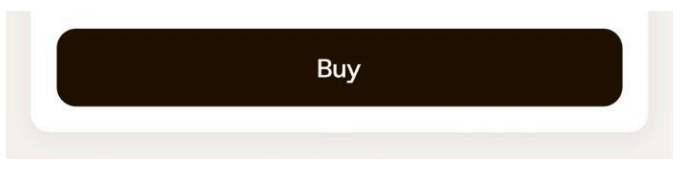
  - **Out-of-stock state** — product image background turns light gray, matching the disabled Buy button style.

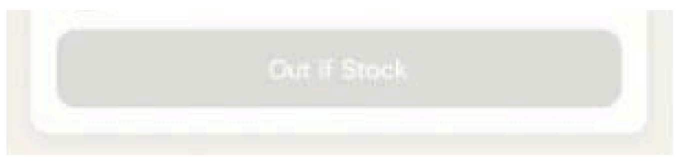
Circular Stock Indicator



- Implemented using **CircularProgressIndicator**.
- Displays:
  - Remaining quantity inside the circle.
  - Label *“Remaining at discounted price*” positioned **next to** the circle.
- When the **Buy** button is pressed:
  - The progress smoothly animates to the new value.
  - The number briefly scales up (bounce effect) and returns to its original size.
- When **stock = 0**:
  - The circle becomes empty.
  - The number shows “0”.

Buy Button





- **Active state**: dark-colored button labeled “Buy”.
- **Disabled state**: gray button labeled “Out of stock”.
- On press:
  - Decreases the remaining stock by 1.
  - Updates the circular indicator and number accordingly.

⚙️ Logic

1. The screen opens with a product card displaying an initial stock (e.g., 12).
2. Each time the user taps “Buy”, the stock decreases by 1, and the circular progress animates smoothly.
3. The number inside the circle briefly increases in size (bounce effect) and then returns to normal.
4. When **stock reaches 0**, the indicator becomes empty, and the Buy button changes to a disabled state with the label “Out of stock”. Additionally, the product image background changes to a light gray tone to visually indicate that the item is no longer available.

🤔 What's Allowed?

- Standard Android / Jetpack libraries.
- Use Material 3 components or simple custom UI elements.
- No 3rd party libraries are allowed or would be required to complete this challenge

⚠️ What's not important

- Actual shopping cart or purchase logic.

- Data persistence (no database or ViewModel required).
- Complex or physics-based animations.
- Dark mode or tablet adaptation.

🔗 Useful Links for This Challenge

- Progress indicators
- Animations in Compose
- Stateful vs. Stateless Composables
- State Hoisting in Compose
- Managing State in Jetpack Compose (Codelab)

🏆 Submission & Rewards

- Successfully submitting this challenge via the **/submit-challenge** command on Discord grants you **100 XP**.
- Your submission must include:
  a. A **Gist link** with your implementation.
  b. A **screen recording** (max 20 seconds) showing:
     - The initial state with available stock (e.g., 12).
     - Pressing **Buy** — the quantity decreases, progress animates smoothly, and the number scales up briefly.
     - When **stock reaches 0** — the indicator becomes empty and the button shows *"Out of stock"*.