Once specs are finalised and report+git checked (after Wednesday):

* Draw prototypical screens for each panel
* Make the functionality
* WindowBuilder it up
* Make graphics

The elderly generally aren't a fan of "hurry up", "you don't have much time left", etc, which are the ideas given out by a count down timer.

Make use of all the space on screen

Not colourful backgrounds - messy, distracting, confusing

We had been thinking in terms of modularity. All code and features are mode with the idea of extending it in the project in mind, as if this is the first iteration in a software development process, rather than a pair assignment that will get split up and we'll each do our own thing afterwards. Therefore, we kept A3 simple, and fairly general (aka rough in some places, not looking like a typical  submission on completing an assignment, and no where near prefect). In refactoring the A3 code, I already had in mind some features I thought of for the final project, and also things we considered "beyond the scope of this iteration". This being the foundation prototype, I refactored it in such as way that would make it easiest to add those features in. Therefore, when starting the individual part, all the infrastructure was already set up nicely, and mostly it was just a matter along the lines of adding a couple new variables, or a case to the switch statement.

It was also from this sense that we decided pair programming was the best, because the foundation is the most important part, and if we saw A3 as the foundation on which both of our 40%-worth project is to be built on, it only makes sense for us to make every decision together, and stop each other from being carried away by fancy ideas. (Analogy: shouldn't be thinking about what colour the out wall are going to be or how thick will the windows be, when we're just doing the piling for a building.) We just wanted the most solid yet malleable foundation.

If we tried to implement too many features given the limited time and scope in A3, it would most probably end up being more restricting than helpful come the final project.

Healthbot screen research

Tabletop (1.5 wide)

1.5 1

2.9 1.9

0.7 0.5

Guide

0.6 0.9

Cafero

2.5 2 1.25

2.5 1.9 1.315789473684210

0.7 0.55 1.2727272

1.4 1.2 1.166666

1 0.9 1.1111

1.1 0.8 1.375

206

What to do about pop up touch keyboard

* Set focussed to textField after submit button
* Stats column headings
* Music in quiz complete panel and start screen? "Audio rewards"
* Layout help screen with title and text area?
* List builder insert number of levels (must be number>0)
* Choose a list of words from file to add to level??? No
* Add word and level. Saves to data structure (like wordlist) then writeToNewWordlist
* To guarantee wordlist is correctly formatted for custom lists as the elderly won't want to do that themselves. And don't have the experience of what to do if application isn't functioning as expected (due to poor formatting which is beyond the scope to fully test) Formatted will probably be provided by online download or by their nurse/family member
* As simple and minimalistic
* Less advanced functionality if it makes it easier to use / less chance of "doing something wrong as breaking the application"
* Instead of separate login and register, just login (pop up) and it registers the user. Vs drop down choose login so no duplicates. But to keep the entry process as simple as possible and assuming will be aware there are 2 people with the same name.
* Highest score save username and score. Just for 1 quiz/session (#/session words) percentage to avoid making people play longer to increase score (session as quiz and review together)
* Progress bar
* File chooser choose list (NZCER formatted) and video file
* Constructor/quiz start calls create movie. Done sets flag. All correct plays this movie if flag
* If not ffmpeg for movie exists create it?? Or: power button deletes the movie (in case it was partially made)
* Choose level out of those that are unlocked in settings (must complete a lower level before going to a higher one) for<currentlevel
* Separate hidden files for each word list and user (naming)
* Settings hidden file level listLength, split(" ")
* View words in level --Don't allow except for after quiz completed
* ULearn Mac logo for buttons
* Visual rewards
* When spell wrong don't play video of wrong spelling as that reinforces it
* When spell right play video of correct spelling to visually make it more vivid in memory
* Target age group --7 to 10 years old (old people on Health Robot for GUI?)
* Own way, peer reviews, case studies (with people from target demographic)
* Phrase for word before asking them to spell??? They'll see the word in case festival didn't pronounce it properly

Gamify

* Time?
* Multiple users
* Length of list to quiz
* Speed limit set then after that time call fail
* Easy (no limit), medium, hard. Different points
* SwingWorker JLabel countdown
* Add to review button, JLabel "added to review!"
* Add to custom list?? No levels to avoid issues? But custom list has levels