**Task 1 – Squash App**

My first task in the Community Software group was to make an app to digitise the submission of match results for Squash Otago. This was an app I’d been thinking about making for a while and I figured this would be the perfect time to put what had been in my head onto an actual app.

Initially Sam, Abdel and I were working on it until Patricia decided to split us up and have us work on separate tasks, which suited me better because I was really struggling with explaining why all the different parts were needed and how they were used. In the later tasks, I got a lot better at explaining my thinking, through having more experience with working in a group and being able to use more technical terms as I became more familiar with them.

When I started the app, my only sort of plan was to translate the paper form we had to fill in on each interclub night, and put it on an app. I spent a little bit of time thinking about how the players, matches and match scores should be recorded and saved, and I eventually came up with the app’s design, with the user typing in the players’ details and the team details, then using dropdowns to assign who was playing who, and some more typing to input the scores of each match. Looking back, I wish I’d spent some more time planning how the app would work, before jumping in and being too eager to start programming.

After I got all the main functionality completed and working, I changed the app’s design and gave it a couple of menus (one menu for the data management and a main menu to view the previous tie or start a new tie) and added functionality to prevent parts of the app being accessed before it had all the data. Adding the menus and the extra functionality made the code confusing and the code was essentially the definition of spaghetti code. I’ve learnt from that experience and my code in the latest projects is more modular, commented, and a lot tidier.

The biggest problem I had while developing was getting the data saved to be able to access it later. I ended up going with shared preferences to save the data locally to the phone, which makes the app able to be used while offline, but the way I went about saving the data was extremely messy and made it hard to write code to access the data. I was passing the data around in bundles between the activities and it was complicated to keep track of them and accessing them. When it came time to test the app, I got my family who plays squash to test it for me and I used it a couple of times on interclub nights to see if it really was making life even a little bit easier. I came up with a list of tasks that had to be completed and I watched the testers do them, only helping when they were extremely stuck. Doing this proved to be a great idea because it helped me see that even though using the app was intuitive to me but it wasn’t that intuitive for anyone else. It also highlighted bugs and design flaws that I compiled into a list to fix.

At the end of the task, no one was using it on interclub nights, but I believe if I worked on it more and made the navigation cleaner and the data entry required to be less, it could definitely be used during interclub to limit the amount of paperwork and work that the interclub controller has to do. I really enjoyed making this app and if I’d had longer I would have found out if the Squash New Zealand site had an API I could use to get all the player’s details with just a search on the team name, club or even the player’s name.

**Task 2 – AYA App**

The second task was an app that Patricia had gotten screen mock-ups for from the Design School. The design school, along with the DHB, had designed an app for teenagers with a chronic illness to store all their medical information in one place on their phone. I was still doing the squash app so Sam and Abdel made a start on the AYA app in the first week without me. We’d decided together at the start for the app’s data be saved to an XML file on the phone to make data storage and retrieval easy. Sam spent the first week making an XML parser class that did everything we needed to be able to do, writing to and reading from the XML file.

We went through the screen mock-ups to decide what was feasible to program in the time we had and what seemed like would work and made sense. For example, the design had a login screen and we decided that having this seemed redundant, considering the data entry was happening on the app and only one person would have access to the app. Once we’d decided on the screens and tasks we were going to implement we wrote a list of them and split them up evenly between ourselves. I was assigned the medication page, making the layout consistent, the menu, and a couple of static pages. The menu and the static pages only took a couple hours total to make. The hardest part of each was getting the layout to be the same from the android emulator on the computer to the physical phone. The buttons and the layout that the design students had made for the menu was in a slightly complicated layout that was hard to implement. The menu was in a grid format and I spent an afternoon getting the buttons in the right places so they were as evenly spaced as possible. The Android designer (**find out the actual name of this**) can be really fidgety and frustrating to use so once I got the menu looking exactly like the mock-up, I left it alone. One of the tasks I was assigned was to make the layout files for each of the different screens. As a group, we’d decided that having one person make the layout files that are implemented across the board made it a hell of a lot easier to keep the design standard and as close to the mock ups as possible. I spent about a day putting together the different layout files, usually one file including a base file and once the files were completed I wrote up a text file with a how to so Sam and Abdel could use the files without having to ask me the same questions over and over again. It saved so much time doing this and the app ended up looking exactly like the mock ups because of it. I also spent a bit of time using Photoshop to get the different elements we needed from the PDFs of the mock ups to small images we could use as buttons and extra design features. I really enjoyed doing this because I know how to use Photoshop quite well but it got tedious when I realised I kept saving the extracted images as either the wrong size (too big for the app resulting in crashing the app because it was taking all the memory to load the image) or the wrong file type (saving as a .jpg when it should have been saved as a .png).

The other task I was given was to create a screen that had to view and add medications. This was a really fun challenge, because the medication had to be displayed in a custom dynamic list that was updated every time a medication was entered. I got a good handle on passing data from a fragment back to the activity and vice versa while doing this. I made the add medication button come up with a dialog fragment that had input fields and a check to make sure all fields had data entered. I used the same fragment for the editing of the medication, just passing the appropriate information to the fragment. the medication screen ended up being one of the cool features of the app, and Abdel ended up following the same pattern for an appointments screen.

We ended up finishing the main functionality on either the Tuesday or Wednesday before we had a meeting with a lecturer from the design school, Martin Kean to show him the app on the Friday before we handed it in. Martin was really impressed that we’d managed to get the whole app completed in 3 weeks and with the quality of it. He wanted to take it to the people at the DHB to see if it was what they were looking, but had us make a couple of tweaks, adding in a button for credits and some minor design tweaks. We made the few changes he suggested and sent him an APK for him to show to his colleagues. Martin later came back to us with the news that the DHB was really impressed with app and wanted to develop it further, and wanted to turn the further development of the app into potentially a summer internship for us. At the time this was written, we’d heard no further on whether the internship was going ahead.

I really enjoyed doing the app, it was a fun challenge to wrangle the layout files and get them working and the medication screen is one of the best things I believe I’ve programmed to date.

**Task 3 – IoT Database API**

We started our task 3 at the beginning of week 6, after blasting through the first two tasks. Patricia was scratching her head for our next task and one of the guys doing project 2 mentioned that Brendon, one of the IoT Dunedin project group members was spit balling about an idea for a database that could track what pieces of hardware they had, how many of each hardware they had and if it was deployed for a project or if someone had it, so that became our next task. We ended up making a database with a REST API and an API front end. We spent the first week talk to Campbell and Brendon in the IoT group about what they wanted to store in the database and how. From talking to them we designed the first version of the database’s ERD which is something I always enjoy doing, the challenge of creating ERDs is always a fun one. Once we were happy with the design, we had a meeting with the whole IoT group to get their opinion to check it would work how they wanted. From this meeting, we made some changes to the ERD and then began implementing the database. Sam talked to Rob and got us an account on the BIT’s SQL Server so we could put our database on it which gave us access to it when we ended up publishing the API and front end to the BIT’s iis-dev server. Once we worked out what API end points we wanted, we divided them up evenly so we all had an even amount of work to do. Making the API end points was really cool, and something I hadn’t done with C# before. Writing a couple of lines of code and getting a JSON object returning was really satisfying and reminded me why I really enjoy programming.

Getting all the API end points working well and returning what we wanted took about a week and then we started on the front end. We once again decided on what we wanted the front end to be able to do and assigned ourselves tasks which we then put on a Trello board to track who had done what and when. Putting it on a Trello board was really good, it held us accountable for what we had to do and made it easy to reassign tasks if needed. Sam ended up doing nearly all of my tasks, just because he was going through something in his personal life at the time and wanted to work rather than think about it, so he completed his tasks and started on mine and completed them in a night. This annoyed me a little, because I really wanted to work on the front end, and do my share but shit happens and you just have to let it slide sometimes. After this I went onto making the CSS for the pages consistent and look better than they were. By the end of the third week of working on it, we were done and handed it off to the IoT group. We didn’t do the handover too well, so I’m not sure if they’re using it, but the experience making it was well worth it, regardless of it’s use.

**Task 4 – Visual Scan Tools for the Oculus Rift/Otago Access Radio (OAR)**

For our fourth and final task, we initially couldn’t decide if we wanted to focus all our attention on a possible API for the Otago Access Radio or make tools from the Project 2’s Visual Scan Tool for the Oculus Rift. We ended up deciding to make the Visual Scan tools for the Oculus while waiting to hear about the OAR requirements.

Once we decided this, I emailed Leslie and Domi at OAR to set up a meeting with them to see what they wanted the API to do, and we decided on the Visual Scan tool the each of us were going to work on. I picked to do the flash one, it seemed simple and not too hard to do while also being point of contact for Leslie and Domi. I didn’t enjoy making the tools for the Oculus. I really don’t like making games, especially in a 3D world like Unity and the Oculus is. There’s too many finicky little bits that you have to get *just* right or else the whole thing crashes or doesn’t work. Even though we were technically making them as an extra for the Visual Scan Tools app, it still felt like it was pointless and we were bashing around making a game for the hell of it, and I’m all about making software for a purpose and not for the hell of it. The experience of making a game for the Oculus was cool, even though it sucked to a point. Abdel and Sam really liked doing it so there’s that.

We had a meeting with Leslie and Domi, and we went prepared with questions which were all answered and we came away with a lot of information for the database and API they wanted us to build. Once we got back to the project room I, thankfully, made a basic ERD and a bunch of notes from the information they gave us, and then put it on the back burner until we had finished the Visual Scan tool.

When we got back after the mid semester break, we were finished with the Visual Scan tool, so I started working on the database and API, while Sam and Abdel worked on making one of the tools a whole lot better so we could show it off at showcase. Because I’d made the basic ERD and the notes straight after the meeting, it was so easy to refine the ERD and make it work. I enjoyed making the database, manipulating the information to fit the requirements is something I like doing. I annoyed the hell out of everyone around me by constantly talking about the database for two days while I designed it, but the end result was really good and it worked really well when I wrote the SQL for it and put the database onto the server, so the annoying everyone was well worth it. Once the database was on the SQL server, I started making the API and getting the end points returning JSON. After a week, I had a well running API with some interesting code returning all sorts of JSON combinations. Like I said earlier, I really enjoyed making the API end points, it’s cool working out the crazy table joins and selects to get the data you want being returned as JSON objects. Domi from OAR emailed asking if we could have a meeting to catch up, see we were at, so that was set up and I spent a bit of time making up some documentation for them and publishing the API to iis-dev server so I had something substantial to show them. During the meeting, I showed them the API and then they turned around said they didn’t really want the API anymore, they wanted their original app, and asked if we could make it, as per what was going to happen initially. This was when I experienced the client frustration everyone talks about. I was pretty frustrated that they’d gone back to the original plan of an app, when I’d just spent two weeks of precious time at the end of year working on an API that might not even be used. After the meeting, I went and finished the documentation for the API that I had started and commented some of the more complicated code I had in the API and pushed it all up to GitHub, so it can be used next year and shouldn’t be too hard to pick up.