Assignment 3
Beta Release

Henry Morrison Jones and Ryan Spear

Introduction

After a disappointing alpha release, motivation was high to deliver a beta that we could be proud of. Doing so, however, required a massive rethink of our initial requirements. Though many of the problems that plagued our Alpha release were solved, new problems arose, including problems with testing and the use of local storage. A significant restructuring also saw our team drop from 3 members to only 2, which, though potentially affecting the speed of development, also brought with it easier communication and increased rapport. Alongside these new problems were also new accomplishments and although development was tricky, our beta saw the introduction of new accessibility features, notifications and the ability to edit pre-existing tasks. Our view for the future is much more optimistic than it was with our alpha release and we believe that with enough effort, we should be capable of delivering a final product that meets the goals we set for ourselves earlier in the year.

Adapted Requirements

Following our alpha, we understood that the beta release would require a complete rebuild. Our alpha release left us with three seperate projects across two development platforms and while the contents of these projects certainly showed promise, the decision was made early to rebuild a single project from scratch rather than integrate. This rebuild allowed us to reconsider the requirements we had initially set for our App. The consensus among the team was that the requirements we had set were too

ambitious and that given our current situation, we were better off building something much more minimal. Instead of offering daily, weekly and monthly views for our scheduling app, it was agreed that the app would use a single list approach, similar to a ToDo application. This change in design not only simplified development but also offered a more navigable solution for visually impaired users. Secondly, while the automatic scheduling system formed the basis of our app, having to build from scratch left us with no choice but to reconsider this feature. In order to keep with time constraints, the decision was made to move automatic scheduling from our set of core features into our set of desirables and so was not implemented for this release.

Accomplishments

Accessibility

A central focus with our UI-redesign was the ease of use for visually impaired users. For compatibility with Apple's VoiceOver software, we made sure that each of the UI components included an accessibility label. When using VoiceOver, these accessibility labels are read out to the user as a way to ease navigation. Text fields, buttons and various selectors were all given such labels and thus the application is indeed usable for a user with such a disability. Continuing with accessibility, we also adapted buttons to display icons instead of the text used in our alpha build. Our reasoning was that differentiating between symbols rather than text would be much easier for individuals with poor vision. At this point in time, these are the only examples of added accessibility for visually impaired users but as a baseline, we believe these are satisfactory in meeting the accessibility requirements of our project.

Notifications

The second accomplishment of our beta was the incorporation of notifications. Upon creating a task, the user can select a date and time by which they expect to complete

said task. When the task is added to the task-list, a notification is added to the operating system's Notification Center. Currently, notifications are set to occur 15 minutes before the selected time and include the task's title so the user is aware of what it is they should be doing without being in the app itself.

Task Editing

Another massive accomplishment during this phase of development has been the ability for a user to edit tasks. On the main task view, tapping an appropriate cell will now bring up an edit task display whereby the user may change the reminder time, whether notifications are enabled and whether the task is repeated or not. This addition makes the metadata associated with tasks more customisable and also increases the flexibility of the application.

Issues

Organisational

We found that there was a trade off in benefits now that we only have two members in our project group. On the positive side, communication, meetings, and task delegation is much simpler. We have found that using traditionally efficient task delegation methods such as the Agile Framework and Gantt Charts took more overhead time and effort than it took to meet with one another weekly, update the weekly task list throughout the week, and iron out issues throughout the week using text.

On the downside we now both carry much more responsibility for all steps of the development process. Where previously we could've had someone focused on testing, or focused on UI, we now both take on large a large workload for every part of the application.

Before the group split, we found that group meetings were usually ineffective and recurrently circled around the issue of platform choice. With this no longer being a distraction, our group meetings have been more productive, contain substance and now revolve around discussions of ideas. Our mistake earlier in the paper was not being staunch and clear with one another that we needed to choose a platform, a direction, and focus on them. Instead we were much too willing to allow side projects and safety nets incase our ideas didn't work out. This sounded smart at the time, but resulted in the group working ineffectively on completely different projects.

Our github wiki contains a weekly to do list, that we update tasks once completed, or add on new tasks to be brought up at the next meeting. Every week we meet, begin with a debrief on the previous weeks work, a quick brainstorm, and create a new to do list based on the discussions. Finally we delegate tasks based on our skills and which parts of the app each member had mostly influenced. We have found this works really well. The key is making a task list which is ambitious enough to feel accomplished once you complete it, but not ask so much that you constantly feel overwhelmed.

Looking back at how far we've come since the previous deadline, we have an optimistic outlook at what our app will look like in its final release. The challenges facing us in the future will be to make sure our amount of work doesn't fall off with the looming exams approaching us. This will probably be the biggest issue facing us as a small group looking forward, where we may have to work double the amount of time as a group with four members. We believe the cohesiveness we have, by only having two members, may help us to off-set the overhead time spent passing information through a larger group.

Testing

So far we have exclusively tested through the xCode simulator and Accessibility Inspector. These are amazing resources for checking all of our interactive buttons,

making sure our data flows through viewControllers correctly and confirming that our labels and elements display the most helpful accessibility text and hints. Most of the problems we have encountered with our testing phase so far have have been small and easily fixable bugs. Examples of these are creating tasks in the past, having reminders going off at incorrect times, and having incorrect settings showing when editing an existing task. All these specific errors were fixed. Our focus on testing now moves away from backend data flow into the accessibility side of the application. All our elements, labels and buttons have accessibility labels and hints that will be read to the user if selected. But just because the app is accessible doesn't mean it is optimised for our visually impaired users. The final quarter of our project will be focused on creating, and testing, an appropriate UI for our visually impaired audience.

The largest issue with our testing phase is being unable to get our Application onto either of our iPhones to test the VoiceOver functions correctly without a developers licence. Using the xCode Accessibility Inspector is a great help but doesn't let us hear what a visually impaired user would be hearing with VoiceOver turned on. Neither can we make sure that touch commands, such as left/right swipe to select previous/next elements work perfectly.

Storage

We have come into a couple of issues when dealing with xCode's specific way of storing data locally through its CoreData feature. First of all, as outlined in our initial report, there is a risk in depending solely on the lab machines. When creating any sort of database system using CoreData on the lab machines, xCode freezes up and becomes non-responsive. Our research found no answers to this problem, and the issue persists throughout any machine in the Owheo building, so we decided to move on and come back to it a little later.

One of our members has a personal Mac so we were able to start creating a CoreData database when in was high enough on our priority list. The database is created and setup on the /local branch in our github repository, but we chose not to implement a merge so close to our Beta deadline to avoid any merge errors and as task editing and deletion has not yet been fully implemented with core data.

We do not believe that implementing the appropriate methods for data retrieval and storage with this new branch will be a problem, but we chose to play it safe and submit a fully working Beta albeit one that fails to use Core Data.

Looking Forward

Our greatest concern for our beta was making sure that most, if not all, of our data flow and backend worked correctly without any bugs. This is the basis that we built on to ensure that the remainder of the course could be spent on enhancing the visually impaired user's experience. We believe that almost all of this will come from intelligent design of the user interface, colour schemes, and appropriate font choices.

We have a list of UI additions to be made before the final release. We want to focus initially on creating a task list that is not a simple list of cells ordered by date, but instead clearly identifiable tasks ordered by urgency. By making the tasks clearly distinguishable and colour coordinated we are focusing on accessibility. Specifically, the fact that when using iOS VoiceOver, the user must tap on an element to have it read aloud to them. We want to make anything that would be needed to be read, clear enough to not be mixed up with any other elements when trying to select it.

This focus on making elements clearer extends to both New Item and Edit Item pages. So far we have purposely made elements span the entire width of the screen to make sure impaired users do not miss out on any fields, which we thought was a good start. We have much higher aspirations for the UI in these viewControllers. To point out

specific examples, we want to remove all small switches such as that for reminders, and create larger, input-responsive buttons, that make it clear that you have a made a change. This is outlined in our group meeting document in the Wiki. Our previous idea, and currently implemented, was to have label fields that could be easily tapped on to read aloud to the user their current selection of data fields. The idea behind this was, instead of going through the date selector and selecting each individual time section to double check input, the user could simply tap the Label field to have their selection read to them. Also having a label telling the user whether they have turned on reminders for instance, so that a visually impaired user who cannot properly see the reminders switch knows whether it is on or off.

A development on this idea, combined with fields that give users clear visual and auditory feedback as they interact with fields, is what our current focus is on. We feel as though the next steps we take will be much easier than the last. Now we have experience working with Swift and our focus is now solely on iOS development.

Deployment

To deploy and test our current build, download or clone the repository found at https://github.com/henrymjo/345App. After the repository has been downloaded, open the Xcode project and select the active iPhone scheme you would like to run the app on. The current build is not compatible with iPhone 5 and 5s but we will look to change this for final release. Once the model has been selected, the app can be run by selecting the ‘build and run’ play button from within xCode itself. It is important to note that the app has been developed using xCode 9.3.1 and using earlier versions of xCode may not support the build.