Time Analysis

Group R

How estimates were constructed
Initially the project leader would generate all the time estimates by consulting the team member/s in charge of the requirement/task. During Sprint 2 the team decided on a more collaborative approach in calculating the estimates because we thought it would lead to better estimates. So, we held a group meeting in week 8 an excerpt of the minutes is shown below.

Teams interpretation of timesheet
We viewed general tasks as a high-level look at what our team needed to do below is a table describing briefly the main tasks.

Task	**Description**
Learning Techniques	Learning about a specific technology in order to implement a feature e.g. learning a mobile framework
Research and Investigation	Researching about anything related to the project e.g. research on sleep diary questions
System Design	Planning out the system architecture. The implemented architecture can be found here https://github.com/noobling/prof-comp/blob/master/Sprint%203/Software%20Documentation.md#architectural-overview
Administrative Documentation	Writing documents relating to project e.g timesheets, minutes and user manuals
Project Meeting and	All meetings done relating to project e.g. client meetings, mentor meetings

communica tion	

Requirements to us meant the high-level features we had to implement for the client. They valued each feature for us and we subsequently prioritized the highest value ones. We viewed requirements as an overlap with general tasks that is why the resulting hours sheet is not completely accurate. The table below describes some of the requirements.

Requirement	**Description**
Data Visualisation	Graphically display data back to user in a meaningful way e.g. graphs
Record Sleep Data	A form with 23 questions and the backend to support persistent storage of form
Compatible web app	Website working on most modern devices
Convert data into excel format	Data can be downloaded as an excel worksheet

Android App
The total estimated time for completion for the Android app was 50 hours. In general, the time spent on the project can be divided into six tasks: the Requirement analysis, Research and learning, System design, Test design, Project meeting and Documentation. The area that cost the most extra time was learning and implementing techniques.

In sprint 2, the major task was to design the system model based on the requirements given by the client. The estimated time spent on user interface design was about 5 hours. The reason for this was that there were five pages that had to be designed. As our team has an experienced android developer, each page was expected to take 1 hour to create. This hour includes the coding of an XML file and theme file. However, the actual time spent on the interface design was about 10 hours. The screen of a mobile is smaller than that of a computer, so the questionnaire pages did not fit comfortably on a single page, as such, time had to be taken to re-organise the questions from 5 pages down to 3.

Another large difference between estimated time and actual time was found in implementing the communication between the app and the server. The process was found to be much more complicated than expected and many different methods had to be tried before a solution was found. Much time was spend on fixing these problems as there were 23 different fields that had to be sent to the server and if any was not exactly correct the whole process would fail. On particular problem was that we were sending the field “timeGoIntoBed” should have been “timeGotIntoBed”. This slight difference was overlooked over two days of troubleshooting, causing significant delay.

Apart from these issues, most android related tasks were found to take about as long as expected with only slight variations. As such, all other tasks were completed on time so any extra time was spent on fixing the issues mentioned above.

iOS App

We failed to deliver the native iOS applications with around 44 hours left in expected development time even after spending approximately 30 hours already due to the lack of knowledge in the area. In addition, for a long period of time we only had one member working on the iOS branch because the SME (subject matter expert) in iOS left the team midway through Sprint 2, setting back the development significantly.

Much of the time spent during Sprint 2 was learning Swift and conducting research on a good application framework to construct the app, as our team recognised that it was impractical to build the app from scratch after considering the time constraints set on this project, as well as the other commitments the team members had. It was easier to just build off what someone else had created, which has its own advantages and disadvantages. The progress during sprint 2 slowed down quite a fair bit after our iOS SME left the team, leaving the only member left in the iOS development team to carry on without him. It was difficult to help the iOS team during this period as all other sub-teams were occupied with their own tasks.

During Sprint 3, the main problem with the development of the iOS branch was not with using Swift, as the iOS team knew enough Swift to code, but with the application framework we found: ResearchKit. The framework was specifically designed for purposes surrounding recording medical data, which was perfect for our task. Even after looking back, we still believe that ResearchKit was the best framework for our sleep diary application for the iOS, however the framework itself was plagued with horrible documentations which failed to show examples and tutorials on the usage of the framework and at the same time many of the items within the documentation were out of date, leading to many depreciation warnings in the iOS code. This forced us to look up example codes and GitHub bug reporting (dating back to 2015) and adapt code to suit our sleep diary, which took a fair amount of time since we could not understand the methods used in the code due to poor documentation. Another issue we would have faced was the backend framework; ResearchKit does not offer a backend solution for the data collected from the application, meaning even if we managed to implement the native iOS app a backend framework was required to collect and manage the data, which would cost us more time in doing research.

In retrospect, we believe that if we had more knowledge regarding Swift prior to this project and also finding ResearchKit earlier, there is a good chance that enough progress would've been made to have the iOS branch up to a presentable state, satisfying most criteria specified by the client.

Investigation into large discrepancies in estimated and actual times web app

The estimated times for building the web applications fluctuated a lot e.g. between week 6 and week 7 the estimated time dropped from 40 hours to 20 hours because it was way easier to convert data into excel than initially thought. However, the time increased to 28 hours estimated when the client wanted to make changes to the time pickers and add more graphs.

At the end of the project the web app still had about 10 hours of estimated work mainly due to the additional work we had to do after user testing. For example, making the time and duration pickers to the minute as the client felt this more appropriate than rounding times. Since the client prioritized this change we had to hold off on implementing notifications because this was one of the lowest valued features ($5).

Lessons Learnt
The estimates are usually not accurate mainly due to the nature of software, it is an iterative process. When a feature is built it has to be thoroughly tested in a real setting. The results and feedback are taken into consideration for the next iteration. It is very hard to estimate how long a feature will take because we had no clue how many iterations it has to go through. The time picker is a good example of this, it was very quick to implement, only 30 minutes. However, we had to show the client many different time pickers because each one we made did not completely meet the client's needs. Going through this process it added 5 hours to development time. Bugs were another reason why our estimates were inaccurate. It is difficult to know which features would encounter bugs that will consume a lot of our time. Going forward we understand now that the project requirements will continually change due to the iterative process and so the estimates will have to change. The estimates provide a guideline, but we have to remain flexible in order to accommodate for the changes that will occur.