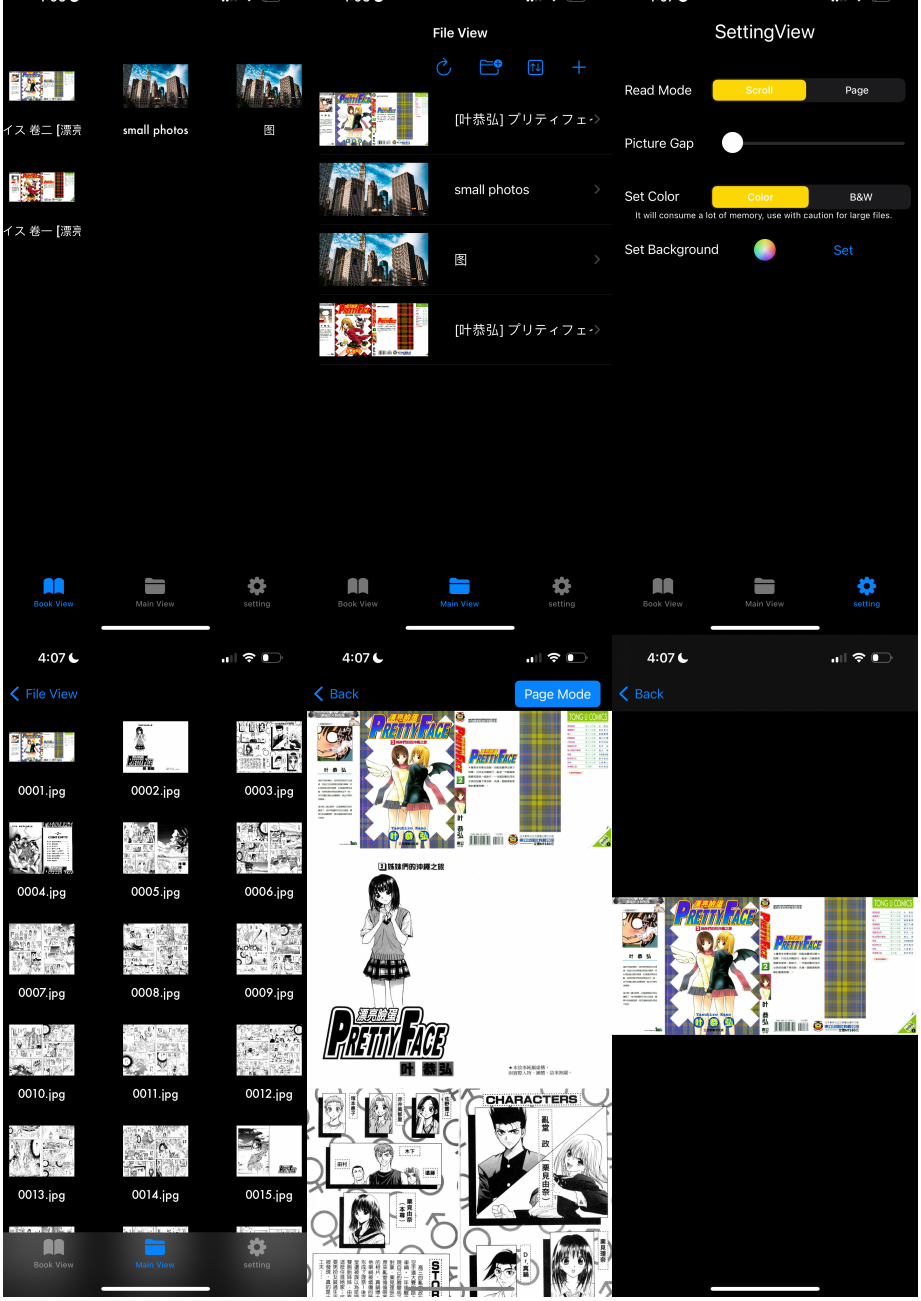
Final Project Documentation

A. Final Project Description and Documentation

Project Name: MyComicReader
Project Overview: This App aims to collect and read downloaded comic zip files or image zip files on the local device.
Detailed Description:
o Feature Description:
Implemented two different comic reading interfaces, allowing users to easily adjust the interface, including adding black and white filters to images and adjusting the spacing of the scrolling interface.

Implemented a simple file manager, allowing users to sort, rename, and delete comic folders.
Created a comic preview interface, allowing users to zoom in and out on images in the preview and reading interfaces.
Users can change the background color.
Integrated an open-source ZIP library based on the MIT license, supporting file decompression and saving files locally in order.
o Screenshots and Explanations:
Screen 1: Main interface, a collection view where users can browse all local comic covers and titles.
Screen 2: File management interface, where users can add, sort, create new folders, delete folders, and rename folders.
Screen 3: Settings interface, where users can make basic reading settings.
Screen 4: Preview interface, where users can freely select a comic and jump directly to the corresponding location to start reading.
Screen 5: Vertical scrolling reading interface, where users can double-tap to zoom in three times, and the top right corner can enter Screen 6.
Screen 6: Horizontal scrolling reading interface, where users can swipe left and right to read comics.



B. Final Project Discussion

API features and their applications:
Many different features and APIs were used in the project, including:
FileManager (not covered)
UIScrollView and Quartz 2D
UIImage filter functionality (not covered)

Multiple touch
Double touch
ZIP-Archive (not covered)

Challenges encountered and solutions during project development:

Initially, I didn't expect the biggest challenge to be switching between interfaces and passing information. I built a basic class and an object to handle all file operations, and there were many issues every time the interface switched. The switch between different interfaces was always accompanied by data transfer and modification, such as clicking on a picture to enter the corresponding position, and how changes in settings affect a specific interface, as well as how to allocate data storage and access for different interfaces, which was very difficult.
I solved this problem by using many variables in a public class to determine various settings and values, and although it took a lot of time, it was eventually resolved.

Application limitations and improvements:

I think my app is just a basic version (similar to an alpha version)
1.
I did a lot of testing, and my reading logic is that all comic pages are loaded when entering the reading interface, so a lot of memory is used to store these comics. I think if the file is too large, the reading will be slow (there will be latency, especially for older iPhones) and may cause crashes due to insufficient memory (especially for older iPhones with smaller memory). So, I can modify my comic loading interface based on this to make it a more dynamic process, saving some memory and opening faster.

2.
Currently, only zip files are supported for import, but support for other different file formats can be considered.
More editing options for images and files can also be added.
iOS SDK and Xcode limitations:

First, there is the limitation of language; many features that are familiar in Java or other languages often require looking up documentation and examples when using Swift.
Second, many available features (open-source packages) do not support the version of Xcode used in this programming task, so I had to search for suitable versions.
Moreover, while Xcode provides an intuitive and decent testing simulator, sometimes the results on the simulator and the actual device can differ. At first, I was able to import files

into the app by recording URLs in the simulator, but after installing it on the phone, it seemed that Apple did not grant me permission to directly copy entire files via the URL. Lastly, the relatively closed environment of the iPhone creates some inconveniences for development and testing. However, since I have not studied Android development, I do not have good examples for comparison. I just have this feeling, especially when it comes to file management, that it is clearly not as "transparent" as Android. While this is good for users, it poses a significant obstacle for developers during debugging.