SELP Assignment Part 2 – s1136463

In the ‘doc’ folder I have included a folder containing screenshots of the application running in the emulator.

The initial activity has four options : search courses, browse courses by year, show timetable and re-download data.

Search courses opens an activity allowing the user to input text and the app will only show courses that match the input. This is to help the user find specific courses quickly. Unfortunately due to limitations the search is only applicable to the course name and does not match courses according to their course code like “DBS” for Database Systems.

Browse courses allows the user to filter by a year. This filter cross-references tables so it not only returns courses matching the year in the courses.xml file but also the timetable.xml. This showed to be a huge benefit to using an SQL database throughout the application.

Show timetable checks if the timetable is empty and only opens if there are courses that have been added to timetable.
It shows a list of all lectures chosen by the user by their course code, the day and the time period the lecture is on. Clicking any of these options opens a dialog that displays more information on where the lecture takes place.
Due to there being some mismatching and missing values in the xml files, the selection of a lecture first attempts a “strict” SQL query in the attempt to get as much information as possible and if that fails it attempts a less strict search as there is no data linking some of the rooms to the buildings. This ensures that the user gets as much data as possible and the app doesn’t crash if there is missing data.

When browsing / searching through the list of courses, selecting a course will take the user to an activity showing much more information on the chosen course, including an option to go to the course webpage so if the user wants even more information that option is there. Viewing courses in this manner is how the user adds courses to their timetable.

Initially, using SQL databases seemed excessive, as there was not that much information and constant querying could slow the app down and reduce responsiveness. However there have shown to be many benefits to building the databases even after the initial stage of getting them to work. In order to tackle speed/responsiveness I have tried to use Arraylists containing just the course name when displaying all of the courses. Hopefully this reduces the memory needed and will not cause speed or responsiveness to suffer.

There is one issues that I was unable to overcome in my project and that I was unable to use my SQL database outside of an activity class so I could not insert freshly parsed data into a database during the parse and had to return an

Arraylist which was then iterated through and inserted into the database. If I had more time I would do more research into this issue and try to improve the speed.

Also my application currently does not check if the files on the webpage has been updated so I have implemented a re-download button that force updates the information, however this is a naïve way of updating the information and I could have implemented a feature to check the hashing of a file with the previous version to ensure the data is not being downloaded if it is not necessary or even could inform the user that their data is out of date. On first run the app will download the information anyway.