ULife

Introduction

In the modern era, people bring their phone with them no matter where they go. When something remarkable or important needs to be recorded, the easiest way is to take notes on their phone. That’s why we designed our app, ULife, to help people mark down and take notes in ease.

ULife is an app designed for the users to keep track of their life. The app is thoughts-oriented and focuses on the textualization of any ideas that pop into your mind at the moment the app is opened. There are three major components of the app: diary, to-do list, and notes. Each of them is designed for specific needs and represents a part of a person's daily life. We carefully designed customized logic flows and interfaces for each component, connecting three parts by gestures.

The main design philosophy of our app is intuitive. For instance, our app only allows jotting down one diary per day, which works just like the natural and old-fashioned way of writing diaries but blending in with the modern ways of recording. Other parts of the app also obey the principle of being intuitive and fluid by adopting gestures as the main way of communication inside the app. The app is also capable of grouping your content, so that you know exactly what happens on a certain date. Therefore, we can confidently say that we don’t break any usual habits that people already cultivated for recording; the mission of the app is to ensure everyone is offered with the help from the contemporary world to fulfill their textual demands in an intuitive way, as previously stated. ULife is supposed and will always be about you.

Goals, functionality, and logic

The goal of our app will be broken down into the following four categories: To-do list, Diary, Notes, and UI Design.

1. **To-do list**

We aim to design a to-do list feature that can be easily accessed from the app so that the user would be able to create new to-dos or cross out what has been already finished.

2. **Diary**

We want to make the diary feature to be as personal and realistic as possible, and keep the usual habit that one person would have when writing a diary on paper. As the main purpose of our app is to keep track of people's daily life, the app feature will be at the center of the app.

3. **Notes**

The notes section is for the user to write down the sudden thought or brilliant ideas as freely as they want. Hence we want to allow users to take multiple notes per day, and mark the important notes as favorites so that they can filter to see them if needed. In addition, since we allow users to take multiple notes per day, the notes might become hard to find in the future. We want to also create a feature that users can use to navigate to specific dates to view the notes.

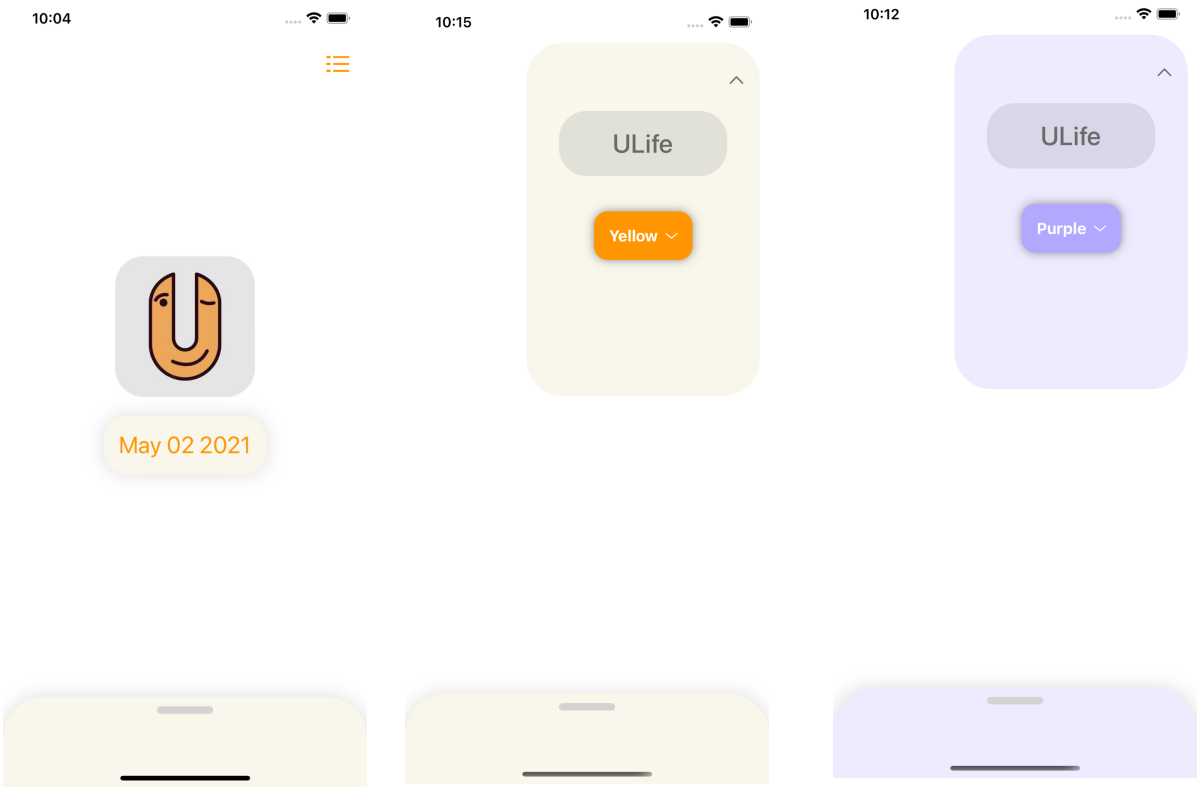
4. **UI Design**

To enable users to write down their ideas as fast as they can, we aim to design our UI to be easily navigable and highly supported by gestures. This part will be discussed more thoroughly in the Visual Interaction section.

User interactions

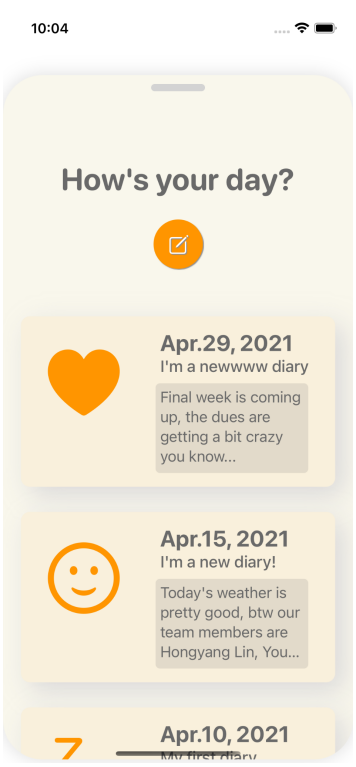
Home Page

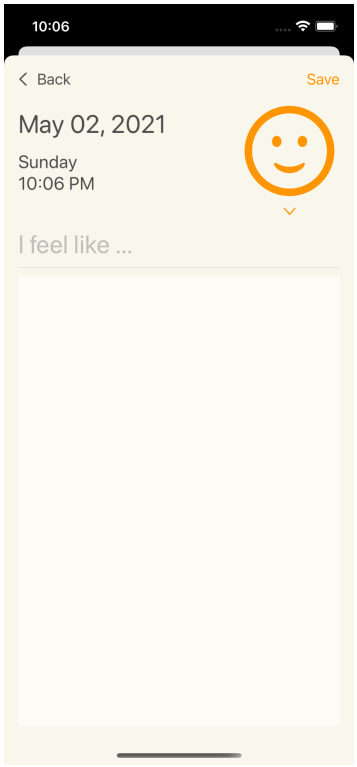
1. When you first start the app, you will see the home page of our app with the logo and current date in the middle.
2. The upper right corner is the app setting panel,
3. The drop down button, allows the user to switch the color theme for this app.

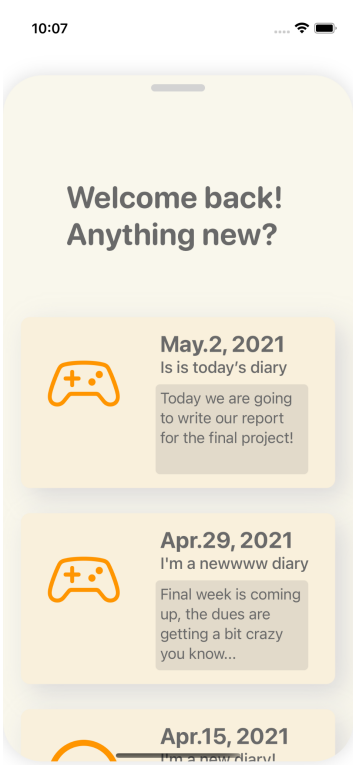


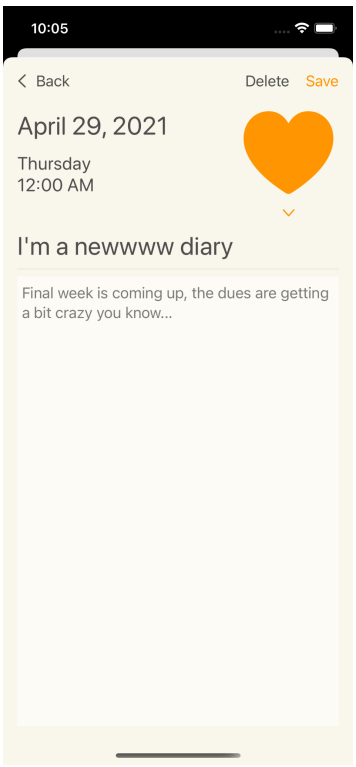
Diary

1. To enter the diary section of this app, swipe up the card at the bottom of the screen.
2. When you haven't written a diary for today yet, you will be able to click the center button to write a new diary. You can edit the title, content, and mode. After that simply click save.
3. Then you will be directed back to the diary page. And notice that there's no more option for writing a diary since you are only allowed to write one diary per day.
4. To view the existing diaries, simply click on the specific diary and you can modify, save, or delete this diary.



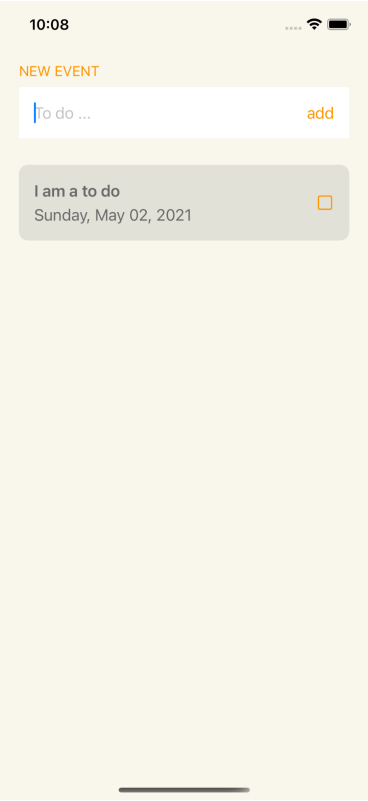


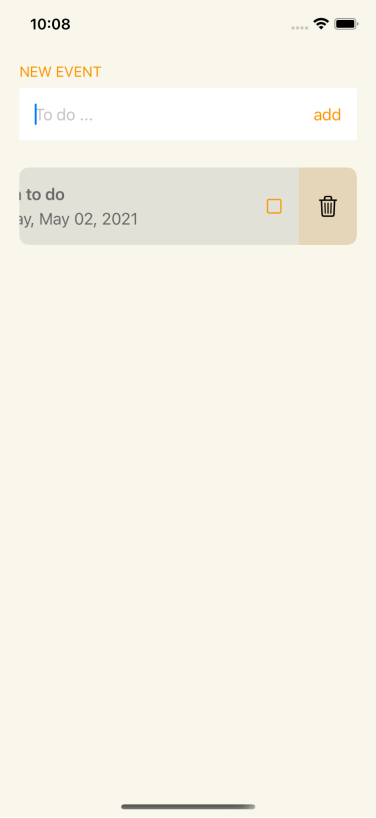


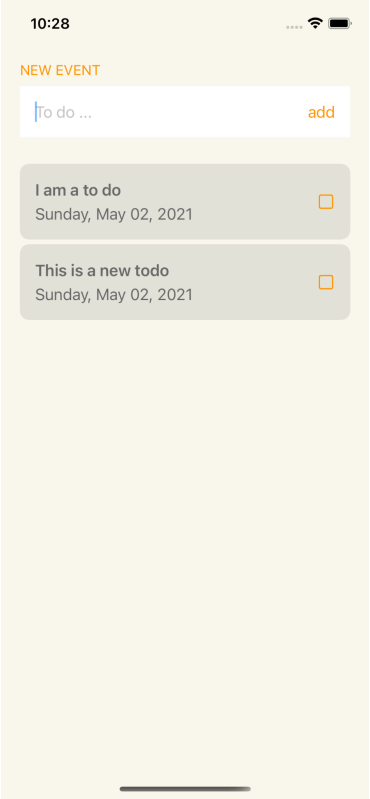


To-do List

5. To enter the diary section of this app, swipe down to go back to the home screen and swipe right from the left edge of the screen.
6. When you will be able to see the current to-dos, check off the task, or delete the to-do.
7. To create a new to-do, type your task and click the add button on the right side.

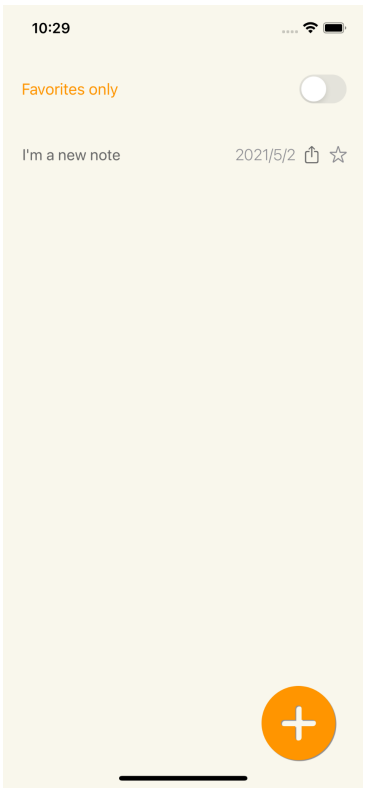


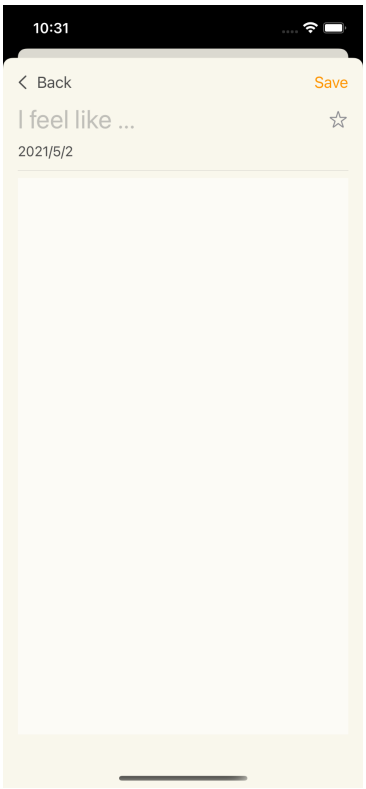


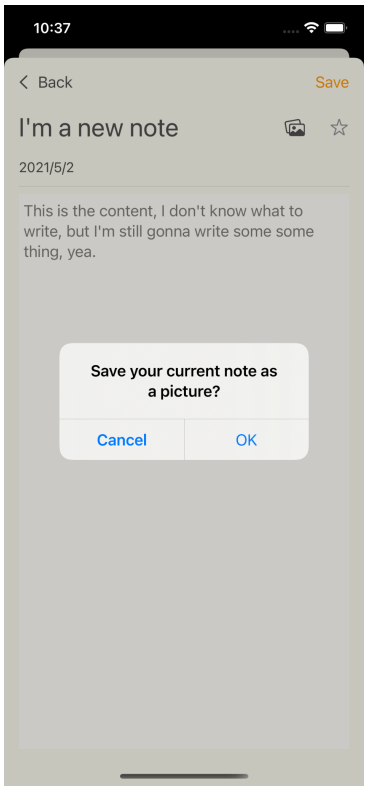


Notes

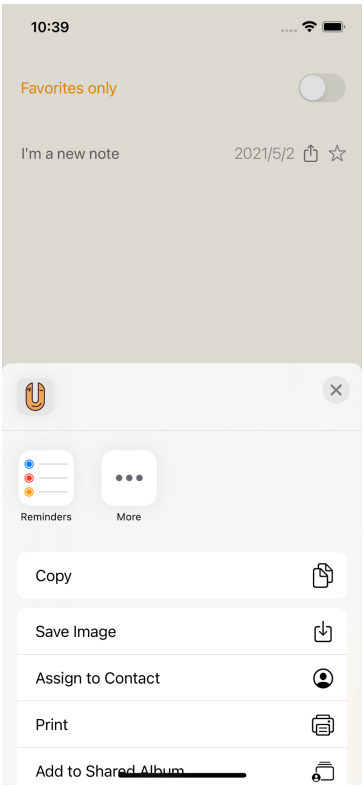
8. To enter the note section of this app, swipe from the right edge one time to go back to the home page, and swipe again to go to the note section. Now you can see your existing notes.
9. You can create a note by clicking the bottom right floating button, and you can edit the title content, and add the note to your favorite. After that simply click save.
10. To modify your existing notes, click on the row of the note, and you will be able to modify it.
11. You can also save an image from the current note by clicking the image button left to the star button. Then an alert will pop up asking if you want to save a picture.

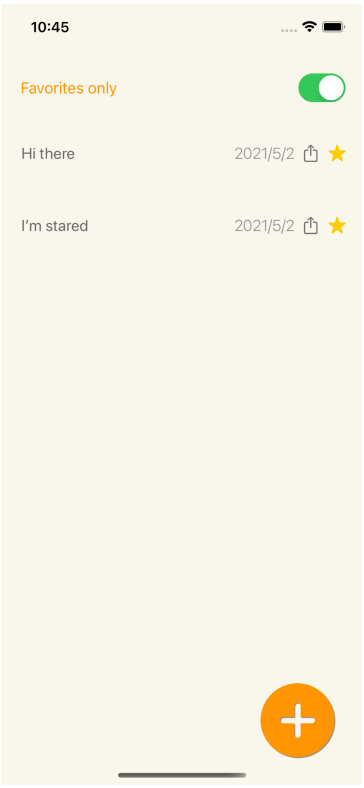


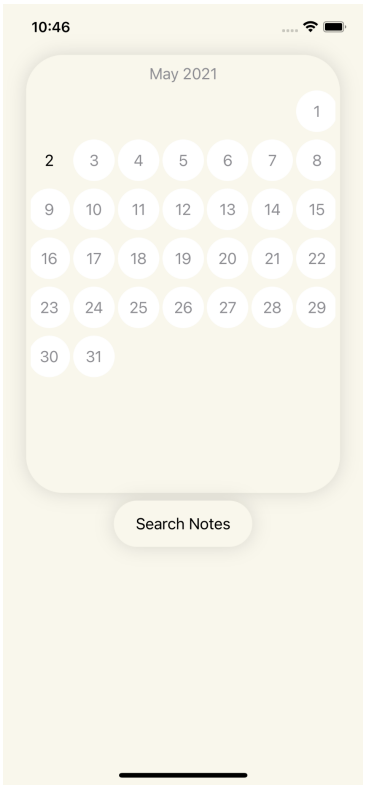




12. Go back to the note page and you can share your saved image by clicking the share button left to the star button.
13. You have the option to see the favorite only (or starred only) notes. Toggle the button to see only your favoirated notes.
14. You also have the option to view only the notes on a specific day. To do this, swipe down from the top of the screen, there will be a calendar popup window, where you can choose to view the notes on a specific day.







Thinking process

Stage 1 - setup

Here is a detailed breakdown of our thinking process. Before Milestone1, the major work for us was the general app setup. The first and foremost task was to determine the main panel that holds the core of our app. We decided to use a slide-card as the main panel to contain the most important component, and the gesture of slide-up was intuitive enough for mobile devices users that have been trained for more than a decade.

We initially chose Navigation View as our default method of switching views. Moreover, we decided to delete an item by swiping the custom list to the left. The initial demo when in milestone1 wasn't decent enough for our goal, but it basically achieved the functionality of adding and deleting events. One of the difficulties was to find the suitable color and modify the color of the background. The slide-up card was implemented by using offset and ZStack, and we were not quite familiar with these properties at that time and usually experienced UI constraint warning. Besides, we also spent quite a period of time finding usable tutorials and guidance online, as well as getting our hands dirty to try out different possibilities of Swiftui. Learning knowledge ahead of the class was virtually how things were during the initial stage of developing the app.

Here are two important initial setup:

Setup of ToDoEvent - First, events should have two basic attributes, content of events and the time of creation. So the struct we originally designed did not have the id attribute, which means that it did not implement the identifiable protocol. This caused us to be unable to import

the event list when using list or ForEach. Then we learned that SwiftUI requires the elements in the list to implement the Identifiable protocol while importing a list. Later, we added the id attribute and the identifiable protocol to them. At this time, we decided to use the number of elements in the event list as the id of the new event. Later, we found that there are several constraints when applying style onto the Swift List. Instead, we defined our own custom list that allows the same functionality and provides more freedom on creating more aesthetically appealing components.

Setup of Note - When we started designing the note section of our app, we first laid out the overall view structure of this section. The overall view contains a NoteList view, the NoteList have numerous NoteRow views listed, each NoteRow will be able to direct to a NoteDetail view to see and modify the notes, and also a extra button for creating a new note which directs to the NoteNew view. After that, we decided what are the essential components that need to be stored in a note, and defined our NoteType for storing notes into our model data. With a clear structure of the NoteType model early on helped us to implement the feature of viewing the favorite-only notes. Only a filter on the notes were needed to be able to separate the favorite and unfavorite notes.

The initial implementation of the NoteList was to embed NoteRows into a navigationView, and each NoteRow contains a corresponding navigation Link. After a few weeks of working on this structure, we found that navigationView can only transition in from the right side, and also there are difficulties when trying to customize the background and navigation bars. So again, we decided to swap out the implementation with the navigationView. We created another version of NoteList which utilized our own custom list that we implemented for the

to-do list and the action sheet to allow the NoteDetail to pop up. Implementing custom components has helped us create and design our desired features.

Stage 2 - calendar

Including a calendar that allows the users to view notes of a certain date is listed as the most important stretch goal in our initial proposal. We had a rough time starting implementing the functionality of aggregation and search, since we never learned it in the class. We did research and learned a lot of knowledge related to Date, including DateComponent and DateFormatter. However, most information and references were based on UIKit, which is outdated. Their ideas are worth learning.

At first, we thought that DatePicker could be used to make a calendar, but later we found that datePicker can only pick a date. Even if the calendar can be displayed, it cannot add a navigationLink to a specific date, so this idea is unsuccessful.The method we later used was to treat the calendar as a list of years. Each year is a list of months. Each month is a list of weeks, and each week is a list of days. A day is just a text but After creating these lists, you can use LazyVGrid to implement a calendar. Using Navigation Link to go to the package on a specific day and you can click on that day to jump to that day, thereby displaying information related to that day, such as note.

After we implemented Calendar, we allocated a temporary location for its entry. At that time, we were not sure what form would be the ultimate way for the users to access these grouped notes. We also made some improvements on app logics and animations related to calendar and other parts of the app.

Stage 3 - File manager, diary setup, and dropdown menu

At this point, we should make sure that what the user enters stays within the app. We want to store a list of objects instead of some simple user settings, and we don't need to transfer data to the outside. Therefore, we chose to convert the event into PropertyList data, and then use FileManager to store it.

Because our task at this time was to store note data in addition to storing event data, we created a class specifically to save and load data, which is equivalent to establishing a data manager. Then we establish an instance of SaveLoadData class in modelData. The addEvent function invokes the functionality to implement data storage of SaveLoadData.

We also decided that it's reasonable to put calendar in a dropdown menu, and this would make the interface more clean and compact. Moreover, we replace the content in the slide-up card to the initial setup of the diary instead of notes. Then, the slide-up card officially started to hold the most important content of the app.

Final Stage - delete files, diary, and share panel

We first ensure that all files we store into FileManager are able to be removed. Delete event is supposed to be a very simple thing, just delete an event in the eventList, and then save an eventList again. However, because we used the number of elements in the event list as the id while doing Struct ToDoEvent in milestone1. When we tried to delete an event, it would cause the program to crash directly. The reason for the crash is that the index is out of range. To figure out this, we modified the id, using UUID as the new id. But the problem is still not solved, the program still crashes directly. At this time, we modified the logic. We used to delete a certain event from the eventList immediately before, but now we did not delete them immediately but to

change the hasDeleted property of this event to true, and then added an if statement to ForEach. If the hasDeleted property of the event is true, it won't work. Every time the program is started and modelData is initialized, I delete the event whose hasDeleted attribute is true and update the data. As a result, this new logic perfectly implements the event deletion function.

A feature we then decided to implement beyond class content is the share panel. Users should be able to share their notes as an image to other apps or simply save into their local album. This just makes the whole app more connected to the whole ecosystem of the users' iPhone.

We also added a feature taking advantage of the mobility of the phone. User can easily switch between the theme color of the app by shaking their phone. This is being achieved by detecting the shaking gesture from our own custom extension.

For the final implementation of the diary, instead of having simple lists like the to-do list and notes, we designed a card for previewing the content of each diary. Inside of each diary, the user has the option to select their mood icon for this specific day, which is shown in both preview and detail of the diary. To make the one-diary-per-day feature to be more intuitive, the welcome text is changed to be able to update based on whether the user has taken a note or not today. The logics and interfaces of the diary are now completed - all of them should come out from the bottom part of the screen, just like the slide-up card direction. Using an actionsheet was deemed by us as the appropriate way to handle the content of a diary. At this point, we also clear some redundant animations and finally wrap up three core parts of our app.

Collaboration

Throughout this project, we have learned more than just how to design and write a functional app. More importantly, this project has allowed us to practice our communication skills and basic principles of software engineering from the team collaboration.

1. Project Ideation

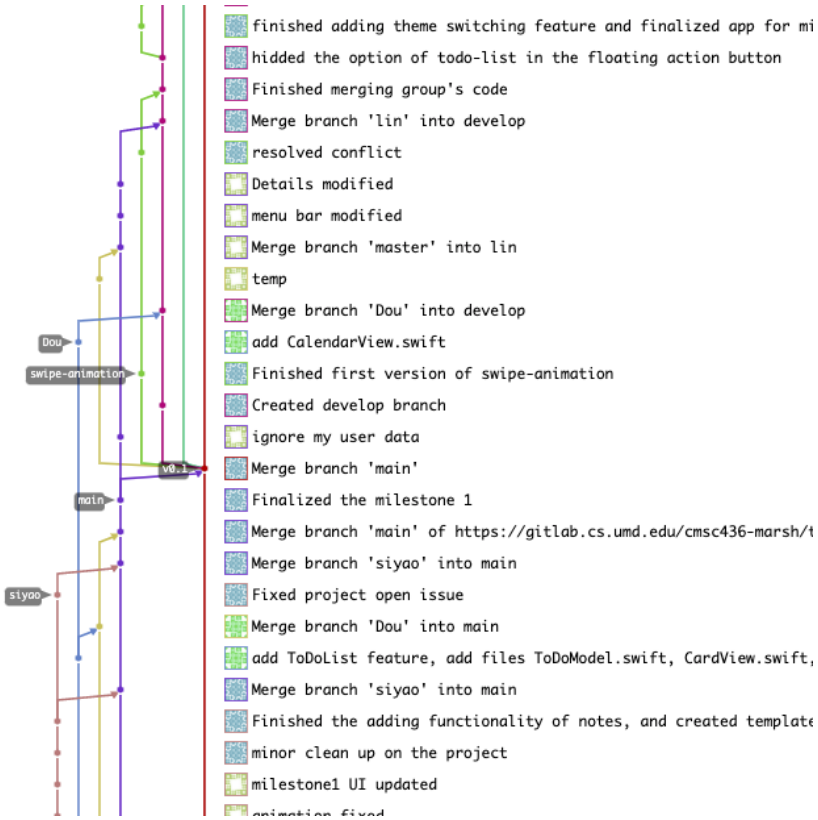
   In the very early stage after the team formation, our team has decided to each brainstorm several potential project ideas for this project. We were able to come up with more than five project ideas based on our own needs in daily life. We decided to implement ULife as our final project, since this is a project that we all wanted to have in real life and we want to implement this app so that we can feel comfortable using it by ourselves.

2. Agile & Scrum Meetings

   To maximize the productivity for this project, we have decided to follow the agile methodology and planned to create features as we progress each milestone. At the beginning of each sprint, we had sprint meetings discussing the sprint expectations and work splitting. Most of the cases, we assign each three of our team members an independent task so that each person can work on their part without having to wait for other people's code. With only a few exceptions such as UI design, and navigation gesture, which may need one person to integrate the parts together to ensure the consistency. During the sprints, we have weekly scrum meetings to talk about what has been accomplished and what the issues impeded us. We ensured immediate communication when issues happen so that we can solve it together. Without such efficient communication, we probably won't be able to finish and deliver most of the features in the app at all.

3. Git

   Throughout this project, each of our team members was able to practice and familiarize with the commands and features of git.



   As you can see in the branch graph, we have frequently created feature branches for each task. We have followed the standard git architecture of merging frequently into the develop branch, and only merge production versions into the master branch.

Potential future directions

The more advanced search ability is what we plan to implement for this app in the future. So far, we have implemented a complete grouping of notes and make them accessible through a calendar interface. Our goal is to achieve the global level of search, which would include to-do

events, notes, and most importantly, diaries in an aggregated page that allows the users to absorb any information they need at a quick glance. Searching based on text from notes and diary titles is hard to achieve, but we have implemented the search entry and logic for them. We all have confidence to raise the search experience to a whole new level.

Another feature that would make a significant difference in our app is the integration of a weather icon. We are considering adding it to the main interface both for aesthetic and functional consideration. The users should be aware of more about what is going on around them, and this undoubtedly includes weather information.

Although our app was designed as a text-based app, we could also consider the ability of storing images into the app in the future. Besides, adding voice memos are also a viable way of enriching the way for the users to keep track of themselves. However, we realized what we considered as the most important stretch goal in our initial proposal, which is allowing the users to view the notes on specific dates or months. Therefore, adding up other stretch goals to our future development checklist should not be an obstacle.

The most important part of our future directions is and will always be the perfection of app logic. Intuition is the most precious characteristic that everyone is born with, and it's the app designers' job to adapt their work with people's intuition. The most straightforward improvement of our app could be a starting animation, or an essential modification of the setting panel that would potentially fit in more content. There are other details of app animations that are worth contemplating: should a new view come out from bottom, left, right, or up? Have we ensured that all shapes conform to our design protocol rounded rectangle? Should we make more animations and transitions to asynchronous events to smooth the visual transition of the whole app? There are far more questions than what we can ask to make an app flawless and meet our

satisfactory standard. An app is more than code; instead, it is the complication of developers' temporary ideas, working ethics, and commitment to make an app that is beloved by the users.