COSC345: Assignment 4
Final Release

Henry Morrison Jones and Ryan Spear

Introduction

Our initial aim was to create an easy to use time management app, which, through it's simple user interface and accessibility controls, could facilitate use from a visually impaired user. While our original plan for a 'self-scheduling' algorithm was never implemented, the intention for creating a scheduling system that required as little user input as possible persisted and manifested itself in the form of our minimalistic design.

Currently, the build is stable for iPhones 5 through to X and, although we explored the possibility of extending use to an Apple Watch, time constraints prompted us to restrict use exclusively to the iPhone. With regards to design, we made an effort to ensure that no view would ever contain more than 4 or 5 interactive components. This rule was used to simplify use both for abled and disabled users as we believe that overloading such components hinders the app's navigability, especially for visually impaired users. An exception to this rule can be found in the TaskList view, which adds an interactive component for each task created by a user, as tasks can be tapped to be edited and swiped to be deleted.

While the core features and aims of our app have changed to a large degree, we believe what we have built is a viable time management product that caters to visually impaired users. Although the transition from a scheduling style app to a 'To Do' style was not intended, our understanding is that this simplifies use and still satisfies the majority of the core features we had set ourselves in our opening proposal.

Features:

Adding Tasks

A user can add a task by tapping the ‘+’ icon in the top right corner of the initial Task List view, whereupon they will be presented with the New Item view. Rather than including all task customising options within this view, we decided to create buttons which would segue to controls for managing these options separately. This avoids clutter and allows a user to focus on a single task attribute at a time. Through this view, a user may set a task description or move to other views that allow them to set the respective attributes of a task. This customizability includes setting the urgency of a task, creating an estimated task time and also scheduling a reminder for ‘x’ amount of days in the future. Tasks created with a reminder in 0 days will not have a reminder set at all and should expect no alert. It should also be noted that alerts are always set to go off at 6:00am on the day selected but we intend to make this customisable in the future.

Editing Tasks

Created tasks also have the ability to be edited through tapping them in the main Task View display. Here, the New Item view will be presented, with task attributes, pre-loaded with what was originally set. We also made sure that reminders that are initially set are removed from the alert center and replaced with the edited reminder date.

Persistent Storage

Tasks are all added to a Core Data stack and thus, provide persistent storage beyond closing the app and turning the phone on and off. Tasks that are deleted are not only removed from the initial task list view but also the Core Data stack itself, ensuring that dead tasks are not occupying memory elsewhere. Alerts follow the same principle, and so a deleted task will have its corresponding alert removed also, even if the task is removed before the reminder date.

Accessibility

Our app also provides a basic black and white ‘Colourblind mode’ as the urgency colours we have used are known to clash from a colorblind user’s perspective. At present, this mode presents all tasks as a grey cell but we will look to include alternative urgency colours, which do not clash at all in future releases. All elements within every view have also been provided with accessibility labels. These labels are read out to a user when using VoiceOver and as such, facilitate use for those who are visually impaired. Fundamental checks have been made through the accessibility inspector. Extensive testing using the VoiceOver inbuilt to the MacOS also ensures that this feature is working as intended. Without a developers license, testing has yet to be completed on a physical device, however, if we continue with our idea, we expect this to work much the same as it does through the simulator.

Intuitive Ordering/Presentation of Tasks

Currently, our application orders tasks by urgency. Here, tasks with a high urgency attribute appear at the top of the list, with lower urgency tasks appearing down the bottom. From a user’s standpoint, we believe that this encourages a user to focus on more pressing issues, rather than dwell on tasks that carry little value. Our intention is to make this customisable in the future so that users may order tasks in the best way they see fit.

Accomplishments Since Beta Release

Unit tests

Although we didn’t spend as much time testing as we would have liked, we did manage to create a number of unit tests which check whether the app is successfully passing task information between the pages accessed through the NewItem view. These tests were created to guarantee that a task would not lose any of its attributes during the task creation process, as our UI Redesign meant that multiple views were switched between before a task was added. These tests can be found in the ControllerSegueTests.swift file and can be run through pressing command + u. Currently, all such tests pass, although we expect to expand the number of unit tests in future releases.

Core Data

A massive shortfall of our beta release was the lack of persistent storage. Despite having this implemented on a separate branch, too many changes were committed on the main branch and merging the two proved to be near-impossible. Thus, we decided to re-implement core data following our UI overhaul straight to the main branch. At present, our model includes a single 'Task' entity with attributes for date, time, title and urgency. The CoreDataStack.swift file provides a NSManageObjectContext that allows interaction with the set of Core Data. This managed context is used to populate the main table view for tasks and is also passed around when creating new tasks so that tasks can be added to this stack, rather than included in a global scoped array or dictionary.

UI Overhaul

After learning through to the beta stage of our concept, we decided to adopt the "throw one away" method and start from scratch. We identified that our beta was not clear nor intuitive enough to be the interface that we could justify releasing.

Using our original idea of a list format for our tasks, we have changed the way that these are displayed. Each task has a colour associated with it to identify urgency of that task. Red tasks are of immediate attention, yellow tasks show they are of medium importance and green being the least important. Tasks in our home view are ordered by most important to least important, so the user can easily identify the next most urgent task to complete. Task cells also use a bigger font size, to continue with our theme of bold simplicity.

Building upon the clarity of the task list, the user can select how long they think a task will take. Longer tasks span over more area of the task list to convey to the user that these tasks will take more work. Just like our Beta, users can tap on a task cell to edit any of its values.

Some of the biggest and clearest changes are in our new interface for creating a new item. Previously, we tried to jam all of the task options into one overwhelming view. This time we decided to strip down the interface, making each view cater to a specific setting. For example, instead of having a page to set every setting, our new item view acts as an index page to take us to different task values. Tapping the Urgency button takes the user to a page to easily select the urgency and segue back to the index page. This is the same for reminders and time.

Colourblind mode

After researching, we knew that some of the colours we have used for the urgency indicators may be tough to see for users with certain kinds of colour-blindness. Considering our app was made for those with visual impairments in mind, we believe it was only natural to cater to all levels of color-blindness too.

The colour blind option toggles between our usual UI and a grey scale equivalent. We toyed with the idea of having the colour-blind mode change the colour scheme of our urgency indicators to something more neutral, but decided to go for a grayscale theme. The first reason was to cater for even the most extreme cases of colour-blindness. Furthermore, we thought changing the colour theme would be more of a detriment than a help, confusing users who possibly didn't realise they were in colourblind mode. While this grayscale theme serves as our baseline solution, we intend to further our research into colourblind tendencies so that tasks may have their urgency indicated by other, non clashing, colours in the future.

Issues and Shortfalls

Testing

While basic unit tests have been created for the segueing between views during a tasks creation, we have failed to implement any UI unit tests. This is due primarily to our lack of experience with testing in Xcode as well as the timeframe that we have been restricted by. Without a developers license, we have also limited our own use to the simulator alone. Despite facilitating the same essential use, we recognise that the running of an app on simulated hardware is not substantial enough to deem the release fully 'stable'. We have however, been able to replicate use effectively and, thanks to the inbuilt VoiceOver capabilities of the MacOS, are relatively certain that a visually impaired user can operate our product with relative ease.

Introductory Screen

We had originally planned to have a screen that appears on the initial, first time, opening of the app. This screen would give a small walk through of how to use the features on the app, allowing a user to be instantly confident in how the app functions. A combination of time and technical issues kept us from including this feature in our final release, though, we believe that the app has been designed with enough clarity that this feature isn't a necessity. The accessibility labels and readme are also available for users to clear up anything a user may be confused about.

Settings and Customizability

Our initial aim was to have a more customisable and tailored experience for the user, but were unfortunately forced to make such settings take a backseat in lieu of more important features.

We managed to implement a colour-blind mode, which changes the colour scheme to grayscale. While this is good, we think that being able to choose from a couple different colour-blind themes or choose element colours individually would have been a good addition.

Our alerts work well, but are restricted to going off at 6am. This was a mixture of trying to keep the UI as simple and intuitive as possible, but also not having time to implement an option to choose a reminder time. We think it would be a good idea to keep these separate. Mainly because our app isn't a calendar, but a todo list. Tasks will often need to be completed *by* a certain date/time, and not *on* a certain time.

These options were the ones that jumped out to us as needing the most initial attention, however, we wouldn't want to compromise the simplicity of the app by adding too many customisable features.

Looking Forward

Our goals looking forward would be to implement some of the functionalities that we didn't get a chance to finish off.

Firstly the introductory page of our app with a small walk through would be the priority. From there we have an abundance of choice. We would like to have the colours for the urgency customisable for users than struggle to see the default colours, but want more than the grayscale option. We would also want to expand into the apple watch to get reminders and add a simplified version of tasks while on the go.

Although we would like to add more features to the app, we think that it does a great job of what it was designed to do. We would be very wary in adding too many more features that take away from the core idea of the app, that it is simplistic and very easy to identify features visually.

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. 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.