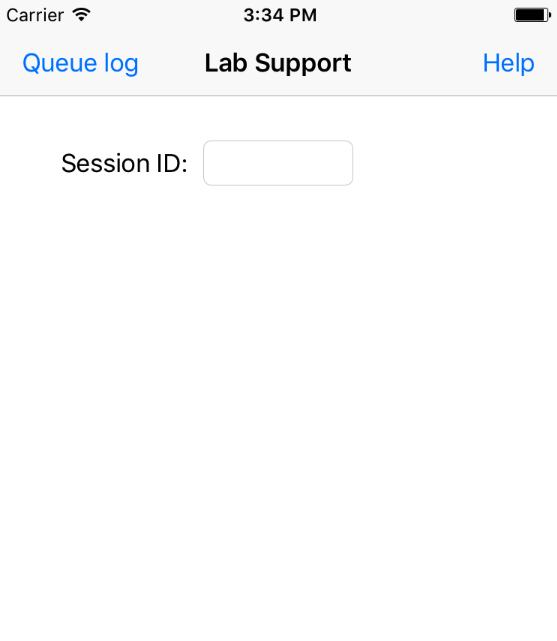
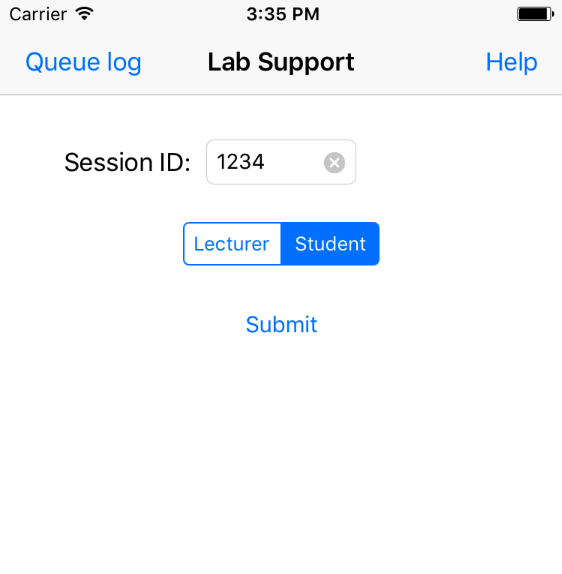
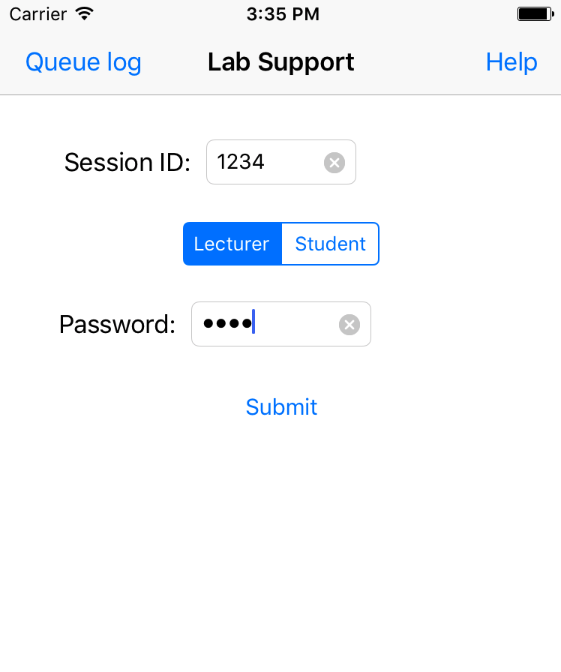
SOFT254 (iOS) – Chris Dyson (Group L)

Student Lab Support Ticketing System

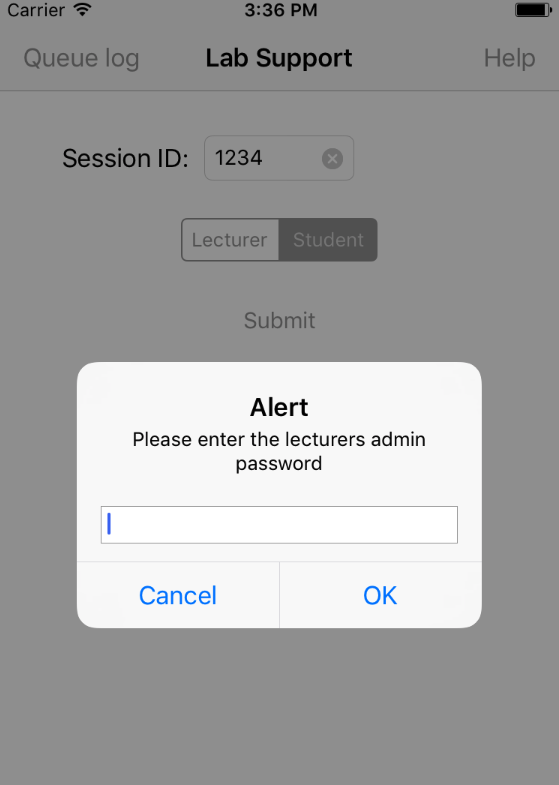
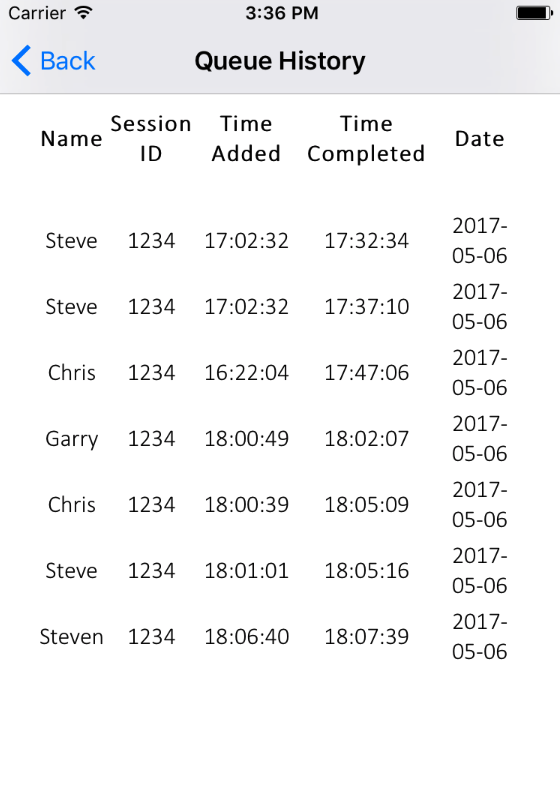
For this project, I have created a student lab support ticketing system app which runs on iPhone and iPad. There is one app which both lecturers and students use, but there is different functionality depending on the user type. 

When the app opens, you will be presented with a basic login screen where you will enter a unique 4-digit session ID code, or this could be a lecturer ID code regardless of user type. After a code has been input, you select the user type from the segmented control, either lecturer or student. If you select student, a button will appear and that will take you to the student screen.

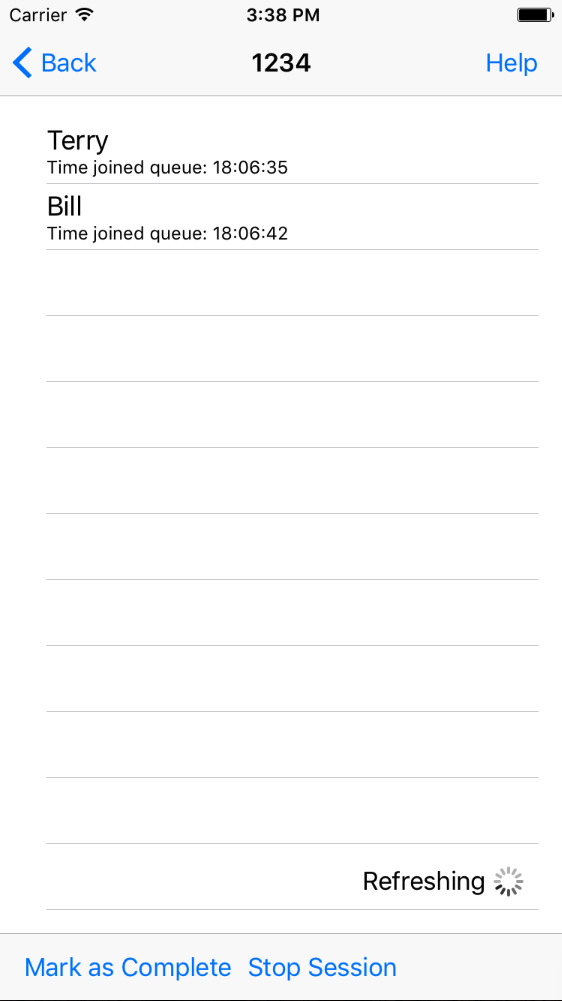


If you choose lecturer, a password textbox will appear which you need to fill in with the correct lecturer password, which is the same for all lecturers, and can be changed at any time as its stored on a web server if its compromised. When the password has been input, and if its correct, a button will show which will take you to the lecturer screen. 

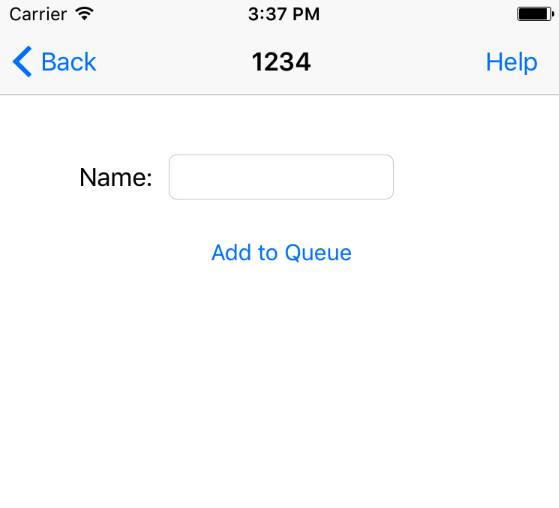
Also from the login screen, lectures can access a log of past students who have been in a lab queue. The button to access this screen is in the top navigation bar and it can be used to see waiting times and show management how long students are having to wait for help. This feature is also password protected, using the same login password when a prompt appears.

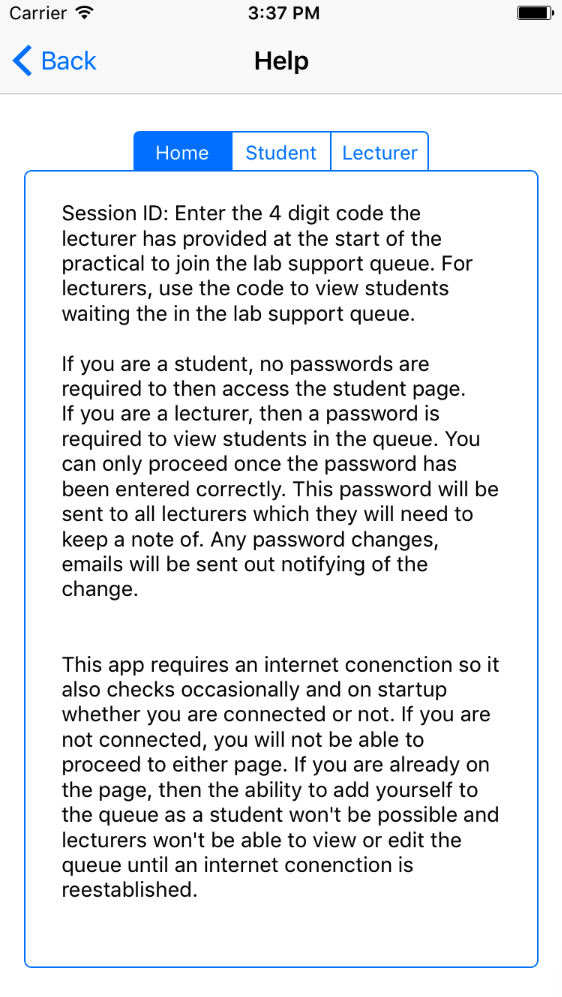
From the lecturer screen, the primary item is a tableview which shows a list of students who are currently waiting in the lab queue for the session ID entered on the homescreen. You can see the name of the student and when they joined the queue, and the students at the top of the list were added first to the queue. Once the lecturer has seen to the student, they select the student in the list, then click the button ‘Mark as Complete’. This will remove them from the list. At the end of a lab session, the lecturer can click the button ‘Stop Session’ to remove all outstanding students. The list of students is refreshed every few seconds to show the most recent queue.



From the student screen, a textbox for the students name is shown and a button to then add that student, with the inputted name, to the lab queue with the session ID.



From the home screen, lecturer screen and student screen, there is another button in the top navigation bar, ‘Help’. This takes you to another screen with a segmented control and a textview component. This is aimed at giving help to users, with details for the specific screen.



The app requires an internet connection to function, so it checks on start-up and every 10 seconds after that, to see if the device is connected to the internet. If its not connected to the internet, then an alert will show and some app functionality will be disabled.

For this project, a different ViewController class is used for each of the different screens, (home screen, student screen, lecturer screen, help screen, queue log screen). There is another class file with a function that is used to check for internet connectivity and can be called from any ViewController.

This mobile app has been designed very differently to what it would have been if it was a desktop app. Apple has many restrictions on app design to make sure there is a continuous look and feel between all apps that can be downloaded from the app store. This look is also used to help the user be able to interact and view components with ease, especially on the small screen and using a touch screen. With the mobile apps, you can’t have lots of links and huge amounts of detail like you can have on a desktop, only a link or two can be on screen at a time, which usually takes you to another screen instead of doing something with the current screen.

In this project, I worked by myself to create the mobile app. Even though I worked alone, GitHub and planner were still used as project management is very important. GitHub was used to have backups of the project and to be able to revert to previous versions when a piece of code wasn’t working correctly or stopped other parts of the app from working.

One part of code which wasn’t written by myself was the class file for checking internet connectivity, this was sourced from Isuru Nanayakkara on his GitHub. The way I handled the output from this function was code I wrote myself, including the error messages and disabling functionality. Other code from sources includes handling the JSON responses from the RESTful API I created. I tweaked the code a lot so it was tailored to the response and outcome I required.

I feel my prototype works very well. It is one app that handles both the student and lecturer side of things, with a simple login screen to get to the appropriate screens. There are security measures such as needing a password, known only to lecturers to view the lab queues and to view the lab queue history logs. It has a simple interface for the lecturer to view the queue and mark a student as complete and it refreshes automatically to show the latest queue.

If I was to do this project differently I would use Google Firebase to connect a database and scripts to the app to handle the lab queue. By doing it this way, there is much more support for having the updates pushed to the device, with push notification support. At the moment, there isn’t push support as I created a set of php files to connect to a database, all hosted on a web server. Other changes I would make is adding more options to the login screen, such as choosing the university you are studying at, so only relevant logs are shown in the history log and the same Session ID could be used at multiple places.

No unit tests were created or run for this project, but I tested any part of the app that I created straight after creating, to make sure that it is working as expected. I tested things on the desktop simulator and on physical devices such as an iPhone and iPad, of varying ages, to make sure the product works across all devices.

One feature I have noticed to not work perfectly is when a lecturer selects a student. Normally the student would be highlighted in the tableView, but if you happened to select as the data was refreshing, it could cause the app to crash. A prevention was to deselect all rows when the refresh occurred, which partially fixed this error, but still doesn’t fix it 100% of the time.