ACME Mobile Browser [New Grad]

Please read this before starting the Acme Mobile Browser project

By submitting the Acme Mobile Browser project, you will be considered for a position with our Mobile Browser team. Here are a few things we'll look for as we evaluate your project and as you go through our interview process:

- General knowledge of mobile application development for iOS
- Proficiency using Swift with SwiftUI and/or UIKit to build UI components
- Sense of visual design and attention to detail

Should you make it to the next stage, we ask that candidates have good critical thinking skills, a solid foundation in mobile technology, and a passion for UI development.

Please reach out to 2021-coding-project@neeva.co if you have questions or concerns prior to completing the project. We're looking forward to seeing what you make!

Problem

Build a simple web browser for the iOS operating system.

Requirements

PART 1: base app
Build a basic ***web browser UI*** that supports navigating the web on a mobile device. Your browser will be built on top of iOS's built-in WKWebView component. Your browser should provide a URL entry field, back, forward and reload buttons as well as a way to switch tabs.

As the user browses around the web, the URL entry field should be updated to reflect the URL of the currently loaded web page. The tab switching UI can be a simple list of URLs from which the user may choose a different tab to bring to the foreground.

PART 2: pick ONE of these "advanced" features to implement:
IMPORTANT: choose a feature that demonstrates a skill beyond what you've used to build the base app.

1. **Bookmarks**: add support for bookmarks, enabling the user to save a page to a list, with functions to add and remove from bookmarks as well as a UI to display all bookmarks.
2. **Thumbnails:** add thumbnails of webpages to the tab switcher to make it easier for the user to see what tab they are switching to.
3. **Suggest**: add support for suggesting URLs that match the user's input as they type into the URL entry field. This can be based on a fixed set of URLs or based on historical input into the URL entry field.
4. **Your choice**: another product feature that you want to design and implement that shows a similar level of complexity. Feel free to be creative!

IMPORTANT

- Your project must run successfully and have a README in the root folder with instructions on how to build and run the project.
- Your frontend code must be written in Swift. You're free to use SwiftUI and/or UIKit. Please do not use ComponentKit or other alternative frameworks such as Flutter or React Native.
    - *At Neeva, we're using a mixture of both SwiftUI and UIKit.*

Deliverables

1. Any and all code, tests, or mocks you write/create
2. A README that describes:
    a. How to run your project
    b. Any additional features you implemented
    c. Your approach to the product, including any design decisions or tradeoffs you made

The project must be loadable and buildable by Xcode and runnable through the Xcode simulator. You may assume the latest release version of Xcode and iOS.

How we evaluate the project

This project is designed to reflect the day-to-day work a mobile software engineer does at Neeva. As such, considering the user experience, the overall functionality, and the readability of code are important criteria. Here's how we evaluate the project:

- **Functionality**
  Does the app work smoothly? Are there no obvious or glaring bugs? For example, what happens if a user submits an empty string into the URL entry field?
- **Software engineering**
  Does your project show good understanding and use of your selected framework? Are you following good software engineering practices? Is the code logically structured and easy to read? Does the project show that you understand how to handle application state, respond to user input, and/or correctly utilize APIs?
- **Design**
  We're more interested in user experience than polished visual design. The functionality you choose to support should be easily discoverable by the user, and make sense without the need for help text, etc.