Technical Requirements – Parcel Pigeon Race

Challenge is accessible on Memberspot: https://pl-coding.mymemberspot.io/library/jx3b7Qik9ip5qpNI8IF2/g98LzeMNxdeqdhGDiYMn/m2xbRQtPjICDp2nfX0IK/details

🎭 Scenario

This challenge simulates parallel downloading of six images to test concurrency, show individual progress, and measure actual time and size for each task. The user can restart the process and observe different results each time.

🎨 Figma Mockups

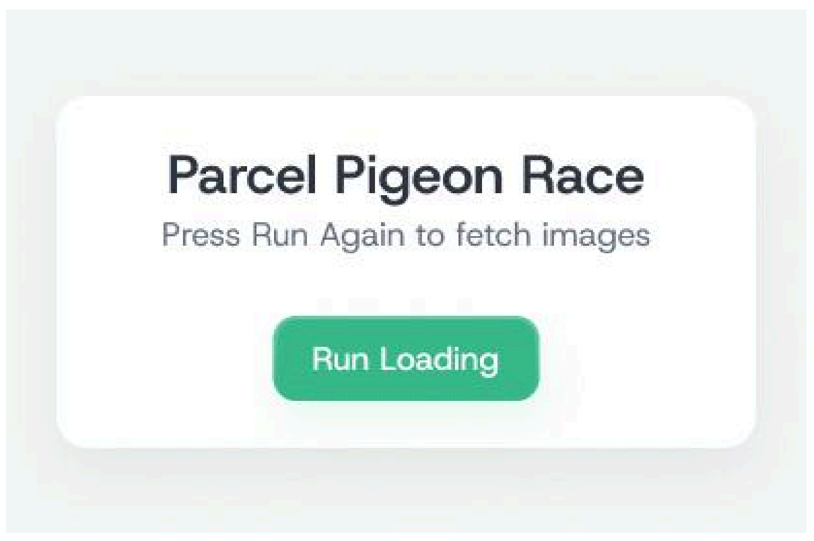
> https://www.figma.com/design/SAX4hR2rhXFUZgQjm1iZXF/Async-Adventures?node-id=5-318

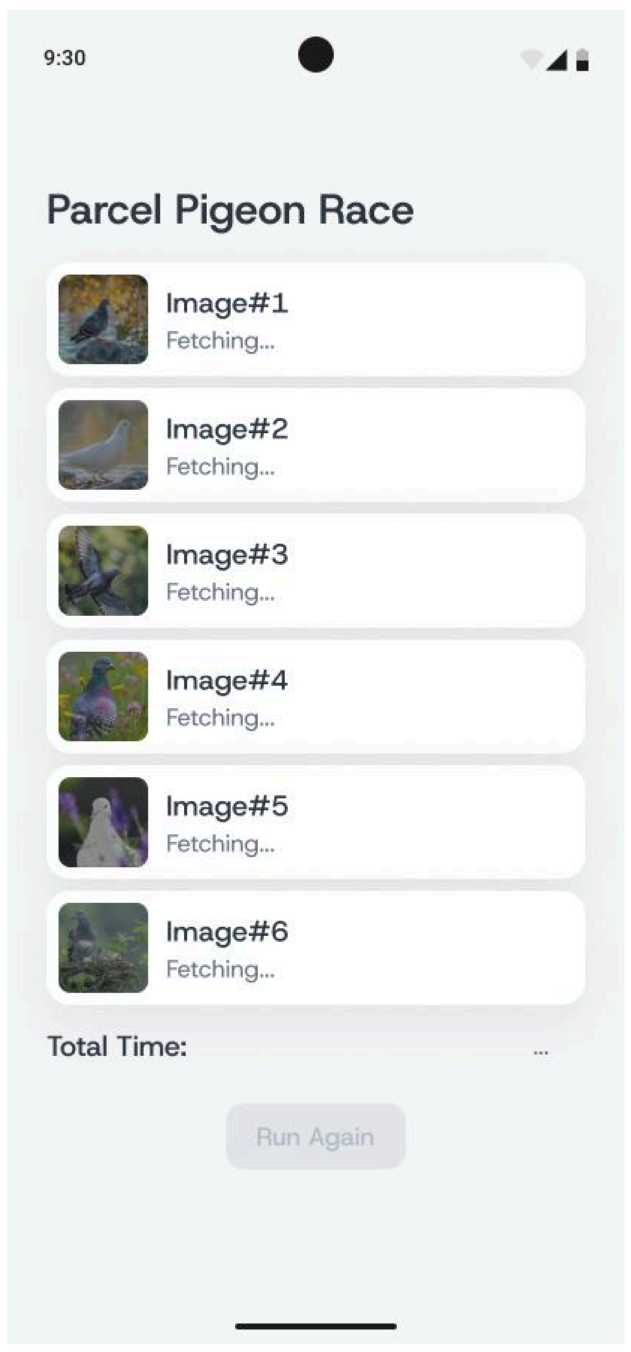
🎯 Feature Goal

Create a simple interface that displays the download progress of each image, shows final metrics, and indicates the total time it took to complete all tasks.

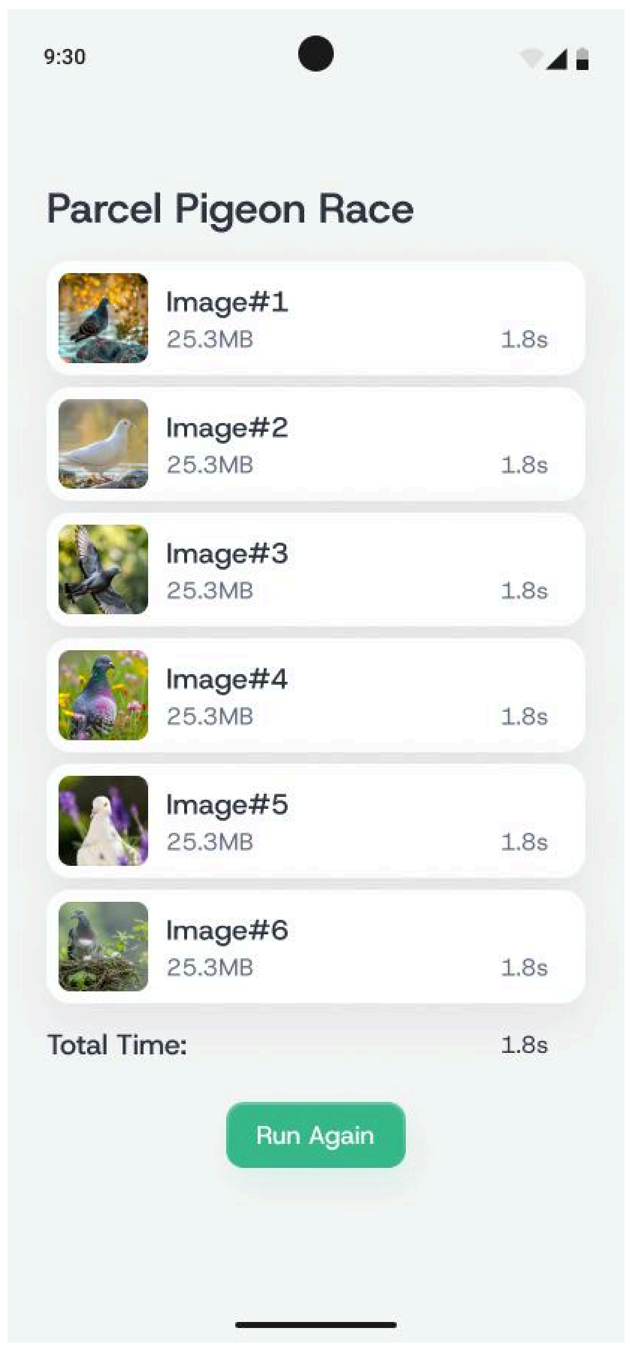
📌 Requirements

- In the initial (empty) state, a centered card is displayed with the challenge title, the message: "Press Run Again to fetch images" and a **Run Loading** button.
- Once the download starts, a list of six image entries appears, each initially labeled **Fetching...**
- The order of items in the list must remain fixed and reflect the launch order, not the completion order.





- When an image finishes downloading, its row is updated in-place to show:
  - a thumbnail of the image,
  - the download duration in seconds (e.g., 1.8s),
  - and the file size in megabytes (e.g., 2.3MB).
- After all downloads complete, the **Total Time** section displays the overall time taken — which should match the longest individual download.
- The **Run Again** button below the Total Time section restarts the process with a new set of random images.
- Downloads must run in parallel, independently, without waiting for one another to finish.
- Duration and size must be measured based on actual data transfer, not set manually.



🔢 Initial Data for Testing

- Each session must download **6 random images:**
  - 3 small square images from https://picsum.photos/200
  - 3 large square images from https://source.unsplash.com/random/800×800 or similar
- Image URLs must be generated dynamically on each run.

🤔 What's Allowed?

- Standard Android/Jetpack libraries
- You may use any appropriate image loading library, such as **Glide**, **Coil**, or **Picasso**.

⚠️ What's not important

- Full screen size or orientation adaptability
- Stability of image content or file size — full randomness is expected.
- Exact simulation of network latency — concurrency is more important than specific timings.

🏆 Submission & Rewards

- Successfully submitting this challenge via the /submit-challenge command on Discord grants you **100 XP.**
- Your submission must include:
  a. A link to a Gist with your implementation
  b. A screen recording (max 20 seconds) showing:
     - Tapping the start button
     - All items showing **Fetching...**

- Completion of items with thumbnails, duration, and size
- The result appearing in the **Total Time** section
- Restarting via the **Run Again** button.