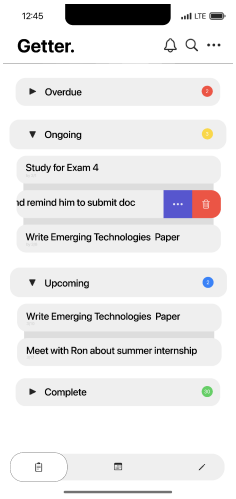
GETTER JOURNAL ONE

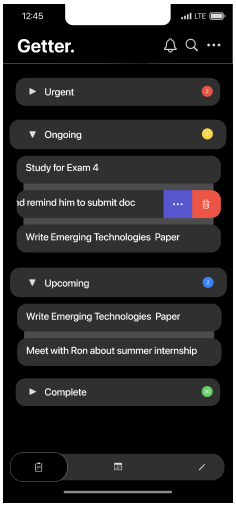
Dhruv Ruttala

https://github.com/karthikponnapalli/Getter-Repo.git

In the initial stages of development, we have been mainly working on the app flow and functionality. The app idea itself is a to do list app that organizes items and events based on start date rather than due date or end date. Since students normally organize their events based on due date, they often misallocate time and fail in being able to manage their calendars effectively.

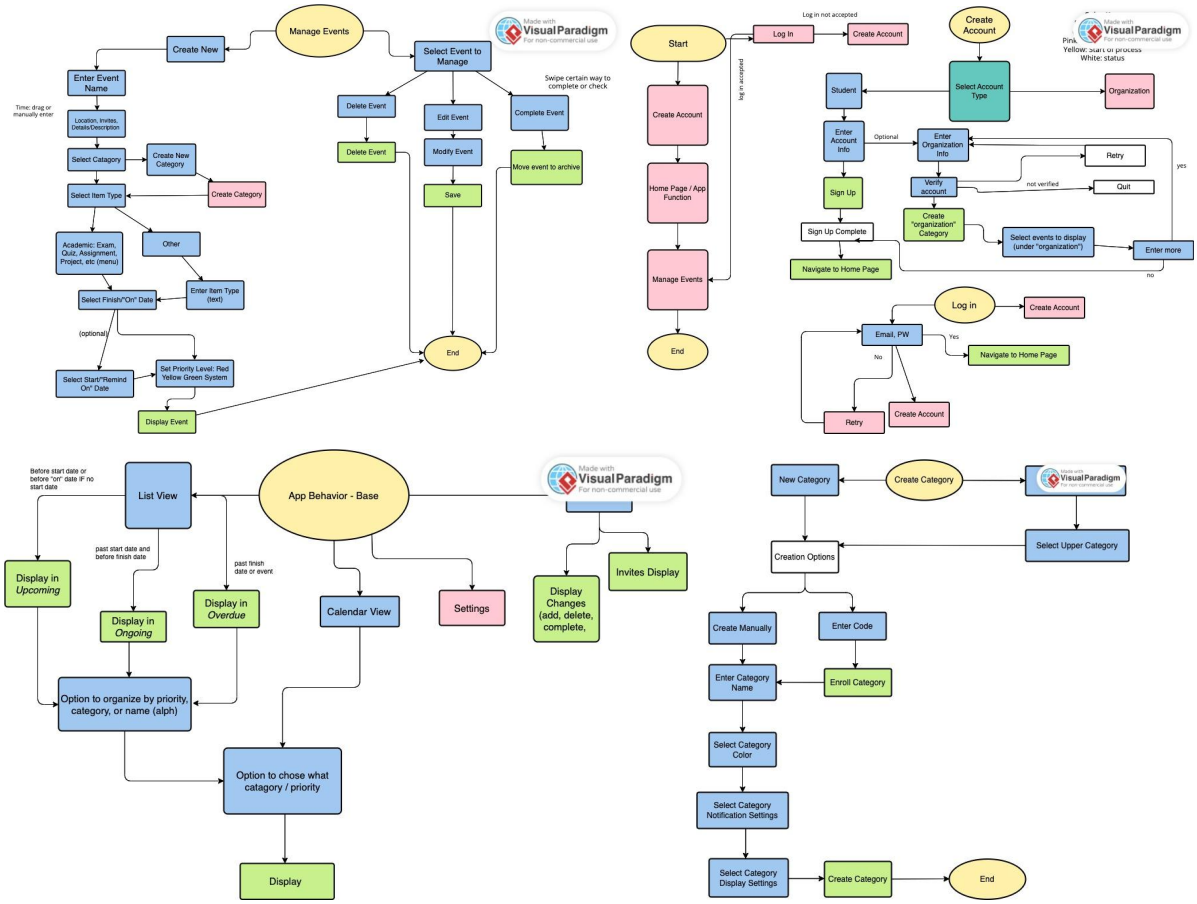
The functionality of the app will contain three main screens. The first is a list view, where items and events are organized in several menus based on the date they are due and the date they start. The second view is the calendar, which we are still developing the look and functionality of. It will be difficult to illustrate start and end dates on a calendar, since this view is normally used to present single events. The third view is a “new item” view which is the area of the app where new items are created. This view hasn’t yet been developed visually, but it will likely be the most simple menu to define in terms of the visual appeal.





Our current biggest obstacle will be dynamic databases that are located on devices so that users would be able to modify the information in the application natively. Since the app will be modifying the location of items with user actions, there has to be a database that users can modify through the app itself. The database will be one where the items themselves are stored, and modifying that database will change what items are shown in the app itself.

Below, I have attached some initial flow charts for how the functionality of the app would work. These are obviously subject to change, as they were created with the very simplest form of the app in mind with some features that have since been removed.



We will likely use a similar code that would refer to a JSON to define how items would show up in menus. Since the actual frame of each item would be the same, as seen in the images above of the first screen, we could use some kind of frame that just uses strings to define what goes in the frames.

```
ZStack{
    Rectangle()
        .foregroundColor(.white)
        .cornerRadius(10)
        .shadow(radius: 5)
        .aspectRatio(CGSize(width:335, height:175), contentMode: .fit)
    HStack{
        Image(image)
            .resizable()
            .frame(width:116, height:116)
            .clipShape(Circle())


        VStack(alignment:.leading, spacing:10){
            Text(title)
                .bold()
            Text(description)
                .padding(.bottom, 20)
            HStack{
                Image(systemName: "book.closed")
                    .resizable()
                    .frame(width:18, height:20)
                Text(count)
                    .font(.caption)
                Image(systemName: "clock")
                    .resizable()
                    .frame(width:20, height:20)
                Text(time)
                    .font(.caption)
            }
        }

    }
}
.padding(.horizontal, 20)
```

```
LazyVStack{
    ForEach(model.modules){module in
        HomeViewRow(image:
            module.content.image, title:
            "Learn \(module.category)",
            description:
            module.content.description,
            count:
            "\(module.content.lessons.count)
            Lessons", time:
            module.content.time)
        Spacer()
            .padding(.bottom)
```

We would be able to refer to a JSON file just like we did in class to use a predefined structure for the frames of each item and then be able to access the json files that would be modified based on user inputs. This would allow the frames to be populated with information that the user would input. I'm not actually sure if this would work perfectly, or if there are better ways to do this, but the initial idea works like this.