OnTop

Optimize Your Free Time

Team: Matt Millett, Alec Douglas, Pascal Odek, Pallabi Ghosh

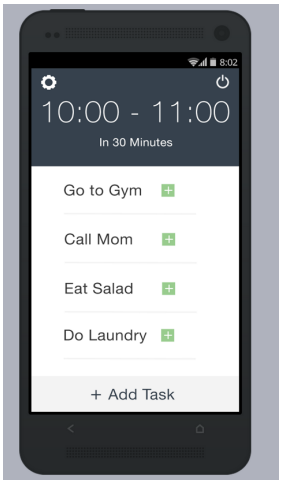

Fig. 1 : Our Design

Problem/Solution: Busy people often have small tasks they would like to accomplish, such as organizing their desk or working out, but that they feel they never have time for. While apps such as Reminders currently allow such people to organize these tasks into lists, they offer no help when it comes to scheduling, leaving users to put off these tasks in an endless cycle of procrastination. We at OnTop have found a solution to this common problem. Using data from existing calendars, OnTop finds all the free time user's have in their day and helps them schedule tasks which fit each of those time periods. OnTop will then remind users when they have a scheduled task approaching, allowing them to efficiently use their free time.

Tasks & Final Interface Scenarios: The three main tasks our app supports are as follows:

1. Adding a new task. By tapping the button located at the bottom of the screen, a user can create a custom task and specify how long this task will take. This new task will then be added to the list shown on the homescreen. This feature is essential as it allows the user to customize their OnTop experience.
2. Putting the app in "sleep mode." By tapping the power button located in the top right corner of the app, the user can temporarily suspend any notifications. We found this feature very important as schedules change constantly, so the app must be flexible to suit this.
3. Inviting a friend to complete a task with you. By tapping the invite button located at the bottom of the screen, the user can send a message inviting someone else to complete a task along with them. This feature enhances the motivational capacity of this app: if a user has a task they have to do but do not want to, a friend can help them go through with it.

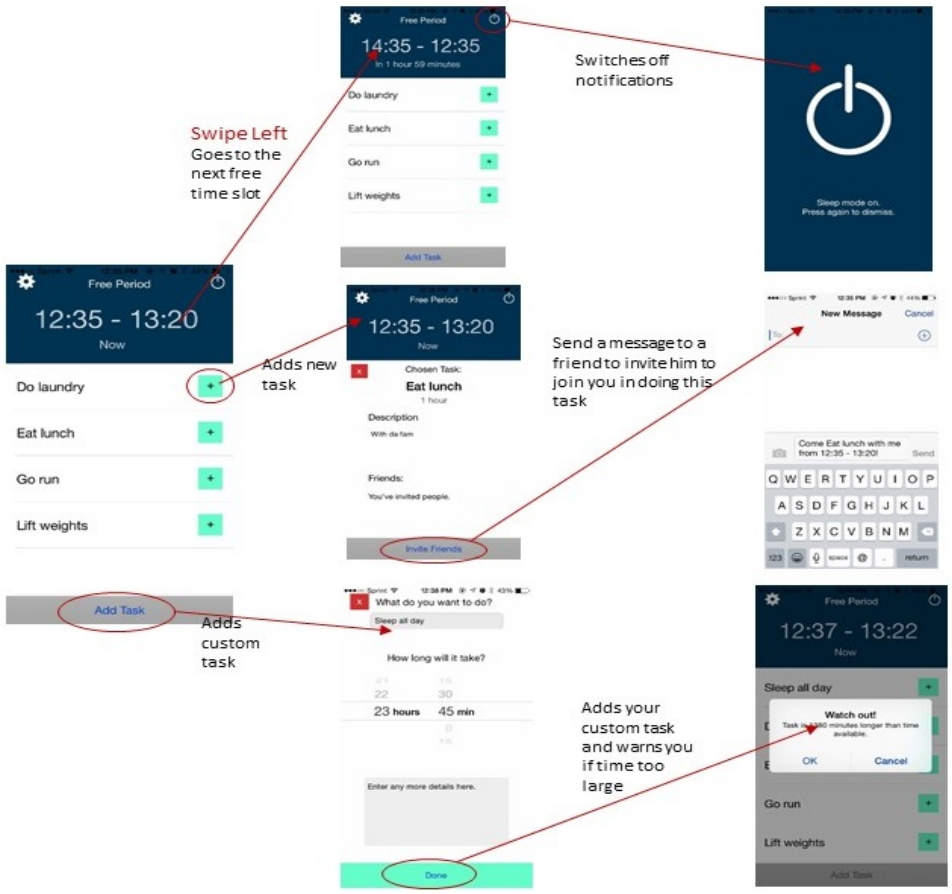

Fig. 2: High Fidelity Prototype Storyboard

Major Usability Problems Addressed: Based on a heuristic evaluation of our medium-fi prototype, we fixed the following problems while developing our hi-fi prototype:

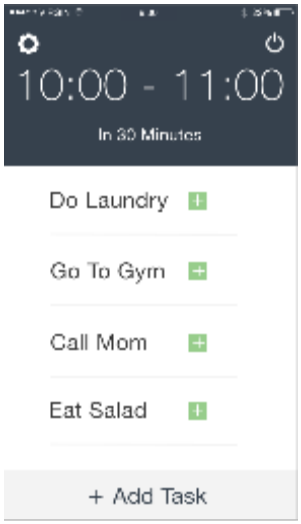

Fig. 3: Medium-Fi Task List Screen

1. H2-1: Visibility of System Status**:** Our evaluators found some bugs in the flow of the prototype in which trying to cancel an operation (such as adding a new task or inviting a friend to complete a task) would take the user to the next step in that task. In the real iOS prototype, this is a bug we fixed.

2. H2-4: Consistency and Standards**:** Our evaluators were unsure as to the purpose of a feature which allowed users to swipe left on a task in the task list in order to remove it from the list. Because this was simply a misunderstanding of the feature,

we did not make any changes. (Though in the iOS prototype, this particular swiping feature wasn't implemented).

3. H2-3 User Control and Freedom**:** Because the medium-fi prototype was all hard coded, our evaluators found it limiting that the user could not add free time slots on their own. The high fidelity prototype uses data from the user's calendar, so a feature allowing them to add extra time slots would be unnecessary.

Next we show some of our changes through the following images:

1. Users were confused about the Add More button in the invitation sent screen. So now in the Hi Fi prototype, we have a normal messaging screen that people are used to. They can use the '+' symbol to add more people.

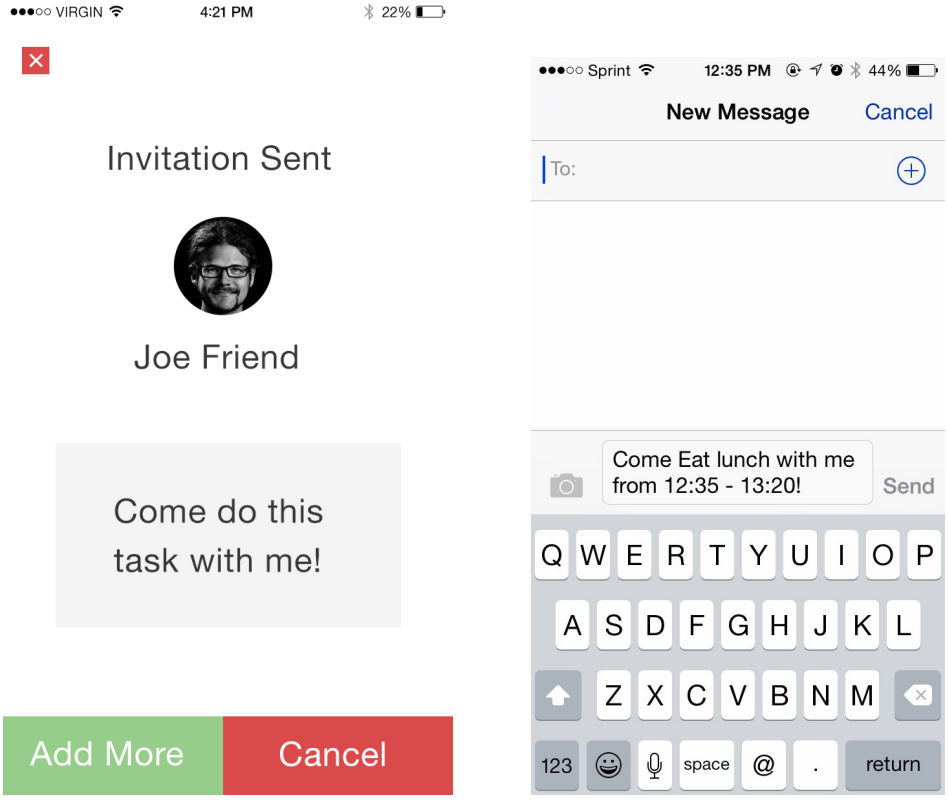

Fig 4: Changes in invitation to Friends screen

2. Next people had problem understanding how to return back to the normal mode from the sleep mode. The sleep mode stops all notifications. To return to the normal mode we just have to click on the sleep mode screen. The instructions are given in the High Fidelity Prototype as shown below.

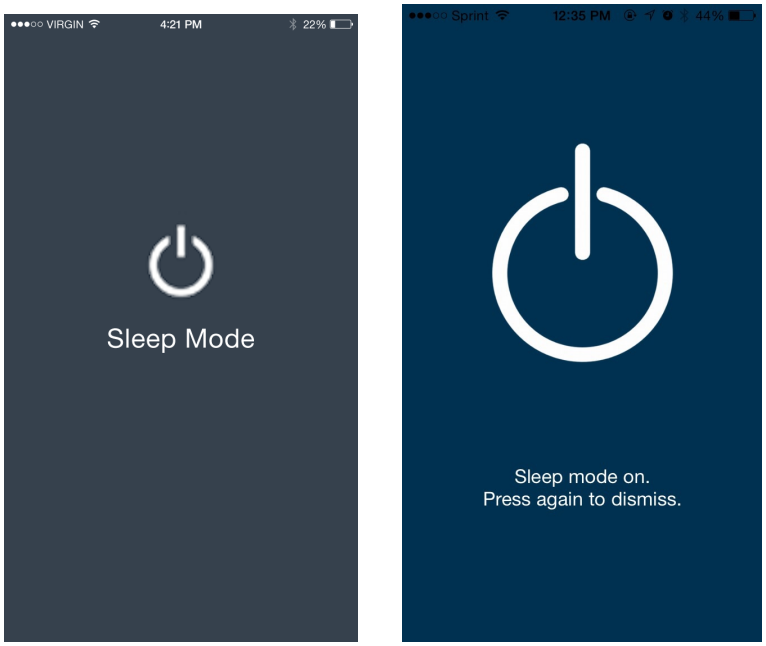

Fig 5: Changes to The sleep mode Screen

3. We were also able to implement some features in the iOS Prototype on the custom task entry screen. These features could not be implemented in the inVision Prototype and hence was hardcoded. We were able to implement the user entry of task name and time on the iOS prototype.

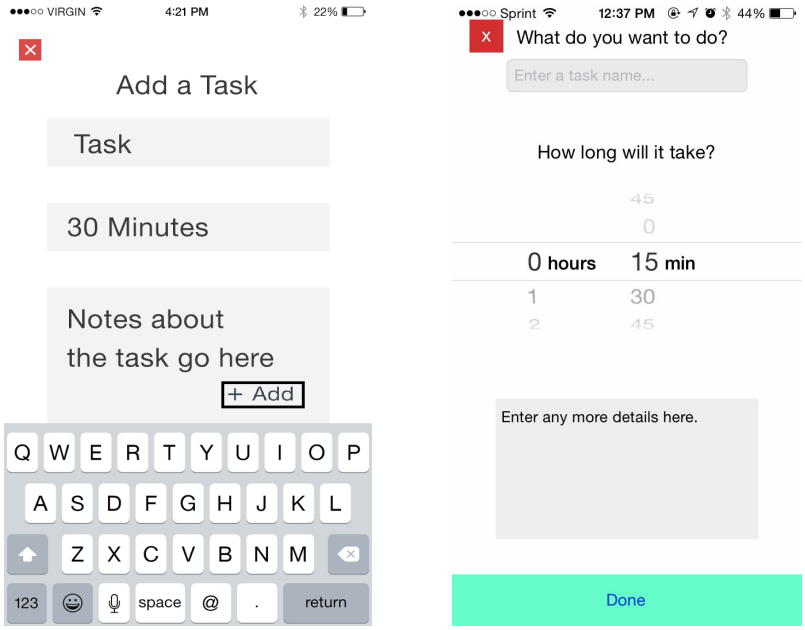

Fig 6:Featured that were originally hardcoded in the Medium Fidelity Prototype was implemented on the iOS prototype

4. Also we were able to give error notifications if anything seemed to be wrong.

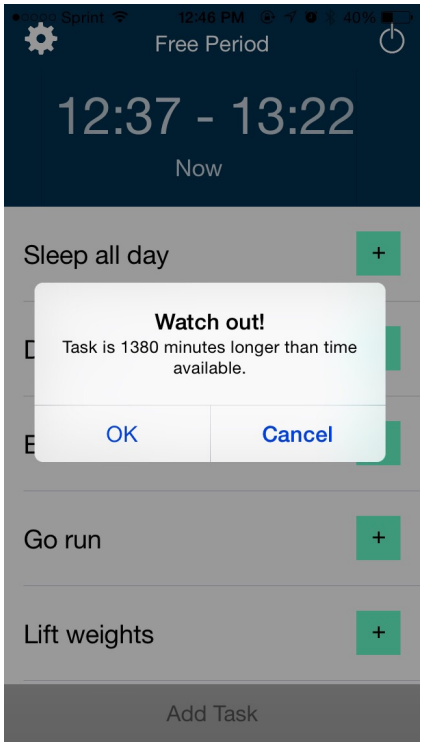

Fig 7:Error Notifications

Design Evolution: We used an iterative technique to design our app with review and user input at every stage to help improve the design. This helped make the system easy to use with all the features that are necessary, without any unnecessary clutter. The various steps in this process are as follows:

1. Project Proposal: Our initial project idea was a rough overview of what we wanted in our system. We wanted to develop a system that helped people utilize the free time blocks in between their scheduled tasks to help them achieve their goals.
2. Contextual Inquiry and Task Analysis: We interviewed 3 people, a research assistant, a student and a professor and asked for their opinion on our idea. We asked them all the 11 task analysis questions. Based on their input we decided on 3 main tasks for our system:

    1. Make/Track/Remind Tasks
    2. Allow flexibility in user's plan
    3. Find Emergency Free Time

    We also considered three alternative platforms:

    1. Cloud-integrated calendar

2. Wall projected Calendar
3. Wearables

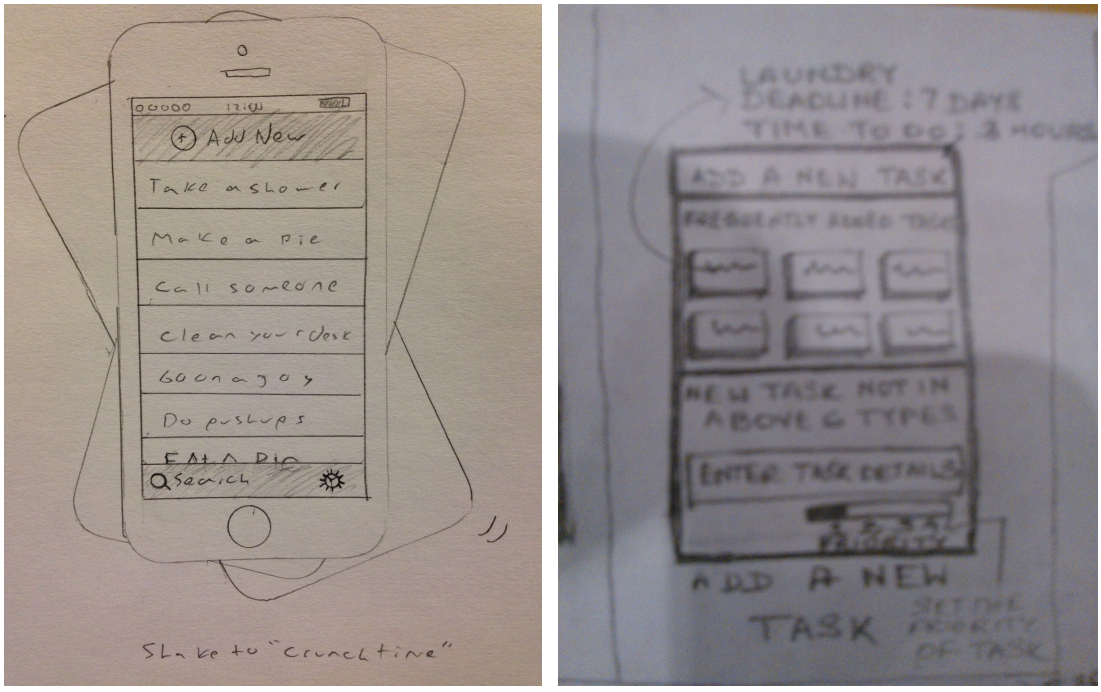

Fig. 8: Initial sketches of our design ideas

Concept Video and Sketches: In the next step we built a storyboard and made a concept video that can convey our idea to any user. We also changed our 3 tasks based on user input to:

1. Adding a new Task
2. Invite a friend to the task
3. Turn on the" Do not Disturb" Feature

We sketched one possible scenario for using all these 3 tasks and then used these sketches to develop the video that demonstrates each of these scenarios.

Low Fidelity Prototype and test: Next we sketched our app, and incorporated these sketches in the POP app to develop our low fidelity prototype. We also asked 3 people to try out this app,and let us know their recommendations and suggestions.

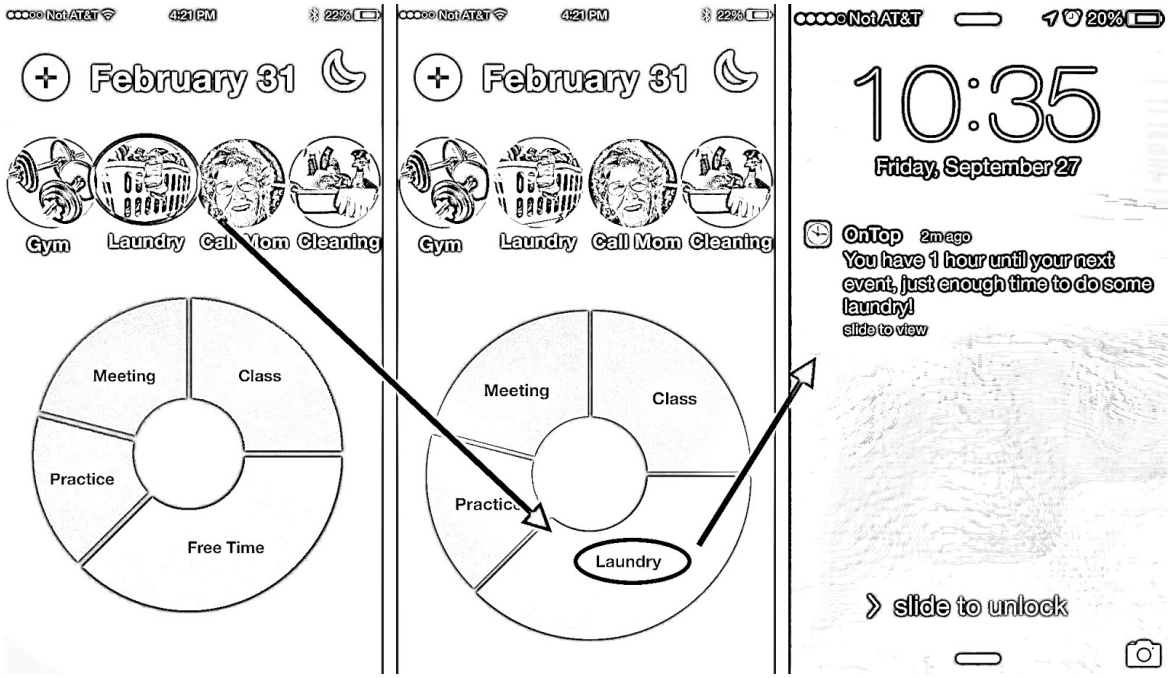

Fig 9: Low Fidelity Task 1 (Add a task) Storyboard

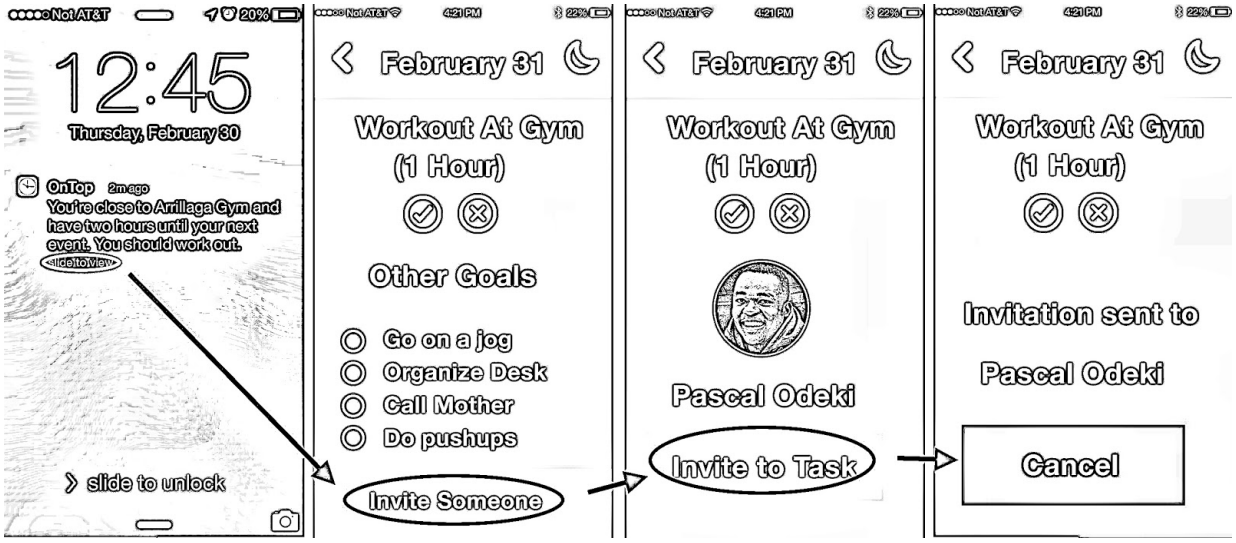

Fig 10: Low Fidelity Storyboard: Invite a Friend (Task 2)

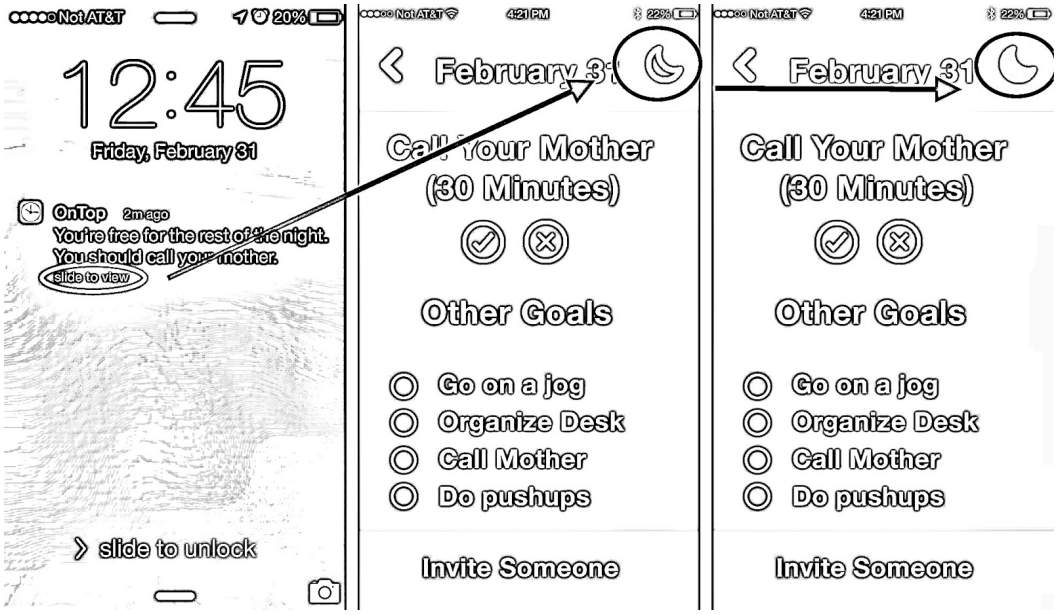

Fig. 11: Low Fidelity Task 3 (Turn on do not disturb feature) storyboard

Medium Fidelity prototype : Through our interviews on low level prototype, we came to know that our design was slightly cluttered. Also people didn't understand the pie chart format of presenting schedule and the "moon sign" of sleep mode. So we changed our design to adjust for these complications. Now we designed our medium fidelity prototype on InVision prototyping tool.

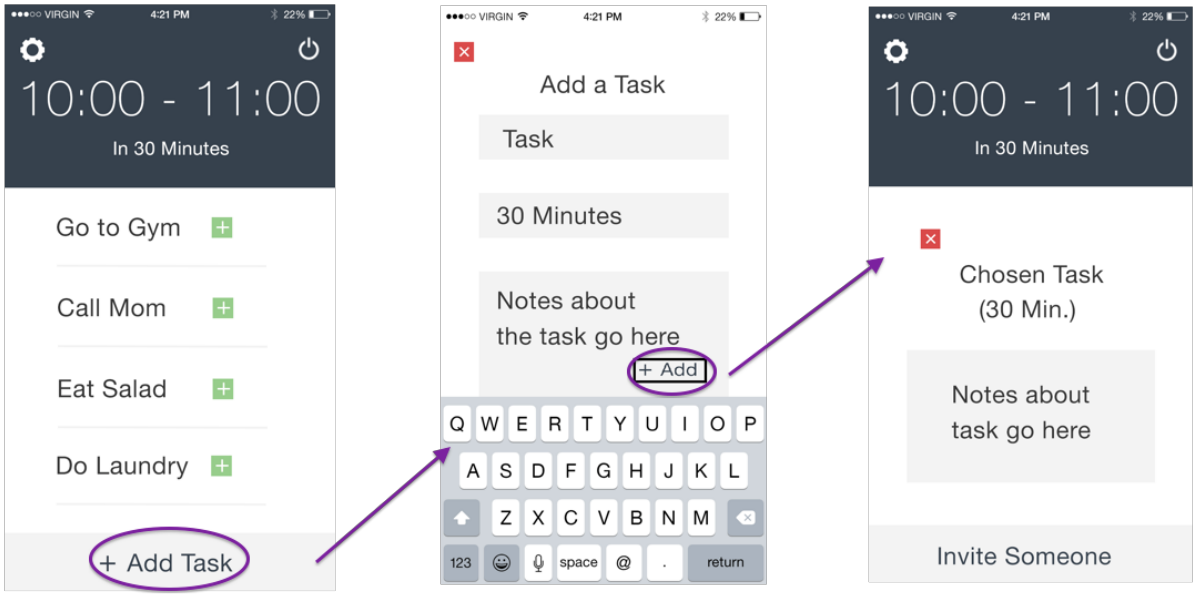

Fig 12: Medium Fidelity Task 1 (Add Task) Storyboard

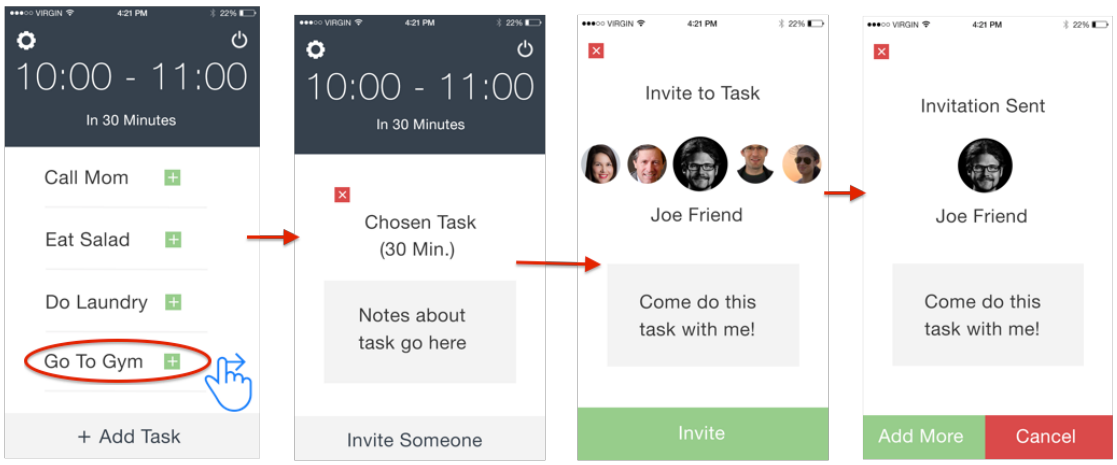

Fig 13: Medium Fidelity Task 2 (Invite Friends) Storyboard

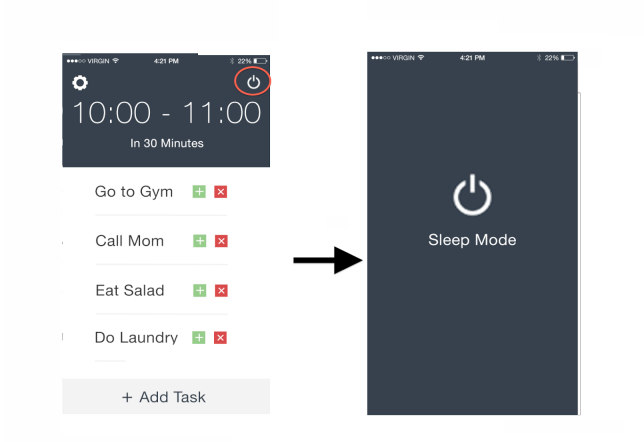

Fig 14: Medium Fidelity Task 3(Turn on sleep mode) Storyboard

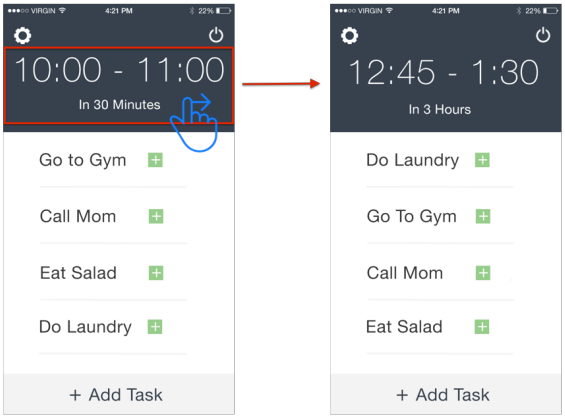

Fig 15: Swiping action to show different blocks of free time

Heuristic Evaluation: A group of 3 people did heuristic evaluation on our medium fidelity prototype design based on 10 classes of errors. These errors were classified into 5 groups with increasing severity rating. They help us improve our results and get a better high fidelity prototype.

Category	# Viol. (sev 0)	# Viol. (sev 1)	# Viol. (sev 2)	# Viol. (sev 3)	# Viol. (sev 4)	# Viol. (total)
[H2-1: Visibility of Status]			1	1		**2**
[H2-2: Match Sys & World]			1			**1**
[H2-3: User Control		1	1	1		**3**
[H2-4: Consistency]		2		1		**3**
[H2-5: Error Prevention]			2			**2**
[H2-6: Recognition not Recall]			1			**1**
[H2-7: Efficiency of Use]		1				**1**
[H2-8: Minimalist Design]						**0**
[H2-9: Help Users with Errors]						**0**
[H2-10: Documentation]						**0**
Total Violations by Severity						**13**
Note: check your answer for the green box by making sure the sum of the last column is equal to the sum of the last row (not including the green box)						

Fig 16: Heuristic Evaluation Results

High Fidelity Prototype: Our High-Fidelity prototype was developed for iOS using Swift. We were able to implement almost all of our tasks without hard-coding anything. We also were able to address the heuristic evaluation results. A lot of the information from the Heuristic Evaluation was hard to understand or not well-written. We did add a “Free Period” label so people knew what a task meant, and also integrated the “Invite friends” screen into the standard iOS Messages environment to make it more efficient and easy to do.

Prototype Implementation:

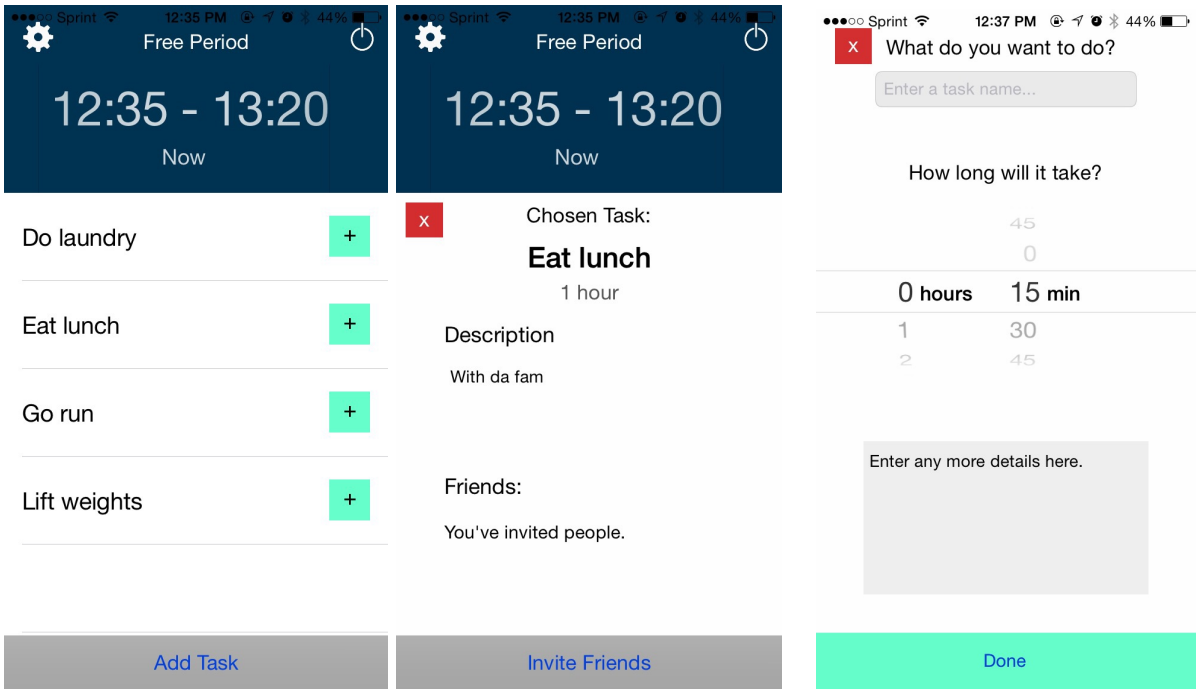

Fig 17: High Fidelity Prototype Add task screens

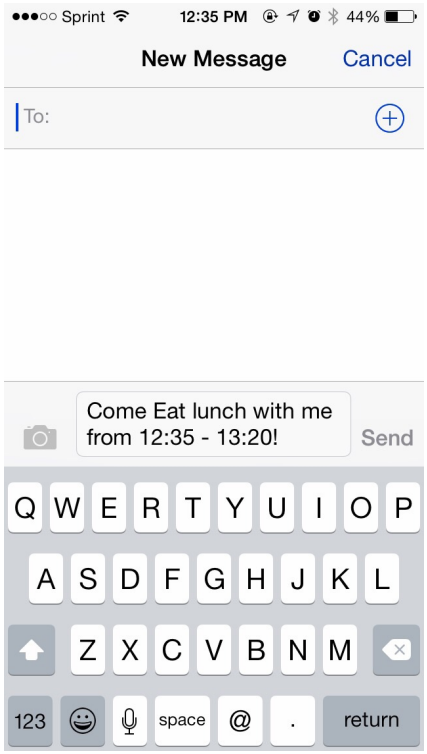

Fig 18: High Fidelity Prototype Invite Friends screen

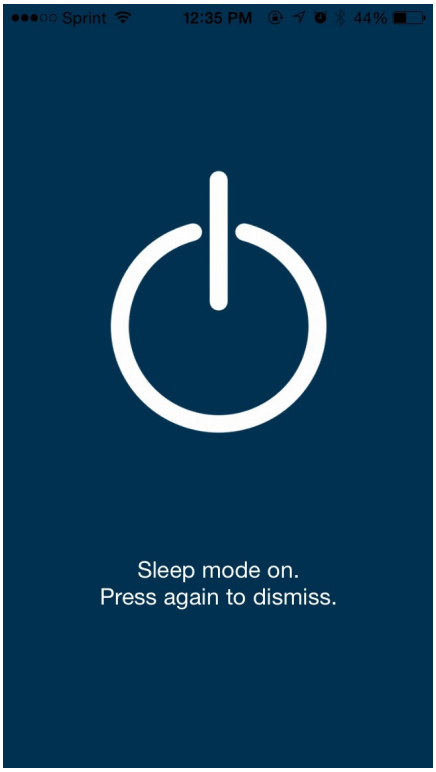

Fig 19 : High Fidelity Prototype Sleep mode screen

Development in Swift/XCode:

Advantages:

Swift made the implementation easier. It enabled us to develop all the tasks without much hardcoding or wizard of oz techniques. It is a high-level language that takes care of a lot of stuff like memory allocation. It is also type safe. Objective C is comparatively a much more difficult language which is also tedious to code. Swift is more concise and can be integrated with existing Objective-C code if written correctly. XCode 6 also has a feature called Playgrounds which allows Swift code to be interactive: you can check the functioning of your code in real time and shows you your results and progress. Also, developing on XCode is a good experience. It has a drag-and-drop GUI to design the storyboard, which makes it easy to add transitions and UI elements like buttons and text boxes.

Disadvantages:

The iOS developer (Matt) was completely new to iPhone programming and started learning Swift the day after Thanksgiving. As a relatively new platform, Swift doesn't have nearly as much help and libraries available online. We had to depend on attempts to translate Objective-C documentation into Swift documentation, so some problems we

faced were tedious to solve. The developer had to spend a while integrating Objective C libraries into his Swift code, which brought in a lot of nasty, untype-safe elements into the program. Overall it was a time-consuming and difficult process, but it was a satisfactory learning experience.

Future Directions:

Our code did not use any Wizard-of-Oz techniques or hard coded data. We were able to incorporate the calendar with our app. We were also able to add contact information on the phone to be able to message friends.

In the future, we would like to incorporate several things into our app. We still need to implement the Settings page in the to make it customizable (e.g. not searching for free blocks between 1 am and 9 am). We also are going to implement notifications that remind the user when a chosen task is coming up. We also might incorporate some machine learning approaches to our app to make it more intelligent. For example, we might learn a person's goals and desires to suggest activities that he might be interested in doing. We also might incorporate location information. For example, if a person interested in physical activities is near his gym and he has some free time, the app would give him a suggestion to go to the gym. Also right now the app is developed for iOS. We would like to do build it for more platforms, or maybe even build a version for glass or wearables.