COSC 345 Assignment 1

Outline

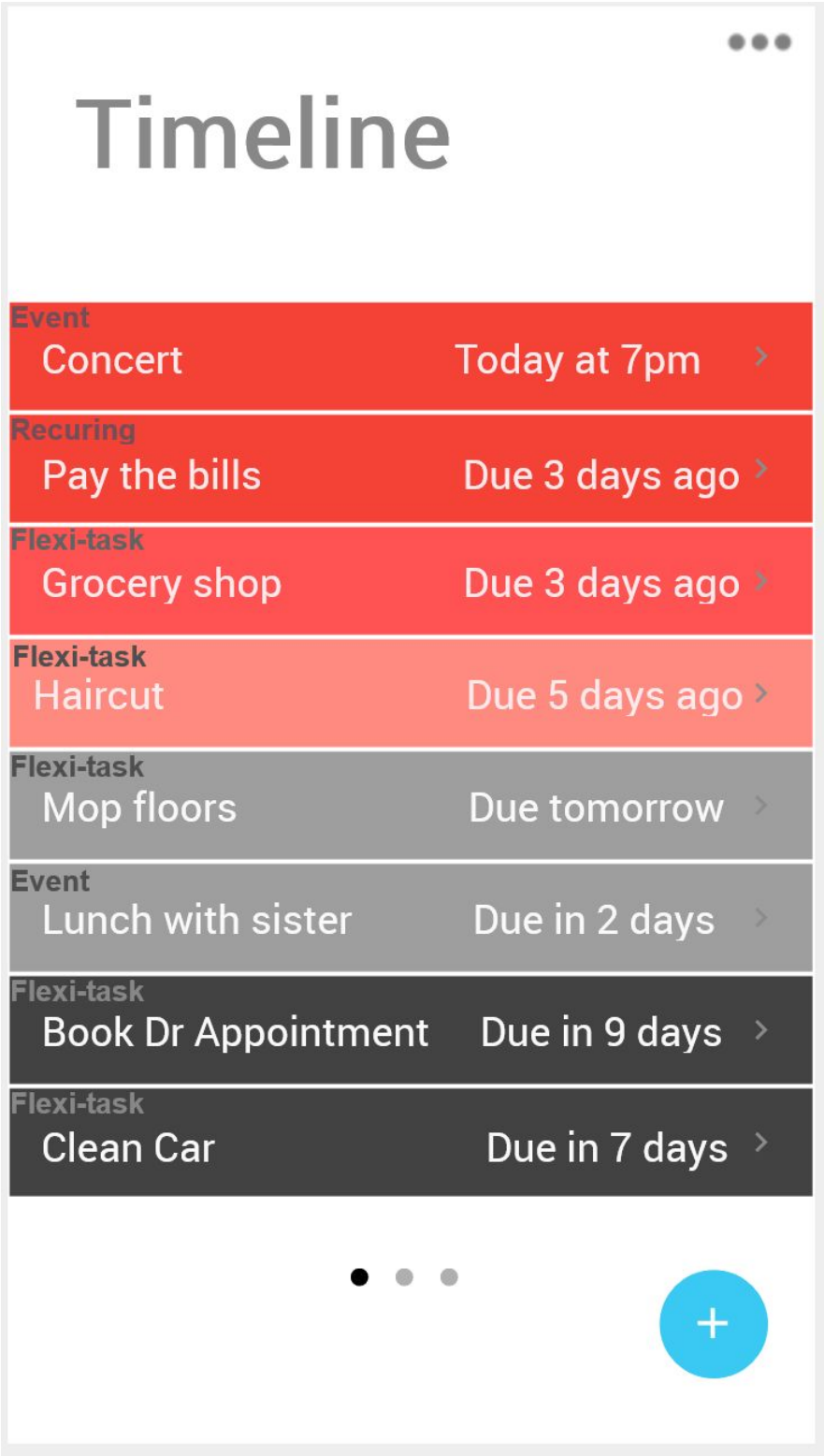
Our app's goal is to facilitate reminders for both fixed and non-fixed tasks. When you create a reminder, you can either select the type to be for an event, a recurring task or a Flexitask. An event is a one time reminder for a fixed task, for example "going to Sally's Birthday on Tuesday" whereas recurring tasks are repeating tasks that must be completed on a specified date, for example: "paying a power bill every Monday". Flexitasks are the unique addition that we have come up with. They are repeating tasks which don't have fixed calendar schedules. Maybe you clean the fridge roughly once a week, but you don't care if it's on a different day every week, or perhaps you visit the Doctors every six months, but a visit every five or seven months is okay too. After a Flexitask is completed the next reminder will be rescheduled based on the *last time* you did that task, as opposed to the set out timetable. In this way, we allow our users to have a more flexible schedule as they plan their lives.

Features

Our app prioritises events and recurring reminders first and foremost, as these need to be completed on a fixed date. Flexitasks are below these as they are (as the name suggests), more flexible. They are organised based on a urgency algorithm which assigns weights to determine which tasks are most overdue taking into account they type of task and frequency. For instance, our app will realise that a weekly task that's 5 days overdue is more urgent than a yearly task that is 2 weeks overdue. This urgency rating determines how high up a Flexitask is displayed on the timeline as well as the color code assigned to the task. The task's color slowly changes from grey to red depending on the urgency, thus making it easy to quickly see whether you have any tasks requiring your immediate attention.

Our app also keeps a log of all the tasks you create and complete. This allows the user to view the last time you they completed an activity and to what frequency, for example how many times you have visited the dentist this year. In the case of a Flexitask, you can see how often you have completed that task as well as whether you are usually late in doing so.

Diagram 1 below shows an design mockup for our XML layouts



As you can see, tasks are displayed based on their urgency; due events and recurring reminders (fixed tasks) are displayed first, followed by the overdue tasks. Upcoming tasks are shown after this. Fixed tasks are organised simply by chronological order, but Flexitasks take into account how overdue or urgent they are. For instance the Flexitask “Grocery Shop”

is displayed above “haircut”, despite being three days overdue verses five days overdue. This is due to “Haircut” being a *yearly* task that is five days overdue and “Grocery Shop” being a *weekly* task that is 3 days overdue (thus it is more urgent). Similarly, the upcoming task “book Drs appointment” is shown above “clean car” because a yearly task that is due in 9 days is more urgent relative to a weekly task that is due in seven days.

Our urgency rating works as a fraction:

Number of days since task was last completed / period (ie: yearly)

The larger the fraction the more urgent a task is

Ie:
Grocery shop > Haircut
(7+3)/7 >(60 + 5) / 60

Doctors Appointment > Clean car
(365-9)/ 365 > (7-7)/7

Development process

The members or our team are Jerry, Ryan and Jaydin. We have split the developmental responsibilities between us to make the process of creating this app as efficient as possible. Ryan is going to be in charge of the design/layout of the app, focusing on placement of views/buttons/text to make it as easy as possible for the user to navigate and find the things they are looking for. Jerry will we focusing on implementation of data storage/persistence using SQLite and Jaydin will be focusing on the java coding required to connect Ryan and Jerry’s work together. We will be helping each other and keeping each other up to date on our progress so everyone will be involved in these tasks and learning how everything works along the way. When someone is making an app decision or design/code addition, it won’t be added to the master code until everyone in the group as reviewed and agreed to it. We will be using GitHub to maintain version control and allow us to work individually as well together.

Decisions

We will be building out app using android studio, utilizing the languages of java, SQL Lite and XML. Because we are new to this environment, we have decided to work through Google’s Udacity tutorials [1] along with various YouTube tutorials. These tutorials cover a

range of topics that will be relevant to our app such as fragments, SQL database management and content providers. Our estimations are that the app will take 5-6 weeks to get to an alpha stage we are all happy with. These estimations are taking into account all of our work schedules, extra activities, other paper workloads etc. We have already started testing a basic list using recyclerview, this will act as the foundations of our app - using the recyclerview tutorial on android studio as a guide[4].

Risk Register

A risk register is a master document shared between developers, detailing each active risks involved in the project; including a description of the risk, the date identified, possible solutions and a target date for completion. We need to be proactive and keep track of the risks associated with our project that may prevent us from finishing our project in a timely manner. Our register so far is almost empty, but as we work through our project, we will encounter more risks and add them to the register.

ID	Date Found	Description	Impact	Mitigation Plan	Resolved (Y/N)
1	17/03	Team's inexperience using Android Studio	Learning a new developer environment and languages could take more time than we anticipated.	Work through Google's udacity program as quickly as possible	N
2	29/3	Memory Management	We found with our initial prototype that each time we added a new textView, the performance of our app was impacted severely. Because our user could add a number of different tasks, this is an import issue to address.	Using recyclerview we can reuse the textviews as they scroll off screen. This means we will only be loading the necessary data into memory, as opposed to loading everything into memory and creating separate text fields for each item. Even if the user has a dataset of 100 different reminders, we will only need to create enough viewholders to fill the screen (as our app recycles them as the user scrolls).	Y
3	30/3	Time management	With holidays, different work schedules - it's important we manage our time efficiently	Using time management software, that incorporates our work schedules and availability. We will also create extra time/"catch up" periods in case we fall behind on work.	Y

Disability Support

The disability we have chosen to support in our app is visual impairment. Users with visual impairment, whether it is blindness or color deficiencies use apps quite differently to non impaired people. They have to use mobile phones and tablets with screen readers (talkBack on android) to allow them to navigate and use apps and good design is very important to make sure our app will work for them.

As a few examples of features we are going to target:

- We will be using content descriptor features for relevant items so that screen readers can properly relay what information is on the screen. This will be ‘null’ for irrelevant items as to not slow down the process of navigating through the app.
- We will be using focusable containers from groups (a list item, its content, urgency and date) to minimise swipes that visually impaired users need in order to get all the information they need.
- Proper color contrasting as stated by the WWW Consortium [2].
- Queues other than color for buttons/options for color deficient individuals.
- Large touch targets to aid visually or even motor impaired people (the recommended minimum here is 48x48dp [3]).

References

1. Udacity.com. (2018). *Developing Android Apps | Udacity*. [online] Available at: https://www.udacity.com/course/new-android-fundamentals--ud851 [Accessed 14 March. 2018].
2. W3C Web Accessibility Initiative (WAI). (2018). *Colors with Good Contrast - Web Accessibility Perspectives Videos*. [online] Available at: https://www.w3.org/WAI/perspectives/contrast.html [Accessed 23 March. 2018].
3. Developer.android.com. (2018). *Testing Your App's Accessibility | Android Developers*. [online] Available at: https://developer.android.com/training/accessibility/testing.html?utm_campaign=android_update_appsaccessibility_051717&utm_source=anddev&utm_medium=yt-desc [Accessed 20 March. 2018].
4. Developer.android.com. (2018). *Create a List with RecyclerView | Android Developers*. [online] Available at: https://developer.android.com/guide/topics/ui/layout/recyclerview.html [Accessed 25 March. 2018].