

Team: *Don't Forget!*

Andy Hou

Elisabeth K. Olson

Kenneth Kuan

Kevin Chiu

CSE 490F

Assignment #9 – Interactive Prototype

5 March 2007

Problem

Modern life requires many accessories. It is not uncommon for a single person to have more than a dozen items that they need for their activities during the day. Furthermore, the items each person needs can change when their schedule changes from day to day. However, the human brain has difficulty remembering long lists and schedules, and it is quite common for a person to leave their house without a necessary item.

Solution Overview

The solution to this problem is to add RFID technology to a calendar interface to allow a computer to keep track of the necessary items. RFID tags would be placed on all important items, and the code on the RFID tag would be used to identify the item to the system. Those items, in turn, could be attached to events that require them. An RFID scanner and touch screen interface near the door would quickly scan a person as they left the house and remind them of any items they may have forgotten to bring with them, as well as giving them the possibility to recheck their schedule.

Tasks

The tasks we used to evaluate our system evolved slightly during the testing process. When we began our tasks were the following (ranked easiest to most difficult):

1. You are leaving the house, and although you're pretty sure you have everything you need for the day you use the scanner interface to verify. When you've verified your items, you also want to recheck your schedule before you leave.
2. You just bought a new wallet, and want to add it to the system.
3. You want to add a dancing class from 6-7 PM to your schedule, and want to remember to bring your wallet and glasses.

These tasks were supposed to explore as much functionality as we could support. To complete the first task, the user would have to figure out how to log into the scanner interface and interpret what the scanner told them. Usually when testing we set the scanner to automatically tell them they were forgetting something to simulate a faulty scanner, which would allow us to test whether or not they tried to re-scan themselves. This task also asked them to look at their schedule, which let us know how discoverable that feature was.

The second task let us test the interface for adding items to the system. The crux of the program is the ability to identify items to the computer for tracking, and if that part

is difficult or unusable, the rest of the program is just another (probably more expensive) calendaring program.

The third task evaluated the ability to not only add events, but also associate items with those events. Again, if adding items to events is difficult, the rest of the program is a glorified calendar, so it was vital that we created a good interface to support this task.

As we iterated over our design and added more features, we found that our tasks didn't fully explore the functionality we'd added, so we changed the tasks slightly. The first task remained the same, but the other two tasks were merged into one task where the user added an event which required an item which wasn't in the system yet, thus requiring the user to add the item as well. The third task was then replaced by a two-part task where the user was asked to first edit and then delete an event on the calendar. This change allowed us to test the functionality we had before, as well as testing the added functionality of modifying and deleting events.

Scenarios

Task 1:

To use the scanner interface to verify that they have all needed items, the user would first start at the home screen:

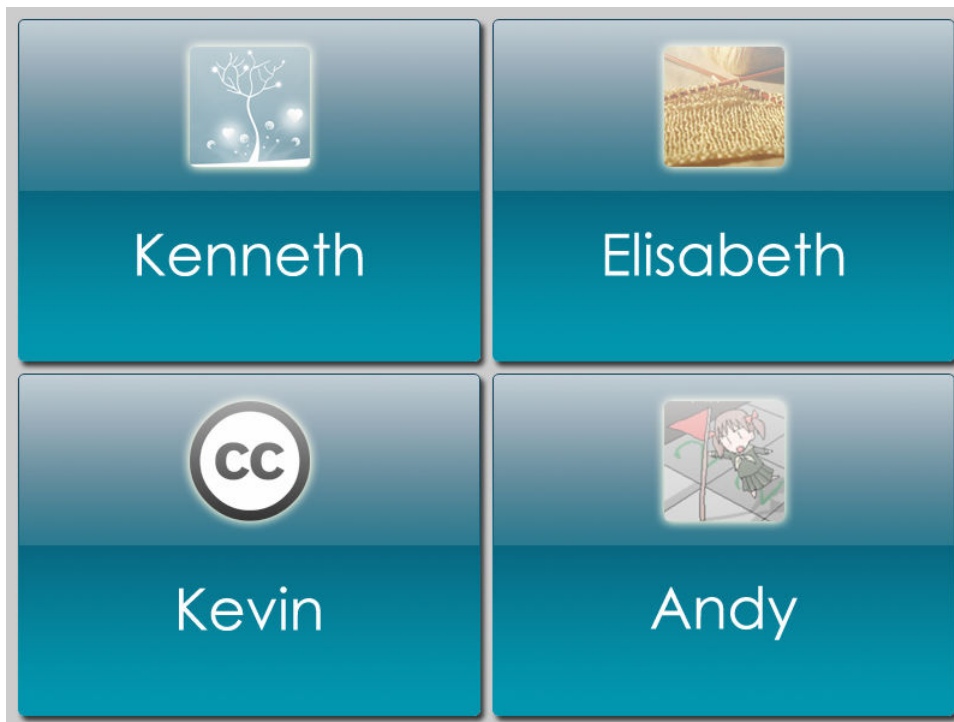


Figure 1: Scanner Home Screen

Assuming our user's name is Andy, he would then touch the section of the screen with his name on it, bringing him to the optional password page (we assume he has passwords enabled because he lives with three other people and wants to keep his schedule and items private):

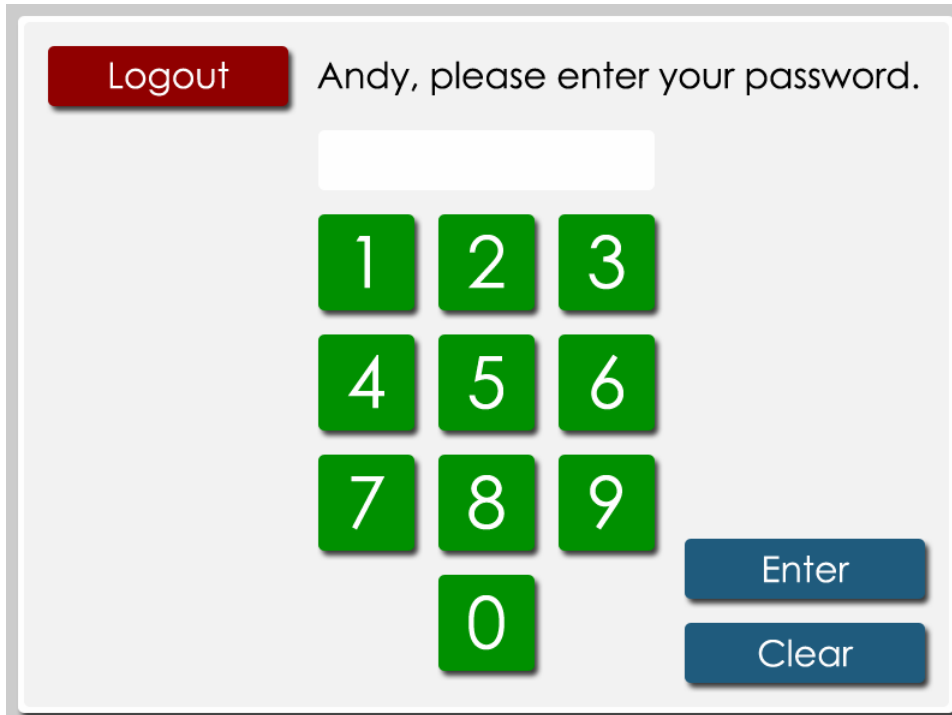


Figure 2: Password Screen

He enters his password using the green number buttons, optionally using the Clear button if he makes a mistake. Inputting the correct password and pressing Enter would bring him to his Items page. The system would have automatically scanned him at this point, so the page shows him the items that the system doesn't see but which are in his schedule. In this case, we assume the scanner is faulty and that it didn't see some of the tags the first time around:

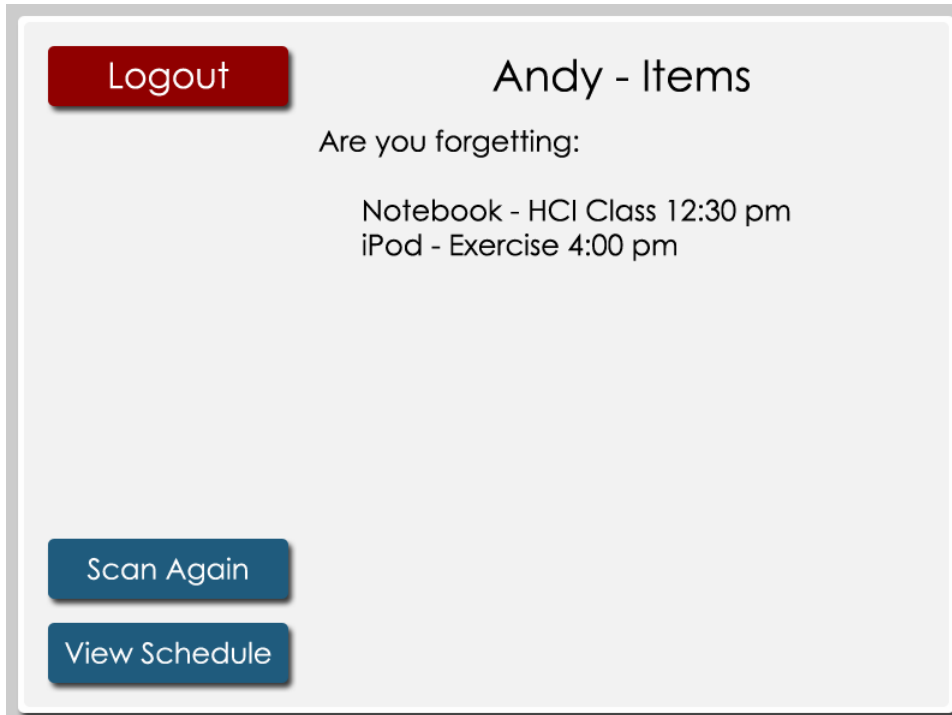


Figure 3: Items Screen - Forgetting

As Andy is rather sure that he does have his notebook and his iPod, he presses the Scan Again button, forcing the system to try scanning his RFID tags again. This time, the scanner works perfectly, telling him that he has all his required items:

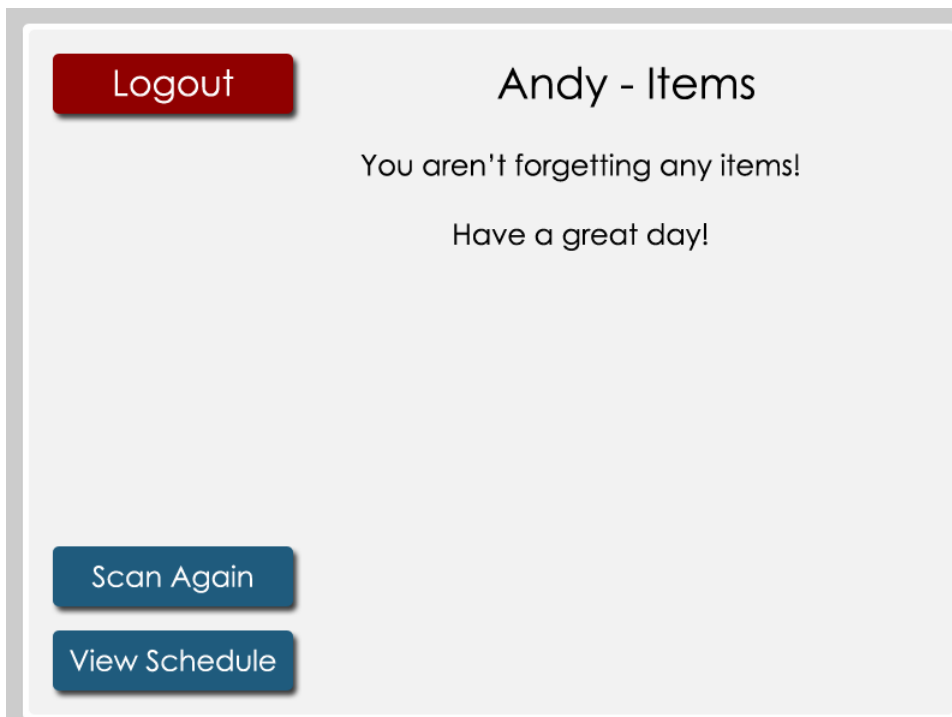


Figure 4: Items Screen – Not Forgetting

Before Andy is done, he wants to verify his schedule for the day, so he presses the View Schedule button, bringing him to the schedule page:

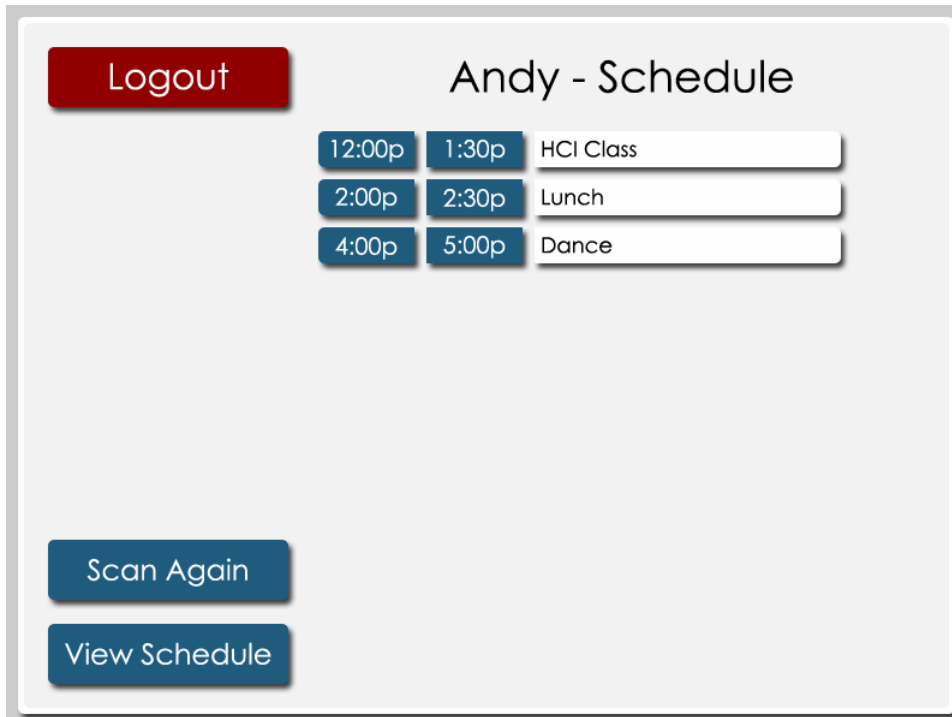


Figure 5: Scanner Schedule

Each item on his schedule is also a button he can press for more information about the event, and if he was busy enough to need more space for events than the screen allows scrolling arrows would appear on the right side. Andy is neither very busy nor interested in the extra details of his events, though, so after viewing his schedule he presses the Logout button to return to the Home screen and leaves for the day. If he forgot to press the Logout button, the system would automatically return to the Home screen after a certain amount of time had passed.

Task 2 (final version):

To add a new event and a new item, Andy would first start at the Calendar interface:

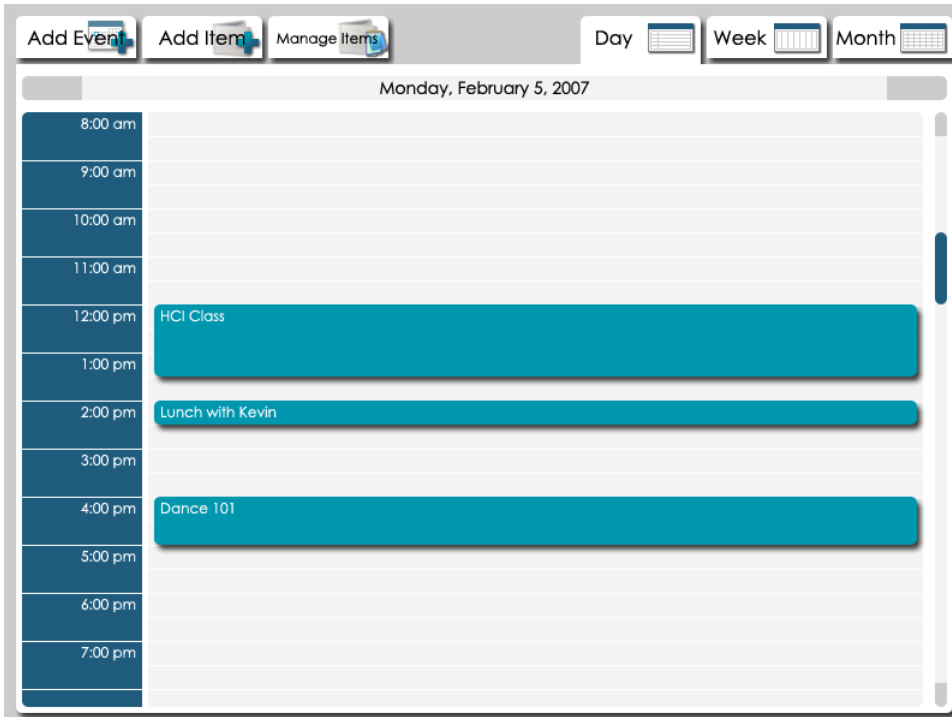


Figure 6: Calendar Interface

He would then either click a time in their schedule or click the Add Event tab, bringing him to the Add Event page:



Figure 7: Add Event Screen

Here Andy would fill in the details for the event, such as the name of the event, the start and end times, the location, and any other information needed. If he had brought up this window by clicking on the calendar, the time information would already be filled out automatically. To associate items with the event, he would select items under the All Items column and use the left-arrow to add them to the Items for Event list. As the item Andy wanted to add isn't here, he would then go to the Add Items tab:

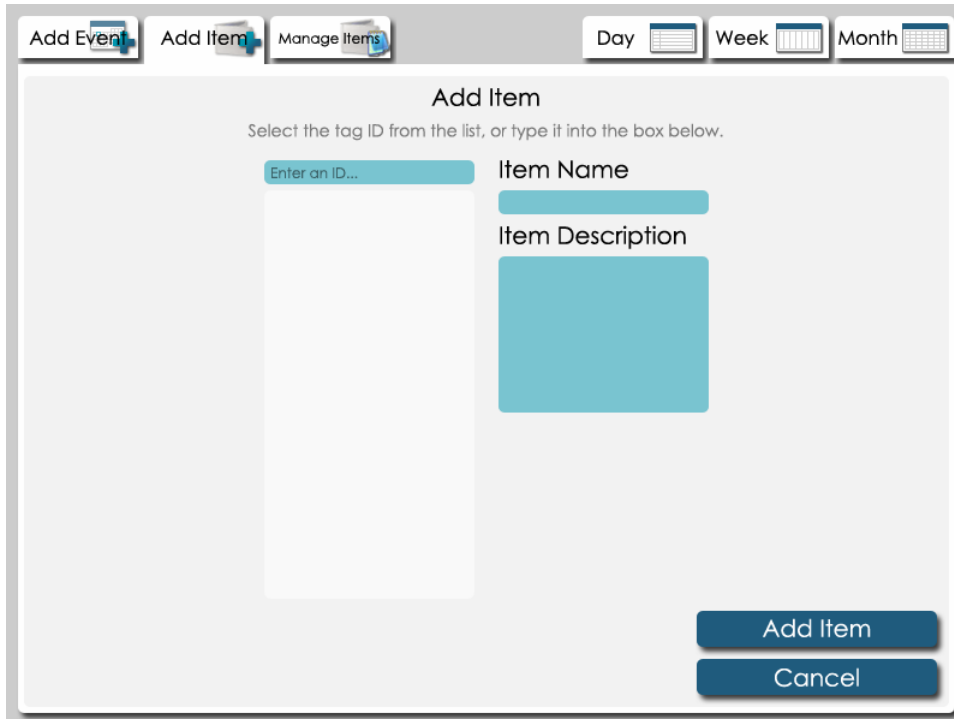


Figure 8: Add Item Screen

Here we make an assumption: we're assuming that the RFID tags we market with the system all have a unique ID printed on them, so when the scanner finds tags it doesn't recognize it will report them to the interface and they can show up here in an identifiable format to be clicked on. Alternatively, if Andy doesn't want to bring the new tag within range of the scanner yet, he can manually input the printed ID number, and the system will then be able to identify the tag whenever it sees it. After he has added a name and optional description for the item, he clicks the Add Item button, which brings him back to the Add Event screen, where he associates the item with the event and clicks the Add Event button to return to the calendar.

Task 3 (Final Version):

To modify an event, Andy clicks on it in the calendar. This brings him to the Edit Event screen:



Figure 9: Edit Event Screen

From this screen he can change any of the parameters of the original event, confirming the changes by clicking the Change Event button. Alternatively – and to satisfy the second part of the task – he could also delete the event altogether by clicking the Delete Event button.

Design Evolution

This design is the result of several iterations. We started with paper prototypes, meaning that we put the entire proposed system on paper and pretended the paper drawings were computer windows when testing on users. Example screens are shown in figures 10 and 11. An example of the “Human Computer” in action is shown in figure 12.

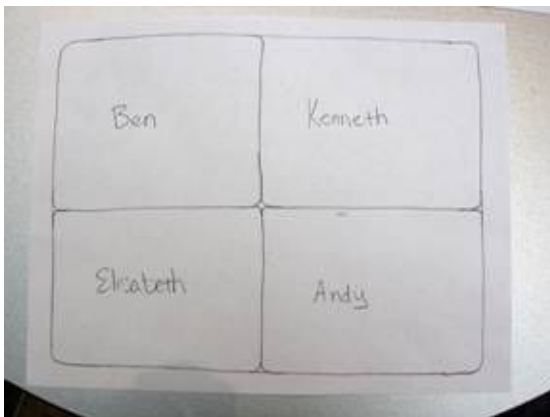


Figure 10: Paper Prototype Scanner Home

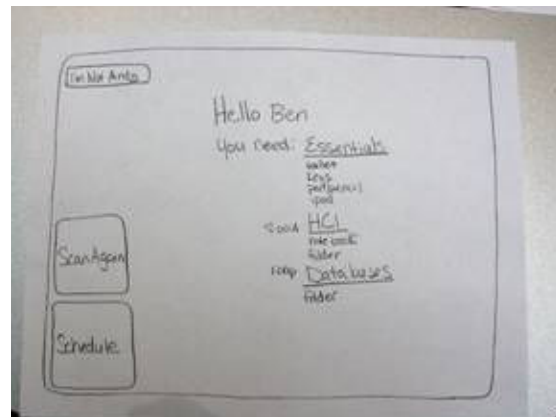


Figure 11: Paper Prototype Scanner Items

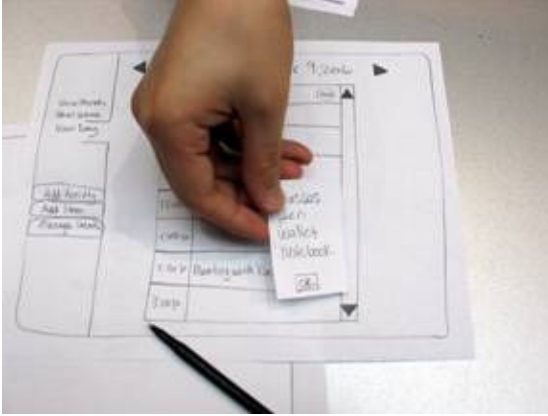


Figure 12: Paper Prototype – Human Computer

In the paper prototype, we didn't have any security implemented at all, and our users were concerned that if they used the system in a shared environment – such as with roommates – they didn't want to expose their schedule and items to the other people in their environment. Based on this, we added the password feature, with the idea that it could be optionally turned off based on user preference.

The second iteration was a low-fidelity prototype using the program Denim, which was created to help people design web pages. While it made our prototype look better than a paper prototype (example screens from the Denim prototype in figures 13 through 17), it actually ended up providing less functionality than it could before, as Denim doesn't allow state or input to be saved, and the majority of our system requires both. Things such as which window a user came from, which we could remember when “playing computer” in our paper prototype, were impossible with Denim. Also, when the user added an item or activity, we couldn't show it to them once it had been added. We tried to deal with this situation by adding some humor to our confirmation dialog boxes simply for testing purposes (shown in figures 18 and 19), but this backfired during a heuristic evaluation when our testers were simply confused by the text and considered it a serious usability problem. This is also the first time that having “Groups” (in figure 17, the starred items in the “Items Needed” box) of items was identified as a problem because it was hard to differentiate Groups from Items. We tried to make the two more distinct in the next iteration to fix this problem. The other problems identified by the heuristic evaluation were mostly aesthetic, such as the wording on buttons or the layout of buttons on pages.

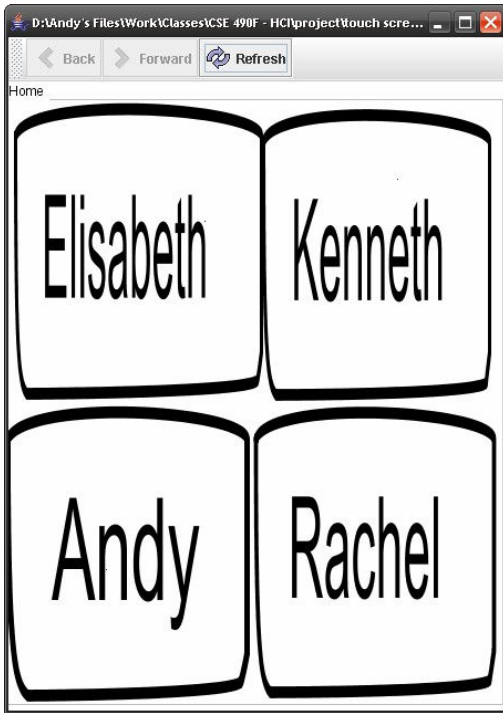


Figure 13: Denim Prototype Scanner Home

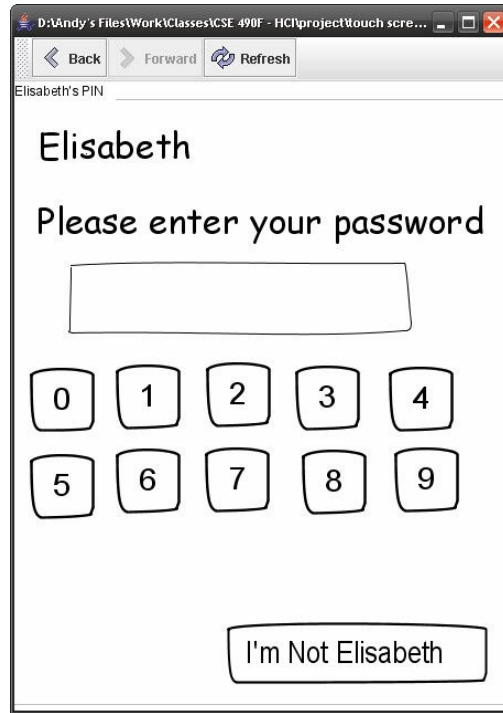


Figure 14: Denim Prototype Scanner Password



Figure 15: Denim Prototype Scanner Items

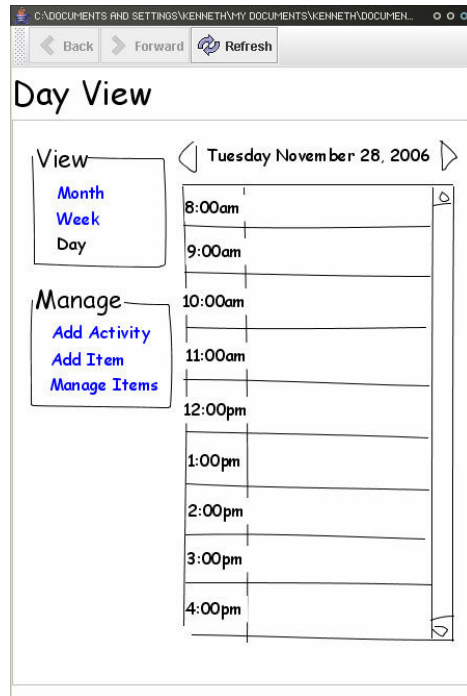


Figure 16: Denim Prototype Calendar Interface



Figure 17: Denim Prototype Add Activity Screen

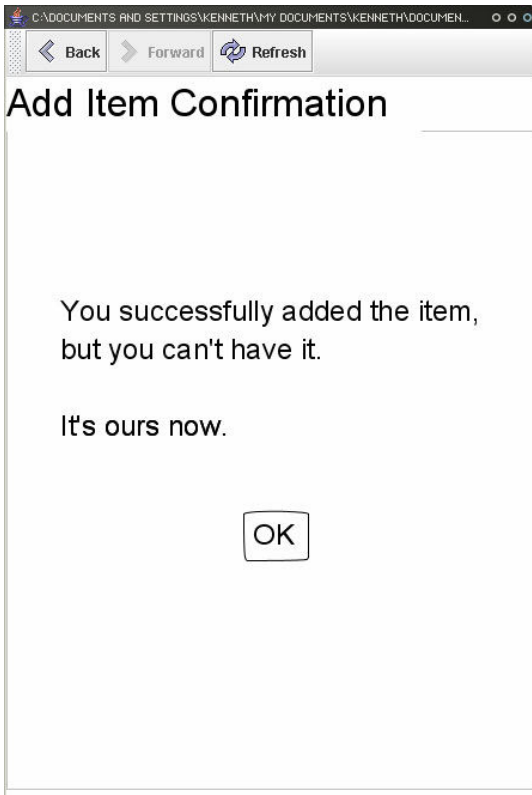


Figure 18: Denim Prototype – Failed Humor Message 1

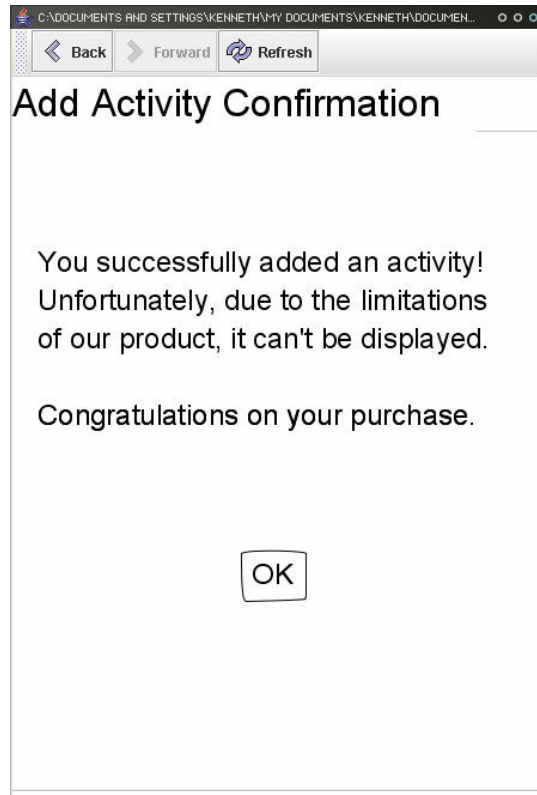


Figure 19: Denim Prototype – Failed Humor Message 2

After making appropriate changes based on the feedback from the heuristic evaluations, we put the two interfaces online as part of survey. The questions on the survey led the testers through the three tasks, which at this point were still the original tasks. The biggest problem our testers had was differentiating between Groups and Items. This caused us to rethink the idea of Groups of items and eventually decide to throw them out entirely and simply work with Events and Items. Our testers also indicated that they wanted more feedback from the system, because the Denim prototype was unable to show them the results of their input, such as showing an event on the calendar once it was created.

At this point, we switched development environments again and started working in Java, building an actual, working prototype. Aside from throwing out the idea of Groups, we didn't make many changes from our Denim prototype, as the limited nature of our testing meant we didn't uncover many problems in our design. For the first iteration in Java, we purposefully shied away from making the interface pretty so that our testers would focus on functionality over aesthetics. This meant that the system didn't look much better than it did in Denim, although the functionality was dramatically increased. Some representative screens from this prototype are shown in figures 20 through 23.

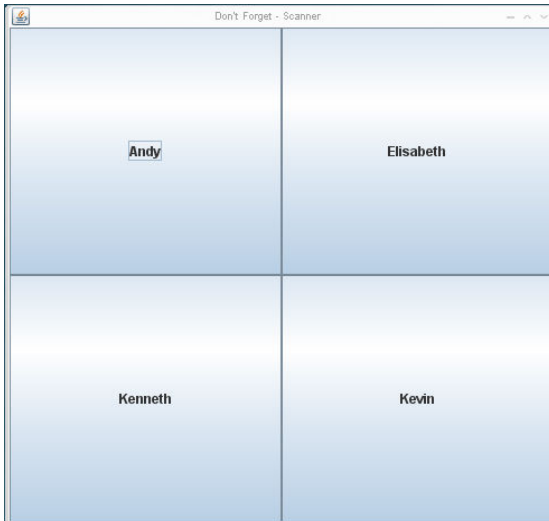


Figure 20: Java Prototype 1 Scanner Home Screen

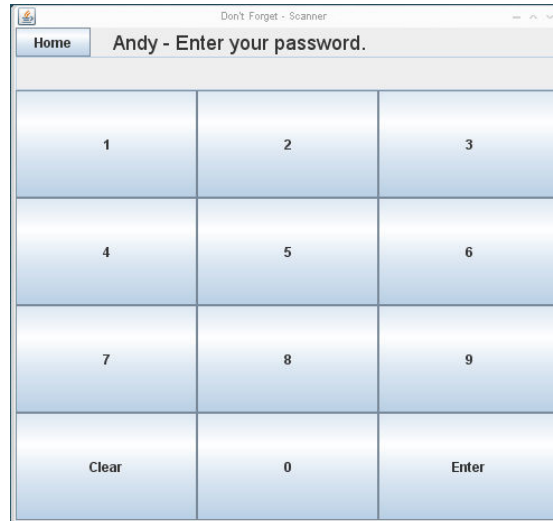


Figure 21: Java Prototype 1: Scanner Password Screen

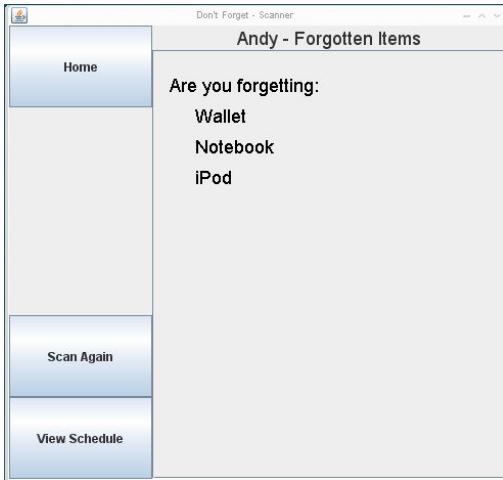


Figure 22: Java Prototype 1 Scanner Items

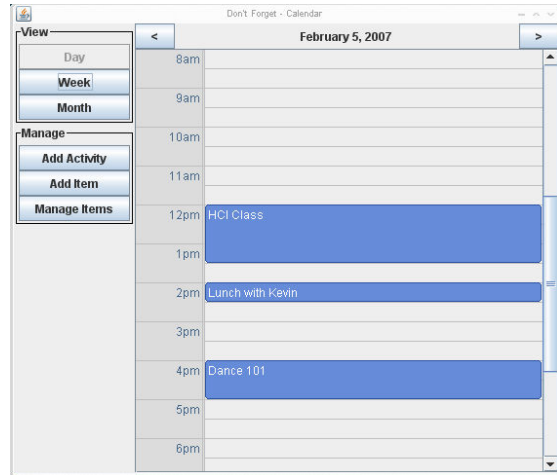


Figure 23: Java Prototype 1 Calendar Interface

Once this prototype was up and running, we tested on a limited number of users to find gross errors. This was by far the most useful phase of testing, as we found a few major functionality problems we hadn't really been able to examine before. The major problems centered around the Add Activity window (figure 24). The start and end times both default to 12:00AM, which is inconvenient for the vast majority of scheduled activities. Much more problematic, though, was the way items were added to the activity. In this iteration, the user has to highlight items by clicking them to add them to the activity. This is made more difficult by the fact that the items must be simultaneously selected, as simply clicking the items individually will only select the last clicked item. Furthermore, the list itself is confusing, as it implies that the items are already connected to the event.

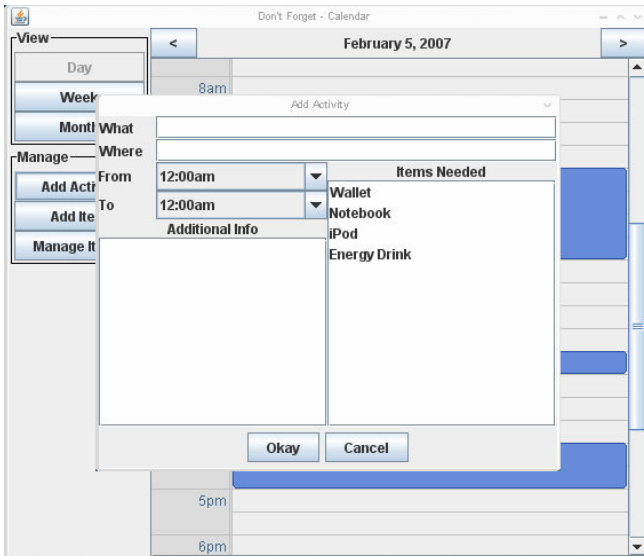


Figure 24: Java Prototype 1 Add Activity Window

The other major problems this iteration revealed were that the layout of the password page was flawed at best, and that the calendar needed to be more interactive. In this iteration, we only supported adding activities by clicking the Add Activity button,

and only supported modifying them by clicking the event on the calendar, then clicking the “Edit” button, then modifying the fields. The testers wanted to be able to click on the calendar to add an event at a certain time and wanted drag and drop features to be able to change the time of an event.

These changes were incorporated into the most recent iteration of the system, which is demonstrated in the Scenarios section.

Final Interface

To develop the final interface, we used Java for the underlying code and Photoshop to develop the images that are skinned on to provide aesthetic appeal. While Java was very simple to use to provide the basic functionality, and Photoshop was an ideal image editor, combining the two proved to be difficult, and took the bulk of our time. Ultimately, the full screen images created in Photoshop had to be cut up into usable bits to plug into the Java code, so a single screen could generate up to 30 different images, all of which had to be cut out and coded in by hand. However, the aesthetic change from the default Java widgets to the designed images is overwhelming, and well worth the effort.

In its final state, our interface supports the functions that examine the new concept of adding RFID tags to a calendar, but it leaves out a lot of the functionality necessary to actually use it. For example, the calendar backend is virtually non-existent, so the calendar is stuck on a single day in time, which would not be very useful if one wished to actually use the calendar. Also, although it’s easy to imagine how an actual RFID scanner would fit into this program, we didn’t focus on the physical implementation, and thus avoided the hardware difficulties to focus on the interface development. As is, the “scanner” is programmed to be slightly faulty, so even when the user is supposed to have everything (which, as there is no scanner, is hard coded to be always) there is a slight chance the scanner won’t see some of the items and thus report them as missing. We also left out the development of Manage Items, as we felt that would expand the scope too far to allow for meaningful iteration at each stage.

However, most of the system is working. The scanner interface, in particular, is virtually ready to go, requiring only a scanner and a connection to the database maintained by the program. Practically, of course, these two “minor” things would take a large amount of time to get working, which is why we’ve left them off here. However, a user can certainly log in, check their imaginary items, verify their imaginary schedule, and even simulate rescanning multiple times. All of these actions are demonstrated in the Scenarios section, in Task 1, and illustrated with figures 1 through 5.

The calendar interface, although it doesn’t work as a calendar in real life, does support the calendar operations of adding, editing, and deleting events. Furthermore, it supports adding items to the system, and allows those items to be added to events so the scanner will keep track of them. Again, these are demonstrated in the Scenarios section, in Tasks 2 and 3, and illustrated with figures 6 through 9.

Overall, the system is designed to be streamlined and simple. The number of screens is purposefully limited, although not at the sake of functionality. The calendar, like any good calendar interface, should be almost ubiquitous, which means that ideally when used in the real world it could be synced with a portable device such as a PDA or Pocket PC. Furthermore, using the calendar should be automatic, not something the user

needs to think about, so the functions must be simple and intuitive. This is even more true about the scanner interface, which will be used as the user is leaving the house. Morning is generally a rushed, hectic time in most households, and any system that takes more than a few seconds to use would be impractical at best. In our system, if the user doesn't have a password enabled, getting to the scan page is a single button push, with another single button to the day's schedule. We believe we've reached this level of simplicity without sacrificing any functionality the user might desire, and over all are quite proud of our design.