PM3 Deliverable

Team Hygiene
Eugene Min, Adalia Truong, Claire Holmes, Anthony Donberger, and Harris Naseh

High-level Design (4%)

Describe which architectural pattern you would use to structure the system. Justify your answer.

The architectural pattern our team would use to structure our system would be the Event-Based architecture. The Event-Based architecture uses the idea of consuming, producing, and detecting based on a reaction to events. Within our project, our users will solely rely on a GUI to navigate the app and utilize its functionalities. The functionality of the app will then correspond to the user pressing a specific button and triggering an event within the system. This event will notify the system to do something based on which button was pressed. For example, if the user is using our application on their phone and they wanted to add an event to their calendar then they would press the “add event to calendar” button. The event being triggered is the user pressing the “add event” button. After this event is triggered the software will understand to “consume” the event by adding the user’s event to the calendar.

Low-level Design (4%)

Discuss which design pattern family might be helpful for implementing this project. Justify your answer, providing a code or pseudocode representation and an informal class diagram.

The Creational design pattern would be helpful for implementing our team’s project. Our project can be split up into various classes that can interact with one large data structure. For example, an events class can be made that is meant for specifically being placed on a calendar. These event class objects can store a date, description, and title as a String. These classes can be created each time a new calendar event is made and stored in some data structure meant to keep track of all calendar events. This same concept can be applied to our app’s note functionality. A notes class can be made that stores a description, title, and date of creation as a string.

Pseudocode using builder pattern:

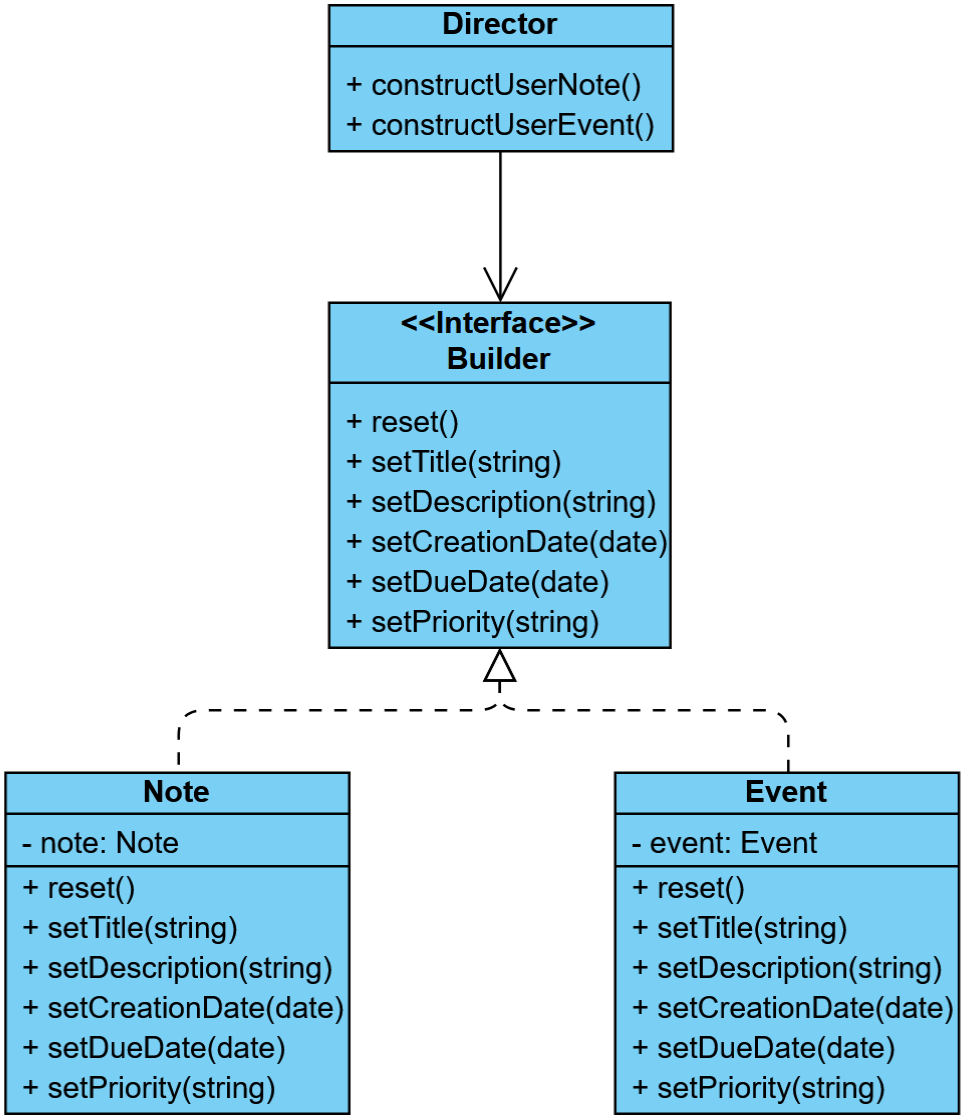
```
class Note is
     // A note on the user's to do list

class Event is
```

```
     // An event that the user adds to their calendar


interface Builder is
     method reset()
     method setTitle(...)
     method setDescription(...)
     method setCreationDate(...)
     method setDueDate(...)
     method setPriority(...)


class NoteBuilder implements Builder is
     private field note:Note

     constructor NoteBuilder() is
     this.reset()

     // The reset method clears the object being built.
     method reset() is
     this.note = new Note()

     method setTitle(...) is
     // Set the title of the note.

     method setDescription(...) is
     // Set the description of the note.

     method setCreationDate(...) is
     // Set the creation date of the note.

     method setDueDate(...) is
     // Set the due date of the note.

     method setPriority(...) is
     // Set the priority of the note.

     method getProduct():Note is
     // Return the product and reset the builder


class EventBuilder implements Builder is
     private field event:Event

     constructor EventBuilder() is
     this.reset()

     // The reset method clears the object being built.
     method reset() is
     this.event = new Event()

     method setTitle(...) is
```

```
     // Set the title of the event.

     method setDescription(...) is
     // Set the description of the event.

     method setCreationDate(...) is
     // Set the creation date of the event.

     method setDueDate(...) is
     // Set the due date of the event.

     method setPriority(...) is
     // Set the priority of the event.

     method addToCalendar(...) is
     // Add the event to the user's calendar

     method getProduct():Event is
     // Return the product and reset the builder, also add to
calendar


class Director is

     method constructUserNote(builder: Builder) is
    // Get user input and create a note
     builder.reset()
     builder.setTitle(...)
     builder.setDescription(...)
     builder.setCreationDate(...)
     builder.setDueDate(...)
    builder.setPriority(...)

     method constructUserEvent(builder: Builder) is
     // Get user input and create event
    builder.reset()
     builder.setTitle(...)
     builder.setDescription(...)
     builder.setCreationDate(...)
     builder.setDueDate(...)
    builder.setPriority(...)


class Application is

     director = new Director()

     method makeNote() is
     NoteBuilder builder = new NoteBuilder()
     director.constructUserNote(builder)
     Note note = builder.getProduct()
```

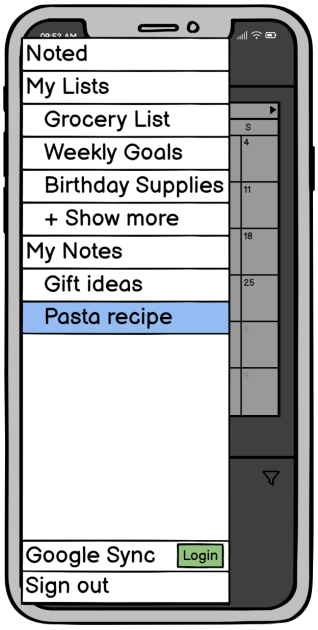
```
 method makeEvent() is
 EventBuilder builder = new EventBuilder()
director.constructUserEvent(builder)
Event event = builder.getProduct()
```

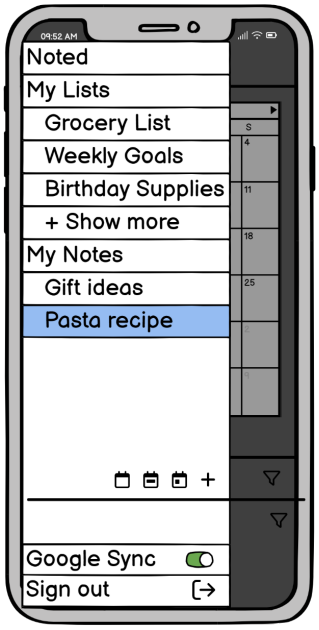
Class Diagram:



Design Sketch (4%)

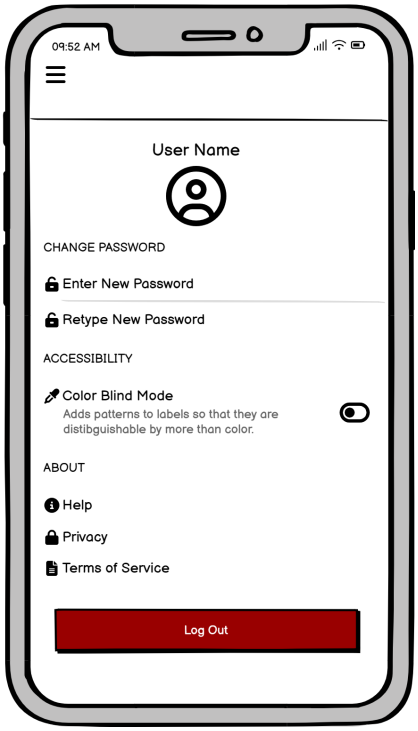
Wireframe mockup (more detailed mockup in next section):


Noted
My Lists
Grocery List
Weekly Goals
Birthday Supplies
+ Show more
My Notes
Gift ideas
Pasta recipe
Google Sync
Login
Sign out

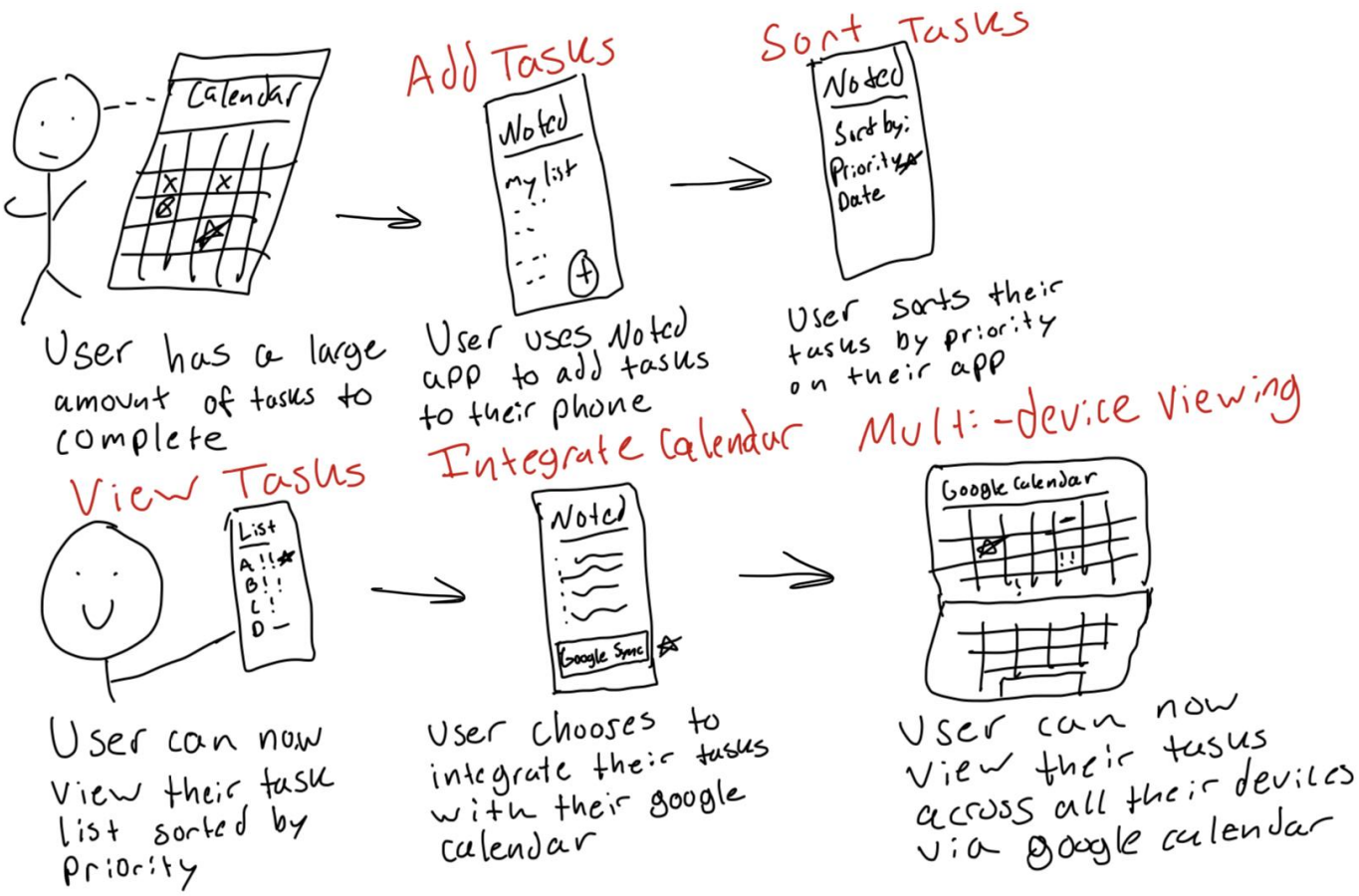

Noted
My Lists
Grocery List
Weekly Goals
Birthday Supplies
+ Show more
My Notes
Gift ideas
Pasta recipe
Google Sync
Sign out


09:52 AM
NOVEMBER 2023
Sign In
Connect your Google Account to sync to your Google Calendar
Continue with Gmail
Finish HW
Do laundry
Buy groceries
Make dinner
+ Add a task


09:52 AM
User Name
Acme user since April 2017
BASIC PROFILE
User Name
Google Information
User@gmail.com
Change Password
Log Out


09:52 AM
User Name
CHANGE PASSWORD
Enter New Password
Retype New Password
ACCESSIBILITY
Color Blind Mode
Adds patterns to labels so that they are distibguishable by more than color.
ABOUT
Help
Privacy
Terms of Service
Log Out

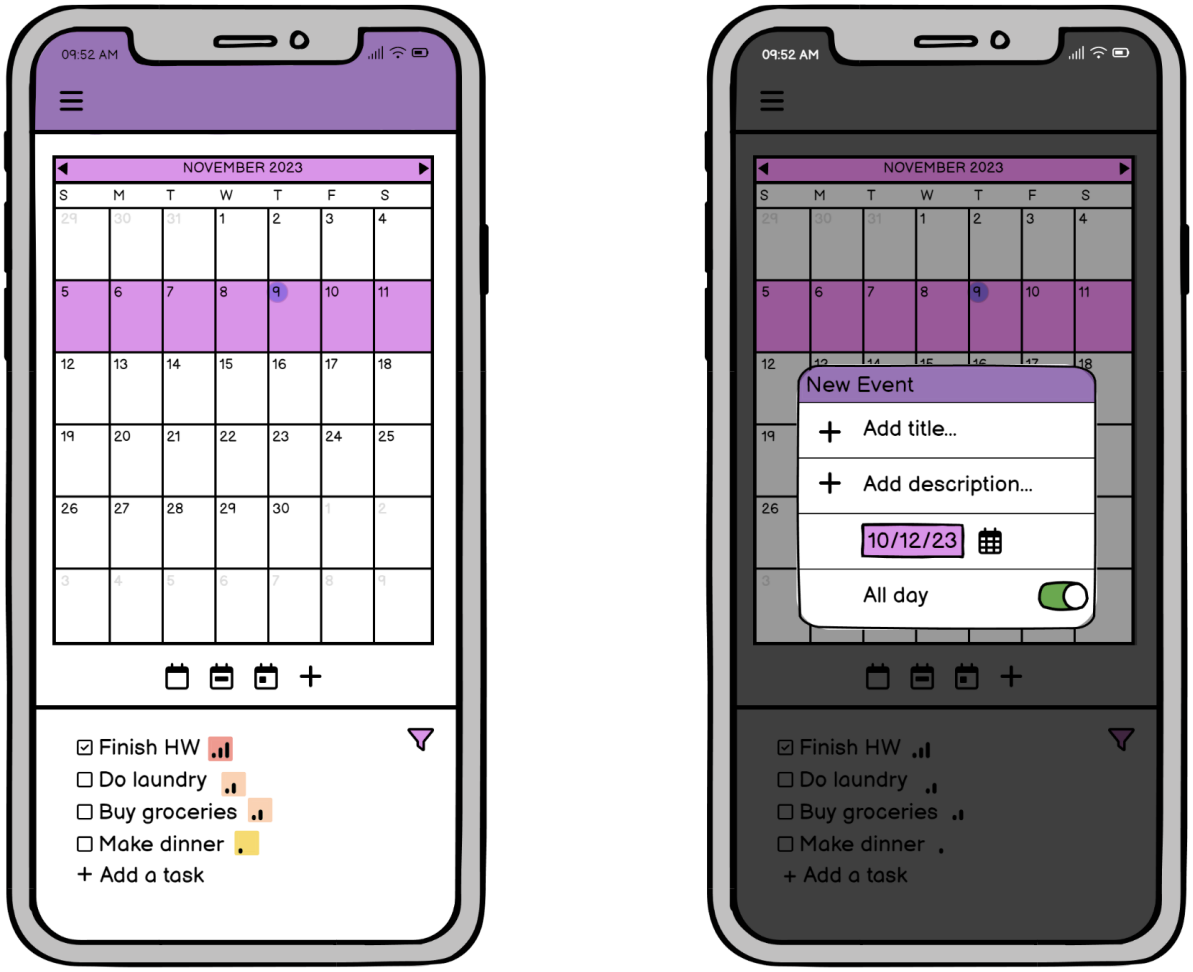
Storyboard:



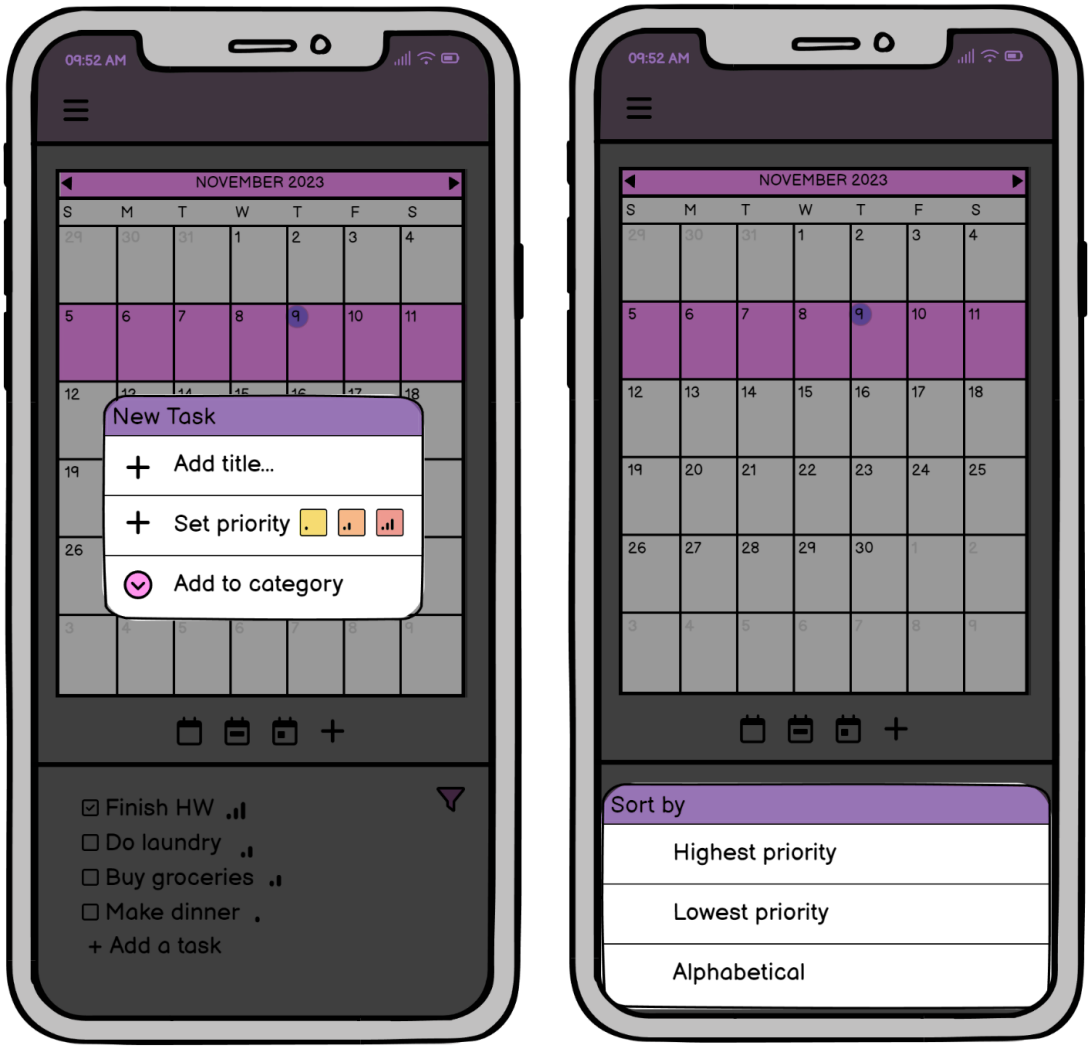
Rationale:
In this storyboard we want to highlight the importance of the basic functionality our app Noted provides. We wanted the user to have the option to sort their tasks by their priority. This design is shown in the storyboard when the user adds their tasks and chooses to sort by priority. The user can then view their tasks sorted on the app. Another design choice our storyboard highlights is google calendar integration. This allows the user to view their tasks in a calendar format on any device supported by google calendar.

Project Check-In - Completed

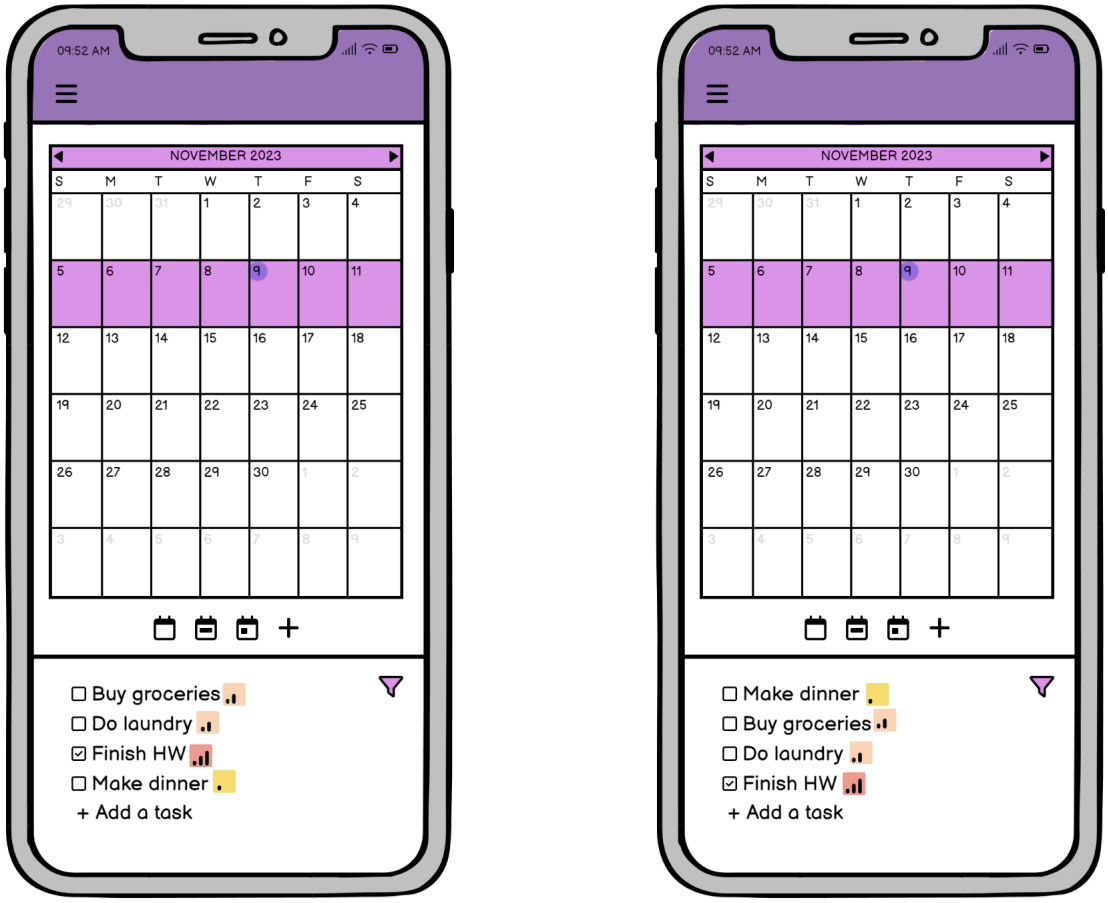
Process Deliverable (3%)
Prototyping: submit a prototype of your system



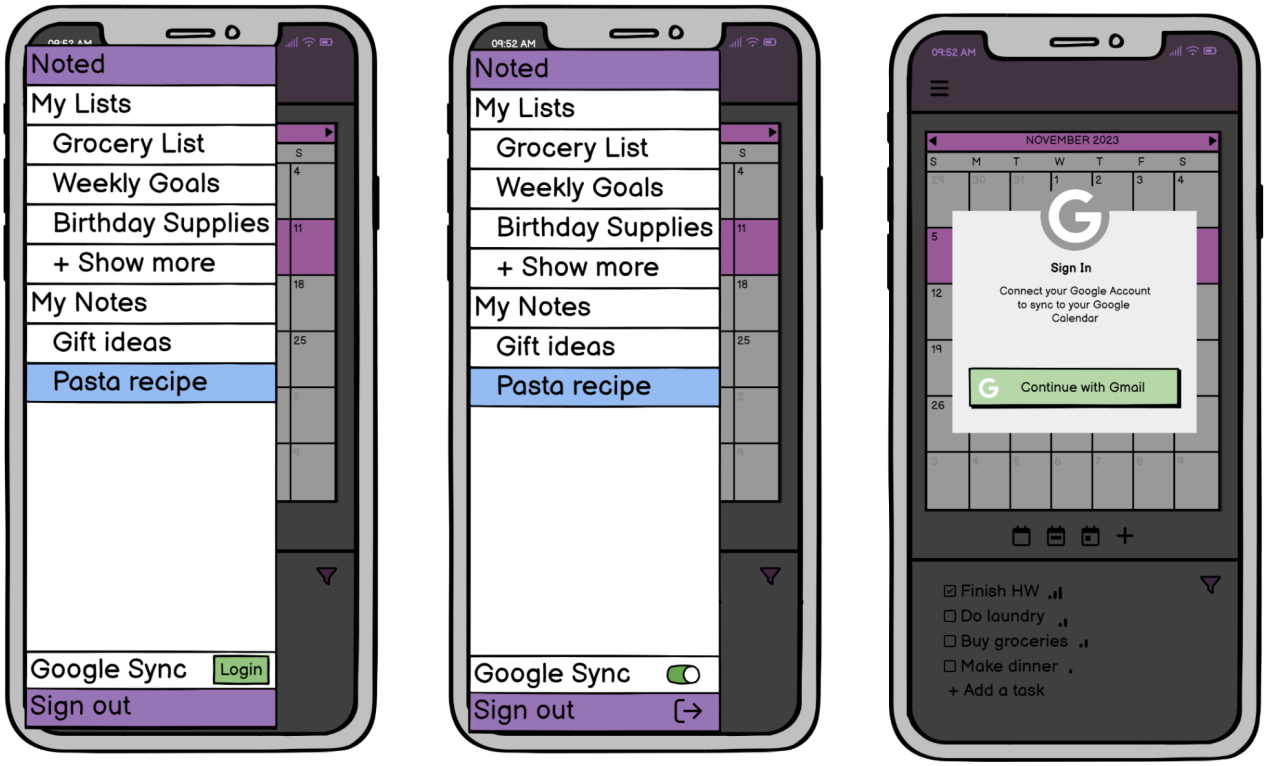
Calendar view and adding a new event



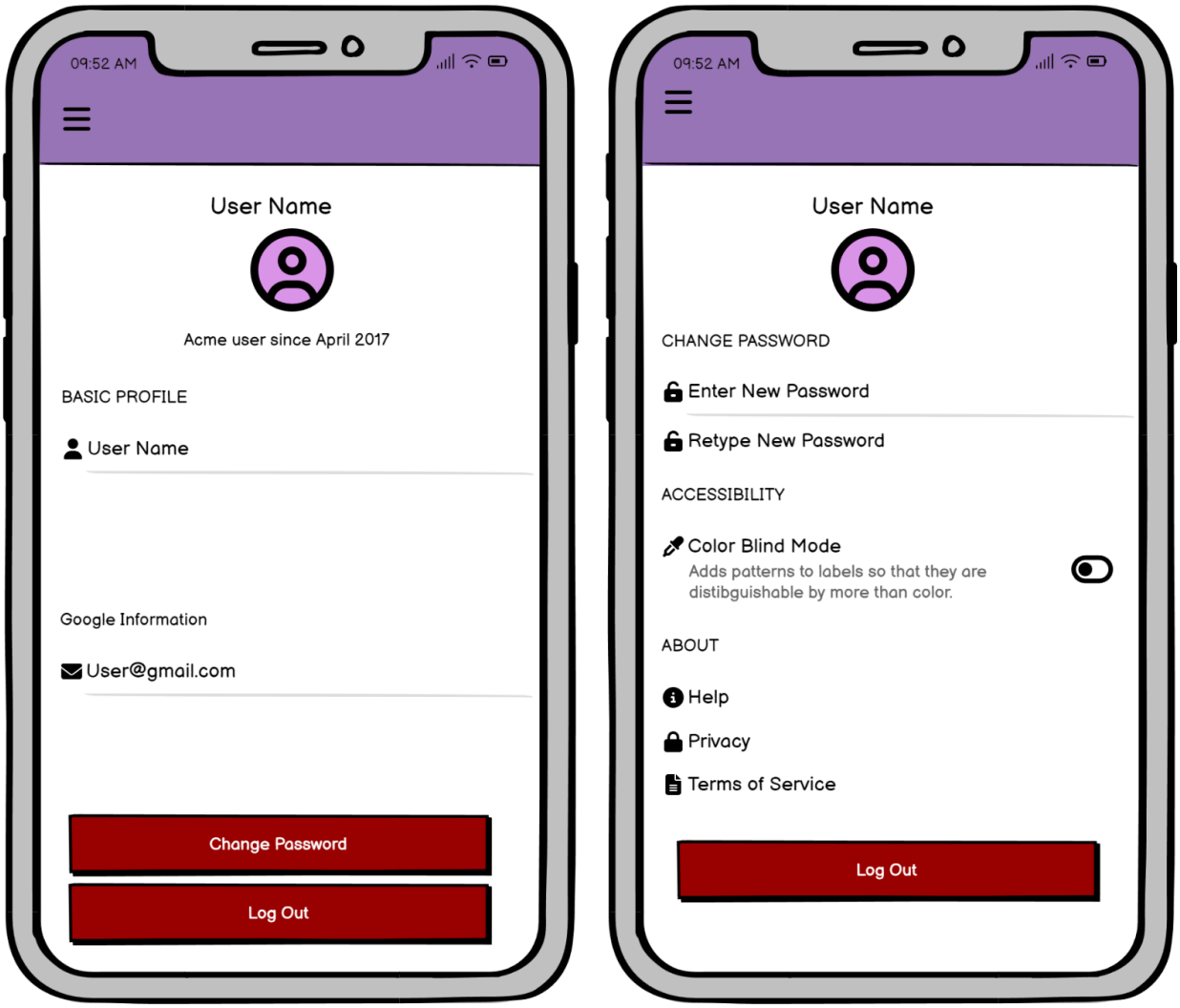
Example of adding a new task and sorting by priority



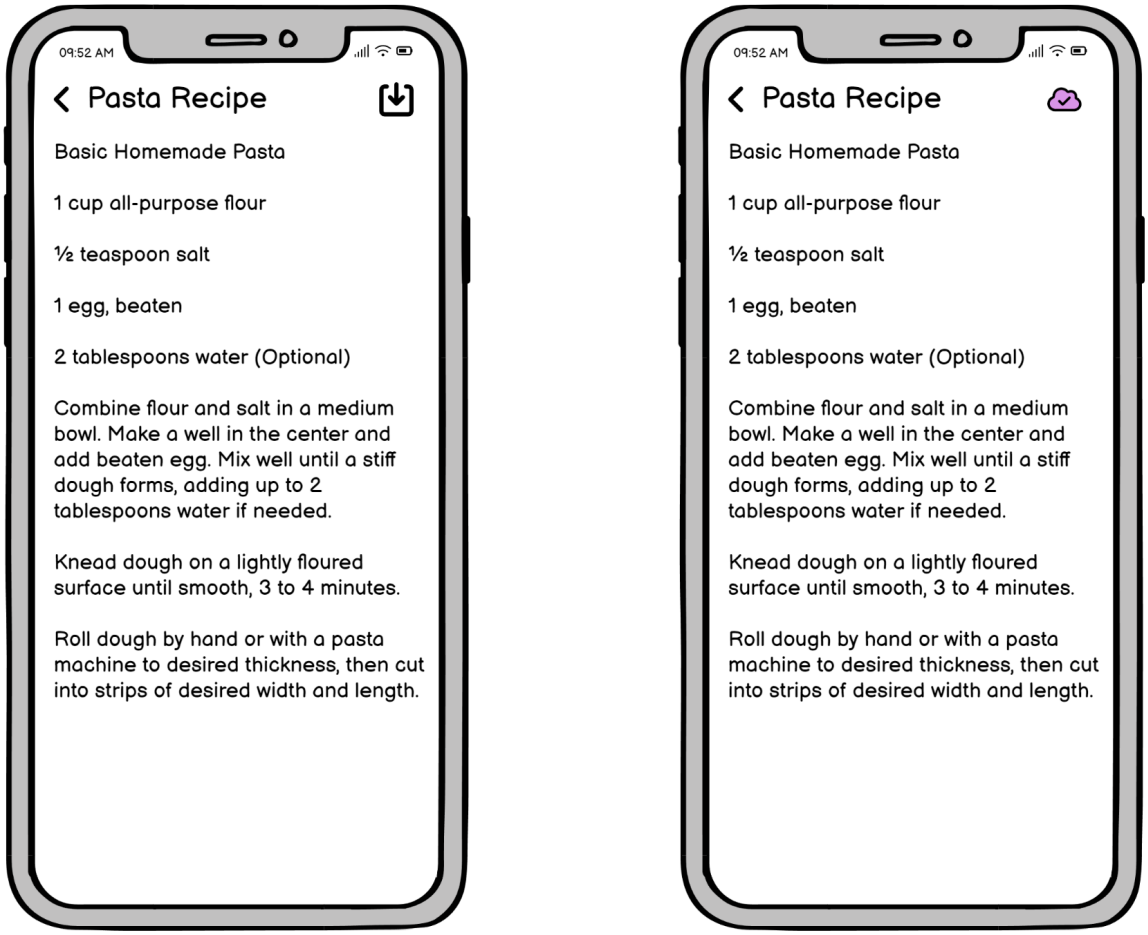
Calendar view with tasks sorted different ways



Google Sync workflow



User profile screens



Visual differences between a synced and unsynced notes