Tech Support for the Elderly

AQA A LEVEL COMPUTER SCIENCE COURSEWORK 2017

Ashwin Ahuja

1 - ANALYSIS 4
1.1 - Background to and identification of problem 4
1.1.1 - Elderly and Technology 4
1.1.2 - The Millennials 5
1.1.3 – Conclusion 6
1.2 – Description of current solutions 6
1.2.1 – Online forums 6
1.2.2 - Professional Technical Support 6
1.2.3 – Remote Technical Support 8
1.2.4 – Conclusion 8
1.3 - Identifying prospective end user(s) 8
1.3.1 – Elderly 8
1.3.2 – Teenagers 9
1.4 - Identifying user priorities (interviews) 9
1.4.1 - Gill Wilson 9
1.4.2 – Philip Fernandes 11
1.4.3 – Elderly neighbours 11
1.5 – Android vs iOS 13
1.6 – Phones vs Tablets 14
1.7 – Payment of helpers 14
1.8 – Acceptable limitations 14
1.9 – SMS System 15
1.10 - Objectives of the proposed system 15
1.10.1 - Elderly System 15
1.10.2 - Teenagers' System 16
1.10.3 - General 16
1.11 – Explanations for objectives 16
1.12 - Safeguarding 18
2 – Design 19
2.0 - Overview of entire system 19
2.1 - Introduction 19
2.2 - Data 20
2.2.1 - Storing Data 20
2.2.2 - Communications with applications 29
2.3 – WebAPI (Appendix A – Web API Code) 30
2.3.1 - General Structure of .NET Core API 30
2.3.2 - Communicating with JSON 31
2.3.3 - Communicating with the SMS Server (SmsStuff.cs) 31
2.3.4 - Functions 31
2.3.5 - Securing data 41
2.3.6 - Verifying data entry 41
2.4 - Android App 42
2.4.1 - Important Features 42

2.4.2 - Helpers *43*
2.4.3 - Elderly *55*
2.5 – SMS Server System 73
2.6 – Server Side Management 74
2.6.1 – Traffic Management *74*
2.6.2 – Crontabs *74*
2.7 – Testing Strategy 75
3 – Implementation 78
4 – Testing 79
4.1 – Test Series 1 79
4.2 – Test Series 2 84
4.3 – Test Series 3 85
4.4 – Test Series 4 90
4.12 and 4.17 ..... *93*
5 – Evaluation 94
5.1 – User Feedback 98
5.1.1 – Mrs Wilson (AgeUK) *98*
5.1.2 – Phil Fernandes *99*
5.2 – Analysis 99
5.2.1 – Learnability *99*
5.2.2 – Latency *99*
5.2.3 – Improvements *100*
5.2.4 – Extensions *100*
Appendices 102
Appendix A – Web API Code 102
Backend *102*
Types *120*
Controllers *134*
Main Files *170*
Appendix B – Android App Code 174
Activities *174*
Resources *244*
Backend *265*
Types *267*
Root Folder *282*
Appendix C – SMS Server Code 287
Appendix D – Test Data and Expected Results 290
Test Series 1 ..... *290*
Test Series 2 ..... *446*
Test Series 3 ..... *449*
Test Series 4 ..... *499*
Appendix E – Works Cited 537

1 - Analysis

1.1 - Background to and identification of problem

1.1.1 - Elderly and Technology

The world is getting more connected by the internet. For the first time, over 50% of people[1] in the world (and over 80% of the western world) are connected by a single network. Ever since the CEO of IBM stated that the market for personal computers is 'around 5'[2], people have attempted to prove him wrong. Fortune magazine estimated that over 250 million personal computers were sold in 2016[3]. Alongside the rise of personal computers, mobiles and tablets have also come into the picture, reaching an even greater group of people. And yet, this increased connectivity has somehow bypassed a section of the population. Elderly people today are termed the 'generation that tech forgot'[4], living through a period of great technological advance, but left feeling ostracized and isolated by it.

Figure 1: Internet use by age group, 2000 – 2012

Source: Pew Research Center [5]

The graph in Figure 1 captures the stark difference between the number of people between the ages of 50 and 64 using the internet as opposed to the number above the age of 65. Only half the population over 65 use the internet. This is unfortunate considering the benefits that internet can provide to the elderly, from using Video Messaging services such as Skype to connecting with friends and relatives to using online diagnostic systems. This is often not for lack of trying. Over 70% of the elderly, according to a recent poll[6], want to use technology and the internet. Talking to a group of elderly neighbours, one of them told me that 'most of my friends and I do want to learn how to use technology but it is

[1] (Internet Live Stats, 2017)
[2] (PCWorld, 2008)
[3] (Fortune, 2017)
[4] (BBC, 2015)
[5] (Madden, 2012)
[6] (Smith, 2014)

very difficult'. There are two main problems that they face, the first is learning how to use it and the second is in obtaining ongoing technical support.

For the first problem, training sessions targeted for the elderly are available, with many organisations teaching the elderly how to use the internet and organising classes in public libraries and universities. According to a recent survey, over 30% of those over the age of 70 are in some form of a technology class[7]. These are reasonably successful in achieving their objective as a sizeable majority of people appear to appreciate them. At a recent AgeUK session that I attended, members mentioned that in a few months they had gone from knowing nothing to being able to communicate well with their family and friends using online tools.

The second issue of providing ongoing support appears harder to solve. Online forums which can answer questions and solve problems which the elderly are facing could provide one solution. However, while these work well for the relatively younger more computer literate population, the elderly need more personalised technical support. Technical support companies who may have the potential to offer this support are expensive, impersonal and at times, not very reliable.

Meanwhile, there is an untapped resource of young people who have the requisite skills to help, and would in all likelihood, be willing to help. Young people have the instinct to understand how computers can break, from checking that the power wire is plugged in, to going to command line and debugging the wireless adapter of the computer. By connecting the elderly to young people, these instincts can be harnessed to help solve technical problems of the elderly.

1.1.2 - The Millennials

Technology is engrained in the youth of today, as we rely upon Snapchat and Facebook Messenger to connect with friends, and Wikipedia and the online Encyclopaedia Britannica to write essays for school. The millennials are without exception highly technologically savvy. A paper by Susan Herring of Indiana University talks of the 'technological divide'[8], wherein the majority of children, by the age of ten, are more technologically savvy than their parents.

Yet, there is nothing that is harnessing this skill of the youth. Taking an informal survey among my friends and peer-group, a vast majority indicated that they had been approached by elderly neighbours asking them to help out with fixing a broken computer or tablet. Moreover, these technical problems are a regular occurrence. These could clearly be easily fixed with better access to a larger group of willing and enthusiastic young people.

Meanwhile, it is clear that young people would largely be interested in helping out the elderly, with positive responses to the idea when asked as part of my survey. And while this was a reasonably small sample size for a survey, it is clear that young people today are more interested in helping out than the previous generation. Percentages of young people involved in volunteering increases year on year and participation in social activism (indicating a caring about others and society in general) is also on the rise.

[7] (AgeUK, n.d.)
[8] (Herring, 2008)

1.1.3 – Conclusion

There is a genuine challenge for the elderly, to learn how to use technology, and to find good technical support once they start using it. Existing support options are inadequate, expensive and ineffective. Meanwhile, there exists a group of people, the teenagers, who have the requisite skills to help, and are eager to do so. By attempting to connect the two groups of people, one can solve the problem of providing technical support for the elderly, simultaneously getting the youth more involved in society.

1.2 – Description of current solutions

1.2.1 – Online forums

One of the primary methods for solving any technological problems, for the elderly and for the rest of society alike, is through the use of online forums. Online forums enable people to ask questions and others to respond to them. They allow people to rank responses based on their usefulness and thereby build up a knowledge base of relevant information that people can tap into. In order to benefit from online forums, the elderly need to have access to an internet connection. Additionally, they need a certain competency with computers and the internet to be able to effectively search for and locate solutions to their problems.

Even after overcoming the initial hurdles, the elderly face additional problems. The forums can largely be split into two types – professional and amateur. In professional forums, the authors of answers are largely professionals and these sites are often run as a side service for companies which deal mostly with other issues – including professional technical support companies. One example of a 'professional' technical support forum is techsupportforum.com[9] – a site where those giving answers are 'IT professionals'. Yet, these sites are often not comprehensive, only answering a small subset of the possible problems that could occur. In fact, even looking up a basic router not working on techsupportforum yielded a single answer, suggesting that the software setup was incorrect. They failed to suggest simple, and often more likely, problems such as the router not being connected to power or to the telephone connection. Moreover, despite the best efforts of the authors the answers to problems were often complex and hard to follow for those without adequate knowledge of technology. This problem is even more prevalent in the second form of online forums, which are largely community run, with answers being provided by people around the world with varying degrees of knowledge. Examples of these sites include r/techsupport[10] (Reddit), StackExchange[11] and even Yahoo Answers[12]. Though they largely cover every possible problem that one might have, they generally fail to provide answers which are easy to understand for the elderly. Instead the answers are generally intended for people with a certain level of technological ability, certainly not elderly people with little experience of using the internet.

1.2.2 - Professional Technical Support

Another possible avenue to get technical support is through professional services offered by many companies both local and global. These offer a much more personal experience than online forums. The person visiting the elderly person can offer a tailored service, ensuring that the problem gets fixed, as well as educating the elderly person for the next time. Personal contact provides an opportuinity

[9] (Tech Support Forum, 2017)
[10] (Reddit, 2017)
[11] (StackExchange, 2017)
[12] (Yahoo, 2017)

for the elderly person to ask all relevant questions to ensure that they understand how to fix similar issues in the future. However, there are a number of drawbacks with this option too. Firstly, since the tech-support is run like a business, people are incentivised to not showing the elderly how they fixed the problem, instead ensuring that they get repeat business if the problem reappears. In fact, a neighbour who had used a Barnes based small tech support company complained that they charged 'ridiculous' amounts without actually fixing the problem. Even for one problem that they did manage to fix, my neighbour claimed that they declined to explain the problem, instead claiming it was too complex and unworthy of explanation.

The issue of cost is an important issue with professional tech support, as they charge fees for services which are even trivial and not too complex. I gathered information based on call-out fees and hourly rates for ten companies in the UK.

Company Name	**Location**	**Call Out Fee**	**Hourly Rate**
HTL London[13]	Central London	£100	£50
IT Guy[14]	West London	£75	£50
Andrew Tsai Technical Support[15]	North London	£25	£70
Mitchell Wayne Technologies[16]	Birmingham	£30	£50
D-Tech-IT[17]	Liverpool	£0	£50
EverythingTech[18]	Manchester	£60	£60
Grant McGregor[19]	Edinburgh	£50	£65
AVERAGE		**£49**	**£56**

Figure 2: Survey of cost of professional technical support around the country

Meanwhile, in order to show the fact that the youth could help to solve this problem, I conducted an informal poll of ten teenagers who said that they would be interested in helping the elderly in my area, and they all suggested no call out fee, with an average hourly fee of ten to twenty pounds.

Finally, it is important to state that it is also often challenging to find people in your area as the vast majority of technically qualified people attempt to join jobs in industry and banks as opposed to

[13] (HTL London, n.d.)
[14] (IT Guy, 2017)
[15] (Tsai, 2017)
[16] (Wayne, 2017)
[17] (D-Tech IT, 2017)
[18] (EverythingTech, n.d.)
[19] (McGregor, n.d.)

offering technical support. Even finding eight companies across the country which offer technical support offered a certain challenge.

1.2.3 – Remote Technical Support

While remote technical support can be much cheaper than professional technical support, with no call-out fees and lower hourly fees, there are a number of issues with this system. Firstly, this would be unable to solve some issues, especially those which are associated with hardware rather than with software. Furthermore, people are often unable to show how they are solving the problem, as they have limited English skills and even if they are able to talk in English, they have little training or experience talking to the elderly, and so are ineffective at communicating the most salient points, instead puzzling their clients.

Another issue with remote technical support are the problems associated with scammers, who instead of genuinely solving software problems on a computer can instead use the opportunity to install Trojans and other viruses. This would then allow them to have complete access whenever they want, gaining access to everything from people's accounts as they log onto them on the computer to their webcam. This has led to a number of well-documented cases of identity fraud as the scammers gain enough information to imitate their victims, taking all their money.

1.2.4 – Conclusion

Though each of the three current methods of technical support for the elderly have flaws, it is clear that from an ideological perspective the second, professional in-person tech support is by far the best, as human contact allows the elderly to both learn from the problem solving as well as have the problems fixed most effectively. Hence, it was important to ensure that I tried to implement that method or a variation thereof and also attempted to fix the problems associated with cost and lack of availability of tech support people.

1.3 - Identifying prospective end user(s)

There are two main groups of end-users who would both determine and provide a sounding board to test if the objectives of the system were being met. It was important to consider each in turn.

1.3.1 – Elderly

For the selection of the end user to represent the elderly, it was important that the user would use technology as well as either be elderly themselves or else understand the needs and constraints of the elderly. In the end, I considered the two main possibilities, either a group of neighbours who I had previously interacted with about their problems or an elderly technology expert. By involving a group of elderly neighbours, I would achieve a greater range of opinions than from one person. However, as people from the same street, attending the same lessons and of a similar age, I realised that their perspective was largely the same. It was therefore important to seek out a person who had broader experience of the problem who could represent the entire elderly community. I therefore contacted a number of charities which give technology lessons to the elderly. I attempted to reach a person who had experience in teaching the elderly and would be aware of the various problems that affected them. In the end, Mrs Gill Wilson, the IT Project Manager of AgeUK Richmond agreed to be the main end-user representing the elderly end-users. Her involvement helped test that the user interface of the end-product (solution) was simple for a technologically challenged elderly person, while ensuring that it solved the problem of tech support for the elderly.

1.3.2 – Teenagers

The other primary user group of the system are the teenagers. It was important that the sytem worked well for them, as otherwise few teenagers would signup, rendering the system useless. In order to gain a teenager's perspective, I interviewed Philip Fernandes, a technologically savvy eighteen year old at St Paul's School who would be interested in using my system to help those around him.

1.4 - Identifying user priorities (interviews)

1.4.1 - Gill Wilson

What are some common problems that the elderly face when using technology?

A number of the most common problems are problems which would otherwise be obvious to those who had grown up with technology, for example failing to plug in one wire or not double clicking on a shortcut to open the application.

How do most of the elderly currently get tech support?

While we encourage the people we teach to attempt to find solutions to their problems online, we find that a number of them are forced to seek professional tech support as the online forums are difficult or impossible to understand or they require a certain amount of basic knowledge to access sucessfully.

How effective is the professional tech support?

From talking to some of the people I have taught about their experiences in using tech support that they have got from flyers and posters they see around, they have had very mixed experiences, with some finding that they are very good. Some talk about how they are very attentive, help them solve their problem while teaching them how the problem arose and how to detect and fix it in the future. However, others complained of particularly poor tech support, where they were charged exorbitant amounts and did not even fix the problem that they were asked to fix.

However, a general problem that I find throughout talking to people is the relative difficulty of getting tech support. Unless they receive a flyer, they are forced to ask around and attempt to find people who can help. In fact, they often are forced to rely upon asking family or neighbours who are often busy or unable to help.

Would the elderly be comfortable getting tech support from teenagers?

I think in many ways the elderly would be more comfortable with teenagers than others, as they often can be the nicest people. I've found in talking to teenage volunteers, that they are among the most enthusiastic people wanting to help others. Especially when the amount of money that stand to gain is minimal (or nothing), only those who want to help will actually do so. I think they would be very effective; they would probably be able to fix the problems as well as a professional, since the problems are generally fairly simple.

I think though, that there is also a certain comfort in professionals, where there are specific references and reviews that they have if the person has been recommended by a friend. If you can emulate that with your system, possibly by getting customers to review the teenager who helps them, I think you could reduce this problem.

What is the best way for the elderly to interact with a technological system? (Mobile app, website, phone call, SMS)

There's a growing trend that people aren't really bothering to learn how to use desktop or laptop computers, instead they are trying to learn how to use a smartphone or tablet. I think that you therefore need to have some mobile interface. Maybe it should just be a website which could be accessed from all devices, but I'd actually recommend mobile apps. They appear to be the easiest for the elderly to use – they have been finding them easier than anything else I've seen.

But, I'd also be aware that if people are having technological issues they may not be being able to use their phones or tablets. It might be worthwhile having another system. Maybe they could send an SMS to a phone number and then automatically a person would be set up to be sent to their house. I think SMS is probably the most reliable thing that most elderly people are relatively happy being able to use, so if you can tap into this, it would be very useful.

For the mobile apps, iOS or Android?

Ideally, both. However, if you had to pick one, I'd go with Android. From talking to people around the area, while lots of people have iPhones, more have Android phones. This is becoming especially more significant as there is a boom in cheap Android smarthphones, and elderly people are purchasing these devices.

Another advantage of Android would be the immediate compatability with Android tablets. Contrary to what I would expect, most elderly use android tablets. Things like the Hudl and other cheap Android tablets seem to be especially popular with the elderly and they have been really taking to them. That being said, there are a large number of people with iOS devices, so if you could, I would suggest attempting to make a system which worked with both.

What are the major points that you would suggest for creating a user interface for the system?

I think the most important thing by a long way is to keep it simple. Think, what do they really need. I mean all they actually need is a list of people, how far away they are and their reviews. Minimise how many screens they need to deal with and simplify each screen. One piece of advice I once received and have always thought valuable is to always ensure that there are only two paths from any given screen. This ensures it is truly simple enough for anyone to be able to use the system.

Take it for granted that they don't necessarily have the instinct of just trying things so try and make each screen very clear. What should they do to get from one screen to the other?

With regards to payment, how do you think that the elderly would be most comfortable?

I know that the teenagers would probably be most comfortable with payment over the internet and going directly to a bank account or something like that, I'd suggest that this would probably not work for the elderly who would anyway be unhappy buying things online, let alone paying some random website.

I'd therefore stick with cash payment, with them paying the teenagers directly. With regards to how much they should get paid, I think the main thing is making it as clear and transparent as possible. I

don't really think the amounts particularly matter, as long as it is cheaper than the competition, though this is not really challenging. I might suggest something like 15-20 pounds per hour.

1.4.2 – Philip Fernandes

Have you got any experience helping the elderly in your area with technological issues?

I have a little experience of helping the elderly, mostly just helping my neighbours when their internet wasn't working. However, I love the fact that your system can help me do it more often – I find that idea very valuable and powerful. I also think that you could potentially have a large number of people in the teenage group who would be willing to help out, especially if they get paid for their work.

What are the most important aspects of the system for you?

I think it's basically a system to manage all the people that are requesting help and being able to message them and arrange a time when to work. It might actually be helpful to have a calendar, booking like interface for each person so the elderly person could see when I might be free and request to get support at that point – even if it is that evening. Therefore, being able to have the ability to input when I would be free would be important – as well as other stuff like how far I would be willing to travel.

Would you prefer a mobile app or a website?

I think I would prefer the freedom of having both options available – a website would be useful so I could check the status of things even if I don't have my phone. However, I think a mobile app would be very important. Certain things like messaging people are just more natural on a phone. Also, I carry my phone everywhere, so being able to check on jobs from my phone would streamline my life.

iOS or Android?

Though I think having both an android and iOS app would be useful, I'd say the Android app is the most important. Just from my observations, I've found that not too many techie people have iPhones in my age group, perhaps because of the cost.

1.4.3 – Elderly neighbours

Another important input was obtained by talking to a number of elderly neighbours. As compared to Mrs Wilson, they were more likely to actually use the product, therefore would be able to assess it as a client of the eventual service.

When talking to my elderly neighours, I largely focused on a few main parts: their use of technology, their problems (and any possible solutions they had attempted) and the possible solution to the problem of tech support.

Their use of technology

The answer to the question of how much technology they used varied from just utilised a mobile phone to comfortably accomplishing complex tasks like video editing on a laptop. The interesting commonality was that they were most comfortable with mobile devices, with the majority saying they were most comfortable with their phone, followed by tablets. Majority of the people I talked to owned

tablets, whether they be cheap Tesco Hudls or Apple iPads. However, these were in addition to smartphones and not as a replacement for smartphones.

I also talked in depth about how they had learnt how to use technology. Most of them had learned from family members, and sometimes supplemented by classes run by charities such as AgeUK. They all felt that they were reluctant in approaching their family and would have preferred to use other means, but often found group classes (the type of class mostly offered by charities) to be highly stressful environments which were not condusive to learning.

Their experience of technical support

Only one person had any experience of getting technical support from a source other than family, having found someone on the internet. She however found the experience 'terrible', saying that she had paid an exorbitant amount for someone to come and flick her router on and off. She felt she had been robbed and though the person who came was kind and courteous, she would not use the same service again.

Everyone else said that though they hadn't sought professional help, the issues with their technology not working were common, and in fact caused an enormous amount of anxiety. One particularly hyperbolic neighbour said that she was in tears after having spent an entire week trying to figure out how to transfer data from an old phone to a new one. This was an aspect of the problem that I had not considered before – just how much havoc a technological problem can play on the mind of the elderly. Tackling the issues becomes even more important, both for fixing the problem and in order to help the mental state of the elderly.

The possible solution

Everyone I talked to seemed very enthusiastic about the possibility of an application which would put them in touch with teenagers who would help solve the problem of technical support. While one might have thought this was simply because they were cheaper, they also loved the idea of being able to talk to young people in their area, and said they would definitely prefer to use a service which used teenagers over one with adults (even if it were the same cost).

I was surprised that most of my neighbours were perfectly comfortable with the idea that the system would be run over the internet, as they were comfortable with the idea of using their phones. They mentioned that even if they would struggle to book the initial appointment because of their phone not working, it would be possible for the initial appointment to be booked by a friend who could then direct the teenager to help their friend. They also loved the idea of an SMS service which could be used instead of the app. One person made the comment that while she probably wouldn't use it as she would be more comfortable with the application, she could see where others would use it.

I also asked my neighbours whether they had any particular advice for how to design my application. The overwhelming comment was that it should be simple. One person commented that she found most applications tried to do far too much, making it just too hard to do something simple. She advised that I attempt to remove as much unnecessary content from the application as possible, instead focusing on the core function – allowing elderly people get technical support from teenagers in their area.

1.5 – Android vs iOS

A primary decision in creating the system is considering whether to develop the application specifically for iOS or for Android. Though it could be possible in the future – as the product is rolled out and advertised more specifically, that an iOS application can be created, for now, it was decided, in keeping with the discussion with Mrs Wilson to concentrate on an Android application for the elderly.

For the teenagers, the decision was similarly made to focus on Android, after talking to Phil and learning that most of the teenage clients would have Android devices. This was reinforced by research which shows that majority of children aged below 24 have Android devices[20].

It was also considered whether a web interface was required, to ensure that everyone, whether they own iOS, Android or no mobile device, could access. However, it was decided that this is largely unnecessary as the elderly people would find the internet harder to use than an app on their phone. In fact, Mrs Wilson mentioned that most people are opting for phones or tablets, rather than desktops and laptops as their device of choice to plunge into the digitally connected world.

With the decision made to target Android, it was also important to consider the compatibility of the application. As Mrs Wilson mentioned that many of the devices that the elderly are using tend to be cheap, they are likely to use outdated versions of Android software. Therefore, it is important to ensure that the application can work with old versions of software.

Figure 3: Split of Android Versions

Version	Codename	API	Distribution
2.2	Froyo	8	0.1%
2.3.3 - 2.3.7	Gingerbread	10	2.6%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	2.3%
4.1.x	Jelly Bean	16	8.1%
4.2.x		17	11.0%
4.3		18	3.2%
4.4	KitKat	19	34.3%
5.0	Lollipop	21	16.9%
5.1		22	19.2%
6.0	Marshmallow	23	2.3%

Data collected during a 7-day period ending on March 7, 2016.
Any versions with less than 0.1% distribution are not shown.

Source: The Official Ionic Blog[21]

However, it was also important to consider that newer APIs introduce many important features which would make developing the app simpler. For example, only from Android Ice Cream Sandwich onwards (4.0+) is there inbuilt software to help with REST API requests, which is likely to be an important

[20] (Keating, 2014)
[21] (Max, 2016)

feature of the application. Since over 97% of Android devices would work on an application designed for Android 4.0+, this would be a good version to set as a basic minimum requirement.

1.6 – Phones vs Tablets

Another important decision to be made was whether to design the applications specifically for mobiles or tablets. Though Android would automatically make the application compatible with all screen sizes, if the application were to be specifically designed for tablets, the UI should be designed for this, taking advantage of the larger screen size.

Since, the biggest consideration in designing the UI for the elderly is that it be as simple as possible, containing as little information per screen as necessary, this decision of utilising screen space became less important. Since a greater volume of Android phones exist than Android tablets, it would be better to produce an application specifically for Android phones, with the possibility of producing a tablet specific application as a follow-up. Specifically, the UI would be designed to be a best-fit for a 4.7" device, the most common screen size of phones sold today (and incidentally the screen size of my phone, therefore ensuring it was easy to test).

1.7 – Payment of helpers

Another important aspect to be considered was how the teenagers would be paid for their services. Though Phil mentioned how he would be happier if the payment were to be done through the apps, Mrs Wilson was very clear that this would be less than acceptable for the elderly. Though it might be possible for a system in the future where a local conduit is paid in cash by the elderly and then the helper would get paid by the app, this adds unnecessary complexity and was descoped for now. Throughout the process, the most important point was to ensure that the system is as simple for the elderly person as possible. Therefore, the payment should happen by cash, directly from the elderly person to the helper.

As per Mrs Wilson's advice, the system should also have a £20 per hour standard rate. Since the people would primarily be coming from nearby, they do not need a callout fee and could simply be paid from an hourly rate.

1.8 – Acceptable limitations

One important aspect which was ignored for the purpose of the project was the fact that each teenager might have special areas of knowledge, and might not be equally competent on all topics. This was ignored because the project was mostly targeting the trivial problems faced by the elderly which would often not require specialist knowledge. Additionally, reviews could easily cover the specific task that the person did, therefore enabling the elderly people to select appropriately competent people for the task.

Additionally, the fact that the helper probably has a maximum distance that they would be happy to travel has been omitted for simplicity. Instead, the system will allow a helper to cancel the appointment for any reason, including if they are unwilling to make the long journey. However, the elderly person should be able to see the location of the helper, which would both add a sense of trust and also so they can see how far away the helpers are.

1.9 – SMS System

The use of SMS, a possible idea mentioned by Mrs Wilson, would allow an entirely separate path by which the elderly could create an appointment with a helper, simply sending a text with information which would allow the system to create an appointment. This would allow elderly people who may not otherwise be able to use the application, in case they are unable to get their devices working for example, to get assistance. SMS is much simpler to use and therefore is likely to be understood by majority of the elderly people.

To enable this option, the amount of data required to be sent by the elderly person needs to be considered. There were a few options, from defining the date, time and helper, to simply sending a text with any content and then the next available appointment would be created. While the former requires too much information, and the elderly person might already know who they want; the latter might be too simple, as the elderly person might not be free at that exact time.

Therefore, an intermediate option was finalised, wherein the SMS system would work off a date and time. If no date was defined, it would default to today's date, assuming that the elderly person wanted an appointment as soon as possible.

1.10 - Objectives of the proposed system

1.10.1 - Elderly System

1. **Input**
    a. There must be an Android app, which works with all Android phones with all OS versions upward of Android Ice Cream Sandwich (4.0).
    b. The elderly person should be able to create an appointment with an SMS message to a specified phone number.
    c. The SMS should only require the user to send a Date and Time and the information about the appointment should be communicated directly back to the user over SMS.
    d. To log onto the Android application, the system should remember the password of the user once they have logged in once, thereby avoiding the hassle of having to remember passwords and dealing with forgotten passwords.
2. **User Interface (on all systems)**
    a. The screens should be as simple as possible, increasing the number of screens instead of complexity of each screen.
    b. There should be a maximum of two possible (very different) paths to follow from every screen, with the difference in what each button does being marked clearly.
    c. All text should be large, above 12pt, to ensure that the elderly person can see easily.
    d. There should be a screen where the user should be able to scroll through the nearby teenagers who can provide tech support.
    e. This screen should be organised by how near the helper is, therefore meaning that the helper is unlikely to have to travel from far.
    f. For each teenager, the elderly person should be able to see their average rating (if they have been rated), location and all of their reviews.
    g. The user should be able to look at the various appointments a teenager has free, allowing them to choose a time which would suit them.

h. They should be able to book an appointment for any available time at least within the next two weeks.
i. The user should be able to see the list of upcoming and past appointments, seeing the details about the appointment.
j. For appointments in the past, the user should be able to rate and review the helper.
k. For appointments in the future, the user should be able to cancel the appointment.

1.10.2 - Teenagers' System

3. **Input**
    a. There must be an Android app through which the person should be able to interact with the system.
    b. There should be a system to ensure to remember the password for the android application, so that they would not have to remember the password every time they log into the application.
4. **User interface**
    a. The user must be able to see everyone who has requested their help.
    b. They should be able to cancel any appointments which they are no longer able to be keep.
    c. The user should be able to define when they are free for at least the next six weeks, allowing them to set the times that they would be available for appointments.

1.10.3 - General

5. The appointments should be for two hours as a standard for all appointments.
6. The payment should be defined as £20 per hour for the appointments, therefore being £20 if the appointment takes up less than half the appointment slot, or £40 otherwise.
7. The system should be secure, with passwords hashed to ensure that they are not easy to view in case the server is hacked.

1.11 – Explanations for objectives

1. **Input**
    a. In talking to Mrs Wilson, she deemed it important that the system be at least available on both phones and tablets, especially on Android based ones, since this is what majority of the people she worked with used. She stated that this was by far more important than having a web version of the application.
    b. She also said that having a system which would receive SMS messages would be highly useful as most elderly people would be able to send SMS messages. Though it would be possible to insist upon a certain format for the messages to be sent in, it would make for a highly clunky interface. It would be optimal if the system could simply interpret whatever the user was attempting to say. It would also be very important that the system be as simple as possible, for example removing the ability for the elderly person to choose a helper, instead selecting the person closest to them.
    c. As discussed above, this system offers the optimal compromise between allowing the elderly person flexibility, while still keeping it as simple as possible.
    d. One issue faced by elderly people is dealing with passwords, especially storing (or remembering) them. Therefore, by ensuring that the system does not require them to remember their passwords to the system, this reduces the problems. This also means that there is no requirement for a 'Forgot my password' functionality. Though this means that

anyone who is able to unlock the elderly person's phone would be able to book an appointment, it was decided that this would be a worthwhile compromise.

2. **User Interface (on all systems)**
    a. Mrs Wilson was clear that the simplicity per screen was far more important than minimising the number of screens.
    b. She advised two paths from each screen as a maximum, as a guideline to help to reduce the complexity as much as possible.
    c. According to the United Kingdom government, all printed (and digital) materials should use size 12 as a minimum font size[22]. Therefore, this would a worthwhile limit for this application as well.
    d. The ability to view a list of people who would be willing to help is important, so an elderly person can see the helper in more detail including where they are and seeing their reviews and ratings.
    e. Since the maximum distance that a helper is willing to travel has been omitted, it is important to ensure that the the elderly person selects the nearest available suitable person. The likelihood of this has been increased by ordering the list based on proximity.
    f. By seeing the ratings and reviews of a teenager, the elderly person can make sure that the person is reliable and would be able to fix the problem. Additionally, the review may include the nature of the problems they have fixed in the past, indicating whether the teenager has existing knowledge specific to the problem area. By being able to see their location, the elderly person can ensure the helper does not have to travel from very far.
    g. Especially for cases when the elderly person have a reasonably free day, the ability to simply request the teenager's services is a useful one. However, it would also be important for an elderly person to get an appointment to work with their schedule.
    h. Two weeks ensures that the elderly person can book appointments well in advance.
    i. It ensures that the elderly person can check all the appointments they made in the past, allowing them to ensure they are around for the next appointment or to cancel it.
    j. Mrs Wilson described how this was important as the people who are helping out are teenagers and not professional, so a number of them may not be very good and a few bad people, if not obvious to the elderly when booking an appointment may ruin the system.
    k. If the problem has been solved or for any other reason the appointment might no longer be needed. Therefore the ability to cancel the appointment is an important one.
3. **Input**
    a. Philip and my own research indicated that the majority of technological people use Android phones so iOS can be omitted.
    b. Though less important for the helpers, who would be technologically savvy, this would be a welcome addition.
4. **User interface**
    a. This is relatively self-explanatory as the user must be able to see who has requested their services.
    b. Especially important for the helpers, this ensures that they can cancel an appointment if they are no longer available or if they have been requested by an elderly person who is outside of the radius which the teenager would be willing to travel within.

[22] (Department for Work and Pensions, 2014)

   c. The teenager may have an often-changing timetable of when they are free therefore it would be important that they can easily adjust this timetable.
5. This length was decided in conjunction with my elderly neighbours, who said this was the maximum time which they would require. This fixed length of appointment also simplifies the system, ensuring the person can only book appointments at set (even hour) times.
6. This was decided in section 1.7.
7. Though it is highly advisable that passwords should not be reused, it is likely that not everyone would follow this advice. Therefore, it would be important that passwords are stored securely, hashed in the database and never transmitted unencrypted over a network, in order to make it harder for them to be stolen.

1.12 - Safeguarding

Throughout the conversation with Mrs Wilson and a number of other elderly people, there was another issue which was often raised. This was the issue of safeguarding and insurance, in case there is damage which is caused by the helpers, or if the helpers are assaulted or encounter other issues. Mrs Wilson talked about when AgeUK sent adults to help others (for plumbing and suchlike), they had to be insured to ensure that if there were any problems AgeUK were not liable for the costs for fixing anything.

She also suggested that things would be even more complex for my idea, as I would be sending untrained children. This could require everything from DBS (and other criminal) checks to more costly insurance.

With more research, the legality of my system seems to be hard to define and the system may require anything from large scale insurance to simply adding a statement when a helper registers that I would have no liability in whatever happens at the meeting. However, this issue would require further investigation and is being ignored for the purposes of this coursework project.

2 – Design

2.0 - Overview of entire system

Figure 4: Diagram of Entire System Design

Source: Designed using draw.io[23]

2.1 - Introduction

In designing the solution to the problem, much attention was paid to centralizing the system as much as possible, both for reliability and security and to ensure that minimal amounts of coding as possible would be necessary on the applications side. This ensures that the applications will run fast, as well as easier to produce. Hence, the solution relies upon a server which will respond to requests to return information which can then be interpreted by the application.

[23] (Draw.io, 2017)

While creating the user interface, I spent more time with my elderly end user, considering how I could specifically design the user interface to be as easy to use as possible, discussing everything from the sizes of button to the colours which would be the most obvious.

2.2 - Data

2.2.1 - Storing Data

Since all the data had to be accessible from the various applications on all devices, the data had to be stored centrally. Therefore, I decided to use a centralised server which could act as a hub for the data.

While I considered some databases such as MariaDB, I decided that the MySQL packages by Oracle provided the most secure features, while having acceptable documentation for the syntax of SQL commands. Additionally, as compared to Microsoft SQL Server, it could easily run on an Ubuntu 14.04 server that I already had access to, as it was already being used as an Apache Web Server.

In considering what data would be required to be stored, I went through the specification that I created and considered where the data would be needed to be stored. Starting from a general number of tables, I began to draw out the connections between the tables of data, ensuring that I got the database into third normal form, to ensure that the data would be stored most efficiently.

ER Diagram

Figure 5: Entity Relationship Diagram

Source: Produced using draw.io[24]

Figure 6: Key for Entity Relationship Diagram

Source: [25]

[24] (Draw.io, 2017)
[25] (LucidChart, n.d.)

User Data

Though the accounts for helpers and accounts for the elderly could easily be combined together, this would create a relationship between appointments and the accounts table which would not be desired. Therefore, separating these two ensures that the relationship is One-Many and is as efficient as possible. Additionally, this ensures that if in the future, more or less information is desired from one of the groups, then this could be easily adjusted without affecting the other group.

However, mostly the same information is being stored about both parties with the exception of the inclusion of a timetable id for the helpers.

Hence, we can consider them together for now.

Name	Variable Type	Description
id	Int	This is the key for the table, acting as the counter. This auto-increments every time a new user is added to the table – using this function when creating the MySQL table.
emailAddress	Varchar(50)	The email address acts as part of the contact information provided to both parties when an appointment is set up. It is also used as the piece of information which is used to login. This ensures that there is no additional username which is required to be memorised.
password	Varchar(50)	The password contains information (though hashed) of the password which the user needs to use to access the application and connect them to their account information.
firstname	Varchar(25)	Contains the user's firstname.
surname	Varchar(25)	Contains the user's surname.
firstLineOfAddress	Varchar(100)	Contains the user's first line of the address.
secondLineOfAddress	Varchar(100)	Contains the user's second line of their address. Optional piece of information, as some people

		do not require it to describe their location.
telephoneNumber	Varchar(100)	Contains the user's telephone number.
postalCode	Varchar(25)	Contains the user's postcode.
City	Varchar(25)	Contains the user's city.
Country	Varchar(35)	Contains the user's country.
timetableId	*Int*	*In the case of the helpers database, this contains the id of the weeklyTimetables database which contains information of the timetable.*

Figure 7: Example of data from 'AccountForHelpers' database

id	username	emailAddress	password	firstname	surname	firstlineOfAddress	secondlineOfAddress	telephoneNumber	postalCode	city	country	timetableId
1	ahujaas	ashwin@gmail.co	ash	Third	Closest-Helper	a	a	a	W1A 0AX	a	a	1
5	NULL	testj	ah7	Closest	Helper	sjkdhs	sds	dsdsd	NW35NR	sds	dsdsd	2
9	ashwinahuja	ashwin@gmail.com	3ea7121feb7564ddfe2d1f2ebb41706e	Helper	Ahuja	10 Lyndhurst Gardens	Hampstead	447597711497	SW138QY	London	United Kingdom	68
10	jdnsna	usnamms	47bce5c74f589f4867dbd57e9ca9f808	hshshs	jsjsjs	ndnsjs	usjsjshsj	447597711496	sw138qz	jajsjjd	nsnaja	8
11	vech	vechtest2@gmail.com	bf2bc2545a4a5f5683d9ef3ed0d977e0	oli	vech	hshsjs		447768147040	SW100XD	London	United Kingdom	36
12	ahuja	ashwinahuja@gmail.com	3ea7121feb7564ddfe2d1f2ebb41706e	Ashwin	Ahuja	32 Rivers Road		447597711493	SW138QY	London	United Kingdom	39
14	a	aab	0cc175b9c0f1b6a831c399e269772661	a	a	nsn	jsns	447597711495	SW138QY	ndnd	hdbd	51
17	Hi Ashwin	philafern82@gmail.com	4944b366079773bebd70485ca180aafc	Philip	Fernandes	82 West Hill	Wales	447946505344	HA99RR	London	UK	57
19	testUsernameNEW	testEmailAddressNEW	testPasswordNEW	testFirstNameNEW	testSurnameNEW	testFirstLineOfAddressNEW	testSecondLineOfAddressNEW	testTelephoneNumberNEW	SW139JT	testCityNEW	testCountryNEW	59
21	username	email	5f4dcc3b5aa765d61d8327deb882cf99	name	surname	firstline	secondline	telephone	postalcode	city		63

Source: SQL database viewed using Sequel Pro[26]

Timetables

While the timetables table could theoretically be combined with the main user's database for the helpers, by separating these two tables, the system is clearer and easier to work with.

The timetable system itself is separated into a few different tables. The first consists of a table with multiple columns, each corresponding to a week of the calendar (referred to by the Monday date of the week). There is also a default calendar which is used for all weeks which are not otherweise referred to specifically by a column. When an appointment is created, the timetable must be altered to make that time on that day no longer be free. If the week is already referred to by a column (there has been another appointment by anybody during that week), then the entry of the week can simply be changed. Otherwise, the column is created, and every other entry has a clone of their default week added in that column.

Each row, in each column refers to an entry in the weeks table, which contains the specific ids of the days to which the week refers. Finally, in the day table, the specific information about the day in question is stored, with details about each two hours slot and the person's availability at this time.

[26] (SequelPro, n.d.)

Name	Variable Type	Description
Id	Int	This is the key of the table, and auto increments whenever a new entry is added.
defaultWeekId	Int	This contains the id of the default week which is created when the user first signs up to an account.
DD_MM_YYYY **(i.e. Week Beginning DD/MM/YYYY)**	int	There may be n columns of this, where each column contains either a null, if this entry simply uses their default timetable for this week, or contains an id for another week entry linking to the week table. Whenever a new column is added, a check is run whether the DD/MM/YYYY is in the past, and whether it can therefore be deleted. This ensures that the data remains reasonably small and cannot threaten to break the server.

Figure 8: Example of data from the 'Timetable' database

id	defaultWeek	13_03_2017	06_03_2017	27_03_2017	10_04_2017	03_04_2017	17_04_2017	01_05_2017	20_03_2017
1	1	1	1	1	1	1	1	1	1
2	1	24	30	32	38	44	50	94	131
3	13	21	27	33	39	45	51	95	132
4	1	22	28	34	40	46	52	96	133
5	1	23	29	35	41	47	53	97	134
6	55	56	57	58	59	60	62	98	135
7	63	64	65	66	67	68	69	99	136
8	70	71	72	73	74	75	76	100	137
9	77	78	79	80	81	84	83	101	138
10	85	86	87	88	89	90	91	92	139
11	103	111	105	106	107	108	109	110	140
12	112	120	114	115	116	117	118	119	141
13	121	122	123	124	125	126	127	128	129
14	143	144	145	146	147	148	149	150	152
15	153	154	155	156	157	158	159	160	161
16	162	171	164	165	166	167	168	169	170
17	172	173	174	175	176	177	178	179	180
18	181	182	183	190	185	186	187	188	189
19	191	192	193	194	195	196	197	198	199
20	200	209	202	203	204	205	206	207	208
21	210	211	212	213	214	215	216	217	218
22	219	228	221	222	223	224	225	226	227
23	229	230	231	232	233	234	235	236	237
24	238	239	240	241	242	243	244	245	246
25	247	248	249	250	251	252	253	254	255
26	256	257	258	259	260	261	262	263	264
27	265	266	267	268	269	270	271	272	273
28	274	275	276	277	278	279	280	281	282
29	283	284	285	286	287	288	289	290	291
30	292	293	294	295	296	297	298	299	300
31	301	302	303	304	305	306	307	308	309
32	310	311	312	313	314	315	316	317	318
33	319	320	321	322	323	324	325	326	327
34	328	329	330	331	332	333	334	335	336
35	337	338	339	340	341	342	343	344	345
36	346	347	348	349	350	351	352	353	354
37	355	356	357	358	359	360	361	362	363
38	364	365	366	367	368	369	370	371	372
39	373	374	375	376	377	378	379	380	381
40	382	383	384	385	386	387	388	389	390
41	432	433	434	435	436	437	438	439	440
42	442	443	444	445	446	447	448	449	450
43	452	453	454	455	456	457	458	459	460
44	462	463	464	465	466	467	468	469	470

Source: SQL database viewed using Sequel Pro[27]

As mentioned, defaultWeekId and WeekBeginning… link to the week table, which provides links to an entry in the day table for each day of the week.

[27] (SequelPro, n.d.)

Name	Variable Type	Description
Id	int	This is the key of the table, and is referred to by the WeeklyTimetables table.
MondayId	Int	This contains a link to an entry of the day table, containing the specific time when a helper is free.
TuesdayId	Int	…
…	Int	

Figure 9: Example of data from the 'Weeks' database

id	mondayId	tuesdayId	wednesdayId	thursdayId	fridayId	saturdayId	sundayId
1	1	1	1	1	1	2	2
2	2	2	1	2	1	2	1
3	1	10	11	1	1	1	1
4	1	1	1	1	9	7	6
5	12	1	1	1	9	7	6
6	13	1	1	1	9	7	6
7	12	1	1	1	9	7	6
8	12	1	14	1	9	7	6
9	12	1	1	1	9	7	6
10	12	1	15	1	9	7	6
11	12	1	1	1	9	7	6
12	12	1	1	1	9	7	6
13	12	1	1	1	9	7	6
14	12	1	1	1	9	7	6
15	12	1	1	1	9	7	6
16	12	1	1	1	9	7	6
17	12	1	1	1	9	7	6
18	12	1	16	1	9	7	6
19	25	1	1796	1	9	7	38
20	12	1	1	1	9	7	6
21	12	1800	2058	1	9	7	6

Source: SQL database viewed using Sequel Pro[28]

This then has links to the ids of entries in the day table, where the actual information as to whether a user is free is stored.

Name	Variable Type	Description
Id	Int	This is the key of the table, and is linked to by the entries of the week table.
a02	Bool	This contains the information as to whether the user is free at

[28] (SequelPro, n.d.)

		this time. It was decided that 2 hour intervals were the correct length – offering simplicity for both booking appointments and for creating a timetable as a helper.
a24	Bool	As above
…	Bool	As above.
a2224	bool	As above. Though it was considered unlikely that appointments would be booked in some of these timeslots it was decided that it would be better if they existed anyway, never to be used. However, it is important that in the User Interface for creating this timetable, that the work is minimised by defaulting that the timeslots which are the least likely are defaulted as being false.

Figure 10: Example of data in the 'Day' database

id	a02	a24	a46	a68	a810	a1012	a1214	a1416	a1618	a1820	a2022	a2224
1	0	0	0	0	0	0	0	0	1	1	1	0
2	0	0	0	0	1	1	1	1	1	1	1	0
3	1	1	0	0	0	1	0	0	0	0	0	1
4	1	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	1	1	1	0	0	1	1	0
7	1	1	0	0	0	0	0	0	0	1	1	1
8	1	1	0	0	0	0	0	0	0	1	1	1
9	0	0	0	0	0	0	0	0	0	0	0	0
10	1	0	0	0	0	0	0	0	0	1	1	1
11	1	0	1	0	0	0	0	0	0	1	1	1
12	1	0	1	0	0	0	0	0	0	1	1	1
13	1	0	1	0	0	0	0	0	0	1	1	1
14	1	1	1	0	0	0	0	0	0	1	1	1
15	1	1	1	0	0	0	0	0	0	1	1	1
16	1	1	1	0	0	0	0	0	0	1	1	1
17	1	0	1	0	0	0	0	0	0	1	1	1
18	1	0	0	0	0	0	0	0	0	1	1	1
19	0	0	0	0	1	1	1	0	0	1	0	0
20	0	0	0	0	1	1	0	0	0	1	0	0

Source: SQL database viewed using Sequel Pro[29]

[29] (SequelPro, n.d.)

Default Calendar

When a new user is created, a new timetable is also created, with an initial timetable which consists of being free during the afternoon and evening during the week and during the entire day during the weekend. This was because we anticipate people being at school during the week, and therefore being busy during those times.

Appointments

The appointments information contains the information of the appointment for both the helper and the elderly person, with an entry being created when an elderly person creates an appointment, either via the app or via the SMS system.

Name	Variable Type	Description
Id	Int	This is the key for the table.
helperId	Int	This contains the information of who is the helper who has been assigned to solve the problem.
elderlyId	Int	This contains the information of the elderly person to be helped.
dateAndTime	DateTime	This contains the date and time of the appointment
dateAndTimeCreated	DateTime	This contains the date and time that the appointment has been created.
ratingId	Int	This contains the id (reference to the ratings table) of the rating given to the helper after this job has been completed.
reviewId	Int	This contains the id (reference to the reviews table) of the helper once the job has been completed.

Ratings

The ratings table is where the ratings of the job are stored, with each appointment having one or no entry in the ratings table.

Name	Variable Type	Description

Id	int	Key of the table
starRating	Int (range between 0 and 10)	This contains the information of how many stars the helper has been given for this particular job.

It's important to note that it was decided that the options for the ratings of the helpers would only be integers between 1 and 10. This was decided in order to simplify the system as much as possible – ensuring that the system would be easy to use for the elderly person who is trying to rate their helper.

Reviews

The reviews table is where the reviews of the job are stored, with each appointment having one or no entry in the table.

Name	Variable Type	Description
id	int	Key of the table.
review	Varchar(10000)	This contains the review – which has been decided to have a maximum length of 10,000 character – around 1,500 words. This has largely been put as an exceedingly large maximum. We expect the length of each review to be significantly shorter. Therefore, the MySQL database has been programmed to reduce the amount of space used by this field to be reduced to the length of the text, once it has been added.

2.2.2 - Communications with applications

There were a couple of ways to communicate with the database to get all the required data from it. The first was through immediately communicating with it, while the second involved producing a local API which could be used to communicate to the SQL server and return the required information. The first, while inherently simpler, offers the challenges that code would have to be repeated on all applications when tasks like verification of data entry would be required. Even more importantly, it

means that the security is much reduced as the data could be intercepted en-route as encrypting this transfer of data would be much more challenging. In fact, though Oracle do provide a system by which this connection can be established, they expressly suggest that you avoid it. Therefore, I decided to produce an API which could allow the code to be securely centralised.

2.3 – WebAPI (Appendix A – Web API Code)

I decided to use the REST standard for creating an API, which offers a number of important steps for communicating with HTTP requests, ensuring they are only of type: "GET", "POST", "PATCH", "DELETE", "PUT". It also established the use of Json to both send and receive data between the client and the server.

I also decided to use a .NET core API, as I had prior knowledge of .NET programming, therefore would not have to learn how to use PHP, the other language which appears to be largely used for this. By using Microsoft's IIS and .NET core libraries for Ubuntu 14.04 meant that I could largely test using my computer using a local SQL server as the database to communicate with as they were identical on my computer as on the server.

2.3.1 - General Structure of .NET Core API

The .NET core API effectively creates a tunnel through which a number of controllers can act as listeners, awaiting a command which meets the HTTP request type that they are defined to listen to – responding to the command when the command type and URL are in one of the controllers.

In order to produce this, there must be a number of files which define how the API works, including defining the namespace of the controllers and defining the dependencies and the runtimes that the API should work on.

All the controllers must inherit from the AspNetCore.Mvc.Controller class. In fact, in order to simplify in dealing with the SQL server, there is an intermediary class, SQLController (SQLStuff.cs), which inherits from Controller. This is inherited by all of the controllers. Hence, there is the following hierarchy chart.

Figure 11: Hierachy Chart of Controllers

In order to communicate with the SQL server, the .NET Oracle SQL Server package is used, being acquired with a NuGet request to ensure that it can be retrieved every time the server script is run, thereby ensuring it exists everywhere.

2.3.2 - Communicating with JSON

Communicating with JSON was the obvious choice, with the only other option being XML. However, the greater availability of libraries to parse the JSON data on both the android application side and the Web API side meant that JSON was a better choice. An example of communicating with JSON in the WebAPI is available here: HttpCommandOut.cs

For the Web API, in order to deserialise data which is inputted to the WebApi, the '[From Body]' tag is used, which calls the System.Net Json deserializer to process the data to the object which is defined in the definition of the function. When data is to be sent as Json, it is either passed directly as the object, as this would automatically be converted to Json or it is explicitly converted to Json using the NewtonSoft.Json library if this is necessary.

2.3.3 - Communicating with the SMS Server (SmsStuff.cs)

In order to communicate with the SMS server and the Google Maps API, which require GET and POST requests, the HttpWebClient library (accountsForHelper.cs) is used (in a composition fashion, in order to allow the rest of the system to function without adaptation).

2.3.4 - Functions

	Function Name	**HTTP type**	**URL**	**Input**	**Output**
Accounts (Accounts.cs)			/ac		
	HelloWorld	GET	/ac		"Hello World"
	AddNewUser	POST	/ac/users/ad	User	Boolean (success or failure)
	VerifyUser	POST	/ac/users/vf	UserToBeVerified	User (empty if failed, populated if successful)
AccountsForHelpers (accountsForHelper.cs)					
	HelloWorld	GET	/ach		"Hello World"
	GetInfo	GET	/ach/users/gi{id}	Int id	Returns a user (see the

					definition of the user class below).
	GetNearestHelpers	GET	/ach/users/gl{id}	Int Id of the current user,	List of the nearest helpers, to allow the elderly to pick which of the people to pick.
	DeleteUser	DELETE	/ach/users/rm	Int id of the user	Returns a bool of success or failure.
	UpdateUser	PATCH	/ach/users/up	Int id of user, user (see definition of class below)	Returns bool of success or failure
	AddNewUser	POST	/ach/users/ad	User (see definition of class below)	Returns bool of success or failure
	VerifyUser	POST	/ach/vf	UserToBeVerified (see definition of class below)	Returns id of userLoggedInto or null if failedToLogin
	mergeSort	NA	NA	List of users and address of elderly to sort by nearest to	Sorted list of users
	merge	NA	NA	Two lists of user to merge	One list of merged users
	getDistance	GET	/ach/users/gd	Two string addresses	Double distance between the two in km
	fileGetContents	NA	NA	String filename	String contents
AccountsForElderly (accountsForElderly.cs)					
	HelloWorld	GET	/ace		"Hello World"
	GetInfo	GET	/ace/users/gi{id}	Int id	Returns a user (see the

					definition of the user class below).
	GetInfoByTelephoneNumber	NA	NA	String telephoneNumber	Returns a user (see the definition of the user class below).
	DeleteUser	DELETE	/ace/users/rm	Int id of the user	Returns a bool of success or failure.
	UpdateUser	PATCH	/ace/users/up	Int id of user, user (see definition of class below)	Returns bool of success or failure
	AddNewUser	POST	/ace/users/ad	User (see definition of class below)	Returns bool of success or failure
	VerifyUser	POST	/ace/vf	UserToBeVerified (see definition of class below)	Returns id of userLoggedInto or null if failedToLogin
Timetables (Timetable.cs)			/tt		
	getTimetablesOfAUser	GET	/tt/gt	Int helperID	Returns timetable (see below for definition)
	addNewTimetable	POST	/tt/adt	timetable	Returns bool for success or failure
	UpdateWeek	NA	NA	week	
	AddDay	NA	NA	day	Int id (in database)
	UpdateDay	POST	/tt/ud	day	Returns bool for success for success or failure.
	addWeek	NA	NA	week	Int id of week

	getWeek	NA	NA	Int id	week
	getDay	GET	/tt/gd	Int id	day
Appointments (Appointments.cs)					
	Create Appointment From SMS Request	POST		DateTime time, string telephoneNumber	Returns appointment (see definition below)
	Create New Appointment From Android App	POST		Appointment (see definition below)	Returns bool of success or failure
	Get Appointment information	GET		Int AppointmentID	Returns appointment
	Remove Appointment	DELETE		Int appointmentID	Returns bool of success or failure
	Get Appointments by Elderly	GET		Int elderlyID	Return a list of appointments
	Get Appointments by helper	GET		Int helperID	Returns a list of appointments
Ratings (Ratings.cs)					
	Get All Ratings Of a User	GET		Int helperId	Returns list of ratings (for definition see below)
	Get One specific rating	GET		Int appointmentID	Returns rating
	Add a rating to an appointment	POST		Int appointmentID, rating	Returns bool of success or failure

	Update rating	PATCH		Int appointmentID, rating	Returns bool of success or failure
	Delete rating	DELETE		Int appointmentID	Returns bool of success or failure
Reviews (Reviews.cs)					
	Get all reviews of a helper	GET		Int helperID	Returns a list of reviews
	Get one specific review	GET		Int appointmentID	Returns review
	Add a review to appointment	POST		Int appointmentID, review	Returns bool of success or failure
	Update a review	PATCH		Int appointmentID, review	Returns bool of success or failure
	Delete a review	DELETE		Int appointmentID	Returns bool of success or failure

The classes which are used on both the clients and the server which define the types and objects for the objects user in communicating with the WebAPI are as follows:

For each item in the classes, they contain private items which are held as well as public getter and setter methods which allow the private items value to be changed. For simplicity, only the private item is referred to.

Class	**Item Name**	**Type**
User (User.cs)		
	Id	int
	Username	String
	emailAddress	String
	Password	String

	Firstname	String
	Surname	String
	firstlineOfAddress	String
	secondLineOfAddress	String
	telephoneNumber	String
	postalCode	string
	City	String
	Country	String
	timetable	timetable
Timetable (timetables.cs)		
	id	Int
	defaultWeek	week
	listOfOtherWeeks	List<week>
Week (Week.cs)		
	id	int
	weekBeginning	DateTime
	monday	Day
	tuesday	Day
	Wednesday	Day
	Thursday	Day
	Friday	Day
	Saturday	Day
	Sunday	Day

Day (Day.cs)		
	id	int
	timesFree	Bool[12]
Appointment (Appointment.cs)		
	Id	
	helperid	Int
	elderlyid	int
	dateAndTime	DateTime
	dateAndTimeCreated	DateTime
	Rating	Rating
	review	Review
Rating (Rating.cs)		
	id	Int
	rating	Int
Review (Review.cs)		
	Id	Int
	review	String
UserToBeVerified (UserToBeVerified.cs)		
	emailAddress	String
	password	String
	helper	Boolean

DateTimeExtensions (DateTimeExtensions.cs)		
	Func getWeekBeginning (DateTime dt)	DateTime

While the majority of these simply require a number of SQL commands to return specific data from the MySQL database, a couple of the functions require a little more explanation of how they will work.

*Returning List of Nearest Helpers (**accountsForHelper.cs**)*

To order the list of helpers, there are broadly three stages which must be followed:

1. Get all helpers
2. Find the distance from the elderly person for each helper.
3. Sort the helpers by this distance.

Get all helpers

In order to get all the helpers, it is simply a question of calling a get command of all helpers (this command is only available internally and is not available outside of the API and can't therefore be exploited).

Find the distance from the elderly person for each helper

There were two methods considered to find the distance between two addresses. The first, utilises, just the Google Maps API and treats the direction distance (distance on the road) as the distance. The second uses the Google Maps API to get the longitude and latitude and uses the Haversine Formula to extract the distance between two points using this formula.

It was decided that the former is better as it is more representative of the time taken by the helper to get to the elderly person's house. Therefore, the Haversine formula is not needed.

Sort the helpers by this distance

In order to sort the helpers by this distance, a recursive MergeSort is used, recursively slitting the list into halves, sorting each half before combining them together using a merge. The process is outlined in the below pseudocode:

```
MergeSort (List a):

        If (a.Count() == 1)

                Return a;

        Else

                Return merge(MergeSort(firstHalfOfA), MergeSort(secondHalfOfA))
```

```
Merge (List a, List b):
    List c
    While(a and b have elements):
        If (a[0].distance < b[0].distance)
            c.Add(a[0]);
            a.RemoveAt(0);
        Else
            c.Add(b[0]);
            b.RemoveAt(0);
    While(a has elements):
        c.Add(a[0]);
        a.RemoveAt(0);
    While(b has elements):
        c.Add(b[0]);
        b.RemoveAt(0);
    return c
```

Create New Appointment from SMS request (Appointments.cs)

In order to do this, the function completes the following tasks:

1. Gets the user information of the elderly person from the telephone number
2. Gets the list of nearby helpers to the elderly person
3. Parses the sms into a date and time
    a. ^
4. Find the first helper who is free at that date and time, book appointment (using create new appointment from Android App)
    a. *
5. Send an SMS to the original sender with the information about the appointment.

Considering ^

In order to parse the SMS and return the date and time that the user wants to have the appointment, there were a few possible ways of doing it. The first involves creating my own system to search for all possibilities including things like 'tomorrow' and 'today' which would each need to separately catered for.

Therefore, it would be simpler to make use of existing APIs. While it would be significantly more effective to use something like the IBM Watson Natural Language processing API, it proved much simpler to use a DateTimeNaturalLanguageParser for C# that I found published under the MIT license on GitHub (DateTimeRoutines.cs). While not perfectly effective – for example the year must always be stated otherwise, it does not return any of the correct date – it generally works.

Pseudo Code of *

```
Foreach helper in list of helper:

        Timetable t =  Get Timetable (helper.timetableId)

        DateTime dt =  findWeekBeginning(dateTimeSMS)

        If dt.ToString() exists in t.weeks[x].weekBeginning:

                Day correctDay = t.weeks[x].week.(dateTimeSMS.DayOfWeek.ToString() + "id")

                If correctDay.timesFree[dateTimeSMS.Hour / 2]:

                        Create appointment with this helper at this time

                        Send SMS to user with this information

                Else

                        Next helper

        Else

                Day correctDay = t.defaultWeek.(dateTimeSMS.DayOfWeek.ToString() + "id")

                If correctDay.timesFree[dateTimeSMS.Hour / 2]:

                        Create appointment with this helper at this time

                        Send SMS to user with this information

                Else

                        Next helper

If still haven't found a helper, then send SMS saying "Noone could be found"
```

Create New Appointment from AndroidApp (Appointments.cs)

The function must complete the following tasks:

1. Convert the dateAndTime into the week beginning data and the day.
2. Check whether that week beginning is a special week for the helper, or whether it falls inside the default week.
    a. It does this by getting the timetable for the helper, and checks whether the date of the week beginning is the same as the week beginning date for any of the custom weeks.

        i. If it isn't one of the weeks already there, then it creates a new column and clones the defaultWeek from all the items and puts their ids into the new column.
        ii. Then, it recurs on the same function, this time confident that it will find the new column
3. Check whether the helper is free at the time on that day, looking inside the correct entry of the 'day' table.
4. If it is free, the appointment can be booked and the time should be changed to being used and the appointment added to the tableOfAppointments.

Retrieving timetable from database (*timetable.cs*)

1. Go into the INFORMATION-SCHEMA for the table and retrieve the names of the columns.
2. Then retrieve the value of the entry for each of these columns – converting the name of the column into a DateTime value for the weekbeginning date for the column
    a. Then for each of these weeks, extract the week from the week value.
        i. Do this by extracting a day from the day ids from each day of the week table of the week entry.

2.3.5 - Securing data

An important part of the decision to choose to use a WebAPI was in to maximise the security of the system, by ensuring that people couldn't retrieve the placement of the server and simply issue SQL commands to retrieve all the data on all the users. There are a couple of other simple measures which can be taken to further maximise the security of the data, even if the connection is insecure.

Hashing Passwords (MD5Hash.cs)

By hashing password, we can maximise the security of the passwords in a couple of different ways. Firstly, from a user's perspective, it assures them that we don't have access to the password and thereby ensures that they are comfortable with using the site (though it would still be advisable to use a different password than others for this account). It also means that should hackers manage to penetrate the security of the server, which will be minimised as far as possible, through root security in Ubuntu 14.04, they will find it difficult to retrieve passwords from the hashed values of the passwords.

Therefore, the passwords will be stored as the hashed value – with the passwords being hashed using the MD5 cryptographic hash, which produces a 128-bit hash. This, while not perfectly secure, draws a good compromise between the time taken to hash on the client side (on the app) and security.

2.3.6 - Verifying data entry

It has been decided that for speed of the system, the main verification will be done on the application side, being completed without requiring the application to make any calls to the WebAPI. This also ensures that the end user can have more coherent error messages. However, there are a few things which cannot be checked by the application and therefore must be checked during the server's functions

Therefore, the following checks must be carried out during verification:

Task	Input	Check
Adding or updating a user **(accountsForElderly.cs, accountsForHelper.cs)**		
	emailAddress	Already used?
	telephoneNumber	Already used?

2.4 - Android App

While I had otherwise intended to create two android apps, one for the helpers and one for the elderly since I assumed that they would largely be entirely separated, I have found they are in fact rather similar, and by creating 'one' app, they could inherit large portions of the backend code from one another. Therefore, I am only creating one android app, though with two separate timelines for the elderly and for the helpers. Therefore, for the purposes of this report, I am going to deal with the design together, though they share a couple of screens.

For the Android apps, I chose to work in Xamarin.Android, which allows me to use existing skills in C# and XML and combine them to produce a functioning Android Application. This also means that I can reuse much of the code for the classes in the WebAPI in order to produce the Android Apps, for example the classes for the users and timetables.

2.4.1 - Important Features

Communicating with Web API

In order to communicate with the WebApi, the Android App makes use of the RESTSharp library, which deals with the communication entirely, allowing the programmer to just say the method, the data, and the URL of the communication. (An example of this is in the UserDetails function here: WelcomeHelpers.cs)

Unfortunately, these functions must be often repeated on a number of screens as most of the RestSharp functions require an activity as a context for the request to be sent, therefore they must be included as part of the activity itself rather than any other class.

In order to serialise the data to be sent to the API and to deserialise the JSON input from the API, the NewtonSoft JSON library is used, with it defining the object and object type to convert to JSON.

Communicate with SMS Server (RegisterActivity.cs)

To communicate the SMS server, the Android App again makes use of the RESTSharp and NewtonSoft.Json Libraries which can be acquired using a NuGet packages and therefore are included as part of the package of the Android application when it is installed.

Remembering Logins (LoginActivity.cs)

In order to remember the login and password, the Android application stores the information in a SQL database, using the SQLite database to store the following information:

Name	Type
emailAddress	VARCHAR(50)
password	VARCHAR(50)

When the user logs in, the username and password is added to the database. The password is added in plaintext as it would be impossible for anyone to gain access to the database. When the login screen is reached again at any point, the application first checks through the database and attempts to login with information from the database.

2.4.2 - Helpers

Main Flowchart

The flowchart shows the system without the addition of in case of problems.

Figure 12: Flowchart of Helpers Android Application

Source: Produced using draw.io[30]

User Interface

Dialog Methods

In order to give critical information to the user, two main methods were used, where one entirely removes everything else from the page, and forces the user to look at it. This is an Android Alert Dialog. This is used mostly during the login and registration, where the information to be gained is vitally

[30] (Draw.io, 2017)

important to the function of the application and the success of the user to get into the application properly.

The other type are toast notifications, which are used more subtally, for example to tell the user that a change that they have made has gone through successfully, as in the below example

Alert Dialogs	Toast Notifications

Start

UI	Activity
	OnCreate: Initialises view. **LoginButtonClick:** When the button is clicked, go to the login screen. **RegisterButtonClick:** When the button is clicked, go to the register screen.

The start allows a user to create a new account or login by clicking one of the buttons.

Log In

OnCreate: Go through the database and attempt to login with any of the accounts stored in the database or create a table if it doesn't already exist.

LoginAsHelperClick: Attempt to login as a helper.

LoginAsElderlyClick: Attempt to login as an elderly.

checkIfEmpty: Shows a dialog box if all information is not present

login: Returns a true or false dependent on whether the login is successful.

The login window provides two text boxes with text explaining what is to go in the boxes. The password boxes will, once filled in, appear as a series of dots, by using the Android password box, in order that those in the vicinity of the user not be able to see the password.

Register

The registration screen, common for both the elderly and helpers offers the opportunity for the users to enter all the required information to register. Though I considered creating a toggle switch for elderly or helper, this separate buttons method appears more clear and easy to use – this is very important for the elderly. Once the registration has gone through, there is an alert saying "registration succeeded" or "registration failed". If the registration is successful, they receive a message with the verification code for the twilio call which they will receive within the next two minutes.

OnCreate:

RegisterRegisterAsHelperClick:

- checkValid
- registerTheUser
- verifyNumber
- checkVerify to the get the userId
- if it doesn't get verified, deleteUser (and force them to reregister)

RegisterRegisterAsElderlyClick:

checkValid:

- Checks if any fields are empty or if the telephone number is invalid.
- If invalid then show dialog box and returns false

deleteUser:

- Sends the REST request to the API to delete the user account.

testVerify:

- Completes the same function as in login and returns the user.

register:

- POSTs the data to the API to add the account.

verifyNumber:

- POSTs the number to the SMS server and shows the return, the verification code, to verify the telephoneNumber.

96% 17:50
Email Address: ashwinanuja@gmail.com
Username:
Password: •••••••••
Confirm Password: •••••••••
First Line of Address:
Registration Unsuccessful
Not all required fields have been filled.
Country: United Kingdom
Postal Code: SW138QY
Telephone Number: 07597711493
REGISTER AS HELPER
REGISTER AS ELDERLY

Logged in

Upon login, the user gets to access either part of the app, the appointment side, or the timetable side through two buttons.

	OnCreate: • Populate the name textview using information from GetUserDetails. **SetTimetableClick:** • Open the setATimetable activity**.** **ListOfAppointmentClick:** • Open the listAnAppointment activity. **GetUserDetails:** • Send the REST request to the Web API and returns the user details.

View Appointments

From any deeper than this in the app, there are always back buttons (on the android device) which allow the user to move back in the pages they have visited, therefore allowing them to visit every site. This screen easily shows all the appointments, displaying their times.

The view appointment screen simply offers the list of all of the appointments which exist. They are ordered such that the upcoming appointments are listed first (closest first) before they go backwards in time. For each appointment, it says the name of the elderly person with whom the appointment will (or has) occurs as well as the time.

All the appointments can be clicked on to get more details about the appointments.

OnCreate:

- Get list of appointments.
- Foreach of the appointments, get the name of the elderly person (using getUserDetails).
- Populate the list view in the order of upcoming first (closest first), then going back in history

OnListItemClick:

- Call the activity to show more details about that appointment.

returnListOfAppointments:

- From the helperId, this makes a call to the REST api and returns the list of appointments.

getUserDetails:

- Makes a call to the Web API and returns information about the user.

See details of Appointment

This is the view which is displayed when the user clicks on any one of the appointment. It offers more information about the appointment, including things like the time, contact number and address of the appointment. Additionally, if the appointment is in the future, the appointment can also be cancelled.

OnCreate:

- GetAppointmentDetails of appointment to show in more details.
- Populate the text views using details from GetUserDetails
- If in the past, ensure that cancel appointment is not enabled. If in the future, it should be enabled.

CancelAppointmentClick:

- Send the request to the Web API.

GetUserDetails:

- Send request to Web API and return the user information.

GetAppointmentDetails

- Send request to Web API and return the appointment.

Set Timetable

The process for setting up a timetable is split into a number of parts, with the person first choosing which week to tackle (with the next seven weeks displayed), followed by which day, followed by then changing the timetable for that particular day.

The first screen shows the week beginning dat for the next seven weeks, allowing them to click one of them and alter their timetable in any one of those weeks.

	OnCreate: • Populates the view with the readable name (MMMM dd, yyyy) of the week beginning date of the next seven weeks which could be set. **OnListItemClick:** • Starts the activity of the setting timetable of week view – passing the date beginning date as a parameter.

Upon clicking upon one of the weeks, the next screen displays the date of all the days in that week, each of which can be clicked to set the timetable for that day.

OnCreate:

- Populates the view with the dates of all of the days of the week – therefore any of them can be selected and the free time can be changed.

OnListItemClick:

- Starts the activity of the setting the free times, passing the parameter of the exact day whose timetable should be changed (based on which item has been clicked).

On clicking any one of the days, the screen for setting the timetable for that day is opened. It is a simple interface consisting of only toggle switched, which show whether the person is free for that specific time. Though the free and busy positions are not labelled, due to the clear 'on-off' colouring, it is reasonably obvious that on means free.

There is a submit button at the bottom which when clicked commits the changes to the timetable. When the change occurs, there is a toast notification, which tells the user that the change has successfully occurred.

OnCreate:

- Initialises the view and calls the getCorrectStatusOfButtons

getCorrectStatusOfButtons:

- Calls the getTimetableOfHelper and finds the correct day value (either from a specific week or from the default week entry).
- Populates the toggle switches on the view with 'on' or 'off' based on whether they are currently free at that time or not.

submitButtonClick:

- Calls assemble day and replaces the existing day in the timetable object of the user with this day.
- If the week already exists, this is simply changing a day, otherwise a new week is created and the day can then be replaced.
- Then a request is sent to the WebAPI to add the timetable and replace the timetableId in the the accountsTable.

assembleDay:

- Converts the status of the buttons into a day object.

getTimetableOfHelper:

- Sends a request to the WebAPI with the parameter of the id of the timetableId and returns the timetable of the user.

2.4.3 - Elderly

Main Flowchart

The flowchart below is the main flowchart of the app, showing how a user can carry out all of the task- with each of the boxes corresponding to a single screen on the design. It must however be noted that this assumes successes in all activities, such as assuming that the password is valid and the user successfully logs in.

Figure 13: Flowchart of Elderly Android Application

Source: Produced using draw.io[31]

User Interface

Design Ethos and Colours

Foremost in my mind when designing the system for the elderly was the suggestion by Mrs Wilson that no screen should have more than two things happening on that screen. Therefore, a large (possibly excessively so) number of screens exist in the application in order to allow the user to be able to cope easily with each of the screens.

With regards to colours, additional research revealed that due to issues with eyesight, in particular the fact that the retina tends to break down more quickly than the rest of elderly eyesight, they tend to lose the ability to see colour. Therefore, it is more important to focus on contrast, to show differences in parts of the design. Therefore, the design generally consists of a 'Black-White' colour scheme, to maximise the contrast.

Start

The first few screens are entirely identical to those in the helpers side, so no additional commentary is provided.

[31] (Draw.io, 2017)

	OnCreate: Initialises view. **LoginButtonClick:** When the button is clicked, go to the login screen. **RegisterButtonClick:** When the button is clicked, go to the register screen.

Login

OnCreate: Go through the database and attempt to login with any of the accounts stored in the database or create a table if it doesn't already exist.

LoginAsHelperClick: Attempt to login as a helper.

LoginAsElderlyClick: Attempt to login as an elderly.

checkIfEmpty: Shows a dialog box if all information is not present

login: Returns a true or false dependent on whether the login is successful.

Register

OnCreate:

RegisterRegisterAsHelperClick:

- checkValid
- registerTheUser
- verifyNumber
- checkVerify to the get the userId
- if it doesn't get verified, deleteUser

RegisterRegisterAsElderlyClick:

checkValid:

- Checks if any fields are empty or if the telephone number is invalid.
- If invalid then show dialog box and returns false

deleteUser:

- Sends the REST request to the API to delete the user account.

testVerify:

- Completes the same function as in login and returns the user.

register:

- POSTs the data to the API to add the account.

verifyNumber:

- POSTs the number to the SMS server and shows the return, the verification code, to verify the telephoneNumber.

96% 17:50
Email Address: ashwinanuja@gmail.com
Username:
Password: •••••••••
Confirm Password: •••••••••
First Line of Address:
Registration Unsuccessful
Not all required fields have been filled.
Country: United Kingdom
Postal Code: SW138QY
Telephone Number: 07597711493
REGISTER AS HELPER
REGISTER AS ELDERLY

Logged in Interface

Here, for the elderly, instead of offering the timetable, the user can book an appointment. However, the process for booking an appointment is similar to that of setting a timetable in the sense that it is dealt with in a number of pages, each one making more specific the time and person involved with the appointment.

OnCreate:

- Sets up the view of the interface and populates the name of the elderly person.
    - Does this by getting the name from the user returned by the GetUserDetails function.
- It also checks if an appointment has just been booked. If this is true, then the user is redirected immediately to the page with the list of appointments, through Click2.
- *The only reason this exists is so that, if the user were to now press the back button on their phones, they would return to this screen rather than to the booking confirmation screen.*

BookAnAppointmentClick:

- Starts the activity related to the booking of an appointment (where the nearest helpers are shown).

ListOfAppointmentsClick:

- Starts the activity related to listing all the appointments.

Click2:

- Calls the ListOfAppointmentsClick (with required event arguments.

GetUserDetails:

- Makes a call to the Web API and returns information about the user.

Create new appointment

When the person goes into the booking section, they are presented with a list of the nearest helpers as well as their average rating. By clicking any one of the users, they are presented with more information about that helper as well as being offered the opportunity to see the users reviews.

OnCreate:

- Gets the list of nearby users (using GetListOfNearbyUsers) and converts this into a list of names which can be displayed on the ListView.
- For each of these names, the getAverageRating function is used to get the average rating of the user and display this alongside their name.

OnListItemClick:

- Start the helper details activity, passing the helperId of the particular item clicked as a parameter.

GetListOfNearbyUsers:

- Sends a request to the WebAPI to return the list of nearby users to the location of the elderlyId sent to the API.

GetAverageRating:

- Completes ReturnListOfAppointments for the helper id and for each appointment checks if the ratingId = 0 (it has no rating). If it has a rating, then it goes to the WebAPI, requests the rating and adds this to a counter.
- This is divided by the number of ratings found to find the average rating for the user.

ReturnListOfAppointments:

- Sends request to the WebAPI to return a list of appointments, based on the helperId and parses the JSON input into an array of appointments.

Book Appointment

The details which can be seen when clicking one of the users consist of everything from their name, to their contact number and address.

The user can also click to see reviews (this button is only enabled if the user has any reviews) or book an appointment with them.

OnCreate:

- Populates the details of the helper (including name, address, telephone number etc).
- Also, checks if there are reviews for the helper using ReviewsForUser. If there are, then the 'See reviews' button is enabled. Otherwise, it is hidden from the user.

SeeReviewsClick:

- Starts the see reviews activity.

BookAppointmentClick:

- Activity for setting the time to book the appointment is started.

ReviewsForUser:

- Checks if the response to WebAPI asking for list of reviews of user is empty or not.

GetUserDetails:

- Makes a call to the Web API and returns information about the user.

See all reviews of helper

By clicking the see all reviews button, the user can see a simple list of all the reviews of a user, with no additional clutter around the review.

The only place that the user can go from here is back to the previous screen, by using the back buttons on the android device.

	OnCreate: • Populates the ListView with the content from the array of reviews returned by GetListOfReviews (with parameter of the helperId) **GetListOfReviews:** • Sends request to the WebAPI with the helperId, and parses the JSON response into the array of reviews for that user.

Set time to book appointment

If they instead click to book an appointment with the helper, they are presented with a list of weeks which they can book appointments in. This interface is similar in style to the 'Set Timetable' interface, in this case with a few changes.

Firstly, the number of weeks they can book an appointment in, is reduced, from seven weeks to only the next four weeks. Additionally, the week will only appear if the helper has any time in that week. Otherwise, the week will simply not appear.

The user can click on any one of the weeks to move onto the next screen.

OnCreate:

- Populates the listview with the date of the weekbeginning date for the next several weeks in a readable way, first checking whether there are any appointments available in that week – using CheckIfAnyTimeInWeek.
    - To do this, it first gets the timetable of the user – using GetTimetableOfUser
    - Then, it checks if this week is one of the specific columns which are defined, if it is, then this week is used.
    - Otherwise, the default week is used.
    - This week is passed to the CheckIfAnyTimeInWeek function.

CheckIfAnyTimeInWeek:

- Returns false if and only if a checkIfAnyTimeInDay fails for all days in the week.

CheckIfAnyTimeInDay:

- Checks if there are any free times in the day – returns true if there are, false if not.

OnListItemClick:

- Starts the activity to choose the exact day to have the appointment, passing the weekbeginning date as a parameter and the weekId (the default week if it isn't a custom week in the timetable).

GetTimetableOfHelper:

- Sends a request to the WebAPI with the parameter of the id of the timetableId and returns the timetable of the user.

Once the user has selected a week, they can then select a day from that week to have an appointment. In the same way as the week list, this is presented in a simple list, with only days which have free times being available to be clicked on.

If any day is clicked, the user can then move onto the next screen, to select the time of the appointment.

OnCreate:

- Get the week to be displayed using the GetWeek.
- Populates the list view with all the days of the week – first checking if there is any time in that day before displaying it.
- Also checks if the day is before today before displaying it.

GetWeek:

- Sends request to the WebAPI and parses the JSON response into a week.

CheckIfTimeInDay:

- Given the day, checks if there is any free appointments in that day – returning a false if there are not, returning a true if there are.

OnListItemClick:

- Starts the activity of selecting the time of the appointment passing the DateTime of the selected day as well as the dayId.

In this screen, the user can select one of the available times to have the appointment, again being presented in a simple list.

OnCreate:

- Populated the list based on which appointments are available during the selected day (day got using the GetDay function).

GetDay:

- Sends request to the WebAPI and parses the JSON response into a day.

OnListItemClick:

- Starts the activity of the submission of the appointment screen, passing the exact date and time of the appointment as a parameter, as well as the helper and elderlyId.

Finally, when this is selected, the complete information about the appointment is displayed, with the user then being able to book the appointment by pressing the 'Confirm' button.

OnCreate:

- Populates the screen with the details of the helper, using the GetUserDetails function and the details of the appointment passed from the previous activity.

SubmitButtonClick:

- Creates an appointment object and POSTs this to the WebAPI.
- Displays a toast notification that the appointment has been created.
- Starts the Welcome screen on elderly with a sentBack parameter as true.
    - This then redirects the screen to the list of appointments with the welcome screen as the screen which is last in the stack of screens.

	○ Therefore, if the back button is pressed, the welcome screen is returned to. **GetUserDetails:** • Makes a call to the Web API and returns information about the user.

View appointments

If, instead of booking an appointment at the main screen, the user clicked list of appointments, the user would be presented with a screen similar to this. This screen, identically to the 'list of appointments' screen for the helpers, displays all the appointments the elderly person has. These are arranged with the upcoming appointments first, before going into past appointments, latest first. For each appointment, the helper name and time and date is listed.

By clicking any one of the appointments, more details about the appointment can be retrieved.

	OnCreate: • Get list of appointments. • Foreach of the appointments, get the name of the elderly person (using getUserDetails). • Populate the list view in the order of upcoming first (closest first), then going back in history **OnListItemClick:** • Call the activity to show more details about that appointment. **returnListOfAppointments:** • From the helperId, this makes a call to the REST api and returns the list of appointments. **getUserDetails:** • Makes a call to the Web API and returns information about the user.

See details of appointments

In this screen, as opposed to the helper equivalent, the only difference is the fact that the 'Set Rating' and 'Set Review' buttons are available to be pressed if the appointment has already occurred.

OnCreate:

- GetAppointmentDetails of appointment to show in more details.
- Populate the text views using details from GetUserDetails
- If in the past, ensure that cancel appointment is not enabled. If in the future, it should be enabled.

CancelAppointmentClick:

- Send the request to the Web API.

GetUserDetails:

- Send request to Web API and return the user information.

GetAppointmentDetails

- Send request to Web API and return the appointment.

SetRatingClick:

- Starts the rating activity.

SetReviewClick:

- Starts the review activity.

Set rating

If the 'Set Rating' button is pressed, the user is taken to this screen, which allows them to set (or edit an existing) a rating. They do that by pressing on the desired location on the star – half stars are also possible. Therefore, they are effectively rating on a scale of 0 to 10.

When they first come to the screen, if a rating has already been added, the stars are prepopulated with this rating, otherwise they all appear empty.

By pressing the submit button, the rating is added to the database. The success of this addition is displayed by a toast notification which informs the user of this success.

OnCreate:

- The view is initialised and the name of the helper populated (using the GetUserDetails function).
- The rating is also populated by the GetRating function. If it has no rating, then it is populated as 0 stars. Otherwise, it gets the rating as it was given.

SubmitRatingClick:

- It is checked whether a rating exists or not.
- If one already exists, the new rating is POSTed to the WebAPI so that it can be updated on the database.
- If not, then it is sent to the WebAPI as a rating to be added, alongside the appointmentId, so the rating information can be added to the appointments table.
- Finally, a toast notification is used to tell the user that the rating has been submitted.

GetRating:

- The rating (based on the ratingId) is retrieved from the database, using a request from the WebAPI.

getUserDetails:

- Makes a call to the Web API and returns information about the user.

Set review

If the 'Set Review' button is pressed, the user is taken to this screen, which allows them to set (or edit an existing) a review. They do that by typing their review on the expanding textview, which allows them to write very long reviews if they so desire.

When they first come to the screen, if a review has already been added, the textview are prepopulated with this review, otherwise it appears empty.

By pressing the submit button, the review is added to the database. The success of this addition is displayed by a toast notification which informs the user of this success.

OnCreate:

- The view is initialised and the name of the helper populated (using the GetUserDetails function).
- The review is also populated by the GetReview function. If it has no review, then the review box is empty. Otherwise, it gets the review as it was given.

SubmitReviewClicked:

- It is checked whether a review exists or not.
- If one already exists, the new review is POSTed to the WebAPI so that it can be updated on the database.
- If not, then it is sent to the WebAPI as a review to be added, alongside the appointmentId, so the review information can be added to the appointments table.
- Finally, a toast notification is used to tell the user that the review has been submitted.

GetReview:

- The review (based on the reviewId) is retrieved from the database, using a request from the WebAPI.

getUserDetails:

- Makes a call to the Web API and returns information about the user.

2.5 – SMS Server System

The SMS system relies upon the Twilio system which provides a phone number and an API through which text messages can be received and sent. In order to communicate with the Twilio REST API, the helper library is used with Python.

This has only one class, and consists of four functions:

1. “@app.route('/sms', methods = ['GET','POST'])”
    a. This deals with incoming SMS messages and simply passes them onto the .NET core Web Api as well as sending a generic response: 'Thanks for your request. Your appointment details will be sent to you shortly'
2. “@app.route('/sendSms', methods = ['POST'])”
    a. This deals with any messages to be sent to other number and passes them on, using the Twilio helper library, to be sent.
    b. (This receives the required data in JSON, which it uses the Requests library to deal with)
3. “@app.route('/verifyNumber', methods = ['POST'])”
    a. This deals with verifying a new phone number (so it can text the Twilio number and receive messages from it),
    b. It POSTs lots of information to the Twilio REST API and receives a validation code which is then returned to the Android application.
4. “if __name__ == '__main__':”
    a. This defines important things like the port number on which the API should be run.

Figure 14: Example of SMS Appointment Server in action (appointment allocated by .NET Core API)

2.6 – Server Side Management

2.6.1 – Traffic Management

In order to ensure the server can respond to multiple sources of traffic, attempting to communicate with different APIs, an Nginx server was set up, with the config file being setup to have listeners on a number of different ports which can then forward the traffic to the correct open ports from the correct applications. This ensures that traffic from existing Apache Web Servers (which run multiple websites), the .NET core WebAPI and the Python SMS API can be dealt with individually. Additionally, the port for the database traffic can be configured such that only traffic from the localhost will be allowed. This is an effective safety mechanism to ensure that the database cannot be easily accessed and SQL commands cannot simply be run and the data retrieved from the database. Therefore, the network traffic route is therefore hence:

Figure 15: Network traffic diagram of server

2.6.2 – Crontabs

The other important aspect on the server side was ensuring that all the various APIs and the database ran at the same time. In fact, normally, even running a single one of them would not occur at the same time normally, as they each took focus over a console window. Therefore, by creating a bash script which would execute each of them in the background was required. This would carry out the console commands that would be otherwise required to run the programs to be bypassed, as simply running the bash script would run all these commands, while allowing all the APIs to run in the background.

In case the server ever went down – the provider often complete a number of scheduled maintenance sessions, another measure was taken to ensure that the APIs would restart and there would be no problems. By defining a crontab, a specific script could be run at a particular time. Therefore using a crontab which would run at startup, the bash script to startup the APIs could be run at startup, therefore meaning that the service of the APIs would be up again, after any issues.

2.7 – Testing Strategy

The general aim of the testing is to test each part in turn, before testing the entirety of the system. Hence, the following test series were developed:

Test Series 1: Testing the Web API (Black-Box Testing)

In order to test the WebAPI, the requests from the Android App will be spoofed, using Postman, which allows one to send HTTP requests to each of the HTTP functions

The testing of the WebAPI gets further split into testing the following:

Test Series Number	Purpose of Test Series
1.1	Test SQL command class
1.2	Test DateTime Parsing
1.3	Test HTTP GET (get it to send commands to a computer which will process and log the requests)
1.4	Test Accounts controller
1.5	Test AccountsForHelper controller
1.6	Test AccountsForElderly controller
1.7	Test Appointments controller
1.8	Test Ratings controller
1.9	Test Reviews controller
1.10	Test Timetable controller

Test Series 2: Testing the Python SMS Server API (Black-Box Testing)

In order to test the Python SMS Server, the tests can be triggered by a combination of SMS messages from a phone and using Postman to trigger the API functions.

Test Series Number	Purpose of Test Series
2.1	Test GET on '/'
2.2	Test SMS receipt on /SMS
2.3	Test /sendSMS
2.4	Test /verifyNumber

Test Series 3: Testing the Android Application

In order to test the Android Application, the flow of the android app is tested, ensuring that buttons send the application to correct places.

Test Series Number	Purpose of Test Series
3.1	Test 'Login or Register' screen
3.2	Test 'Login' screen
3.3	Test 'Register' screen
3.4	Test 'List of Appointments' screen
3.5	Test 'Appointment Details' screen
3.6	Test 'Ratings submit' screen
3.7	Test 'Review submit' screen
3.8	Test 'Helper details' screen
3.9	Test 'See reviews' screen
3.10	Test 'Book appointment weeks' screen
3.11	Test 'Book appointment days' screen
3.12	Test 'Book appointment times' screen
3.13	Test 'Book details confirm' screen
3.14	Test 'Appointment details (helper)' screen
3.15	Test 'Set Timetable' screen
3.16	Test 'Set Timetable days' screen
3.17	Test 'Set Timetable times' screen
3.18	Test 'Welcome Elderly' screen
3.19	Test 'Welcome Helper' screen
3.20	Test 'Book Appointment – List of nearby helpers' screen

Test Series 4: Full system testing

Since much of this data relies upon data which is not static, it is hard to have tests which have set results, since the results will vary based on when the test is completed. Therefore, it is important that the results are generally reasonably easy to verify by using an application with a GUI to look at the raw data in the database and ensure that things like the timetable point to correct days. The application which is being used for this is HeidiSQL.

Test Series Number	**Purpose of Test Series**
4.1	Test registering
4.2	Test logging in
	ElderlyTests
4.4	Test list of appointments
4.5	Test one appointment
4.6	Test rating
4.7	Test review
	ElderlyTests.BookAppointment
4.9	Test list of helpers
4.10	Test helper details
4.11	Test book appointment
4.12	**General UI Testing**
4.13	Test SMS appointment creation system
	HelperTests
4.14	Test set timetable (and subsequently booking appointments)
4.17	**General UI Testing**

3 – Implementation

See the appendices for the code.

4 – Testing

4.1 – Test Series 1

Test Number	Purpose / Description	Test Data	Reason	Expected Result	Passed
1.1	**Test SQLCommand Class**				
1.1.1	Test general SELECT from database	**T1.1.1**	Normal Data	**E1.1.1**	Yes
1.1.2	Test incorrect syntax SELECT	**T1.1.2**	Erroneous Data	**E1.1.2**	Yes
1.1.3	Test SELECT of ungettable data in database	**T1.1.3**	Erroneous Data	**E1.1.2**	Yes
1.1.4	Test UPDATE query on database.	**T1.1.4**	Normal Data	**E1.1.3**	Yes
1.2	**Test DateTime Parsing**				
1.2.1	Test parsing with valid date and time	**T1.2.1**	Normal Data	**E1.2.1**	Yes
1.2.2	Test parsing with keyword 'tomorrow'	**T1.2.2**	Normal Data	**E1.2.2**	Yes
1.2.3	Test parsing text without time	**T1.2.3**	Boundary Data	**E1.2.3**	Yes
1.2.4	Test parsing text without date	**T1.2.4**	Boundary Data	**E1.2.4**	Yes
1.2.5	Test parsing text with correct date and time with American date format format	**T1.2.5**	Normal Data	**E1.2.5**	Yes
1.3	**Test HTTP GET**				
1.3.1	Test with correct HTTP GET	**T1.3.1**	Normal Data	**E1.3.1**	Yes
1.3.2	Test with incorrect URL HTTP GET	**T1.3.2**	Erroneous Data	**E1.3.2**	Yes

1.3.3	Test with incorrect URL HTTP GET	**T1.3.3**	Erroneous Data	**E1.3.3**	Yes
1.4	**Test Accounts controller**				
1.4.1	Test "Hello World" function	**T1.4.1**	Normal Data	**E1.4.1**	Yes
1.4.2	Test add new user function	**T1.4.2**	Normal Data	**E1.4.2**	Yes
1.4.3	Test verify new user function	**T1.4.3**	Normal Data	**E1.4.3**	Yes
1.5	**Test AccountsForHelper controller**				
1.5.1	Test "Hello World" function	**T1.5.1**	Normal Data	**E1.5.1**	Yes
1.5.2	Test GetInfo function with correct id	**T1.5.2**	Normal Data	**E1.5.2**	Yes
1.5.3	Test GetInfo using an incorrect id	**T1.5.3**	Erroneous Data	**E1.5.3**	Yes
1.5.4	Test AddNewUser with new unique information.	**T1.5.4**	Normal Data	**E1.5.4**	Yes
1.5.5	Test AddNewUser with email address which already exists in database.	**T1.5.5**	Erroneous Data	**E1.5.5**	Yes
1.5.6	Test Delete user with correct userId.	**T1.5.6**	Normal Data	**E1.5.6**	Yes
1.5.7	Test Delete userwith incorrect userid.	**T1.5.7**	Erroneous Data	**E1.5.7**	Yes
1.5.8	Test update user success (using correct id)	**T1.5.8**	Normal Data	**E1.5.8**	Yes
1.5.9	Test update user failure (using incorrect id)	**T1.5.9**	Erroneous Data	**E1.5.9**	Yes
1.5.10	Test verifyUser using a wrong password	**T1.5.10**	Normal Data	**E1.5.10**	Yes

1.5.11	Test verifyUser correct information (existing account)	**T1.5.11**	Normal Data	**E1.5.11**	Yes
1.5.12	Test GetListOfUsersByDistance using a correct user account	**T1.5.12**	Normal Data	**E1.5.12**	Yes
1.5.13	Test GetListOfUsersByDistance with an incorrect user id.	**T1.5.13**	Erroneous Data	**E1.5.13**	Yes
1.6	**Test AccountsForElderly controller**				
1.6.1	Test "Hello World" function	**T1.6.1**	Normal Data	**E1.6.1**	Yes
1.6.2	Test GetInfo with correct id	**T1.6.2**	Normal Data	**E1.6.2**	Yes
1.6.3	Test GetInfo with incorrect id	**T1.6.3**	Erroneous Data	**E1.6.3**	Yes
1.6.4	Test AddNewUser with unique information	**T1.6.4**	Normal Data	**E1.6.4**	Yes
1.6.5	Test AddNewUser with already existing information	**T1.6.5**	Erroneous Data	**E1.6.5**	Yes
1.6.6	Test Delete of valid user	**T1.6.6**	Normal Data	**E1.6.6**	Yes
1.6.7	Test Delete of invalid user	**T1.6.7**	Erroneous Data	**E1.6.7**	Yes
1.6.8	Test update success (correct id)	**T1.6.8**	Normal Data	**E1.6.8**	Yes
1.6.9	Test update failure (incorrect id)	**T1.6.9**	Erroneous Data	**E1.6.9**	Yes
1.6.10	Test verifyUser wrongPassword	**T1.6.10**	Normal Data	**E1.6.10**	Yes
1.6.11	Test verifyUser with correct information	**T1.6.11**	Normal Data	**E1.6.11**	Yes

1.7	**Test Appointments controller**				
1.7.1	Test create new SMS impossible because there are no available appointments.	**T1.7.1**	Boundary Data	**E1.7.1**	Yes
1.7.2	Test create new SMS with correct phone number and available appointment time.	**T1.7.2**	Normal Data	**E1.7.2**	Yes
1.7.3	Test create new SMS from unrecognised number	**T1.7.3**	Erroneous Data	**E1.7.3**	Yes
1.7.4	Test sendSMS function, sending SMS to verified number.	**T1.7.4**	Normal Data	**E1.7.4**	Yes
1.7.5	Test createNewAppointmentsForAndroidApp with valid appointment information, with availability at that time.	**T1.7.5**	Normal Data	**E1.7.5**	Yes
1.7.6	Test GetAppointmentInformation failure, using an incorrect appointment id.	**T1.7.6**	Erroneous Data	**E1.7.6**	Yes
1.7.7	Test GetAppointmentInformation success, using correct appointment id.	**T1.7.7**	Normal Data	**E1.7.7**	Yes
1.7.8	Test RemoveAppointment on an incorrect appointment id.	**T1.7.8**	Erroneous Data	**E1.7.8**	Yes
1.7.9	Test RemoveAppointment success, using a correct appointment id.	**T1.7.9**	Normal Data	**E1.7.9**	Yes
1.7.10	Test AppointmentsByElderly with an incorrect elderlyId.	**T1.7.10**	Erroneous Data	**E1.7.10**	Yes
1.7.11	Test AppointmentsByElderly with a correct elderlyId.	**T1.7.11**	Normal Data	**E1.7.11**	Yes
1.7.12	Test AppointmentsByHelper with an incorrect helperId.	**T1.7.12**	Erroneous Data	**E1.7.12**	Yes

1.7.13	Test AppointmentsByHelper correctId	**T1.7.13**	Normal Data	**E1.7.13**	Yes
1.8	**Test Ratings controller**				
1.8.1	Test GetAllRatingsForHelper with valid helperId.	**T1.8.1**	Normal Data	**E1.8.1**	Yes
1.8.2	Test GetAllRatingsForHelper with invalid helperId.	**T1.8.2**	Erroneous Data	**E1.8.2**	Yes
1.8.3	Test GetOneSpecificRating with correct ratingId.	**T1.8.3**	Normal Data	**E1.8.3**	Yes
1.8.4	Test GetOneSpecificRating with incorrectId.	**T1.8.4**	Erroneous Data	**E1.8.4**	Yes
1.8.5	Test AddARating with correct rating information	**T1.8.5**	Normal Data	**E1.8.5**	Yes
1.8.6	Test UpdateRating with correct rating information	**T1.8.6**	Normal Data	**E1.8.6**	Yes
1.8.7	Test DeleteRating with correct rating id.	**T1.8.7**	Normal Data	**E1.8.7**	Yes
1.9	**Test Reviews controller**				
1.9.1	Test GetAllReviewsForHelper with correct review id.	**T1.9.1**	Normal Data	**E1.9.1**	Yes
1.9.2	Test GetAllReviewsForHelper with correct helperId.	**T1.9.2**	Erroneous Data	**E1.9.2**	Yes
1.9.3	Test GetOneSpecificreview with correct review id.	**T1.9.3**	Normal Data	**E1.9.3**	Yes
1.9.4	Test GetOneSpecificReview with incorrect reviewId.	**T1.9.4**	Erroneous Data	**E1.9.4**	Yes
1.9.5	Test AddAReview with valid review.	**T1.9.5**	Normal Data	**E1.9.5**	Yes

1.9.6	Test UpdateReview with valid review.	**T1.9.6**	Normal Data	**E1.9.6**	Yes
1.9.7	Test DeleteReview with valid review id.	**T1.9.7**	Normal Data	**E1.9.7**	Yes
1.10	**Test Timetable controller**				
1.10.1	Test AddTimetable with valid timetable.	**T1.10.1**	Normal Data	**E1.10.1**	Yes
1.10.2	Test GetTimetable with valid timetableId.	**T1.10.2**	Normal Data	**E1.10.2**	Yes
1.10.3	Test GetTimetable with incorrect timetableId.	**T1.10.3**	Erroneous Data	**E1.10.3**	Yes
1.10.4	test getWeek fail with incorrect weekId.	**T1.10.4**	Erroneous Data	**E1.10.4**	Yes
1.10.5	Test getWeek success with correct weekId.	**T1.10.5**	Normal Data	**E1.10.5**	Yes
1.10.6	Test GetDay success with correct dayId.	**T1.10.6**	Normal Data	**E1.10.6**	Yes
1.10.7	Test GetDay fail with invalid dayId.	**T1.10.7**	Erroneous Data	**E1.10.7**	Yes

4.2 – Test Series 2

Test Number	Purpose / Description	Test Data	Reason	Expected Result	Passed
2.1.1	Test GET on '/'	**T2.1.1**	Normal Data	**E2.1.1**	Yes
2.2.1	Test SMS receipt from validated number	**T2.2.1**	Normal Data	**E2.2.1**	Yes
2.2.2	Test SMS receipt from invalidated number	**T2.2.2**	Boundary Data	**E2.2.2**	Yes

2.3.1	Test /sendSMS with correct information and valid number.	**T2.3.1**	Normal Data	**E2.3.1**	Yes
2.3.2	Test /sendSMS to invalidated number	**T2.3.2**	Erroneous Data	**E2.3.2**	Yes
2.3.3	Test /sendSMS with no body	**T2.3.3**	Boundary Data	**E2.3.3**	Yes
2.3.4	Test /sendSMS with different phone format	**T2.3.4**	Boundary Data	**E2.3.4**	Yes
2.3.5	Test /sendSMS with incorrect phone number	**T2.3.5**	Erroneous Data	**E2.3.5**	Yes
2.3.6	Test /sendSMS without correct information	**T2.3.6**	Erroneous Data	**E2.3.6**	Yes
2.4.1	Test /verifyNumber using correct information	**T2.4.1**	Normal Data	**E2.4.1**	Yes
2.4.2	Test /verifyNumber without correct information	**T2.4.2**	Erroneous Data	**E2.4.2**	Yes

4.3 – Test Series 3

Test Number	Purpose / Description	Test Data	Reason	Expected Result	Passed
3.1	**Test 'Login or Register' screen**				

3.1.1	Test clicking login button	**T3.1.1**	Normal Data	**E3.1.1**	Yes
3.1.2	Test clicking register button	**T3.1.2**	Normal Data	**E3.1.2**	Yes
3.2	**Test 'Login' screen**				
3.2.1	Test clicking back button	**T3.2.1**	Normal Data	**E3.2.1**	Yes
3.2.2	Test registering with incomplete fields in registration screen.	**T3.2.2**	Erroneous Data	**E3.2.2**	Yes
3.3	**Test 'Register' screen**				
3.3.1	Test registering with incomplete fields in registration screen.	**T3.3.1**	Erroneous Data	**E3.3.1**	Yes
3.3.2	Test registering with incomplete fields in registration screen.	**T3.3.2**	Erroneous Data	**E3.3.2**	Yes
3.3.3	Test registering with incomplete fields in registration screen.	**T3.3.3**	Erroneous Data	**E3.3.3**	Yes
3.3.4	Test registering with incomplete fields in registration screen.	**T3.3.4**	Erroneous Data	**E3.3.4**	Yes
3.3.5	Test registering with incomplete fields in registration screen.	**T3.3.5**	Erroneous Data	**E3.3.5**	Yes
3.3.6	Test registering with incomplete fields in registration screen.	**T3.3.6**	Erroneous Data	**E3.3.6**	Yes
3.3.7	Test registering with incomplete fields in registration screen.	**T3.3.7**	Erroneous Data	**E3.3.7**	Yes
3.3.8	Test registering with incomplete fields in registration screen.	**T3.3.8**	Erroneous Data	**E3.3.8**	Yes

3.3.9	Test registering with incomplete fields in registration screen.	**T3.3.9**	Erroneous Data	**E3.3.9**	Yes
3.3.10	Test registering with incomplete fields in registration screen.	**T3.3.10**	Erroneous Data	**E3.3.10**	Yes
3.3.11	Test registering with passwords not matching	**T3.3.11**	Erroneous Data	**E3.3.11**	Yes
3.3.12	Test clicking back button	**T3.3.12**	Normal Data	**E3.3.12**	Yes
3.4	**Test 'List of Appointments' screen**				
3.4.1	Test clicking on an appointment	**T3.4.1**	Normal Data	**E3.4.1**	Yes
3.4.2	Test clicking back button	**T3.4.2**	Normal Data	**E3.4.2**	Yes
3.5	**Test 'Appointment Details' screen**				
3.5.1	Test clicking back button	**T3.5.1**	Normal Data	**E3.5.1**	Yes
3.5.2	Test clicking set rating button.	**T3.5.2**	Normal Data	**E3.5.2**	Yes
3.5.3	Test clicking set review button.	**T3.5.3**	Normal Data	**E3.5.3**	Yes
3.6	**Test 'Ratings submit' screen**				
3.6.1	Test clicking back button	**T3.6.1**	Normal Data	**E3.6.1**	Yes
3.7	**Test 'Review submit' screen**				
3.7.1	Test clicking back button	**T3.7.1**	Normal Data	**E3.7.1**	Yes
3.8	**Test 'Helper details' screen**				

3.8.1	Test clicking see reviews button.	**T3.8.1**	Normal Data	**E3.8.1**	Yes
3.8.2	Test clicking book appointment button.	**T3.8.2**	Normal Data	**E3.8.2**	Yes
3.9	**Test 'See reviews' screen**				
3.9.1	Test clicking back button	**T3.9.1**	Normal Data	**E3.9.1**	Yes
3.10	**Test 'Book appointment weeks' screen**				
3.10.1	Test clicking back button	**T3.10.1**	Normal Data	**E3.10.1**	Yes
3.10.2	Test clicking on week.	**T3.10.2**	Normal Data	**E3.10.2**	Yes
3.11	**Test 'Book appointment days' screen**				
3.11.1	Test clicking back button	**T3.11.1**	Normal Data	**E3.11.1**	Yes
3.11.2	Test clicking on day.	**T3.11.2**	Normal Data	**E3.11.2**	Yes
3.12	**Test 'Book appointment times' screen**				
3.12.1	Test clicking back button	**T3.12.1**	Normal Data	**E3.12.1**	Yes
3.12.2	Test clicking on a timeslot.	**T3.12.2**	Normal Data	**E3.12.2**	Yes
3.13	**Test 'Book details confirm' screen**				
3.13.1	Test back button	**T3.13.1**	Normal Data	**E3.13.1**	Yes
3.14	**Test 'Appointment details confirm' screen**				
3.14.1	Test back button	**T3.14.1**	Normal Data	**E3.14.1**	Yes

3.15	**Test 'Set timetable' screen**				
3.15.1	Test back button	**T3.15.1**	Normal Data	**E3.15.1**	Yes
3.15.2	Test clicking on a week.	**T3.15.2**	Normal Data	**E3.15.2**	Yes
3.16	**Test 'Set timetable days' screen**				
3.16.1	Test back button	**T3.16.1**	Normal Data	**E3.16.1**	Yes
3.16.2	Test clicking on day (for appointment to be created).	**T3.16.2**	Normal Data	**E3.16.2**	Yes
3.17	**Test 'Set Timetable times' screen**				
3.17.1	Test clicking back button	**T3.17.1**	Normal Data	**E3.17.1**	Yes
3.18	**Test 'Welcome Elderly' screen**				
3.18.1	Test clicking list of appointments button.	**T3.18.1**	Normal Data	**E3.18.1**	Yes
3.18.2	Test clicking book appointment button.	**T3.18.2**	Normal Data	**E3.18.2**	Yes
3.19	**Test 'Welcome Helper' screen**				
3.19.1	Test clicking set timetable button.	**T3.19.1**	Normal Data	**E3.19.1**	Yes
3.19.2	Test clicking list of appointments button.	**T3.19.2**	Normal Data	**E3.19.2**	Yes
3.20	**Test 'Book Appointment – List of nearby helpers' screen**				
3.20.1	Test clicking helper on list of helpers screen.	**T3.20.1**	Normal Data	**E3.20.1**	Yes

3.20.2	Test clicking back button.	**T3.20.2**	Normal Data	**E3.20.2**	Yes

4.4 – Test Series 4

Test Number	Purpose / Description	Test Data	Reason	Expected Result	Passed
4.1	**Test Registering**				
4.1.1	Test registering (using correct details) and ensure that Twilio verification can occur successfully.	**T4.1.1**	Normal Data	**E4.1.1**	Yes
4.1.2	Test registering without a valid internet connection.	**T4.1.2**	Boundary Data	**E4.1.2**	Yes
4.2	**Test logging in**				
4.2.1	Test logging in without a valid internet connection.	**T4.2.1**	Boundary Data	**E4.2.1**	Yes
4.2.2	Test logging in without a correct password.	**T4.2.2**	Boundary Data	**E4.2.2**	Yes
4.2.3	Test logging in with a correct username and password.	**T4.2.3**	Normal Data	**E4.2.3**	Yes
4.2.4	Test logging in on a device which has previously logged in – i.e. test whether it autologs in.	**T4.2.4**	Normal Data	**E4.2.4**	Yes
4.4	**Test list of appointments**				
4.4.1	Test viewing all appointments and verify that they all exist and are ordered correctly.	**T4.4.1**	Normal Data	**E4.4.1**	Yes
4.5	**Test one appointment**				
4.5.1	Test viewing more information about one appointment and verify that information about the appointment is correct.	**T4.5.1**	Normal Data	**E4.5.1**	Yes
4.5.2	Test viewing an appointment which has taken place in the past and verify that you are able to set reviews and	**T4.5.2**	Normal Data	**E4.5.2**	Yes

	ratings and are unable to cancel the appointment.				
4.5.3	Test viewing an upcoming appointment and verify the ability to cancel the appointment. Also, ensure that it is impossible to set reviews and ratings.	**T4.5.3**	Normal Data	**E4.5.3**	Yes
4.6	**Test rating**				
4.6.1	Test setting a rating and ensure that the updating of the rating works through a shift in the average rating as well as through the rating existing when it is attempted to be changed again.	**T4.6.1**	Normal Data	**E4.6.1**	Yes
4.7	**Test review**				
4.7.1	Test setting a review for an appointment and ensure that the review is added by checking the reviews of the helper in the book appointment system.	**T4.7.1**	Normal Data	**E4.7.1**	Yes
4.9	**Test list of helpers**				
4.9.1	Test viewing a list of nearby helpers on an account with a faulty postcode.	**T4.9.1**	Erroneous Data	**E4.9.1**	Yes
4.9.2	Test viewing a list of nearby helpers (in the book appointment system) on an account with a valid postcode and verify the list is correct.	**T4.9.2**	Normal Data	**E4.9.2**	Yes
4.10	**Test helper details**				
4.10.1	Test viewing a single helper (from the list of nearby helpers) and ensure the data is valid about the helper.	**T4.10.1**	Normal Data	**E4.10.1**	Yes
4.10.2	Test see reviews about a single helper and ensure the reviews are valid.	**T4.10.2**	Normal Data	**E4.10.2**	Yes
4.11	**Test book appointment**				

4.11.1	Test the ability to only book appointments in the future – i.e. attempt to book an appointment in the past.	**T4.11.1**	Erroneous Data	**E4.11.1**	Yes
4.11.2	Try to book appointment on a day with no available appointments (as set from set timetable side on helper).	**T4.11.2**	Boundary Data	**E4.11.2**	Yes
4.11.3	Test successfully booking an appointment and verifying that is shows on list of appointments.	**T4.11.3**	Normal Data	**E4.11.3**	Yes
4.11.4	Test book an appointment and then cancel it (from the list of appointments), and check it gets cancelled.	**T4.11.4**	Normal Data	**E4.11.4**	Yes
4.11.5	Test cancelling an appointment from list of appointments and check it no longer appears on list of appointments when the list is refreshed.	**T4.11.5**	Normal Data	**E4.11.5**	Yes
4.13	**Test SMS appointment creation system**				
4.13.1	Test sending a text with correct information and ensure that an appointment is created and the details of this are received by text.	**T4.13.1**	Normal Data	**E4.13.1**	Yes
4.13.2	Test sending a text message with incorrect information (i.e. no valid date and time) and ensure that the correct error text message is received.	**T4.13.2**	Erroneous Data	**E4.13.2**	Yes
4.13.3	Test sending an SMS to create an appointment when no helpers are free – therefore testing whether the correct 'no helpers could be found' error text message is received.	**T4.13.3**	Boundary Data	**E4.13.3**	Yes
4.14	**Test set timetable (and subsequently booking appointments)**				

4.14.1	Test setting timetable and see if changes by reloading the same day and seeing if changes prevail.	**T4.14.1**	Normal Data	**E4.14.1**	Yes
4.14.2	Test setting a helper's timetable and see if you can book an appointment from an helper account and check the available times are the same as those set.	**T4.14.2**	Normal Data	**E4.14.2**	Yes

4.12 and 4.17

In order to attempt to complete test 4.12 and 4.17, the applications were placed into the hands of two random users (different to the end users), with a bare minimum explanation to see if they could complete the following tasks (determined as they tested all the important features of the application and would be enough for them to successfully have an appointment):

Elderly App

1. Create an account
2. Book an appointment with a nearby helper tomorrow after looking at the helper's rating and reviews.
3. Cancel the appointment.
4. Book another appointment tomorrow.
5. (At this point, the raw data was altered by changing the data in the database (this is not accessable to the user) and the appointment changed to be in the past)
6. Rate the helper.
7. Review the helper.
8. Book an appointment with an SMS message for two days time at 12pm.

Helper App

1. Create an account
2. Set the timetable to reasonable hours for the next two weeks.
3. (At this point an appointment was created with the user using another device)
4. Cancel the newly created appointment.

Both test users managed to complete all tasks with no additional assistance, so the tests can be deemed to be successful.

5 – Evaluation

Objective	Objective met?	Details
1 - Input		
There must be an Android app, which works with all Android phones with software later than Android Ice Cream Sandwich (4.0).	Yes	Though the application has only been tested to be working above Android 5.0 (due to a lack of test devices), it has been simulated at lower device versions (including 4.0) and has been working.
The elderly person should also be able to create an appointment with an SMS messages to a specified phone number.	Yes	Tests 4.13 show that this system is working.
The SMS should only require the user to send a Date and Time and the information about the appointment should be communicated directly back to the user over SMS.	Yes	Tests 4.13 show that the system works when a date and time is sent over SMS and successfully returns errors when not all data is present. 4.13.1 shows the receipt of the information of an appointment.
To log onto the Android application, the system should remember the password of the user once they have logged in once, thereby allowing them to not need to remember the password.	Yes	Test 4.2.4 shows that the autologin system ensures that users do not need to log onto the system multiple times on the same device.
2 - User Interface (on all systems)		
The screens should be as simple as possible, removing unnecessary information to other screens if necessary.	Yes	Test 4.12 shows that the interface is very simple as a person with little instruction is able to use it. Additionally, having given it to an elderly neighbour at a few points throughout the development, he now

		believes it is at a point at which anybody can use the system with little to no training.
As a maximum, there should be two possible (very different) paths to follow from every screen, with the difference in what each button does being marked clearly.	Yes	Looking at the design shows that there are never more than two different paths through the application at one time – with the exception of ListViews, where each lead to the same path, just with different data.
All text should be large, above 12pt, to ensure that the elderly person can see it easily, with worse eye-sight.	Yes	Looking at the code, it is clear to see that all text is at least of size 12pt.
There should be a screen where the user will be able to scroll through the nearby teenagers who can provide tech support.	Yes	This is the 'Book Appointment' screen, shown in section 2.4.3 and tested in 4.9.2.
This screen should be organised by how near the helper is, therefore meaning that the helper is unlikely to have to travel from very far.	Yes	This is tested and shown to be working in test 4.9.2.
For each teenager, the elderly person should be able to see their average rating (if they have been rated), location and all of their reviews.	Yes	The average rating is visible in the 'Book Appointment' screen (2.4.3), while the 'See Reviews' screen is described in 2.4.3 and tested in 4.10.2
The user should be able to look at the various appointments a teenager has free, allowing them to choose a time which would suit them.	Yes	The user can view the times that the helpers are free over the next four weeks, therefore being able to choose whichever one of them suits the elderly person best.
They should be able to book an appointment for any available time at least within the next two weeks.	Yes	They can pick an appointment within the next four weeks.

The user should be able to see the list of upcoming and past appointments, seeing the details about the appointment.	Yes	They can see all this information in the 'List of Appointments' screen, described in 2.4.2 and tested in 4.4.1.
For appointments in the past, the user should be able to rate and review the helper, therefore helping future elderly people to decide whether the helper would be able to help them.	Yes	Designed in 2.4.3 and tested in 4.5.2 vs 4.5.3.
For appointments in the future, the user should be able to cancel the appointment, if they are no longer free.	Yes	This was designed in 2.4 and tested in 4.5.4
3 – Teenager's Input		
There must be an Android app through which the person should be able to interact with the system.	Yes	There is an android app which has been designed in section 2.4 and tested in section 4.
There should be a system to ensure to remember the password for the android application, so that they would not have to remember the password every time they log onto the application.	Yes	Test 4.2.4 shows that the autologin system ensures that users do not need to log onto the system multiple times on the same device.
4 – Teenager's User Interface		
The user must be able to see everyone who has requested their help.	Yes	They can see all this information in the 'List of Appointments' screen, described in 2.4.2 and tested in 4.4.1.
They should be able to cancel any appointments which they are no longer able to be available for.	Yes	This was designed in 2.4 and tested in 4.5.4
The user should also be able to define when they are free	Yes	This ability was designed in 2.4.2 and tested in 4.14. The application allows

for the next six weeks, allowing them to set the times that they would be free for appointments.		the user to set their times that they are free for the next eight weeks, firstly choosing the week to edit, then the day and finally choosing the times that they are free for that day.
5 - The appointments should be two hours in length for all appointments.	Yes	As throughout the design and in the implementation, the appointments are designed to be two hours long, running between even hours.
6 - The payment should be defined as £20 per hour for the appointments, therefore being £20 if the appointment takes up less than half the appointment slot, or £40 otherwise.	Yes	The payment, while not being discussed throughout the app has been defined to be £20 per hour and this should be discussed in any user manual.
7 - The system should be secure, with passwords hashed to ensure that they are not easy to utilise in case the server is hacked.	Yes	The passwords are hashed with MD5, see section 2.3.3

5.1 – User Feedback

5.1.1 – Mrs Wilson (AgeUK)

Mrs Wilson, on being shown the completed system, found the system impressive, observing that it had not only met the specification that had been defined but most importantly that the system was very well designed, being aesthetically pleasing for both the elderly and the helpers.

In showing Mrs Wilson the system, I largely focused on the elderly side, as this was the part of the specification which she had helped to define. Upon picking up the application for the first time, she mentioned that the system was very easy to use, and could easily be used by any members of the elderly that she taught without too much trouble.

During the registration process, the only part which she found poorly documented and difficult to understand was the Twilio verification step, since if the elderly person left the code screen too quickly or failed to get the call the first time, then there was no method by which this could be resolved. She commented that it would be useful to have the opportunity to do this within the logged in section.

Another comment she made was the lack of a 'forgot my password' section which would be useful if an elderly person forgets their password. However, she particularly liked the autologin features (which we hadn't previously discussed) finding that this feature makes the lack of a 'forgot my password' section far more acceptable. She also commented that biometrics on newer phones could be another feature which could be integrated to further reduce the amount of work elderly people would have to do.

Within the logged in section, she again liked the simplicity, with each screen having a well-defined purpose. She was somewhat surprised that I had managed to keep to her guideline of a maximum of two separate things on a single screen. She particularly enjoyed the booking system which she found intuitive – with plenty of messages to the user wherever necessary but offering a simple process.

She commented on a problem that an elderly person could book an appointment with someone who was miles away. She recommended therefore, at the very least, the distance could be displayed for the elderly person to see – since this is already calculated in the backend.

Another minor issue she had found was the time delay in finding the nearest helpers was rather longer than she would have liked and therefore, if possible, this could be reduced. However, for the rest of the interface, she found it to be snappy and without much latency.

A final discussion we had was based on the use of the word elderly throughout the application. She commented that it would likely disinsentivise users from using the application as people are likely to prefer not to think they are elderly. Though I had considered and dismissed the problem having talked to my elderly neighbours, Mrs Wilson felt it was important that I reassess this. She also mentioned that the application had also reached a point that it was not only useful for the elderly, instead would be useful for a much wider range of people. Therefore, by simply changing the terminology, for example using 'client' or 'customer' instead of elderly, I could attract a larger audience, as well as making my current audience able to use the application with less uneasiness.

Overwhelmingly however, the feedback was positive and she was excited for the possibilities for the application.

5.1.2 – Phil Fernandes

Phil largely found that the system was exceedingly effective and easily met his requirements. He commented on the ease of use of the system, saying that anyone could definitely use the helper side of the software, which, though not a particularly important a thing (this was not a specification point), is always a positive.

He says the interface is intuitive and easy to use, as long as there is a little bit of understanding with regards to what the application does. Therefore, if the application were to be more broadly launched, it would be useful to produce a 'User Manual' or something to this effect which could briefly show how to use the application.

He mentioned that an area where he anticipated that problems may have arisen from, the setting of the timetable and booking of timetables has an effective interface, by allowing the users to first choose the week, then the day and then edit the hour or book an appointment at a particular hour.

He however suggested that this could be further improved, for example by bringing in a calendar-esque interface. Though perhaps harder to produce, the calendar approach would allow for more effective usage of the application.

He also mentioned that the aspect of no maximum distance for each helper user being unhelpful and something which should be added if there were to be time.

5.2 – Analysis

5.2.1 – Learnability

Throughout the process, the ease of use of the application by the elderly has been foremost in my mind, with a number of tests throughout the programming, to ensure that the application could be used by a random person without any instruction. This culminated in test 4.12, which shows that someone, given little to no instruction, can perform a set of tasks with the application. This, alongside the comments by Mrs Wilson and Phil seem to vindicate this effort as they all mention that they find it very easy to use and therefore usable by all elderly people without any issues.

However, even given this, it would almost certainly be useful, before the product were to be released, to produce a User Manual to outline the various things the application can do and outline how they can be done.

The User Manual would also offer an opportunity to highlight two areas where the least information is provided to users, payment and the SMS messaging system. At no point in the application is payment discussed, despite a fixed fee system having been decided in the Analysis. In the application however, there is no correct page where this information would fit. Additionally, the exact format for SMS messages to be sent by the elderly person to the server and the phone number to send them to is not properly defined and could be easily made mistakes on. Though not a particularly challenging issue, it would be an important thing to mention in any user manual.

5.2.2 – Latency

Mrs Wilson complained that there was a time lag in the transition between two screens, the login screen and the welcome screen (for the elderly) and between the welcome screen and 'Book Appointment' (i.e. the list of nearby helpers) screen.

For the login screen, the time lag is almost certainly down to the time taken by the MD5 hashing algorithm. The MD5 algorithms gets its impressive security by being vastly complex an algorithm and therefore this conversion inevitably will take some time. Additionally, since this is only likely to be completed once every time the app is used, this seems acceptable.

For the book appointment screen, it seems far more a consequence of the speed of the Google Maps API, since it returns lots of information, including the route between the two points, traffic, etc. Since all I need in fact, is a longitude and latitude for each point, and then the distance can be calculated using Haversine Formula, this should be saved in the database and calculated upon registration. Alternatively, another API, such as the OpenMaps API could be used instead for simpler results. Another important matter to consider when considering the solution is that if this were to be scaled up, there might be problems due to the number of calls made to the Google Maps API – which limits this to ten per second, unless you pay for the paid account.

5.2.3 – Improvements

In order to improve the system, it would be better to store the location of each helper as a longitude and latitude in the database as this would allow for a much faster sorting of the system.

Additionally, it would be better to make use of a more significant Natural Language processing system, since the one I am currently using is very specific as to the information it requires to work. For example, if a year is not specified when the entirety of a date is specified then it assumes the date is today's date, rather than parsing the rest of the date and assuming the year is this year (the far more logical assumption). Additionally, it relies upon the user abiding by a specific timing format (HH:MM). If I had used a library like IBM Watson's inbuilt NLP API, it would be able to deal with all formats, in addition to dealing with things like 'quarter past' etc which are substituted instead of the time. It would also continue to work even if there are a few spelling mistakes, something which currently prevents the system from parsing the date.

Finally, throughout the system, references to the elderly should be removed, instead being replace by 'client' or 'customer', thereby attracting a wider target market and ensuring those who may still have technical problems, but do not regard themselves as elderly, could still use the application.

5.2.4 – Extensions

iOS Applications

The first extension would be to ensure that the application could be produced for iOS to ensure that elderly people and teenagers with iOS phones and tablets could also use the system.

In fact, since the Android application was produced using Xamarin, this shouldn't pose much of a challenge as the vast majority of the code could be reused if the application for iOS were to be coded in Xamarin iOS.

Tablet Specific Application

Another extension would be to produce a tablet specific application, especially catering to cheap Android tablets, which have proved very successful with the elderly people, according to Mrs Wilson. Though the application as it currently stands can be used on a tablet (and has been informally tested

on a Nexus 7 tablet), the system should be redesigned to cater for the larger screen and the added functionality which can be gotten from this.

Variable time-slot appointments

As mentioned by Phil, another change that could be made is introducing variable length appointments or offering a number of possible lengths. This would increase the efficiency of the process as the teenagers could help more elderly people in the same amount of time. This would be especially important when the elderly person knows the problem would be quick to solve but can't solve it themselves.

Maximum travelable distance for helpers

The current solution has no method to stop elderly people booking appointments with teenagers who live hours, or even further, away. Therefore, if a maximum distance decision could be added when the helper signs up for an account, this can be taken into account into whether to display their account in the 'Book an Appointment' view for a specific elderly person, by simply comparing this decided maximum distance with the distance between the helper and the elderly person.

Appendices

Appendix A – Web API Code

Backend

HttpCommandOut.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Net.Http;
using System.Net;
using System.IO;
namespace TestApi.Backend
{
    public class HTTPCommandOut
    {
        /// <summary>
        /// Completes a HTTP Get
        ///
        /// </summary>
        public HTTPCommandOut() { }
        /// <summary>
        /// Since it's asynchronous, it returns a task. However, in reality, the
only
        /// thing which is used from the task return is the result.
        /// </summary>
        /// <param name="url">This is the URL to complete the GET command
from</param>
        /// <returns>Returns a Task (of a stream). From here, the stream can be
created and the contents read using a StreamReader</returns>
        public async Task<Stream> get(string url)
        {

            using (var client = new HttpClient()) // The using ensures the
HTTPClient cannot be used outside of scope.
            {

                client.BaseAddress = new Uri(url); // Important to note that since
it is used directly as a URI, the url must be fully defined
                                                   // i.e. also including the
http://

                var response = await client.GetAsync(""); // Asynchronously calls
the task
                Stream stringResponse = await response.Content.ReadAsStreamAsync();
//Gets the response
                StreamReader x = new StreamReader(stringResponse); //
                return stringResponse; // Returns the task of the stream
```

```
            }
        }
    }
}
```

SQLStuff.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using MySql.Data.MySqlClient;
using MySql.Data.Common;
using MySql.Data.Types;

namespace TestApi.Sql
{
    public class SQLStuffOnTopOfController : Controller
    {
        public const bool server = false; // on the test machine, this is set to
false to set the url to point to the server. Otherwise, for speed, it should point
to localhost


        /// <summary>
        /// Returns a command - follows the following process:
        ///     (1) Creates a new connection with the db.
        ///     (2) Completes the SQL query
        ///     (3) Returns the value got from the MySQL server
        /// </summary>
        /// <param name="isResultExpected">true if a result is expected, false if
not</param>
        /// <param name="command">String value of the query</param>
        /// <param name="numberExpectedPerRow">Defaults at one, but otherwise knows
how to split the response</param>
        /// <returns></returns>
        public static string sqlCommand(bool isResultExpected, string command, int
numberExpectedPerRow = 1)
        {
            MySqlConnection dbConnection = setupConnection(); // Setup Connection
and return MySqlConnection object

            MySqlCommand commandForSql = new MySqlCommand(command, dbConnection);
            if(isResultExpected)
            {
                MySqlDataReader a = commandForSql.ExecuteReader();
                string toReturn = String.Empty;
                while(a.Read())
                {
                    for (int i = 0; i < numberExpectedPerRow; i++)
```

```
                    {
                        string ab = a.GetDataTypeName(i); //In case we are
expecting a DateTime - deal with this
                        if (ab == "DATETIME")
                        {
                            toReturn += a.GetDateTime(i).ToString() + "#"; // Gets
the DateTime as a DateTimeObject which can definitely be parsed by a
DateTime.Parse()
                        }
                        else
                            toReturn += a.GetValue(i) + "#";
                    }
                    //# is used to seperate items in the same line
                    toReturn += "\n";
                    //NewLine ('\n') is used to seperate different lines
                }
                a.Close(); //For stability, the DataReader is closed
                closeConnection(dbConnection); // Connection is closed
                return toReturn;
            }
            else
            {
                commandForSql.ExecuteScalar(); // just execute the command
            }
            closeConnection(dbConnection); // Close the connection
            return String.Empty; // if you haven't already returned the string,
then return an empty string
        }

        public static void closeConnection(MySqlConnection a)
        {
            a.Close();
        }

        /// <summary>
        /// Setup connection
        ///     (1) Defines the correct string for the connection
        ///     (2) Sets up the connection and returns the connection object.
        /// </summary>
        public static MySqlConnection setupConnection()
        {
            string connectionString = String.Empty;
            if(server)
                connectionString = "server=localhost;user
id=ashwin;password=Poonam123;port=3306;database=ComputingProject;SslMode=None;Conve
rt Zero DateTime=True;Allow Zero DateTime=True";
            else
                connectionString = "server=178.62.87.28;user
id=ashwin;password=Poonam123;port=3306;database=ComputingProject;SslMode=None;Conve
rt Zero DateTime=True;Allow Zero DateTime=True";
            MySqlConnection dbConnection = new MySqlConnection(connectionString);
```

```
            dbConnection.Open();
            return dbConnection;
        }
    }
}
```

SmsStuff.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Net;
using System.Net.Http;
using System.IO;
using TestApi.Backend;
using TestApi.Controllers;
using Newtonsoft.Json;
using TestApi.Types;
using System.Text;

namespace TestApi.Backend
{
    public class sms
    {
        /// <summary>
        /// Send SMS to the number listed.
        /// Method is asynchronous as it calls assynchronous methods in HTTPClient
methods.
        /// </summary>
        /// <param name="from">The phone number to send the SMS to</param>
        /// <param name="body">The contents of the message to send</param>
        async void sendSMS(string from, string body)
        {
            Controllers.sms s = new Controllers.sms(); // Creates an SMS object
            s.Body = body;
            s.From = from;
            string json = JsonConvert.SerializeObject(s); // Converts the object
into JSON
            string url = "http://178.62.87.28:700/sendSms"; // Defines the URL for
the POST call to the SMS server.
            var httpContent = new StringContent(json, Encoding.UTF8,
"application/json"); // Adds the correct header to the JSON
            using (var httpClient = new HttpClient())
            {
                var httpResponse = await httpClient.PostAsync(url, httpContent);

                if (httpResponse.Content != null) // if there is a response - deal
with it.
                {
                    var responseContent = await
httpResponse.Content.ReadAsStringAsync(); // gets the value of the response
```

```
                }
            }

        }


    }
}
```

DateTimeRoutines.cs

```
//*****************************************************************************
***********
//Author: Sergey Stoyan, CliverSoft.com
//        http://cliversoft.com
//        stoyan@cliversoft.com
//        sergey.stoyan@gmail.com
//        27 February 2007
//*****************************************************************************
***********
using System;
using System.Collections.Generic;
using System.Text;
using System.IO;
using System.Text.RegularExpressions;
using System.Threading;
using System.Collections;
/// <summary>
/// API which is used to help to parse the SMS received by the user and therefore
returns a date
/// and time which can be used to help to create an appointment.
/// </summary>
namespace TestApi.Backend
{
    /// <summary>
    /// Miscellaneous and parsing methods for DateTime
    /// </summary>
    public static class DateTimeRoutines
    {
        #region miscellaneous methods

        /// <summary>
        /// Amount of seconds elapsed between 1970-01-01 00:00:00 and the date-
time.
        /// </summary>
        /// <param name="date_time">date-time</param>
        /// <returns>seconds</returns>
        public static uint GetSecondsSinceUnixEpoch(this DateTime date_time)
        {
            TimeSpan t = date_time - new DateTime(1970, 1, 1);
            int ss = (int)t.TotalSeconds;
```

```
    if (ss < 0)
        return 0;
    return (uint)ss;
}

#endregion

#region parsing definitions

/// <summary>
/// Defines a substring where date-time was found and result of conversion
/// </summary>
public class ParsedDateTime
{
    /// <summary>
    /// Index of first char of a date substring found in the string
    /// </summary>
    readonly public int IndexOfDate = -1;
    /// <summary>
    /// Length a date substring found in the string
    /// </summary>
    readonly public int LengthOfDate = -1;
    /// <summary>
    /// Index of first char of a time substring found in the string
    /// </summary>
    readonly public int IndexOfTime = -1;
    /// <summary>
    /// Length of a time substring found in the string
    /// </summary>
    readonly public int LengthOfTime = -1;
    /// <summary>
    /// DateTime found in the string
    /// </summary>
    readonly public DateTime DateTime;
    /// <summary>
    /// True if a date was found within the string
    /// </summary>
    readonly public bool IsDateFound;
    /// <summary>
    /// True if a time was found within the string
    /// </summary>
    readonly public bool IsTimeFound;
    /// <summary>
    /// UTC offset if it was found within the string
    /// </summary>
    readonly public TimeSpan UtcOffset;
    /// <summary>
    /// True if UTC offset was found in the string
    /// </summary>
    readonly public bool IsUtcOffsetFound;
    /// <summary>
```

```
            /// Utc gotten from DateTime if IsUtcOffsetFound is True
            /// </summary>
            public DateTime UtcDateTime;

            internal ParsedDateTime(int index_of_date, int length_of_date, int
index_of_time, int length_of_time, DateTime date_time)
            {
                IndexOfDate = index_of_date;
                LengthOfDate = length_of_date;
                IndexOfTime = index_of_time;
                LengthOfTime = length_of_time;
                DateTime = date_time;
                IsDateFound = index_of_date > -1;
                IsTimeFound = index_of_time > -1;
                UtcOffset = new TimeSpan(25, 0, 0);
                IsUtcOffsetFound = false;
                UtcDateTime = new DateTime(1, 1, 1);
            }

            internal ParsedDateTime(int index_of_date, int length_of_date, int
index_of_time, int length_of_time, DateTime date_time, TimeSpan utc_offset)
            {
                IndexOfDate = index_of_date;
                LengthOfDate = length_of_date;
                IndexOfTime = index_of_time;
                LengthOfTime = length_of_time;
                DateTime = date_time;
                IsDateFound = index_of_date > -1;
                IsTimeFound = index_of_time > -1;
                UtcOffset = utc_offset;
                IsUtcOffsetFound = Math.Abs(utc_offset.TotalHours) < 12;
                if (!IsUtcOffsetFound)
                    UtcDateTime = new DateTime(1, 1, 1);
                else
                {
                    if (index_of_date < 0)//to avoid negative date exception when
date is undefined
                    {
                        TimeSpan ts = date_time.TimeOfDay + utc_offset;
                        if (ts < new TimeSpan(0))
                            UtcDateTime = new DateTime(1, 1, 2) + ts;
                        else
                            UtcDateTime = new DateTime(1, 1, 1) + ts;
                    }
                    else
                        UtcDateTime = date_time + utc_offset;
                }
            }
        }

        /// <summary>
```

```
        /// Date that is accepted in the following cases:
        /// - no date was parsed by TryParseDateOrTime();
        /// - no year was found by TryParseDate();
        /// It is ignored if DefaultDateIsNow = true was set after DefaultDate
        /// </summary>
        public static DateTime DefaultDate
        {
            set
            {
                _DefaultDate = value;
                DefaultDateIsNow = false;
            }
            get
            {
                if (DefaultDateIsNow)
                    return DateTime.Now;
                else
                    return _DefaultDate;
            }
        }
        static DateTime _DefaultDate = DateTime.Now;

        /// <summary>
        /// If true then DefaultDate property is ignored and DefaultDate is always
DateTime.Now
        /// </summary>
        public static bool DefaultDateIsNow = true;

        /// <summary>
        /// Defines default date-time format.
        /// </summary>
        public enum DateTimeFormat
        {
            /// <summary>
            /// month number goes before day number
            /// </summary>
            USA_DATE,
            /// <summary>
            /// day number goes before month number
            /// </summary>
            UK_DATE,
            ///// <summary>
            ///// time is specifed through AM or PM
            ///// </summary>
            //USA_TIME,
        }

        #endregion

        #region parsing derived methods for DateTime output
```

```
        /// <summary>
        /// Tries to find date and time within the passed string and return it as
DateTime structure.
        /// </summary>
        /// <param name="str">string that contains date and/or time</param>
        /// <param name="default_format">format to be used preferably in ambivalent
instances</param>
        /// <param name="date_time">parsed date-time output</param>
        /// <returns>true if both date and time were found, else false</returns>
        static public bool TryParseDateTime(this string str, DateTimeFormat
default_format, out DateTime date_time)
        {
            ParsedDateTime parsed_date_time;
            if (!TryParseDateTime(str, default_format, out parsed_date_time))
            {
                date_time = new DateTime(1, 1, 1);
                return false;
            }
            date_time = parsed_date_time.DateTime;
            return true;
        }

        /// <summary>
        /// Tries to find date and/or time within the passed string and return it
as DateTime structure.
        /// If only date was found, time in the returned DateTime is always 0:0:0.
        /// If only time was found, date in the returned DateTime is DefaultDate.
        /// </summary>
        /// <param name="str">string that contains date and(or) time</param>
        /// <param name="default_format">format to be used preferably in ambivalent
instances</param>
        /// <param name="date_time">parsed date-time output</param>
        /// <returns>true if date and/or time was found, else false</returns>
        static public bool TryParseDateOrTime(this string str, DateTimeFormat
default_format, out DateTime date_time)
        {
            ParsedDateTime parsed_date_time;
            if (!TryParseDateOrTime(str, default_format, out parsed_date_time))
            {
                date_time = new DateTime(1, 1, 1);
                return false;
            }
            date_time = parsed_date_time.DateTime;
            return true;
        }

        /// <summary>
        /// Tries to find time within the passed string and return it as DateTime
structure.
        /// It recognizes only time while ignoring date, so date in the returned
DateTime is always 1/1/1.
```

```
        /// </summary>
        /// <param name="str">string that contains time</param>
        /// <param name="default_format">format to be used preferably in ambivalent
instances</param>
        /// <param name="time">parsed time output</param>
        /// <returns>true if time was found, else false</returns>
        public static bool TryParseTime(this string str, DateTimeFormat
default_format, out DateTime time)
        {
            ParsedDateTime parsed_time;
            if (!TryParseTime(str, default_format, out parsed_time, null))
            {
                time = new DateTime(1, 1, 1);
                return false;
            }
            time = parsed_time.DateTime;
            return true;
        }

        /// <summary>
        /// Tries to find date within the passed string and return it as DateTime
structure.
        /// It recognizes only date while ignoring time, so time in the returned
DateTime is always 0:0:0.
        /// If year of the date was not found then it accepts the current year.
        /// </summary>
        /// <param name="str">string that contains date</param>
        /// <param name="default_format">format to be used preferably in ambivalent
instances</param>
        /// <param name="date">parsed date output</param>
        /// <returns>true if date was found, else false</returns>
        static public bool TryParseDate(this string str, DateTimeFormat
default_format, out DateTime date)
        {
            ParsedDateTime parsed_date;
            if (!TryParseDate(str, default_format, out parsed_date))
            {
                date = new DateTime(1, 1, 1);
                return false;
            }
            date = parsed_date.DateTime;
            return true;
        }

        #endregion

        #region parsing derived methods for ParsedDateTime output

        /// <summary>
        /// Tries to find date and time within the passed string and return it as
ParsedDateTime object.
```

```
        /// </summary>
        /// <param name="str">string that contains date-time</param>
        /// <param name="default_format">format to be used preferably in ambivalent
instances</param>
        /// <param name="parsed_date_time">parsed date-time output</param>
        /// <returns>true if both date and time were found, else false</returns>
        static public bool TryParseDateTime(this string str, DateTimeFormat
default_format, out ParsedDateTime parsed_date_time)
        {
            if (DateTimeRoutines.TryParseDateOrTime(str, default_format, out
parsed_date_time)
                && parsed_date_time.IsDateFound
                && parsed_date_time.IsTimeFound
                )
                return true;

            parsed_date_time = null;
            return false;
        }

        /// <summary>
        /// Tries to find time within the passed string and return it as
ParsedDateTime object.
        /// It recognizes only time while ignoring date, so date in the returned
ParsedDateTime is always 1/1/1
        /// </summary>
        /// <param name="str">string that contains date-time</param>
        /// <param name="default_format">format to be used preferably in ambivalent
instances</param>
        /// <param name="parsed_time">parsed date-time output</param>
        /// <returns>true if time was found, else false</returns>
        static public bool TryParseTime(this string str, DateTimeFormat
default_format, out ParsedDateTime parsed_time)
        {
            return TryParseTime(str, default_format, out parsed_time, null);
        }

        /// <summary>
        /// Tries to find date and/or time within the passed string and return it
as ParsedDateTime object.
        /// If only date was found, time in the returned ParsedDateTime is always
0:0:0.
        /// If only time was found, date in the returned ParsedDateTime is
DefaultDate.
        /// </summary>
        /// <param name="str">string that contains date-time</param>
        /// <param name="default_format">format to be used preferably in ambivalent
instances</param>
        /// <param name="parsed_date_time">parsed date-time output</param>
        /// <returns>true if date or time was found, else false</returns>
```

```
        static public bool TryParseDateOrTime(this string str, DateTimeFormat
default_format, out ParsedDateTime parsed_date_time)
        {
            parsed_date_time = null;

            ParsedDateTime parsed_date;
            ParsedDateTime parsed_time;
            if (!TryParseDate(str, default_format, out parsed_date))
            {
                if (!TryParseTime(str, default_format, out parsed_time, null))
                    return false;

                DateTime date_time = new DateTime(DefaultDate.Year,
DefaultDate.Month, DefaultDate.Day, parsed_time.DateTime.Hour,
parsed_time.DateTime.Minute, parsed_time.DateTime.Second);
                parsed_date_time = new ParsedDateTime(-1, -1,
parsed_time.IndexOfTime, parsed_time.LengthOfTime, date_time,
parsed_time.UtcOffset);
            }
            else
            {
                if (!TryParseTime(str, default_format, out parsed_time,
parsed_date))
                {
                    DateTime date_time = new DateTime(parsed_date.DateTime.Year,
parsed_date.DateTime.Month, parsed_date.DateTime.Day, 0, 0, 0);
                    parsed_date_time = new ParsedDateTime(parsed_date.IndexOfDate,
parsed_date.LengthOfDate, -1, -1, date_time);
                }
                else
                {
                    DateTime date_time = new DateTime(parsed_date.DateTime.Year,
parsed_date.DateTime.Month, parsed_date.DateTime.Day, parsed_time.DateTime.Hour,
parsed_time.DateTime.Minute, parsed_time.DateTime.Second);
                    parsed_date_time = new ParsedDateTime(parsed_date.IndexOfDate,
parsed_date.LengthOfDate, parsed_time.IndexOfTime, parsed_time.LengthOfTime,
date_time, parsed_time.UtcOffset);
                }
            }

            return true;
        }

        #endregion

        #region parsing base methods

        /// <summary>
        /// Tries to find time within the passed string (relatively to the passed
parsed_date if any) and return it as ParsedDateTime object.
```

```
        /// It recognizes only time while ignoring date, so date in the returned
ParsedDateTime is always 1/1/1
        /// </summary>
        /// <param name="str">string that contains date</param>
        /// <param name="default_format">format to be used preferably in ambivalent
instances</param>
        /// <param name="parsed_time">parsed date-time output</param>
        /// <param name="parsed_date">ParsedDateTime object if the date was found
within this string, else NULL</param>
        /// <returns>true if time was found, else false</returns>
        public static bool TryParseTime(this string str, DateTimeFormat
default_format, out ParsedDateTime parsed_time, ParsedDateTime parsed_date)
        {
            parsed_time = null;

            string time_zone_r;
            if(default_format == DateTimeFormat.USA_DATE)
                time_zone_r = @"(?:\s*(?'time_zone'UTC|GMT|CST|EST))?";
            else
                time_zone_r = @"(?:\s*(?'time_zone'UTC|GMT))?";

            Match m;
            if (parsed_date != null && parsed_date.IndexOfDate > -1)
            {//look around the found date
                //look for <date> hh:mm:ss <UTC offset>
                m = Regex.Match(str.Substring(parsed_date.IndexOfDate +
parsed_date.LengthOfDate), @"(?<=^\s*,?\s+|^\s*at\s*|^\s*[T\-
]\s*)(?'hour'\d{2})\s*:\s*(?'minute'\d{2})\s*:\s*(?'second'\d{2})\s+(?'offset_sign'
[\+\-])(?'offset_hh'\d{2}):?(?'offset_mm'\d{2})(?=$|[^\d\w])",
RegexOptions.Compiled);
                if (!m.Success)
                    //look for <date> [h]h:mm[:ss] [PM/AM] [UTC/GMT]
                    m = Regex.Match(str.Substring(parsed_date.IndexOfDate +
parsed_date.LengthOfDate), @"(?<=^\s*,?\s+|^\s*at\s*|^\s*[T\-
]\s*)(?'hour'\d{1,2})\s*:\s*(?'minute'\d{2})\s*(?::\s*(?'second'\d{2}))?(?:\s*(?'am
pm'AM|am|PM|pm))?"+time_zone_r+@"(?=$|[^\d\w])", RegexOptions.Compiled);
                if (!m.Success)
                    //look for [h]h:mm:ss [PM/AM] [UTC/GMT] <date>
                    m = Regex.Match(str.Substring(0, parsed_date.IndexOfDate),
@"(?<=^|[^\d])(?'hour'\d{1,2})\s*:\s*(?'minute'\d{2})\s*(?::\s*(?'second'\d{2}))?(?
:\s*(?'ampm'AM|am|PM|pm))?"+time_zone_r+@"(?=$|[\s,]+)", RegexOptions.Compiled);
                if (!m.Success)
                    //look for [h]h:mm:ss [PM/AM] [UTC/GMT] within <date>
                    m = Regex.Match(str.Substring(parsed_date.IndexOfDate,
parsed_date.LengthOfDate),
@"(?<=^|[^\d])(?'hour'\d{1,2})\s*:\s*(?'minute'\d{2})\s*(?::\s*(?'second'\d{2}))?(?
:\s*(?'ampm'AM|am|PM|pm))?"+time_zone_r+@"(?=$|[\s,]+)", RegexOptions.Compiled);
            }
            else//look anywhere within string
            {
                //look for hh:mm:ss <UTC offset>
```

```
                m = Regex.Match(str,
@"(?<=^|\s+|\s*T\s*)(?'hour'\d{2})\s*:\s*(?'minute'\d{2})\s*:\s*(?'second'\d{2})\s+
(?'offset_sign'[\+\-])(?'offset_hh'\d{2}):?(?'offset_mm'\d{2})?(?=$|[^\d\w])",
RegexOptions.Compiled);
                if (!m.Success)
                    //look for [h]h:mm[:ss] [PM/AM] [UTC/GMT]
                    m = Regex.Match(str,
@"(?<=^|\s+|\s*T\s*)(?'hour'\d{1,2})\s*:\s*(?'minute'\d{2})\s*(?::\s*(?'second'\d{2
}))?(?:\s*(?'ampm'AM|am|PM|pm))?"+time_zone_r+@"(?=$|[^\d\w])",
RegexOptions.Compiled);
            }

            if (!m.Success)
                return false;

            //try
            //{
            int hour = int.Parse(m.Groups["hour"].Value);
            if (hour < 0 || hour > 23)
                return false;

            int minute = int.Parse(m.Groups["minute"].Value);
            if (minute < 0 || minute > 59)
                return false;

            int second = 0;
            if (!string.IsNullOrEmpty(m.Groups["second"].Value))
            {
                second = int.Parse(m.Groups["second"].Value);
                if (second < 0 || second > 59)
                    return false;
            }

            if (string.Compare(m.Groups["ampm"].Value, "PM", true) == 0 && hour <
12)
                hour += 12;
            else if (string.Compare(m.Groups["ampm"].Value, "AM", true) == 0 &&
hour == 12)
                hour -= 12;

            DateTime date_time = new DateTime(1, 1, 1, hour, minute, second);

            if (m.Groups["offset_hh"].Success)
            {
                int offset_hh = int.Parse(m.Groups["offset_hh"].Value);
                int offset_mm = 0;
                if (m.Groups["offset_mm"].Success)
                    offset_mm = int.Parse(m.Groups["offset_mm"].Value);
                TimeSpan utc_offset = new TimeSpan(offset_hh, offset_mm, 0);
                if (m.Groups["offset_sign"].Value == "-")
                    utc_offset = -utc_offset;
```

```
                parsed_time = new ParsedDateTime(-1, -1, m.Index, m.Length,
date_time, utc_offset);
                return true;
            }

            if (m.Groups["time_zone"].Success)
            {
                TimeSpan utc_offset;
                switch (m.Groups["time_zone"].Value)
                {
                    case "UTC":
                    case "GMT":
                    utc_offset = new TimeSpan(0, 0, 0);
                        break;
                    case "CST":
                        utc_offset = new TimeSpan(-6, 0, 0);
                        break;
                    case "EST":
                        utc_offset = new TimeSpan(-5, 0, 0);
                        break;
                    default:
                        throw new Exception("Time zone: " +
m.Groups["time_zone"].Value + " is not defined.");
                }
                parsed_time = new ParsedDateTime(-1, -1, m.Index, m.Length,
date_time, utc_offset);
                return true;
            }

            parsed_time = new ParsedDateTime(-1, -1, m.Index, m.Length, date_time);
            //}
            //catch(Exception e)
            //{
            //    return false;
            //}
            return true;
        }

        /// <summary>
        /// Tries to find date within the passed string and return it as
ParsedDateTime object.
        /// It recognizes only date while ignoring time, so time in the returned
ParsedDateTime is always 0:0:0.
        /// If year of the date was not found then it accepts the current year.
        /// </summary>
        /// <param name="str">string that contains date</param>
        /// <param name="default_format">format to be used preferably in ambivalent
instances</param>
        /// <param name="parsed_date">parsed date output</param>
        /// <returns>true if date was found, else false</returns>
```

```
        static public bool TryParseDate(this string str, DateTimeFormat
default_format, out ParsedDateTime parsed_date)
        {
            parsed_date = null;

            if (string.IsNullOrEmpty(str))
                return false;

            //look for dd/mm/yy
            Match m = Regex.Match(str,
@"(?<=^|[^\d])(?'day'\d{1,2})\s*(?'separator'[\\/\.])+\s*(?'month'\d{1,2})\s*\'sepa
rator'+\s*(?'year'\d{2}|\d{4})(?=$|[^\d])", RegexOptions.Compiled |
RegexOptions.IgnoreCase);
            if (m.Success)
            {
                DateTime date;
                if ((default_format ^ DateTimeFormat.USA_DATE) ==
DateTimeFormat.USA_DATE)
                {
                    if (!convert_to_date(int.Parse(m.Groups["year"].Value),
int.Parse(m.Groups["day"].Value), int.Parse(m.Groups["month"].Value), out date))
                        return false;
                }
                else
                {
                    if (!convert_to_date(int.Parse(m.Groups["year"].Value),
int.Parse(m.Groups["month"].Value), int.Parse(m.Groups["day"].Value), out date))
                        return false;
                }
                parsed_date = new ParsedDateTime(m.Index, m.Length, -1, -1, date);
                return true;
            }

            //look for [yy]yy-mm-dd
            m = Regex.Match(str,
@"(?<=^|[^\d])(?'year'\d{2}|\d{4})\s*(?'separator'[\-
])\s*(?'month'\d{1,2})\s*\'separator'+\s*(?'day'\d{1,2})(?=$|[^\d])",
RegexOptions.Compiled | RegexOptions.IgnoreCase);
            if (m.Success)
            {
                DateTime date;
                if (!convert_to_date(int.Parse(m.Groups["year"].Value),
int.Parse(m.Groups["month"].Value), int.Parse(m.Groups["day"].Value), out date))
                    return false;
                parsed_date = new ParsedDateTime(m.Index, m.Length, -1, -1, date);
                return true;
            }

            //look for month dd yyyy
            m = Regex.Match(str,
@"(?:^|[^\d\w])(?'month'Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec)[uarychiles
```

```
tmbro]*\s+(?'day'\d{1,2})(?:-?st|-?th|-?rd|-
?nd)?\s*,?\s*(?'year'\d{4})(?=$|[^\d\w])", RegexOptions.Compiled |
RegexOptions.IgnoreCase);
            if (!m.Success)
                //look for dd month [yy]yy
                m = Regex.Match(str, @"(?:^|[^\d\w:])(?'day'\d{1,2})(?:-?st\s+|-
?th\s+|-?rd\s+|-?nd\s+|-
|\s+)(?'month'Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec)[uarychilestmbro]*(?:
\s*,?\s*|-)'?(?'year'\d{2}|\d{4})(?=$|[^\d\w])", RegexOptions.Compiled |
RegexOptions.IgnoreCase);
            if (!m.Success)
                //look for yyyy month dd
                m = Regex.Match(str,
@"(?:^|[^\d\w])(?'year'\d{4})\s+(?'month'Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|No
v|Dec)[uarychilestmbro]*\s+(?'day'\d{1,2})(?:-?st|-?th|-?rd|-?nd)?(?=$|[^\d\w])",
RegexOptions.Compiled | RegexOptions.IgnoreCase);
            if (!m.Success)
                //look for month dd hh:mm:ss MDT|UTC yyyy
                m = Regex.Match(str,
@"(?:^|[^\d\w])(?'month'Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec)[uarychiles
tmbro]*\s+(?'day'\d{1,2})\s+\d{2}\:\d{2}\:\d{2}\s+(?:MDT|UTC)\s+(?'year'\d{4})(?=$|
[^\d\w])", RegexOptions.Compiled | RegexOptions.IgnoreCase);
            if (!m.Success)
                //look for  month dd [yyyy]
                m = Regex.Match(str,
@"(?:^|[^\d\w])(?'month'Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec)[uarychiles
tmbro]*\s+(?'day'\d{1,2})(?:-?st|-?th|-?rd|-
?nd)?(?:\s*,?\s*(?'year'\d{4}))?(?=$|[^\d\w])", RegexOptions.Compiled |
RegexOptions.IgnoreCase);
            if (m.Success)
            {
                int month = -1;
                int index_of_date = m.Index;
                int length_of_date = m.Length;

                switch (m.Groups["month"].Value)
                {
                    case "Jan":
                    case "JAN":
                        month = 1;
                        break;
                    case "Feb":
                    case "FEB":
                        month = 2;
                        break;
                    case "Mar":
                    case "MAR":
                        month = 3;
                        break;
                    case "Apr":
                    case "APR":
```

```
                    month = 4;
                    break;
                case "May":
                case "MAY":
                    month = 5;
                    break;
                case "Jun":
                case "JUN":
                    month = 6;
                    break;
                case "Jul":
                    month = 7;
                    break;
                case "Aug":
                case "AUG":
                    month = 8;
                    break;
                case "Sep":
                case "SEP":
                    month = 9;
                    break;
                case "Oct":
                case "OCT":
                    month = 10;
                    break;
                case "Nov":
                case "NOV":
                    month = 11;
                    break;
                case "Dec":
                case "DEC":
                    month = 12;
                    break;
            }

            int year;
            if (!string.IsNullOrEmpty(m.Groups["year"].Value))
                year = int.Parse(m.Groups["year"].Value);
            else
                year = DefaultDate.Year;

            DateTime date;
            if (!convert_to_date(year, month, int.Parse(m.Groups["day"].Value),
out date))
                return false;
            parsed_date = new ParsedDateTime(index_of_date, length_of_date, -1,
-1, date);
            return true;
        }

        return false;
```

```
        }

        static bool convert_to_date(int year, int month, int day, out DateTime
date)
        {
            if (year >= 100)
            {
                if (year < 1000)
                {
                    date = new DateTime(1, 1, 1);
                    return false;
                }
            }
            else
                if (year > 30)
                    year += 1900;
                else
                    year += 2000;

            try
            {
                date = new DateTime(year, month, day);
            }
            catch
            {
                date = new DateTime(1, 1, 1);
                return false;
            }
            return true;
        }

        #endregion
    }
}
```

Types

Appointment.cs

```
using System;

namespace TestApi.Types
{
    public class appointment : Object // Inherits from the object to get the
compare function
    {
        private int _id;
        private int _helperId;
        private int _elderlyId;
        private DateTime _dateAndTime;
        private DateTime _dateCreated;
```

```
private int _ratingId;
private int _reviewId;
public int id {
    get
    {
        return _id;
    }
    set
    {
        _id = value;
    }
}
public int helperId
{
    get
    {
        return _helperId;
    }
    set
    {
        _helperId = value;
    }
}
public int elderlyId
{
    get
    {
        return _elderlyId;
    }
    set
    {
        _elderlyId = value;
    }
}
public DateTime dateAndTime
{
    get
    {
        return _dateAndTime;
    }
    set
    {
        _dateAndTime = value;
    }
}
public DateTime dateCreated
{
    get
    {
        return _dateCreated;
    }
```

```
            set
            {
                _dateCreated = value;
            }
        }
        public int ratingId
        {
            get
            {
                return _ratingId;
            }
            set
            {
                _ratingId = value;
            }
        }
        public int reviewId
        {
            get
            {
                return _reviewId;
            }
            set
            {
                _reviewId = value;
            }
        }

        public static bool Equals(appointment a, appointment b) // Check if two
appointments are the same
        {
            if (a.id != b.id)
                return false;
            if (a.helperId != b.helperId)
                return false;
            if (a.elderlyId != b.elderlyId)
                return false;
            if (DateTime.Compare(a.dateAndTime, b.dateAndTime) != 0)
                return false;
            if (DateTime.Compare(a.dateCreated, b.dateCreated) != 0)
                return false;
            if (a.ratingId != b.ratingId)
                return false;
            if (a.reviewId != b.reviewId)
                return false;
            return true;


        }
    }
}
```

Day.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace TestApi.Types
{
    public class day
    {
        private int _id;
        private bool[] _timesFree;
        public int id {
            get
            {
                return _id;
            }
            set
            {
                _id = value;
            }
        }
        public bool[] timesFree
        {
            get
            {
                return _timesFree;
            }
            set
            {
                _timesFree = value;
            }
        }

    }
}
```

Rating.cs

```
using System;
namespace TestApi.Controllers
{
    public class rating
    {
        private int _id;
        private int _starRating;
        private int _helperId;

        public int id
        {
            get
```

```
            {
                return _id;
            }
            set
            {
                _id = value;
            }
        }
        public int starRating
        {
            get
            {
                return _starRating;
            }
            set
            {
                _starRating = value;
            }
        }
        public int helperId
        {
            get
            {
                return _helperId;
            }
            set
            {
                _helperId = value;
            }
        }
    }
}
```

Review.cs

```
using System;
namespace TestApi.Controllers
{
    public class review
    {
        private int _id;
        private int _helperId;
        private string _comment;

        public int id
        {
            get
            {
                return _id;
            }
            set
            {
```

```
                _id = value;
            }
        }
        public int helperId
        {
            get
            {
                return _helperId;
            }
            set
            {
                _helperId = value;
            }
        }
        public string comment
        {
            get
            {
                return _comment;
            }
            set
            {
                _comment = value;
            }
        }

    }
}
```

SMS.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace TestApi.Controllers
{
    public class sms
    {
        private string _From;
        private string _Body;
        public string From {
            get
            {
                return _From;
            }
            set
            {
                _From = value;
            }
```

```
        }
        public string Body {
            get
            {
                return _Body;
            }
            set
            {
                _Body = value;
            }
        }
    }
}
```

Timetable.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace TestApi.Types
{
    public class names
    {
        private DateTime _weekBeginning;
        private week _week;
        public DateTime weekBeginning {
            get
            {
                return _weekBeginning;
            }
            set
            {
                _weekBeginning = value;
            }
        }
        public week week {
            get
            {
                return _week;
            }
            set
            {
                _week = value;
            }
        }

    }
    public class timetable
    {
```

```
        private int _id;
        private week _defaultWeek;
        private List<names> _weeks = new List<names>();
        public int id {
            get
            {
                return _id;
            }
            set
            {
                _id = value;
            }
        }
        public week defaultWeek {
            get
            {
                return _defaultWeek;
            }
            set
            {
                _defaultWeek = value;
            }
        }
        public List<names> weeks {
            get
            {
                return _weeks;
            }
            set
            {
                _weeks = value;
            }
        }

    }
}
```

User.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace TestApi.Types
{
    public class user
    {
        private int _id;
        private string _username;
        private string _firstname;
```

```
private string _surname;
private string _password;
private string _emailAddress;
private string _firstLineOfAddress;
private string _secondLineOfAddress;
private string _telephoneNumber;
private string _postalCode;
private string _city;
private string _country;
private bool _helper = false;
private int _timetableId;
private double _distance = double.MaxValue;
public int id {
    get
    {
        return _id;
    }
    set
    {
        _id = value;
    }
}
public string username {
    get
    {
        return _username;
    }
    set
    {
        _username = value;
    }
}
public string firstName {
    get
    {
        return _firstname;
    }
    set
    {
        _firstname = value;
    }
}
public string surname {
    get
    {
        return _surname;
    }
    set
    {
        _surname = value;
    }
```

```
}
public string password {
    get
    {
        return _password;
    }
    set
    {
        _password = value;
    }
}
public string emailAddress {
    get
    {
        return _emailAddress;
    }
    set
    {
        _emailAddress = value;
    }
}
public string firstLineOfAddress {
    get
    {
        return _firstLineOfAddress;
    }
    set
    {
        _firstLineOfAddress = value;
    }
}
public string secondLineOfAddress{
    get
    {
        return _secondLineOfAddress;
    }
    set
    {
        _secondLineOfAddress = value;
    }
}
public string telephoneNumber {
    get
    {
        return _telephoneNumber;
    }
    set
    {
        _telephoneNumber = value;
    }
}
```

```
public string postalCode{
    get
    {
        return _postalCode;
    }
    set
    {
        _postalCode = value;
    }
}
public string city {
    get
    {
        return _city;
    }
    set
    {
        _city = value;
    }
}
public string country {
    get
    {
        return _country;
    }
    set
    {
        _country = value;
    }
}
public bool helper {
    get
    {
        return _helper;
    }
    set
    {
        _helper = value;
    }
}
public int timetableId {
    get
    {
        return _timetableId;
    }
    set
    {
        _timetableId = value;
    }
}
```

```
        public double distance {
            get
            {
                return _distance;
            }
            set
            {
                _distance = value;
            }
        }
    }
}
```

UserToBeVerified.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace TestApi.Controllers
{
    public class userToBeVerified
    {
        private string _emailAddress;
        private string _password;
        private bool _helper;
        public string emailAddress {
            get
            {
                return _emailAddress;
            }
            set
            {
                _emailAddress = value;
            }
        }
        public string password {
            get
            {
                return _password;
            }
            set
            {
                _password = value;
            }
        }
        public bool helper {
            get
            {
                return _helper;
```

```
            }
            set
            {
                _helper = value;
            }
        }

    }
}
```

Week.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace TestApi.Types
{
    public class week
    {
        private int _id;
        private day _monday;
        private day _tuesday;
        private day _wednesday;
        private day _thursday;
        private day _friday;
        private day _saturday;
        private day _sunday;
        public int id {
            get
            {
                return _id;
            }
            set
            {
                _id = value;
            }
        }
        public day monday {
            get
            {
                return _monday;
            }
            set
            {
                _monday = value;
            }
        }
        public day tuesday {
```

```
        get
        {
            return _tuesday;
        }
        set
        {
            _tuesday = value;
        }
    }
    public day wednesday {
        get
        {
            return _wednesday;
        }
        set
        {
            _wednesday = value;
        }
    }
    public day thursday {
        get
        {
            return _thursday;
        }
        set
        {
            _thursday = value;
        }
    }
    public day friday {
        get
        {
            return _friday;
        }
        set
        {
            _friday = value;
        }
    }
    public day saturday {
        get
        {
            return _saturday;
        }
        set
        {
            _saturday = value;
        }
    }
    public day sunday {
        get
```

```
            {
                return _sunday;
            }
            set
            {
                _sunday = value;
            }
        }
    }
}
```

Controllers

Accounts.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using TestApi.Sql;
using TestApi.Backend;
using TestApi.Types;
namespace TestApi.Controllers


{
    /// <summary>
    /// Requests to be sent to .../api/ac...
    /// </summary>
    [Route("api/ac")]
    public class accounts : SQLStuffOnTopOfController
    {
        /// <summary>
        /// Useful test left - used by the app to test whether the API is working.
        /// </summary>
        /// <returns></returns>
        [HttpGet("")]
        public string GetHelloWorld()
        {
            return "Hello World!";
        }


        /// <summary>
        /// Add a new user to the database
        /// </summary>
        /// <param name="a"> Gets the user from the json of the object (FROM
BODY)</param>
```

```
        /// <returns></returns>
        [HttpPost("users/ad")]
        public bool AddNewUser([FromBody] user a)
        {

            if (!a.helper) // Send it to the right command in either
accountsForElderly or accountsFoHelper
            {
                accountsForElderly ae = new accountsForElderly();
                return ae.AddNewUser(a);
            }
            else if (a.helper)
            {
                accountsForHelper ah = new accountsForHelper();
                return ah.AddNewUser(a); // This is static so this can be done...
            }
            else
                return false;
        }
        /// <summary>
        /// VERIFY USER
        ///     Just sends the contents to the correct section - accounts for
elderly or accounts for helpers.
        ///     Saves having to have a different login process.
        /// </summary>
        /// <param name="a"></param>
        /// <returns></returns>
        [HttpPost("users/vf")]
        public user verifyUser([FromBody] userToBeVerified a)
        {
            if (!a.helper)
            {
                accountsForElderly ae = new accountsForElderly();
                return ae.verifyUser(a);
            }
            else if (a.helper)
            {
                accountsForHelper ah = new accountsForHelper();
                return ah.verifyUser(a);
            }
            else
                return new user();
        }
    }
}
```

accountsForElderly.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
```

```
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using TestApi.Sql;
using TestApi.Types;
namespace TestApi.Controllers
{

    /// <summary>
    /// This is one of the paths of the accounts for elderly people
    /// Called through /api/ace
    /// </summary>
    [Route("api/ace")]
    public class accountsForElderly : SQLStuffOnTopOfController
    {
        /// <summary>
        /// Used as a test for the api if necessary.
        ///
        /// </summary>
        /// <returns>"Hello World"</returns>
        [HttpGet("")]
        public string GetHelloWorld()
        {
            return "Hello World!";
        }

        /// <summary>
        /// Get user info by integer id
        /// </summary>
        /// <param name="id"></param>
        /// <returns>
        /// User with all associated properties
        /// </returns>
        [HttpGet("users/gi{id}")]
        public user GetInfo(int id)
        {
            string response = sqlCommand(true, "SELECT * FROM accountsForElderly
WHERE id=" + Convert.ToString(id)+";", 12);
            user a = new user();
            string[] completelySplitStrings = response.Split('#');
            a.id = int.Parse(completelySplitStrings[0]);
            a.username = completelySplitStrings[1];
            a.emailAddress = completelySplitStrings[2];
            a.password = completelySplitStrings[3];
            a.firstName = completelySplitStrings[4];
            a.surname = completelySplitStrings[5];
            a.firstLineOfAddress = completelySplitStrings[6];
            a.secondLineOfAddress = completelySplitStrings[7];
            a.telephoneNumber = completelySplitStrings[8];
            a.postalCode = completelySplitStrings[9];
            a.city = completelySplitStrings[10];
            a.country = completelySplitStrings[11];
```

```
            return a;
        }
        /// <summary>
        /// Returns the user of a particular phone number
        /// Only called internally by the appointment calling system when an SMS is
sent
        /// </summary>
        /// <param name="phoneNumber">Correct user is sent or an empty user if
phone number not found</param>
        /// <returns></returns>
        public user getInfoByPhoneNumber(string phoneNumber)
        {
            string response = sqlCommand(true, "SELECT * FROM accountsForElderly
WHERE telephoneNumber=" + phoneNumber + ";", 12);
            user a = new user();
            try
            {
                string[] completelySplitStrings = response.Split('#');
                a.id = int.Parse(completelySplitStrings[0]);
                a.username = completelySplitStrings[1];
                a.emailAddress = completelySplitStrings[2];
                a.password = completelySplitStrings[3];
                a.firstName = completelySplitStrings[4];
                a.surname = completelySplitStrings[5];
                a.firstLineOfAddress = completelySplitStrings[6];
                a.secondLineOfAddress = completelySplitStrings[7];
                a.telephoneNumber = completelySplitStrings[8];
                a.postalCode = completelySplitStrings[9];
                a.city = completelySplitStrings[10];
                a.country = completelySplitStrings[11];
            }
            catch(Exception) {}
            return a;
        }

        /// <summary>
        /// Add new user to database
        /// </summary>
        /// <param name="a">New user to be added</param>
        /// <returns>Returns a boolean value as to whether the user has been
successfully added,</returns>
        [HttpPost("users/ad")]
        public bool AddNewUser([FromBody] user a)
        {
            int response = int.Parse(sqlCommand(true, "SELECT count(1) FROM
accountsForElderly WHERE emailAddress = '" + a.emailAddress + "'").Split('#')[0]);
// Checks if the email address already exists or not
            int response2 = int.Parse(sqlCommand(true, "SELECT count(1) FROM
accountsForElderly WHERE telephoneNumber = '" + a.telephoneNumber +
"'").Split('#')[0]); // Checks if the phone number already exists or not.
            //Console.WriteLine(b);
```

```
            if (response == 0 && response2 == 0)
            {
                sqlCommand(false, "INSERT INTO accountsForElderly (username,
emailAddress, password, firstname, surname, firstlineOfAddress,
secondlineOfAddress, telephoneNumber, postalCode, city, country) VALUES
('"+a.username + "','" + a.emailAddress + "','" + a.password + "','" + a.firstName
+ "','" + a.surname + "','" + a.firstLineOfAddress + "','" + a.secondLineOfAddress
+ "','" + a.telephoneNumber + "','" + a.postalCode + "','" + a.city + "','" +
a.country + "');");
                return true;
            }
            else
                return false;
        }

        /// <summary>
        /// Delete a user from the database based on their id.
        /// </summary>
        /// <param name="id"></param>
        /// <returns></returns>
        [HttpDelete("users/rm{id}")]
        public bool Delete(int id)
        {
            try{
                sqlCommand(false, "DELETE FROM accountsForElderly WHERE id=" +
Convert.ToString(id)+";");
                return true;
                        }
            catch(Exception){
                return false;
            }

        }

        /// <summary>
        /// Update the user changing it to the contents of the user whose json is
sent to the api.
        /// </summary>
        /// <param name="a">User to be updated</param>
        /// <returns>Boolean according to whether it was successful or
not</returns>
        [HttpPut("users/up")]
        public bool Update([FromBody] user a)
        {
            int response = int.Parse(sqlCommand(true, "SELECT count(1) FROM
accountsForElderly WHERE id = " + a.id + ";").Split('#')[0]);
            if(response==0)
                return false; // returns a false if there is no id

            sqlCommand(false, String.Format("UPDATE accountsForElderly SET
username='{0}',emailAddress='{1}',password='{2}',firstname='{3}',surname='{4}',firs
```

```
tlineOfAddress='{5}',secondlineOfAddress='{6}',telephoneNumber='{7}',postalCode='{8
}',city='{9}',country='{10}' WHERE id={11}",a.username, a.emailAddress, a.password,
a.firstName, a.surname, a.firstLineOfAddress,
a.secondLineOfAddress,a.telephoneNumber, a.postalCode, a.city, a.country, a.id));
                return true;
        }

        /// <summary>
        /// Verify User
        ///     Returns a user
        ///         Empty if the login was unsuccessful
        ///         Otherwise populated with the contents of the user if it was
successful
        /// </summary>
        /// <param name="b"></param>
        /// <returns></returns>
        [HttpPost("users/vf")]
        public user verifyUser([FromBody] userToBeVerified b)
        {
            string output = ("SELECT count(1) FROM accountsForElderly WHERE
emailAddress='" + b.emailAddress + "' AND password='" + b.password +
"'").Split('#')[0];

            if (int.Parse(sqlCommand(true, "SELECT count(1) FROM accountsForElderly
WHERE emailAddress='" + b.emailAddress + "' AND password='" + b.password + "'",
1).Split('#')[0]) > 0)
            {
                string response = sqlCommand(true, "SELECT * FROM
accountsForElderly WHERE emailAddress='" + b.emailAddress + "';", 12);
                user a = new user();
                string[] completelySplitStrings = response.Split('#');
                a.id = int.Parse(completelySplitStrings[0]);
                a.username = completelySplitStrings[1];
                a.emailAddress = completelySplitStrings[2];
                a.password = completelySplitStrings[3];
                a.firstName = completelySplitStrings[4];
                a.surname = completelySplitStrings[5];
                a.firstLineOfAddress = completelySplitStrings[6];
                a.secondLineOfAddress = completelySplitStrings[7];
                a.telephoneNumber = completelySplitStrings[8];
                a.postalCode = completelySplitStrings[9];
                a.city = completelySplitStrings[10];
                a.country = completelySplitStrings[11];
                a.helper = false;
                return a;
            }
            else
                return new user() ;


        }
```

```
    }
}
```

accountsForHelper.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using TestApi.Sql;
using Newtonsoft.Json.Linq;
using System.Net;
using System.Net.Http;
using System.IO;
using TestApi.Types;
namespace TestApi.Controllers
{
    /// <summary>
    /// AccountForHelper: All the functions for communicating with the helper
database
    /// </summary>
    [Route("api/ach")]
    public class accountsForHelper : SQLStuffOnTopOfController
    {
        /// <summary>
        ///
        /// Test function
        /// </summary>
        /// <returns></returns>
        [HttpGet("")]
        public string GetHelloWorld()
        {
            return "Hello World!";
        }
        /// <summary>
        /// Get a list of users
        /// </summary>
        /// <returns>
        /// List of all the users in the file of users
        /// Should soon be deprecated as this is definitely not safe and should not
be used in a final project.
        /// </returns>
        [HttpGet("users")]
        public List<user> Get()
        {

            string listOfUsers = sqlCommand(true, "SELECT * FROM
accountsForHelpers", 13);
```

```
            List<user> listToReturn = new List<user>();
            string[] splitList = listOfUsers.Split('\n');
            //string c = splitList[0];

            for (int i = 0; i < splitList.Length - 1; i++)
            {
                user a = new user();
                string[] completelySplitStrings = splitList[i].Split('#');
                a.id = int.Parse(completelySplitStrings[0]);
                a.username = completelySplitStrings[1];
                a.emailAddress = completelySplitStrings[2];
                a.password = completelySplitStrings[3];
                a.firstName = completelySplitStrings[4];
                a.surname = completelySplitStrings[5];
                a.firstLineOfAddress = completelySplitStrings[6];
                a.secondLineOfAddress = completelySplitStrings[7];
                a.telephoneNumber = completelySplitStrings[8];
                a.postalCode = completelySplitStrings[9];
                a.city = completelySplitStrings[10];
                a.country = completelySplitStrings[11];
                int l = 0;
                int.TryParse(completelySplitStrings[12], out l);
                a.timetableId = l;
                listToReturn.Add(a);
            }
            return listToReturn;
        }
        /// <summary>
        /// Get user info by integer id
        /// </summary>
        /// <param name="id"></param>
        /// <returns>
        /// User with all associated properties
        /// </returns>
        [HttpGet("users/gi{id}")]
        public user GetInfo(int id)
        {
            string response = sqlCommand(true, "SELECT * FROM accountsForHelpers
WHERE id=" + Convert.ToString(id) + ";", 13);
            user a = new user();
            string[] completelySplitStrings = response.Split('#');
            a.id = int.Parse(completelySplitStrings[0]);
            a.username = completelySplitStrings[1];
            a.emailAddress = completelySplitStrings[2];
            a.password = completelySplitStrings[3];
            a.firstName = completelySplitStrings[4];
            a.surname = completelySplitStrings[5];
            a.firstLineOfAddress = completelySplitStrings[6];
            a.secondLineOfAddress = completelySplitStrings[7];
            a.telephoneNumber = completelySplitStrings[8];
            a.postalCode = completelySplitStrings[9];
```

```
            a.city = completelySplitStrings[10];
            a.country = completelySplitStrings[11];
            a.timetableId = int.Parse(completelySplitStrings[12]);
            a.helper = true;

            return a;
        }

        /// <summary>
        /// Add a new user to the database if and only if there is no user with the
same email address and / or telephone number
        /// </summary>
        /// <param name="a"></param>
        /// <returns></returns>
        [HttpPost("users/ad")]
        public bool AddNewUser([FromBody] user a)
        {
            int response = int.Parse(sqlCommand(true, "SELECT count(1) FROM
accountsForHelpers WHERE emailAddress = '" + a.emailAddress + "'").Split('#')[0]);
            int response2 = int.Parse(sqlCommand(true, "SELECT count(1) FROM
accountsForHelpers WHERE telephoneNumber = '" + a.telephoneNumber +
"'").Split('#')[0]);

            if (response == 0 && response2 == 0)
            {
                timetables t = new timetables(); // Gets a new timetable and
associates it with the user
                timetable defaultTimetable = t.getTimetable(1); // The default
timetable is timetable 1
                t.addTimetable(defaultTimetable, false); // Insert the new
timetable
                int timetableId = int.Parse(sqlCommand(true, "SELECT max(id) FROM
timetables", 1).Split('#')[0]); // and get the new timetablesId
                sqlCommand(false, "INSERT INTO accountsForHelpers (username,
emailAddress, password, firstname, surname, firstlineOfAddress,
secondlineOfAddress, telephoneNumber, postalCode, city, country, timetableId)
VALUES ('" + a.username + "','" + a.emailAddress + "','" + a.password + "','" +
a.firstName + "','" + a.surname + "','" + a.firstLineOfAddress + "','" +
a.secondLineOfAddress + "','" + a.telephoneNumber + "','" + a.postalCode + "','" +
a.city + "','" + a.country + "'," + timetableId + ");");
                return true;
            }
            else
                return false; // Returns a false if thre user has not been added
        }

        /// <summary>
        /// Remove the user based on the id of the user
        /// </summary>
        /// <param name="id">Integer id in the database</param>
        /// <returns></returns>
```

```
        [HttpDelete("users/rm{id}")]
        public bool Delete(int id)
        {
            try {
                sqlCommand(false, "DELETE FROM accountsForHelpers WHERE id=" +
Convert.ToString(id) + ";");
                return true;
            }
            catch (Exception) { //If it doesn't exist
                return false;
            }

        }

        /// <summary>
        /// Update the user to a new user
        /// </summary>
        /// <param name="a"></param>
        /// <returns></returns>
        [HttpPut("users/up")]
        public bool Update([FromBody] user a)
        {
            int response = int.Parse(sqlCommand(true, "SELECT count(1) FROM
accountsForHelpers WHERE emailAddress = '" + a.emailAddress + "'").Split('#')[0]);
            if (response != 0)
                return false;

            sqlCommand(false, String.Format("UPDATE accountsForHelpers SET
username='{0}',emailAddress='{1}',password='{2}',firstname='{3}',surname='{4}',firs
tlineOfAddress='{5}',secondlineOfAddress='{6}',telephoneNumber='{7}',postalCode='{8
}',city='{9}',country='{10}' WHERE id={11}", a.username, a.emailAddress,
a.password, a.firstName, a.surname, a.firstLineOfAddress, a.secondLineOfAddress,
a.telephoneNumber, a.postalCode, a.city, a.country, a.id));
            return true;

        }


        /// <summary>
        /// Verify the user - returns an empty user if doesn't exist.
        /// Otherwise returns the contents of the new user.
        /// </summary>
        /// <param name="b"></param>
        /// <returns></returns>
        [HttpPost("users/vf")]
        public user verifyUser([FromBody] userToBeVerified b)
        {
            if (int.Parse(sqlCommand(true, "SELECT count(1) FROM accountsForHelpers
WHERE emailAddress='" + b.emailAddress + "' AND password='" + b.password + "'",
1).Split('#')[0]) > 0)
            {
```

```
                string response = sqlCommand(true, "SELECT * FROM
accountsForHelpers WHERE emailAddress='" + b.emailAddress + "';", 12);
                user a = new user();
                string[] completelySplitStrings = response.Split('#');
                a.id = int.Parse(completelySplitStrings[0]);
                a.username = completelySplitStrings[1];
                a.emailAddress = completelySplitStrings[2];
                a.password = completelySplitStrings[3];
                a.firstName = completelySplitStrings[4];
                a.surname = completelySplitStrings[5];
                a.firstLineOfAddress = completelySplitStrings[6];
                a.secondLineOfAddress = completelySplitStrings[7];
                a.telephoneNumber = completelySplitStrings[8];
                a.postalCode = completelySplitStrings[9];
                a.city = completelySplitStrings[10];
                a.country = completelySplitStrings[11];
                a.helper = true;
                return a;
            }
            else
                return new user();
        }

        /// <summary>
        /// Return the list of helpers who are able to help (ordered by how near
they are to the user)
        /// </summary>
        /// <param name="idOfElderlyPerson"></param>
        /// <returns></returns>
        [HttpGet("gl{idOfElderlyPerson}")]
        public List<user> getListOfUsersByDistanceFromElderlyPerson(int
idOfElderlyPerson = 4)
        {
            accountsForHelper ah = new accountsForHelper();
            accountsForElderly ae = new accountsForElderly();
            List<user> listOfAllElderly = ah.Get();
            user elderlyPerson = ae.GetInfo(idOfElderlyPerson);
            string address = elderlyPerson.postalCode; // Entirely rely upon the
postcode since this is the most reliable.
            //string address = elderlyPerson.firstLineOfAddress + ", " +
elderlyPerson.secondLineOfAddress;
            List<user> sortedList = mergeSort(listOfAllElderly, address);
            return sortedList;

        }
        public static List<user> mergeSort(List<user> listOfAllElderly, string
address)
        {
            if (listOfAllElderly.Count() == 1) // Base Case
                return listOfAllElderly;
```

```
            List<user> firstHalf = new List<user>(); // Split the entirety of the
list into a first half and second half
            List<user> secondHalf = new List<user>();
            for (int i = 0; i < listOfAllElderly.Count() / 2; i++) // Effectively
copy until half way (or just before halfway)
            {
                firstHalf.Add(listOfAllElderly[0]);
                listOfAllElderly.RemoveAt(0);
            }
            for (int j = 0; j < listOfAllElderly.Count(); j++) // Take the rest
            {
                secondHalf.Add(listOfAllElderly[j]);
            }
            return merge(mergeSort(firstHalf, address), mergeSort(secondHalf,
address), address); // Recursively mergesort
        }
        public static List<user> merge(List<user> listOne, List<user> ListTwo,
string address)
        {

            List<user> sortedListToReturn = new List<user>(); // Setup the new
merged (sorted list)
            while(listOne.Count() != 0 || ListTwo.Count() != 0) // While there are
objects in both lists
            {
                if (listOne.Count() <= 0) // If one list is empty
                {
                    sortedListToReturn.Add(ListTwo[0]);
                    ListTwo.RemoveAt(0);
                }
                else if (ListTwo.Count() <= 0) // If other list is empty
                {
                    sortedListToReturn.Add(listOne[0]);
                    listOne.RemoveAt(0);
                }

                else
                {
                    accountsForHelper ah = new accountsForHelper();
                    accountsForElderly ae = new accountsForElderly();
                    string AddressOne = listOne[0].postalCode;
                    string AddressTwo = ListTwo[0].postalCode;
                    if (listOne[0].distance == double.MaxValue) // If distance
hasn't been already found (defaults to max value of double)
                        listOne[0].distance =  ah.getDistance(address, AddressOne);
                    double distanceOne = listOne[0].distance;
                    if (ListTwo[0].distance == double.MaxValue)
                        ListTwo[0].distance = ah.getDistance(address, AddressTwo);
                    double distanceTwo = ListTwo[0].distance;
                    if (distanceOne <= distanceTwo) // Adds something to list and
removes that from the other list depending on which distance is shorter.
```

```
                    {
                        sortedListToReturn.Add(listOne[0]);
                        listOne.RemoveAt(0);
                    }
                    else
                    {
                        sortedListToReturn.Add(ListTwo[0]);
                        ListTwo.RemoveAt(0);
                    }
                }

            }
            return sortedListToReturn;
        }

        /*
        Gets the distance between two points.
         */

        [HttpGet("users/gd")]
        public double getDistance(string origin, string destination)
        {

            double distance = double.MaxValue; // In case of a failure of the API
(especially if postcode is invalid)
            string url =
"http://maps.googleapis.com/maps/api/directions/json?origin=" + origin +
"&destination=" + destination + "&sensor=false"; // Gets value of the Google API
            string requesturl = url;
            string content = fileGetContents(requesturl);
            JObject o = JObject.Parse(content); // Parses the object
            try
            {
                distance = (int)o.SelectToken("routes[0].legs[0].distance.value");
// Gets the value of the distance.
                return distance / 1000;
            }
            catch
            {
                return distance;
            }
        }
        /*
        Completes the actual HTTP Request
         */
        protected static string fileGetContents(string fileName)
        {
            string sContents = string.Empty;
            string me = string.Empty;
            try
            {
```

```
                if (fileName.ToLower().IndexOf("http:") > -1)
                {
                    HttpClient wc = new HttpClient();
                    byte[] response = wc.GetByteArrayAsync(fileName).Result;
                    sContents = System.Text.Encoding.ASCII.GetString(response);

                }
                else
                {

                }
            }
            catch { }
            return sContents;
        }



    }
}
```

Appointments.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using MySql.Data.MySqlClient;
using MySql.Data.Common;
using MySql.Data.Types;
using System.Net.Http;
using Newtonsoft.Json;
using System.IO;
using System.Net;
using TestApi.Backend;
using TestApi.Sql;
using TestApi.Types;
using System.Text;

namespace TestApi.Controllers
{

    [Route("api/app")]
    public class appointments : SQLStuffOnTopOfController
    {
       /// <summary>
       /// Responds to an SMS request - finds the nearest person who is available
at the requested time.
       /// </summary>
```

```
        /// <param name="c">String of the contents received by the SMS
server</param>
        /// <returns></returns>
        [HttpPostAttribute("sms{c}")]
        public bool createNewAppointmentsFromSMSRequest(string c)
        {
             //Parses the string into an SMS object
             string[] split = c.Split(',');
             sms a = new sms();
             a.From = split[0];
             a.Body = split[1];
             //Gets the date and time from the message
             DateTime fromTheMessage = new DateTime();
             bool foundTime = true;
             bool goneInto = false;
             if (a.Body.Contains("today") || a.Body.Contains("Today")) // If the
message contains today in either capitalised or non capitalised format
             {
                 goneInto = true;
                 foundTime =
a.Body.TryParseTime(DateTimeRoutines.DateTimeFormat.UK_DATE, out fromTheMessage);
// Get the time
                 DateTime newDt = new DateTime(DateTime.Now.Year,
DateTime.Now.Month, DateTime.Now.Day, fromTheMessage.Hour, fromTheMessage.Minute,
fromTheMessage.Second); // Create new object with today's date but the asked for
time
                 fromTheMessage = newDt; // Makes the from the message this new
datetime
             }
             if (a.Body.Contains("tomorrow") || a.Body.Contains("Tomorrow"))
             {
                 goneInto = true;
                 foundTime =
a.Body.TryParseTime(DateTimeRoutines.DateTimeFormat.UK_DATE, out fromTheMessage);
                 DateTime newDt = new DateTime(DateTime.Now.AddDays(1).Year,
DateTime.Now.AddDays(1).Month, DateTime.Now.AddDays(1).Day, fromTheMessage.Hour,
fromTheMessage.Minute, fromTheMessage.Second);
                 fromTheMessage = newDt;
             }
             if (!goneInto)
             {
                 fromTheMessage = getDateTimeTest(a.Body); // Otherwise, if no today
or tomorrow, go into the main test of finding date and time.
             }
             if(fromTheMessage.Year == 1 || !foundTime || fromTheMessage <
DateTime.Now) // Doesn't continue if appointment in the past, or datetime not found
or no time in today
             {
                 a.Body = "You failed to create an appointment as your message text
was not of the right type or you asked to create an appointment in the past. Ensure
the time is in the format of HH:MM."; // sends error message as SMS
```

```
                sendSMS(a.From, a.Body);
                return false;
            }
            else
            {
                accountsForHelper ah = new accountsForHelper();
                accountsForElderly ae = new accountsForElderly();
                List<user> allElderlyUsers =
ah.getListOfUsersByDistanceFromElderlyPerson(ae.getInfoByPhoneNumber(a.From).id);
// Gets list of nearby helpers based on the phone number
                bool foundAFreeUser = false;
                while(!foundAFreeUser) // While no appointment can be created
                {
                    foreach (user nextNearest in allElderlyUsers)
                    {
                        timetables ttt = new timetables();
                        timetable t = ttt.getTimetable(nextNearest.timetableId); //
GetTimetable of the particular user
                        DateTime dt = fromTheMessage;
                        while(dt.DayOfWeek != DayOfWeek.Monday) { dt = dt.AddDays(-
1); } // Gets the date to the date of the weekBeginning
                        int hour = fromTheMessage.Hour;
                        hour = hour / 2;
                        hour = hour * 2; // effectively rounds to the nearest two
hour mark.
                        int nextHour = hour + 2;
                        string weekBeginning = dt.ToString("dd/MM/yyyy"); // Finds
the stringValue of the weekBeginning
                        bool found = false;
                        week correctWeek = new week();
                        for (int i = 0; i < t.weeks.Count(); i++) // Tries to find
if the week is in the custom weeks of the timetable or if it is in the default.
                        {
                            if (t.weeks[i].weekBeginning.ToString("dd/MM/yyyy") ==
weekBeginning) // This avoids issues in attempting to equate DateTimes.
                            {
                                found = true;
                                correctWeek = t.weeks[i].week;
                            }
                        }
                        if(!found)
                        {
                            correctWeek = t.defaultWeek; // sets the week to focus
on as the default week
                        }

                        day correctDay = new day();
                        //GOT THE CORRECT WEEK
                        switch(fromTheMessage.DayOfWeek) // depending on the day of
the week, focuses on the correct day.
                        {
```

```
                                case DayOfWeek.Monday:
                                    correctDay = correctWeek.monday;
                                    break;
                                case DayOfWeek.Tuesday:
                                    correctDay = correctWeek.tuesday;
                                    break;
                                case DayOfWeek.Wednesday:
                                    correctDay = correctWeek.wednesday;
                                    break;
                                case DayOfWeek.Thursday:
                                    correctDay = correctWeek.thursday;
                                    break;
                                case DayOfWeek.Friday:
                                    correctDay = correctWeek.friday;
                                    break;
                                case DayOfWeek.Saturday:
                                    correctDay = correctWeek.saturday;
                                    break;
                                case DayOfWeek.Sunday:
                                    correctDay = correctWeek.sunday;
                                    break;
                                default:
                                    correctDay = correctWeek.monday;
                                    break;
                            }
                            int index = hour / 2; // WHich part of the timesFree array
to check
                            if(correctDay.timesFree[index])
                            {
                                foundAFreeUser = true; // Found a user!! Populates an
appointment and creates it.
                                appointment app = new appointment();
                                app.helperId = nextNearest.id;
                                app.elderlyId = ae.getInfoByPhoneNumber(a.From).id;
                                app.dateAndTime = fromTheMessage;
                                app.dateCreated = DateTime.Now;
                                user helper = ah.GetInfo(app.helperId); // Gets the
helper details for the SMS message
                                createNewAppointmentsForAndroidApp(app);
                                sendSMS(a.From, "You have an appointment with " +
helper.firstName + " " + helper.surname + " at " + app.dateAndTime.ToString("H:mm")
+ " on " + app.dateAndTime.ToString("MMMM dd, yyyy"));
                                foundAFreeUser = true;
                            }
                            if (foundAFreeUser)
                                break;
                        }
                        if (!foundAFreeUser)
                        {
```

```
                        sendSMS(a.From, "Nobody could be found."); // If gone
through everyone and noone available, then sends a message saying noone could be
found.                          return false;
                    }
                }
            }
            return true;
        }
         [HttpPut("sendSMS")]
         public bool sendSMS3([FromBody] sms s) // Since the sendSMS2 is async, it
can't be called with a result of a boolean.
            //Therefore, this completes s, but returning a boolean conversion of
the result.
         {
            if (sendSMS2(s).Result)
                return true;
            else
                return false;
         }
         private async Task<bool> sendSMS2( sms s)
         {
            try // In case of server issues, the catch is there.
            {
                string json = JsonConvert.SerializeObject(s);
                string url = "http://178.62.87.28:700/sendSms";
                var httpContent = new StringContent(json, Encoding.UTF8,
"application/json");
                using (var httpClient = new HttpClient())
                {
                    var httpResponse = await httpClient.PostAsync(url,
httpContent);

                    if (httpResponse.Content != null) // Should be a response of
"SUCCESS" but we can ignore this really
                        //This is here incase we want to do more stuff in the
future with the response.
                    {
                        var responseContent = await
httpResponse.Content.ReadAsStringAsync();
                    }
                }
            }
            catch
            {
                return false;
            }
            return true;
         }
         public async void sendSMS(string from, string body) // Effectively the
same, but used within this solution, rather than by the API.
            // Therefore kept seperate to ensure that one doesn't break the other.
```

```
        {
            sms s = new sms();
            s.Body = body;
            s.From = from;
            string json = JsonConvert.SerializeObject(s);
            string url = "http://178.62.87.28:700/sendSms";
            var httpContent = new StringContent(json, Encoding.UTF8,
"application/json");
            using (var httpClient = new HttpClient())
            {
                var httpResponse = await httpClient.PostAsync(url, httpContent);

                if (httpResponse.Content != null)
                {
                    var responseContent = await
httpResponse.Content.ReadAsStringAsync();
                }
            }
        }

        /// <summary>
        /// Gets the datetime from a string, calling the DateTimeRoutines class.
        /// </summary>
        /// <param name="input"></param>
        /// <returns></returns>
       [HttpGet("tdt{input}")]
       public DateTime getDateTimeTest(string input)
       {
            DateTime dt;
            input.TryParseDateOrTime(DateTimeRoutines.DateTimeFormat.USA_DATE, out
dt);
            if (dt.Year == 1)
                input.TryParseDateOrTime(DateTimeRoutines.DateTimeFormat.UK_DATE,
out dt);
            return dt;
       }



        /// <summary>
        /// Creates an appointment given the complete time and date and details of
who the appointment is with
        /// Only occurs when the system knows the appointment is available.
        /// </summary>
        /// <param name="a"></param>
        /// <param name="recurring">If a new column needs to be created, it is,
then the entire process is repeated. However, the appointment shouldn't be readded
to the database.</param>
        /// <returns></returns>
       [HttpPost("an")]
```

```
        public bool createNewAppointmentsForAndroidApp([FromBody] Types.appointment
a, bool recurring = false)
        {

            if(!recurring)
            {
                string test = "INSERT INTO appointments (helperId, elderlyId,
dateAndTime, dateAndTimeCreated) VALUES (" + a.helperId + "," + a.elderlyId + ",'"
+ a.dateAndTime.ToString("yyyy-MM-dd H:mm:ss") + "','" +
a.dateCreated.ToString("yyyy-MM-dd H:mm:ss") + "');";
                sqlCommand(false, test); // Insert the appointment into the SQL
database.
            }
            string test2 = sqlCommand(true, "SELECT max(id) FROM appointments;",
1); // Gets the id of the appointment
            a.id = int.Parse(test2.Split('#')[0]);
            appointment b = getAppointmentInformation(a.id); // ensures it works
with a complete appointment
            accountsForElderly ae = new accountsForElderly();
            accountsForHelper ah = new accountsForHelper();
            timetables ttt = new timetables();
            user helper = ah.GetInfo(a.helperId);
            user elderlyPerson = ae.GetInfo(a.elderlyId);

            timetable tt = ttt.getTimetable(helper.timetableId);
            //Check if the week already exists, if it doesn't then it needs to be
added.
            //Get the week beginning of the appointment
            DateTime dt = a.dateAndTime;
            while (dt.DayOfWeek != DayOfWeek.Monday) { dt = dt.AddDays(-1); }
            string toCheck = dt.ToString("dd/MM/yyyy");

            bool found = false;
            week correctWeek = new week(); ;
            for (int i = 0; i < tt.weeks.Count(); i++)
            {
                if (tt.weeks[i].weekBeginning.ToString("dd/MM/yyyy") == toCheck)
                {
                    found = true;
                    correctWeek = tt.weeks[i].week;
                }
            }
            if(found) // We already have a custom week with this week
            {
                int weekId = correctWeek.id;
                day correctDay = new day();
                //GOT THE CORRECT WEEK, now let's get the correct day
                switch (a.dateAndTime.DayOfWeek)
                {
                    case DayOfWeek.Monday:
                        correctDay = correctWeek.monday;
```

```
                    break;
                case DayOfWeek.Tuesday:
                    correctDay = correctWeek.tuesday;
                    break;
                case DayOfWeek.Wednesday:
                    correctDay = correctWeek.wednesday;
                    break;
                case DayOfWeek.Thursday:
                    correctDay = correctWeek.thursday;
                    break;
                case DayOfWeek.Friday:
                    correctDay = correctWeek.friday;
                    break;
                case DayOfWeek.Saturday:
                    correctDay = correctWeek.saturday;
                    break;
                case DayOfWeek.Sunday:
                    correctDay = correctWeek.sunday;
                    break;
                default:
                    correctDay = correctWeek.monday;
                    break;
            }
            int dayId = correctDay.id;
            int indexOfWhichThingToUpdate = a.dateAndTime.Hour / 2; // Which
member of the timesFree array.
            day dayToMakeIt = correctDay; // Gets the correct day
            dayToMakeIt.timesFree[indexOfWhichThingToUpdate] = false; //
changes the value.
            int newId = timetables.addDay(dayToMakeIt); // Doesn't update day,
instead adds a new day
            correctDay = dayToMakeIt; // Changes the day in the week.
            correctDay.id = newId; // Changes the week id.
            switch (a.dateAndTime.DayOfWeek) // updates the week
            {
                case DayOfWeek.Monday:
                    correctWeek.monday = correctDay;
                    break;
                case DayOfWeek.Tuesday:
                    correctWeek.tuesday = correctDay;
                    break;
                case DayOfWeek.Wednesday:
                    correctWeek.wednesday = correctDay;
                    break;
                case DayOfWeek.Thursday:
                    correctWeek.thursday = correctDay;
                    break;
                case DayOfWeek.Friday:
                    correctWeek.friday = correctDay;
                    break;
                case DayOfWeek.Saturday:
```

```
                    correctWeek.saturday = correctDay;
                    break;
                case DayOfWeek.Sunday:
                    correctWeek.sunday = correctDay;
                    break;
                default:
                    correctWeek.monday = correctDay;
                    break;
            }
            timetables.updateWeek(correctWeek);
            return true;

        }
        else // must create a new column of the database and populate it.
        {
            sqlCommand(false, "ALTER TABLE timetables ADD " +
toCheck.Replace('/','_') + " INT NULL;"); // creates a new column
            // toCheck is the weekBeginning eg 01/01/2016 - however the name of
the columns are 01_01_2016 etc.
            string beenReturn = sqlCommand(true, "SELECT id, defaultWeek FROM
timetables", 2); // get's all the columns of timetable
            string[] havingBeenSplit = beenReturn.Split('\n');
            for(int i = 0; i < havingBeenSplit.Length - 1; i++)
            {
                string[] havingBeenSplitMore = havingBeenSplit[i].Split('#');
                int idOfUser = int.Parse(havingBeenSplitMore[0]);
                int defaultWeekIdOfUser = int.Parse(havingBeenSplitMore[1]);
                week newWeek = timetables.getWeek2(defaultWeekIdOfUser); // get
the default week from the database.
                int newId = timetables.addWeek(newWeek); // Adds a copy of the
default week to the database.
                sqlCommand(false, "UPDATE timetables SET " +
toCheck.Replace('/', '_') + " = " + newId + " WHERE id = " + idOfUser); // Updates
the timetable
            }
            week weekToInsert = tt.defaultWeek;
            int id = timetables.addWeek(weekToInsert);
            sqlCommand(false, "UPDATE timetables SET " +
toCheck.Replace('/','_') + " = " + id + " WHERE id = " + tt.id);
            return createNewAppointmentsForAndroidApp(a, true); // Reccurs on
the same process but this time ensuring that the appointment is not readded to the
database of appointments
        }
    }

    /// <summary>
    /// Gets the appointment information
    /// </summary>
    /// <param name="id"></param>
    /// <returns></returns>
   [HttpGetAttribute("gi{id}")]
```

```
        public  Types.appointment getAppointmentInformation(int id)
        {
             string response = sqlCommand(true,"SELECT id, helperId, elderlyId,
dateAndTime, dateAndTimeCreated, ratingId, reviewId FROM appointments WHERE id = '"
+ Convert.ToString(id) + "'", 7);
             appointment appointmentToReturn = new appointment();
             string[] responseSplit = response.Split('#');
             appointmentToReturn.id = Convert.ToInt32(responseSplit[0]);
             appointmentToReturn.helperId = Convert.ToInt32(responseSplit[1]);
             appointmentToReturn.elderlyId = Convert.ToInt32(responseSplit[2]);
             appointmentToReturn.dateAndTime = DateTime.Parse(responseSplit[3]);
             appointmentToReturn.dateCreated = DateTime.Parse(responseSplit[4]);
             int t1 = new int();
             int t2 = new int();
             int.TryParse(responseSplit[5], out t1) ;
             int.TryParse(responseSplit[6], out t2);
             appointmentToReturn.ratingId = t1;
             appointmentToReturn.reviewId = t2;
             return appointmentToReturn;
        }

         /// <summary>
         /// Deletes appointment
         /// </summary>
         /// <param name="id"></param>
         /// <returns></returns>
        [HttpDeleteAttribute("re{id}")]
        public bool removeAppointment(int id)
        {
             sqlCommand(false, "DELETE FROM appointments WHERE id = " +
Convert.ToString(id));
             return true;
        }

         /// <summary>
         /// Gets all the appointments of an elderly person
         /// </summary>
         /// <param name="id"></param>
         /// <returns></returns>
        [HttpGetAttribute("gAE{id}")]
        public List<appointment> appointmentsByElderly(int id)
        {
             string response = sqlCommand(true, "SELECT id, helperId, elderlyId,
dateAndTime, dateAndTimeCreated, ratingId, reviewId FROM appointments WHERE
elderlyId = '" + Convert.ToString(id) + "'", 7);
             string[] lines = response.Split('\n');
             List<appointment> listToReturn = new List<appointment>();
             for (int i = 0; i < lines.Length - 1; i++)
             {
                 string line = lines[i];
                 string[] responseSplit = line.Split('#');
```

```
                appointment appointmentToReturn = new appointment();
                appointmentToReturn.id = Convert.ToInt32(responseSplit[0]);
                appointmentToReturn.helperId = Convert.ToInt32(responseSplit[1]);
                appointmentToReturn.elderlyId = Convert.ToInt32(responseSplit[2]);
                appointmentToReturn.dateAndTime = DateTime.Parse(responseSplit[3]);
                appointmentToReturn.dateCreated = DateTime.Parse(responseSplit[4]);
                appointmentToReturn.ratingId = Convert.ToInt32(responseSplit[5]);
                appointmentToReturn.reviewId = Convert.ToInt32(responseSplit[6]);
                listToReturn.Add(appointmentToReturn);
            }
            return listToReturn;
        }

         /// <summary>
         /// Gets all the appointments of a helper
         /// </summary>
         /// <param name="id"></param>
         /// <returns></returns>
        [HttpGetAttribute("gAH{id}")]
        public List<appointment> appointmentsByHelper(int id)
        {
            string response = sqlCommand(true, "SELECT id, helperId, elderlyId,
dateAndTime, dateAndTimeCreated, ratingId, reviewId FROM appointments WHERE
helperId = '" + Convert.ToString(id) + "'", 7);
            string[] lines = response.Split('\n');
            List<appointment> listToReturn = new List<appointment>();
            for (int i = 0; i < lines.Length - 1; i++)
            {
                string line = lines[i];
                string[] responseSplit = line.Split('#');
                appointment appointmentToReturn = new appointment();
                appointmentToReturn.id = Convert.ToInt32(responseSplit[0]);
                appointmentToReturn.helperId = Convert.ToInt32(responseSplit[1]);
                appointmentToReturn.elderlyId = Convert.ToInt32(responseSplit[2]);
                appointmentToReturn.dateAndTime = DateTime.Parse(responseSplit[3]);
                appointmentToReturn.dateCreated = DateTime.Parse(responseSplit[4]);
                appointmentToReturn.ratingId = Convert.ToInt32(responseSplit[5]);
                appointmentToReturn.reviewId = Convert.ToInt32(responseSplit[6]);
                listToReturn.Add(appointmentToReturn);
            }

            return listToReturn;
        }



    }
}
```

Ratings.cs

```
using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using MySql.Data.MySqlClient;
using MySql.Data.Common;
using MySql.Data.Types;
using TestApi.Backend;
using TestApi.Sql;
using TestApi.Types;
namespace TestApi.Controllers
{

    [Route("api/[Controller]")]
    public class ratings : SQLStuffOnTopOfController
    {
        /// <summary>
        /// Get all ratings of a helper
        /// </summary>
        /// <param name="id"></param>
        /// <returns></returns>
       [HttpGetAttribute("ga{id}")]
       public List<rating> getAllRatingsForUser(int id)
       {
            string listOfAllRatings = sqlCommand(true, "SELECT id, starRating,
helperId FROM ratings WHERE helperId = " + Convert.ToString(id),3);
            string[] splitListOfAllRatings = listOfAllRatings.Split('\n');
            List<rating> ratingsToReturn = new List<rating>();
            for (int i = 0; i < splitListOfAllRatings.Length - 1; i++) //
Parses the ratings into a list of ratings.
            {
                rating a = new rating();
                string[] b = splitListOfAllRatings[i].Split('#');
                a.id = Convert.ToInt32(b[0]);
                a.starRating = Convert.ToInt32(b[1]);
                a.helperId = Convert.ToInt32(b[2]);
                ratingsToReturn.Add(a);
            }
            return ratingsToReturn;
       }

        /// <summary>
        /// Get one specific rating based on the ratingId
        /// </summary>
        /// <param name="ratingId"></param>
        /// <returns></returns>
       [HttpGetAttribute("go{ratingId}")]
       public rating getOneSpecificRating(int ratingId)
```

```
        {
            string oneRating = sqlCommand(true, "SELECT r.id,
r.starRating,r.helperId FROM ratings r WHERE r.id = " +
Convert.ToString(ratingId),3);
            rating toReturn = new rating();
            string[] split = oneRating.Split('#');
            toReturn.id = Convert.ToInt32(split[0]);
            toReturn.starRating = Convert.ToInt32(split[1]);
            toReturn.helperId = Convert.ToInt32(split[2]);
            return toReturn;
        }

        /// <summary>
        /// Add a rating
        /// Appointment id is stored in the position of the ratingId - which
isn't known yet.
        /// </summary>
        /// <param name="newRating"></param>
        /// <returns></returns>
        [HttpPost("ar")]
        public bool addARating([FromBody] rating newRating)
        {
            int appointmentId = newRating.id;
            int rating = newRating.starRating;
            int helperId = newRating.helperId;
            try
            {
                sqlCommand(false, "INSERT INTO ratings (starRating, helperId)
VALUES('" + rating + "','" + helperId + "');");
                sqlCommand(false, "UPDATE appointments SET ratingId = (SELECT
max(id) FROM ratings) WHERE id = '" + appointmentId + "'");
                return true;
            }
            catch
            {
                return false;
            }
        }

        /// <summary>
        /// Update rating
        /// </summary>
        /// <param name="newRating"></param>
        /// <returns></returns>
        [HttpPatchAttribute("ud")]
        public bool updateUserRating([FromBody] rating newRating)
        {
            int ratingId = newRating.id;
```

```
            int rating = newRating.starRating;

            try
            {
                sqlCommand(false, "UPDATE ratings SET starRating = '" +
Convert.ToString(rating) + "' WHERE id = '" + ratingId + "';");
                return true;
            }
            catch
            {
                return false;
            }
        }

         /// <summary>
         /// Delete rating
         /// </summary>
         /// <param name="ratingId"></param>
         /// <returns></returns>
        [HttpDeleteAttribute("rm{ratingId}")]
        public bool deleteUserRating(int ratingId)
        {
            if (Convert.ToInt32(sqlCommand(true, "SELECT count(1) FROM ratings
WHERE id = '" + ratingId + "'", 1).Split('#')[0]) >= 1)
            {
                try
                {
                    sqlCommand(false, "DELETE FROM ratings WHERE id = '" +
Convert.ToString(ratingId) + "'", 1);
                    return true;
                }
                catch
                {
                    return false;
                }
            }
            else
            {
                return false;
            }
        }
    }
}
```

Reviews.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
```

```
using Microsoft.AspNetCore.Mvc;
using MySql.Data.MySqlClient;
using MySql.Data.Common;
using MySql.Data.Types;
using TestApi.Sql;
using TestApi.Types;
namespace TestApi.Controllers
{

    [Route("api/[Controller]")]
    public class reviews : SQLStuffOnTopOfController
    {
        /// <summary>
        /// Get all reviews of a user
        /// </summary>
        /// <param name="id"></param>
        /// <returns></returns>
        [HttpGetAttribute("ga{id}")]
        public List<review> getAllReviewsOfAUser(int id)
        {
            string listOfAllReviews = sqlCommand(true, "SELECT id, review FROM
reviews WHERE helperId = " + Convert.ToString(id), 2);
            string[] splitListOfAllReviews = listOfAllReviews.Split('\n');
            List<review> reviewsToReturn = new List<review>();
            for (int i = 0; i < splitListOfAllReviews.Length - 1; i++) //
Parses the reviews into a list of review objects.
            {
                review a = new review();
                string[] b = splitListOfAllReviews[i].Split('#');
                a.id = Convert.ToInt32(b[0]);
                a.comment = b[1];
                a.helperId = id;
                reviewsToReturn.Add(a);
            }
            return reviewsToReturn;
        }

        /// <summary>
        /// Gets one specific review based on the reviewId
        /// </summary>
        /// <param name="appointmentId"></param>
        /// <returns></returns>
        [HttpGetAttribute("go{appointmentId}")]
        public review getOneSpecificReview(int appointmentId)
        {
            string oneReview = sqlCommand(true, "SELECT r.id, r.review,
r.helperId FROM reviews r WHERE r.id = " + Convert.ToString(appointmentId),
3);
```

```
            review toReturn = new review();
            string[] split = oneReview.Split('#');
            toReturn.id = Convert.ToInt32(split[0]);
            toReturn.comment = split[1];
            toReturn.helperId = Convert.ToInt32(split[2]);
            return toReturn;
        }

        [HttpGet("geta{appointmentId}")]
        public review returnReviewBasedOnAppointmentId(int appointmentId)
        {
            string oneReview = sqlCommand(true, "SELECT r.id, r.review,
r.helperId FROM reviews r JOIN appointments a ON a.reviewId = r.id", 3);
            review toReturn = new review();
            string[] split = oneReview.Split('#');
            toReturn.id = Convert.ToInt32(split[0]);
            toReturn.comment = split[1];
            toReturn.helperId = Convert.ToInt32(split[2]);
            return toReturn;
        }
        /// <summary>
        /// Add a review
        /// AppointmentId is stored in the reviewId location, which is not yet
known.
        /// </summary>
        /// <param name="newReview"></param>
        /// <returns></returns>
        [HttpPost("ar")]
        public bool addAreview([FromBody] review newReview)
       {
            string review = newReview.comment;
            int appointmentId = newReview.id;
            int helperId = newReview.helperId;
            try
            {
                sqlCommand(false, "INSERT INTO reviews (review, helperId)
VALUES('" + review + "','" + helperId + "');");
                sqlCommand(false, "UPDATE appointments SET reviewId = (SELECT
max(id) FROM reviews) WHERE id = '" + appointmentId + "'");
                return true;
            }
            catch
            {
                return false;
            }
        }

        /// <summary>
```

```
        /// Review is updated.
        /// </summary>
        /// <param name="newReview"></param>
        /// <returns></returns>
       [HttpPatchAttribute("ud")]
       public bool updateUserreview([FromBody] review newReview)
       {
            string review = newReview.comment;
            int reviewId = newReview.id;
            try
            {
                sqlCommand(false, "UPDATE reviews SET review = '" +
Convert.ToString(review) + "' WHERE id = '" + reviewId + "';");
                return true;
            }
            catch
            {
                return false;
            }
        }

        /// <summary>
        /// Review is removed.
        /// </summary>
        /// <param name="reviewId"></param>
        /// <returns></returns>
       [HttpDeleteAttribute("rm{reviewId}")]
       public bool deleteUserreview(int reviewId)
       {
            if (Convert.ToInt32(sqlCommand(true, "SELECT count(1) FROM reviews
WHERE id = '" + reviewId + "'", 1).Split('#')[0]) >= 1)
            {
                try
                {
                    sqlCommand(false, "DELETE FROM reviews WHERE id = '" +
Convert.ToString(reviewId) + "'", 1);
                    return true;
                }
                catch
                {
                    return false;
                }
            }
            else
            {
                return false;
            }
        }
```

```
    }

}
```

Timetable.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using MySql.Data.MySqlClient;
using MySql.Data.Common;
using MySql.Data.Types;
using TestApi.Backend;
using TestApi.Sql;
using System.Runtime.Serialization;
using TestApi.Types;
using System.Globalization;

namespace TestApi.Controllers
{
    /// <summary>
    /// Things it needs to do - add a new week to the timetable or update. These
can effectively be classed as the same thing
    ///                         retrieve an existing timetable
    ///                         create a new timetable and return the timetableid,
set the defaultweek to 1
    /// </summary>
    [Route("api/tt")]
    public class timetables : SQLStuffOnTopOfController
    {
        ///
        public int createTimetable()
        {
            sqlCommand(false, "INSERT INTO timetables (defaultWeek) VALUES (1)");
            return int.Parse(sqlCommand(true, "SELECT max(id) FROM timetables",
1).Split('#')[0]);
        }
        /// <summary>
        /// Add a timetable to the system
        /// </summary>
        /// <param name="t"></param>
        /// <param name="doIt"></param>
        [HttpPost("adt")]
        public void addTimetable([FromBody] timetable t, bool doIt = true)
        {
            //First Deal with Default Week
            t.defaultWeek.monday.id = addDay(t.defaultWeek.monday); // add each day
to the database and get their ids
            t.defaultWeek.tuesday.id = addDay(t.defaultWeek.tuesday);
            t.defaultWeek.wednesday.id = addDay(t.defaultWeek.wednesday);
```

```
            t.defaultWeek.thursday.id = addDay(t.defaultWeek.thursday);
            t.defaultWeek.friday.id = addDay(t.defaultWeek.friday);
            t.defaultWeek.saturday.id = addDay(t.defaultWeek.saturday);
            t.defaultWeek.sunday.id = addDay(t.defaultWeek.sunday);

            t.defaultWeek.id = addWeek(t.defaultWeek); // add the week
            sqlCommand(false, "INSERT INTO timetables (defaultWeek) VALUES (" +
t.defaultWeek.id + ");");
            int oldId = t.id;
            t.id = int.Parse(sqlCommand(true, "SELECT max(id) FROM timetables",
1).Split('#')[0]);

            //Then Each Week
            for (int i = 0; i < t.weeks.Count(); i++)
            {
                string nameOfColumn =
t.weeks[i].weekBeginning.ToString("dd/MM/yyyy");
                nameOfColumn = nameOfColumn.Replace('/', '_'); // Gets the name of
the column
                t.weeks[i].week.monday.id = addDay(t.weeks[i].week.monday);
                t.weeks[i].week.tuesday.id = addDay(t.weeks[i].week.tuesday);
                t.weeks[i].week.wednesday.id = addDay(t.weeks[i].week.wednesday);
                t.weeks[i].week.thursday.id = addDay(t.weeks[i].week.thursday);
                t.weeks[i].week.friday.id = addDay(t.weeks[i].week.friday);
                t.weeks[i].week.saturday.id = addDay(t.weeks[i].week.saturday);
                t.weeks[i].week.sunday.id = addDay(t.weeks[i].week.sunday);
                t.weeks[i].week.id = addWeek(t.weeks[i].week);
                try
                {
                    sqlCommand(false,"UPDATE timetables SET " + nameOfColumn + "="
+ t.weeks[i].week.id + " WHERE id = " + t.id + ";");
                    //This only doesn't throw an exception if the column already
exists. If not, the catch is used to create a new column
                }
                catch
                {
                    sqlCommand(false, "ALTER TABLE timetables ADD " + nameOfColumn
+ " INT NULL;"); // Add column
                    string beenReturn = sqlCommand(true, "SELECT id, defaultWeek
FROM timetables", 2);
                    string[] havingBeenSplit = beenReturn.Split('\n');
                    for (int k = 0; k < havingBeenSplit.Length - 1; k++) // Similar
to appointments, this effectively creates a copy of default week for each entry and
populates the column
                    {
                        string[] havingBeenSplitMore =
havingBeenSplit[k].Split('#');
                        int idOfUser = int.Parse(havingBeenSplitMore[0]);
                        int defaultWeekIdOfUser =
int.Parse(havingBeenSplitMore[1]);
                        week newWeek = timetables.getWeek2(defaultWeekIdOfUser);
```

```
                        int newId = timetables.addWeek(newWeek);
                        sqlCommand(false, "UPDATE timetables SET " + nameOfColumn +
" = " + newId + " WHERE id = " + idOfUser);
                    }
                    sqlCommand(false, "UPDATE timetables SET " + nameOfColumn + "="
+ t.weeks[i].week.id + " WHERE id = " + t.id + ";");
                }
            }
            if (doIt) // If set as an update
            {
                sqlCommand(false, "UPDATE accountsForHelpers SET timetableId = " +
t.id + " WHERE timetableId = " + oldId + ";");
            }
        }
        /// <summary>
        /// Get a timetable
        /// </summary>
        /// <param name="i"></param>
        /// <returns></returns>
        [HttpGetAttribute("gt{i}")]
        public  Types.timetable getTimetable(int i)
        {
            string columnNames = sqlCommand(true, "SELECT COLUMN_NAME FROM
INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME = N'timetables'", 1); // Gets all the
column names
            string[] split = columnNames.Split('\n');
            for(int k = 0; k < split.Length - 1; k++)
            {
                split[k] = split[k].Substring(0, split[k].Length - 1); // Remove
spaces between the column names
            }
            int lengthOfInputExpected = split.Length - 1;
            string input = sqlCommand(true, "SELECT * FROM timetables WHERE id = '"
+ i + "';",lengthOfInputExpected);
            string[] inputSplit = input.Split('#');
            TestApi.Types.timetable toReturn = new Types.timetable();
            toReturn.id = int.Parse(inputSplit[0]);
            toReturn.defaultWeek = getWeek(int.Parse(inputSplit[1])); // calls the
get week on the defaultWeek
            for(int j = 2; j < inputSplit.Length - 1; j++) // For each other week
            {
                Types.week week = new Types.week();
                Types.names name = new Types.names();
                if (String.IsNullOrEmpty(inputSplit[j]))
                { } // Incase it is empty
                else
                {
                    split[j] = split[j].Replace('_', '/');
                    name.weekBeginning = DateTime.ParseExact(split[j],
"dd/MM/yyyy", CultureInfo.InvariantCulture); // Parse the column as a datebeginning
date
```

```
                    name.week = getWeek(int.Parse(inputSplit[j]));
                    toReturn.weeks.Add(name); // See the timetable object to
understand what names is.
                }
            }
            return toReturn;
        }
        public static void updateWeek(week input)
        {
            string a = "UPDATE week SET mondayId = " + input.monday.id + ",
tuesdayId = " + input.tuesday.id + ", wednesdayId = " + input.wednesday.id + ",
thursdayId = " + input.thursday.id + ", fridayId = " + input.friday.id + ",
saturdayId = " + input.saturday.id + ", sundayId = " + input.sunday.id + " WHERE id
= " + input.id + ";";
            sqlCommand(false, a);
        }


        public static int addDay(Types.day input)
        {
            // converts a timesFree into an integer array.
            int[] timesFree2 = new int[12];
            for (int k = 0; k < input.timesFree.Length; k++)
            {
                if (input.timesFree[k])
                    timesFree2[k] = 1;
                else
                    timesFree2[k] = 0;
            }
            // Creates the SQL command to be executed
            string a = "INSERT INTO day (a02, a24, a46, a68,
a810,a1012,a1214,a1416,a1618,a1820,a2022,a2224) VALUES(";
            for (int i = 0; i < timesFree2.Length-1; i++)
            {
                a += Convert.ToString(timesFree2[i]) + ",";
            }
            a += Convert.ToString(timesFree2[timesFree2.Length - 1]) + ");";
            sqlCommand(false, a);
            return int.Parse(sqlCommand(true, "SELECT max(id) FROM day",
1).Split('#')[0]); // gets the id of the day just added.
        }


        [HttpPostAttribute("ud")]
        public bool updateDay([FromBody] Types.day input)
        {
            //Updates the day
            int[] timesFree2 = new int[12];
            for (int k = 0; k < input.timesFree.Length; k++)
            {
```

```
                if(input.timesFree[k])
                    timesFree2[k] = 1;
                else
                    timesFree2[k] = 0;
            }
            string a = "UPDATE day SET a02 = " + timesFree2[0] + ", a24 = " +
timesFree2[1] + ", a46 = " + timesFree2[2] + ", a68 = " + timesFree2[3] + ", a810 =
" + timesFree2[4] + ",a1012=" + timesFree2[5] + ",a1214=" + timesFree2[6] +
",a1416=" + timesFree2[7] + ",a1618=" + timesFree2[8] + ",a1820=" + timesFree2[9] +
",a2022=" + timesFree2[10] + ",a2224=" + timesFree2[11] + " WHERE id =" + input.id
+ ";";

            sqlCommand(false, a);
            return true;
        }

        public static int addWeek(Types.week input)
        {
            //Adds a week
            int[] days = new int[7];
            days[0] = input.monday.id;
            days[1] = input.tuesday.id;
            days[2] = input.wednesday.id;
            days[3] = input.thursday.id;
            days[4] = input.friday.id;
            days[5] = input.saturday.id;
            days[6] = input.sunday.id;
            string url = "INSERT INTO week (mondayId, tuesdayId, wednesdayId,
thursdayId, fridayId, saturdayId, sundayId) VALUES (";
            for (int i = 0; i < 6; i++)
                url += Convert.ToString(days[i]) + ",";
            url += Convert.ToString(days[6]) + ")";
            sqlCommand(false, url);
            return int.Parse(sqlCommand(true, "SELECT max(id) FROM week",
1).Split('#')[0]); // returns the weekId of the thing just added
        }

        [HttpGetAttribute("gw{i}")]
        public Types.week getWeek(int i)
        {
            //Defines SQL command, executes it and parses the response into a week
object which can be returned
            string returned = sqlCommand(true, "SELECT id, mondayId, tuesdayId,
wednesdayId, thursdayId, fridayId, saturdayId, sundayId FROM week WHERE id = '" + i
+ "'", 8);
            string[] returnedSplit = returned.Split('#');
            Types.week toReturn = new Types.week();
            toReturn.id = int.Parse(returnedSplit[0]);
            timetables ttt = new timetables();
            toReturn.monday = ttt.getDay(int.Parse(returnedSplit[1]));
            toReturn.tuesday = ttt.getDay(int.Parse(returnedSplit[2]));
```

```
            toReturn.wednesday = ttt.getDay(int.Parse(returnedSplit[3]));
            toReturn.thursday = ttt.getDay(int.Parse(returnedSplit[4]));
            toReturn.friday = ttt.getDay(int.Parse(returnedSplit[5]));
            toReturn.saturday = ttt.getDay(int.Parse(returnedSplit[6]));
            toReturn.sunday = ttt.getDay(int.Parse(returnedSplit[7]));
            return toReturn;
        }

        public static Types.week getWeek2(int i)
        {
            string returned = sqlCommand(true, "SELECT id, mondayId, tuesdayId,
wednesdayId, thursdayId, fridayId, saturdayId, sundayId FROM week WHERE id = '" + i
+ "'", 8);
            string[] returnedSplit = returned.Split('#');
            Types.week toReturn = new Types.week();
            toReturn.id = int.Parse(returnedSplit[0]);
            timetables ttt = new timetables();
            toReturn.monday = ttt.getDay(int.Parse(returnedSplit[1]));
            toReturn.tuesday = ttt.getDay(int.Parse(returnedSplit[2]));
            toReturn.wednesday = ttt.getDay(int.Parse(returnedSplit[3]));
            toReturn.thursday = ttt.getDay(int.Parse(returnedSplit[4]));
            toReturn.friday = ttt.getDay(int.Parse(returnedSplit[5]));
            toReturn.saturday = ttt.getDay(int.Parse(returnedSplit[6]));
            toReturn.sunday = ttt.getDay(int.Parse(returnedSplit[7]));
            return toReturn;
        }

        /// <summary>
        /// Gets the day
        /// </summary>
        /// <param name="i"></param>
        /// <returns></returns>
        [HttpGetAttribute("gd{i}")]
        public Types.day getDay(int i)
        {
            //Defines the SQL command and parses response into a day object.
            Types.day toReturn = new Types.day();
            string[] responseSplit = sqlCommand(true, "SELECT id, a02, a24, a46,
a68, a810, a1012, a1214, a1416, a1618, a1820, a2022, a2224 FROM day WHERE id = '" +
i + "';", 13).Split('#');
            toReturn.id = int.Parse(responseSplit[0]);
            for (int j = 1; j < responseSplit.Length - 1; j++ )
            {
                if (responseSplit[j] == "0")
                    toReturn.timesFree[j - 1] = false;
                else if (responseSplit[j] == "1")
                    toReturn.timesFree[j - 1] = true;
                else
                    toReturn.timesFree[j - 1] = false;
            }
            return toReturn;
```

```
        }


    }

}
```

Main Files

AppSettings.Json

```
{
  "Logging": {
    "IncludeScopes": false,
    "LogLevel": {
      "Default": "Debug",
      "System": "Information",
      "Microsoft": "Information"
    }
  },
  "ApplicationSettings": {
    "TwilioUrl": "https://api.twilio.com/2010-04-01/Accounts/Original Twilio
Id/Messages.json",
    "TwilioId": "Original Twilio Id",
    "TwilioToken": "Original Twilio Token",
    "TwilioPhoneNumber": "+15133277836"
  }
}
```

Program.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Builder;

namespace TestApi
{
    public class Program
    {
        public static void Main(string[] args)
        {
            var host = new WebHostBuilder()
                .UseKestrel()
                .UseContentRoot(Directory.GetCurrentDirectory())
                .UseIISIntegration()
                .UseStartup<Startup>()
                .Build();

            host.Run();
```

```
        }
    }
}
```

Project.json

```
{
  "dependencies": {
    "NETStandard.Library": "1.6.1",
    "Microsoft.AspNetCore.Mvc": "1.0.1",
    "Microsoft.AspNetCore.Routing": "1.0.1",
    "Microsoft.AspNetCore.Server.IISIntegration": "1.0.0",
    "Microsoft.AspNetCore.Server.Kestrel": "1.0.1",
    "Microsoft.Extensions.Configuration.EnvironmentVariables": "1.0.0",
    "Microsoft.Extensions.Configuration.FileExtensions": "1.0.0",
    "Microsoft.Extensions.Configuration.Json": "1.0.0",
    "Microsoft.Extensions.Logging": "1.0.0",
    "Microsoft.Extensions.Logging.Console": "1.0.0",
    "Microsoft.Extensions.Logging.Debug": "1.0.0",
    "Microsoft.Extensions.Options.ConfigurationExtensions": "1.0.0",
    "Mysql.Data.Core": "7.0.4-IR-191",
    "Newtonsoft.Json": "9.0.1"
  },

  "tools": {
    "Microsoft.AspNetCore.Server.IISIntegration.Tools": "1.0.0-preview2-final"
  },

  "frameworks": {
    "netcoreapp1.0": {
      "imports": [
        "dotnet5.6",
        "portable-net45+win8"
      ]
    }

  },

  "buildOptions": {
    "emitEntryPoint": true,
    "preserveCompilationContext": true
  },

  "runtimeOptions": {
    "configProperties": {
      "System.GC.Server": true
    }
  },
  "runtimes": {
    "win7-x64": {}
```

```
  },

  "publishOptions": {
    "include": [
      "wwwroot",
      "**/*.cshtml",
      "appsettings.json",
      "web.config"
    ]
  },

  "scripts": {
    "postpublish": [ "dotnet publish-iis --publish-folder %publish:OutputPath% --
framework %publish:FullTargetFramework%" ]
  }
}
```

Startup.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Logging;

namespace TestApi
{
    public class Startup
    {
        public Startup(IHostingEnvironment env)
        {
            var builder = new ConfigurationBuilder()
                .SetBasePath(env.ContentRootPath)
                .AddJsonFile("appsettings.json", optional: true, reloadOnChange:
true)
                .AddJsonFile($"appsettings.{env.EnvironmentName}.json", optional:
true)
                .AddEnvironmentVariables();
            Configuration = builder.Build();
        }

        public IConfigurationRoot Configuration { get; }

        // This method gets called by the runtime. Use this method to add services
to the container.
        public void ConfigureServices(IServiceCollection services)
        {
```

```
            // Add framework services.
            services.AddMvc();
        }

        // This method gets called by the runtime. Use this method to configure the
HTTP request pipeline.
        public void Configure(IApplicationBuilder app, IHostingEnvironment env,
ILoggerFactory loggerFactory)
        {
            loggerFactory.AddConsole(Configuration.GetSection("Logging"));
            loggerFactory.AddDebug();

            app.UseMvc();
        }
    }
}
```

Appendix B – Android App Code

Activities

MainActivity.cs

```
using System;
using Android.App;
using Android.Content;
using Android.Runtime;
using Android.Views;
using Android.Widget;
using Android.OS;
using Android_Application.Activities; // Allows us to call the other activities
using System.IO;
using RestSharp;
using SQLite;
using Android_Application.Types;
namespace Android_Application
{
    [Activity(Label = "Tech Support", MainLauncher = true, Icon =
"@drawable/icon")] // Defines the name and icon of the app
    public class MainActivity : Activity
    {

        protected override void OnCreate(Bundle bundle)
        {
            base.OnCreate(bundle);
            // Set our view from the "main" layout resource
            RequestWindowFeature(WindowFeatures.NoTitle);
            SetContentView(Resource.Layout.loginOrRegister); // Selects the correct
layout

            // Get our button from the layout resource,
            // and attach an event to it
            Button loginOrRegisterLogin =
FindViewById<Button>(Resource.Id.loginOrRegisterLogin);
            loginOrRegisterLogin.Click += LoginOrRegisterLogin_Click;


            Button loginOrRegisterRegister =
FindViewById<Button>(Resource.Id.loginOrRegisterRegister);
            loginOrRegisterRegister.Click += loginOrRegisterRegister_Click;

        }

        private void LoginOrRegisterLogin_Click(object sender, EventArgs e) //When
you click login button
        {
            StartActivity(typeof(Activities.LoginActivity)); // Starts login
        }
```

```
        private void loginOrRegisterRegister_Click(object sender, EventArgs e)
        {
            StartActivity(typeof(Android_Application.Activities.RegisterActivity));
// Starts register
        }
    }
}
```

AppointmentActivity.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Android.App;
using Android.Content;
using Android.OS;
using Android.Runtime;
using Android.Views;
using Android.Widget;

using RestSharp;
using Newtonsoft.Json;
using Android_Application.Types;

namespace Android_Application.Activities
{
    [Activity(Label = "AppointmentsActivity")]
    public class AppointmentsActivity : Activity
    {
        protected override void OnCreate(Bundle savedInstanceState)
        {
            base.OnCreate(savedInstanceState);
            //Gets the parameters from previous screen
            int appointmentId = Intent.GetIntExtra("appointmentId", 16);
            int currentUser = Intent.GetIntExtra("currentUserId", 20);
            bool currentUserHelper = Intent.GetBooleanExtra("currentUserHelper",
false);

            //Sets up screen
            RequestWindowFeature(WindowFeatures.NoTitle);
            SetContentView(Resource.Layout.appointmentDetailsScreen);

            //Defines views in the screen
            TextView appointmentDetailsDate =
FindViewById<TextView>(Resource.Id.appointmentDetailsDate);
            TextView appointmentDetailsTime =
FindViewById<TextView>(Resource.Id.appointmentDetailsTime);
            TextView appointmentDetailsName =
FindViewById<TextView>(Resource.Id.appointmentDetailsName);
```

```
            TextView appointmentDetailsContactNumber =
FindViewById<TextView>(Resource.Id.appointmentDetailsContactNumber);
            TextView appointmentDetailsFirstLineOfAddress =
FindViewById<TextView>(Resource.Id.appointmentDetailsFirstLineOfAddressOfAppointmen
t);
            TextView appointmentDetailsSecondLineOfAddress =
FindViewById<TextView>(Resource.Id.appointmentDetailsSecondLineOfAddressOfAppointme
nt);
            TextView appointmentDetailsCityAddress =
FindViewById<TextView>(Resource.Id.appointmentDetailsCityAddressOfAppointment);
            TextView appointmentDetailsPostalCode =
FindViewById<TextView>(Resource.Id.appointmentDetailsPostalCodeOfAppointment);
            Button appointmentDetailsOfCancelAppointment =
FindViewById<Button>(Resource.Id.cancelAppointment);
            Button appointmentDetailsSetRating =
FindViewById<Button>(Resource.Id.setRating);
            Button appointmentDetailsSetReview =
FindViewById<Button>(Resource.Id.setReview);

            //Populates the view
            try
            {
                appointment appointmentDetails =
getAppointmentDetails(appointmentId);
                appointmentDetailsDate.Text =
appointmentDetails.dateAndTime.ToString("MMMM dd, yyyy");
                appointmentDetailsTime.Text =
appointmentDetails.dateAndTime.ToString("H:mm");
                user helper = new user();
                user elderly = new user();
                if (currentUserHelper)
                {
                    helper = getUserDetails(currentUser, true);
                    elderly = getUserDetails(appointmentDetails.elderlyId, false);
                    appointmentDetailsName.Text = elderly.firstName + " " +
elderly.surname; // populates name and contact number views
                    appointmentDetailsContactNumber.Text = elderly.telephoneNumber;
                }
                else
                {
                    helper = getUserDetails(appointmentDetails.helperId, true);
                    elderly = getUserDetails(appointmentDetails.elderlyId, false);
                    appointmentDetailsName.Text = helper.firstName + " " +
helper.surname;
                    appointmentDetailsContactNumber.Text = helper.telephoneNumber;
                }

                //Populates address views

                appointmentDetailsFirstLineOfAddress.Text =
elderly.firstLineOfAddress;
```

```
                if (String.IsNullOrEmpty(elderly.secondLineOfAddress))
                    appointmentDetailsSecondLineOfAddress.Visibility =
ViewStates.Gone;
                else
                {
                    appointmentDetailsSecondLineOfAddress.Text =
elderly.secondLineOfAddress;
                }
                appointmentDetailsCityAddress.Text = elderly.city;
                appointmentDetailsPostalCode.Text = elderly.postalCode;

                //Enables cancel appointment if current
                if (appointmentDetails.dateAndTime > DateTime.Now)
                {
                    appointmentDetailsOfCancelAppointment.Enabled = true;
                    appointmentDetailsSetRating.Enabled = false;
                    appointmentDetailsSetReview.Enabled = false;
                }

                //Disables cancel appointment if old - also enables set review and
setrating if the person is an elderly person
                else
                {
                    appointmentDetailsOfCancelAppointment.Enabled = false;
                    if (!currentUserHelper)
                    {
                        appointmentDetailsSetRating.Enabled = true;
                        appointmentDetailsSetReview.Enabled = true;
                    }
                    else
                    {
                        appointmentDetailsSetRating.Enabled = false;
                        appointmentDetailsSetReview.Enabled = false;
                    }
                }

                //Sets up click events
                appointmentDetailsOfCancelAppointment.Click += (sender, e) =>
AppointmentDetailsOfCancelAppointment_Click(sender, e, appointmentId);
                appointmentDetailsSetRating.Click += (sender, e) =>
AppointmentDetailsSetRating_Click(sender, e, appointmentId,
appointmentDetails.ratingId);
                appointmentDetailsSetReview.Click += (sender, e) =>
AppointmentDetailsSetReview_Click(sender, e, appointmentId,
appointmentDetails.reviewId);
            }
            catch
            {
                //If internet is not working
                Toast.MakeText(BaseContext, "Ensure that you are connected to the
internet", ToastLength.Short).Show();
```

```
                StartActivity(typeof(MainActivity));
            }


        }

        private void AppointmentDetailsSetReview_Click(object sender, EventArgs e,
int appointmentId, int reviewId)
        {
            //Goes into set review activity
            var activity = new Intent(this,
typeof(Android_Application.Activities.reviewActivity));
            activity.PutExtra("appointmentId", appointmentId);
            activity.PutExtra("reviewId", reviewId);
            StartActivity(activity);
        }

        private void AppointmentDetailsSetRating_Click(object sender, EventArgs e,
int appointmentId, int ratingId)
        {
            var activity = new Intent(this,
typeof(Android_Application.Activities.RatingsActivity));
            activity.PutExtra("appointmentId", appointmentId);
            activity.PutExtra("ratingId", ratingId);
            StartActivity(activity);
            Finish();
            //throw new NotImplementedException();
        }

        private void AppointmentDetailsOfCancelAppointment_Click(object sender,
EventArgs e, int id)
        {
            try
            {
                string url = "http://178.62.87.28:600/api/app/re" +
Convert.ToString(id);
                var client = new RestClient(url);
                var request = new RestRequest();
                request.Method = Method.DELETE;
                request.AddHeader("Accept", "application/json");
                request.Parameters.Clear();
                client.Execute(request);
                Finish();
            }
            catch
            {
                Toast.MakeText(BaseContext, "Ensure that you are connected to the
internet", ToastLength.Short).Show();
                StartActivity(typeof(MainActivity));
            }
        }
```

```
        private user getUserDetails(int id, bool helper)
        {
            try
            {
                string url = String.Empty;
                if (helper)
                    url = "http://178.62.87.28:600/api/ach/users/gi" +
Convert.ToString(id);
                else
                    url = "http://178.62.87.28:600/api/ace/users/gi" +
Convert.ToString(id);
                var client = new RestClient(url);
                var request = new RestRequest();
                request.Method = Method.GET;
                request.AddHeader("Accept", "application/json");
                request.Parameters.Clear();
                var response = client.Execute(request);
                string result = response.Content;
                user toReturn = JsonConvert.DeserializeObject<user>(result);
                return toReturn;
            }
            catch
            {
                Toast.MakeText(BaseContext, "Ensure that you are connected to the
internet", ToastLength.Short).Show();
                StartActivity(typeof(MainActivity));
                return new user();
            }
        }
        private appointment getAppointmentDetails(int id)
        {
            try
            {
                string url = "http://178.62.87.28:600/api/app/gi" +
Convert.ToString(id);
                var client = new RestClient(url);
                var request = new RestRequest();
                request.Method = Method.GET;
                request.AddHeader("Accept", "application/json");
                request.Parameters.Clear();
                var response = client.Execute(request);
                string result = response.Content;
                appointment toReturn =
JsonConvert.DeserializeObject<appointment>(result);
                return toReturn;
            }
            catch
            {
                Toast.MakeText(BaseContext, "Ensure that you are connected to the
internet", ToastLength.Short).Show();
```

```
                StartActivity(typeof(MainActivity));
                return new appointment();
            }
        }
    }
}
```

bookAppointment.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Android.App;
using Android.Content;
using Android.OS;
using Android.Runtime;
using Android.Views;
using Android.Widget;

using Android_Application.Types;
using RestSharp;
using Newtonsoft.Json;
using Android_Application.Backend;

namespace Android_Application.Activities
{
    [Activity(Label = "bookAppointment")]
    public class bookAppointment : ListActivity
    {
        // Define global variables which can be accessed within the class
        int helperId;
        int elderlyId;
        string[] toDisplayArray;
        List<DateTime> existingWeeksDateTime = new List<DateTime>();
        List<DateTime> originalList;
        List<week> existingWeeksWeek;
        timetable ttOfHelper;
        protected override void OnCreate(Bundle savedInstanceState)
        {
            try
            {
                //Get parameters from previous screen
                base.OnCreate(savedInstanceState);
                helperId = Intent.GetIntExtra("helperId", 8);
                elderlyId = Intent.GetIntExtra("elderlyId", 20);

                //Setup view
                RequestWindowFeature(WindowFeatures.NoTitle);
```

```
                //Get timetable of helper
                ttOfHelper = new timetable();
                ttOfHelper = getTimetableOfHelper(helperId);

                //Populates the screen with available weeks
                List<string> toDisplay = new List<string>();
                existingWeeksWeek = new List<week>();
                for (int i = 0; i < ttOfHelper.weeks.Count(); i++)
                {
                    if (ttOfHelper.weeks[i].weekBeginning < DateTime.Now.AddDays(-
7) && ttOfHelper.weeks[i].weekBeginning < DateTime.Now.AddDays(28)) // Do nothing
if before now or after 6 weeks time
                    { }
                    else
                    {
                        //Since the weekId is necessary to be known, first we add
all the custom weeks

existingWeeksDateTime.Add(ttOfHelper.weeks[i].weekBeginning);
                        existingWeeksWeek.Add(ttOfHelper.weeks[i].week);
                    }
                }
                originalList = new List<DateTime>(existingWeeksDateTime); //
creates a list
                if (existingWeeksDateTime.Count() < 4) // only next four weeks
                {
                    DateTime dt = DateTime.Now.StartOfWeek(DayOfWeek.Monday); //
gets the week beginning date of current week
                    for (int i = 0; i < 4; i++)
                    {
                        if (existingWeeksDateTime.Contains(dt)) // check if the
list already contains the date
                        {
                            //CHECK IF THERE IS ANY FREE TIME
                            int index = originalList.IndexOf(dt);
                            bool freeTime =
checkIfAnyTime(existingWeeksWeek[index]); // if no time, then don't show it.
                            if (!freeTime)
                            {
                                existingWeeksDateTime.RemoveAt(index);
                            }
                            else
                            { }
                        }
                        else // not in custom list
                        {
                            bool freeTime = checkIfAnyTime(ttOfHelper.defaultWeek);
// check if there would be free time
                            if (!freeTime)
                            { }
                            else
```

```
                            { existingWeeksDateTime.Add(dt); } // If possible to
get appointments, then add it to the list to be displayed
                        }

                        dt = dt.AddDays(7);
                    }
                }
                existingWeeksDateTime = SortAscending(existingWeeksDateTime); //
Sort the list of dateTimes
                for (int i = 0; i < existingWeeksDateTime.Count(); i++)
                {
                    toDisplay.Add(existingWeeksDateTime[i].ToString("MMMM dd,
yyyy")); // Set the display format
                }

                toDisplayArray = new string[toDisplay.Count()]; // Array to display
                for (int j = 0; j < toDisplayArray.Length; j++)
                {
                    toDisplayArray[j] = toDisplay[j];
                }
                ListAdapter = new ArrayAdapter<String>(this,
Android.Resource.Layout.SimpleListItem1, toDisplayArray);

            }
            catch
            {
                Toast.MakeText(BaseContext, "Ensure that you are connected to the
internet", ToastLength.Short).Show();
                StartActivity(typeof(MainActivity));
            }
        }
        protected bool checkIfAnyTime(week week)
        {
            // Checks if there is any time in a week
            if(!checkDay(week.monday) && !checkDay(week.tuesday) &&
!checkDay(week.wednesday) && !checkDay(week.thursday) && !checkDay(week.friday) &&
!checkDay(week.saturday) && !checkDay(week.saturday) && !checkDay(week.sunday))
            {
                return false;
            }
            else
            {
                return true;
            }
        }
        /// <summary>
        /// Checks if there is any time in a day
        /// </summary>
        /// <param name="day"></param>
        /// <returns></returns>
        protected bool checkDay(day day)
```

```
        {
            for (int i = 0; i < day.timesFree.Length; i++)
            {
                if (day.timesFree[i])
                    return true;
            }
            return false;
        }

        /// <summary>
        /// Sorts a list of DateTimes in an ascending order - from earliest to
latest.
        /// </summary>
        /// <param name="list"></param>
        /// <returns></returns>
        static List<DateTime> SortAscending(List<DateTime> list)
        {
            list.Sort((a, b) => a.CompareTo(b));
            return list;
        }

        //When an item is clicked
        protected override void OnListItemClick(ListView l, View v, int position,
long id)
        {
            //Firstly, attempts to find the weekId of the item clicked.
            base.OnListItemClick(l, v, position, id);
            string dateTimeOfSelectedItem =
existingWeeksDateTime[position].ToString("dd/MM/yyyy");
            DateTime dateTime = existingWeeksDateTime[position];
            int weekId;
            if(originalList.Contains(dateTime)) // see if custom week
            {
                int index = originalList.IndexOf(dateTime);
                weekId = existingWeeksWeek[index].id;
            }
            else
            {
                weekId = ttOfHelper.defaultWeek.id;
            }

            //starts new activity to select day
            var newActivity = new Intent(this, typeof(bookAppointmentDays));
            newActivity.PutExtra("weekId", weekId);
            newActivity.PutExtra("weekBeginning", dateTimeOfSelectedItem);
            newActivity.PutExtra("helperId", helperId);
            newActivity.PutExtra("elderlyId", elderlyId);
            StartActivity(newActivity);
        }

        private timetable getTimetableOfHelper(int id)
```

```
        {
            //Make call to the WebAPI, and parse JSON input into a 'timetable'
object
            try
            {
                int timetableId = getUserDetails(id, true).timetableId;
                string url = "http://178.62.87.28:600/api/tt/gt" +
Convert.ToString(timetableId);
                var client = new RestClient(url);
                var request = new RestRequest();
                request.Method = Method.GET;
                request.AddHeader("Accept", "application/json");
                request.Parameters.Clear();
                var response = client.Execute(request);
                string result = response.Content;
                timetable toReturn =
JsonConvert.DeserializeObject<timetable>(result);
                return toReturn;
            }
            catch
            {
                Toast.MakeText(BaseContext, "Ensure that you are connected to the
internet", ToastLength.Short).Show();
                StartActivity(typeof(MainActivity));
                return new timetable();
            }
        }
        private user getUserDetails(int id, bool helper)
        {
            //Make call to the WebAPI, and parse JSON input into a 'user' object
            try
            {
                string url = String.Empty;
                if (helper)
                    url = "http://178.62.87.28:600/api/ach/users/gi" +
Convert.ToString(id);
                else
                    url = "http://178.62.87.28:600/api/ace/users/gi" +
Convert.ToString(id);
                var client = new RestClient(url);
                var request = new RestRequest();
                request.Method = Method.GET;
                request.AddHeader("Accept", "application/json");
                request.Parameters.Clear();
                var response = client.Execute(request);
                string result = response.Content;
                user toReturn = JsonConvert.DeserializeObject<user>(result);
                return toReturn;
            }
            catch
            {
```

```
                Toast.MakeText(BaseContext, "Ensure that you are connected to the
internet", ToastLength.Short).Show();
                StartActivity(typeof(MainActivity));
                return new user();
            }
        }
    }
}
```

bookAppointmentDays.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Android.App;
using Android.Content;
using Android.OS;
using Android.Runtime;
using Android.Views;
using Android.Widget;

using RestSharp;
using Android_Application.Backend;
using Android_Application.Types;
using Newtonsoft.Json;

namespace Android_Application.Activities
{
    [Activity(Label = "bookAppointmentDays")]
    public class bookAppointmentDays : ListActivity
    {
        week weekToBeDisplayed;
        int helperId;
        int elderlyId;
        DateTime dtOfWeekBeginning;
        List<string> listOfDisplayed;
        protected override void OnCreate(Bundle savedInstanceState)
        {
            try
            {
                base.OnCreate(savedInstanceState);
                weekToBeDisplayed = getWeek(Intent.GetIntExtra("weekId", 1));
                helperId = Intent.GetIntExtra("helperId", 8);
                elderlyId = Intent.GetIntExtra("elderlyId", 20);
                dtOfWeekBeginning =
DateTime.Parse(Intent.GetStringExtra("weekBeginning"));
                listOfDisplayed = new List<string>();
                //MONDAY
```

```
                if (checkDay(weekToBeDisplayed.monday) && DateTime.Now <=
dtOfWeekBeginning.AddDays(1))
                {

listOfDisplayed.Add(dtOfWeekBeginning.AddDays(0).DayOfWeek.ToString() + " " +
dtOfWeekBeginning.AddDays(0).ToString("MMMM dd, yyyy"));
                }
                //TUESDAY
                if (checkDay(weekToBeDisplayed.tuesday) && DateTime.Now <=
dtOfWeekBeginning.AddDays(2))
                {

listOfDisplayed.Add(dtOfWeekBeginning.AddDays(1).DayOfWeek.ToString() + " " +
dtOfWeekBeginning.AddDays(1).ToString("MMMM dd, yyyy"));
                }
                //WEDNESDAY
                if (checkDay(weekToBeDisplayed.wednesday) && DateTime.Now <=
dtOfWeekBeginning.AddDays(3))
                {

listOfDisplayed.Add(dtOfWeekBeginning.AddDays(2).DayOfWeek.ToString() + " " +
dtOfWeekBeginning.AddDays(2).ToString("MMMM dd, yyyy"));
                }
                //THURSDAY
                if (checkDay(weekToBeDisplayed.thursday) && DateTime.Now <=
dtOfWeekBeginning.AddDays(4))
                {

listOfDisplayed.Add(dtOfWeekBeginning.AddDays(3).DayOfWeek.ToString() + " " +
dtOfWeekBeginning.AddDays(3).ToString("MMMM dd, yyyy"));
                }
                //FRIDAY
                if (checkDay(weekToBeDisplayed.friday) && DateTime.Now <=
dtOfWeekBeginning.AddDays(5))
                {

listOfDisplayed.Add(dtOfWeekBeginning.AddDays(4).DayOfWeek.ToString() + " " +
dtOfWeekBeginning.AddDays(4).ToString("MMMM dd, yyyy"));
                }
                //SATURDAY
                if (checkDay(weekToBeDisplayed.saturday) && DateTime.Now <=
dtOfWeekBeginning.AddDays(6))
                {

listOfDisplayed.Add(dtOfWeekBeginning.AddDays(5).DayOfWeek.ToString() + " " +
dtOfWeekBeginning.AddDays(5).ToString("MMMM dd, yyyy"));
                }
                //SUNDAY
                if (checkDay(weekToBeDisplayed.sunday) && DateTime.Now <=
dtOfWeekBeginning.AddDays(7))
                {
```

```
listOfDisplayed.Add(dtOfWeekBeginning.AddDays(6).DayOfWeek.ToString() + " " +
dtOfWeekBeginning.AddDays(6).ToString("MMMM dd, yyyy"));
                }

                string[] thingsToBeDisplayed = listOfDisplayed.ToArray();
                if (listOfDisplayed.Count() == 0)
                {
                    Toast.MakeText(BaseContext, "No times available on this week",
ToastLength.Short).Show();
                    Finish();

                }
                RequestWindowFeature(WindowFeatures.NoTitle);
                ListAdapter = new ArrayAdapter<String>(this,
Android.Resource.Layout.SimpleListItem1, thingsToBeDisplayed);
            }
            catch
            {
                Toast.MakeText(BaseContext, "Ensure that you are connected to the
internet", ToastLength.Short).Show();
                StartActivity(typeof(MainActivity));
            }

            // Create your application here
        }
        protected week getWeek(int id)
        {
            try
            {
                string url = "http://178.62.87.28:600/api/tt/gw" +
Convert.ToString(id);
                var client = new RestClient(url);
                var request = new RestRequest();
                request.Method = Method.GET;
                request.AddHeader("Accept", "application/json");
                request.Parameters.Clear();
                var response = client.Execute(request);
                string result = response.Content;
                week toReturn = JsonConvert.DeserializeObject<week>(result);
                return toReturn;
            }
            catch
            {
                Toast.MakeText(BaseContext, "Ensure that you are connected to the
internet", ToastLength.Short).Show();
                StartActivity(typeof(MainActivity));
                return new week();
            }
        }
        protected bool checkDay(day day)
```

```
        {
            for (int i = 0; i < day.timesFree.Length; i++)
            {
                if (day.timesFree[i])
                    return true;
            }
            return false;
        }
        protected override void OnListItemClick(ListView l, View v, int position,
long id)
        {
            base.OnListItemClick(l, v, position, id);
            //helperId
            //elderlyId
            //date
            //dayId
            int dayId = 1;
            int numberToAdvance = 0;
            string a = listOfDisplayed[position];
            if(a.Contains("Monday"))
            {
                dayId = weekToBeDisplayed.monday.id;
                numberToAdvance = 0;
            }
            else if (a.Contains("Tuesday"))
            {
                dayId = weekToBeDisplayed.tuesday.id;
                numberToAdvance = 1;
            }
            else if (a.Contains("Wednesday"))
            {
                dayId = weekToBeDisplayed.wednesday.id;
                numberToAdvance = 2;
            }
            else if (a.Contains("Thursday"))
            {
                dayId = weekToBeDisplayed.thursday.id;
                numberToAdvance = 3;
            }
            else if (a.Contains("Friday"))
            {
                dayId = weekToBeDisplayed.friday.id;
                numberToAdvance = 4;
            }
            else if (a.Contains("Saturday"))
            {
                dayId = weekToBeDisplayed.saturday.id;
                numberToAdvance = 5;
            }
            else if (a.Contains("Sunday"))
            {
```

```
                dayId = weekToBeDisplayed.sunday.id;
                numberToAdvance = 6;
            }
            DateTime dayToSend = dtOfWeekBeginning.AddDays(numberToAdvance);
            string dateTimeToSend = dayToSend.ToString("dd/MM/yyyy");
            var newActivity = new Intent(this, typeof(bookAppointmentTimes));
            newActivity.PutExtra("dayId", dayId);
            newActivity.PutExtra("helperId", helperId);
            newActivity.PutExtra("elderlyId", elderlyId);
            newActivity.PutExtra("dateTime", dateTimeToSend);

            StartActivity(newActivity);
        }


    }
}
```

bookAppointmentDetailsConfirm.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Android.App;
using Android.Content;
using Android.OS;
using Android.Runtime;
using Android.Views;
using Android.Widget;

using RestSharp;
using Android_Application.Types;
using Android_Application.Backend;
using Newtonsoft.Json;
using System.Globalization;

namespace Android_Application.Activities
{
    [Activity(Label = "bookAppointmentDetailsConfirm")]
    public class bookAppointmentDetailsConfirm : Activity
    {
        //Global variables used throughout the program
        int helperId;
        int elderlyId;
        int index;
        int dayId;
        DateTime dateTimeOfAppointment;

        protected override void OnCreate(Bundle savedInstanceState)
        {
```

```
            try
            {
                //Sets up the view
                base.OnCreate(savedInstanceState);
                RequestWindowFeature(WindowFeatures.NoTitle);
                SetContentView(Resource.Layout.appointmentDetails);

                //Gets the parameters passed when the activity was called
                helperId = Intent.GetIntExtra("helperId", 8);
                elderlyId = Intent.GetIntExtra("elderlyId", 20);
                dayId = Intent.GetIntExtra("dayId", 1);
                index = Intent.GetIntExtra("index", 1);
                dateTimeOfAppointment =
DateTime.ParseExact(Intent.GetStringExtra("dateTime"), "dd/MM/yyyy HH:mm",
CultureInfo.InvariantCulture);

                //Populates the elements of the view
                TextView nameOfHelper =
FindViewById<TextView>(Resource.Id.nameOfHelper);
                TextView dateOfAppointment =
FindViewById<TextView>(Resource.Id.dateOfAppointment);
                TextView timeOfAppointment =
FindViewById<TextView>(Resource.Id.timeOfAppointment);
                Button submitButton =
FindViewById<Button>(Resource.Id.confirmAppointment);
                submitButton.Click += SubmitButton_Click;
                user helper = getUserDetails(helperId, true);
                nameOfHelper.Text = helper.firstName + " " + helper.surname;
                dateOfAppointment.Text = dateTimeOfAppointment.ToString("MMMM dd,
yyyy");
                timeOfAppointment.Text = dateTimeOfAppointment.ToString("HH:mm");
            }
            catch
            {
                Toast.MakeText(BaseContext, "Ensure that you are connected to the
internet", ToastLength.Short).Show();
                StartActivity(typeof(MainActivity));
            }
        }

        private void SubmitButton_Click(object sender, EventArgs e)
        {
            //Submit button is clicked - appointment is to be booked.
            try
            {
                appointment newAppointment = new appointment();
                newAppointment.dateAndTime = dateTimeOfAppointment;
                newAppointment.dateCreated = DateTime.Now;
                newAppointment.elderlyId = elderlyId;
                newAppointment.helperId = helperId;
                string url = String.Empty;
```

```
                string data = JsonConvert.SerializeObject(newAppointment);
                url = "http://178.62.87.28:600/api/app/an";
                var client = new RestClient(url);
                var request = new RestRequest();
                request.Method = Method.POST;
                request.AddHeader("Accept", "application/json");
                request.Parameters.Clear();
                request.AddParameter("application/json", data,
ParameterType.RequestBody);
                var response = client.Execute(request);
                string result = response.Content;

                //Goes into welcome elderly with a sentBack parameter - so it is
passed to the listOfAppointments
                var newActivity = new Intent(this, typeof(welcomeElderly));
                newActivity.PutExtra("currentUserId", elderlyId);
                newActivity.PutExtra("sentBack", true);
                StartActivity(newActivity);
            }
            catch
            {
                Toast.MakeText(BaseContext, "Ensure that you are connected to the
internet", ToastLength.Short).Show();
                StartActivity(typeof(MainActivity));
            }

        }
        private user getUserDetails(int id, bool helper)
        {
            //Make call to the WebAPI, and parse JSON input into a 'user' object
            try
            {
                string url = String.Empty;
                if (helper)
                    url = "http://178.62.87.28:600/api/ach/users/gi" +
Convert.ToString(id);
                else
                    url = "http://178.62.87.28:600/api/ace/users/gi" +
Convert.ToString(id);
                var client = new RestClient(url);
                var request = new RestRequest();
                request.Method = Method.GET;
                request.AddHeader("Accept", "application/json");
                request.Parameters.Clear();
                var response = client.Execute(request);
                string result = response.Content;
                user toReturn = JsonConvert.DeserializeObject<user>(result);
                return toReturn;
            }
            catch
            {
```

```
                Toast.MakeText(BaseContext, "Ensure that you are connected to the
internet", ToastLength.Short).Show();
                StartActivity(typeof(MainActivity));
                return new user();
            }
        }
    }
}
```

bookAppointmentTimes.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Android.App;
using Android.Content;
using Android.OS;
using Android.Runtime;
using Android.Views;
using Android.Widget;

using RestSharp;
using Newtonsoft.Json;
using Android_Application.Types;
using Android_Application.Backend;
using System.Globalization;

namespace Android_Application.Activities
{
    [Activity(Label = "bookAppointmentTimes")]
    public class bookAppointmentTimes : ListActivity
    {
        //Global variables which are used throughout the program
        int helperId;
        int elderlyId;
        DateTime dt;
        day day;
        int dayId;
        List<string> times = new List<string>();
        protected override void OnCreate(Bundle savedInstanceState)
        {
            try
            {
                //Gets all the parameters which are passed to the activity
                base.OnCreate(savedInstanceState);
                helperId = Intent.GetIntExtra("helperId", 8);
                elderlyId = Intent.GetIntExtra("elderlyId", 20);
                dayId = Intent.GetIntExtra("dayId", 1);

                //Populates the listView
```

```
                /*
                 * For each time, check if it free and check if it is in the past.
                 * */
                day = getDay(dayId);
                dt = DateTime.Parse(Intent.GetStringExtra("dateTime"));

                for (int i = 0; i < day.timesFree.Count(); i++)
                {
                    if (day.timesFree[i])
                    {
                        DateTime dt2 = new DateTime();
                        dt2 = dt.AddHours(2 * i);
                        if (DateTime.Now < dt2)
                        {
                            times.Add(Convert.ToString(2 * i) + ":00 to " +
Convert.ToString(2 * (i + 1)) + ":00");
                        }
                    }
                    else
                    { }
                }
                string[] timesForList = times.ToArray();
                if (times.Count() == 0)
                {
                    //If no available times
                    Toast.MakeText(BaseContext,"No times available on this day -
Please try a different day",ToastLength.Long).Show();
                    Finish();

                }
                RequestWindowFeature(WindowFeatures.NoTitle);
                ListAdapter = new ArrayAdapter<String>(this,
Android.Resource.Layout.SimpleListItem1, timesForList);
            }
            catch
            {
                Toast.MakeText(BaseContext, "Ensure that you are connected to the
internet", ToastLength.Short).Show();
                StartActivity(typeof(MainActivity));
            }
        }
        protected day getDay (int id)
        {
            //Make call to the WebAPI, and parse JSON input into a 'day' object
            try
            {
                string url = "http://178.62.87.28:600/api/tt/gd" +
Convert.ToString(id);
                var client = new RestClient(url);
                var request = new RestRequest();
                request.Method = Method.GET;
```

```
                request.AddHeader("Accept", "application/json");
                request.Parameters.Clear();
                var response = client.Execute(request);
                string result = response.Content;
                day toReturn = JsonConvert.DeserializeObject<day>(result);
                return toReturn;
            }
            catch
            {
                Toast.MakeText(BaseContext, "Ensure that you are connected to the
internet", ToastLength.Short).Show();
                StartActivity(typeof(MainActivity));
                return new day();
            }
        }
        protected override void OnListItemClick(ListView l, View v, int position,
long id)
        {
            //When an item is clicked, find the time which was clicked
            base.OnListItemClick(l, v, position, id);
            string a = times[position];
            string b = a.Split(' ')[0];
            DateTime dateTimeToSend = dt.AddHours(DateTime.ParseExact(b,"HH:mm",
CultureInfo.InvariantCulture).Hour);

            //Start activity to confirm details of an appointment and allow them to
confirm it.
            var newActivity = new Intent(this,
typeof(bookAppointmentDetailsConfirm));
            newActivity.PutExtra("dayId", dayId);
            newActivity.PutExtra("helperId", helperId);
            newActivity.PutExtra("elderlyId", elderlyId);
            string x = dateTimeToSend.ToString("dd/MM/yyyy HH:mm");
            newActivity.PutExtra("dateTime", x);
            StartActivity(newActivity);

            //When returned, pass back to previous screen
            Finish();
        }
    }
}
```

bookNewAppointment.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Android.App;
using Android.Content;
```

```
using Android.OS;
using Android.Runtime;
using Android.Views;
using Android.Widget;

using RestSharp;
using Android_Application.Types;
using Newtonsoft.Json;

namespace Android_Application.Activities
{
    [Activity(Label = "bookNewAppointment")]
    public class bookNewAppointment : ListActivity
    {
        //Global variables used throughout the code
        public user[] listOfPeopleNearby;
        int currentUserId;
        protected override void OnCreate(Bundle savedInstanceState)
        {
            try
            {
                //Gets parameters from the previous activity.
                base.OnCreate(savedInstanceState);
                currentUserId = Intent.GetIntExtra("currentUserId", 20);

                //Gets list of nearby people
                listOfPeopleNearby = getListOfNearbyUsers(currentUserId);
                int x = 0;
                try
                {
                    if (listOfPeopleNearby.Length < 100)
                        x = listOfPeopleNearby.Length;
                    else
                        x = 100;
                }
                //limits the view to 100 people
                catch
                {
                    Toast.MakeText(BaseContext, "Ensure that you are connected to
the internet", ToastLength.Short).Show();
                    StartActivity(typeof(MainActivity));
                }
                string[] thingsToShow = new string[x];
                for (int i = 0; i < x; i++)
                {
                    //For each person to display, show name and their average
rating (to the nearest tenth)
                    string temp = listOfPeopleNearby[i].firstName + " " +
listOfPeopleNearby[i].surname;
                    double averageRating =
getAverageRating(listOfPeopleNearby[i].id);
```

```
                    if (!Double.IsNaN(averageRating))
                        temp += " (" + Convert.ToString(Math.Round(averageRating,
1)) + "/10)";
                    thingsToShow[i] = temp;
                }
                RequestWindowFeature(WindowFeatures.NoTitle);
                ListAdapter = new ArrayAdapter<String>(this,
Android.Resource.Layout.SimpleListItem1, thingsToShow);
            }
            catch
            {
                Toast.MakeText(BaseContext, "Ensure that you are connected to the
internet", ToastLength.Short).Show();
                StartActivity(typeof(MainActivity));
            }
        }
        protected override void OnListItemClick(ListView l, View v, int position,
long id)
        {
            //If item is clicked

            base.OnListItemClick(l, v, position, id);
            user selected = listOfPeopleNearby[position];

            //Start activity of seeing more details about that person
            var newActivity = new Intent(this, typeof(helperDetails));
            newActivity.PutExtra("currentUserId", currentUserId);
            newActivity.PutExtra("helperId", selected.id);
            StartActivity(newActivity);
        }


        protected user[] getListOfNearbyUsers(int id)
        {
            //Make call to the WebAPI, and parse JSON input into a list of the
'user' object
            try
            {
                string url = "http://178.62.87.28:600/api/ach/gl" +
Convert.ToString(id);
                var client = new RestClient(url);
                var request = new RestRequest();
                request.Method = Method.GET;
                request.AddHeader("Accept", "application/json");
                request.Parameters.Clear();
                var response = client.Execute(request);
                string result = response.Content;
                user[] toReturn = JsonConvert.DeserializeObject<user[]>(result);
                return toReturn;
            }
            catch
```

```
            {
                Toast.MakeText(BaseContext, "Ensure that you are connected to the
internet", ToastLength.Short).Show();
                StartActivity(typeof(MainActivity));
                user[] toReturn = new user[1];
                return toReturn;
            }
        }

        protected double getAverageRating(int id)
        {
            //Make call to the WebAPI, getting a list of the ratings. Then gets the
average rating from this.
            try
            {
                string url = "http://178.62.87.28:600/api/ratings/ga" +
Convert.ToString(id);
                var client = new RestClient(url);
                var request = new RestRequest();
                request.Method = Method.GET;
                request.AddHeader("Accept", "application/json");
                request.Parameters.Clear();
                var response = client.Execute(request);
                string result = response.Content;
                try
                {
                    rating[] toReturn =
JsonConvert.DeserializeObject<rating[]>(result);
                    double x = 0;
                    for (int i = 0; i < toReturn.Length; i++)
                    {
                        x += toReturn[i].starRating;
                    }
                    x = x / toReturn.Length;
                    return x;
                }
                catch
                {
                    return 11;
                }
            }
            catch
            {
                Toast.MakeText(BaseContext, "Ensure that you are connected to the
internet", ToastLength.Short).Show();
                StartActivity(typeof(MainActivity));
                return 0;
            }

        }
        private appointment[] returnListOfAppointments(int id, bool helper)
```

```
        {
            ////Make call to the WebAPI, and parse JSON input into a list of
'appointments' object
            try
            {
                string url = "http://178.62.87.28:600/api/app/g";
                if (helper)
                    url += "AH" + Convert.ToString(id);
                else
                    url += "AE" + Convert.ToString(id);
                var client = new RestClient(url);
                var request = new RestRequest();
                request.Method = Method.GET;
                request.AddHeader("Accept", "application/json");
                request.Parameters.Clear();
                var response = client.Execute(request);
                string result = response.Content;
                appointment[] toReturn =
JsonConvert.DeserializeObject<appointment[]>(result);
                return toReturn;
            }
            catch
            {
                Toast.MakeText(BaseContext, "Ensure that you are connected to the
internet", ToastLength.Short).Show();
                StartActivity(typeof(MainActivity));
                appointment[] a = new appointment[1];
                return a;
            }
        }
        private user getUserDetails(int id, bool helper)
        {
            //Make call to the WebAPI, and parse JSON input into a 'user' object
            try
            {
                string url = String.Empty;
                if (helper)
                    url = "http://178.62.87.28:600/api/ach/users/gi" +
Convert.ToString(id);
                else
                    url = "http://178.62.87.28:600/api/ace/users/gi" +
Convert.ToString(id);
                var client = new RestClient(url);
                var request = new RestRequest();
                request.Method = Method.GET;
                request.AddHeader("Accept", "application/json");
                request.Parameters.Clear();
                var response = client.Execute(request);
                string result = response.Content;
                user toReturn = JsonConvert.DeserializeObject<user>(result);
                return toReturn;
```

```
            }
            catch
            {
                Toast.MakeText(BaseContext, "Ensure that you are connected to the
internet", ToastLength.Short).Show();
                StartActivity(typeof(MainActivity));
                return new user();
            }
        }
        private appointment getAppointmentDetails(int id)
        {
            try
            {

                string url = "http://178.62.87.28:600/api/app/gi" +
Convert.ToString(id);
                var client = new RestClient(url);
                var request = new RestRequest();
                request.Method = Method.GET;
                request.AddHeader("Accept", "application/json");
                request.Parameters.Clear();
                var response = client.Execute(request);
                string result = response.Content;
                appointment toReturn =
JsonConvert.DeserializeObject<appointment>(result);
                return toReturn;
            }
            catch
            {
                Toast.MakeText(BaseContext, "Ensure that you are connected to the
internet", ToastLength.Short).Show();
                StartActivity(typeof(MainActivity));
                return new appointment();
            }
        }
    }
}
```

helperDetails.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Android.App;
using Android.Content;
using Android.OS;
using Android.Runtime;
using Android.Views;
using Android.Widget;
```

```
using RestSharp;
using Android_Application.Types;
using Newtonsoft.Json;
using Android_Application.Backend;
namespace Android_Application.Activities
{
    [Activity(Label = "helperDetails")]
    public class helperDetails : Activity
    {
        int currentUserId;
        int helperId;
        protected override void OnCreate(Bundle savedInstanceState)
        {
            base.OnCreate(savedInstanceState);
            //Gets parameters from the previous activity
            currentUserId = Intent.GetIntExtra("currentUserId", 20);
            helperId = Intent.GetIntExtra("helperId", 2);
            RequestWindowFeature(WindowFeatures.NoTitle);
            SetContentView(Resource.Layout.helperDetails);

            //Sets up views
            user helper = getUserDetails(helperId, true);
            TextView name = FindViewById<TextView>(Resource.Id.helperDetailsName);
            TextView firstLine =
FindViewById<TextView>(Resource.Id.helperDetailsFirstLine);
            TextView secondLine =
FindViewById<TextView>(Resource.Id.helperDetailsSecondLine);
            TextView city = FindViewById<TextView>(Resource.Id.helperDetailsCity);
            TextView postalCode =
FindViewById<TextView>(Resource.Id.helperDetailsPostcode);
            TextView country =
FindViewById<TextView>(Resource.Id.helperDetailsCountry);
            TextView telephone =
FindViewById<TextView>(Resource.Id.helperDetailsContactNumber);
            Button seeReviews =
FindViewById<Button>(Resource.Id.helperDetailsSeeReviews);
            Button bookAppointment =
FindViewById<Button>(Resource.Id.helperDetailsBookAppointment);

            //Populates them
            try
            {
                name.Text = helper.firstName + " " + helper.surname;
                firstLine.Text = helper.firstLineOfAddress;
                if (!String.IsNullOrEmpty(helper.secondLineOfAddress))
                    secondLine.Text = helper.secondLineOfAddress;
                else
                    secondLine.Visibility = ViewStates.Gone;
                city.Text = helper.city;
                country.Text = helper.country;
                postalCode.Text = helper.postalCode;
```

```
                telephone.Text = helper.telephoneNumber;
                if (reviewForUser(helper.id))
                    seeReviews.Enabled = true;
                else
                    seeReviews.Enabled = false;

                bookAppointment.Click += BookAppointment_Click;
                seeReviews.Click += SeeReviews_Click;
            }
            catch
            {
                Toast.MakeText(BaseContext, "Ensure that you are connected to the
internet", ToastLength.Short).Show();
                StartActivity(typeof(MainActivity));
            }


        }

        private void SeeReviews_Click(object sender, EventArgs e)
        {
            //If see review is clicked, goes into activity to see reviews
            var newActivity = new Intent(this,
typeof(Android_Application.Activities.SeeAllReviewOfHelper));
            newActivity.PutExtra("helperId", helperId);
            StartActivity(newActivity);
            this.Recreate();
        }

        private void BookAppointment_Click(object sender, EventArgs e)
        {
            //Goes into the activity to book an appointment with this helper
            var newActivity = new Intent(this, typeof(bookAppointment));
            newActivity.PutExtra("helperId", helperId);
            newActivity.PutExtra("elderlyId", currentUserId);
            StartActivity(newActivity);
        }

        private bool reviewForUser(int id)
        {
            //Make call to the WebAPI, and parse JSON input into a 'review' object.
Then it returns a true if any of them is a 'real' review (not a null) or false if
there are no reviews
            //Therefore, if there are no reviews, the button is not enabled.
            try
            {
                string url = "http://178.62.87.28:600/api/reviews/ga" +
Convert.ToString(id);
                var client = new RestClient(url);
                var request = new RestRequest();
                request.Method = Method.GET;
```

```
                request.AddHeader("Accept", "application/json");
                request.Parameters.Clear();
                var response = client.Execute(request);
                string result = response.Content;
                if (result == "[]")
                    return false;
                else
                    return true;
            }
            catch
            {
                Toast.MakeText(BaseContext, "Ensure that you are connected to the
internet", ToastLength.Short).Show();
                StartActivity(typeof(MainActivity));
                return false;
            }
        }
        private user getUserDetails(int id, bool helper)
        {
            //Make call to the WebAPI, and parse JSON input into a 'user' object
            try
            {
                string url = String.Empty;
                if (helper)
                    url = "http://178.62.87.28:600/api/ach/users/gi" +
Convert.ToString(id);
                else
                    url = "http://178.62.87.28:600/api/ace/users/gi" +
Convert.ToString(id);
                var client = new RestClient(url);
                var request = new RestRequest();
                request.Method = Method.GET;
                request.AddHeader("Accept", "application/json");
                request.Parameters.Clear();
                var response = client.Execute(request);
                string result = response.Content;
                user toReturn = JsonConvert.DeserializeObject<user>(result);
                return toReturn;
            }
            catch
            {
                Toast.MakeText(BaseContext, "Ensure that you are connected to the
internet", ToastLength.Short).Show();
                StartActivity(typeof(MainActivity));
                return new user();
            }
        }
    }
}
```

listOfAppointments.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Android.App;
using Android.Content;
using Android.OS;
using Android.Runtime;
using Android.Views;
using Android.Widget;

using RestSharp;
using Android_Application.Types;
using Newtonsoft.Json;

namespace Android_Application.Activities
{
    [Activity(Label = "listOfAppointments")]
    public class listOfAppointments : ListActivity
    {
        //Global variables
        appointment[] listOfAppointments2;
        int currentUserId;
        bool currentUserHelper;

        protected override void OnCreate(Bundle savedInstanceState)
        {
            //Gets parameters from previous activity
            base.OnCreate(savedInstanceState);
            currentUserId = Intent.GetIntExtra("currentUserId", 20);
            currentUserHelper = Intent.GetBooleanExtra("currentUserHelper", false);
            listOfAppointments2 = returnListOfAppointments(currentUserId,
currentUserHelper);

            //Populates the list
            try
            {
                //Sorts the list into order of time
                Array.Sort(listOfAppointments2, delegate (appointment x,
appointment y) { return x.dateAndTime.CompareTo(y.dateAndTime); });
                List<appointment> newList =
listOfAppointments2.OfType<appointment>().ToList();
                List<appointment> secondList = new List<appointment>();

                //Get list of all appointments which are old and get them at the
beginning
                while (newList.Count() > 0 && newList[0].dateAndTime <
DateTime.Now)
                {
```

```
                    secondList.Insert(0, newList[0]);
                    newList.RemoveAt(0);
                }

                List<appointment> thirdList = new List<appointment>();
                thirdList = newList.Concat(secondList).ToList();

                //Therefore, order is now... nearest to latest in future, then just
gone backwards in history
                listOfAppointments2 = thirdList.ToArray();
                string[] thingsToShowOnListView = new
string[listOfAppointments2.Length];
                for (int i = 0; i < listOfAppointments2.Length; i++)
                {
                    //Gets the exact details of what to show on the view about the
appointment
                    string name = String.Empty;
                    user x = new user();
                    if (currentUserHelper)
                        x = getUserDetails(listOfAppointments2[i].elderlyId,
false);
                    else
                        x = getUserDetails(listOfAppointments2[i].helperId, true);
                    name = x.firstName + " " + x.surname;
                    thingsToShowOnListView[i] = "With " + name + " at " +
listOfAppointments2[i].dateAndTime.ToString("H:mm") + " on " +
listOfAppointments2[i].dateAndTime.ToString("MMMM dd, yyyy");

                }
                RequestWindowFeature(WindowFeatures.NoTitle);
                ListAdapter = new ArrayAdapter<String>(this,
Android.Resource.Layout.SimpleListItem1, thingsToShowOnListView);

            }
            catch
            {
                Toast.MakeText(BaseContext, "Ensure that you are connected to the
internet", ToastLength.Short).Show();

                StartActivity(typeof(MainActivity));
            }
        }

        protected override void OnListItemClick(ListView l, View v, int position,
long id)
        {
            //If clicked, it starts the appointments details activity
            base.OnListItemClick(l, v, position, id);
            appointment appointmentSelected = listOfAppointments2[position];
            var activityToStart = new Intent(this, typeof(AppointmentsActivity));
            activityToStart.PutExtra("appointmentId", appointmentSelected.id);
```

```
            activityToStart.PutExtra("currentUserId", currentUserId);
            activityToStart.PutExtra("currentUserHelper", currentUserHelper);
            StartActivity(activityToStart);

        }


        private appointment[] returnListOfAppointments(int id, bool helper)
        {
            //Make call to the WebAPI, and parse JSON input into an array of
'appointment' object
            string url = "http://178.62.87.28:600/api/app/g";
            if (helper)
                url += "AH" + Convert.ToString(id);
            else
                url += "AE" + Convert.ToString(id);
            var client = new RestClient(url);
            var request = new RestRequest();
            request.Method = Method.GET;
            request.AddHeader("Accept", "application/json");
            request.Parameters.Clear();
            var response = client.Execute(request);
            string result = response.Content;
            appointment[] toReturn =
JsonConvert.DeserializeObject<appointment[]>(result);
            return toReturn;
        }
        private user getUserDetails(int id, bool helper)
        {
            //Make call to the WebAPI, and parse JSON input into a 'user' object
            try
            {
                string url = String.Empty;
                if (helper)
                    url = "http://178.62.87.28:600/api/ach/users/gi" +
Convert.ToString(id);
                else
                    url = "http://178.62.87.28:600/api/ace/users/gi" +
Convert.ToString(id);
                var client = new RestClient(url);
                var request = new RestRequest();
                request.Method = Method.GET;
                request.AddHeader("Accept", "application/json");
                request.Parameters.Clear();
                var response = client.Execute(request);
                string result = response.Content;
                user toReturn = JsonConvert.DeserializeObject<user>(result);
                return toReturn;
            }
            catch
            {
```

```
                Toast.MakeText(BaseContext, "Ensure that you are connected to the
internet", ToastLength.Short).Show();
                StartActivity(typeof(MainActivity));
                return new user();
            }
        }
        private appointment getAppointmentDetails(int id)
        {
            //Make call to the WebAPI, and parse JSON input into an 'appointment'
object
            try
            {
                string url = "http://178.62.87.28:600/api/app/gi" +
Convert.ToString(id);
                var client = new RestClient(url);
                var request = new RestRequest();
                request.Method = Method.GET;
                request.AddHeader("Accept", "application/json");
                request.Parameters.Clear();
                var response = client.Execute(request);
                string result = response.Content;
                appointment toReturn =
JsonConvert.DeserializeObject<appointment>(result);
                return toReturn;
            }
            catch
            {
                Toast.MakeText(BaseContext, "Ensure that you are connected to the
internet", ToastLength.Short).Show();
                StartActivity(typeof(MainActivity));
                return new appointment();
            }
        }
    }
}
```

LoginActivity.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net;
using Android.App;
using Android.Content;
using Android.OS;
using Android.Runtime;
using Android.Views;
using Android.Widget;
using Newtonsoft.Json;
using System.Security.Cryptography;
using System.Net.Http;
```

```
using System.Threading.Tasks;
using RestSharp;
using Android_Application.Types;
using System.IO;
using Android_Application.Backend;
using SQLite;
using SQLitePCL;


namespace Android_Application.Activities
{
    [Activity(Label = "LoginActivity")]
    public class LoginActivity : Activity
    {
        //Database info
        string dbPath =
Path.Combine(System.Environment.GetFolderPath(System.Environment.SpecialFolder.Pers
onal), "dbUser.db3");
        SQLiteConnection db;
        protected override void OnCreate(Bundle savedInstanceState)
        {
            //Setup view
            base.OnCreate(savedInstanceState);
            RequestWindowFeature(WindowFeatures.NoTitle);
            SetContentView(Resource.Layout.login);

            //setup view parts
            Button loginLoginAsHelper =
FindViewById<Button>(Resource.Id.loginLoginAsHelper);
            Button loginLoginAsElderly =
FindViewById<Button>(Resource.Id.loginLoginAsElderly);
            EditText loginEmailAddress =
FindViewById<EditText>(Resource.Id.loginEmailAddress);
            EditText loginPassword =
FindViewById<EditText>(Resource.Id.loginPassword);

            //setup event handlers
            loginLoginAsHelper.Click += LoginLoginAsHelper_Click;
            loginLoginAsElderly.Click += LoginLoginAsElderly_Click;

            //Check if there are any existing logins and passwords.
            //If there are, then try and login with them.
            db = new SQLiteConnection(dbPath);
            try
            {
                var table = db.Table<SQLLiteLogin>();
                foreach(var item in table)
                {
                    user x = login(item.emailAddress, item.password,
item.helper).Result;
```

```
                    //Try and login and go onto correct activities if login is
successful
                    if (x.id != 0)
                    {
                        if (!item.helper)
                        {
                            var newActivity = new Intent(this,
typeof(welcomeElderly));
                            newActivity.PutExtra("currentUserId", x.id);
                            StartActivity(newActivity);
                        }
                        else
                        {
                            var newActivity = new Intent(this,
typeof(welcomeHelper));
                            newActivity.PutExtra("currentUserId", x.id);
                            StartActivity(newActivity);
                        }
                    }
                }
            }
            catch
            {
                //If no table already, create one.
                db.CreateTable<SQLLiteLogin>();
            }
        }

        private void LoginLoginAsElderly_Click(object sender, EventArgs e)
        {
            //If login
            try
            {
                //Access correct views
                EditText eA =
FindViewById<EditText>(Resource.Id.loginEmailAddress);
                EditText p = FindViewById<EditText>(Resource.Id.loginPassword);

                //Checks not empty
                if (checkIfEmpty(eA, p))
                    return;

                //Try and login
                user x = login(eA.Text, p.Text, false).Result;

                bool successfulLogin;
                if (x.id == 0)
                    successfulLogin = false;
                else
                    successfulLogin = true;
                if (successfulLogin)
```

```
                {
                    //Dialog box saying successful login
                    Dialogs newDialog = new Dialogs("Login Success", "You've
successfully logged in", this);
                    SQLLiteLogin c = new SQLLiteLogin();
                    c.emailAddress = eA.Text;
                    c.password = p.Text;
                    c.helper = false;
                    db.Insert(c);
                    //Inserts the login into the database for a successful auto
login next time
                    //Starts the correct activity
                    var newActivity = new Intent(this, typeof(welcomeElderly));
                    newActivity.PutExtra("currentUserId", x.id);
                    StartActivity(newActivity);

                }
                else
                {
                    Dialogs newDialog = new Dialogs("Login Failed", "You've got
your email address or password wrong.", this);
                }
            }
            catch
            {
                Toast.MakeText(BaseContext, "Ensure that you are connected to the
internet", ToastLength.Short).Show();
                StartActivity(typeof(MainActivity));
            }
        }

        private void LoginLoginAsHelper_Click(object sender, EventArgs e)
        {
            // Effectively the same as for helper.
            try
            {
                EditText eA =
FindViewById<EditText>(Resource.Id.loginEmailAddress);
                EditText p = FindViewById<EditText>(Resource.Id.loginPassword);
                if (checkIfEmpty(eA, p))
                    return;
                user x = login(eA.Text, p.Text, true).Result;
                bool successfulLogin;
                if (x.id == 0)
                    successfulLogin = false;
                else
                    successfulLogin = true;
                if (successfulLogin)
                {
                    Dialogs newDialog = new Dialogs("Login Success", "You've
successfully logged in", this);
```

```
                    SQLLiteLogin c = new SQLLiteLogin();
                    c.emailAddress = eA.Text;
                    c.password = p.Text;
                    c.helper = true;
                    db.Insert(c);
                    var newActivity = new Intent(this, typeof(welcomeHelper));
                    newActivity.PutExtra("currentUserId", x.id);
                    StartActivity(newActivity);
                }
                else
                {
                    Dialogs newDialog = new Dialogs("Login Failed", "You've got
your email address or password wrong", this);
                }
            }
            catch
            {
                Toast.MakeText(BaseContext, "Ensure that you are connected to the
internet", ToastLength.Short).Show();
                StartActivity(typeof(MainActivity));
            }
        }

        private bool checkIfEmpty(EditText eA, EditText p)
        {
            //Check if things are valid
            if (String.IsNullOrEmpty(eA.Text) || String.IsNullOrEmpty(p.Text))
            {
                //Dialog created
                AlertDialog.Builder a = new AlertDialog.Builder(this);
                a.SetTitle("Login Failed");
                a.SetMessage("You have not filled in all required information.");
                Dialog dialog = a.Create();
                dialog.Show();
                return true;
            }
            else
                return false;

        }

        private async Task<user> login(string emailAddress, string password, bool
helper)
        {
            //Try and login with provided information
            try
            {
                userToBeVerified user = new userToBeVerified(); // create
usertobeverified object
                user.emailAddress = emailAddress;
```

```
                user.password = MD5Hash.MD5HashReturn(password); // MD5 hash
password
                user.helper = helper;

                //Serialized object, POSTS to WebAPI and deserialises input into a
user object
                String data = JsonConvert.SerializeObject(user);
                var client = new
RestClient("http://178.62.87.28:600/api/ac/users/vf");
                var request = new RestRequest();
                request.Method = Method.POST;
                request.AddHeader("Accept", "application/json");
                request.Parameters.Clear();
                request.AddParameter("application/json", data,
ParameterType.RequestBody);
                var response = client.Execute(request);
                string result = response.Content;
                user userreturned = JsonConvert.DeserializeObject<user>(result);
                return userreturned;
            }
            catch
            {
                Toast.MakeText(BaseContext, "Ensure that you are connected to the
internet", ToastLength.Short).Show();
                StartActivity(typeof(MainActivity));
                return new user();
            }

        }

    }
}
```

RatingsActivity.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Android.App;
using Android.Content;
using Android.OS;
using Android.Runtime;
using Android.Views;
using Android.Widget;

using RestSharp;
using Newtonsoft.Json;
using Android_Application.Types;

namespace Android_Application.Activities
```

```
{
    [Activity(Label = "RatingsActivity")]
    public class RatingsActivity : Activity
    {


        protected override void OnCreate(Bundle savedInstanceState)
        {
            base.OnCreate(savedInstanceState);
            //Gets parameters from the previous activity
            int appointmentId = Intent.GetIntExtra("appointmentId", 0);
            int ratingId = Intent.GetIntExtra("ratingId", 0);
            RequestWindowFeature(WindowFeatures.NoTitle);
            SetContentView(Resource.Layout.ratingsScreen);

            //Gets views of the screen
            TextView nameOfHelper =
FindViewById<TextView>(Resource.Id.ratingNameOfHelper);
            RatingBar ratingBar = FindViewById<RatingBar>(Resource.Id.ratingBar);
            Button submitRating = FindViewById<Button>(Resource.Id.submitRating);

            //Get's existing rating if one exists
            if (ratingId != 0)
            {
                //POPULATE RATING
                ratingBar.Progress = rating(ratingId);
            }
            else
            {
                ratingBar.Progress = 0;
            }

            //POPULATE THE NAME
            try
            {
                user x =
getUserDetails(getAppointmentDetails(appointmentId).helperId, true);
                nameOfHelper.Text = x.firstName + " " + x.surname;
            }
            catch
            {
                Toast.MakeText(BaseContext, "Ensure that you are connected to the
internet", ToastLength.Short).Show();
                StartActivity(typeof(MainActivity));
            }

            //Get event handler
            submitRating.Click += (sender, e) => SubmitRating_Click(sender, e,
appointmentId, ratingId);
        }
```

```
        private void SubmitRating_Click(object sender, EventArgs e, int
appointmentId, int ratingId)
        {
            //Make call to the WebAPI, POSTing the next rating
            try
            {
                RatingBar ratingBar =
FindViewById<RatingBar>(Resource.Id.ratingBar);
                rating newRating = new Types.rating();
                newRating.starRating = ratingBar.Progress;
                newRating.helperId = getAppointmentDetails(appointmentId).helperId;
                //uses the ratingId bit as a place to keep appointmentId if it is a
new rating
                if (ratingId == 0)
                    newRating.id = appointmentId;
                else
                    newRating.id = ratingId;
                string data = JsonConvert.SerializeObject(newRating);
                string url = String.Empty;
                if (ratingId == 0)
                    url = "http://178.62.87.28:600/api/ratings/ar"; // add if it a
new rating
                else
                    url = "http://178.62.87.28:600/api/ratings/ud"; // update if it
is an existing rating
                var client = new RestClient(url);
                var request = new RestRequest();
                if (ratingId == 0)
                    request.Method = Method.POST;
                else
                    request.Method = Method.PATCH;
                request.AddHeader("Accept", "application/json");
                request.Parameters.Clear();
                request.AddParameter("application/json", data,
ParameterType.RequestBody);
                var response = client.Execute(request);
                string result = response.Content;
                Toast.MakeText(BaseContext, "Rating submitted.",
ToastLength.Short).Show();

            }
            catch
            {
                Toast.MakeText(BaseContext, "Ensure that you are connected to the
internet", ToastLength.Short).Show();
                StartActivity(typeof(MainActivity));
            }
        }

        private int rating(int id)
        {
```

```
            //Make call to the WebAPI, and gets the integer value of the rating
from the rating object.
            try
            {
                string url = "http://178.62.87.28:600/api/ratings/go" +
Convert.ToString(id);
                var client = new RestClient(url);
                var request = new RestRequest();
                request.Method = Method.GET;
                request.AddHeader("Accept", "application/json");
                request.Parameters.Clear();
                var response = client.Execute(request);
                string result = response.Content;
                rating toReturn = JsonConvert.DeserializeObject<rating>(result);
                return toReturn.starRating;
            }
            catch
            {
                Toast.MakeText(BaseContext, "Ensure that you are connected to the
internet", ToastLength.Short).Show();
                StartActivity(typeof(MainActivity));
                return 0;
            }
        }
        private user getUserDetails(int id, bool helper)
        {
            //Make call to the WebAPI, and parse JSON input into a 'user' object
            try
            {
                string url = String.Empty;
                if (helper)
                    url = "http://178.62.87.28:600/api/ach/users/gi" +
Convert.ToString(id);
                else
                    url = "http://178.62.87.28:600/api/ace/users/gi" +
Convert.ToString(id);
                var client = new RestClient(url);
                var request = new RestRequest();
                request.Method = Method.GET;
                request.AddHeader("Accept", "application/json");
                request.Parameters.Clear();
                var response = client.Execute(request);
                string result = response.Content;
                user toReturn = JsonConvert.DeserializeObject<user>(result);
                return toReturn;
            }
            catch
            {
                Toast.MakeText(BaseContext, "Ensure that you are connected to the
internet", ToastLength.Short).Show();
                StartActivity(typeof(MainActivity));
```

```
                return new user();
            }
        }
        private appointment getAppointmentDetails(int id)
        {
            //Make call to the WebAPI, and parse JSON input into a 'appointment'
object
            try
            {
                string url = "http://178.62.87.28:600/api/app/gi" +
Convert.ToString(id);
                var client = new RestClient(url);
                var request = new RestRequest();
                request.Method = Method.GET;
                request.AddHeader("Accept", "application/json");
                request.Parameters.Clear();
                var response = client.Execute(request);
                string result = response.Content;
                appointment toReturn =
JsonConvert.DeserializeObject<appointment>(result);
                return toReturn;
            }
            catch
            {
                Toast.MakeText(BaseContext, "Ensure that you are connected to the
internet", ToastLength.Short).Show();
                StartActivity(typeof(MainActivity));
                return new appointment();
            }
        }
    }
}
```

RegisterActivity.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Android_Application.Backend;
using Android.App;
using Android.Content;
using Android.OS;
using Android.Runtime;
using Android.Views;
using Android.Widget;
using RestSharp;
using Newtonsoft.Json;
using Android_Application.Types;
using System.Threading.Tasks;

namespace Android_Application.Activities
```

```
{
    [Activity(Label = "RegisterActivity")]
    public class RegisterActivity : Activity
    {
        protected override void OnCreate(Bundle savedInstanceState)
        {
            try
            {
                //Sets up view
                base.OnCreate(savedInstanceState);
                RequestWindowFeature(WindowFeatures.NoTitle);
                SetContentView(Resource.Layout.register);

                //Defines items in view
                EditText registerFirstName =
FindViewById<EditText>(Resource.Id.registerFirstName);
                EditText registerSurname =
FindViewById<EditText>(Resource.Id.registerSurname);
                EditText registerUsername =
FindViewById<EditText>(Resource.Id.registerUsername);
                EditText registerEmailAddress =
FindViewById<EditText>(Resource.Id.registerEmailAddress);
                EditText registerPassword =
FindViewById<EditText>(Resource.Id.registerPassword);
                EditText registerFirstLine =
FindViewById<EditText>(Resource.Id.registerFirstLine);
                EditText registerSecondLine =
FindViewById<EditText>(Resource.Id.registerSecondLine);
                EditText registerCity =
FindViewById<EditText>(Resource.Id.registerCity);
                EditText registerCountry =
FindViewById<EditText>(Resource.Id.registerCountry);
                EditText registerPostalCode =
FindViewById<EditText>(Resource.Id.registerPostalCode);
                EditText registerTelephoneNumber =
FindViewById<EditText>(Resource.Id.registerTelephoneNumber);
                Button registerRegisterAsHelper =
FindViewById<Button>(Resource.Id.registerRegisterAsHelper);
                Button registerRegisterAsElderly =
FindViewById<Button>(Resource.Id.registerRegisterAsElderly);
                // Create your application here

                //Sets up event handlers
                registerRegisterAsElderly.Click += RegisterRegisterAsElderly_Click;
                registerRegisterAsHelper.Click += RegisterRegisterAsHelper_Click;
            }
            catch
            {
                Toast.MakeText(BaseContext, "Ensure that you are connected to the
internet", ToastLength.Short).Show();
                StartActivity(typeof(MainActivity));
```

```
            }

        }

        private void RegisterRegisterAsHelper_Click(object sender, EventArgs e)
        {
            if (!checkValid()) // Check that the contents of the screen are valid
(things are filled in)
            {
                Dialogs newDialog = new Dialogs("Registration Unsuccessful", "Not
all required fields have been filled or your passwords do not match.", this);
                return;
            }
            EditText registerFirstName =
FindViewById<EditText>(Resource.Id.registerFirstName);
            EditText registerSurname =
FindViewById<EditText>(Resource.Id.registerSurname);
            EditText registerUsername =
FindViewById<EditText>(Resource.Id.registerUsername);
            EditText registerEmailAddress =
FindViewById<EditText>(Resource.Id.registerEmailAddress);
            EditText registerPassword =
FindViewById<EditText>(Resource.Id.registerPassword);
            EditText registerFirstLine =
FindViewById<EditText>(Resource.Id.registerFirstLine);
            EditText registerSecondLine =
FindViewById<EditText>(Resource.Id.registerSecondLine);
            EditText registerCity =
FindViewById<EditText>(Resource.Id.registerCity);
            EditText registerCountry =
FindViewById<EditText>(Resource.Id.registerCountry);
            EditText registerPostalCode =
FindViewById<EditText>(Resource.Id.registerPostalCode);
            EditText registerTelephoneNumber =
FindViewById<EditText>(Resource.Id.registerTelephoneNumber);
            Button registerRegisterAsHelper =
FindViewById<Button>(Resource.Id.registerRegisterAsHelper);
            Button registerRegisterAsElderly =
FindViewById<Button>(Resource.Id.registerRegisterAsElderly);

            //Creates a user object and populates it
            user user = new user();
            user.city = registerCity.Text;
            user.country = registerCountry.Text;
            user.emailAddress = registerEmailAddress.Text;
            user.firstLineOfAddress = registerFirstLine.Text;
            user.secondLineOfAddress = registerSecondLine.Text;
            user.surname = registerSurname.Text;
            user.password = MD5Hash.MD5HashReturn(registerPassword.Text);
//Password is MD5 hashed
            string telephoneNumber = registerTelephoneNumber.Text;
```

```
            if (telephoneNumber[0] == '0')
            {
                telephoneNumber = "44" + telephoneNumber.Substring(1); // phone
numbers are stored as 44...
            }

            user.telephoneNumber = telephoneNumber;
            user.username = registerUsername.Text;
            user.postalCode = registerPostalCode.Text;
            user.firstName = registerFirstName.Text;
            user.helper = true;
            if (register(user)) // try and register user
            {
                //SUCCESS
                string returnFromVerificationProcess = string.Empty;
                TwilioRegistration newUser = new TwilioRegistration();
                //try to get a phone verification
                newUser.Number = telephoneNumber;
                newUser.friendlyName = registerFirstName.Text + " " +
registerSurname.Text;
                returnFromVerificationProcess = verifyNumber(newUser); // verify
number
                if (returnFromVerificationProcess.Length > 10) // indicative of
failure
                {
                    Dialogs newDialog2 = new Dialogs("Registration Unsuccessful",
"Phone number was invalid or has already been used for another account", this);
                    user addedUser = testVerify(user.emailAddress,
registerPassword.Text, false).Result; // effectively used to get the id of the user
which has been added
                    deleteUser(addedUser.id, false); // then delete it
                }
                else
                {
                    Dialogs newDialog = new Dialogs("Registration Successful", "You
should now respond to the incoming call with the following code: " +
returnFromVerificationProcess, this);
                }
            }
            else
            {
                Dialogs newDialog = new Dialogs("Registration Unsuccessful", "There
is already an account with that email address or phone number", this);
            }
            return;
            //
            //

        }
        private bool checkValid()
```

```
        {
            //Check if any important fields are empty
            //AND if the passwords do not match
            EditText registerFirstName =
FindViewById<EditText>(Resource.Id.registerFirstName);
            EditText registerSurname =
FindViewById<EditText>(Resource.Id.registerSurname);
            EditText registerUsername =
FindViewById<EditText>(Resource.Id.registerUsername);
            EditText registerEmailAddress =
FindViewById<EditText>(Resource.Id.registerEmailAddress);
            EditText registerPassword =
FindViewById<EditText>(Resource.Id.registerPassword);
            EditText registerConfirmPassword =
FindViewById<EditText>(Resource.Id.registerConfirmPassword);
            EditText registerFirstLine =
FindViewById<EditText>(Resource.Id.registerFirstLine);
            EditText registerSecondLine =
FindViewById<EditText>(Resource.Id.registerSecondLine);
            EditText registerCity =
FindViewById<EditText>(Resource.Id.registerCity);
            EditText registerCountry =
FindViewById<EditText>(Resource.Id.registerCountry);
            EditText registerPostalCode =
FindViewById<EditText>(Resource.Id.registerPostalCode);

            EditText registerTelephoneNumber =
FindViewById<EditText>(Resource.Id.registerTelephoneNumber);
            Button registerRegisterAsHelper =
FindViewById<Button>(Resource.Id.registerRegisterAsHelper);
            Button registerRegisterAsElderly =
FindViewById<Button>(Resource.Id.registerRegisterAsElderly);
            if (String.IsNullOrEmpty(registerFirstName.Text))
                return false;
            if (String.IsNullOrEmpty(registerSurname.Text))
                return false;
            if (String.IsNullOrEmpty(registerUsername.Text))
                return false;
            if (String.IsNullOrEmpty(registerEmailAddress.Text))
                return false;
            if (String.IsNullOrEmpty(registerPassword.Text))
                return false;
            if (String.IsNullOrEmpty(registerConfirmPassword.Text))
                return false;
            if (String.IsNullOrEmpty(registerFirstLine.Text))
                return false;
            if (String.IsNullOrEmpty(registerPostalCode.Text))
                return false;
            if (String.IsNullOrEmpty(registerCity.Text))
                return false;
            if (String.IsNullOrEmpty(registerTelephoneNumber.Text))
```

```
                return false;
            if (registerPassword.Text != registerConfirmPassword.Text)
                return false;
            return true;
        }


        private void RegisterRegisterAsElderly_Click(object sender, EventArgs e)
        {
            //Effectively the same as the register as helper click
            if (!checkValid())
            {
                Dialogs newDialog = new Dialogs("Registration Unsuccessful", "Not
all required fields have been filled.", this);
                return;
            }
            EditText registerFirstName =
FindViewById<EditText>(Resource.Id.registerFirstName);
            EditText registerSurname =
FindViewById<EditText>(Resource.Id.registerSurname);
            EditText registerUsername =
FindViewById<EditText>(Resource.Id.registerUsername);
            EditText registerEmailAddress =
FindViewById<EditText>(Resource.Id.registerEmailAddress);
            EditText registerPassword =
FindViewById<EditText>(Resource.Id.registerPassword);
            EditText registerFirstLine =
FindViewById<EditText>(Resource.Id.registerFirstLine);
            EditText registerSecondLine =
FindViewById<EditText>(Resource.Id.registerSecondLine);
            EditText registerCity =
FindViewById<EditText>(Resource.Id.registerCity);
            EditText registerCountry =
FindViewById<EditText>(Resource.Id.registerCountry);
            EditText registerPostalCode =
FindViewById<EditText>(Resource.Id.registerPostalCode);
            EditText registerTelephoneNumber =
FindViewById<EditText>(Resource.Id.registerTelephoneNumber);
            Button registerRegisterAsHelper =
FindViewById<Button>(Resource.Id.registerRegisterAsHelper);
            Button registerRegisterAsElderly =
FindViewById<Button>(Resource.Id.registerRegisterAsElderly);

            user user = new user();
            user.city = registerCity.Text;
            user.country = registerCountry.Text;
            user.emailAddress = registerEmailAddress.Text;
            user.firstLineOfAddress = registerFirstLine.Text;
            user.secondLineOfAddress = registerSecondLine.Text;
            user.surname = registerSurname.Text;
            string telephoneNumber = registerTelephoneNumber.Text;
```

```
            if (telephoneNumber[0] == '0')
            {
                telephoneNumber = "44" + telephoneNumber.Substring(1);
            }
            user.telephoneNumber = telephoneNumber;
            user.username = registerUsername.Text;
            user.postalCode = registerPostalCode.Text;
            user.firstName = registerFirstName.Text;
            user.password = MD5Hash.MD5HashReturn(registerPassword.Text);
            user.helper = false;
            if (register(user))
            {
                string returnFromVerificationProcess = string.Empty;
                TwilioRegistration newUser = new TwilioRegistration();
                newUser.Number = telephoneNumber;
                newUser.friendlyName = registerFirstName.Text + " " +
registerSurname.Text;
                returnFromVerificationProcess = verifyNumber(newUser);
                if (returnFromVerificationProcess.Length > 10)
                {
                    Dialogs newDialog2 = new Dialogs("Registration Unsuccessful",
"There is already an account with that phone number", this);
                    user addedUser = testVerify(user.emailAddress,
registerPassword.Text, false).Result;
                    deleteUser(addedUser.id, false);
                }
                else
                {
                    Dialogs newDialog = new Dialogs("Registration Successful", "You
should now respond to the incoming call with the following code: " +
returnFromVerificationProcess, this);
                }
            }
            else
            {
                Dialogs newDialog = new Dialogs("Registration Unsuccessful", "There
is already an account with that email address or phone number", this);
            }
            return;
        }

        private void deleteUser(int id, bool helper)
        {
            //Make call to the WebAPI to delete the user based on it's user id
            //URL differs based on whether trying to delete a helper or an elderly
person
            try
            {
                string url = String.Empty;
                if (helper)
```

```
                    url = "http://178.62.87.28:600/api/ach/users/rm" +
Convert.ToString(id);
                else
                    url = "http://178.62.87.28:600/api/ace/users/rm" +
Convert.ToString(id);
                var client = new RestClient(url);
                var request = new RestRequest();
                request.Method = Method.DELETE;
                request.AddHeader("Accept", "application/json");
                request.Parameters.Clear();
                var response = client.Execute(request);
            }
            catch
            {
                Toast.MakeText(BaseContext, "Ensure that you are connected to the
internet", ToastLength.Short).Show();
                StartActivity(typeof(MainActivity));
            }
        }
        private async Task<user> testVerify(string emailAddress, string password,
bool helper)
        {
            //Effectively tries to login and returns the user...
            try
            {
                userToBeVerified user = new userToBeVerified();
                user.emailAddress = emailAddress;
                user.password = MD5Hash.MD5HashReturn(password);
                //user.password = password;
                user.helper = helper;
                String data = JsonConvert.SerializeObject(user);
                var client = new
RestClient("http://178.62.87.28:600/api/ac/users/vf");
                var request = new RestRequest();
                request.Method = Method.POST;
                request.AddHeader("Accept", "application/json");
                request.Parameters.Clear();
                request.AddParameter("application/json", data,
ParameterType.RequestBody);
                var response = client.Execute(request);
                string result = response.Content;
                user userreturned = JsonConvert.DeserializeObject<user>(result);
                return userreturned;
            }
            catch
            {
                Toast.MakeText(BaseContext, "Ensure that you are connected to the
internet", ToastLength.Short).Show();
                StartActivity(typeof(MainActivity));
                return new user();
            }
```

```
        }
        private bool register(user user)
        {
            //Makes a POST call to the WebAPI and therefore attempts to register
the user.
            try
            {
                String data = JsonConvert.SerializeObject(user);
                var client = new
RestClient("http://178.62.87.28:600/api/ac/users/ad");
                var request = new RestRequest();
                request.Method = Method.POST;
                request.AddHeader("Accept", "application/json");
                request.Parameters.Clear();
                request.AddParameter("application/json", data,
ParameterType.RequestBody);
                var response = client.Execute(request);
                string result = response.Content;
                return bool.Parse(result);
            }
            catch
            {
                Toast.MakeText(BaseContext, "Ensure that you are connected to the
internet", ToastLength.Short).Show();
                StartActivity(typeof(MainActivity));
                return false;
            }
        }

        private string verifyNumber(TwilioRegistration user)
        {
            //Verifies the number by making a POST call to the SMS server
            try
            {
                String data = JsonConvert.SerializeObject(user);
                var client = new
RestClient("http://178.62.87.28:700/verifyNumber");
                var request = new RestRequest();
                request.Method = Method.POST;
                request.AddHeader("Accept", "application/json");
                request.Parameters.Clear();
                request.AddParameter("application/json", data,
ParameterType.RequestBody);
                var response = client.Execute(request);
                string result = response.Content;
                return result;
            }
            catch
            {
```

```
                Toast.MakeText(BaseContext, "Ensure that you are connected to the
internet", ToastLength.Short).Show();
                StartActivity(typeof(MainActivity));
                return String.Empty;
            }
        }
    }
}
```

ReviewActivity.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Android.App;
using Android.Content;
using Android.OS;
using Android.Runtime;
using Android.Views;
using Android.Widget;

using RestSharp;
using Newtonsoft.Json;
using Android_Application.Types;

namespace Android_Application.Activities
{
    [Activity(Label = "reviewActivity")]
    public class reviewActivity : Activity
    {
        protected override void OnCreate(Bundle savedInstanceState)
        {
            base.OnCreate(savedInstanceState);
            //Get parameters from previous activity
            int appointmentId = Intent.GetIntExtra("appointmentId", 0);
            int reviewId = Intent.GetIntExtra("reviewId", 0);

            //Sets up view
            RequestWindowFeature(WindowFeatures.NoTitle);
            SetContentView(Resource.Layout.reviewScreen);

            //Sets up items in view
            TextView nameOfHelper =
FindViewById<TextView>(Resource.Id.reviewNameOfHelper);
            EditText reviewComment =
FindViewById<EditText>(Resource.Id.reviewComment);
            Button submitReview = FindViewById<Button>(Resource.Id.submitReview);

            if (reviewId != 0)
```

```
            {
                //POPULATE RATING
                reviewComment.Text = review(reviewId);
            }
            else
            {
                reviewComment.Text = String.Empty;
            }

            //POPULATE THE NAME
            try
            {
                user x =
getUserDetails(getAppointmentDetails(appointmentId).helperId, true);
                nameOfHelper.Text = x.firstName + " " + x.surname;
            }
            catch
            {
                Toast.MakeText(BaseContext, "Ensure that you are connected to the
internet", ToastLength.Short).Show();
                StartActivity(typeof(MainActivity));
            }

            submitReview.Click += (sender, e) => SubmitReview_Click(sender, e,
appointmentId, reviewId);
        }

        private void SubmitReview_Click(object sender, EventArgs e, int
appointmentId, int ratingId)
        {
            //Submits the review
            /*
             * If existing review, it is updated - otherwise it is added with the
appointment id also passed to the WebAPI to be placed in the database.
             *
             */
            try
            {
                EditText reviewComment =
FindViewById<EditText>(Resource.Id.reviewComment);
                review newReview = new Types.review();
                newReview.comment = reviewComment.Text;
                newReview.helperId = getAppointmentDetails(appointmentId).helperId;
                if (ratingId == 0)
                    newReview.id = appointmentId;
                else
                    newReview.id = ratingId;
                string data = JsonConvert.SerializeObject(newReview);
                string url = String.Empty;
                if (ratingId == 0)
                    url = "http://178.62.87.28:600/api/reviews/ar";
```

```
                else
                    url = "http://178.62.87.28:600/api/reviews/ud";
                var client = new RestClient(url);
                var request = new RestRequest();
                if (ratingId == 0)
                    request.Method = Method.POST;
                else
                    request.Method = Method.PATCH;
                request.AddHeader("Accept", "application/json");
                request.Parameters.Clear();
                request.AddParameter("application/json", data,
ParameterType.RequestBody);
                var response = client.Execute(request);
                string result = response.Content;
                Toast.MakeText(BaseContext, "Review submitted.",
ToastLength.Short).Show();

            }
            catch
            {
                Toast.MakeText(BaseContext, "Ensure that you are connected to the
internet", ToastLength.Short).Show();
                StartActivity(typeof(MainActivity));
            }
        }

        private string review(int id)
        {
            //Make call to the WebAPI, and returns the review (as a string) after
it being deserialised
            try
            {
                string url = "http://178.62.87.28:600/api/reviews/go" +
Convert.ToString(id);
                var client = new RestClient(url);
                var request = new RestRequest();
                request.Method = Method.GET;
                request.AddHeader("Accept", "application/json");
                request.Parameters.Clear();
                var response = client.Execute(request);
                string result = response.Content;
                review toReturn = JsonConvert.DeserializeObject<review>(result);
                return toReturn.comment;
            }
            catch
            {
                Toast.MakeText(BaseContext, "Ensure that you are connected to the
internet", ToastLength.Short).Show();
                StartActivity(typeof(MainActivity));
                return String.Empty;
            }
```

```
        }
        private user getUserDetails(int id, bool helper)
        {
            try
            {
                string url = String.Empty;
                if (helper)
                    url = "http://178.62.87.28:600/api/ach/users/gi" +
Convert.ToString(id);
                else
                    url = "http://178.62.87.28:600/api/ace/users/gi" +
Convert.ToString(id);
                var client = new RestClient(url);
                var request = new RestRequest();
                request.Method = Method.GET;
                request.AddHeader("Accept", "application/json");
                request.Parameters.Clear();
                var response = client.Execute(request);
                string result = response.Content;
                user toReturn = JsonConvert.DeserializeObject<user>(result);
                return toReturn;
            }
            catch
            {
                Toast.MakeText(BaseContext, "Ensure that you are connected to the
internet", ToastLength.Short).Show();
                StartActivity(typeof(MainActivity));
                return new user();
            }
        }
        private appointment getAppointmentDetails(int id)
        {
            //Make call to the WebAPI, and parse JSON input into an 'appointment'
object
            try
            {
                string url = "http://178.62.87.28:600/api/app/gi" +
Convert.ToString(id);
                var client = new RestClient(url);
                var request = new RestRequest();
                request.Method = Method.GET;
                request.AddHeader("Accept", "application/json");
                request.Parameters.Clear();
                var response = client.Execute(request);
                string result = response.Content;
                appointment toReturn =
JsonConvert.DeserializeObject<appointment>(result);
                return toReturn;
            }
            catch
            {
```

```
                Toast.MakeText(BaseContext, "Ensure that you are connected to the
internet", ToastLength.Short).Show();
                StartActivity(typeof(MainActivity));
                return new appointment();
            }
        }
    }
}
```

SeeAllReviewsOfHelper.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Android.App;
using Android.Content;
using Android.OS;
using Android.Runtime;
using Android.Views;
using Android.Widget;

using Newtonsoft.Json;
using RestSharp;
using Android_Application.Types;

namespace Android_Application.Activities
{
    [Activity(Label = "SeeAllReviewOfHelper")]
    public class SeeAllReviewOfHelper : ListActivity
    {
        //Global variables
        int helperId;
        review[] listOfReviews;
        protected override void OnCreate(Bundle savedInstanceState)
        {
            //get parameters from previous activity
            base.OnCreate(savedInstanceState);
            helperId = Intent.GetIntExtra("helperId", 8);


            RequestWindowFeature(WindowFeatures.NoTitle);

            //populate screen
            listOfReviews = getListOfReviews(helperId);
            try
            {
                string[] toDisplay = new string[listOfReviews.Length];
                for (int i = 0; i < listOfReviews.Length; i++)
                {
                    toDisplay[i] = listOfReviews[i].comment;
```

```
                }
                ListAdapter = new ArrayAdapter<String>(this,
Android.Resource.Layout.SimpleListItem1, toDisplay);
            }
            catch
            {
                Toast.MakeText(BaseContext, "Ensure that you are connected to the
internet", ToastLength.Short).Show();
                StartActivity(typeof(MainActivity));
            }
            // Create your application here

        }
        private review[] getListOfReviews(int id)
        {
            //Make call to the WebAPI, and parse JSON input into an array of
'review' object
            try
            {
                string url = "http://178.62.87.28:600/api/reviews/ga" +
Convert.ToString(id);
                var client = new RestClient(url);
                var request = new RestRequest();
                request.Method = Method.GET;
                request.AddHeader("Accept", "application/json");
                request.Parameters.Clear();
                var response = client.Execute(request);
                string result = response.Content;
                review[] toReturn =
JsonConvert.DeserializeObject<review[]>(result);
                return toReturn;
            }
            catch
            {
                Toast.MakeText(BaseContext, "Ensure that you are connected to the
internet", ToastLength.Short).Show();
                StartActivity(typeof(MainActivity));
                review[] a = new review[1];
                return a;
            }
        }
    }
}
```

SetTimetableDays.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Android.App;
```

```
using Android.Content;
using Android.OS;
using Android.Runtime;
using Android.Views;
using Android.Widget;
using System.Globalization;

namespace Android_Application.Activities
{
    [Activity(Label = "setTimetableDays")]
    public class setTimetableDays : ListActivity
    {
        //Global variables
        int helperId;
        string[] listOfThingsToDisplay;

        protected override void OnCreate(Bundle savedInstanceState)
        {
            try
            {
                //Get parameters from previous activity
                base.OnCreate(savedInstanceState);
                helperId = Intent.GetIntExtra("helperId", 8);
                DateTime d = DateTime.ParseExact(Intent.GetStringExtra("dateTime"),
"dd/MM/yyyy", CultureInfo.InvariantCulture);

                //gets a list of the days of the week to display on the screen
                listOfThingsToDisplay = new string[7];
                for (int i = 0; i < listOfThingsToDisplay.Length; i++)
                {
                    DateTime dt = d.AddDays(i);
                    listOfThingsToDisplay[i] = dt.ToString("MMMM dd, yyyy");
                }
                RequestWindowFeature(WindowFeatures.NoTitle);
                ListAdapter = new ArrayAdapter<String>(this,
Android.Resource.Layout.SimpleListItem1, listOfThingsToDisplay);
            }
            catch
            {
                Toast.MakeText(BaseContext, "Ensure that you are connected to the
internet", ToastLength.Short).Show();
                StartActivity(typeof(MainActivity));
            }
        }
        protected override void OnListItemClick(ListView l, View v, int position,
long id)
        {
            //Starts the activity of setting the free time of the clicked day
            base.OnListItemClick(l, v, position, id);
            DateTime toPass = DateTime.ParseExact(listOfThingsToDisplay[position],
"MMMM dd, yyyy", CultureInfo.InvariantCulture);
```

```
            string toPassString = toPass.ToString("dd/MM/yyyy");
            var newActivity = new Intent(this, typeof(setTimetableTimes));
            newActivity.PutExtra("helperId", helperId);
            newActivity.PutExtra("dateTime", toPassString);
            StartActivity(newActivity);
        }
    }
}
```

SetTimetableTimes.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Android.App;
using Android.Content;
using Android.OS;
using Android.Runtime;
using Android.Views;
using Android.Widget;

using Android_Application.Types;
using RestSharp;
using Newtonsoft.Json;
using System.Globalization;
using Android_Application.Backend;

namespace Android_Application.Activities
{
    [Activity(Label = "setTimetableTimes")]
    public class setTimetableTimes : Activity
    {
        timetable t;
        DateTime dt;
        protected override void OnCreate(Bundle savedInstanceState)
        {
            try
            {
                //Get things from previous activity
                base.OnCreate(savedInstanceState);
                t = getTimetableOfHelper(Intent.GetIntExtra("helperId", 8));
                dt = DateTime.ParseExact(Intent.GetStringExtra("dateTime"),
"dd/MM/yyyy", CultureInfo.InvariantCulture);

                //Setup view
                RequestWindowFeature(WindowFeatures.NoTitle);
                SetContentView(Resource.Layout.setTimetableTimes);

                //Setup items in the views
                Switch a02 = FindViewById<Switch>(Resource.Id.a02);
```

```
                Switch a24 = FindViewById<Switch>(Resource.Id.a24);
                Switch a46 = FindViewById<Switch>(Resource.Id.a46);
                Switch a68 = FindViewById<Switch>(Resource.Id.a68);
                Switch a810 = FindViewById<Switch>(Resource.Id.a810);
                Switch a1012 = FindViewById<Switch>(Resource.Id.a1012);
                Switch a1214 = FindViewById<Switch>(Resource.Id.a1214);
                Switch a1416 = FindViewById<Switch>(Resource.Id.a1416);
                Switch a1618 = FindViewById<Switch>(Resource.Id.a1618);
                Switch a1820 = FindViewById<Switch>(Resource.Id.a1820);
                Switch a2022 = FindViewById<Switch>(Resource.Id.a2022);
                Switch a2224 = FindViewById<Switch>(Resource.Id.a2224);


                getCorrectStatusOfButtons(); // Literally, get correct status of
buttons
                Button submitButton =
FindViewById<Button>(Resource.Id.setTimetableSubmit);

                //Event handlers
                submitButton.Click += SubmitButton_Click;
            }
            catch
            {
                Toast.MakeText(BaseContext, "Ensure that you are connected to the
internet", ToastLength.Short).Show();
                StartActivity(typeof(MainActivity));
            }
        }
        private void getCorrectStatusOfButtons()
        {
            //Setup items
            Switch a02 = FindViewById<Switch>(Resource.Id.a02);
            Switch a24 = FindViewById<Switch>(Resource.Id.a24);
            Switch a46 = FindViewById<Switch>(Resource.Id.a46);
            Switch a68 = FindViewById<Switch>(Resource.Id.a68);
            Switch a810 = FindViewById<Switch>(Resource.Id.a810);
            Switch a1012 = FindViewById<Switch>(Resource.Id.a1012);
            Switch a1214 = FindViewById<Switch>(Resource.Id.a1214);
            Switch a1416 = FindViewById<Switch>(Resource.Id.a1416);
            Switch a1618 = FindViewById<Switch>(Resource.Id.a1618);
            Switch a1820 = FindViewById<Switch>(Resource.Id.a1820);
            Switch a2022 = FindViewById<Switch>(Resource.Id.a2022);
            Switch a2224 = FindViewById<Switch>(Resource.Id.a2224);
            day day = new day();
            bool found = false;

            //Try to find which day of the timetable is relevant
            for (int i = 0; i < t.weeks.Count(); i++)
            {
```

```
                if (t.weeks[i].weekBeginning == dt.StartOfWeek(DayOfWeek.Monday))
// For each week in the timetable (custom weeks) check if it the one we are looking
for
                {
                    found = true;
                    switch (dt.DayOfWeek) // find exactly which day it is
                    {
                        case DayOfWeek.Monday:
                            day = t.weeks[i].week.monday;
                            break;
                        case DayOfWeek.Tuesday:
                            day = t.weeks[i].week.tuesday;
                            break;
                        case DayOfWeek.Wednesday:
                            day = t.weeks[i].week.wednesday;
                            break;
                        case DayOfWeek.Thursday:
                            day = t.weeks[i].week.thursday;
                            break;
                        case DayOfWeek.Friday:
                            day = t.weeks[i].week.friday;
                            break;
                        case DayOfWeek.Saturday:
                            day = t.weeks[i].week.saturday;
                            break;
                        case DayOfWeek.Sunday:
                            day = t.weeks[i].week.sunday;
                            break;
                        default:
                            day = t.weeks[i].week.monday;
                            break;
                    }
                }
            }
            if(!found) // instead need to use the defaultweek as the correct week
            {
                switch (dt.DayOfWeek)
                {
                    case DayOfWeek.Monday:
                        day = t.defaultWeek.monday;
                        break;
                    case DayOfWeek.Tuesday:
                        day = t.defaultWeek.tuesday;
                        break;
                    case DayOfWeek.Wednesday:
                        day = t.defaultWeek.wednesday;
                        break;
                    case DayOfWeek.Thursday:
                        day = t.defaultWeek.thursday;
                        break;
                    case DayOfWeek.Friday:
```

```
                    day = t.defaultWeek.friday;
                    break;
                case DayOfWeek.Saturday:
                    day = t.defaultWeek.saturday;
                    break;
                case DayOfWeek.Sunday:
                    day = t.defaultWeek.sunday;
                    break;
                default:
                    day = t.defaultWeek.monday;
                    break;
            }
        }

        //populate the switches based on the day we have found.

        a02.Checked = day.timesFree[0];
        a24.Checked = day.timesFree[1];
        a46.Checked = day.timesFree[2];
        a68.Checked = day.timesFree[3];
        a810.Checked = day.timesFree[4];
        a1012.Checked = day.timesFree[5];
        a1214.Checked = day.timesFree[6];
        a1416.Checked = day.timesFree[7];
        a1618.Checked = day.timesFree[8];
        a1820.Checked = day.timesFree[9];
        a2022.Checked = day.timesFree[10];
        a2224.Checked = day.timesFree[11];
        return;

    }
    private void SubmitButton_Click(object sender, EventArgs e)
    {
        //if submit button is pressed
        bool found = false;
        for (int i = 0; i < t.weeks.Count(); i++) // find which week we need to
update or if it is the default week
        {
            if(t.weeks[i].weekBeginning == dt.StartOfWeek(DayOfWeek.Monday)) //
if it this week
            {
                //If yes
                found = true;
                switch (dt.DayOfWeek) // Find which day to update
                {
                    case DayOfWeek.Monday:
                        t.weeks[i].week.monday = assembleDay(); // Change the
day as required
                        break;
                    case DayOfWeek.Tuesday:
                        t.weeks[i].week.tuesday = assembleDay();
```

```
                        break;
                    case DayOfWeek.Wednesday:
                        t.weeks[i].week.wednesday = assembleDay();
                        break;
                    case DayOfWeek.Thursday:
                        t.weeks[i].week.thursday = assembleDay();
                        break;
                    case DayOfWeek.Friday:
                        t.weeks[i].week.friday = assembleDay();
                        break;
                    case DayOfWeek.Saturday:
                        t.weeks[i].week.saturday = assembleDay();
                        break;
                    case DayOfWeek.Sunday:
                        t.weeks[i].week.sunday = assembleDay();
                        break;
                    default:
                        t.weeks[i].week.monday = assembleDay();
                        break;
                }
            }
        }
        if(!found) // nope, not found
        {
            names x = new names(); // create a new week in the views
            x.weekBeginning = dt.StartOfWeek(DayOfWeek.Monday); // populate the
week beginning
            week week = t.defaultWeek; // now make a clone of the default week
            switch (dt.DayOfWeek) // then change one of the days as required.
            {
                case DayOfWeek.Monday:
                    week.monday = assembleDay();
                    break;
                case DayOfWeek.Tuesday:
                    week.tuesday = assembleDay();
                    break;
                case DayOfWeek.Wednesday:
                    week.wednesday = assembleDay();
                    break;
                case DayOfWeek.Thursday:
                    week.thursday = assembleDay();
                    break;
                case DayOfWeek.Friday:
                    week.friday = assembleDay();
                    break;
                case DayOfWeek.Saturday:
                    week.saturday = assembleDay();
                    break;
                case DayOfWeek.Sunday:
                    week.sunday = assembleDay();
                    break;
```

```
                    default:
                        week.monday = assembleDay();
                        break;
                }
                x.week = week;
                t.weeks.Add(x); // add it to the existing timetable
            }
            try
            {
                //Now add the timetable and therefore it is updated
                String data = JsonConvert.SerializeObject(t);
                var client = new RestClient("http://178.62.87.28:600/api/tt/adt");
                var request = new RestRequest();
                request.Method = Method.POST;
                request.AddHeader("Accept", "application/json");
                request.Parameters.Clear();
                request.AddParameter("application/json", data,
ParameterType.RequestBody);
                client.Execute(request);
                Toast.MakeText(BaseContext, "The day has been successfully
updated.", ToastLength.Short).Show(); // Notify the user that the day has been
updated


            }
            catch
            {
                Toast.MakeText(BaseContext, "Ensure that you are connected to the
internet", ToastLength.Short).Show();
                StartActivity(typeof(MainActivity));
            }
        }
        private day assembleDay()
        {
            //Convert the position of the switches into a 'day' object based on
their position
            Switch a02 = FindViewById<Switch>(Resource.Id.a02);
            Switch a24 = FindViewById<Switch>(Resource.Id.a24);
            Switch a46 = FindViewById<Switch>(Resource.Id.a46);
            Switch a68 = FindViewById<Switch>(Resource.Id.a68);
            Switch a810 = FindViewById<Switch>(Resource.Id.a810);
            Switch a1012 = FindViewById<Switch>(Resource.Id.a1012);
            Switch a1214 = FindViewById<Switch>(Resource.Id.a1214);
            Switch a1416 = FindViewById<Switch>(Resource.Id.a1416);
            Switch a1618 = FindViewById<Switch>(Resource.Id.a1618);
            Switch a1820 = FindViewById<Switch>(Resource.Id.a1820);
            Switch a2022 = FindViewById<Switch>(Resource.Id.a2022);
            Switch a2224 = FindViewById<Switch>(Resource.Id.a2224);
            day day = new day();
            day.timesFree[0] = a02.Checked;
            day.timesFree[1] = a24.Checked;
```

```
            day.timesFree[2] = a46.Checked;
            day.timesFree[3] = a68.Checked;
            day.timesFree[4] = a810.Checked;
            day.timesFree[5] = a1012.Checked;
            day.timesFree[6] = a1214.Checked;
            day.timesFree[7] = a1416.Checked;
            day.timesFree[8] = a1618.Checked;
            day.timesFree[9] = a1820.Checked;
            day.timesFree[10] = a2022.Checked;
            day.timesFree[11] = a2224.Checked;
            return day;
        }

        private timetable getTimetableOfHelper(int id)
        {
            //Make call to the WebAPI, and parse JSON input into a 'timetable'
object
            try
            {
                int timetableId = getUserDetails(id, true).timetableId;
                string url = "http://178.62.87.28:600/api/tt/gt" +
Convert.ToString(timetableId);
                var client = new RestClient(url);
                var request = new RestRequest();
                request.Method = Method.GET;
                request.AddHeader("Accept", "application/json");
                request.Parameters.Clear();
                var response = client.Execute(request);
                string result = response.Content;
                timetable toReturn =
JsonConvert.DeserializeObject<timetable>(result);
                return toReturn;
            }
            catch
            {
                Toast.MakeText(BaseContext, "Ensure that you are connected to the
internet", ToastLength.Short).Show();
                StartActivity(typeof(MainActivity));
                return new timetable();
            }
        }
        private user getUserDetails(int id, bool helper)
        {
            //Make call to the WebAPI, and parse JSON input into a 'user' object
            try
            {
                string url = String.Empty;
                if (helper)
                    url = "http://178.62.87.28:600/api/ach/users/gi" +
Convert.ToString(id);
                else
```

```
                    url = "http://178.62.87.28:600/api/ace/users/gi" +
Convert.ToString(id);
                var client = new RestClient(url);
                var request = new RestRequest();
                request.Method = Method.GET;
                request.AddHeader("Accept", "application/json");
                request.Parameters.Clear();
                var response = client.Execute(request);
                string result = response.Content;
                user toReturn = JsonConvert.DeserializeObject<user>(result);
                return toReturn;
            }
            catch
            {
                Toast.MakeText(BaseContext, "Ensure that you are connected to the
internet", ToastLength.Short).Show();
                StartActivity(typeof(MainActivity));
                return new user();
            }
        }
    }
}
```

SetTimetableWeeks.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Android.App;
using Android.Content;
using Android.OS;
using Android.Runtime;
using Android.Views;
using Android.Widget;

using Android_Application.Backend;
using RestSharp;
using System.Globalization;

namespace Android_Application.Activities
{
    [Activity(Label = "setTimetableWeeks")]
    public class setTimetableWeeks : ListActivity
    {
        int helperId;
        string[] listOfThingsToDisplay;
        protected override void OnCreate(Bundle savedInstanceState)
        {
            try
            {
```

```
                //Gets parameters from the previous screen
                base.OnCreate(savedInstanceState);
                helperId = Intent.GetIntExtra("helperId", 8);
                listOfThingsToDisplay = new string[8];

                //Places the next 8 weeks in the screen and creates an array of
them
                for (int i = 0; i < listOfThingsToDisplay.Length; i++)
                {
                    DateTime dt = DateTime.Now.AddDays(7 *
i).StartOfWeek(DayOfWeek.Monday);
                    listOfThingsToDisplay[i] = dt.ToString("MMMM dd, yyyy");
                }

                //display them
                RequestWindowFeature(WindowFeatures.NoTitle);
                ListAdapter = new ArrayAdapter<String>(this,
Android.Resource.Layout.SimpleListItem1, listOfThingsToDisplay);
            }
            catch
            {
                Toast.MakeText(BaseContext, "Ensure that you are connected to the
internet", ToastLength.Short).Show();
                StartActivity(typeof(MainActivity));
            }
        }
        protected override void OnListItemClick(ListView l, View v, int position,
long id)
        {
            //Start the activity to choose which day's timetable to change, passing
the week beginning date.
            base.OnListItemClick(l, v, position, id);
            DateTime toPass = DateTime.ParseExact(listOfThingsToDisplay[position],
"MMMM dd, yyyy", CultureInfo.InvariantCulture);
            string toPassString = toPass.ToString("dd/MM/yyyy");
            var newActivity = new Intent(this, typeof(setTimetableDays));
            newActivity.PutExtra("helperId", helperId);
            newActivity.PutExtra("dateTime", toPassString);
            StartActivity(newActivity);
        }
    }
}
```

WelcomeElderly.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Android.App;
using Android.Content;
```

```
using Android.OS;
using Android.Runtime;
using Android.Views;
using Android.Widget;

using RestSharp;
using Newtonsoft.Json;
using Android_Application.Types;

namespace Android_Application.Activities
{
    [Activity(Label = "welcomeElderly")]
    public class welcomeElderly : Activity
    {
        /// <summary>
        /// What goes into this:
        ///     the id of the user
        ///
        /// </summary>
        /// <param name="savedInstanceState"></param>
        int currentUserId;
        protected override void OnCreate(Bundle savedInstanceState)
        {
            try
            {
                //Gets parameters from previous activity
                base.OnCreate(savedInstanceState);
                currentUserId = Intent.GetIntExtra("currentUserId", 20);
                bool sentBack = Intent.GetBooleanExtra("sentBack", false);

                //If appointment just booked, then go to list of appointments
                if (sentBack)
                {
                    click2();
                }

                //Sets up view
                RequestWindowFeature(WindowFeatures.NoTitle);
                SetContentView(Resource.Layout.loginWelcomeElderly);

                //Sets up items in view
                TextView nameOfUser =
FindViewById<TextView>(Resource.Id.welcomeElderlyName);
                Button listOfAppointments =
FindViewById<Button>(Resource.Id.welcomeElderlyListOfAppointments);
                Button bookAnAppointment =
FindViewById<Button>(Resource.Id.welcomeElderlyBookAnAppointment);


                //Populates name of user
                user example = getUserDetails(currentUserId, false);
```

```
                nameOfUser.Text = example.firstName + " " + example.surname;

                //Sets up event handlers
                listOfAppointments.Click += ListOfAppointments_Click;
                bookAnAppointment.Click += BookAnAppointment_Click;
            }
            catch
            {
                Toast.MakeText(BaseContext, "Ensure that you are connected to the
internet", ToastLength.Short).Show();
                StartActivity(typeof(MainActivity));
            }
        }

        private void BookAnAppointment_Click(object sender, EventArgs e)
        {
            //Starts the book appointment activity
            var newActivity = new Intent(this, typeof(bookNewAppointment));
            newActivity.PutExtra("currentUserId", currentUserId);
            StartActivity(newActivity);
        }

        private void ListOfAppointments_Click(object sender, EventArgs e)
        {
            //Starts the list of appointments activity
            var newActivity = new Intent(this, typeof(listOfAppointments));
            newActivity.PutExtra("currentUserId", currentUserId);
            newActivity.PutExtra("currentUserHelper", false);
            StartActivity(newActivity);
        }
        private void click2()
        {
            //Starts list of appointment activity
            Intent newActivity = new Intent(this, typeof(listOfAppointments));
            newActivity.AddFlags(ActivityFlags.ClearTop);
            newActivity.PutExtra("currentUserId", currentUserId);
            newActivity.PutExtra("currentUserHelper", false);
            StartActivity(newActivity);
        }
        private user getUserDetails(int id, bool helper)
        {
            //Make call to the WebAPI, and parse JSON input into a 'user' object
            try
            {
                string url = String.Empty;
                if (helper)
                    url = "http://178.62.87.28:600/api/ach/users/gi" +
Convert.ToString(id);
                else
                    url = "http://178.62.87.28:600/api/ace/users/gi" +
Convert.ToString(id);
```

```
                var client = new RestClient(url);
                var request = new RestRequest();
                request.Method = Method.GET;
                request.AddHeader("Accept", "application/json");
                request.Parameters.Clear();
                var response = client.Execute(request);
                string result = response.Content;
                user toReturn = JsonConvert.DeserializeObject<user>(result);
                return toReturn;
            }
            catch
            {
                Toast.MakeText(BaseContext, "Ensure that you are connected to the
internet", ToastLength.Short).Show();
                StartActivity(typeof(MainActivity));
                return new user();
            }
        }
    }
}
```

WelcomeHelpers.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Android.App;
using Android.Content;
using Android.OS;
using Android.Runtime;
using Android.Views;
using Android.Widget;

using RestSharp;
using Newtonsoft.Json;
using SQLite;
using Android_Application.Types;

namespace Android_Application.Activities
{
    [Activity(Label = "welcomeHelper")]
    public class welcomeHelper : Activity
    {
        int currentUserId;
        protected override void OnCreate(Bundle savedInstanceState)
        {
            try
            {
                //Sets up the view
                base.OnCreate(savedInstanceState);
```

```
                currentUserId = Intent.GetIntExtra("currentUserId", 20);
                RequestWindowFeature(WindowFeatures.NoTitle);
                SetContentView(Resource.Layout.loginWelcomeHelpers);

                //Sets up things in the view
                TextView WelcomeHelperName =
FindViewById<TextView>(Resource.Id.welcomeHelperName);
                Button listOfAppointments =
FindViewById<Button>(Resource.Id.welcomeHelperListOfAppointments);
                Button setTimetable =
FindViewById<Button>(Resource.Id.welcomeHelperSetTimetable);

                //populates the name
                user example = getUserDetails(currentUserId, true);
                WelcomeHelperName.Text = example.firstName + " " + example.surname;

                //Sets up event handlers
                listOfAppointments.Click += ListOfAppointments_Click;
                setTimetable.Click += SetTimetable_Click;
            }
            catch
            {
                Toast.MakeText(BaseContext, "Ensure that you are connected to the
internet", ToastLength.Short).Show();
                StartActivity(typeof(MainActivity));
            }
        }

        private void SetTimetable_Click(object sender, EventArgs e)
        {
            //Goes into set timetable activity
            var newActivity = new Intent(this, typeof(setTimetableWeeks));
            newActivity.PutExtra("helperId", currentUserId);
            StartActivity(newActivity);
        }

        private void ListOfAppointments_Click(object sender, EventArgs e)
        {
            //Goes into the list of appointments activity
            var newActivity = new Intent(this, typeof(listOfAppointments));
            newActivity.PutExtra("currentUserId", currentUserId);
            newActivity.PutExtra("currentUserHelper", true);
            StartActivity(newActivity);
        }

        private user getUserDetails(int id, bool helper)
        {
            //Make call to the WebAPI, and parse JSON input into a 'user' object
            try
            {
                string url = String.Empty;
```

```
                if (helper)
                    url = "http://178.62.87.28:600/api/ach/users/gi" +
Convert.ToString(id);
                else
                    url = "http://178.62.87.28:600/api/ace/users/gi" +
Convert.ToString(id);
                var client = new RestClient(url);
                var request = new RestRequest();
                request.Method = Method.GET;
                request.AddHeader("Accept", "application/json");
                request.Parameters.Clear();
                var response = client.Execute(request);
                string result = response.Content;
                user toReturn = JsonConvert.DeserializeObject<user>(result);
                return toReturn;
            }
            catch
            {
                Toast.MakeText(BaseContext, "Ensure that you are connected to the
internet", ToastLength.Short).Show();
                StartActivity(typeof(MainActivity));
                return new user();
            }
        }
    }
}
```

Resources

Layouts

AppointmentDetails.axml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:text="Name of Helper:"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/textView1" />
    <TextView
        android:text=""
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/nameOfHelper" />
    <TextView
        android:text="Date of Appointment:"
        android:textAppearance="?android:attr/textAppearanceLarge"
```

```
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/textView3" />
    <TextView
        android:text=""
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/dateOfAppointment" />
    <TextView
        android:text="Time of Appointment:"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/textView5" />
    <TextView
        android:text=""
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/timeOfAppointment" />
    <Button
        android:text="Confirm"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/confirmAppointment" />
</LinearLayout>
```

AppointmentDetailsScreen.axml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/linearLayout2">
        <TextView
            android:text="Appointment Date: "
            android:textAppearance="?android:attr/textAppearanceMedium"
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:id="@+id/textView1" />
        <TextView
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:id="@+id/appointmentDetailsDate" />
    </LinearLayout>
    <LinearLayout
        android:orientation="horizontal"
```

```
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/linearLayout2">
        <TextView
            android:text="Appointment Time: "
            android:textAppearance="?android:attr/textAppearanceMedium"
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:id="@+id/textView1" />
        <TextView
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:id="@+id/appointmentDetailsTime" />
    </LinearLayout>
    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/linearLayout2">
        <TextView
            android:text="Name: "
            android:textAppearance="?android:attr/textAppearanceMedium"
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:id="@+id/textView1" />
        <TextView
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:id="@+id/appointmentDetailsName" />
    </LinearLayout>
    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/linearLayout2">
        <TextView
            android:text="Contact Number: "
            android:textAppearance="?android:attr/textAppearanceMedium"
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:id="@+id/textView1" />
        <TextView
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:id="@+id/appointmentDetailsContactNumber" />
    </LinearLayout>
    <TextView
        android:text="Address of Appointment: "
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
        android:id="@+id/textView1" />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:visibility="visible"
        android:id="@+id/appointmentDetailsFirstLineOfAddressOfAppointment" />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/appointmentDetailsSecondLineOfAddressOfAppointment" />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/appointmentDetailsCityAddressOfAppointment" />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/appointmentDetailsPostalCodeOfAppointment" />
    <TextView
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/textView4" />
    <Button
        android:text="Cancel Appointment"
        android:layout_width="match_parent"
        android:layout_height="85dp"
        android:visibility="visible"
        android:id="@+id/cancelAppointment" />
    <Button
        android:text="Set Rating"
        android:layout_width="match_parent"
        android:layout_height="85.0dp"
        android:enabled="false"
        android:visibility="visible"
        android:id="@+id/setRating" />
    <Button
        android:text="Set Review"
        android:layout_width="match_parent"
        android:layout_height="85dp"
        android:enabled="false"
        android:visibility="visible"
        android:id="@+id/setReview" />
</LinearLayout>
```

BookAppointment.axml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
```

```
</LinearLayout>
```

HelperDetails.axml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/helperDetailsName"
        android:gravity="center"
        android:text="Test" />
    <TextView
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/textView1" />
    <TextView
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/helperDetailsFirstLine"
        android:gravity="center" />
    <TextView
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/helperDetailsSecondLine"
        android:gravity="center" />
    <TextView
        android:text="Large Text"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/helperDetailsCity"
        android:gravity="center" />
    <TextView
        android:text="Large Text"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/helperDetailsPostcode"
        android:gravity="center" />
    <TextView
        android:text="Large Text"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
```

```
        android:id="@+id/helperDetailsCountry"
        android:gravity="center" />
    <TextView
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/textView1" />
    <TextView
        android:text="Large Text"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/helperDetailsContactNumber"
        android:gravity="center" />
    <TextView
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/textView1" />
    <Button
        android:text="See Reviews"
        android:layout_width="match_parent"
        android:layout_height="85dp"
        android:id="@+id/helperDetailsSeeReviews"
         />
    <Button
        android:layout_width="match_parent"
        android:layout_height="85dp"
        android:id="@+id/helperDetailsBookAppointment"
        android:text="Book Appointment"
         />
</LinearLayout>
```

Login.axml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/textView3" />
    <TextView
        android:text="Email Address"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/textView1" />
    <TextView
        android:textAppearance="?android:attr/textAppearanceMedium"
```

```
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/textView3" />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/loginEmailAddress" />
    <TextView
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/textView4" />
    <TextView
        android:text="Password"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/textView2" />
    <TextView
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/textView3" />
    <EditText
        android:inputType="textPassword"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/loginPassword" />
    <TextView
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/textView3" />
    <Button
        android:text="Login as Helper"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/loginLoginAsHelper" />
    <Button
        android:text="Login As Elderly"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/loginLoginAsElderly" />
    <TextView
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/textView3" />
    <TextView
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content"
        android:id="@+id/jnksdks" />
</LinearLayout>
```

LoginOrRegister.axml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/textView1" />
    <Button
        android:text="Login"
        android:layout_width="match_parent"
        android:layout_height="200dp"
        android:id="@+id/loginOrRegisterLogin"
        android:textSize="12pt" />
    <TextView
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/textView1" />
    <Button
        android:text="Register"
        android:layout_width="match_parent"
        android:layout_height="200dp"
        android:id="@+id/loginOrRegisterRegister"
        android:textSize="12pt" />
</LinearLayout>
```

loginWelcomeElderly.axml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:text="Welcome: "
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:id="@+id/textView1"
        android:textAlignment="center" />
    <TextView
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
```

```
        android:id="@+id/welcomeElderlyName" />
    <TextView
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/textView2" />
    <Button
        android:text="Book an Appointment"
        android:layout_width="match_parent"
        android:layout_height="85dp"
        android:id="@+id/welcomeElderlyBookAnAppointment"
         />
    <Button
        android:text="List of Appoinments"
        android:layout_width="match_parent"
        android:layout_height="85dp"
        android:id="@+id/welcomeElderlyListOfAppointments"
         />
</LinearLayout>
```

loginWelcomeHelpers.axml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:text="Welcome: "
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:id="@+id/textView1"
        android:textAlignment="center" />
    <TextView
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:id="@+id/welcomeHelperName" />
    <TextView
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/textView2" />
    <Button
        android:text="Set Timetable"
        android:layout_width="match_parent"
        android:layout_height="85dp"
        android:id="@+id/welcomeHelperSetTimetable"/>
```

```
    <Button
        android:text="List of Appoinments"
        android:layout_width="match_parent"
        android:layout_height="85dp"
        android:id="@+id/welcomeHelperListOfAppointments" />
</LinearLayout>
```

ratingsScreen.axml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:text="Name of Helper"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/textView1" />
    <TextView
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/ratingNameOfHelper" />
    <TextView
        android:text="Set Rating"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/textView3" />
    <TextView
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/textView2" />
    <RatingBar
        android:layout_width="244.5dp"
        android:layout_height="wrap_content"
        android:id="@+id/ratingBar"
        android:max="5"
        android:progress="0"
        android:secondaryProgress="0" />
    <Button
        android:text="Submit"
        android:layout_width="match_parent"
        android:layout_height="85dp"
        android:id="@+id/submitRating"
        android:textSize="20pt" />
</LinearLayout>
```

Register.axml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

```
android:orientation="vertical"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:minWidth="25px"
android:minHeight="25px"
android:weightSum="100">
<ScrollView
    android:minWidth="25px"
    android:minHeight="25px"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/scrollView1"
    android:layout_weight="100">
    <LinearLayout
        android:orientation="vertical"
        android:minWidth="25px"
        android:minHeight="25px"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/linearLayout1">
        <TextView
            android:textAppearance="?android:attr/textAppearanceMedium"
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:id="@+id/textView2" />
        <LinearLayout
            android:orientation="horizontal"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/linearLayout2">
            <TextView
                android:text="First Name: "
                android:textAppearance="?android:attr/textAppearanceMedium"
                android:layout_width="wrap_content"
                android:layout_height="match_parent"
                android:id="@+id/textView1" />
            <EditText
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:id="@+id/registerFirstName" />
        </LinearLayout>
        <TextView
            android:textAppearance="?android:attr/textAppearanceMedium"
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:id="@+id/textView2" />
        <LinearLayout
            android:orientation="horizontal"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/linearLayout2">
```

```
    <TextView
        android:text="Surname:"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:id="@+id/textView1" />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/registerSurname" />
</LinearLayout>
<TextView
    android:textAppearance="?android:attr/textAppearanceMedium"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:id="@+id/textView2" />
<LinearLayout
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/linearLayout2">
    <TextView
        android:text="Email Address:"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:id="@+id/textView1" />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/registerEmailAddress" />
</LinearLayout>
<TextView
    android:textAppearance="?android:attr/textAppearanceMedium"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:id="@+id/textView2" />
<LinearLayout
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/linearLayout2">
    <TextView
        android:text="Username:"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:id="@+id/textView1" />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="match_parent"
```

```
        android:id="@+id/registerUsername" />
</LinearLayout>
<TextView
    android:textAppearance="?android:attr/textAppearanceMedium"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:id="@+id/textView2" />
<LinearLayout
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/linearLayout2">
    <TextView
        android:text="Password:"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:id="@+id/textView1" />
    <EditText
        android:inputType="textPassword"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/registerPassword" />
</LinearLayout>
<TextView
    android:textAppearance="?android:attr/textAppearanceMedium"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:id="@+id/textView2" />
<LinearLayout
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/linearLayout2">
    <TextView
        android:text="Confirm Password:"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:id="@+id/textView1" />
    <EditText
        android:inputType="textPassword"
        android:id="@+id/registerConfirmPassword"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</LinearLayout>
<TextView
    android:textAppearance="?android:attr/textAppearanceMedium"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:id="@+id/textView2" />
```

```
<LinearLayout
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/linearLayout2">
    <TextView
        android:text="First Line of Address:"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:id="@+id/textView1" />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/registerFirstLine" />
</LinearLayout>
<TextView
    android:textAppearance="?android:attr/textAppearanceMedium"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:id="@+id/textView2" />
<LinearLayout
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/linearLayout2">
    <TextView
        android:text="Second Line of Address:"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:id="@+id/textView1" />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/registerSecondLine" />
</LinearLayout>
<TextView
    android:textAppearance="?android:attr/textAppearanceMedium"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:id="@+id/textView2" />
<LinearLayout
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/linearLayout2">
    <TextView
        android:text="City:"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:layout_width="wrap_content"
```

```
        android:layout_height="match_parent"
        android:id="@+id/textView1" />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/registerCity" />
</LinearLayout>
<TextView
    android:textAppearance="?android:attr/textAppearanceMedium"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:id="@+id/textView2" />
<LinearLayout
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/linearLayout2">
    <TextView
        android:text="Country: "
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:id="@+id/textView1" />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/registerCountry" />
</LinearLayout>
<TextView
    android:textAppearance="?android:attr/textAppearanceMedium"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:id="@+id/textView2" />
<LinearLayout
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/linearLayout2">
    <TextView
        android:text="Postal Code:"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:id="@+id/textView1" />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/registerPostalCode" />
</LinearLayout>
<TextView
    android:textAppearance="?android:attr/textAppearanceMedium"
```

```
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:id="@+id/textView2" />
        <LinearLayout
            android:orientation="horizontal"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/linearLayout2">
            <TextView
                android:text="Telephone Number: "
                android:textAppearance="?android:attr/textAppearanceMedium"
                android:layout_width="wrap_content"
                android:layout_height="match_parent"
                android:id="@+id/textView1" />
            <EditText
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:id="@+id/registerTelephoneNumber" />
        </LinearLayout>
        <TextView
            android:textAppearance="?android:attr/textAppearanceMedium"
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:id="@+id/textView2" />
        <LinearLayout
            android:orientation="horizontal"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/linearLayout2"
            android:weightSum="100">
            <Button
                android:text="Register as Helper"
                android:layout_width="wrap_content"
                android:layout_height="match_parent"
                android:id="@+id/registerRegisterAsHelper"
                android:layout_weight="50" />
            <Button
                android:text="Register as Elderly"
                android:layout_width="wrap_content"
                android:layout_height="match_parent"
                android:id="@+id/registerRegisterAsElderly"
                android:layout_weight="50" />
        </LinearLayout>
        <TextView
            android:textAppearance="?android:attr/textAppearanceMedium"
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:id="@+id/textView2" />
    </LinearLayout>
  </ScrollView>
</LinearLayout>
```

ReviewScreen.axml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:text="Name of Helper"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/textView1" />
    <TextView
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/reviewNameOfHelper" />
    <TextView
        android:text="Set Review"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/textView3" />
    <EditText
        android:inputType="textMultiLine"
        android:layout_width="match_parent"
        android:layout_height="200dp"
        android:id="@+id/reviewComment" />
    <Button
        android:text="Submit"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/submitReview" />
</LinearLayout>
```

setTimetableTimes.axml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:minWidth="25px"
    android:minHeight="25px">
    <LinearLayout
        android:orientation="horizontal"
        android:minWidth="25px"
        android:minHeight="25px"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/linearLayout1">
        <TextView
            android:text="12am to 2am"
```

```
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:id="@+id/textView1" />
    <Switch
        android:layout_width="match_parent"
        android:gravity="right"
        android:layout_height="match_parent"
        android:id="@+id/a02" />
</LinearLayout>
<LinearLayout
    android:orientation="horizontal"
    android:minWidth="25px"
    android:minHeight="25px"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/linearLayout1">
    <TextView
        android:text="2am to 4am"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:id="@+id/textView1" />
    <Switch
        android:layout_width="match_parent"
        android:gravity="right"
        android:layout_height="match_parent"
        android:id="@+id/a24" />
</LinearLayout>
<LinearLayout
    android:orientation="horizontal"
    android:minWidth="25px"
    android:minHeight="25px"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/linearLayout1">
    <TextView
        android:text="4am to 6am"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:id="@+id/textView1" />
    <Switch
        android:layout_width="match_parent"
        android:gravity="right"
        android:layout_height="match_parent"
        android:id="@+id/a46" />
</LinearLayout>
<LinearLayout
    android:orientation="horizontal"
    android:minWidth="25px"
```

```
        android:minHeight="25px"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/linearLayout1">
        <TextView
            android:text="6am to 8am"
            android:textAppearance="?android:attr/textAppearanceLarge"
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:id="@+id/textView1" />
        <Switch
            android:layout_width="match_parent"
            android:gravity="right"
            android:layout_height="match_parent"
            android:id="@+id/a68" />
    </LinearLayout>
    <LinearLayout
        android:orientation="horizontal"
        android:minWidth="25px"
        android:minHeight="25px"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/linearLayout1">
        <TextView
            android:text="8am to 10am"
            android:textAppearance="?android:attr/textAppearanceLarge"
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:id="@+id/textView1" />
        <Switch
            android:layout_width="match_parent"
            android:gravity="right"
            android:layout_height="match_parent"
            android:id="@+id/a810" />
    </LinearLayout>
    <LinearLayout
        android:orientation="horizontal"
        android:minWidth="25px"
        android:minHeight="25px"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/linearLayout1">
        <TextView
            android:text="10am to 12pm"
            android:textAppearance="?android:attr/textAppearanceLarge"
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:id="@+id/textView1" />
        <Switch
            android:layout_width="match_parent"
            android:gravity="right"
```

```
        android:layout_height="match_parent"
        android:id="@+id/a1012" />
</LinearLayout>
<LinearLayout
    android:orientation="horizontal"
    android:minWidth="25px"
    android:minHeight="25px"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/linearLayout1">
    <TextView
        android:text="12pm to 2pm"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:id="@+id/textView1" />
    <Switch
        android:layout_width="match_parent"
        android:gravity="right"
        android:layout_height="match_parent"
        android:id="@+id/a1214" />
</LinearLayout>
<LinearLayout
    android:orientation="horizontal"
    android:minWidth="25px"
    android:minHeight="25px"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/linearLayout1">
    <TextView
        android:text="2pm to 4pm"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:id="@+id/textView1" />
    <Switch
        android:layout_width="match_parent"
        android:gravity="right"
        android:layout_height="match_parent"
        android:id="@+id/a1416" />
</LinearLayout>
<LinearLayout
    android:orientation="horizontal"
    android:minWidth="25px"
    android:minHeight="25px"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/linearLayout1">
    <TextView
        android:text="4pm to 6pm"
        android:textAppearance="?android:attr/textAppearanceLarge"
```

```
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:id="@+id/textView1" />
    <Switch
        android:layout_width="match_parent"
        android:gravity="right"
        android:layout_height="match_parent"
        android:id="@+id/a1618" />
</LinearLayout>
<LinearLayout
    android:orientation="horizontal"
    android:minWidth="25px"
    android:minHeight="25px"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/linearLayout1">
    <TextView
        android:text="6pm to 8pm"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:id="@+id/textView1" />
    <Switch
        android:layout_width="match_parent"
        android:gravity="right"
        android:layout_height="match_parent"
        android:id="@+id/a1820" />
</LinearLayout>
<LinearLayout
    android:orientation="horizontal"
    android:minWidth="25px"
    android:minHeight="25px"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/linearLayout1">
    <TextView
        android:text="8pm to 10pm"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:id="@+id/textView1" />
    <Switch
        android:layout_width="match_parent"
        android:gravity="right"
        android:layout_height="match_parent"
        android:id="@+id/a2022" />
</LinearLayout>
<LinearLayout
    android:orientation="horizontal"
    android:minWidth="25px"
    android:minHeight="25px"
```

```
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/linearLayout1">
        <TextView
            android:text="10pm to 12am"
            android:textAppearance="?android:attr/textAppearanceLarge"
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:id="@+id/textView1" />
        <Switch
            android:layout_width="match_parent"
            android:gravity="right"
            android:layout_height="match_parent"
            android:id="@+id/a2224" />
    </LinearLayout>
    <Button
        android:text="Submit"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/setTimetableSubmit" />
</LinearLayout>
```

Backend

DateTimeExtensions.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Android.App;
using Android.Content;
using Android.OS;
using Android.Runtime;
using Android.Views;
using Android.Widget;

namespace Android_Application.Backend
{
    public static class DateTimeExtensions
    {
        public static DateTime StartOfWeek(this DateTime dt, DayOfWeek startOfWeek
= DayOfWeek.Monday) // Gets the date of the start of the week, given a date
        {
            int diff = dt.DayOfWeek - startOfWeek;
            if (diff < 0)
            {
                diff += 7;
            }
            return dt.AddDays(-1 * diff).Date;
```

```
        }
    }
}
```

Dialogs.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Android.App;
using Android.Content;
using Android.OS;
using Android.Runtime;
using Android.Views;
using Android.Widget;

namespace Android_Application.Backend
{
    class Dialogs
    {
        internal Dialogs(string title, string content, Context y)
        {
            AlertDialog.Builder a = new AlertDialog.Builder(y);
            a.SetTitle(title);
            a.SetMessage(content);
            Dialog dialog = a.Create();
            dialog.Show();
        }
    }
}
```

MD5Hash.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Android.App;
using Android.Content;
using Android.OS;
using Android.Runtime;
using Android.Views;
using Android.Widget;
using System.Security.Cryptography;
namespace Android_Application.Backend
{
    class MD5Hash
    {
        //Hashes something with MD5 - using the .NET crypto libraries.
```

```
        //UTILISES SAMPLE CODE FROM: https://msdn.microsoft.com/en-
us/library/system.security.cryptography.md5(v=vs.110).aspx
        internal static string MD5HashReturn(string text)
        {
            MD5 md5 = new MD5CryptoServiceProvider();

            //compute hash from the bytes of text
            md5.ComputeHash(ASCIIEncoding.ASCII.GetBytes(text));

            //get hash result after compute it
            byte[] result = md5.Hash;

            StringBuilder strBuilder = new StringBuilder();
            for (int i = 0; i < result.Length; i++)
            {
                //change it into 2 hexadecimal digits
                //for each byte
                strBuilder.Append(result[i].ToString("x2"));
            }

            return strBuilder.ToString();
        }
    }
}
```

Types

Appointment.cs

```
using System;

namespace Android_Application.Types
{
    public class appointment : Object // Inherits from the object to get the
compare function
    {
        private int _id;
        private int _helperId;
        private int _elderlyId;
        private DateTime _dateAndTime;
        private DateTime _dateCreated;
        private int _ratingId;
        private int _reviewId;
        public int id {
            get
            {
                return _id;
            }
            set
            {
                _id = value;
            }
```

```
}
public int helperId
{
    get
    {
        return _helperId;
    }
    set
    {
        _helperId = value;
    }
}
public int elderlyId
{
    get
    {
        return _elderlyId;
    }
    set
    {
        _elderlyId = value;
    }
}
public DateTime dateAndTime
{
    get
    {
        return _dateAndTime;
    }
    set
    {
        _dateAndTime = value;
    }
}
public DateTime dateCreated
{
    get
    {
        return _dateCreated;
    }
    set
    {
        _dateCreated = value;
    }
}
public int ratingId
{
    get
    {
        return _ratingId;
    }
```

```
            set
            {
                _ratingId = value;
            }
        }
        public int reviewId
        {
            get
            {
                return _reviewId;
            }
            set
            {
                _reviewId = value;
            }
        }

        public static bool Equals(appointment a, appointment b) // Check if two
appointments are the same
        {
            if (a.id != b.id)
                return false;
            if (a.helperId != b.helperId)
                return false;
            if (a.elderlyId != b.elderlyId)
                return false;
            if (DateTime.Compare(a.dateAndTime, b.dateAndTime) != 0)
                return false;
            if (DateTime.Compare(a.dateCreated, b.dateCreated) != 0)
                return false;
            if (a.ratingId != b.ratingId)
                return false;
            if (a.reviewId != b.reviewId)
                return false;
            return true;


        }
    }
}
```

Day.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace Android_Application.Types
{
    public class day
    {
```

```
        private int _id;
        private bool[] _timesFree;
        public int id {
            get
            {
                return _id;
            }
            set
            {
                _id = value;
            }
        }
        public bool[] timesFree
        {
            get
            {
                return _timesFree;
            }
            set
            {
                _timesFree = value;
            }
        }

    }
}
```

Rating.cs

```
using System;
namespace Android_Application.Types
{
    public class rating
    {
        private int _id;
        private int _starRating;
        private int _helperId;

        public int id
        {
            get
            {
                return _id;
            }
            set
            {
                _id = value;
            }
        }
        public int starRating
        {
```

```
            get
            {
                return _starRating;
            }
            set
            {
                _starRating = value;
            }
        }
        public int helperId
        {
            get
            {
                return _helperId;
            }
            set
            {
                _helperId = value;
            }
        }
    }
}
```

Review.cs

```
using System;
namespace Android_Application.Types
{
    public class review
    {
        private int _id;
        private int _helperId;
        private string _comment;

        public int id
        {
            get
            {
                return _id;
            }
            set
            {
                _id = value;
            }
        }
        public int helperId
        {
            get
            {
                return _helperId;
            }
            set
```

```
            {
                _helperId = value;
            }
        }
        public string comment
        {
            get
            {
                return _comment;
            }
            set
            {
                _comment = value;
            }
        }

    }
}
```

SMS.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace Android_Application.Types
{
    public class sms
    {
        private string _From;
        private string _Body;
        public string From {
            get
            {
                return _From;
            }
            set
            {
                _From = value;
            }
        }
        public string Body {
            get
            {
                return _Body;
            }
            set
            {
                _Body = value;
            }
        }
```

```
    }
}
```

SQLLiteLogin.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Android.App;
using Android.Content;
using Android.OS;
using Android.Runtime;
using Android.Views;
using Android.Widget;

using SQLite;
using SQLitePCL;
namespace Android_Application.Types
{
    public class SQLLiteLogin
    {
        [PrimaryKey, AutoIncrement]
        public int id { get; set; }
        public string emailAddress { get; set; }
        public string password { get; set; }
        public bool helper { get; set; }
        public SQLLiteLogin()
        { }

    }
}
```

Timetable.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace Android_Application.Types
{
    public class names
    {
        private DateTime _weekBeginning;
        private week _week;
        public DateTime weekBeginning {
            get
            {
                return _weekBeginning;
```

```
        }
        set
        {
            _weekBeginning = value;
        }
    }
    public week week {
        get
        {
            return _week;
        }
        set
        {
            _week = value;
        }
    }

}
public class timetable
{
    private int _id;
    private week _defaultWeek;
    private List<names> _weeks = new List<names>();
    public int id {
        get
        {
            return _id;
        }
        set
        {
            _id = value;
        }
    }
    public week defaultWeek {
        get
        {
            return _defaultWeek;
        }
        set
        {
            _defaultWeek = value;
        }
    }
    public List<names> weeks {
        get
        {
            return _weeks;
        }
        set
        {
            _weeks = value;
```

```
            }
        }

    }
}
```

TwilioRegistration.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Android.App;
using Android.Content;
using Android.OS;
using Android.Runtime;
using Android.Views;
using Android.Widget;

namespace Android_Application.Types
{
    public class TwilioRegistration
    {
        public string Number { get; set; }
        public string friendlyName { get; set; }
    }
}
```

User.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace Android_Application.Types
{
    public class user
    {
        private int _id;
        private string _username;
        private string _firstname;
        private string _surname;
        private string _password;
        private string _emailAddress;
        private string _firstLineOfAddress;
        private string _secondLineOfAddress;
        private string _telephoneNumber;
        private string _postalCode;
        private string _city;
```

```
private string _country;
private bool _helper = false;
private int _timetableId;
private double _distance = double.MaxValue;
public int id {
    get
    {
        return _id;
    }
    set
    {
        _id = value;
    }
}
public string username {
    get
    {
        return _username;
    }
    set
    {
        _username = value;
    }
}
public string firstName {
    get
    {
        return _firstname;
    }
    set
    {
        _firstname = value;
    }
}
public string surname {
    get
    {
        return _surname;
    }
    set
    {
        _surname = value;
    }
}
public string password {
    get
    {
        return _password;
    }
    set
    {
```

```
            _password = value;
        }
    }
    public string emailAddress {
        get
        {
            return _emailAddress;
        }
        set
        {
            _emailAddress = value;
        }
    }
    public string firstLineOfAddress {
        get
        {
            return _firstLineOfAddress;
        }
        set
        {
            _firstLineOfAddress = value;
        }
    }
    public string secondLineOfAddress{
        get
        {
            return _secondLineOfAddress;
        }
        set
        {
            _secondLineOfAddress = value;
        }
    }
    public string telephoneNumber {
        get
        {
            return _telephoneNumber;
        }
        set
        {
            _telephoneNumber = value;
        }
    }
    public string postalCode{
        get
        {
            return _postalCode;
        }
        set
        {
            _postalCode = value;
```

```
        }
    }
    public string city {
        get
        {
            return _city;
        }
        set
        {
            _city = value;
        }
    }
    public string country {
        get
        {
            return _country;
        }
        set
        {
            _country = value;
        }
    }
    public bool helper {
        get
        {
            return _helper;
        }
        set
        {
            _helper = value;
        }
    }
    public int timetableId {
        get
        {
            return _timetableId;
        }
        set
        {
            _timetableId = value;
        }
    }

    public double distance {
        get
        {
            return _distance;
        }
        set
        {
            _distance = value;
```

```
            }
        }
    }
}
```

userToBeVerified.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace Android_Application.Types
{
    public class userToBeVerified
    {
        private string _emailAddress;
        private string _password;
        private bool _helper;
        public string emailAddress {
            get
            {
                return _emailAddress;
            }
            set
            {
                _emailAddress = value;
            }
        }
        public string password {
            get
            {
                return _password;
            }
            set
            {
                _password = value;
            }
        }
        public bool helper {
            get
            {
                return _helper;
            }
            set
            {
                _helper = value;
            }
        }
```

```
    }
}
```

Week.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace Android_Application.Types
{
    public class week
    {
        private int _id;
        private day _monday;
        private day _tuesday;
        private day _wednesday;
        private day _thursday;
        private day _friday;
        private day _saturday;
        private day _sunday;
        public int id {
            get
            {
                return _id;
            }
            set
            {
                _id = value;
            }
        }
        public day monday {
            get
            {
                return _monday;
            }
            set
            {
                _monday = value;
            }
        }
        public day tuesday {
            get
            {
                return _tuesday;
            }
            set
            {
                _tuesday = value;
            }
```

```
}
public day wednesday {
    get
    {
        return _wednesday;
    }
    set
    {
        _wednesday = value;
    }
}
public day thursday {
    get
    {
        return _thursday;
    }
    set
    {
        _thursday = value;
    }
}
public day friday {
    get
    {
        return _friday;
    }
    set
    {
        _friday = value;
    }
}
public day saturday {
    get
    {
        return _saturday;
    }
    set
    {
        _saturday = value;
    }
}
public day sunday {
    get
    {
        return _sunday;
    }
    set
    {
        _sunday = value;
    }
}
```

```
    }
}
```

Root Folder

App.config

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
    <runtime>
        <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
            <dependentAssembly>
                <assemblyIdentity name="System.Net.Http"
publicKeyToken="B03F5F7F11D50A3A" culture="neutral"/>
                <bindingRedirect oldVersion="0.0.0.0-4.0.0.0"
newVersion="4.0.0.0"/>
            </dependentAssembly>
            <dependentAssembly>
                <assemblyIdentity name="System.Runtime"
publicKeyToken="B03F5F7F11D50A3A" culture="neutral"/>
                <bindingRedirect oldVersion="0.0.0.0-4.1.0.0"
newVersion="4.1.0.0"/>
            </dependentAssembly>
        </assemblyBinding>
    </runtime>
</configuration>
```

Packages.config

```
<?xml version="1.0" encoding="utf-8"?>
<packages>
  <package id="Microsoft.Bcl" version="1.1.10" targetFramework="monoandroid71" />
  <package id="Microsoft.Bcl.Build" version="1.0.21"
targetFramework="monoandroid71" />
  <package id="Microsoft.Net.Http" version="2.2.29" targetFramework="monoandroid71"
/>
  <package id="Microsoft.NETCore.Platforms" version="1.0.1"
targetFramework="monoandroid71" />
  <package id="Microsoft.Win32.Primitives" version="4.0.1"
targetFramework="monoandroid71" />
  <package id="NETStandard.Library" version="1.6.0" targetFramework="monoandroid71"
/>
  <package id="Newtonsoft.Json" version="9.0.1" targetFramework="monoandroid71" />
  <package id="RestSharp" version="105.2.3" targetFramework="monoandroid71" />
  <package id="sqlite-net-pcl" version="1.3.1" targetFramework="monoandroid71" />
  <package id="SQLitePCLRaw.bundle_green" version="1.1.2"
targetFramework="monoandroid71" />
  <package id="SQLitePCLRaw.core" version="1.1.2" targetFramework="monoandroid71"
/>
```

```
  <package id="SQLitePCLRaw.lib.e_sqlite3.android" version="1.1.2"
targetFramework="monoandroid71" />
  <package id="SQLitePCLRaw.provider.e_sqlite3.android" version="1.1.2"
targetFramework="monoandroid71" />
  <package id="System.AppContext" version="4.1.0" targetFramework="monoandroid71"
/>
  <package id="System.Collections" version="4.0.11" targetFramework="monoandroid71"
/>
  <package id="System.Collections.Concurrent" version="4.0.12"
targetFramework="monoandroid71" />
  <package id="System.Console" version="4.0.0" targetFramework="monoandroid71" />
  <package id="System.Diagnostics.Debug" version="4.0.11"
targetFramework="monoandroid71" />
  <package id="System.Diagnostics.Tools" version="4.0.1"
targetFramework="monoandroid71" />
  <package id="System.Diagnostics.Tracing" version="4.1.0"
targetFramework="monoandroid71" />
  <package id="System.Globalization" version="4.0.11"
targetFramework="monoandroid71" />
  <package id="System.Globalization.Calendars" version="4.0.1"
targetFramework="monoandroid71" />
  <package id="System.IO" version="4.1.0" targetFramework="monoandroid71" />
  <package id="System.IO.Compression" version="4.1.0"
targetFramework="monoandroid71" />
  <package id="System.IO.Compression.ZipFile" version="4.0.1"
targetFramework="monoandroid71" />
  <package id="System.IO.FileSystem" version="4.0.1"
targetFramework="monoandroid71" />
  <package id="System.IO.FileSystem.Primitives" version="4.0.1"
targetFramework="monoandroid71" />
  <package id="System.Linq" version="4.1.0" targetFramework="monoandroid71" />
  <package id="System.Linq.Expressions" version="4.1.0"
targetFramework="monoandroid71" />
  <package id="System.Net.Http" version="4.1.0" targetFramework="monoandroid71" />
  <package id="System.Net.Primitives" version="4.0.11"
targetFramework="monoandroid71" />
  <package id="System.Net.Sockets" version="4.1.0" targetFramework="monoandroid71"
/>
  <package id="System.ObjectModel" version="4.0.12" targetFramework="monoandroid71"
/>
  <package id="System.Reflection" version="4.1.0" targetFramework="monoandroid71"
/>
  <package id="System.Reflection.Extensions" version="4.0.1"
targetFramework="monoandroid71" />
  <package id="System.Reflection.Primitives" version="4.0.1"
targetFramework="monoandroid71" />
  <package id="System.Resources.ResourceManager" version="4.0.1"
targetFramework="monoandroid71" />
  <package id="System.Runtime" version="4.1.0" targetFramework="monoandroid71" />
  <package id="System.Runtime.Extensions" version="4.1.0"
targetFramework="monoandroid71" />
```

```
  <package id="System.Runtime.Handles" version="4.0.1"
targetFramework="monoandroid71" />
  <package id="System.Runtime.InteropServices" version="4.1.0"
targetFramework="monoandroid71" />
  <package id="System.Runtime.InteropServices.RuntimeInformation" version="4.0.0"
targetFramework="monoandroid71" />
  <package id="System.Runtime.Numerics" version="4.0.1"
targetFramework="monoandroid71" />
  <package id="System.Security.Cryptography.Algorithms" version="4.2.0"
targetFramework="monoandroid71" />
  <package id="System.Security.Cryptography.Encoding" version="4.0.0"
targetFramework="monoandroid71" />
  <package id="System.Security.Cryptography.Primitives" version="4.0.0"
targetFramework="monoandroid71" />
  <package id="System.Security.Cryptography.X509Certificates" version="4.1.0"
targetFramework="monoandroid71" />
  <package id="System.Text.Encoding" version="4.0.11"
targetFramework="monoandroid71" />
  <package id="System.Text.Encoding.Extensions" version="4.0.11"
targetFramework="monoandroid71" />
  <package id="System.Text.RegularExpressions" version="4.1.0"
targetFramework="monoandroid71" />
  <package id="System.Threading" version="4.0.11" targetFramework="monoandroid71"
/>
  <package id="System.Threading.Tasks" version="4.0.11"
targetFramework="monoandroid71" />
  <package id="System.Threading.Timer" version="4.0.1"
targetFramework="monoandroid71" />
  <package id="System.Xml.ReaderWriter" version="4.0.11"
targetFramework="monoandroid71" />
  <package id="System.Xml.XDocument" version="4.0.11"
targetFramework="monoandroid71" />
</packages>
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="AshwinAhuja.TechSupport" android:versionName="3.1.0"
android:installLocation="auto" android:versionCode="3">
	<uses-sdk android:minSdkVersion="16" />

	<uses-permission android:name="android.permission.INTERNET" />
	<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
	<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
	<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
	<uses-permission android:name="android.permission.ACCESS_NOTIFICATION_POLICY"
/>
	<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
	<application android:label="Tech Support" android:icon="@drawable/Icon" />
	<application android:name="TechSupport" />
```

```
        <application android:label="TechSupport"
android:icon="@drawable/Icon"></application>
</manifest>
```

AssemblyInfo.cs

```
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;
using Android.App;

// General Information about an assembly is controlled through the following
// set of attributes. Change these attribute values to modify the information
// associated with an assembly.
[assembly: AssemblyTitle("Tech Support")]
[assembly: AssemblyDescription("")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("")]
[assembly: AssemblyProduct("Tech Support")]
[assembly: Application(Label = "Tech Support")]
[assembly: AssemblyCopyright("Copyright ©  2017")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCulture("")]
[assembly: ComVisible(false)]

// Version information for an assembly consists of the following four values:
//
//      Major Version
//      Minor Version
//      Build Number
//      Revision
//
// You can specify all the values or you can default the Build and Revision Numbers
// by using the '*' as shown below:
// [assembly: AssemblyVersion("1.0.*")]
[assembly: AssemblyVersion("1.0.0.0")]
[assembly: AssemblyFileVersion("1.0.0.0")]
```

Appendix C – SMS Server Code

App.py

```
#!flask/bin/python
from twilio.rest import TwilioRestClient
from flask import Flask, request
from twilio import twiml
import requests

account_sid = "AC501288d47afc284142feded345c61a0e" # My specific account ID from
twilio
auth_token  = "786c97bcb800f367d67e9a2bd882c58a"  # My specific authentication
token from Twilio

client = TwilioRestClient(account_sid, auth_token) # Setup object for Twilio API



app = Flask(__name__)

@app.route('/', methods = ['GET']) # Test function - useful to test if the API is
working (returns Hello World)
def helloWorld():
    return "Hello World!"

@app.route('/sms', methods = ['GET','POST']) #This is the function which the Twilio
API sends a message to if a message is received on the number
def index():
        number = request.form['From'] # Gets the number from which it is sent
        message_body = request.form['Body'] #Gets the body of the text
        response = twiml.Response()
        headers = {'content-type': 'application/json'}
        url = "http://localhost:5000/api/app/sms" + number + "," + message_body
        r = requests.post(url) # Sends a message to the WebAPI with the number and
contents of the message.
        response.message('Thanks for your request. Your appointment details will be
sent to you shortly') # Also responds to the message directly (in case the WebAPI
is broken or takes a while)
        return str(response)

@app.route('/sendSms', methods = ['POST']) #When called by the WebAPI
def sendMessage():
        data = request.json
        number = data['From'] # Parses the JSON and gets the number and
message_body of the text to be sent
        message_body = data['Body']
        message = client.messages.create(body=message_body,
            to=number,
            from_="+441202237638") #Defines the message
        return "SUCCESS" # Returns a string "SUCCESS" instead of a boolean as the
Flask API does not allow for returning booleans
```

```
@app.route('/verifyNumber', methods = ['POST']) # When called by the Android
Application
def verifyNumber():
    data = request.json # Parses the JSON object into number and friendlyName
    number = data['Number']
    friendlyName = data['friendlyName']
    caller_id = client.caller_ids.validate("+" + number, friendly_name =
friendlyName) # Sends request to the Twilio API and gets the validation code
    a = caller_id ['validation_code']
    return a # returns the validation code


if  __name__ == '__main__': # Entry point for application
    app.debug=True #Allows things to be printed to the screen
    app.run(port=5001) # Begins the listener
```

Appendix D – Test Data and Expected Results

Test Series 1

1.1

T1.1.1

SELECT * FROM accountsForElderly WHERE id = 1

E1.1.1

Check that correct field is displayed – looking at the SQL GUI application

```
SELECT * FROM accountsForElderly WHERE id = 1
+----+----------+-----------------+----------+-----------+---------+--------------------+---------------------+----------------+------------+------+---------+-------------+
| id | username | emailAddress    | password | firstname | surname | firstlineOfAddress | secondlineOfAddress | telephoneNumber | postalCode | city | country | timetableid |
+----+----------+-----------------+----------+-----------+---------+--------------------+---------------------+----------------+------------+------+---------+-------------+
| 1  | ahujaas  | ashwin@gmail.co | ash      | ahu       | aj      |                    | a                   | a              | W23LRP     | a    | a       | NULL        |
+----+----------+-----------------+----------+-----------+---------+--------------------+---------------------+----------------+------------+------+---------+-------------+
1 rows in set
```

T1.1.2

SELECT notAField FROM accountsForElderly WHERE id = 1

E1.1.2

```
SELECT notAField FROM accountsForElderly WHERE id = 1
Error: Query execution failed: Unknown column 'notAField' in 'field list'
```

T1.1.3

SELECT * FROM accountsForElderly WHERE id = 1008

E1.1.3

```
SELECT * FROM accountsForElderly WHERE id = 1008

Empty set
```

T1.1.4

UPDATE accountsForElderly SET username = "ahujaa" WHERE id = 1

E1.1.4

```
UPDATE accountsForElderly SET username = "ahujaa" WHERE id = 1
Query OK, 1 rows affected
```

1.2

T1.2.1

"27th April 2016 at 10:00"

E1.2.1

27/04/2016 10:00

T1.2.2

"tomorrow at 20:00"

E1.2.2

01/04/2017 20:00:00

T1.2.3

"27th April 2017 at an undefined time"

E1.2.3

27/04/2017 00:00:00

T1.2.4

"10:00"

E1.2.4

31/03/2017 10:00:00

T1.2.5

"3/30/2017 12:00"

E1.2.5

30/03/2017 12:00:00

1.3

T1.3.1

GET http://localhost:5000/api/ac

E1.3.1

"Hello World"

T1.3.2

GET http://localNOThost:5000/api/ac

E1.3.2

Could Not Get A Response

T1.3.3

GET http://localhost:5000/api/ac22

E1.3.3

404 Not Found

1.4

T1.4.1

GET http://localhost:5000/api/ac

E1.4.1

"Hello World"

T1.4.2

POST http://localhost:5000/api/ac/users/ad

Parameters

"{

""id"": 0,

""username"": ""testUsername"",

""firstName"": ""testFirstName"",

""surname"": ""testSurname"",

""password"": ""testPassword"",

""emailAddress"": ""testEmailAddress"",

""firstLineOfAddress"": ""testFirstLineOfAddress"",

""secondLineOfAddress"": ""testSecondLineOfAddress"",

""telephoneNumber"": ""testTelephoneNumber"",

""postalCode"": ""testPostcode"",

""city"": ""testCity"",

""country"": ""testCountry"",

""helper"": false,

""timetableId"": 1,

""distance"": 1.7976931348623157e+308

}"

E1.4.2

true

&

Check if new entry in elderly person database with entered properties

T1.4.3

POST http://localhost:5000/api/ac/users/vf

Parameters

"{

""password"": ""testPassword"",

""emailAddress"": ""testEmailAddress"",

""helper"": false

}"

E1.4.3

The details of the verified user should be returned in JSON, as below:

"{

""id"": 34,

""username"": ""testUsername"",

""firstName"": ""testFirstName"",

""surname"": ""testSurname"",

""password"": ""testPassword"",

""emailAddress"": ""testEmailAddress"",

""firstLineOfAddress"": ""testFirstLineOfAddress"",

""secondLineOfAddress"": ""testSecondLineOfAddress"",

""telephoneNumber"": ""testTelephoneNumber"",

""postalCode"": ""testPostcode"",

""city"": ""testCity"",

""country"": ""testCountry"",

""helper"": false,

""timetableId"": 0,

""distance"": 1.7976931348623157e+308

}"

1.5

T1.5.1

GET http://localhost:5000/api/ach

E1.5.1

"Hello World"

T1.5.2

GET http://localhost:5000/api/ach/users/gi9

E1.5.2

The information of the user should be returned in JSON, for example, like this:

"{

""id"": 9,

""username"": ""ashwinahuja"",

""firstName"": ""Helper"",

""surname"": ""Ahuja"",

""password"": ""3ea7121feb7564ddfe2d1f2ebb41706e"",

""emailAddress"": ""ashwin@gmail.com"",

""firstLineOfAddress"": ""10 Lyndhurst Gardens"",

""secondLineOfAddress"": ""Hampstead"",

""telephoneNumber"": ""447597711497"",

""postalCode"": ""SW138QY"",

""city"": ""London"",

""country"": ""United Kingdom"",

""helper"": true,

""timetableId"": 53,

""distance"": 1.7976931348623157e+308

}"

T1.5.3

GET http://localhost:5000/api/ach/users/gi1008

E1.5.3

500 Internal Server Error

T1.5.4

POST http://localhost:5000/api/ach/users/ad

Parameters

"{

""id"": 0,

""username"": ""testUsername"",

""firstName"": ""testFirstName"",

""surname"": ""testSurname"",

""password"": ""testPassword"",

""emailAddress"": ""testEmailAddress"",

""firstLineOfAddress"": ""testFirstLineOfAddress"",

""secondLineOfAddress"": ""testSecondLineOfAddress"",

""telephoneNumber"": ""testTelephoneNumber"",

""postalCode"": ""SW139JT"",

""city"": ""testCity"",

""country"": ""testCountry"",

""helper"": true,

""timetableId"": 1,

""distance"": 1.7976931348623157e+308

}"

E1.5.4

TRUE

&

Check that new entry exists in the database

T1.5.5

POST http://localhost:5000/api/ach/users/ad

Parameters

"{

""id"": 0,

""username"": ""testUsername"",

""firstName"": ""testFirstName"",

""surname"": ""testSurname"",

""password"": ""testPassword"",

""emailAddress"": ""testEmailAddress"",

""firstLineOfAddress"": ""testFirstLineOfAddress"",

""secondLineOfAddress"": ""testSecondLineOfAddress"",

""telephoneNumber"": ""testTelephoneNumber"",

""postalCode"": ""SW139JT"",

""city"": ""testCity"",

""country"": ""testCountry"",

""helper"": true,

""timetableId"": 1,

""distance"": 1.7976931348623157e+308

}"

E1.5.5
FALSE

&

Check there is no new entry in database

T1.5.6
DELETE http://localhost:5000/api/ach/users/rm18

E1.5.6
TRUE

&

Check the entry is removed in database

T1.5.7
DELETE http://localhost:5000/api/ach/users/rm19

E1.5.7
FALSE

T1.5.8
PUT http://localhost:5000/api/ach/users/up

Parameters

"{

""id"": 19,

""username"": ""testUsernameNEW"",

""firstName"": ""testFirstNameNEW"",

""surname"": ""testSurnameNEW"",

""password"": ""testPasswordNEW"",

""emailAddress"": ""testEmailAddressNEW"",

""firstLineOfAddress"": ""testFirstLineOfAddressNEW"",

""secondLineOfAddress"": ""testSecondLineOfAddressNEW"",

""telephoneNumber"": ""testTelephoneNumberNEW"",

""postalCode"": ""SW139JT"",

""city"": ""testCityNEW"",

""country"": ""testCountryNEW"",

""helper"": true,

""timetableId"": 1,

""distance"": 1.7976931348623157e+308

}"

E1.5.8

TRUE

&

Check entry is updated in database

T1.5.9

PUT http://localhost:5000/api/ach/users/up

Parameters

"{

""id"": 1008,

""username"": ""testUsernameNEW"",

""firstName"": ""testFirstNameNEW"",

""surname"": ""testSurnameNEW"",

""password"": ""testPasswordNEW"",

""emailAddress"": ""testEmailAddressDoesntExist"",

""firstLineOfAddress"": ""testFirstLineOfAddressNEW"",

""secondLineOfAddress"": ""testSecondLineOfAddressNEW"",

""telephoneNumber"": ""testTelephoneNumberNEW"",

""postalCode"": ""SW139JT"",

""city"": ""testCityNEW"",

""country"": ""testCountryNEW"",

""helper"": true,

""timetableId"": 1,

""distance"": 1.7976931348623157e+308

}"

E1.5.9

FALSE

T1.5.10

POST http://localhost:5000/api/ach/users/vf

Parameters

"{

""password"": ""testPasswordNEW"",

""emailAddress"": ""testEmailAddressWRONG"",

""helper"": true

}"

E1.5.10

An empty user should be returned in JSON.

"{

""id"": 0,

""username"": null,

""firstName"": null,

""surname"": null,

""password"": null,

""emailAddress"": null,

""firstLineOfAddress"": null,

""secondLineOfAddress"": null,

""telephoneNumber"": null,

""postalCode"": null,

""city"": null,

""country"": null,

""helper"": false,

""timetableId"": 0,

""distance"": 1.7976931348623157e+308

}"

T1.5.11

POST http://localhost:5000/api/ach/users/vf

Parameters

"{

""password"": ""testPasswordNEW"",

""emailAddress"": ""testEmailAddressNEW"",

""helper"": true

}"

E1.5.11

The information of the verified user should be returned in JSON, for example, like this:

"{

""id"": 19,

""username"": ""testUsernameNEW"",

""firstName"": ""testFirstNameNEW"",

""surname"": ""testSurnameNEW"",

""password"": ""testPasswordNEW"",

""emailAddress"": ""testEmailAddressNEW"",

""firstLineOfAddress"": ""testFirstLineOfAddressNEW"",

""secondLineOfAddress"": ""testSecondLineOfAddressNEW"",

""telephoneNumber"": ""testTelephoneNumberNEW"",

""postalCode"": ""SW139JT"",

""city"": ""testCityNEW"",

""country"": ""testCountryNEW"",

""helper"": true,

""timetableId"": 0,

""distance"": 1.7976931348623157e+308

}"

T1.5.12

GET http://localhost:5000/api/ach/gl28

E1.5.12

The list of users (with how near they are to KT109HZ) should be returned as a JSON array item.

"[

{

""id"": 10,

""username"": ""jdnsna"",

""firstName"": ""hshshs"",

""surname"": ""jsjsjs"",

""password"": ""47bce5c74f589f4867dbd57e9ca9f808"",

""emailAddress"": ""usnamms"",

""firstLineOfAddress"": ""ndnsjs"",

""secondLineOfAddress"": ""usjsjshsj"",

""telephoneNumber"": ""447597711496"",

""postalCode"": ""sw138qz"",

""city"": ""jajsjjd"",

""country"": ""nsnaja"",

""helper"": false,

""timetableId"": 8,

""distance"": 23.712

},
{
""id"": 9,
""username"": ""ashwinahuja"",
""firstName"": ""Helper"",
""surname"": ""Ahuja"",
""password"": ""3ea7121feb7564ddfe2d1f2ebb41706e"",
""emailAddress"": ""ashwin@gmail.com"",
""firstLineOfAddress"": ""10 Lyndhurst Gardens"",
""secondLineOfAddress"": ""Hampstead"",
""telephoneNumber"": ""447597711497"",
""postalCode"": ""SW138QY"",
""city"": ""London"",
""country"": ""United Kingdom"",
""helper"": false,
""timetableId"": 53,
""distance"": 23.828
},
{
""id"": 12,
""username"": ""ahuja"",
""firstName"": ""Ashwin"",
""surname"": ""Ahuja"",
""password"": ""3ea7121feb7564ddfe2d1f2ebb41706e"",
""emailAddress"": ""ashwinahuja@gmail.com"",
""firstLineOfAddress"": ""32 Rivers Road"",
""secondLineOfAddress"": """",

""telephoneNumber"": ""447597711493"",
""postalCode"": ""SW138QY"",
""city"": ""London"",
""country"": ""United Kingdom"",
""helper"": false,
""timetableId"": 39,
""distance"": 23.828
},
{
""id"": 14,
""username"": ""a"",
""firstName"": ""a"",
""surname"": ""a"",
""password"": ""0cc175b9c0f1b6a831c399e269772661"",
""emailAddress"": ""aab"",
""firstLineOfAddress"": ""nsn"",
""secondLineOfAddress"": ""jsns"",
""telephoneNumber"": ""447597711495"",
""postalCode"": ""SW138QY"",
""city"": ""ndnd"",
""country"": ""hdbd"",
""helper"": false,
""timetableId"": 51,
""distance"": 23.828
},
{
""id"": 19,

""username"": ""testUsernameNEW"",

""firstName"": ""testFirstNameNEW"",

""surname"": ""testSurnameNEW"",

""password"": ""testPasswordNEW"",

""emailAddress"": ""testEmailAddressNEW"",

""firstLineOfAddress"": ""testFirstLineOfAddressNEW"",

""secondLineOfAddress"": ""testSecondLineOfAddressNEW"",

""telephoneNumber"": ""testTelephoneNumberNEW"",

""postalCode"": ""SW139JT"",

""city"": ""testCityNEW"",

""country"": ""testCountryNEW"",

""helper"": false,

""timetableId"": 59,

""distance"": 24.26

},

{

""id"": 11,

""username"": ""vech"",

""firstName"": ""oli"",

""surname"": ""vech"",

""password"": ""bf2bc2545a4a5f5683d9ef3ed0d977e0"",

""emailAddress"": ""vechtest2@gmail.com"",

""firstLineOfAddress"": ""hshsjs"",

""secondLineOfAddress"": """",

""telephoneNumber"": ""447768147040"",

""postalCode"": ""SW100XD"",

""city"": ""London"",

""country"": ""United Kingdom"",
""helper"": false,
""timetableId"": 36,
""distance"": 25.322
},
{
""id"": 1,
""username"": ""ahujaas"",
""firstName"": ""Third"",
""surname"": ""Closest-Helper"",
""password"": ""ash"",
""emailAddress"": ""ashwin@gmail.co"",
""firstLineOfAddress"": ""a"",
""secondLineOfAddress"": ""a"",
""telephoneNumber"": ""a"",
""postalCode"": ""W1A 0AX"",
""city"": ""a"",
""country"": ""a"",
""helper"": false,
""timetableId"": 1,
""distance"": 34.661
},
{
""id"": 5,
""username"": """",
""firstName"": ""Closest"",
""surname"": ""Helper"",

""password"": ""ah7"",

""emailAddress"": ""testj"",

""firstLineOfAddress"": ""sjkdhs"",

""secondLineOfAddress"": ""sds"",

""telephoneNumber"": ""dsdsd"",

""postalCode"": ""NW35NR"",

""city"": ""sds"",

""country"": ""dsdsd"",

""helper"": false,

""timetableId"": 2,

""distance"": 35.877

},

{

""id"": 17,

""username"": ""Hi Ashwin"",

""firstName"": ""Philip"",

""surname"": ""Fernandes "",

""password"": ""4944b366079773bebd70485ca180aafc"",

""emailAddress"": ""philafern82@gmail.com "",

""firstLineOfAddress"": ""82 West Hill "",

""secondLineOfAddress"": ""Wales "",

""telephoneNumber"": ""447946505344"",

""postalCode"": ""HA99RR "",

""city"": ""London"",

""country"": ""UK"",

""helper"": false,

""timetableId"": 57,

""distance"": 64.213

}

]"

&

Check that the people are arranged in order of distance to: KT10 9HZ

T1.5.13

GET http://localhost:5000/api/ach/gl1008

E1.5.13

500 Internal Server Error

1.6

T1.6.1

GET http://localhost:5000/api/ace

E1.6.1

"Hello World"

T1.6.2

GET http://localhost:5000/api/ace/users/gi20

E1.6.2

The information of the user should be returned in JSON, for example, like this:

"{

""id"": 20,

""username"": ""ahujaas"",

""firstName"": ""Ashwin"",

""surname"": ""Ahuja"",

""password"": ""3ea7121feb7564ddfe2d1f2ebb41706e"",

""emailAddress"": ""ashwin.ahuja@gmail.com"",

""firstLineOfAddress"": ""32 Riverview Gardens"",

""secondLineOfAddress"": """",

""telephoneNumber"": ""447597711495"",

""postalCode"": ""SW138QY"",

""city"": ""London"",

""country"": ""United Kingdom"",

""helper"": false,

""timetableId"": 0,

""distance"": 1.7976931348623157e+308

}"

T1.6.3

GET http://localhost:5000/api/ace/users/gi1008

E1.6.3

HTTP 500 Server Error

T1.6.4

POST http://localhost:5000/api/ace/users/ad

Parameters

"{

""id"": 20,

""username"": ""ahujaas"",

""firstName"": ""Ashwin"",

""surname"": ""Ahuja"",

""password"": ""3ea7121feb7564ddfe2d1f2ebb41706e"",

""emailAddress"": ""NEW EMAIL ADDRESS"",

""firstLineOfAddress"": ""32 Riverview Gardens"",

""secondLineOfAddress"": """",

""telephoneNumber"": ""NEW TELEPHONE NUMBER"",

""postalCode"": ""SW138QY"",

""city"": ""London"",

""country"": ""United Kingdom"",

""helper"": false,

""timetableId"": 0,

""distance"": 1.7976931348623157e+308

}"

E1.6.4

TRUE

&

Check user added to the database.

T1.6.5

POST http://localhost:5000/api/ace/users/ad

Parameters

"{

""id"": 20,

""username"": ""ahujaas"",

""firstName"": ""Ashwin"",

""surname"": ""Ahuja"",

""password"": ""3ea7121feb7564ddfe2d1f2ebb41706e"",

""emailAddress"": ""ashwin.ahuja@gmail.com"",

""firstLineOfAddress"": ""32 Riverview Gardens"",

""secondLineOfAddress"": """",

""telephoneNumber"": ""447597711495"",

""postalCode"": ""SW138QY"",

""city"": ""London"",

""country"": ""United Kingdom"",

""helper"": false,

""timetableId"": 0,

""distance"": 1.7976931348623157e+308

}"

E1.6.5

FALSE

&

Check user not added

T1.6.6

DELETE http://localhost:5000/api/ace/users/rm20

E1.6.6

TRUE

&

Check that database entry has been deleted

T1.6.7

DELETE http://localhost:5000/api/ace/users/rm21

E1.6.7

FALSE

T1.6.8

PUT http://localhost:5000/api/ace/users/up

Parameters

"{

""id"": 20,

""username"": ""ahujaas"",

""firstName"": ""Ashwin"",

""surname"": ""Ahuja"",

""password"": ""3ea7121feb7564ddfe2d1f2ebb41706e"",

""emailAddress"": ""NEW EMAIL ADDRESS VERSION TWO"",

""firstLineOfAddress"": ""32 Riverview Gardens"",

""secondLineOfAddress"": """",

""telephoneNumber"": ""NEW TELEPHONE NUMBER VERSION TWO"",

""postalCode"": ""SW138QY"",

""city"": ""London"",

""country"": ""United Kingdom"",

""helper"": false,

""timetableId"": 0,

""distance"": 1.7976931348623157e+308

}"

E1.6.8

TRUE

&

Check that database entry has been updated

T1.6.9

PUT http://localhost:5000/api/ace/users/up

Parameters

"{

""id"": 1008,

""username"": ""ahujaas"",

""firstName"": ""Ashwin"",

""surname"": ""Ahuja"",

""password"": ""3ea7121feb7564ddfe2d1f2ebb41706e"",

""emailAddress"": ""NEW EMAIL ADDRESS VERSION TWO"",

""firstLineOfAddress"": ""32 Riverview Gardens"",

""secondLineOfAddress"": """",

""telephoneNumber"": ""NEW TELEPHONE NUMBER VERSION TWO"",

""postalCode"": ""SW138QY"",

""city"": ""London"",

""country"": ""United Kingdom"",

""helper"": false,

""timetableId"": 0,

""distance"": 1.7976931348623157e+308

}"

E1.6.9

FALSE

&

Check nothing updated in database

T1.6.10

POST http://localhost:5000/api/ach/users/vf

Parameters (incorrect user in JSON form)

"{

  ""password"": ""testPasswordNEW"",

  ""emailAddress"": ""NEW EMAIL ADDRESS VERSION TWO"",

  ""helper"": false

}"

E1.6.10

An empty user should be returned in JSON form.

"{

  ""id"": 0,

  ""username"": null,

  ""firstName"": null,

  ""surname"": null,

  ""password"": null,

  ""emailAddress"": null,

  ""firstLineOfAddress"": null,

  ""secondLineOfAddress"": null,

  ""telephoneNumber"": null,

  ""postalCode"": null,

  ""city"": null,

  ""country"": null,

  ""helper"": false,

  ""timetableId"": 0,

  ""distance"": 1.7976931348623157e+308

}"

T1.6.11

POST http://localhost:5000/api/ach/users/vf

Parameters (correct user in JSON form)

"{

  ""password"": ""3ea7121feb7564ddfe2d1f2ebb41706e"",

  ""emailAddress"": ""NEW EMAIL ADDRESS VERSION TWO"",

  ""helper"": false

}"

E1.6.11

The information of the verified user should be returned in JSON, for example, like this:

"{

  ""id"": 20,

  ""username"": ""ahujaas"",

  ""firstName"": ""Ashwin"",

  ""surname"": ""Ahuja"",

  ""password"": ""3ea7121feb7564ddfe2d1f2ebb41706e"",

  ""emailAddress"": ""NEW EMAIL ADDRESS VERSION TWO"",

  ""firstLineOfAddress"": ""32 Riverview Gardens"",

  ""secondLineOfAddress"": """",

  ""telephoneNumber"": ""NEW TELEPHONE NUMBER VERSION TWO"",

  ""postalCode"": ""SW138QY"",

  ""city"": ""London"",

  ""country"": ""United Kingdom"",

  ""helper"": false,

  ""timetableId"": 0,

  ""distance"": 1.7976931348623157e+308

}"

1.7

T1.7.1

POST http://localhost:5000/api/app/sms+447597711495,20th April 2017 at 10:00

&

Ensure that there is no helper which is free at 10am on 20th April 2017 (by going into the timetable of each and changing it to false at that time)

E1.7.1
FALSE

&

SMS received saying "Nobody could be found."

T1.7.2
POST http://localhost:5000/api/app/sms+447597711495,9th April 2017 at 00:00

E1.7.2
TRUE

&

SMS received saying "You have an appointment with … at 10:00 on 9th April 2017"

T1.7.3
POST http://localhost:5000/api/app/sms+44797711495,21st April 2017 at 00:00

&

Ensure that the number (+447597711495) is removed from the all elderly accounts.

E1.7.3
FALSE

&

No SMS received

T1.7.4
PUT http://localhost:5000/api/app/sendSMS

Parameters

"{

""Body"":""Hello Test Send"",

""From"":""+447597711495""

}"

E1.7.4
TRUE

&

SMS received

T1.7.5

GET http://localhost:5000/api/app/gi1008

E1.7.5

An empty appointment is returned in JSON form.

"{

  ""id"": 0,

  ""helperId"": null,

  ""elderlyId"": null,

  ""dateAndTime"": null,

  ""dateCreated"": null,

  ""ratingId"": 0,

  ""reviewId"": 0

}"

T1.7.6

GET http://localhost:5000/api/app/gi71

E1.7.6

An appointment (filled in) is returned in JSON form.

"{

  ""id"": 71,

  ""helperId"": 10,

  ""elderlyId"": 20,

  ""dateAndTime"": ""2017-03-15T02:00:00"",

  ""dateCreated"": ""2017-03-13T13:34:01"",

  ""ratingId"": 0,

  ""reviewId"": 0

}"

T1.7.7

DELETE http://localhost:5000/api/app/re18

E1.7.7

FALSE

T1.7.8

DELETE http://localhost:5000/api/app/re71

E1.7.8

TRUE

&

Appointment deleted on database

T1.7.9

GET http://localhost:5000/api/app/gae1008

E1.7.9

An empty list is returned in JSON form.

[]

T1.7.10

GET http://localhost:5000/api/app/gae20

E1.7.10

A JSON array containing all the appointments of user 20 will be returned.

"[

{

""id"": 63,

""helperId"": 9,

""elderlyId"": 20,

""dateAndTime"": ""2016-03-15T18:00:00"",

""dateCreated"": ""2017-03-13T07:22:49"",

""ratingId"": 8,

""reviewId"": 7

},

{

""id"": 64,

""helperId"": 9,

""elderlyId"": 20,

""dateAndTime"": ""2016-03-19T20:00:00"",

""dateCreated"": ""2017-03-13T07:22:59"",
""ratingId"": 7,
""reviewId"": 6
},
{
""id"": 66,
""helperId"": 9,
""elderlyId"": 20,
""dateAndTime"": ""2017-03-13T22:00:00"",
""dateCreated"": ""2017-03-13T10:08:15"",
""ratingId"": 0,
""reviewId"": 0
},
{
""id"": 67,
""helperId"": 9,
""elderlyId"": 20,
""dateAndTime"": ""2017-03-13T10:00:00"",
""dateCreated"": ""2017-03-13T16:10:22"",
""ratingId"": 0,
""reviewId"": 0
},
{
""id"": 68,
""helperId"": 9,
""elderlyId"": 20,
""dateAndTime"": ""2017-03-13T10:00:00"",

""dateCreated"": ""2017-03-13T16:52:25"",
""ratingId"": 0,
""reviewId"": 0
},
{
""id"": 69,
""helperId"": 9,
""elderlyId"": 20,
""dateAndTime"": ""2017-03-14T10:00:00"",
""dateCreated"": ""2017-03-13T16:55:49"",
""ratingId"": 0,
""reviewId"": 0
},
{
""id"": 72,
""helperId"": 1,
""elderlyId"": 20,
""dateAndTime"": ""2017-03-15T02:00:00"",
""dateCreated"": ""2017-03-13T13:34:01"",
""ratingId"": 0,
""reviewId"": 0
},
{
""id"": 73,
""helperId"": 9,
""elderlyId"": 20,
""dateAndTime"": ""2017-03-15T02:00:00"",

""dateCreated"": ""2017-03-13T13:34:01"",
""ratingId"": 0,
""reviewId"": 0
},
{
""id"": 74,
""helperId"": 9,
""elderlyId"": 20,
""dateAndTime"": ""2017-03-15T14:00:00"",
""dateCreated"": ""2017-03-13T13:35:33"",
""ratingId"": 12,
""reviewId"": 10
},
{
""id"": 75,
""helperId"": 10,
""elderlyId"": 20,
""dateAndTime"": ""2017-03-29T18:00:00"",
""dateCreated"": ""2017-03-14T10:12:37"",
""ratingId"": 0,
""reviewId"": 0
},
{
""id"": 79,
""helperId"": 11,
""elderlyId"": 20,
""dateAndTime"": ""2017-03-13T20:00:00"",

""dateCreated"": ""2017-03-14T10:53:01"",
""ratingId"": 0,
""reviewId"": 0
},
{
""id"": 80,
""helperId"": 11,
""elderlyId"": 20,
""dateAndTime"": ""2017-03-13T20:00:00"",
""dateCreated"": ""2017-03-14T10:53:27"",
""ratingId"": 0,
""reviewId"": 0
},
{
""id"": 81,
""helperId"": 11,
""elderlyId"": 20,
""dateAndTime"": ""2017-03-14T16:00:00"",
""dateCreated"": ""2017-03-14T10:53:54"",
""ratingId"": 0,
""reviewId"": 11
},
{
""id"": 85,
""helperId"": 9,
""elderlyId"": 20,
""dateAndTime"": ""2017-03-19T08:00:00"",

""dateCreated"": ""2017-03-17T10:18:17"",
""ratingId"": 0,
""reviewId"": 0
},
{
""id"": 102,
""helperId"": 9,
""elderlyId"": 20,
""dateAndTime"": ""2017-03-19T20:00:00"",
""dateCreated"": ""2017-03-19T08:38:42"",
""ratingId"": 0,
""reviewId"": 0
},
{
""id"": 103,
""helperId"": 9,
""elderlyId"": 20,
""dateAndTime"": ""2017-03-20T10:00:00"",
""dateCreated"": ""2017-03-19T12:10:48"",
""ratingId"": 0,
""reviewId"": 0
},
{
""id"": 105,
""helperId"": 9,
""elderlyId"": 20,
""dateAndTime"": ""2017-03-25T18:00:00"",

""dateCreated"": ""2017-03-19T13:54:31"",
""ratingId"": 0,
""reviewId"": 0
},
{
""id"": 106,
""helperId"": 9,
""elderlyId"": 20,
""dateAndTime"": ""2017-03-20T08:00:00"",
""dateCreated"": ""2017-03-19T13:57:02"",
""ratingId"": 0,
""reviewId"": 0
},
{
""id"": 107,
""helperId"": 9,
""elderlyId"": 20,
""dateAndTime"": ""2017-03-20T22:00:00"",
""dateCreated"": ""2017-03-20T06:27:18"",
""ratingId"": 0,
""reviewId"": 0
},
{
""id"": 108,
""helperId"": 9,
""elderlyId"": 20,
""dateAndTime"": ""2017-03-20T20:00:00"",

""dateCreated"": ""2017-03-20T06:27:49"",
""ratingId"": 0,
""reviewId"": 0
},
{
""id"": 112,
""helperId"": 9,
""elderlyId"": 20,
""dateAndTime"": ""2017-04-05T12:00:00"",
""dateCreated"": ""2017-04-03T12:09:31"",
""ratingId"": 0,
""reviewId"": 0
},
{
""id"": 114,
""helperId"": 17,
""elderlyId"": 20,
""dateAndTime"": ""2017-04-06T18:00:00"",
""dateCreated"": ""2017-04-05T15:37:04"",
""ratingId"": 0,
""reviewId"": 0
},
{
""id"": 115,
""helperId"": 17,
""elderlyId"": 20,
""dateAndTime"": ""2017-04-11T18:00:00"",

""dateCreated"": ""2017-04-05T15:37:19"",
""ratingId"": 0,
""reviewId"": 0
},
{
""id"": 116,
""helperId"": 17,
""elderlyId"": 20,
""dateAndTime"": ""1845-04-06T20:00:00"",
""dateCreated"": ""2017-04-05T15:37:41"",
""ratingId"": 15,
""reviewId"": 14
},
{
""id"": 117,
""helperId"": 17,
""elderlyId"": 20,
""dateAndTime"": ""2017-01-02T16:00:00"",
""dateCreated"": ""2017-04-05T15:39:17"",
""ratingId"": 0,
""reviewId"": 0
},
{
""id"": 118,
""helperId"": 17,
""elderlyId"": 20,
""dateAndTime"": ""2016-04-07T20:00:00"",

""dateCreated"": ""2017-04-05T15:40:08"",

""ratingId"": 0,

""reviewId"": 0

},

{

""id"": 119,

""helperId"": 17,

""elderlyId"": 20,

""dateAndTime"": ""2017-04-08T14:00:00"",

""dateCreated"": ""2017-04-05T15:44:15"",

""ratingId"": 0,

""reviewId"": 0

},

{

""id"": 120,

""helperId"": 9,

""elderlyId"": 20,

""dateAndTime"": ""2017-05-10T10:00:00"",

""dateCreated"": ""2017-04-06T15:43:20"",

""ratingId"": 0,

""reviewId"": 0

}

]"

T1.7.11

GET http://localhost:5000/api/app/gah1008

E1.7.11

An empty JSON array will be returned.

[]

T1.7.12

GET http://localhost:5000/api/app/gah9

E1.7.12

A JSON array containing all the appointments with the helper as id 9.

"[

{

""id"": 63,

""helperId"": 9,

""elderlyId"": 20,

""dateAndTime"": ""2016-03-15T18:00:00"",

""dateCreated"": ""2017-03-13T07:22:49"",

""ratingId"": 8,

""reviewId"": 7

},

{

""id"": 64,

""helperId"": 9,

""elderlyId"": 20,

""dateAndTime"": ""2016-03-19T20:00:00"",

""dateCreated"": ""2017-03-13T07:22:59"",

""ratingId"": 7,

""reviewId"": 6

},

{

""id"": 66,

""helperId"": 9,

""elderlyId"": 20,

""dateAndTime"": ""2017-03-13T22:00:00"",

""dateCreated"": ""2017-03-13T10:08:15"",
""ratingId"": 0,
""reviewId"": 0
},
{
""id"": 67,
""helperId"": 9,
""elderlyId"": 20,
""dateAndTime"": ""2017-03-13T10:00:00"",
""dateCreated"": ""2017-03-13T16:10:22"",
""ratingId"": 0,
""reviewId"": 0
},
{
""id"": 68,
""helperId"": 9,
""elderlyId"": 20,
""dateAndTime"": ""2017-03-13T10:00:00"",
""dateCreated"": ""2017-03-13T16:52:25"",
""ratingId"": 0,
""reviewId"": 0
},
{
""id"": 69,
""helperId"": 9,
""elderlyId"": 20,
""dateAndTime"": ""2017-03-14T10:00:00"",

""dateCreated"": ""2017-03-13T16:55:49"",
""ratingId"": 0,
""reviewId"": 0
},
{
""id"": 73,
""helperId"": 9,
""elderlyId"": 20,
""dateAndTime"": ""2017-03-15T02:00:00"",
""dateCreated"": ""2017-03-13T13:34:01"",
""ratingId"": 0,
""reviewId"": 0
},
{
""id"": 74,
""helperId"": 9,
""elderlyId"": 20,
""dateAndTime"": ""2017-03-15T14:00:00"",
""dateCreated"": ""2017-03-13T13:35:33"",
""ratingId"": 12,
""reviewId"": 10
},
{
""id"": 85,
""helperId"": 9,
""elderlyId"": 20,
""dateAndTime"": ""2017-03-19T08:00:00"",

""dateCreated"": ""2017-03-17T10:18:17"",
""ratingId"": 0,
""reviewId"": 0
},
{
""id"": 87,
""helperId"": 9,
""elderlyId"": 28,
""dateAndTime"": ""2017-03-18T18:00:00"",
""dateCreated"": ""2017-03-17T12:37:03"",
""ratingId"": 0,
""reviewId"": 0
},
{
""id"": 91,
""helperId"": 9,
""elderlyId"": 28,
""dateAndTime"": ""2017-03-30T08:00:00"",
""dateCreated"": ""2017-03-19T07:30:49"",
""ratingId"": 0,
""reviewId"": 0
},
{
""id"": 92,
""helperId"": 9,
""elderlyId"": 28,
""dateAndTime"": ""2017-04-06T04:00:00"",

""dateCreated"": ""2017-03-19T07:30:55"",
""ratingId"": 0,
""reviewId"": 0
},
{
""id"": 98,
""helperId"": 9,
""elderlyId"": 28,
""dateAndTime"": ""2017-03-19T04:00:00"",
""dateCreated"": ""2017-03-19T08:20:49"",
""ratingId"": 0,
""reviewId"": 0
},
{
""id"": 99,
""helperId"": 9,
""elderlyId"": 28,
""dateAndTime"": ""2017-03-30T12:00:00"",
""dateCreated"": ""2017-03-19T08:27:37"",
""ratingId"": 0,
""reviewId"": 0
},
{
""id"": 100,
""helperId"": 9,
""elderlyId"": 28,
""dateAndTime"": ""2017-03-19T18:00:00"",

""dateCreated"": ""2017-03-19T08:27:55"",
""ratingId"": 0,
""reviewId"": 0
},
{
""id"": 102,
""helperId"": 9,
""elderlyId"": 20,
""dateAndTime"": ""2017-03-19T20:00:00"",
""dateCreated"": ""2017-03-19T08:38:42"",
""ratingId"": 0,
""reviewId"": 0
},
{
""id"": 103,
""helperId"": 9,
""elderlyId"": 20,
""dateAndTime"": ""2017-03-20T10:00:00"",
""dateCreated"": ""2017-03-19T12:10:48"",
""ratingId"": 0,
""reviewId"": 0
},
{
""id"": 105,
""helperId"": 9,
""elderlyId"": 20,
""dateAndTime"": ""2017-03-25T18:00:00"",

""dateCreated"": ""2017-03-19T13:54:31"",
""ratingId"": 0,
""reviewId"": 0
},
{
""id"": 106,
""helperId"": 9,
""elderlyId"": 20,
""dateAndTime"": ""2017-03-20T08:00:00"",
""dateCreated"": ""2017-03-19T13:57:02"",
""ratingId"": 0,
""reviewId"": 0
},
{
""id"": 107,
""helperId"": 9,
""elderlyId"": 20,
""dateAndTime"": ""2017-03-20T22:00:00"",
""dateCreated"": ""2017-03-20T06:27:18"",
""ratingId"": 0,
""reviewId"": 0
},
{
""id"": 108,
""helperId"": 9,
""elderlyId"": 20,
""dateAndTime"": ""2017-03-20T20:00:00"",

""dateCreated"": ""2017-03-20T06:27:49"",
""ratingId"": 0,
""reviewId"": 0
},
{
""id"": 109,
""helperId"": 9,
""elderlyId"": 33,
""dateAndTime"": ""2017-03-25T20:00:00"",
""dateCreated"": ""2017-03-20T08:17:45"",
""ratingId"": 0,
""reviewId"": 0
},
{
""id"": 112,
""helperId"": 9,
""elderlyId"": 20,
""dateAndTime"": ""2017-04-05T12:00:00"",
""dateCreated"": ""2017-04-03T12:09:31"",
""ratingId"": 0,
""reviewId"": 0
},
{
""id"": 120,
""helperId"": 9,
""elderlyId"": 20,
""dateAndTime"": ""2017-05-10T10:00:00"",

""dateCreated"": ""2017-04-06T15:43:20"",

""ratingId"": 0,

""reviewId"": 0

}

]"

1.8

T1.8.1

GET http://localhost:5000/api/ratings/ga9

E1.8.1

A JSON array will be returned with all the ratings of helperId 9.

"[

{

""id"": 7,

""starRating"": 8,

""helperId"": 9

},

{

""id"": 8,

""starRating"": 6,

""helperId"": 9

},

{

""id"": 12,

""starRating"": 8,

""helperId"": 9

},

{

""id"": 17,

""starRating"": 10,

""helperId"": 9

}

]"

T1.8.2

GET http://localhost:5000/api/ratings/ga1008

E1.8.2

An empty JSON array will be returned.

[]

T1.8.3

GET http://localhost:5000/api/ratings/go8

E1.8.3

The rating will be returned in JSON form.

"{

""id"": 8,

""starRating"": 6,

""helperId"": 9

}"

T1.8.4

GET http://localhost:5000/api/ratings/go72

E1.8.4

HTTP 500 Server Error

T1.8.5

POST http://localhost:5000/api/ratings/ar

Parameters

"{

""id"": 63,

""starRating"": 10,

""helperId"": 9

}"

E1.8.5

TRUE

&

Check rating is added to the database.

T1.8.6
PATCH http://localhost:5000/api/ratings/ud

Parameters

"{

  ""id"": 63,

  ""starRating"": 10,

  ""helperId"": 9

}"

E1.8.6
TRUE

&

Check the item is updated on the database.

T1.8.7
DELETE http://localhost:5000/api/ratings/rm10

E1.8.7
TRUE

&

Check item is removed from the database

1.9

T1.9.1
GET http://localhost:5000/api/reviews/ga9

E1.9.1
All reviews of helper id 9 are returned as a JSON array.

"[

  {

    ""id"": 6,

    ""helperId"": 9,

    ""comment"": ""Annoying and seems incapable of listening""

},

{

""id"": 7,

""helperId"": 9,

""comment"": ""Perfect""

},

{

""id"": 10,

""helperId"": 9,

""comment"": ""ok""

}

]"

T1.9.2

GET http://localhost:5000/api/reviews/ga1008

E1.9.2

An empty JSON array will be returned.

[]

T1.9.3

GET http://localhost:5000/api/reviews/go8

E1.9.3

A review will be returned in JSON form.

"{

""id"": 8,

""helperId"": 10,

""comment"": ""Imperfect""

}"

T1.9.4

GET http://localhost:5000/api/reviews/go72

E1.9.4

HTTP 500 Server Error

T1.9.5

POST http://localhost:5000/api/reviews/ar

Parameters

"{

  ""id"": 83,

  ""helperId"": 10,

  ""comment"": ""Now entirely perfect""

}"

E1.9.5

TRUE

&

Check review added to the database

T1.9.6

PATCH http://localhost:5000/api/reviews/ud

Parameters

"{

  ""id"": 8,

  ""helperId"": 10,

  ""comment"": ""Now entirely perfect""

}"

E1.9.6

TRUE

&

Check review changed in database

T1.9.7

DELETE http://localhost:5000/api/ratings/rm15

E1.9.7

TRUE

&

Check review removed from the database

1.10

T1.10.1

POST http://localhost:5000/api/tt/ad

Parameters

"{

 ""weeks"": [

  {

   ""weekBeginning"": ""2017-03-13T00:00:00"",

   ""week"": {

    ""id"": 171,

    ""monday"": {

     ""timesFree"": [

      true,

      false,

      true,

      false,

      false,

      false,

      false,

      false,

      false,

      true,

      true,

      true

     ],

     ""id"": 691

    },

    ""tuesday"": {

""timesFree"": [
true,
false,
false,
false,
true,
false,
false,
true,
false,
false,
false,
false
],
""id"": 692
},
""wednesday"": {
""timesFree"": [
true,
true,
true,
true,
true,
true,
true,
true,
true,

true,
true,
true
],
""id"": 693
},
""thursday"": {
""timesFree"": [
true,
true,
true,
true,
true,
true,
true,
true,
true,
true,
true,
true
],
""id"": 694
},
""friday"": {
""timesFree"": [
false,
false,

false,
false,
false,
false,
false,
false,
false,
false,
false,
false
],
""id"": 695
},
""saturday"": {
""timesFree"": [
true,
true,
false,
false,
false,
false,
false,
false,
false,
true,
true,
true

```
      ],
      ""id"": 696
    },
    ""sunday"": {
      ""timesFree"": [
        false,
        false,
        false,
        false,
        true,
        true,
        true,
        false,
        false,
        true,
        true,
        false
      ],
      ""id"": 697
    }
  }
},
{
  ""weekBeginning"": ""2017-03-06T00:00:00"",
  ""week"": {
    ""id"": 164,
    ""monday"": {
```

""timesFree"": [
true,
false,
true,
false,
false,
false,
false,
false,
false,
true,
true,
true
],
""id"": 642
},
""tuesday"": {
""timesFree"": [
true,
true,
true,
false,
false,
false,
false,
false,
false,

```
    true,
    true,
    true
  ],
  ""id"": 643
},
""wednesday"": {
  ""timesFree"": [
    true,
    true,
    true,
    false,
    false,
    false,
    false,
    false,
    false,
    true,
    true,
    true
  ],
  ""id"": 644
},
""thursday"": {
  ""timesFree"": [
    true,
    true,
```

true,
false,
false,
false,
false,
false,
false,
true,
true,
true
],
""id"": 645
},
""friday"": {
""timesFree"": [
false,
false,
false,
false,
false,
false,
false,
false,
false,
false,
false,
false

```
  ],
  ""id"": 646
},
""saturday"": {
  ""timesFree"": [
    true,
    true,
    false,
    false,
    false,
    false,
    false,
    false,
    false,
    true,
    true,
    true
  ],
  ""id"": 647
},
""sunday"": {
  ""timesFree"": [
    false,
    false,
    false,
    false,
    true,
```

true,
true,
false,
false,
true,
true,
false
],
""id"": 648
}
}
},
{
""weekBeginning"": ""2017-03-27T00:00:00"",
""week"": {
""id"": 165,
""monday"": {
""timesFree"": [
true,
false,
true,
false,
false,
false,
false,
false,
false,

```
    true,
    true,
    true
  ],
  ""id"": 649
},
""tuesday"": {
  ""timesFree"": [
    true,
    true,
    true,
    false,
    false,
    false,
    false,
    false,
    false,
    true,
    true,
    true
  ],
  ""id"": 650
},
""wednesday"": {
  ""timesFree"": [
    true,
    true,
```

true,
false,
false,
false,
false,
false,
false,
true,
true,
true
],
""id"": 651
},
""thursday"": {
""timesFree"": [
true,
true,
true,
false,
false,
false,
false,
false,
false,
true,
true,
true

```
  ],
  ""id"": 652
},
""friday"": {
  ""timesFree"": [
    false,
    false,
    false,
    false,
    false,
    false,
    false,
    false,
    false,
    false,
    false,
    false
  ],
  ""id"": 653
},
""saturday"": {
  ""timesFree"": [
    true,
    true,
    false,
    false,
    false,
```

false,
false,
false,
false,
true,
true,
true
],
""id"": 654
},
""sunday"": {
""timesFree"": [
false,
false,
false,
false,
true,
true,
true,
false,
false,
true,
true,
false
],
""id"": 655
}

```
  }
},
{
  ""weekBeginning"": ""2017-04-10T00:00:00"",
  ""week"": {
    ""id"": 166,
    ""monday"": {
      ""timesFree"": [
        true,
        false,
        true,
        false,
        false,
        false,
        false,
        false,
        false,
        true,
        true,
        true
      ],
      ""id"": 656
    },
    ""tuesday"": {
      ""timesFree"": [
        true,
        true,
```

true,
false,
false,
false,
false,
false,
false,
true,
true,
true
],
""id"": 657
},
""wednesday"": {
""timesFree"": [
true,
true,
true,
false,
false,
false,
false,
false,
false,
true,
true,
true

```
  ],
  ""id"": 658
},
""thursday"": {
  ""timesFree"": [
    true,
    true,
    true,
    false,
    false,
    false,
    false,
    false,
    false,
    true,
    true,
    true
  ],
  ""id"": 659
},
""friday"": {
  ""timesFree"": [
    false,
    false,
    false,
    false,
    false,
```

false,
false,
false,
false,
false,
false,
false
],
""id"": 660
},
""saturday"": {
""timesFree"": [
true,
true,
false,
false,
false,
false,
false,
false,
false,
true,
true,
true
],
""id"": 661
},

```
    ""sunday"": {
      ""timesFree"": [
        false,
        false,
        false,
        false,
        true,
        true,
        true,
        false,
        false,
        true,
        true,
        false
      ],
      ""id"": 662
    }
  }
},
{
  ""weekBeginning"": ""2017-04-03T00:00:00"",
  ""week"": {
    ""id"": 167,
    ""monday"": {
      ""timesFree"": [
        true,
        false,
```

true,
false,
false,
false,
false,
false,
false,
true,
true,
true
],
""id"": 663
},
""tuesday"": {
""timesFree"": [
true,
true,
true,
true,
true,
true,
true,
true,
true,
true,
true,
true

```
  ],
  ""id"": 664
},
""wednesday"": {
  ""timesFree"": [
    true,
    true,
    true,
    false,
    false,
    false,
    false,
    false,
    false,
    true,
    true,
    true
  ],
  ""id"": 665
},
""thursday"": {
  ""timesFree"": [
    true,
    true,
    true,
    false,
    false,
```

false,
false,
false,
false,
true,
true,
true
],
""id"": 666
},
""friday"": {
""timesFree"": [
false,
false,
false,
false,
false,
false,
false,
false,
false,
false,
false,
false
],
""id"": 667
},

""saturday"": {
  ""timesFree"": [
    true,
    true,
    false,
    false,
    false,
    false,
    false,
    false,
    false,
    true,
    true,
    true
  ],
  ""id"": 668
},
""sunday"": {
  ""timesFree"": [
    false,
    false,
    false,
    false,
    true,
    true,
    true,
    false,

false,
true,
true,
false
],
""id"": 669
}
}
},
{
""weekBeginning"": ""2017-04-17T00:00:00"",
""week"": {
""id"": 168,
""monday"": {
""timesFree"": [
true,
false,
true,
false,
false,
false,
false,
false,
false,
true,
true,
true

```
  ],
  ""id"": 670
},
""tuesday"": {
  ""timesFree"": [
    true,
    true,
    true,
    false,
    false,
    false,
    false,
    false,
    false,
    true,
    true,
    true
  ],
  ""id"": 671
},
""wednesday"": {
  ""timesFree"": [
    true,
    true,
    true,
    false,
    false,
```

false,
false,
false,
false,
true,
true,
true
],
""id"": 672
},
""thursday"": {
""timesFree"": [
true,
true,
true,
false,
false,
false,
false,
false,
false,
true,
true,
true
],
""id"": 673
},

""friday"": {
  ""timesFree"": [
    false,
    false,
    false,
    false,
    false,
    false,
    false,
    false,
    false,
    false,
    false
  ],
  ""id"": 674
},
""saturday"": {
  ""timesFree"": [
    true,
    true,
    false,
    false,
    false,
    false,
    false,
    false,

false,
true,
true,
true
],
""id"": 675
},
""sunday"": {
""timesFree"": [
false,
false,
false,
false,
true,
true,
true,
false,
false,
true,
true,
false
],
""id"": 676
}
}
},
{

""weekBeginning"": ""2017-05-01T00:00:00"",
""week"": {
 ""id"": 169,
 ""monday"": {
  ""timesFree"": [
   true,
   false,
   true,
   false,
   false,
   false,
   false,
   false,
   false,
   true,
   true,
   true
  ],
  ""id"": 677
 },
 ""tuesday"": {
  ""timesFree"": [
   true,
   true,
   true,
   true,
   true,

true,
true,
true,
true,
true,
true,
true
],
""id"": 678
},
""wednesday"": {
""timesFree"": [
true,
true,
true,
false,
false,
false,
false,
false,
false,
true,
true,
true
],
""id"": 679
},

""thursday"": {
  ""timesFree"": [
    true,
    true,
    true,
    true,
    true,
    true,
    true,
    true,
    true,
    true,
    true,
    true
  ],
  ""id"": 680
},
""friday"": {
  ""timesFree"": [
    false,
    false,
    false,
    false,
    false,
    false,
    false,
    false,

false,
false,
false,
false
],
""id"": 681
},
""saturday"": {
""timesFree"": [
true,
true,
false,
false,
false,
false,
false,
false,
false,
true,
true,
true
],
""id"": 682
},
""sunday"": {
""timesFree"": [
false,

false,
false,
false,
true,
true,
true,
false,
false,
true,
true,
false
],
""id"": 683
}
}
},
{
""weekBeginning"": ""2017-03-20T00:00:00"",
""week"": {
""id"": 170,
""monday"": {
""timesFree"": [
true,
false,
true,
false,
false,

false,
false,
false,
false,
true,
true,
true
],
""id"": 684
},
""tuesday"": {
""timesFree"": [
false,
false,
false,
false,
false,
false,
false,
false,
false,
false,
false,
false
],
""id"": 685
},

""wednesday"": {
 ""timesFree"": [
  true,
  true,
  true,
  true,
  true,
  true,
  true,
  true,
  true,
  true,
  true,
  true
 ],
 ""id"": 686
},
""thursday"": {
 ""timesFree"": [
  true,
  true,
  true,
  true,
  true,
  true,
  true,
  true,

true,
true,
true,
true
],
""id"": 687
},
""friday"": {
""timesFree"": [
false,
false,
false,
false,
false,
false,
false,
false,
false,
false,
false,
false
],
""id"": 688
},
""saturday"": {
""timesFree"": [
true,

true,
false,
false,
false,
false,
false,
false,
false,
true,
true,
true
],
""id"": 689
},
""sunday"": {
""timesFree"": [
false,
false,
false,
false,
true,
true,
true,
false,
false,
true,
true,

false
],
""id"": 690
}
}
},
{
""weekBeginning"": ""2017-04-24T00:00:00"",
""week"": {
""id"": 406,
""monday"": {
""timesFree"": [
true,
false,
true,
false,
false,
false,
false,
false,
false,
true,
true,
true
],
""id"": 628
},

""tuesday"": {
  ""timesFree"": [
    true,
    false,
    false,
    false,
    true,
    false,
    false,
    true,
    false,
    false,
    false,
    false
  ],
  ""id"": 629
},
""wednesday"": {
  ""timesFree"": [
    true,
    true,
    true,
    true,
    true,
    true,
    true,
    true,

true,
true,
true,
true
],
""id"": 630
},
""thursday"": {
""timesFree"": [
true,
true,
true,
true,
true,
true,
true,
true,
true,
true,
true,
true
],
""id"": 631
},
""friday"": {
""timesFree"": [
false,

false,
false,
false,
false,
false,
false,
false,
false,
false,
false,
false
],
""id"": 632
},
""saturday"": {
""timesFree"": [
true,
true,
false,
false,
false,
false,
false,
false,
false,
true,
true,

true
],
""id"": 633
},
""sunday"": {
""timesFree"": [
false,
false,
false,
false,
true,
true,
true,
false,
false,
true,
true,
false
],
""id"": 634
}
}
},
{
""weekBeginning"": ""2017-05-08T00:00:00"",
""week"": {
""id"": 647,

""monday"": {
  ""timesFree"": [
    true,
    false,
    true,
    false,
    false,
    false,
    false,
    false,
    false,
    true,
    true,
    true
  ],
  ""id"": 628
},
""tuesday"": {
  ""timesFree"": [
    true,
    false,
    false,
    false,
    true,
    false,
    false,
    true,

false,
false,
false,
false
],
""id"": 629
},
""wednesday"": {
""timesFree"": [
true,
true,
true,
true,
true,
true,
true,
true,
true,
true,
true,
true
],
""id"": 630
},
""thursday"": {
""timesFree"": [
true,

true,
true,
true,
true,
true,
true,
true,
true,
true,
true,
true
],
""id"": 631
},
""friday"": {
""timesFree"": [
false,
false,
false,
false,
false,
false,
false,
false,
false,
false,
false,

false
],
""id"": 632
},
""saturday"": {
""timesFree"": [
true,
true,
false,
false,
false,
false,
false,
false,
false,
true,
true,
true
],
""id"": 633
},
""sunday"": {
""timesFree"": [
false,
false,
false,
false,

true,
true,
true,
false,
false,
true,
true,
false
],
""id"": 634
}
}
}
],
""id"": 16,
""defaultWeek"": {
""id"": 162,
""monday"": {
""timesFree"": [
true,
false,
true,
false,
false,
false,
false,
false,

false,
true,
true,
true
],
""id"": 628
},
""tuesday"": {
""timesFree"": [
true,
false,
false,
false,
true,
false,
false,
true,
false,
false,
false,
false
],
""id"": 629
},
""wednesday"": {
""timesFree"": [
true,

true,
true,
true,
true,
true,
true,
true,
true,
true,
true,
true
],
""id"": 630
},
""thursday"": {
""timesFree"": [
true,
true,
true,
true,
true,
true,
true,
true,
true,
true,
true,

true
],
""id"": 631
},
""friday"": {
""timesFree"": [
false,
false,
false,
false,
false,
false,
false,
false,
false,
false,
false,
false
],
""id"": 632
},
""saturday"": {
""timesFree"": [
true,
true,
false,
false,

false,
false,
false,
false,
false,
true,
true,
true
],
""id"": 633
},
""sunday"": {
""timesFree"": [
false,
false,
false,
false,
true,
true,
true,
false,
false,
true,
true,
false
],
""id"": 634

```
  }
 }
}"
```


E1.10.1

Check timetable added to the database

T1.10.2

GET http://localhost:5000/api/tt/gt16

E1.10.2

A timetable will be returned in JSON form – this is hard to confirm as being correct without deserialising using C# NewtonSoft.


```
"{
 ""weeks"": [
  {
   ""weekBeginning"": ""2017-03-13T00:00:00"",
   ""week"": {
    ""id"": 171,
    ""monday"": {
     ""timesFree"": [
      true,
      false,
      true,
      false,
      false,
      false,
      false,
      false,
      false,
      true,
      true,
```

true
],
""id"": 691
},
""tuesday"": {
""timesFree"": [
true,
false,
false,
false,
true,
false,
false,
true,
false,
false,
false,
false
],
""id"": 692
},
""wednesday"": {
""timesFree"": [
true,
true,
true,
true,

true,
true,
true,
true,
true,
true,
true,
true
],
""id"": 693
},
""thursday"": {
""timesFree"": [
true,
true,
true,
true,
true,
true,
true,
true,
true,
true,
true,
true
],
""id"": 694

},
""friday"": {
""timesFree"": [
false,
false,
false,
false,
false,
false,
false,
false,
false,
false,
false,
false
],
""id"": 695
},
""saturday"": {
""timesFree"": [
true,
true,
false,
false,
false,
false,
false,

false,
false,
true,
true,
true
],
""id"": 696
},
""sunday"": {
""timesFree"": [
false,
false,
false,
false,
true,
true,
true,
false,
false,
true,
true,
false
],
""id"": 697
}
}
},

{
""weekBeginning"": ""2017-03-06T00:00:00"",
""week"": {
""id"": 164,
""monday"": {
""timesFree"": [
true,
false,
true,
false,
false,
false,
false,
false,
false,
true,
true,
true
],
""id"": 642
},
""tuesday"": {
""timesFree"": [
true,
true,
true,
false,

false,
false,
false,
false,
false,
true,
true,
true
],
""id"": 643
},
""wednesday"": {
""timesFree"": [
true,
true,
true,
false,
false,
false,
false,
false,
false,
true,
true,
true
],
""id"": 644

},
""thursday"": {
""timesFree"": [
true,
true,
true,
false,
false,
false,
false,
false,
false,
true,
true,
true
],
""id"": 645
},
""friday"": {
""timesFree"": [
false,
false,
false,
false,
false,
false,
false,

false,
false,
false,
false,
false
],
""id"": 646
},
""saturday"": {
""timesFree"": [
true,
true,
false,
false,
false,
false,
false,
false,
false,
true,
true,
true
],
""id"": 647
},
""sunday"": {
""timesFree"": [

false,
false,
false,
false,
true,
true,
true,
false,
false,
true,
true,
false
],
""id"": 648
}
}
},
{
""weekBeginning"": ""2017-03-27T00:00:00"",
""week"": {
""id"": 165,
""monday"": {
""timesFree"": [
true,
false,
true,
false,

false,
false,
false,
false,
false,
true,
true,
true
],
""id"": 649
},
""tuesday"": {
""timesFree"": [
true,
true,
true,
false,
false,
false,
false,
false,
false,
true,
true,
true
],
""id"": 650

},
""wednesday"": {
""timesFree"": [
true,
true,
true,
false,
false,
false,
false,
false,
false,
true,
true,
true
],
""id"": 651
},
""thursday"": {
""timesFree"": [
true,
true,
true,
false,
false,
false,
false,

false,
false,
true,
true,
true
],
""id"": 652
},
""friday"": {
""timesFree"": [
false,
false,
false,
false,
false,
false,
false,
false,
false,
false,
false,
false
],
""id"": 653
},
""saturday"": {
""timesFree"": [

true,
true,
false,
false,
false,
false,
false,
false,
false,
true,
true,
true
],
""id"": 654
},
""sunday"": {
""timesFree"": [
false,
false,
false,
false,
true,
true,
true,
false,
false,
true,

true,
false
],
""id"": 655
}
}
},
{
""weekBeginning"": ""2017-04-10T00:00:00"",
""week"": {
""id"": 166,
""monday"": {
""timesFree"": [
true,
false,
true,
false,
false,
false,
false,
false,
false,
true,
true,
true
],
""id"": 656

},
""tuesday"": {
""timesFree"": [
true,
true,
true,
false,
false,
false,
false,
false,
false,
true,
true,
true
],
""id"": 657
},
""wednesday"": {
""timesFree"": [
true,
true,
true,
false,
false,
false,
false,

false,
false,
true,
true,
true
],
""id"": 658
},
""thursday"": {
""timesFree"": [
true,
true,
true,
false,
false,
false,
false,
false,
false,
true,
true,
true
],
""id"": 659
},
""friday"": {
""timesFree"": [

false,
false,
false,
false,
false,
false,
false,
false,
false,
false,
false,
false
],
""id"": 660
},
""saturday"": {
""timesFree"": [
true,
true,
false,
false,
false,
false,
false,
false,
false,
true,

true,
true
],
""id"": 661
},
""sunday"": {
""timesFree"": [
false,
false,
false,
false,
true,
true,
true,
false,
false,
true,
true,
false
],
""id"": 662
}
}
},
{
""weekBeginning"": ""2017-04-03T00:00:00"",
""week"": {

""id"": 167,
""monday"": {
  ""timesFree"": [
    true,
    false,
    true,
    false,
    false,
    false,
    false,
    false,
    false,
    true,
    true,
    true
  ],
  ""id"": 663
},
""tuesday"": {
  ""timesFree"": [
    true,
    true,
    true,
    true,
    true,
    true,
    true,

true,
true,
true,
true,
true
],
""id"": 664
},
""wednesday"": {
""timesFree"": [
true,
true,
true,
false,
false,
false,
false,
false,
false,
true,
true,
true
],
""id"": 665
},
""thursday"": {
""timesFree"": [

true,
true,
true,
false,
false,
false,
false,
false,
false,
true,
true,
true
],
""id"": 666
},
""friday"": {
""timesFree"": [
false,
false,
false,
false,
false,
false,
false,
false,
false,
false,

false,
false
],
""id"": 667
},
""saturday"": {
""timesFree"": [
true,
true,
false,
false,
false,
false,
false,
false,
false,
true,
true,
true
],
""id"": 668
},
""sunday"": {
""timesFree"": [
false,
false,
false,

false,
true,
true,
true,
false,
false,
true,
true,
false
],
""id"": 669
}
}
},
{
""weekBeginning"": ""2017-04-17T00:00:00"",
""week"": {
""id"": 168,
""monday"": {
""timesFree"": [
true,
false,
true,
false,
false,
false,
false,

false,
false,
true,
true,
true
],
""id"": 670
},
""tuesday"": {
""timesFree"": [
true,
true,
true,
false,
false,
false,
false,
false,
false,
true,
true,
true
],
""id"": 671
},
""wednesday"": {
""timesFree"": [

true,
true,
true,
false,
false,
false,
false,
false,
false,
true,
true,
true
],
""id"": 672
},
""thursday"": {
""timesFree"": [
true,
true,
true,
false,
false,
false,
false,
false,
false,
true,

true,
true
],
""id"": 673
},
""friday"": {
""timesFree"": [
false,
false,
false,
false,
false,
false,
false,
false,
false,
false,
false,
false
],
""id"": 674
},
""saturday"": {
""timesFree"": [
true,
true,
false,

false,
false,
false,
false,
false,
false,
true,
true,
true
],
""id"": 675
},
""sunday"": {
""timesFree"": [
false,
false,
false,
false,
true,
true,
true,
false,
false,
true,
true,
false
],

```
      ""id"": 676
    }
  }
},
{
  ""weekBeginning"": ""2017-05-01T00:00:00"",
  ""week"": {
    ""id"": 169,
    ""monday"": {
      ""timesFree"": [
        true,
        false,
        true,
        false,
        false,
        false,
        false,
        false,
        false,
        true,
        true,
        true
      ],
      ""id"": 677
    },
    ""tuesday"": {
      ""timesFree"": [
```

true,
true,
true,
true,
true,
true,
true,
true,
true,
true,
true,
true
],
""id"": 678
},
""wednesday"": {
""timesFree"": [
true,
true,
true,
false,
false,
false,
false,
false,
false,
true,

true,
true
],
""id"": 679
},
""thursday"": {
""timesFree"": [
true,
true,
true,
true,
true,
true,
true,
true,
true,
true,
true,
true
],
""id"": 680
},
""friday"": {
""timesFree"": [
false,
false,
false,

false,
false,
false,
false,
false,
false,
false,
false,
false
],
""id"": 681
},
""saturday"": {
""timesFree"": [
true,
true,
false,
false,
false,
false,
false,
false,
false,
false,
true,
true,
true
],

```
      ""id"": 682
    },
    ""sunday"": {
      ""timesFree"": [
        false,
        false,
        false,
        false,
        true,
        true,
        true,
        false,
        false,
        true,
        true,
        false
      ],
      ""id"": 683
    }
  }
},
{
  ""weekBeginning"": ""2017-03-20T00:00:00"",
  ""week"": {
    ""id"": 170,
    ""monday"": {
      ""timesFree"": [
```

true,
false,
true,
false,
false,
false,
false,
false,
false,
true,
true,
true
],
""id"": 684
},
""tuesday"": {
""timesFree"": [
false,
false,
false,
false,
false,
false,
false,
false,
false,
false,

false,
false
],
""id"": 685
},
""wednesday"": {
""timesFree"": [
true,
true,
true,
true,
true,
true,
true,
true,
true,
true,
true,
true
],
""id"": 686
},
""thursday"": {
""timesFree"": [
true,
true,
true,

true,
true,
true,
true,
true,
true,
true,
true,
true
],
""id"": 687
},
""friday"": {
""timesFree"": [
false,
false,
false,
false,
false,
false,
false,
false,
false,
false,
false,
false
],

""id"": 688
},
""saturday"": {
""timesFree"": [
true,
true,
false,
false,
false,
false,
false,
false,
false,
true,
true,
true
],
""id"": 689
},
""sunday"": {
""timesFree"": [
false,
false,
false,
false,
true,
true,

true,
false,
false,
true,
true,
false
],
""id"": 690
}
}
},
{
""weekBeginning"": ""2017-04-24T00:00:00"",
""week"": {
""id"": 406,
""monday"": {
""timesFree"": [
true,
false,
true,
false,
false,
false,
false,
false,
false,
true,

true,
true
],
""id"": 628
},
""tuesday"": {
""timesFree"": [
true,
false,
false,
false,
true,
false,
false,
true,
false,
false,
false,
false
],
""id"": 629
},
""wednesday"": {
""timesFree"": [
true,
true,
true,

true,
true,
true,
true,
true,
true,
true,
true,
true
],
""id"": 630
},
""thursday"": {
""timesFree"": [
true,
true,
true,
true,
true,
true,
true,
true,
true,
true,
true,
true
],

""id"": 631
},
""friday"": {
""timesFree"": [
false,
false,
false,
false,
false,
false,
false,
false,
false,
false,
false,
false,
false
],
""id"": 632
},
""saturday"": {
""timesFree"": [
true,
true,
false,
false,
false,
false,

false,
false,
false,
true,
true,
true
],
""id"": 633
},
""sunday"": {
""timesFree"": [
false,
false,
false,
false,
true,
true,
true,
false,
false,
true,
true,
false
],
""id"": 634
}
}

},
{
""weekBeginning"": ""2017-05-08T00:00:00"",
""week"": {
""id"": 647,
""monday"": {
""timesFree"": [
true,
false,
true,
false,
false,
false,
false,
false,
false,
true,
true,
true
],
""id"": 628
},
""tuesday"": {
""timesFree"": [
true,
false,
false,

false,
true,
false,
false,
true,
false,
false,
false,
false
],
""id"": 629
},
""wednesday"": {
""timesFree"": [
true,
true,
true,
true,
true,
true,
true,
true,
true,
true,
true,
true
],

""id"": 630
},
""thursday"": {
""timesFree"": [
true,
true,
true,
true,
true,
true,
true,
true,
true,
true,
true,
true
],
""id"": 631
},
""friday"": {
""timesFree"": [
false,
false,
false,
false,
false,
false,

false,
false,
false,
false,
false,
false
],
""id"": 632
},
""saturday"": {
""timesFree"": [
true,
true,
false,
false,
false,
false,
false,
false,
false,
true,
true,
true
],
""id"": 633
},
""sunday"": {

""timesFree"": [
false,
false,
false,
false,
true,
true,
true,
false,
false,
true,
true,
false
],
""id"": 634
}
}
}
],
""id"": 16,
""defaultWeek"": {
""id"": 162,
""monday"": {
""timesFree"": [
true,
false,
true,

false,
false,
false,
false,
false,
false,
true,
true,
true
],
""id"": 628
},
""tuesday"": {
""timesFree"": [
true,
false,
false,
false,
true,
false,
false,
true,
false,
false,
false,
false
],

""id"": 629
},
""wednesday"": {
""timesFree"": [
true,
true,
true,
true,
true,
true,
true,
true,
true,
true,
true,
true
],
""id"": 630
},
""thursday"": {
""timesFree"": [
true,
true,
true,
true,
true,
true,

true,
true,
true,
true,
true,
true
],
""id"": 631
},
""friday"": {
""timesFree"": [
false,
false,
false,
false,
false,
false,
false,
false,
false,
false,
false,
false
],
""id"": 632
},
""saturday"": {

""timesFree"": [
true,
true,
false,
false,
false,
false,
false,
false,
false,
true,
true,
true
],
""id"": 633
},
""sunday"": {
""timesFree"": [
false,
false,
false,
false,
true,
true,
true,
false,
false,

true,

true,

false

],

""id"": 634

}

}

}"

T1.10.3

GET http://localhost:5000/api/tt/gt64

E1.10.3

HTTP 500 Server Error

T1.10.4

GET http://localhost:5000/api/tt/gw10

E1.10.4

A week will be returned in JSON form.

"{

""id"": 10,

""monday"": {

""timesFree"": [

true,

false,

true,

false,

false,

false,

false,

false,

false,

true,
true,
true
],
""id"": 12
},
""tuesday"": {
""timesFree"": [
false,
false,
false,
false,
false,
false,
false,
false,
true,
true,
true,
false
],
""id"": 1
},
""wednesday"": {
""timesFree"": [
true,
true,

true,
false,
false,
false,
false,
false,
false,
true,
true,
true
],
""id"": 15
},
""thursday"": {
""timesFree"": [
false,
false,
false,
false,
false,
false,
false,
false,
true,
true,
true,
false

],

""id"": 1

},

""friday"": {

""timesFree"": [

false,

false,

false,

false,

false,

false,

false,

false,

false,

false,

false,

false

],

""id"": 9

},

""saturday"": {

""timesFree"": [

true,

true,

false,

false,

false,

false,
false,
false,
false,
true,
true,
true
],
""id"": 7
},
""sunday"": {
""timesFree"": [
false,
false,
false,
false,
true,
true,
true,
false,
false,
true,
true,
false
],
""id"": 6
}

}"

T1.10.5

GET http://localhost:5000/api/tt/gw1008

E1.10.5

HTTP 500 Server Error

T1.10.6

GET http://localhost:5000/api/tt/gd10

E1.10.6

A day will be returned in JSON form.

"{

""timesFree"": [

true,

false,

false,

false,

false,

false,

false,

false,

false,

true,

true,

true

],

""id"": 10

}"

T1.10.7

GET http://localhost:5000/api/tt/gd10008

E1.10.7

HTTP 500 Server Error

Test Series 2

2.1

T2.1.1

GET http://178.62.87.28:700/

E2.1.1

"Hello World"

2.2

T2.2.1

SMS to twilio phone number from verified number

Parameters

"Hello World Test From Validated Number"

E2.2.1

HTTP GET request from the SMS server at “http://localhost:5000/api/app/sms+447597711495,Hello World Test From Validated Number”

T2.2.2

SMS to twilio phone number from number not verified on twilio

Parameters

"Hello World Test From Not Validated Number"

E2.2.2

No result

2.3

T2.3.1

POST http://178.62.87.28:700/sendSms

Parameters

"{

""From"": ""447597711495"",

""Body"": ""Hello World SMS Receipt""

}"

E2.3.1

"SUCCESS"

&

SMS received by phone.

T2.3.2

POST http://178.62.87.28:700/sendSms

Parameters

"{

""From"": ""447824461022"",

""Body"": ""Hello World SMS Receipt""

}"

E2.3.2

"SUCCESS"

&

SMS not received

T2.3.3

POST http://178.62.87.28:700/sendSms

Parameters

"{

""From"": ""447597711495"",

""Body"": """"

}"

E2.3.3

HTTP 400 Error: Message Body Required

T2.3.4

POST http://178.62.87.28:700/sendSms

Parameters

"{

""From"": ""07597711495"",

""Body"": ""Hello World SMS Receipt""

}"

E2.3.4

"SUCCESS"

&

SMS received

T2.3.5

POST http://178.62.87.28:700/sendSms

Parameters

"{

""From"": ""011495"",

""Body"": ""Hello World SMS Receipt""

}"

E2.3.5

HTTP 400 error: The 'To' number 011495 is not a valid phone number.

&

SMS not received

T2.3.6

POST http://178.62.87.28:700/sendSms

Parameters

"{

""From"": ""011495"",

}"

E2.3.6

HTTP 400 Error: Bad Request

&

SMS not received

2.4

T2.4.1

POST http://178.62.87.28:700/verifyNumber

Parameters

"{

""Number"": ""447597711495"",

""friendlyName"":""Ashwin Ahuja""

}"

E2.4.1

789608

&

Subsequently able to verify number with code (responding to Twilio phone call) and check on Twilio Console (online)

T2.4.2

POST http://178.62.87.28:700/verifyNumber

Parameters

"{

""Number"": ""447495"",

""friendlyName"":""Ashwin Ahuja""

}

"

E2.4.2

HTTP 400 error: The phone number you are attempting to call, 447495, is not valid.

&

No verification of number

Test Series 3

3.1

T3.1.1

Starting Screen:

Action: Click the “login” button.

E3.1.1

The login screen (as below) should appear. If there is already a user which has been signed in on this phone, then the phone may login automatically.

T3.1.2

Starting Screen:

Action: Click the “Register” button.

E3.1.2

3.2

T3.2.1

Starting Screen:

Action: Press “back” button

E3.2.1

T3.2.2

Starting Screen:

Action: Only email address added with no password entered. Then, the “Login as Helper” button is pressed.

E3.2.2

3.3

T3.3.1

Starting State:

Action: Information entered for all items apart from Telephone Number and Register as Helper button is pressed.

E3.3.1

T3.3.2

Starting State:

Action: All information is filled in apart from postal code and then the Register as Helper button is pressed.

E3.3.2

T3.3.3

Starting State:

Action: All information is filled in apart from country and the Register as Helper button is pressed.

E3.3.3

T3.3.4

Starting State:

Action: All information is filled in apart from city and the Register as Helper button is pressed.

E3.3.4

T3.3.5

Starting State:

Actions: All information is filled in apart from first line of address and the register as helper button is pressed.

E3.3.5

T3.3.6

Starting State:

Actions: All the information is filled in apart from the username field and the register as helper (and register as elderly) button is pressed.

E3.3.6

T3.3.7

Starting State:

Actions: All the information is filled in apart from the email address and the register as helper and register as elderly button is pressed.

E3.3.7

T3.3.8

Starting State:

Actions: All information is filled in except the password is entered differently in the password and confirm password fields.

E3.3.8

T3.3.9

Starting State:

Actions: All information is entered except the first name. Then, the register as helper and register as elderly button is pressed.

E3.3.9

T3.3.11

Start State:

Actions: All information is entered correctly, apart from the password and confirm password fields having a different value. Then, the register as helper button (and register as elderly button in the second part of the test) is pressed.

E3.3.11

T3.3.12

Start State:

Actions: The back button (on the phone) is pressed.

E3.3.12

3.4

T3.4.1

Start State: List of appointments screen.

Actions: One of the appointments is pressed and it is checked that the appointments details screen is opened.

E3.4.1

T3.4.2

Starting State: List of appointments screen.

Actions: Back button (on the phone) is pressed.

E3.4.2

3.5

T3.5.1

Start State: Appointment Details screen.

Actions: The back button is pressed on the phone.

E3.5.1

T3.5.2

Start State: Appointment Details Screen.

Actions: Click the Set Ratings Button (and check that the ratings screen is opened).

E3.5.2

T3.5.3

Start State: Appointment Details Screen.

Actions: Set Review button is pressed, and it is checked whether the ratings screen is opened.

41% 03:35
Appointment Date: March 29, 2017
Appointment Time: 18:00
Name: hshshs jsjsjs
Contact Number: 447597711496
Address of Appointment:
32 Riverview Gardens
London
SW138QY
SET RATING
SET REVIEW

E3.5.3

41% 03:35
Name of Helper
hshshs jsjsjs
Set Review
SUBMIT

3.6

T3.6.1

Start State: Ratings screen.

Actions: The back button is pressed and it is checked that you go back to the details about the appointment.

E3.6.1

3.7

T3.7.1

Start State: Set review screen.

Actions: Press back button.

E3.7.1

3.8

T3.8.1

Start State: The helper details screen.

Actions: Click the 'See Reviews' button and check that the list of reviews screen appears.

E3.8.1

3.9

T3.9.1

Start State: See reviews screen.

Actions: The back button is pressed and it is checked that you get to the helper details screen.

E3.9.1

3.10

T3.10.1

Start State: The weeks screen of the book appointment screen.

Actions: Press back button.

E3.10.1

T3.10.2

Start State: The weeks screen of the book appointment screen.

Actions: Click on one of the weeks – in this case 'April 10th, 2017' and check the days screen of this week appears.

E3.10.2

3.11

T3.11.1

Start State: The days screen of booking an appointment.

Actions: Press back button of the phone and check that the weeks screen is opened.

28% 02:33
April 10, 2017
April 11, 2017
April 12, 2017
April 13, 2017
April 14, 2017
April 15, 2017
April 16, 2017

E3.11.1

28% 02:36
April 03, 2017
April 10, 2017
April 17, 2017
April 24, 2017
May 01, 2017
May 08, 2017
May 15, 2017
May 22, 2017

T3.11.2

Start State: The days screen of booking an appointment.

Actions: Click on one of the days and check that the times screen for that day appears.

E3.11.2

3.12

T3.12.1

Start State: The times screen for the booking of an appointment.

Actions: Click the back button on the phone and check that the days screen is reopened.

E3.12.1

T3.12.2

Start State: Tmes screen of booking an appointment.

Actions: Click on one of the times and check that the correct confirm appointment screen appears.

E3.12.2

3.13

T3.13.1

Start State: Helper details screen.

Actions: Press the back button and check the times screen appears.

E3.13.1

3.14

T3.14.1

Start State: Appointment Details screen.

Actions: Press back button and check that you get back to the list of appointments.

E3.14.1

3.15

T3.15.1

Start State: Weeks screen from the set timetable screen.

Actions: Press the back button and check the welcome helper screen appears.

E3.15.1

T3.15.2

Start State: Weeks screen of setting timetable.

Actions: Clicked on one of the weeks (in this case, April 10th, 2017) and check the days screen for setting timetable appears.

E3.15.2

3.16

T3.16.1

Start State: Days screen of setting the timetable.

Actions: Click the back button on the phone and check that the days screen appears.

E3.16.1

T3.16.2

Start State: Days screen of the setting timetable system.

Actions: Click one of the days and check the set times screen opens.

E3.16.2

3.17

T3.17.1

Start State: Set times screen of the set timetable system.

Actions: Press back button and check the set days screen of the timetable system.

E3.17.1

3.18

T3.18.1

Start State: Welcome elderly people screen.

Actions: Click the list of appointments button and check the list of appointments screen appears.

E3.18.1

T3.18.2

Start State: Welcome elderly screen.

Actions: Click the book an appointment screen and check the list of helpers nearby screen appears.

E3.18.2

3.19

T3.19.1

Start State: Welcome helper screen.

Actions: Click the set timetable button and check the weeks screen of the set timetable system appears.

E3.19.1

T3.19.2

Start State: Welcome helpers screen.

Actions: Click the list of appointments button and check if the list of appointments screen appears.

E3.19.2

3.20

T3.20.1

Start State: ListOfNearbyHelpers screen.

Actions: Click on one of the helpers and check the helperDetails screen appears.

E3.20.1

T3.20.2

Start State: ListOfNearbyHelpers screen.

Actions: Press the back button and check that the welcomeHelper screen appears.

E3.20.2

Test Series 4

4.1

T4.1.1

Start State: Register screen.

Actions: Fill in all information correctly and press the register as elderly button.

E4.1.1

Expected Result: Check the registration is successful and check that the phone number can then be verified, by responding using this code in the phone call received from Twilio.

T4.1.2

Start State: Register screen.

Actions: With not connected internet (turn on Airplane mode) attempt to press the register as helper button.

E4.1.2

Expected Result: Check there is a notification saying that there is no internet connection and that the loginOrRegister screen is opened.

4.2

T4.2.1

Start State: Login screen

Actions: Correctly fill in all the information and click the login button (both in turn) with no internet connection.

E4.2.1

Expected Result: Check there is a notification saying that there is no internet connection and that the loginOrRegister screen is opened.

T4.2.2

Start State: Login screen.

Actions: Fill in the email address and password of an incorrect user account and press the login as helper (and the login as elderly button, for another test)

E4.2.2

Expected Result: There is a popup dialog telling you that the login failed and you are not allowed into the rest of the application.

T4.2.3

Start State: Login screen.

Actions: Enter correct details and press the respective login button (for an elderly and helper account).

E4.2.3

Expected Result: You should get taken to the respective welcome screen, helper if logged in as a helper or elderly if logged in as an elderly person.

T4.2.4

Start State: Start screen (after having previously logged in successfully).

Actions: Press the login button.

E4.2.4

Expected Result: Be taken to the correct welcome screen (helper or elderly), depending on which user account had been previously logged into.

4.4

T4.4.1

Start State: Welcome screen for the elderly.

Actions: Click the 'List of Appointments' button.

E4.4.1

Expected Result: The list of appointments should be displayed in the below format. They should be ordered as upcoming appointments first, before going back into the past. The list of appointments can be verified by ensuring the appointments listed in test 1.7.11 are displayed.

4.5

T4.5.1

Start State: List of appointment screen.

Actions: Click any appointment.

E4.5.1

Expected Result: The appointment details screen will be displayed according to information from test 1.7.7.

T4.5.2

Start State: List of appointments screen.

Actions: Click an appointment in the past.

E4.5.2

Expected Result: The Set Rating and Set Review button should be visible, while the cancel appointment button should not be.

T4.5.3

Start State: The List of Appointments screen.

Actions: Click an appointment in the future.

E4.5.3

Expected Result: The Cancel button should be visible while the set review and set ratings buttons should not be.

4.6

T4.6.1

Start State: Appointment details screen of a specific appointment.

Actions:

1. Click the Set Rating Button
2. Pick a rating in the ratings screen.
3. Click submit.
4. Press the back button to return to the list of appointments, before selecting the same appointment to return to the set appointment details screen.

44% 04:10
Appointment Date: March 20, 2017
Appointment Time: 10:00
Name: Helper Ahuja
Contact Number: 447597711497
Address of Appointment:
32 Riverview Gardens
London
SW138QY
SET RATING
SET REVIEW

44% 04:11
Name of Helper
Helper Ahuja
Set Rating
SUBMIT

E4.6.1

Expected Result: From the appointment details screen, the set rating button should be clicked again. The rating should now be populated with the rating which was set in the test, in this case 8/10.

4.7

T4.7.1

Start State: Appointment details screen of a specific appointment.

Actions:

1. Click the Set Review Button
2. Write a review in the reviews screen.
3. Click submit.
4. Press the back button to return to the list of appointments, before selecting the same appointment to return to the set appointment details screen.

E4.7.1

Expected Result: From the appointment details screen, the set review button should be clicked again. The review should now be populated with the review which was set in the test, in this case “test review”.

44% 04:12
Appointment Date: March 20, 2017
Appointment Time: 10:00
Name: Helper Ahuja
Contact Number: 447597711497
Address of Appointment:
32 Riverview Gardens
London
SW138QY
SET RATING
SET REVIEW

44% 04:12
Name of Helper
Helper Ahuja
Set Review
test review
SUBMIT

4.9

T4.9.1

Start State: Logged into an elderly account registered with an invalid postcode on the welcome elderly screen.

Actions: Click the book an appointment button to get the list of helpers.

E4.9.1

Expected Result: No error is thrown and instead all the helpers are shown in a list.

T4.9.2

Start State: Logged into an elderly account registered with a valid postcode on the welcome elderly screen.

Actions: Click the book an appointment button to get the list of helpers.

E4.9.2

Expected Result: The helpers are displayed in the correct order, according to test 1.5.12, with the average rating correct (according to calculations based on test 1.8.1) where they have one.

4.10

T4.10.1

Start State: ListOfNearbyHelpers screen.

Actions: Click on a helper (in this case Helper Ahuja)

E4.10.1

Expected Result: Check the helper details screen is successfully opened and has all the correct details according to test 1.5.2. Check the see reviews button is available if the user has reviews which exist, or is disabled if they have no reviews.

T4.10.2

Start State: HelperDetails screen where the user has reviews which have been written about the helper.

Actions: Click the 'See Reviews' button.

E4.10.2

Expected Result: The screen should be populated with all the reviews of the helper, according to test 1.9.1.

4.11

T4.11.1

Start State: The weeks screen of any helper, having logged in as an elderly person.

Actions: Select the current week.

E4.11.1

Expected Result: No days before the day of testing are listed as available to click to select an appointment.

T4.11.2

Start State: First, on a helper account, set a day as being completely busy, therefore having no timeslots free. Then, on an elderly account, attempt to book an appointment with this helper – selecting the correct helper and then clicking the book appointment.

Actions: Click the correct week, for which the helper has at least one day entirely busy.

E4.11.2

Expected Result: The day should not be listed among the days which can be clicked and an appointment be booked on the day.

T4.11.3

Start State: As a logged in elderly person, book an appointment with any helper and at any time, going to AppointmentDetailsConfirm screen.

Actions: Click the confirm button.

E4.11.3

Expected Result: After being taken to the listOfAppointments screen, the newly created appointment should show up on this screen and be able to clicked, getting into the appointment details screen.

T4.11.4

Start State: In the same way as the previous test, an appointment should be created with any helper at any time.

Actions:

1. The confirm button should be pressed.
2. On the list of appointments screen which appears, the newly created appointment should be selected.
3. Then, the cancel button should be pressed.

E4.11.4

Expected Result: The list of appointments screen, once updated, should no longer show the appointment which was cancelled.

T4.11.5

Start State: Go to the list of appointments and select any appointment in the future.

Actions: Click the cancel appointment button.

E4.11.5

Expected Result: The list of appointments screen, once updated, should no longer show the appointment which was cancelled.

4.13

T4.13.1 & E4.13.1

Actions: Send an SMS to the Twilio SMS number from a verified number when there is definitely time for the appointment wth a helper (for example by setting yourself as available using the nearest helper).

Expected Result: Message saying that an appointment has been created with the helpers name and the time that the appointment has been created.

T4.13.2 & E4.13.2

Actions: Send an SMS from a Twilio verified number with no valid date or time.

Expected Result: Text with the following text: “You failed to create an appointment as your message text was not of the right type. Ensure the time is in the format of HH:MM”

T4.13.3 & E4.13.3

Actions: Send an SMS to the Twilio from a verified number requesting an appointment at a time where no appointment is available – by going into the database and setting everyone as busy at this time.

Expected Result: Text response saying that “Nobody could be found”

4.14

T4.14.1

Start State: Logged in as a helper.

Actions:

1. Click Set Timetable.
2. Pick a week and day.
3. Set a specific format of being free and record this.

40% 04:26
Welcome:
Helper Ahuja
SET TIMETABLE
LIST OF APPOINMENTS

40% 04:26
April 03, 2017
April 10, 2017
April 17, 2017
April 24, 2017
May 01, 2017
May 08, 2017
May 15, 2017
May 22, 2017

E4.14.1

Expected Result: Go back into that day after closing the app and reopening it and check that the day has the same state as it was previously updated with.

T4.14.2

Start State: Logged in as a helper.

Actions:

1. Click Set Timetable.
2. Pick a week and day.
3. Set a specific format of being free and record this.

E4.14.2

Expected Result: Only be able to book an appointment at the times when the helper set themselves as being free.

Actions:

1. Log in as an elderly person.
2. Click book an appointment and select the same helper as for which the timetable was set. In this case, it is “Helper Ahuja”
3. Click book appointment.
4. Select the week and day for which the timetable was set.
5. Check the times which are available correspond to those set previously.

40% 04:26
April 10, 2017
April 11, 2017
April 12, 2017
April 13, 2017
April 14, 2017
April 15, 2017
April 16, 2017

39% 04:29
12:00 to 14:00

Appendix E – Works Cited

AgeUK, n.d. *Technology and Older People Evidence Review.* [Online]
Available at: http://www.ageuk.org.uk/documents/en-gb/for-professionals/computers-and-technology/evidence_review_technology.pdf?dtrk=true
[Accessed 8 01 2017].

BBC, 2015. *The generation that tech forgot.* [Online]
Available at: http://www.bbc.co.uk/news/technology-32511489
[Accessed 09 04 2017].

Department for Work and Pensions, 2014. *Accessible communication formats.* [Online]
Available at: https://www.gov.uk/government/publications/inclusive-communication/accessible-communication-formats
[Accessed 09 04 2017].

Draw.io, 2017. *Draw.io.* [Online]
Available at: https://www.draw.io/
[Accessed 09 04 2017].

D-Tech IT, 2017. *D-Tech IT: The Right IT Support for your Business.* [Online]
Available at: http://www.dtechit.co.uk/
[Accessed 09 04 2017].

EverythingTech, n.d. *EverythingTech: IT Support Manchester.* [Online]
Available at: https://www.everythingtech.co.uk/
[Accessed 09 04 2017].

Fortune, 2017. *Lenovo, HP, And Dell Lead the Shrinking PC Market.* [Online]
Available at: http://fortune.com/2017/01/11/lenovo-hp-dell-pc-market/
[Accessed 09 04 2017].

Herring, S. C., 2008. *Questioning the Generational Divide: Technological Exoticism and Adult Constructions of Online Youth Identity.* [Online]
Available at: https://is.muni.cz/el/1423/jaro2013/ZUR589f/um/herring__2008_.pdf
[Accessed 09 04 2017].

HTL London, n.d. *HTL London.* [Online]
Available at: https://www.htl.london/
[Accessed 09 04 2017].

Internet Live Stats, 2017. *Internet Users.* [Online]
Available at: http://www.internetlivestats.com/internet-users/
[Accessed 04 09 207].

IT Guy, 2017. *IT Guy.* [Online]
Available at: http://www.itguy.com/
[Accessed 09 04 2017].

Keating, L., 2014. *Android or iOS? Which do millennials prefer?.* [Online]
Available at: http://www.techtimes.com/articles/15084/20140908/draft-android-reigns-over-ios-millennials.htm
[Accessed 09 04 2017].

LucidChart, n.d. *ER Diagram Symbols and Notation.* [Online]
Available at: https://www.lucidchart.com/pages/ER-diagram-symbols-and-meaning
[Accessed 09 04 2017].

Madden, K. Z. a. M., 2012. *Internet Adoption.* [Online]
Available at: http://www.pewinternet.org/2012/06/06/main-report-15/

Max, 2016. *Update: New Supported Android versions.* [Online]
Available at: http://blog.ionic.io/market-share-movement-android/
[Accessed 10 03 2017].

McGregor, G., n.d. *IT Support.* [Online]
Available at: https://www.grantmcgregor.co.uk/
[Accessed 09 04 2017].

PCWorld, 2008. *The 7 Worst Tech Predictions of All Time.* [Online]
Available at: http://www.pcworld.com/article/155984/worst_tech_predictions.html
[Accessed 09 04 2017].

Reddit, 2017. */r/techsupport.* [Online]
Available at: https://www.reddit.com/r/techsupport/
[Accessed 09 04 2017].

SequelPro, n.d. *SequelPro.* [Online]
Available at: https://www.sequelpro.com/
[Accessed 09 04 2017].

Smith, A., 2014. *Older Adults and Technology Use.* [Online]
Available at: http://www.pewinternet.org/2014/04/03/older-adults-and-technology-use/
[Accessed 09 04 2017].

StackExchange, 2017. *StackExchange.* [Online]
Available at: http://stackexchange.com/
[Accessed 09 04 2017].

Tech Support Forum, 2017. *Tech Support Forum.* [Online]
Available at: http://www.techsupportforum.com/
[Accessed 09 04 2017].

Tsai, A., 2017. *Andrew Tsai: Expert Service.* [Online]
Available at: http://andrewtsai.co.uk/
[Accessed 09 04 2017].

Wayne, M., 2017. *Mitchell Wayne Technologies.* [Online]
Available at: http://www.mitchellwaynetech.com/
[Accessed 09 04 2017].

Yahoo, 2017. *Yahoo Answers.* [Online]
Available at: https://uk.answers.yahoo.com/
[Accessed 09 04 2017].