Mobile Take-Home Test

Objective:

Build a simple application that demonstrates your ability to design clean UIs, write maintainable code, and use modern development practices. We're evaluating your architecture, code style, and ability to make thoughtful technical decisions.

Project Description:

Create an application that fetches data from the Rick and Morty API and displays a list of characters in a paginated manner.

Requirements:

Functional

- **Screen 1: Character List**
    - Display a list of loaded characters, with pagination (load 20 characters at a time).
        - Each list item should include:
            - Name
            - Image
            - Species
    - Implement searching functionality to allow users to search the list by character name.
    - Handle errors and empty states.
    - When a list item is tapped navigate to **Screen 2: Character Details.**
- **Screen 2: Character Details**
    - Provide a detailed view of a selected character, displaying the following information:
        - Name
        - Image
        - Species
        - Status

Technical

- **Technologies:**
    - iOS: Swift and SwiftUI.
    - Android: Kotlin and Jetpack Compose.
- **Testing:** Unit tests are a bonus.

Submission:

- Please push the source code to a public GitHub repository and share the link with us.
- Include a README file with:
    - Instructions for building and running the application.
    - A brief explanation of your architectural choices.
    - Any assumptions or decisions made during the test.

Evaluation criteria:

- **Functionality**:
    - Does the app meet all the specified requirements?
    - Does it behave as expected, with no crashes or bugs?
- **Code Quality**:
    - Is the code well-organized, easy to read, reusable, and maintainable?
    - Are there any compile-time warnings or runtime issues?
- **Architecture**: Is the app's architecture scalable?
- **Design Patterns**: Did you use appropriate design patterns to solve the problem at hand?
- **Git Understanding**:
    - Does the submission demonstrate an understanding of Git?
    - Are commits frequent, descriptive, and well-organized?
- **Testing**: Did you write valid unit tests? (bonus)

Good luck!