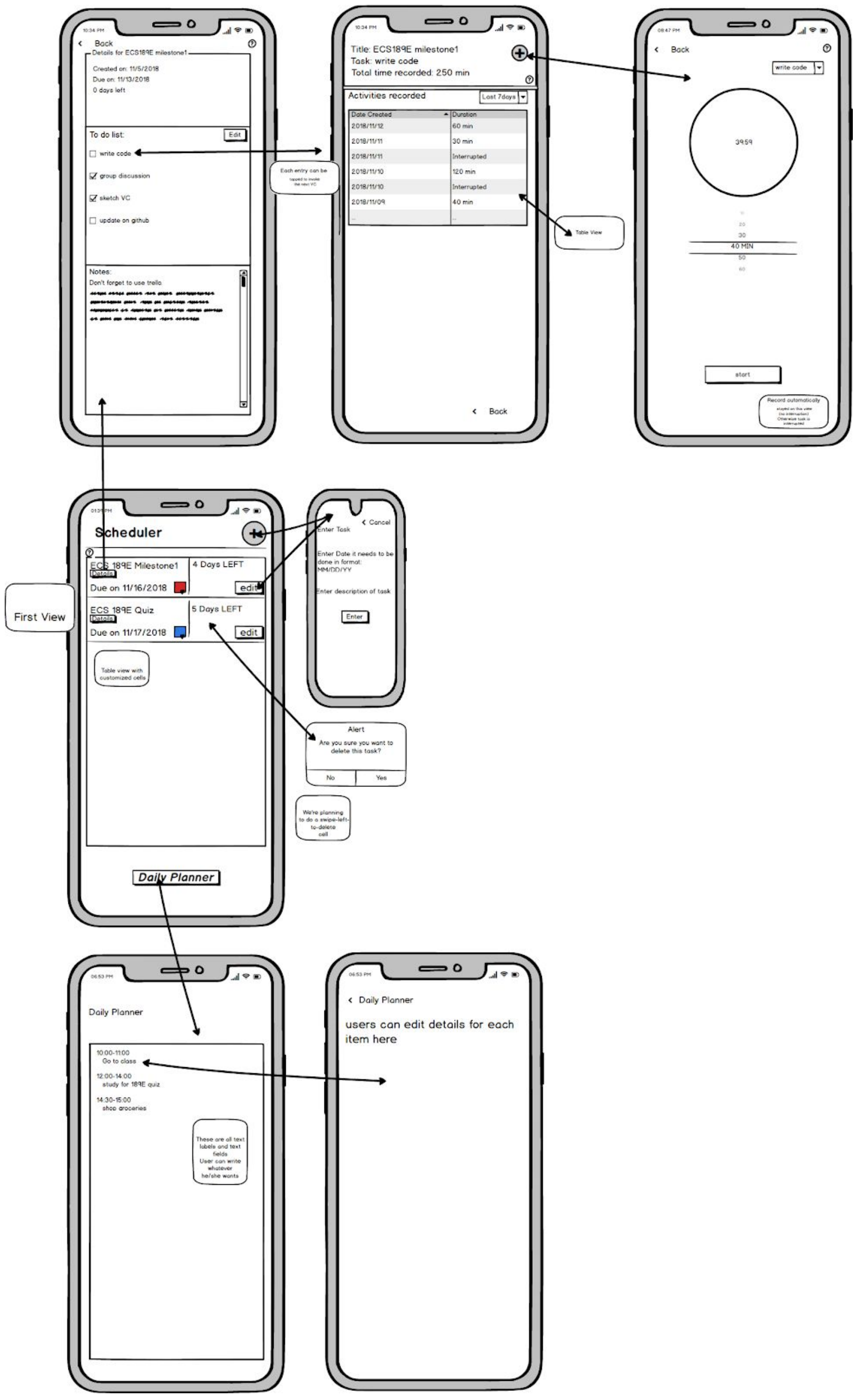
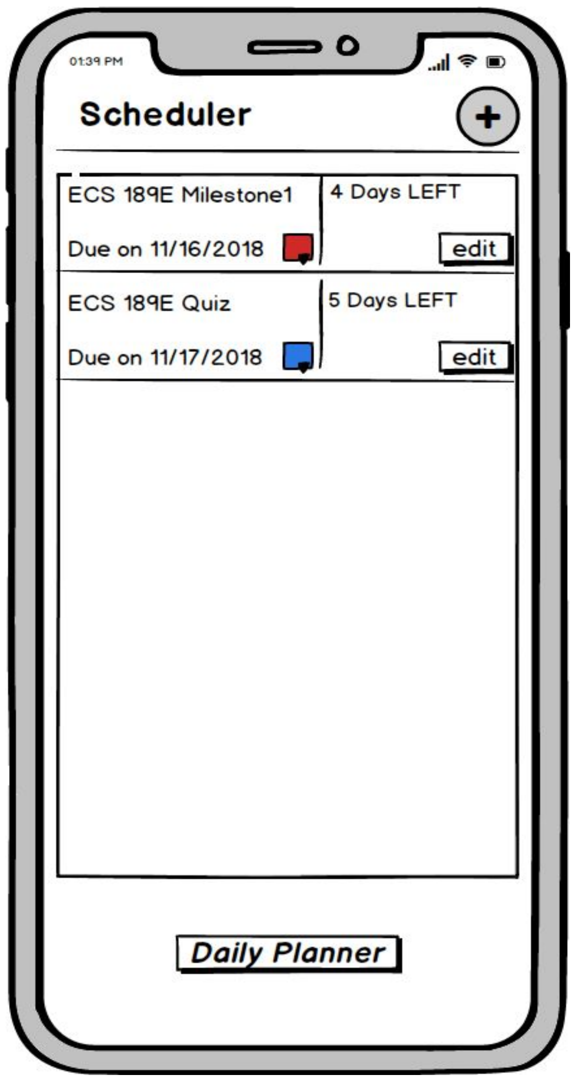
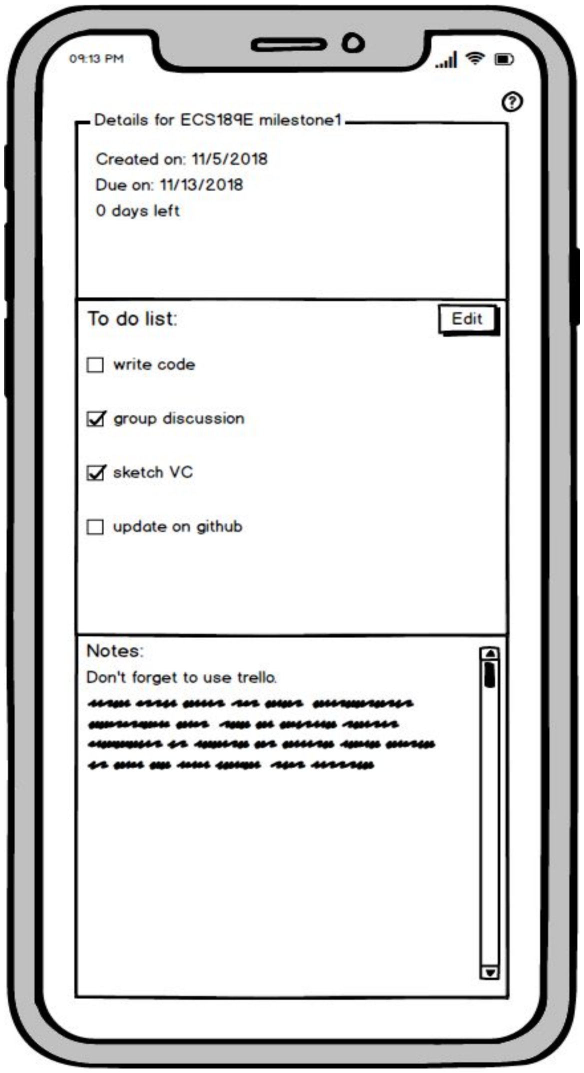
- Designs for all of the screens you plan to have.




01:39 PM
Scheduler
+
ECS 189E Milestone1
Due on 11/16/2018
4 Days LEFT
edit
ECS 189E Quiz
Due on 11/17/2018
5 Days LEFT
edit
Daily Planner


09:13 PM

Details for ECS189E milestone1

Created on: 11/5/2018

Due on: 11/13/2018

0 days left

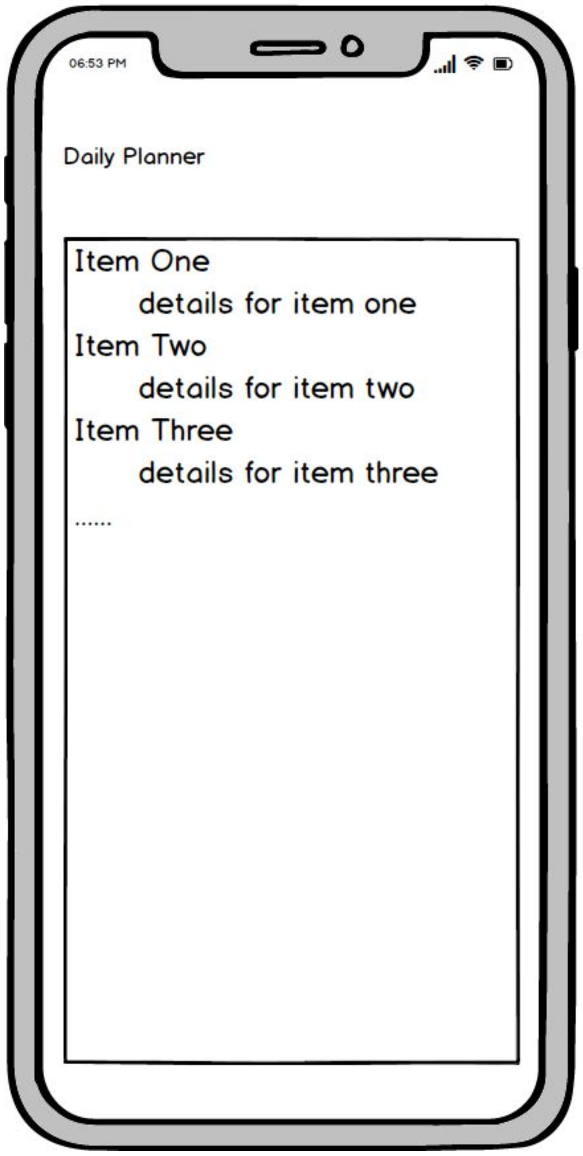
To do list:

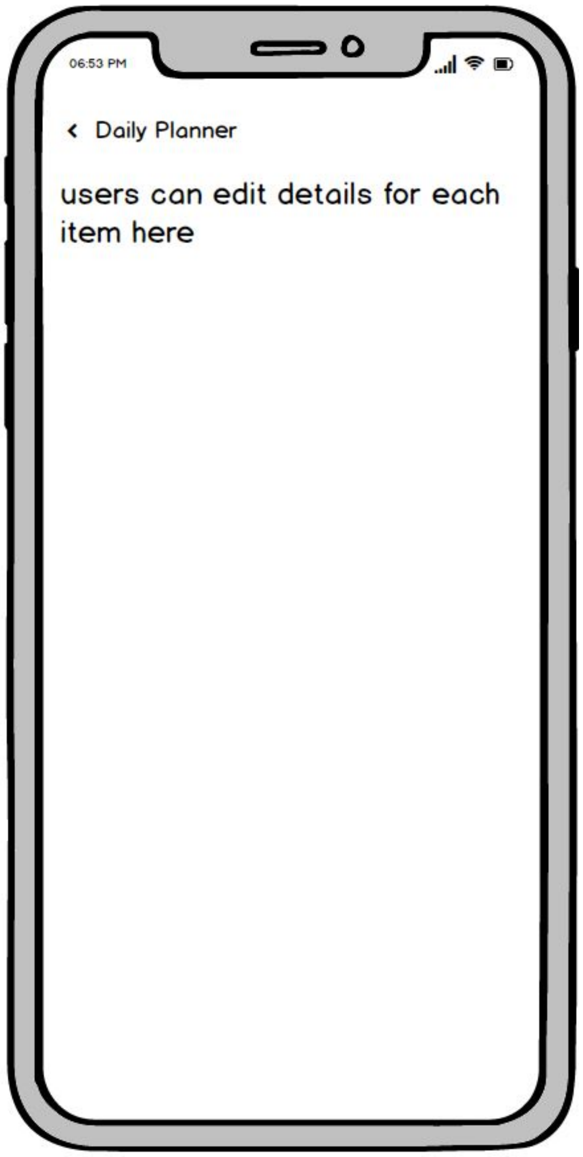
Edit

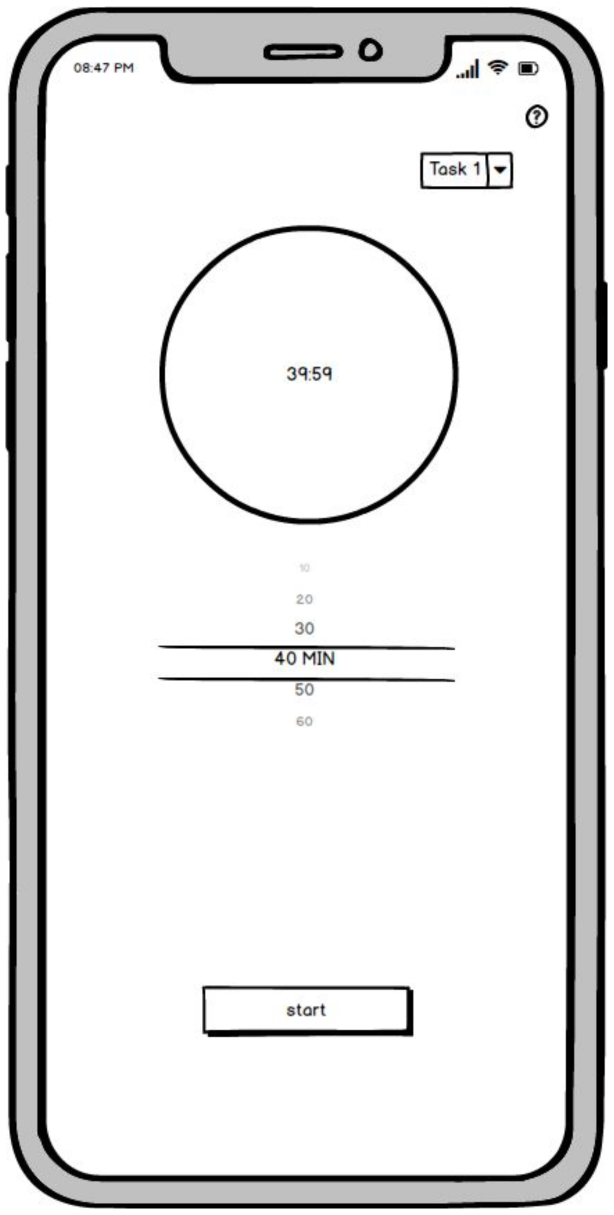
- ☐ write code
- ☑ group discussion
- ☑ sketch VC
- ☐ update on github

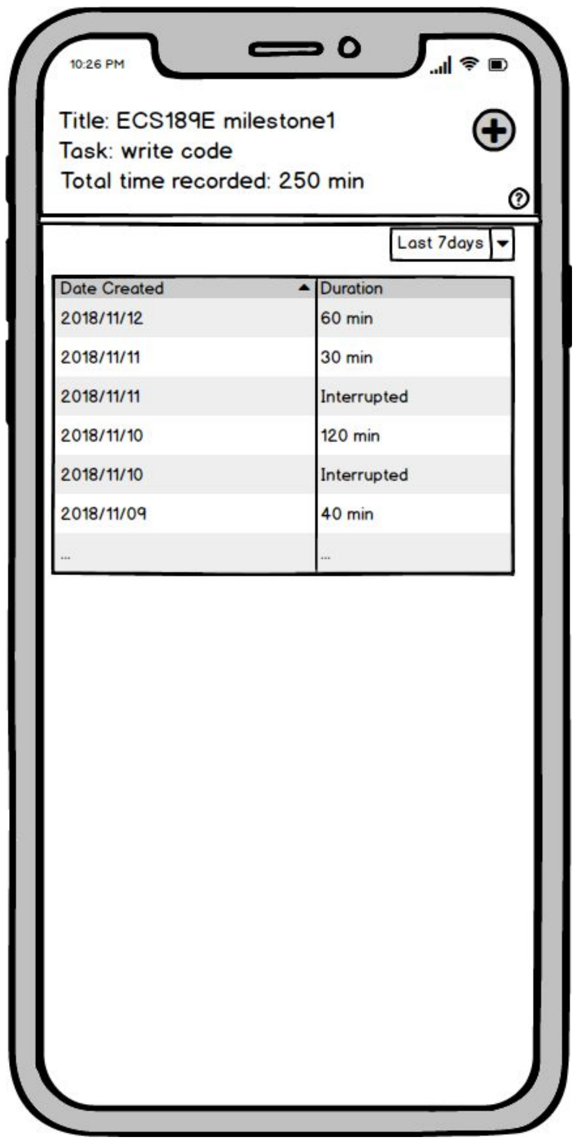
Notes:

Don't forget to use trello.


06:53 PM
Daily Planner
Item One
details for item one
Item Two
details for item two
Item Three
details for item three
......

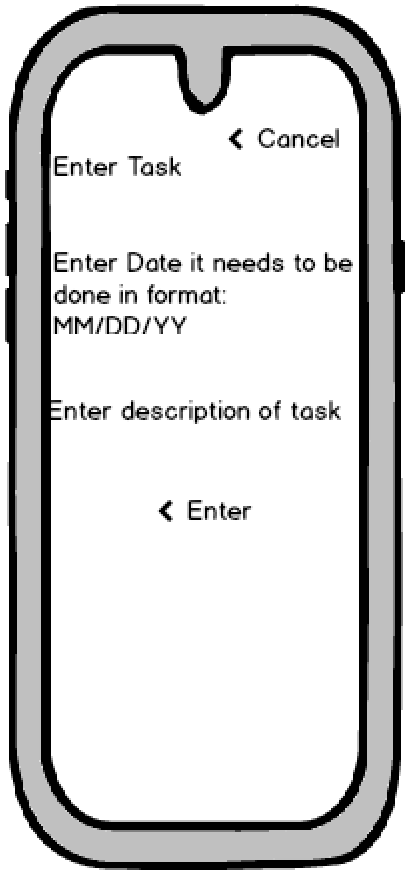

06:53 PM
Daily Planner
users can edit details for each item here


08:47 PM
Task 1
39:59
10
20
30
40 MIN
50
60
start



Title: ECS189E milestone1
Task: write code
Total time recorded: 250 min

Date Created ▲	Duration
2018/11/12	60 min
2018/11/11	30 min
2018/11/11	Interrupted
2018/11/10	120 min
2018/11/10	Interrupted
2018/11/09	40 min
...	...



Cancel
Enter Task
Enter Date it needs to be done in format: MM/DD/YY
Enter description of task
Enter

- A listing of all third party libraries you plan to use.
  For searching, auto completing, filtering, and sorting:
  https://www.smashingmagazine.com/2012/04/ui-patterns-for-mobile-apps-search-sort-filter/

  Background Fetch and Send banner notification:
  https://www.objc.io/issues/5-ios7/multitasking/

  PermissionScope for requesting permissions from users.

  Hedwig for sending users emails for reminders.

  MGSwipeTableCell for adding buttons to table cells.

  Timepiece for intuitive date handling in Swift.

  FSCalendar for a fully customizable iOS calendar library.
  https://medium.com/app-coder-io/33-ios-open-source-libraries-that-will-dominate-2017-4762cf3ce449

- Server support for your app. If you plan to roll your own server support, a full listing of the APIs you plan to build.
  No server support needed. Only Local Storage.
- A listing of all of the models you plan to include in your app

  1. Storage.swift: similar to our HW.

```
Class Task {
        var title: String
        var due_time: Date
        var details: String



        init()
        …
}

Class Scheduler {
        var tasks: [Task]
        sort(task1 : Task, task2 : Task) // sort tasks by date in descending order
yr,month then day
        init()
        …
}
(pseudocode)

Class Date{
private:
        Int Year;
        Int month;
        Int Day
public:
        Date(string year, string month, string day){
                self.year = stoi(year)
                self.day = stoi(day)
                self.month = stoi(month)
        }
}

Class task{
Var taskTitle:Sring
var subtasksArray: [subtask]
//set & get methods
//Create Subtask methods, Delete subtask methods etc...


}
Class subtask{
var subtaskTitle: String
var timelogsPerTask: [timelogs]
var totaltime: float
var taskIsDone: Bool
//methods:
SortTimeLog(...){}
calcTotalTime(...){}
setTaskDone(...){}
setTaskTitle(...){}
clearLogs(...){}
//get methods…
getTaskProgress(){}
getTimeLog(Last7days/Last14days/Last30days){}
//New Idea (milestone2??)-> Visualize stats -> Pie Charts/Bar Charts
//Either keep display the stats per task per day or stats of all tasks per day
//Example: 11/08 -> 3.5hrs 11/09 3hrs using bar charts
// Or 11/08 Color1:write code 1.5hr Color2: Group discussion 2hr 11/09 … …
…

}
```

//Conditions to save the timelog
//If Timer counts down to 0:00 without user dismissing the current view controller -> Success -> add the timelog to the task
//If user dismissed the current view controller (interrupts the timer) save the entry as interrupted and add the timelog to the task

Class timelogs{
var dateCreated: String
var duration: float
var parentTaskTitle: String
var isInterrupted: Bool
//methods set
…
//methods get
…

}

- Implementing checkboxes: can set an uncheckedImage for uncheck button for UIControlStateNormal and a checkedImage for checked -> UIControlStateSelected.
- Implementing swipe to delete:
- `func tableView(_ tableView: UITableView, canEditRowAt indexPath: IndexPath) -> Bool { return true } func tableView(_ tableView: UITableView, commit editingStyle: UITableViewCell.EditingStyle, forRowAt indexPath: IndexPath) { if (editingStyle == .delete) { // handle delete (by removing the data from your array and updating the tableview) } }`

- Implementing Timer: similar to class example, countdown, key is how to track if current view is dismissed

- A listing of all ViewControllers you will implement. Please include how you plan to navigate to / from each ViewController and any protocols / delegates / variables you plan to use for ViewController / ViewController communication.
  - Creating new event:
    - From main menu view controller upon "+" button being pressed
    - Segue to new event VC which allows user to enter in the name of new task, description of task and when the task is due
    - Force user to enter month,day, year in correct format i.e. prevent them from entering in too many characters
    - Event VC passes back this event which is appended to the list of events in the main VC
    - The tasks in the main VC is then resorted based on a custom sort function

        - Implement enter and cancel

Five ViewControllers: Scheduler ViewController, Details ViewController, TimerViewController, Daily-Planner ViewController, Add task view Controller
Details of flow please refer to the PNG file

By clicking daily planner button in the first view, users can go to the 4th view, daily planner, where tasks scheduled for that day will be shown only on the fourth view. The daily planner works as notes. User can click the cell on the 4th view and go to the detailed view to read or edit the details. I plan to filter the data in the first row by start time, and pass the variables with start time == today's date to the fourth view.

By clicking add (the plus button) in the first view user can add tasks to the scheduler app will go to the add task VC from which it will allow user to enter data and send it back to parent VC and resort based on relative closeness of due dates

- A list of approximately week long tasks and a timeline of when each of them will be done

Sunil	Nov 13 - Nov 20: finish add task view contoller and make sure segue is workign correctly (passing back data), also custom date sort fxn for tasks Nov 21 - Nov 28: work with teammates to unify all views/view controllers/models; work on assigned challenges. Nov 29 - Dec 5: continue with assigned challenges and testing.
Jiajun	Nov 13 - Nov 20: finish the main scheduler view as soon as possible so that my teammate could have the scheme to work out Nov 21 - Nov 28: work with teammates to unify all views/view

	controllers/models; work on assigned challenges. Nov 29 - Dec 5: continue with assigned challenges and testing.
Yanxi	Nov 13 - Nov 20: finish the timer view and the activity logs view. Nov 21 - Nov 28: work with teammates to unify all views/view controllers/models; work on assigned challenges. Nov 29 - Dec 5: continue with assigned challenges and testing.
Hanxing	Nov 13 - Nov 20: finish the daily planner view and its detailed view. Nov 21 - Nov 28: work with teammates to unify all views/view controllers/models; work on assigned challenges. Nov 29 - Dec 5: continue with assigned challenges and testing.

- A trello board that includes all of the tasks and their state (up next, in progress, blocked, done). These tasks need to include an owner from the start, but the owner can change during the quarter as needed
https://trello.com/b/SuBG7jG6/scheduler-app

- A github classroom project that includes all of this documentation. The Readme in your project needs to include a link to your Trello board and pictures of each of the team members with their name and github handle
- A testing plan. A good app should get feedback from potential users early and often. List here what you plan to get other people to test out.

We plan to start to test the app once it is unified. We will ask friends in the class and outside the classes for testing. Some interesting questions we would like to ask them include: 1. Are the user interfaces appealing? Rate from not appealing, it is ok, it is appealing, and it’s extremely appealing? 2. Is the app help with organizing one’s daily tasks in any way? Rate from not helping at all, helps a little, helps a lot. 3. Any suggestions on the app?