Checkpoint 5: Making Apps for Everyone

1. Document the ways in which your app does not follow the Human Interface Guidelines. The more the better.

- **Timeline View:**
  - User cannot delete an entry directly from TimeLineView(User Manipulation)

- **Detailed View:**
  - The detailed view doesn't have a clear layout right now: the spaces between cover image*comment and search bar is a little bit large, also we may need to resize*rearrange the elements so the user doesn't need to swipe down to write down their memory.
  - The UITextFiled only supports single-line text display, which is not reasonable.

- **Add/Edit View:**
  - The textView now displays all the texts in a single line, which makes it hard for users to see & edit what they have typed in.
  - The keyboard will not disappear after user typed enter.
  - The "Click to upload custom image" could be more intuitive to use.
  - Users don't know whether a new entry has been successfully added or not.

2. For the most critical of the issues in part 1, propose how you might fix them.

- **Timeline View:**
  - Implement swipe to delete on tableView Cell (from timeline)

- **Detailed View:**
  - (Implemented) Use UITextView instead of UITextField. Create a frame for the textview, use the the defined behavior **scrollable and multiline but not editable**

- **Add/Edit View:**
  - Use UITextView(same as Detailed View)
  - Based on the research, maybe we can do resignFirstResponder in textView's shouldChangeTextIn() function?(need to validate on an iPhone)
  - Change the default image to an icon of "Camera"
  - (In Progress)Implement a notification(might use UIAlertController) to notify the user that whether the entry has been added/updated, if not, what is the potential problem.
    - For the potential problem, we may need to check whether the user's input is legal, and we also need to check if there is any issue when interacting with Core Data(and later on Cloud Kit)

3. Implement at least one of your proposals in part 2. Keep in mind that some of these issues could prevent your app from being accepted.
    - As stated above, we implemented the UITextView for avoiding edits in the `Detailed View` and implemented the swipe-to-delete feature on our `Timeline View`.

    Next, accessibility is a topic that Apple is very, very interested in, and for good reason: the number of potential users who you miss out on by not being accessible is large. Xcode and UIKit provide excellent tools for implementing accessibility, so you might as well take advantage of them.

    Investigate your app to see where it falls short for accessibility. Consider the following:

- Dynamic text
- VoiceOver
- Display Accommodations
- Size of elements in your app
- Color blindness

4. Document potential accessibility issues in your app.

    With a relatively simple UI by design, we do not seem to have issues with support for color blindness and display accommodations. VoiceOver appears to be supported for most of the elements that presents in our view, which doesn't seem to be an issue at this stage. However, we might need to make some adjustment to accommodate for dynamic text and size of elements. At current stage, we think it would be more beneficial to incorporate more considerations for element size in our upcoming UI adjustments.

5. For the most critical of the issues in part 1, propose how you might fix them.

- For element sizes:
    - Currently, some text fields and buttons appear smaller on some devices, and it might be a bit challenging for users to tap on them precisely. We will gently reorganize our UI and increase the size of those smaller elements.

- For font sizes:
    - Set a set of font sizes that changes as system font size changes.
    - As the App launches, we would check for the current system font size and choose the right font size for our App from the universal font size offset.

- **After some research we found out that there is a better way to implement this:**
  - Dynamic Type
  - This supports automatically adjusting font sizes and even element sizes with some simple settings:
    - **Font:**

```
// set the font text style to large title
label.font = UIFont.preferredFont(forTextStyle: .largeTitle)
// set the font to auto adjust to text size changes in Settings app
label.adjustsFontForContentSizeCategory = true
```

    - **Content:**

```
// YourViewController.swift

// the height constraint for the avatar image view
@IBOutlet weak var avatarHeightConstraint: NSLayoutConstraint!

override func traitCollectionDidChange(_ previousTraitCollection:
UITraitCollection?) {
  // if the headline text size increase by 35%, the avatar height will increase
by 35% too.
  let adjustedHeight = UIFontMetrics(forTextStyle: .headline).scaledValue(for:
75.0)
  avatarHeightConstraint.constant = adjustedHeight
}
```

6. Implement at least one of your proposals in part 2. Keep in mind that some of these issues could prevent your app from being accepted.

- We implemented the automatic font adjustment in our App using the method described in part 2 code block.