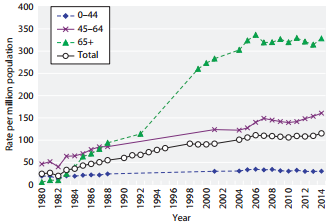
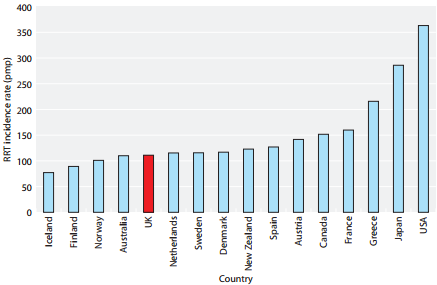
**Abstract -** Chronic kidney disease (CKD) is an important public health issue. It describes abnormal kidney function and/or structure. Despite many other diseases, it is usually asymptomatic until the late stages. In small but significant percentage of people, CKD can progress to End-stage kidney disease, which requires a lot of personal commitment in order to survive. This includes strict diet and regular appointments for Renal Replacement Therapy (RTT**)** for the rest of the individual’s life.In the majority of cases, RTT starts with receiving dialysis treatments, until the living kidney is transplanted. Since the process of receiving a transplant requires a three year waiting in average, it is particularly important to maintain individual’s health until the kidney is transplanted.

The aim of this nine-month project is to develop an **everyday mobile application** that fits the needs of renal dialysis patients. Market research shows that there are very few mobile **applications** available for people on dialysis. Through an agile-based approach, an application was developed to (1) remind patients about their appointments, (2) Check their health by scanning their blood test results, (3) react immediately if health check failed, (4) prepare them for their treatments and assist them during these treatments, (4) allow them an easy access to their personal details, clinical staff contact details and frequently asked questions, (5) offer them calendar of events with an option to add their own events in the system, (6) **Show graphs of how their blood test results changed during time.**

The final solution is an Android mobile application sitting on the top of a SQL database back-end. The **service** utilises a variety of powerfull, unique **frameworks(libraries?),** including Android-volley[**REFERENCE**], MPAndroidChart[**REFERENCE**]. A typical user of the service **can interact with an application every day: before, during and after treatments.** This report gives an in-depth breakdown of the process under which the project was developed.

**Keywords***Dialysis, Haemodialysis, Chronic Kidney Disease, mobile applications, Android*

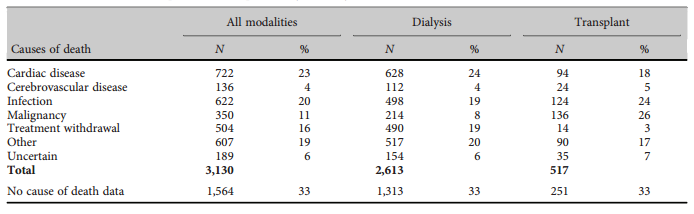
**1 Introduction**According to UK Renal Registry 18th annual report [1.1], there were 7411 new RTT patients in the UK in 2014. Apart from that, between 1980 and 2014, RTT incidence rate increased by more than 2 times and chances of RTT incident increased with population ageing. (Figure 1) Across the UK, as a whole, the renal replacement therapy incidence rate for 2014 was higher than for 2013 and 2012. In the worldwide, UK statistical data is not showing the highest incidence rate, according to data supplied by United States Renal Data System (USRDS) (Figure 1.1). It is self-evident that there are strong reasons to consider Chronic Renal Disease as an important public health issue, since despite of only small amount of CRD cases, where disease progresses to the Kidney failure which requires RTT, the amount of incidents is growing each year. [2]   
  
***Figure 1****: RRT incidence rates between 1980 and 2014*

**

***Figure 1.1****: International comparison of RRT incidence rates in 2013(Data from USRDS)*[2.1]

In 2013, the total of 2613 RRT patients died for different reasons. Infection was the third most popular reason causing death in Dialysis patients. (Table 1)

Tayside Kidney Patients Association (T.K.P.A.) [2.2] is an association ran by volunteers. It aims to support Tayside Renal Patients by providing help and advice, when appropriate. Currently, T.K.P.A. **does not have any IT solutions apart** from the website. All the patients who joined T.K.P.A. are either running dialysis patients or those, who already received a kidney transplant. Unlike in many other associations, T.K.P.A. members are very close to each other, since they meet at least 3 times a week at the Renal Dialysis Unit in Ninewells Hospital. The most experienced patient in the association is receiving RRT for more than 30 years. T.K.P.A. is always here, when a CKD patients requires help or assistance.



***Table******1****: Cause of death in prevalent RRT patients by modality, 2013*

This project is devoted to development and evaluation of Android application that would work as an everyday mobile application for RRT patients. The application is aimed to check patient’s health and help with patient’s preparation to the treatment, through the treatment and in their aftercare. The aim of the project is to produce a mobile application to support daily RRT patient’s needs.

This report describes the **incremental and iterative development procedure with elements of agile approach** the author took to develop a mobile **solution** for T.K.P.A.

**The remainder of this document presents background research providing context for the project (Section 2), the project proposal resulting from client discussions (Section 3), individual project handling techniques (Section 4), the design, implementation and testing process for each phase of development (Sections 5-7), a functionality and testing overview of the final product (Section 8), and a conclusive discussion including student reflection and notions for future work (Section 9).**

**2 Background**

**2.1 Kidneys**The kidneys are two beam-shaped organs, the size of the fist, located on either side of the body, just beneath the ribcage. The main role of the kidneys is to filter waste products from the blood before converting them into urine. Apart from that, the kidneys help to maintain blood pressure and the correct level of chemicals in the body which, in turn, helps heart and muscles to function properly. The kidneys also produce the active form of vitamin D that keeps bones healthy, and simulate production of red blood cells by producing erythropoietin. [1]  
**2.2 Chronic kidney disease**Chronic Kidney disease (CKD) is an important public health issue. It can be defined as the reduced ability of the kidney to carry out its tasks. In spite of it being common, with its popularity increasing with age, chronic kidney disease is an independent risk factor for other diseases, particularly cardiovascular disease. It often coexists with other cardiovascular conditions meaning that it needs to be managed alongside other diseases and risk factors such as diabetes and hypertension as well as the social needs that come with frailty and multiple conditions. In a small, but significant percentage of cases, chronic kidney disease progresses to end stage renal disease, which may require renal replacement therapy. This progression and the risks of other vascular events, such as stroke and heart failure can be reduced if chronic kidney disease is identified and managed, early diagnosis is therefore essential. There are five known stages of CKD. These stages are mainly based on measured or estimated GFR (Glomerular Filtration Rate) (Table 1.1) [2]

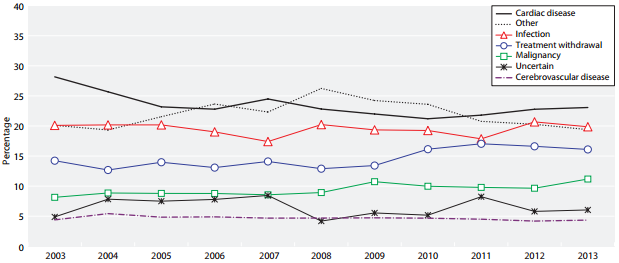
|  |  |  |
| --- | --- | --- |
| Stagea | GFR (ml/min/1.73 m 2 ) | Description |
| 1 | ≥ 90 | Normal or increased GFR, with other evidence of kidney damage |
| 2 | 60–89 | Slight decrease in GFR, with other evidence of kidney damage |
| 3A | 45–59 | Moderate decrease in GFR, with or without other evidence of kidney damage |
| 3B | 30–44 |
| 4 | 15–29 | Severe decrease in GFR, with or without other evidence of kidney damage |
| 5 | < 15 | Established renal failure |
| a Use the suffix (p) to denote the presence of proteinuria when staging CKD. | | |

***Table 1.1****: Stages of chronic kidney disease* [3]

**2.2.1 Causes of chronic kidney disease**A number of conditions can cause permanent damage to the kidneys and/or affect their function and lead to CKD. According to [4], diabetes, high blood pressure and ageing kidneys are the three most popular causes of CKD.

**2.3 End-stage kidney disease**End-stage kidney disease (also called end-stage renal disease (ESRD) or established renal failure (ERF) is the last stage of chronic kidney disease. It is caused by inability of kidneys to support patient’s body needs. Since the kidneys are performing one of the leading roles in the body, it is essential to maintain their functionality, which requires renal replacement therapy (RTT). RTT can take a number of forms; kidney transplantation, haemodialysis or peritoneal dialysis. Receiving a kidney transplant can be a very challenging process. According to [5]:

* Only around one in three people with kidney failure is suitable for a transplant.
* Not all kidney transplants are accepted by body.
* Ideally, kidneys will come from a close relative because of the likelihood of sharing the same blood group and tissue type.
* An individual who needs a kidney transplant, but don’t have a suitable living donor, will have to wait until suitable deceased donor kidney becomes available. On average, the waiting time is two to three years, but if the individual has a least common blood group or tissue type, the waiting time increases.
* From April 2014 to April 2015, around 3,000 kidney transplants were carried out in the UK, but there were still more than 5,000 people on the waiting list for a kidney by the end of this period.



***Figure 2****: Cause of death in RRT patients by year*

Since the chances of obtaining a kidney transplant straight away after diagnosis are small, usually individual is offered a dialysis treatment, while waiting for a transplant.

**2.4 Haemodialysis**Dialysis is a procedure that is a substitute for many of the normal functions of the kidneys. Haemodialysis uses a dialysis machine with a special type of filter to remove excess waste products and water from the blood. There are multiple ways to connect a patient to the dialysis machine. One of them is known as “Cimino fistula”, which is a direct connection between an artery and vein in the arm. Haemodialysis can allow individuals to live productive and useful lives, even though their kidneys no longer work adequately.

**2.4.1 Causes of death during haemodialysis**Over the last years the quality of haemodialysis treatment within NHS Tayside improved with the help of modern medical technologies, but death rate within dialysis patients is still high. According UK Renal Registry [6], infection is one of the leading causes of death within dialysis patients (Figure 2). It is very easy to catch an infection, since fistula is the place, which is used as a link between the patient and dialysis machine. In order to decrease a death rate of Dialysis Patients, infections should be prevented at earliest stages.

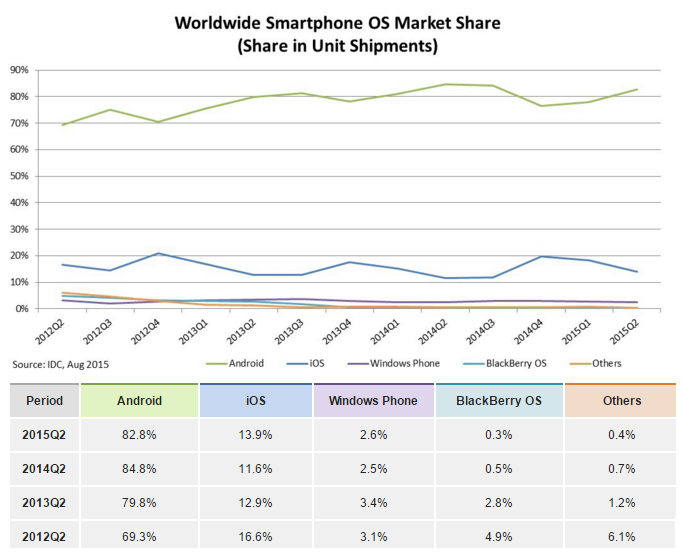
**2.5 Important aspects of CKD patients’ life**Since their kidneys are not working properly, there is a number of life-changing guidelines which should be followed by people, diagnosed with CKD need to follow in order to improve their health. First of all, CKD patients follow special diet, which restricts their fluid intake and avoids potassium, phosphorus, salt, protein rich foods. Then, if the patient is undertaking RRT, attending haemodialysis appointments three times a week, 3 to 5 hours per treatment is essential. For new patients, following this strict schedule is of the most challenging adjustments needing to be made. [7] Apart from that, maintaining a hand hygiene is very important for people, who are undertaking their treatment through the fistula, since it is a very vulnerable place.

**2.6 Access to health check results for CKD patients/NHS medical staff**Currently, it is possible to access health check results from desktop computer using Patient View system [8], where registered patients can access their health check results and view them in a grid or table format. NHS medical staff are using eMed system to input results in the system.

**2.7 Market Research**The amount of mobile applications available on the market is very limited. According to National Kidney Foundation [9], there is a total of 4 applications available for the CKD patients. Most of them are pocket guides to nutrition assessment. Currently there are no applications available that offer RRT patient to view their blood test results explained on a mobile device. Apart from that, there are no applications that offer assistance during dialysis. On the other hand, there is a Patient View website available, it is **mobile friendly(REFERENCE)** and it allows patients to communicate with their doctor using messaging, but the blood test results shown there are very hard to understand due to abbreviations and medical terms being widely used. Moreover, instead of offering explanations for different blood tests using “**on click popup**”, in Patient View these explanations are only accessible from third-party websites. Summing up, there is no such a system that would offer RRT patients to access nutritional information, their personal details, essential clinical staff contacts, prepare them for their treatments, guide them through their treatments and review their blood test results in one mobile application.

**Choosing smartphone OS for development**In order to select the platform for development, it was essential research mobile OS market. According to data from the International Data Corporation (IDC) Worldwide Quarterly Mobile Phone Tracker [REFERENCE IDC.COM], Android OS dominated the market in 2015 with 82.8% share in unit shipments (Figure X market share). After taking this into consideration, the decision was made to develop the application for Android OS.

***Figure X:*** *Worldwide Smartphone OS Market Share in 4 year period (2012-1015) (Share in Unit Shipments) [****REFERENCE ICE MARKET SHARE]***



**2.8 Client – T.K.P.A. (Tayside Kidney Patient Association)**Tayside Kidney Patient Association (hereafter referred as “the client” is an association ran by volunteers. Following a meeting with T.K.P.A., the student consulted with the head of association, Audrey McHugh and **came to an agreement over the Honours Project proposal**. The client’s aim is to help new and ongoing patients with CKD. The client is currently missing a mobile application that could offer both renal specific features and a part of existing functionality of the Patient View system.

**3 Specifications   
3.1 Project Selection**Before having initial meeting with a client, the project required ethical approval. Since RRT patients are considered disabled under Equality Act 2010, primarily the student was advised to apply for NHS ethical approval [10] and Caldicott Guardian ethical approval [11]. Initially there were many reasons for selecting advanced ethics. First of all, such an approval would allow to visit Renal Unit and speak to RRT patients on NHS grounds. Then, easy recruiting of patients for design evaluation would be possible. Finally, Caldicott Guardian ethical approval allows to request a real medical data that could improve the precision of application when performing a health check. Unfortunately, Caldicott Guardian ethical approval was not granted, because it required to state, which specific blood test results and other medical information are required for the project. During initial state of the project, interviews with NHS staff were not held yet, and in order to minimise risks it was decided to abandon the idea of getting Caldicott Guardian ethical approval for study. Apart from that, due to the fact that getting NHS ethics is a very long process, there existed a possibility that it will not be possible to get ethics approved in time, allocated for the project. Instead of getting NHS ethics, the student decided to find a patient association that is not directly connected to NHS and use a standard **Dundee University School of Computing Ethical Committee approval.** This solution significantly decreased time spent on ethics approval. On the other hand, it restricted the student from speaking to patients on NHS grounds, which made a patient recruitment more challenging. While ethics were awaiting approval, the student created a first version of project proposal and made an attempt to contact a client for the first time. Unfortunately, the only way to contact the client was by using a form on their website, which was not working properly. Later, when calling Renal Dialysis Unit, charge nurse transferred student to the client. Once the ethical approval was granted, an initial meeting with a client was set up on 18/11/2015. During this meeting first version of project proposal document was reviewed and updated. The student was not given any specific technical implementation guidelines, the client asked to consider an average age of the patients (55 years old) when developing an application. The client also added, that majority of the patients have either minimal **or no** computer skills. Apart from that, it was mentioned that despite the fact that large part of the existing RRT patients **has** basic skills of understanding their blood test results, there is a significant amount of patients, who do not understand them and require help of understanding them, which usually require waiting until they meet their consultant and discuss them. Client added, that it would be much better, if there was a way to offer patients an easy explanation of their results in order to be able to seek help straight away if they **notice big changes in their results**. It was agreed, that design workshop will take place due to clients interest in considering patients’ age in application development. The client was asked to find a group of volunteering patients from association who would be interested to participate in a **product design evaluation and testing.** The client expressed interest in the **idea** and future collaboration. The following key outcomes of the meeting were delivered: updated project proposal, information on CKD related websites, and related medical contacts of people, who contribute into association and can be helfpful with medical aspects of the research, CRK and, more importantly, RRD medical leaflets.

**3.2 Background**The client had around 60 patients and 5 NHS nurses and 10 other members involved in the organisation. It is important to note that due to nature of the disease (see Figure 1), most of the members are considered as an older age group patients. Apart from that, the client already mentioned, that large part of organisation members has minimal or no computer skills. The client had a basic website, but it was planned to upgrade it in the future, so the website could support user authentication. It was also planned to extend the functionality of the current website so it could provide user with a range of new options, such as contacting their clinical staff, looking up the blood test results, communicating with each other, having a treatment timetable accessible from the website. When these changes will have taken **place**, the client suggested, that a mobile application that provides the user with tools for everyday uses (at home or during their treatment) and part of the website’s functionality will be required. The client proposed the concept to create a **mobile** application, after discussing potential functionality required with RRT patients within organisation and some members of the clinical staff.

**3.3 Designer – Anna Morozova (3rd year Digital Interaction student at University Of Dundee)**Anna Morozova (hereafter referred as “the designer” is 3rd year Digital Interaction student, currently studying at University of Dundee. Designer was recruited by the student to create a user interface for the application, project poster and provide design skethes. Both student and designer would benefit from this decision. First of all, the student would benefit from the proper application design, since Digital Interaction students are taught to create user interfaces for different digital solutions according to international standards. Designer would benefit from being able to include her work to personal portfolio with reference to particular Honours Project.

**3.4 Initial Proposal**Shortly after the initial meeting, the head of T.K.P.A. had to resign from the project for 5 months due to personal circumstances. According to outcomes of the initial meeting with a client, the student generated a proposal document (**Appendix G**) which contained a number of implementation guidelines to outline the basic functionality required from the application. During proposal document design it was decided to minimalize client’s involvement in the project until the client returns from leave. The guidelines stated below were constructed as a result of this action.

**Assistance to be provided by the client**

* Patients on RRT for design evaluation

**Primary functionalities expected from the system:**

* **to have very ”user friendly” design**
* to offer user registration and authentication
* to assist user during the treatment by showing a treatment timer and progress bar
* to prepare user for the treatment by showing information on hand-hygiene importance when starting RRT and giving advice on fluid intake and nutrition during the RRT.
* to retrieve essential contacts from the database and display them under separate menu item
* to retrieve events assigned for user from the database and show them in a calendar
* to retrieve patient’s details from the database and display them under separate menu item
* to retrieve frequently asked questions and answers and show them under separate menu item.

**Secondary functionalities possible for the system:**

* to add personal events
* to change personal events
* to notify user for upcoming events
* a possibility to call telephone numbers in the application.
* a health check that checks blood test results (according to normal values for these blood tests) and indicate if the result failed
* to view graphs of blood test result changes through the time
* a possibility to call NHS 24 if health check failed

**3.5 Further Discussion**Due to client resigning from the project for 5 months, student had to take a risk of waiting until the client comes back and move evaluation of the application to the very end. Such risks were evaluated and risk assesment document was created (**Appendix E**). allocated for the project.

Due to medical nature of data that would be used by a **system** the student decided to recruit medical staff for assistance with medical part of the system. 2 NHS nurses and 1 NHS registrar were recruited as a result of this decision. After meetings were held with medical staff the following information was gathered:

* blood tests used for RRT patients
* normal values for these tests (taking into account that RRT patient’s blood is tested)
* routine checks made by medical staff when they are identifying if there are concerns about patient’s health and how frequent these checks are
* nutrition restrictions for RRT patients
* fluid intake restrictions for RRT patients
* information about dialysis treatments, their difference and special aspects of each, the routine of treatment, time of the treatment by default, how the patient is prepared for the treatment, what happens during the treatment and what steps are taken before patient is allowed to go home
* Information on medical staff that can be considered as core and can be included into essential contacts.
* Information on preferred method of contact when medical staff is contacted by patients
* Information on type of patient’s personal details, that have the most importance for RRD patients
* hand hygiene aspects for RRT patients
* RRD and CKD specific websites with useful information that can be used for the project [12],[13],[14],[15]

**4 Project Management**From the proposal discussion held between client and student, an outline of project development was constructed. Initially, it was planned for the main development procedure to be based on an agile approach, however due to client’s resignation from the project for uncertain time, it was not possible to construct a typical, business-valued product backlog or carry out sprints in the traditional manner. The student decided to become a product owner for uncertain time, either until the project ends or until the client will resume from the leave. There were multiple reasons this action. The main reason was that due to client’s unavailability to participate in the project for uncertain time or provide end users for evaluation, it was not possible to follow strict agile model. The reason for this was the fact that agile model is bounded to its Agile Manifesto [16] and 12 principles [17] and situation with the client made a clash with the following principles of Agile Manifesto:

* “Business people and developers must work   
  together daily throughout the project.”
* “Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.”

At the same time, by becoming a product owner the student could have a full control on the project, perform sprint review and retrospective, create user stories, prepare acceptance tests for these user stories. This meant that the student had a full creative control over the development cycle and the planning of project.

The number of tasks were created for the project**. For time management purposes, these tasks were divided into months until the project deadline.** Then, five clear sprints emerged from the planning process:

* Pre-development: requesting a **PHP namespace** on Dundee University School of Computing ZENO server and SQL database on Dundee University School of Computing SILVA server, SQL Database schema generation, filling database with mock data, submitting ethics, meeting with NHS staff and a client
* Back-end at server side: Developing PHP scripts for data retrieval.
* Back-end at client side: developing server-client based Android Application
* Front end: application user interface
* **Design evaluation and testing**

These cycles were named Phase 0, Phase 1, Phase2, Phase 3, Phase 4 respectively. Expected time allocation for each task in the phase was estimated by the student and by summing up all these time allocations, the total time needed for the phase was estimated. As the result of this action, the project timetable was created (see Table 4).

|  |  |  |  |
| --- | --- | --- | --- |
| **Phase** | **Start** | **End** | **Length** |
| 0 | Nov 2015 | Dec 2015 | 4 weeks |
| 1 | Dec 2015 | Mar 2016 | 14 weeks |
| 2 | Jan 2016 | Apr 2016 | 13 weeks |
| 3 | Mar 2016 | Apr 2016 | 8 weeks |
| 4 | Apr 2016 | Apr 2016 | 1 week |
|  |  |  |  |

***Table 4****: Projected development cycles in timetable*

The project cycle planned for **16 weeks of** development, commencing the week on 6th December 2015 and ending on 15th of April 2016. The final 2 weeks were to allow for code refactoring and academic deliverables.

**Risk Assesment**According to [20],

**Personas**According to [21], the purpose of personas is to create reliable and realistic representations of your key audience segments for reference. These representations should be based on qualitative and some quantitative [user research](http://www.usability.gov/what-and-why/user-research.html) and [web analytics](http://www.usability.gov/what-and-why/web-analytics.html). Creating personas would provide a significant benefit for the project. Personas help to focus decisions surrounding site components by adding a layer of real-world consideration to the conversation. They also offer a quick and inexpensive way to test and prioritize those features throughout the development process. During the Phase 3 of the project, personas, along with use cases, user stories and requirements would allow the system to benefit from better user interface design. Document containing personas **[APPENDIX NUMBER]** was created, after the potential benefits personal could offer to the project were taken into consideration.

**Use cases**According to [22], a use cases is a written description of how users will perform tasks on your website.  It outlines, from a user’s point of view, a system’s behaviour as it responds to a request. Each use case is represented as a sequence of simple steps, beginning with a user's goal and ending when that goal is fulfilled. Use cases offer number of benefits, for example they help in explanation of how the system should behave and help brainstorming. Apart from that by using use cases it is easier to predict what could go wrong. They also provide a list of goals and this list can be used to help in establishing the cost and complexity of the system. Creating use cases document [**APPENDIX NUMBER**] helped student to negotiate which functions become [requirements](http://www.usability.gov/how-to-and-tools/methods/requirements.html) and are built.

**Example of USE CASE**

**Requirements specification**Software requirements are description of features and functionalities of the target system. Requirements convey the expectations of users from the software product. Requirements specification document (**Appendix NUMBER**) was created for the project according to IEEE Recommended Practice for Software Requirements Specifications (IEEE Std 100-1992)[18]. The document included a list of functional and non-functional requirements that were established from the problem specification based on the earlier discussion with the client and NHS Renal Dialysis Ward staff. The rationale for each requirement was created. As stated in [19], requirement rationale is an explanation of the reasoning behind the decision, statement of requirement, design approach etc. The requirements document was created when the client resigned from the project and the student became the owner of the project. During that period, the student was thinking of methodology change and this is the one of the reasons why requirements specification document was created instead of creating the list of user stories**, as usually done when using** agile methodology. The other reason for this action was the fact, that user stories provide a good cover over functional parts of the system, while it is still challenging to produce a good list of user stories that cover non-functional parts of the system, such as security and access times. After considering these **two facts**, the list of user stories was created from the list of generated project requirements.

REQUIREMENTS EXAMPLE

**User stories**User stories are one of the primary development artefacts in agile software development. A user story is a very high-level definition of a requirement, containing just enough information so that the developers can produce a reasonable estimate of the effort to implement it, as stated in [23]. The student decided to use "as an X, I want Y so that Z..." format, because the client initially had basic skills of using a computer and when the client resumed from the leave, such an easy to understand high-level format would make it possible for the client to review the list of user-stories. Due to client’s unavailability the student made a decision to involve some of the NHS clinical staff who supervised RRT patients on daily basis to sort existing user stories, accept or decline them and add new user stories. Also, staff was asked to write **acceptance tests for all the stories**, based on best of their knowledge about RRT patients’ routine. The effort needed to finish each of the user stories was calculated. As a result of this teamwork, the document, containing user stories with acceptance tests and completion estimates for each of them was created [**APPENDIX NUMBER**].

USER STORY EXAMPLE

**Project backlog**According to [24], A product backlog is a prioritized list of work for the software developer that is derived from the roadmap and its requirements. The most important items are shown at the top of the product backlog so the software developer knows what to deliver first. The software developer doesn't work through the backlog at the product owner' space and the product owner isn't pushing work to the development team. Instead, the development team pulls work from the product backlog as there is capacity for it by iteration. The student considered multiple tools available for managing a student backlog. There were 5 tools to choose from: Microsoft Office Excel spreadsheet, Trello, Pivotal Tracker and Yodiz. After considering pros and cons of each tool, the student decided to use Pivotal Tracker, mainly because it offered free subscription for students, it allowed various analytical tools, for example easy generation of productivity charts. It also had extended amount of included features available, if compared to its competitors and offered suitable user interface. The project backlog hosted on Pivotal Tracker [**LINK AND APPENDIX**] was created as the **result of these considerations.** On weekly basis, the backlog was reviewed and updated. An initial estimate (in **points**) for how long the task may take to complete was allocated and progress marked per week as appropriate (**sprint backlog**). Pivatal tracker included 4 columns for different progress of tasks: backlog, current and done. Each task was automatically moved into definite column during its lifecycle. In order to control it, the task had the following buttons: Start, Finish, Deliver, Accept/Reject, each moved the task to **representing column**.

**Definition of Done**According to **[X1**], Definition of Done is a simple list of activities (writing code, coding comments, unit testing, integration testing, release notes, design documents, etc.) that add verifiable/demonstrable value to the product. Focusing on value-added steps allows the developer to focus on what must be completed in order to build software while eliminating wasteful activities that only complicate software development efforts. The Definition of Done was also produced by the student and added to the Product Backlog [**LINK AND APPENDIX**]. **Student decided that in order to consider the feature as “done” must have contained the refactored code, compiled without errors, included coding comments where appropriate and had a user interface.**

**Class Diagram**Class Diagram provides an overview of the target system by describing the objects and classes inside the system and the relationships between them. Class diagram offer a number of potential benefits for the project. Mainly, because it provides a wide variety of usages; from modelling the domain-specific data structure to detailed design of the target system. Class diagram for the project **was generated in Android Studio** and can be viewed in [**APPENDIX NUMBER**]

**Database schema AND ER DIAGRAM**A database schema is the skeleton structure that represents the logical view of the entire database. It defines how the data is organized and how the relations among them are associated. It formulates all the constraints that are to be applied on the data [26].

ER DIAGRAM DEFINITON

In order to generate a proper ER diagram, information gathered from interviewing NHS staff was carefully studied. This information lead to the proper structure of “medical\_history”, “profile” and “contacts” tables. In order to create the rest of the tables, **updated project proposal document**, user stories, use cases and class diagram were carefully studied. This lead into creation of database schema for the project (**Image X) and the following ER Diagram (IMAGE Y). Full ER DIAGRAM of the database can be viewed in [APPENDIX NUMBER]**

Preview of DB SCHEMA

**Weekly meetings with project supervisor**Formal supervisory meetings were used as an opportunity for the student to discuss the progress, describe findings and alert project supervisor of any problems. At the beginning of the project, it was agreed that these meetings will be held every Thursday on weekly basis during first and second semesters. For special occasions, such as going on an arranged holiday, unavailability during other module deadlines or sickness meeting were cancelled or moved to a different date. After every meeting, the meeting minutes were recorded into the project logbook.

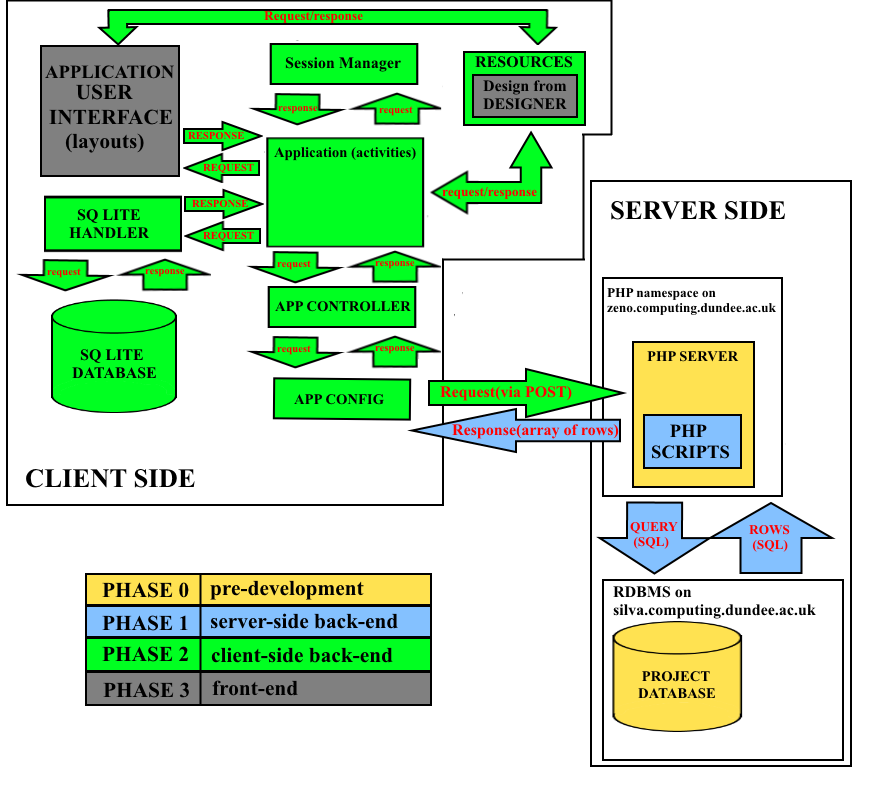
**Meeting minutes**Meeting minutes are the written or recorded documentation that contain information about what was discussed and what happened during a meeting. The meeting minutes are generally taken or recorded during the meeting so that participants have a record of what happened during this meeting. The meeting minutes for the project were recorded into the project logbook during or after every meeting with a project supervisor and then **rewritten into the Google Docs document created specially for this purpose [REFERENCE]**.

EXAMPLE OF MEETING MINUTES

**Sprint review and retrospective**In Scrum, Sprint review is the process of demonstrating the features produced during the sprint to the product owner and anyone else who is interested. After the final sprint, the product owner is demonstrated with the final product. Sprint retrospective, according to [**X2**], is an inspect-and-adapt activity performed at the end of every sprint that involves continuous improvement opportunity for a Scrum team to review its process (approaches to performing Scrum) and to identify opportunities to improve it. Sprint review was performed by the student after completion of each of the project phases. Since the student became a product owner, it was decided to give that presentations to the project supervisor. After the review gathered from the supervisor and according to personal acknowledgement of problems and other challenges that occurred during the sprint, **Sprint Retrospective was performed on weekly basis.** These activities leaded to constant improvement of the software development quality. Every time, when Sprint review and sprint retrospective were performed, the outcomes were recorded into the logbook.

**Ethical approval (ALREADY IN PROJECT SELECTION)**

**Version control system**According to [**X2**], version control system (VCS) is a system that records changes to a file or set of files over time so that it is possible to recall specific versions later. There was a number of VCS to choose from. The student decided to choose Git due to having extensive amount of previous experience with this CVS. **Followed by the fact** that the student had multiple computers in use, the decision was made to store created Git Repository on a web-based Git repository hosting service called GitHub. GitHub offered a number of additional tools, such as good user-friendly interface and tools for analytics in graphs: contributions to master, code frequency, commits per day/week/month. One of the most useful tools available in GitHub is “Issues” section. When spotting a code or design issue, reporting it is the best practice software developer can take in order to make sure that it is solved afterwards. During software development phases of the project, “Issues” was constantly used by the student.[**IMAGE X ISSUES**] Every tool described in this section made it possible to make an additional improvement on planning and time managements during sprint retrospective. It is possible to view a repository in [**X3 LINK TO GITHUB**]



**Figure X:** *Expected system architecture including project phase breakdown.*

Screenshot of Issues section

Screenshot of Github Repository

**Logbook**As specified by requirements for project final portfolio, a diary of effort spent each week on tasks, initial ideas for program designs, design decisions and design sketches, trade-offs, notes, simple evidence of ideas development or, in other words, all the rough work done for the project was included in the project logbook (**Appendix NUMBER**). Undoubtedly, the project benefited from having a logbook, since it was used all the time for sketching ideas, making notes, especially during the software development part. Logbook was also used when arranging meeting with the client and NHS staff, doing design evaluation and application testing. When the student was writing a project report, logbook was used as well for planning purposes.**Twitter Account**In order to get public interested in the project, twitter account @dialysis\_app [**REFERENCE**] was created. It was decided to use social media to post progress made on the project. Time after time, the progess was tweeted. The software development part of the project would not directly benefit from having a twitter account, but undoubtedly, it could be used for increasing public awareness of CKD. Apart from that, the chance existed, that some of the people on RRD will notice this twitter account and provide “handful of additional advice” on features to include in the application.

**4.1 Phases**Each of the development phases contained a clear breakdown of tasks and formed a product backlog. Part of the tasks was dependent on its predecessor, while some parts were not due to some of the features being separated and not dependent on each other. The features that were dependent on their predecessor were prioritising the backlog. A physical agile board was set up by the student to track the progress of each task under the headings “to start”, “in progress”, “testing” and “completed”.

PHOTO OF THE BOARD

**Phase 0:** *pre-development*

1. Ethics submission
2. Meeting with NHS staff and client
3. Request a **PHP namespace** on Dundee University School of Computing ZENO server and SQL database on Dundee University School of Computing SILVA server
4. Set up the local hosting environment to work on the project while the request is being processed
5. Set up Source code control environment (VCS)
6. Creating boards on Pivotal Tracker
7. Set up a project database (RDBMS) with dataset created according to specification provided from the client and NHS staff during the meetings (lab test results)
8. Moving the local database to SILVA server
9. Selecting target Android API version and minimum Android API.

**Phase 1:** *Back-end at server side*

1. Development of DB Configutation and DB Connection PHP scripts
2. Development of DB functions PHP script, that contains all database statements for data retrieval from the database divided into following functions:
   1. DB constructor and Destructor
   2. storeUser() – *function to store user in the database, check for new user to be unique and check for successful store*
   3. getUserByEmailAndPassword() – *gets the user form the database by email and password*
   4. isUserExisted() – *checks if the user exists or not by email*
   5. hashSHA() *– encrypts password using Secure Hash Algorithm 1 (SHA-1) and returns encrypted password and salt*
   6. checkhashSha() – *decrypts password from SHA-1 using salt and password*
   7. getEssentialContacts() – *gets essential contacts from the database using user id*
   8. getProfile() – *gets user profile from database using user id*
   9. getFaq() *– gets FaQ(Frequently asked questions and answers) from the database in a loop*
   10. getEvents() *– gets events assigned for the user, ordered by event start time descending, using user id in a loop*
   11. createEvent() – *adds new event to the database by creating new row with events details, user id and then checks for successful store*
   12. changeEvent() – *updates existing events by using user id and old event data to find event and then replace it with new event data. After that, check for successful store is performed.*
   13. getMedHistory() *– gets user’s medical history using user id, ordered by date added descending*
3. Development of separate PHP scripts for receiving POST parameters, requesting data from database using DB functions, doing certain manipulations with data, packing the result and sending it data back to the client side
   1. change\_event.php
   2. contacts\_request.php
   3. create\_event.php
   4. events\_request.php
   5. faq\_request.php
   6. index.php
   7. login.php
   8. med\_history\_request.php
   9. profile\_request.php
   10. register.php

**Phase 2:** *Back-end at client side*

1. Setting up the project in Android Studio (WHY ANDROID STUDIO)
2. Configuring the project(min Android version, target android version, libraries and dependencies)
3. Add user-permissions for the application
4. Add SQL Lite Handler and Session Manager classes
5. Add AppConfig and AppController classes
6. Developing Activities
   1. Launcher
   2. LoginActivity
   3. RegisterActivity
   4. MainMenu
   5. Treatment
   6. Healthcheck
   7. HealthcheckCheckHealth
   8. Healthchecks\_view\_graphs
   9. CalendarEvents
   10. calendar\_events\_view
   11. CreateEvent
   12. ChangeEvent
   13. Contacts
   14. Profile
   15. FAQ

**Phase 3** *Front end at client side ( user interface )*

1. Create layout for all the activities
2. Add resources:
   1. Video for washing hands tutorial
   2. Temporary drawables
3. Add Values:
   1. Colors.xml
   2. Dimens.xml
   3. Strings.xml
   4. Styles.xml
4. Request design from the designer
5. Add ”drawables” created by the disigner to the resources
6. Apply design created by the designer to the application (layouts)
7. Perform design adoption where needed due to code design limitations

**Phase 4:** *Design evaluation and testing*

1. Arrange meetings with participants, provided by the client (when the client resumes from the leave). Act as described in risk assessment document, if the client does not provide participants for testing or never resumes from the leave.
2. Arrange interviews with participants to perform Black Box testing [**REFERENCE X**]
3. Process results and make final changes for the project if needed

**4.3 Projected system overview**Following the construction of the product backlog, a system overview diagram was sketched to guide the development process **(Figure X).** Phase 0 was to create project database, request PHP namespace and relational database on Dundee University School of Computing servers. Phase 1 was to develop server-side back-end: PHP scripts for requesting POST parameters from the client-side (application), fetching data from the database and sending it back to the client-side (application). Phase 2 was to develop client-side back-end: Android application (activities) functionality, **methods to allow data request using POST parameters**, app controller and app config classes, add essential resources and basic temporary design, SQ lite database for storing user data locally and SQ Lite handler for this database. Phase 3 was to create a front end: request application design from the designer, add drawable resources, received from the designer and apply them to the application (layouts).

**5 Phase 0**

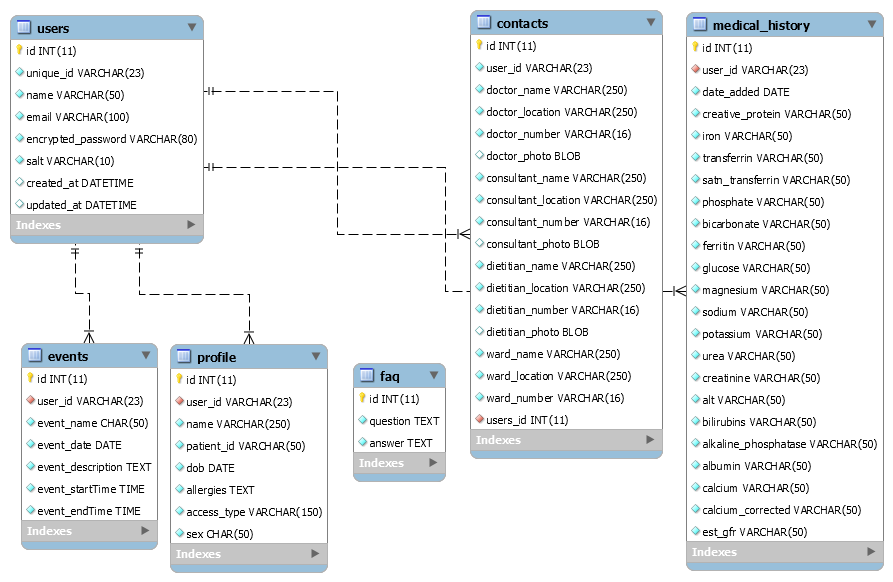
**5.1 Design**

**5.1.1 Server-side Database Design**Initially, local MySQL database with default MyISAM engine was created for the project at the at the beginning of the phase. The problem occurred, when the student decided to create Foreign Keys, due to MyISAM not supporting foreign key constraints. The decision was made to move to InnoDB engine. InnoDB provided advantage over MyISAM. For example, InnoDB supported the following features that were not supported by MyISAM:

* Foreign key constraint support
* Transactions support
* Frequent insert/update/delete
* Row locking (multi-processing on a single table)
* **Relational database design**

Following the discussion with client that took place during initial meeting and interviews with NHS staff, the dataset needed for the served-side database was identified. After that, the database was created locally using MariaDB database and PHP interpreter provided with XAMPP [**REFERENCE X**], free and open source cross-platform web server solution stack package developed by Apache Friends, consisting mainly of the Apache HTTP server, MariaDB database and interpreters for scripts written in the PHP and Perl programming languages. As the result, entity-relationship diagram (E-R Diagram) was drawn for the created database **(Figure X**).

When the requested Database and PHP namespace were received from Dundee University, the server-side database was migrated to MySQL database created on SILVA server (silva.computing.dundee.ac.uk). Additionally the PHP namespace was set up on ZENO server (zeno.computing.dundee.ac.uk). The decision of moving to the real servers was made due to architecture of the created system. By staying with the local setup for server-side database and PHP server it would not be possible to achieve an easy connection to that database and server when leaving the local network. By setting up a Virtual Private Network (VPN) on the phone used for testing and debugging purposes, it made it be possible to access project servers from anywhere, where internet connection existed. This made it possible to avoid carrying a laptop every time, when showing the progress done on application or performing user **testing was** done. Apart from that, the setup time needed for presenting functionality of the application was significantly reduced.

****

**FIGURE X: E-R diagtam for the created database**

**5.2 Implementation and testing**For server-side database management, it was decided to use HeidiSQL [REFERENCE X3], OpenSource tool, designed for browsing and editing data in MySQL databases. It offered simple user interface and all the esential functionality at the same time. The decision was made due to previous experience of working with this tool. Created database was named “renaldialysisdb” to suit project’s name. It consisted of six tables, each having its own purpose. All the tables had primary keys and foreign keys. For the primary key in each table, unique integer field “id” with auto-increment was created. The database had six tables: “users”, “profile”, “contacts”, “events”, “medical history” and “faq”. Apart from “faq”, all the tables were related to “users” table on “unique\_id” field, using foreign key “user\_id”. Using foreign keys allowed to cross-reference related data across table. Due to the fact that “faq” would contain data that will remain same for all the users, it was decided not to include foreign key in this table. Each of the created table had its own purpose:

* users – *storing user authorisation data: id of the user in a table, unique\_id, username, email address, encrypted password and salt for its decryption.*
* profile – *storing user details: profile id, id of the user this profile belongs to, patient’s name, hospital id, date of birth, allergies, access type for dialysis and gender.*
* contacts – *storing essential contacts: contacts id, id of the users these contacts belong to, doctor’s name, phone number, location and photo, consultant’s name, phone number, location and photo, dietitian’s name, phone number, location and photo and ward’s name, phone number and location.*
* events – *storing calendar events assigned for the user: event id, id of the user this event belongs to, event name, date, description, start time and end time.*
* medical history – *storing patient’s medical history (in blood test results): id of the medical history, id of the user this history belongs to, date when results were added, c-reactive protein value, iron value, transferrin value, SATN transferring value, phosphate value, bicarbonate value, ferritin value, glucose value, magnesium value, sodium value, potassium value, urea value, ALT value, bilirubins value, alkaline phosphatase value, albumin value, calcium value, corrected calcium value and estimated GFR value.*
* faq – *storing frequently asked questions and answers: question id, question and answer.*

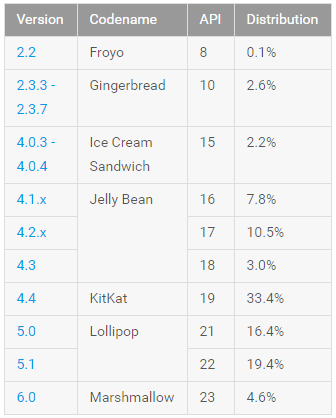
When creating fields, attention was paid to selection of appropriate data types. As the result, INT was selected for storing primary keys, VARCHAR for storing short texts or values, CHAR for calendar event names, TEXT for longer texts, such as calendar event descriptions or question and answers in “faq”, DATE for storing dates and TIME for storing times, DATETIME where combination of both was needed, BLOB for storing images or other large objects.

**Testing**  
Manual testing was performed to test the database for successful store of data. Manual testing is the process of manually testing software for defects. It requires a tester act as an end user to use the most of all application’s functionality to ensure correct behaviour. According to the procedure of manual testing, test plan[APPENDIX X] was created and the test was performed by querying (Code Snippet 1) the database after data addition to check for successful store.

**5.2.1 Selecting minimum Android API.**Application programming interface (API) is a set of routines, protocols and tools for building software and applications. In Android, API version is bounded to version of Android OS. Every time when new API version gets released, in order to stay up-to-date with technological progress on smartphone market, some functionality gets added. At the same time, some functionality gets deprecated. This is the reason why selecting a proper API level, when doing Android development is so important. Today, the developer can choose from ten API levels, each having different distribution across the users. By developing application for the newest Android OS 6.0 Marshmallow (API 23), the developer will be able to cover only 4.6% of all Android users. At the same time, the minimum API available is API 8 and it covers 0.1% of all users. The best approach is to select the minimum API with largest coverage. In order to do that, several resources were studied to identify the most popular Android OS version on the market. [REFERENCE X4] [REFERENCE X5] According to Statista: Distribution of Android operating systems used by Android phone owners in October 2015, by platform version, the most popular Android OS versions are: Kit Kat 4.4 (API 19): 38.9%, Lollipop 5.0 (API 21 and API22): 15.6% Jelly Bean 4.2.X (API 16, API 17 and API 18): 14.5%. According to official website for Android developers, the numbers are slightly different(TABLE X): Kit Kat 4.4 (API 19): 33.4%, Lollipop 5.0 (API 21 and API22): 35.8% Jelly Bean 4.2.X (API 16, API 17 and API 18): 21.3%. In order to cover as many users as possible, but keep most of up-to-date functionality it was decided to set the project for minimum API 17, which covers 90.5% of all current Android users.

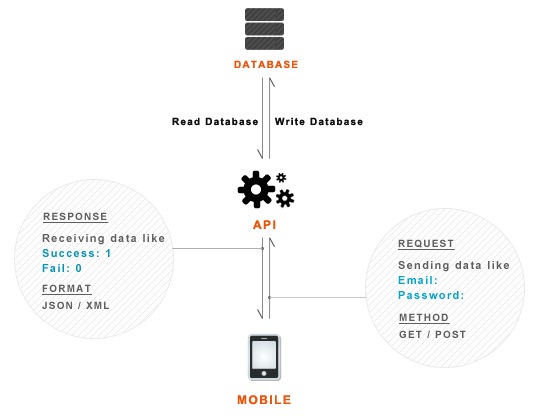
**SELECT** `id`, `user\_id`, `name`, `patient\_id`, `dob`, **LEFT**(`allergies`, 256), `access\_type`, `sex` **FROM** `renaldialysisdb`.`profile` **LIMIT** 1000;

***Code Snippet 1:*** *Making database queries to check for successful store*

  
***Table x:*** *Relative number of devices running a given version of the Android platform (data collected using 7-day period ending on April 4, 2016) [REFERENCE X5]*

**6 Phase 1**

**6.1.1 Server-side back-end design**  
After initial meeting with the client the student made a decision to follow REST architecture (**FIGURE X**) when implementing a system. REST (Representational State Transfer) is architectural style that describes six constraints: Uniform Interface, Stateless, Cacheable, Client-Server, Layered System, Code on Demand. These constraints, applied to the architecture, were originally communicated by Ron Fielding in his doctoral dissertation, where the basis of RESTful-style were defined [REFERENCE REST].   
  
***Figure X:*** *REST Architecture [REFERENCE x7]*There was a multiple reasons for this decision. First of wall, it would provide a flexible way to distribute data across the patients. Then, it would allow to have a flexible user account system. Allowing users to log in and log out would offer a high potential to the client, especially when the organisation **upgrades to the new website**. As the result, a class diagram was created for the server-side backend. **(Figure X)**



***Figure x:*** *Server-side back-end class diagram*

**6.2 Implementation and testing**In order to interact with the MySQL database, the student built a simple REST API. Its job was to get the request from client, interact with database and finally give the response back to client. Multiple PHP scripts were created during implementation part of the phase. Instead of writing a single script with all classes and functions, it was decided to create separate scripts for connection configuration, establishing connection with the database, querying database. Also, separate scripts were created for sending data back to the client-side. Please note that some of the code used for implementing server-side back-end was taken from [REFERENCE TUTORIAL].

**6.2.1 Config.php**Configuration script included definitions for the database server host, username, password and database name. This was done to avoid writing this information at the top of every file. Instead, it was decided to pass Config.php to “require\_once” function in DB\_Connect script. Using “require\_once” instead of more commonly used “include” benefited from different error handling: if an error occurs, “include” generates a warning and continues with script execution, while “require\_once” generates a fatal error and stops the script execution. Apart from that, when using “require\_once”, PHP checks if the file has already been included, and if so, PHP will not include it again.

6**.2.2 DB\_Connect**This script was created for establishing connection to the database. Following the idea of Config script, instead of performing database connection at the top of every script, it was decided to pass DB\_Connect.php to “require\_once” in other scripts, when connection to the database was required.

1. define("DB\_HOST", "silva.computing.dundee.ac.uk");
2. define("DB\_USER", "renaldialysis");
3. define("DB\_PASSWORD", "\*\*\*\*\*\*\*\*\*\*\*");
4. define("DB\_DATABASE", "renaldialysisdb");

***Note:*** *password removed for security reasons*

1. class DB\_Connect {
2. private $conn;
3. // Connecting to database
4. public function connect() {
5. require\_once 'include/Config.php';
6. // Connection
7. $this->conn = new mysqli(DB\_HOST, DB\_USER, DB\_PASSWORD, DB\_DATABASE);
8. // returns database handler
9. return $this->conn;
10. }

***Code Snippets X & Y:*** *creating configuration file and passing it to “require\_once” and using its contents for establishing connection to the database.*

**6.2.3 DB\_Functions.php**Followed by decision to include database functions into the separate script, DB\_Functions script was created. The script contains functions for querying database in different ways: selecting data, inserting data, updating data. Apart from that, the script contains functions for password encryption and decryption. Every function in the script was created to work with the other script that would pass parameter to certain function in DB\_functions, wait for the output, pack it and send back to the client side. The rest of this section describes functions in DB\_functions script.

**Constructor and Destructor**In the constractor, DB\_connect script is passed to “require\_once” and uses it to establish connection to the database. As mentioned earlier in this chapter, this was done to avoid repetition of code.

1. require\_once 'DB\_Connect.php';
2. // connecting to database
3. $db = new Db\_Connect();
4. $this->conn = $db->connect();
5. }
6. // destructor
7. function \_\_destruct() {}
8. /\*\*
9. \* Storing new user
10. \* returns user details
11. \*/
12. public function storeUser($name, $email, $password) {
13. $uuid = uniqid('', true);
14. $hash = $this->hashSSHA($password);
15. $encrypted\_password = $hash["encrypted"];
16. // encrypted password
17. $salt = $hash["salt"]; // salt
18. $stmt = $this->conn->prepare("INSERT INTO users(unique\_id, name, email, encrypted\_password, salt, created\_at) VALUES(?, ?, ?, ?, ?, NOW())");
19. $stmt->bind\_param("sssss", $uuid, $name, $email, $encrypted\_password, $salt);
20. $result = $stmt->execute();
21. $stmt->close();

*Code Snippet X: passing DB\_Functions contents to constructor, preparing connection and establishing connection when required*

**storeUser()**   
This function was created to allow user registration by inserting user details into database (“users” table). Name, email and password are passed from register.php script, password is encrypted by hashSHA() function. Unique ID is generated by uniqid(), PHP function that generates a unique ID based on microtime (current time in microseconds). For uniqid(), prefix “’’” is specified to allow multiple scripts generating IDs at exactly the same microsecond (Code Snippet X). Then, user details are inserted into “users” table in database by prepared statement with required variables bind as parameters. (Code Snippet X) Apart from that, by using email, check for successful store is performed (Code Snippet X).

1. $uuid = uniqid('', true);

***Code Snippet X:*** *Generating unique ID.*

1. $stmt = $this->conn->prepare("INSERT INTO users(unique\_id, name, email, encrypted\_password, salt, created\_at) VALUES(?, ?, ?, ?, ?, NOW())");
2. $stmt->bind\_param("sssss", $uuid, $name, $email, $encrypted\_password, $salt);
3. $result = $stmt->execute();
4. $stmt->close();

***Code Snippet X:*** *Inserting user details into database using prepared statement with bind variables as parameters.*

1. if ($result) {
2. $stmt = $this->conn->prepare("SELECT \* FROM users WHERE email = ?");
3. $stmt->bind\_param("s", $email);
4. $stmt->execute();
5. $user = $stmt->get\_result()->fetch\_assoc();
6. $stmt->close();
7. return $user;
8. } else {
9. return false;
10. }

***Code Snippet X:*** *Check for successful store*

**getUserByEmailAndPassword()**   
This function was created to allow user login. It gets user details from the database by using email and password passed from login.php script.

**isUserExisted()**   
This function was created to check if the user already exists, when registering new user. The check is done by counting number of rows returned by the query (Code Snippet X).

1. if ($stmt->num\_rows > 0) {
2. // user existed
3. $stmt->close();
4. return true;
5. } else {
6. // user not existed
7. $stmt->close();
8. return false;
9. }

**Code Snippet X:** counting rows returned by query when checking if user already exists

**hashSHA()**   
This function encrypts password using Secure Hash Algorithm 1 (SHA-1). The function works by using PHP sha1() and substr() functions. Sha1() converts normal string into SHA-1 hash and returns it in a string. Substr() returns specified part of the string . For salt, random SHA-1 hash is generated in a string and then part of that string gets becomes salt (Code Snippet X). Function returns encrypted password and salt in JSON object.

1. $salt = sha1(rand());
2. $salt = substr($salt, 0, 10);
3. $encrypted = base64\_encode(sha1($password . $salt, true) . $salt);

***Code Snippet X:*** *Generating salt and encrypting password*

**checkhashSha()**   
The function decrypts password when it is required. It uses base64\_encode() PHP function and salt for decryption (Code Snippet X). The function is used for user login.

1. $hash = base64\_encode(sha1($password . $salt, true) . $salt);

***Code Snippet X:*** *decrypting password by using base64\_encode and salt*

**getEssentialContacts()**  
The function is used by contacts\_request.php for requesting essential contacts from the database. The function uses user id passed from contacts\_request Essential contacts are received from the database by querying it using prepared statement and bind variable as statement parameter.

**getProfile()**   
The function is used by profile\_request.php for requesting personal details from the database. The function uses user id passed from profile\_request. Personal details are received from the database by querying it using prepared statement and bind variable as statement parameter.

**getFaq()**   
The function is used by faq\_request.php for requesting frequently asked questions and answers. Questions and answers are received in a while loop with assigned question and answer number to avoid mixing them in the future. This allows to keep ordering of questions and answers automatically for later use. The question count is also added for future reference (Code Snippet X).

1. while($response = $result->fetch\_assoc())
2. {
3. $resp["question$c"] = $response["question"];
4. $resp["answer$c"]= $response["answer"];
5. $c++;
6. }

***Code Snippet X:*** *WHILE loop with counter for number of question/answer, stops when there maximum row is reached.*

getEvents()   
This function is used by events\_request.php for requesting all the calendar events assigned for the user. The function uses prepared statement with bind variable (user\_id) as parameter. Events retrieved one by one in a while loop, with event start time in ascending order. Each event is assigned its number and the total number of events is counted. All event details are returned in JSON object.

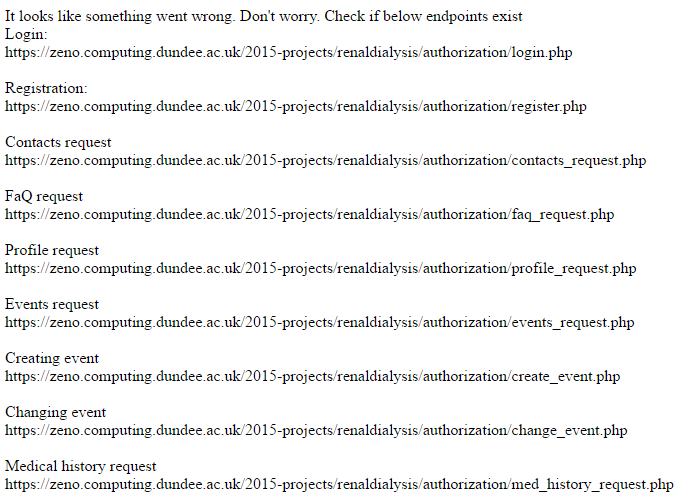
**createEvent()**   
This function is used by create\_event.php for inserting new event details into database. Prepared INSERT statement with bind variables (new event details) as parameters is used to insert new event into database. Event is also checked for successful store.

**changeEvent()**   
This function is used by change\_event.php for updating event details in the database. Old event details and new event details are passed from change\_event.php. Then two prepared statements are executed, first to find event that is about to be updated and second to actually replace that event with new event. After that, new event is checked for successful store.

**getMedHistory()**  
This function is used med\_history\_request.php for requesting user’s medical history from the database. Medical history is received using prepared statement with bind variable (user ID) in while loop. Each row is assigned a count, the total number of medical history rows is returned as well for future reference. Medical history along with number of rows is returned in JSON object.

**6.2.4 index.php**This script was created to indicate if there is problem with finding any of the other PHP scripts. If server-side administrator is unable to find PHP scripts needed for server-side functionality (for example, when during server migration), this page can be used as a reference. (**Figure X)**

***Figure X:*** *Guidance for server administrator.*



**6.2.5 login.php**This script was created to allow user login. The script uses login and password POST parameters and checks these parameters by passing them to getUserByEmailAndPassword() in DB\_Functions and calling it. If the users is found, user details are returned. Otherwise, if the login or password are wrong or missing, the error message is returned indicating that either login credentials are incorrect or required post parameters are missing. (Code Snippet X, Y).

1. if (isset($\_POST['email']) && isset($\_POST['password'])) {
2. // receiving the post params
3. $email = $\_POST['email'];
4. $password = $\_POST['password'];

***Code Snippet X:*** *Getting POST parameters*

1. } else {
2. // user is not found with the credentials
3. $response["error"] = TRUE;
4. $response["error\_msg"] = "Login credentials are wrong. Please try again!";
5. echo json\_encode($response);

***Code Snippet Y:*** *Error message output*

**6.2.6 register.php**This script was created to allow user registration. The script gets POST paramaters (name, email and password), checks if the user already exists by passing POST parameter (email) to isUserExisted() function in DB\_Functions and calling it. If the user already exists, the script returns error message to the client-side. If user does not exist, the user data is inserted into database by passing POST parameters to storeUser() function in DB\_Functions and calling it. Then, JSON object with user data is returned to the client side (name, email, generated unique ID, time the user was created, time the user was updated). Otherwise, the script returns error message to the client side. **6.2.7 profile\_request.php**This script was created to allow receiving profile data at client-side. It first checks if POST parameter user ID was submitted. Then, It uses getProfile() function from DB\_functions and calls it to get data and return it to the client-side as JSON object. Error message gets returned in case of missing parameters or other errors. **6.2.8 contacts\_request.php**This script was created to allow receiving essential contacts at client-side. It first checls of the user ID post parameter was submitted. Then, it passes user ID to getEssentialContacts() function and calls it. In case of success, essential contacts get packed into JSON object and returned to the client-side. Otherwise, error message is returned. **6.2.9 events\_request.php**This script was created to allow receiving calendar events at the client-side. It first checks if POST parameter user ID was submitted. Then, It passes these parameters to getEvents() function from DB\_functions and calls it to get all events and return them to the client-side as JSON object. Error message gets returned in case of missing parameters or other errors.

**6.3 create\_event.php**This script was created to allow receiving calendar events at the client-side. It first checks if POST parameters (user id, event name, event description, event date, start and end times) were submitted. Then, It uses passes these parameters to createEvent() function from DB\_functions and calls it to insert new event into database. Error message gets returned in case of missing parameters or other errors.

**6.3.1 change\_event.php**This script was created to allow changing calendar events at the client-side. It first checks if POST parameters (old event data and new event data) were submitted. Then, It uses passes these parameters to changeEvent() function from DB\_functions and calls it to insert new event into database. Error message gets returned in case of missing parameters or other errors.

**6.3.2 faq\_request.php**This script was created to allow receiving FaQ at the client side. The script call getFaq() function from DB\_functions. The result is packed into JSON object and returned to the client side. Error message gets returned in case of missing parameters or other errors.

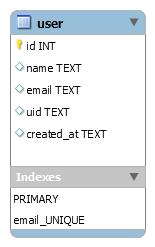
**6.3.3 med\_history\_request.php**This script was created to allow medical history at client-side. It first checks if POST parameter user ID was submitted. Then, It uses getMedHistory() function from DB\_functions and calls it to get data and return it to the client-side as JSON object. Error message gets returned in case of missing parameters or other errors.

**Testing**White-box testing was performed to test PHP scripts. The goal of the test was to make sure that scripts compile without errors and return desired output. White-box testing tests internal structures or workings of a program, as opposed to the functionality exposed to the end-user. One of the test cases was to ensure that every time, when connection with the database was established, it was closed when not required anymore. **(Code Snippet X)** This test case was particularly important due to face every database engine has a limit on a maximum number of simultaneous connections. If the connection is not closed, MySQL can run out of available connections (by default, max\_connections allows 100 open connections). Apart from that, every connection consumes client server’s resources. According to the procedure of white-box testing, test plan [APPENDIX X] was created.

1. $event = $stmt->get\_result()->fetch\_assoc();
2. $stmt->close();

***Code Snippet X:*** *closing connection when it is not required any more*

**7 Phase 2  
7.1 Design**As mentioned in previous chapters, the student made a decision to follow REST architecture, when implementing the system. The Phase 2 of the project was to implement Android application for the client-side of the system that would request data by communicating with server-side of the system (Figure X ARCHITECTURE OF THE SYSTEM)**.** Before developing the application, the student had to choose IDE (Integrated Development Environment). There were multiple IDEs to choose from: Ecliplse, JetBrains IntelliJ IDEA and Android Studio. It was decided to use Android Studio for the client-side implementation due to student’s previous experience with this IDE and the fact that it was official IDE for Android development and it included all the necessary tools for the application implementation. **7.1.1 Client-side database design**For the system to work properly, it was decided to store user details in a client-side database for future use. The system would benefit from caching in a number of ways. First of all, it would reduce the amount of server-side database queries in the future. Apart from that, such decision would allow a quick user ID, name and email retrieval. Every time the user would log into the system, the local SQLite database would be created by the application to store user details in the “user” table. After user logging out from the system, the application would delete local SQLite database for security reasons. As the result, E-R Diagram was drawn for the client-side database **(Figure X).**

 ***Figure X:*** *E-R diagram for the client-side SQLite database.* **7.1.2 Client-side back-end design**Android application development is focused on Activities. Activity in Android is an application component that provides a screen with which users can interact in order to do something, such as dial the phone, take a photo, send an email, or view a map. Each activity is given a window in which to draw its user interface[**REFERENCE ACTIVITY DROID**].  Each Activity in Android has a certain lifecycle. [**Figure X ANDROID LIFECYCLE**].

**  
*Figure X: Android Activity lifecycle***

**7.2 Implementation and testing**In order to build login and registration system, Volley library was used [REFERENCE VOLLEY] along SQLite. Please note that some of the code used for implementing client-side back-end was taken from [REFERENCE TUTORIAL].

**7.2.1 Application Permissions**Android forces applications to declare the permissions they require during installation. This is done to protect users privacy, security and cell phone bill. Application must declare permissions for most of things, starting with writing to SD card, accessing or writing the calendar to internet access, calling and sending SMS messages. For correct functioning of the client-side application, it was essential to allow the application to access to the list of accounts in the Accounts Service, querying profile data, reading user’s call log, allow internet access, reading user’s calendar and writing to it, reading/writing to external storage and reading phone state. The permissions were added to Android manifest file [FIGURE X PERM IN MANIFEST]. The purpose of Android manifest file is to present essential information about the application to the Android OS. This information is used by the system before running any of the application’s code.

1. <uses-permission android:name="android.permission.GET\_ACCOUNTS" />
2. <uses-permission android:name="android.permission.READ\_PROFILE" />
3. <uses-permission android:name="android.permission.READ\_CONTACTS" />
4. <uses-permission android:name="android.permission.INTERNET" />
5. <uses-permission android:name="android.permission.READ\_CALENDAR" />
6. <uses-permission android:name="android.permission.WRITE\_CALENDAR" />
7. <uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" />
8. <uses-permission android:name="android.permission.READ\_PHONE\_STATE" />
9. <uses-permission android:name="android.permission.READ\_EXTERNAL\_STORAGE" />

***Figure X:*** *Declaring android permissions in AndroidManifest.xml*

**7.2.2 AppConfig**All the URLs used for communication with server-side were included in this file for future access by Activities **(Code Snippet X).**

1. //Server events request url
2. public static String EVENTS\_REQUEST = "https://zeno.computing.dundee.ac.uk/2015-projects/renaldialysis/authorization/events\_request.php";
3. //Server create event url
4. public static String CREATE\_EVENT = "https://zeno.computing.dundee.ac.uk/2015-projects/renaldialysis/authorization/create\_event.php";

***Code Snippet X:*** *URLs for communication with API on the server-side (AppConfig.java).*

**7.2.3 AppController**This class is executed on application launch. Volley core objects are initiated in this class. **7.2.4 SessionManager**Session manager class maintains session data across the application using the SharedPreferences. In order to check login status, Boolean flag isLoogedIn is stored in shared preferences (Code Snippet X).

1. public void setLogin(boolean isLoggedIn) {
2. editor.putBoolean(KEY\_IS\_LOGGED\_IN, isLoggedIn);
3. // commit changes
4. editor.commit();
5. Log.d(TAG, "User login session modified!");
6. }
7. public boolean isLoggedIn(){
8. return pref.getBoolean(KEY\_IS\_LOGGED\_IN, false);
9. }

***Code Snippet X:*** *code to check login status (Session Manager.java) [REFERENCE TUTORIAL]*

**7.2.5 SQLiteHandler**This class was created to allow storing user data in SQLite database [Code Snippet X]. In onCreate() method [FIGURE ANDROID LIFECYCLE], new SQLite database is created. As mentioned earlier, this was implemented to allow easy access to user data when required without making request to a server [Code Snippet X]. All database rows are deleted when user is signs out from the system for security reasons. [Code Snippet X]

1. public void addUser(String name, String email, String uid, String created\_at) {
2. SQLiteDatabase db = this.getWritableDatabase();
3. ContentValues values = new ContentValues();
4. values.put(KEY\_NAME, name); // Name
5. values.put(KEY\_EMAIL, email); // Email
6. values.put(KEY\_UID, uid); // Email
7. values.put(KEY\_CREATED\_AT, created\_at); // Created At
8. // Inserting Row
9. long id = db.insert(TABLE\_USER, null, values);
10. db.close(); // Closing database connection

***Code Snippet X: storing user data in SQLite database [REFERENCE TUTORIAL]***

1. HashMap<String, String> user = new HashMap<String, String>();
2. String selectQuery = "SELECT \* FROM " + TABLE\_USER;
3. SQLiteDatabase db = this.getReadableDatabase();
4. Cursor cursor = db.rawQuery(selectQuery, null);
5. // Move to first row
6. cursor.moveToFirst();
7. if (cursor.getCount() > 0) {
8. user.put("name", cursor.getString(1));
9. user.put("email", cursor.getString(2));
10. user.put("uid", cursor.getString(3));
11. user.put("created\_at", cursor.getString(4));
12. }
13. cursor.close();

***Code Snippet X: fetching user data from SQLite database [REFERENCE TUTORIAL]***

1. SQLiteDatabase db = this.getWritableDatabase();
2. // Delete All Rows
3. db.delete(TABLE\_USER, null, null);
4. db.close();

***Code Snippet X: wiping SQLite database [REFERENCE TUTORIAL]***

**7.2.6 LoginActivity**This activity was created to allow user to login to the system. It uses two EditText objects, for password and email that are then sent to the server side using POST method when “Sign in” button is clicked on. The ProgressDialog is used to indicate that data is loading. If the user is already logged in, MainMenu Activity is loaded [Code Snippet X].Also, the login Activity has a button “Sign up” for registering new user. When the button is clicked on, RegisterActivity is loaded. If the login succeeds, MainMenu Activity is loaded. Otherwise, error message is displayed to the user.

1. // Session manager
2. session = new SessionManager(getApplicationContext());
3. // Check if user is already logged in or not
4. if (session.isLoggedIn()) {
5. // User is already logged in. Take him to main menu
6. Intent intent = new Intent(LoginActivity.this, MainMenu.class);
7. startActivity(intent);
8. finish();
9. }

***Code Snippet X: Checking if the user is already logged in by calling SessionManager instance.***

**7.2.7 RegisterActivity**This Activity was created to allow user registration. It works in nearly the same way as LoginActivity, apart from data being sent to server using POST to insert it into database and in the event of success user data is added to SQLite and the user gets sent back to the LoginActivity. Toast is used to notify user about successful registration (Code Snippet X).

1. Toast.makeText(getApplicationContext(), "User successfully registered. Try login now!", Toast.LENGTH\_LONG).show();

***Code Snippet X: use of Toast.makeText for notifying user about the registration success.***

**7.2.8 Launcher**Launcher activity used to show user the main logo of the application for two seconds. In the future, if the client decides to update application, this could be used for loading complex UI elements

**7.2.9 MainMenu**This activity sets up main menu for the application. **7.3 Treatment**This activity guides the user through the process of the treatment. In this Activity, a number of interesting solutions were implemented. First of all, counter class was implemented to allow timer to be displayed during treatment layout part along with progress bar [**Code Snippet X**]. Then, videoView is used to allow displaying video (the guide to wash hands properly) in one of the layout parts [**Code Snippet X**]. Also, timePickerDialog was used to allow setting treatment timer (time in Time Picker is displayer in 24 hour format to allow setting up to 24 hour countdown timers). Apart from that, due to inability of Android applications to read HTML tags from strings, the decision was made to implement HTML spanner **[Code Snippet X**].

1. public void onTick(long millisUntilFinished) {
2. long millis = millisUntilFinished;
3. String hms = String.format("%02d:%02d:%02d", TimeUnit.MILLISECONDS.toHours(millis), TimeUnit.MILLISECONDS.toMinutes(millis) - TimeUnit.HOURS.toMinutes(TimeUnit.MILLISECONDS.toHours(millis)),
4. TimeUnit.MILLISECONDS.toSeconds(millis) - TimeUnit.MINUTES.toSeconds(TimeUnit.MILLISECONDS.toMinutes(millis)));
5. System.out.println(hms);
6. timeLeftValue.setText(hms + " left");
7. progressStart++;
8. progressBar.setProgress(progressStart);

***Code Snippet X:*** *using OnTick() for timer with progress bar bind to it.*

1. //setting videoView
2. videoView = (VideoView)findViewById(R.id.washHandsVideo);
3. Uri video1 = Uri.parse("android.resource://" + getPackageName() + "/" + R.raw.hand\_washing\_converted);
4. //setting washing hands video
5. videoView.setVideoURI(video1);
6. videoView.setMediaController(new MediaController(this));
7. videoView.requestFocus();

***Code Snippet X:*** *setting up videoView for playing video file in the application. Using getPackageName() to avoid hardcoded URIs.*

1. // get html strings
2. String fluid\_text\_html = getString(R.string.treatment\_prep\_fluid\_text\_html);
3. String food\_text\_html = getString(R.string.treatment\_prep\_food\_text\_html);
4. //making spanned for displaying as html for those strings
5. Spanned fluid\_text\_spanned = Html.fromHtml(fluid\_text\_html); // used by textViewFluidHtml
6. Spanned food\_text\_spanned = Html.fromHtml(food\_text\_html); // used by textViewFoodHtml

***Code Snippet X:*** *HTML spanner to allow strings with HTML tags in strings.xml to be displayed.*

**7.3.1 Healthcheck**This activity is used as a “link” between main menu and HealthcheckCheckHealth. In this activity, the user is given choice to perform a health check by pressing a button. **7.3.2 HealthcheckCheckHealth**This activity was created to perform a health check. The most important method in the Activity is requestMedHistory, which requests user’s medical history by posting user ID to the server side. If the request succeeds the JSON gets received, converted to a String. Then, the string gets cleaned from “rubbish” (curly braces, etc.) and split into pairs. After that, the data gets put into hash map for easy access **[Code Snippet X].** The health gets checked by comparing blood test result values with the normal reading, received from NHS medical staff [Code Snippet X]. Method checkIfEmergency() checks if the test was failed and if it was, the user receives an Alert Dialog with alert (asking if the user feels unwell) and option to either call NHS24 from the app or cancel [Code Snippet X].

1. String[] pairs = savedResponse.split("\",\"");
2. for (int i = 0; i < pairs.length; i++) {
3. String pair = pairs[i];
4. String[] keyValue = pair.split("\":\"");
5. Log.d(tag, "keyvalue 0: " + keyValue[0] + " keyvalue1 " + keyValue[1]);
6. med\_history\_map.put(keyValue[0], keyValue[1]);

***Code Snippet X:*** *Splitting cleaned response string into pairs and storing in hash map.*

1. alkaline\_phosphataseValue = med\_history\_map.get("alkaline\_phosphatase" + entryCount);
2. if (Double.parseDouble(alkaline\_phosphataseValue) > 130 || Double.parseDouble(alkaline\_phosphataseValue) < 30)
3. //bold red
4. {
5. alkaline\_phosphatase.setText(alkaline\_phosphataseValue + "U/L");
6. alkaline\_phosphatase.setTypeface(Typeface.create(alkaline\_phosphatase.getTypeface(), Typeface.BOLD));

***Code Snippet X:*** *Checking blood test results with normal values received from NHS staff.*

1. alertDialogBuilder = new AlertDialog.Builder(HealthcheckCheckHealth.this);
2. alertDialogBuilder.setTitle("Are you feeling unwell?");
3. alertDialogBuilder.setMessage("Dialysis Check has detected that you have failed your last health check. If you are feeling unwell, you can call NHS24 now.");
4. alertDialogBuilder.setPositiveButton("Call NHS24", new DialogInterface.OnClickListener() {
5. public void onClick(DialogInterface dialog, int which) {
6. Intent caller = new Intent(Intent.ACTION\_DIAL);
7. caller.setData(Uri.parse("tel:" + 111));
8. startActivity(caller);
9. }

***Code Snippet X:*** *Alert dialog with option to call NHS24.* **7.3.3 CalendarEvents**This activity creates calendar of events. Additional class MyCalendar and GridCellAdapter were added for drawing calendar in a grid view. First of all, the current date is fetched by creating Android calendar instance. The date is then converted using Java Date library. During the process of drawing the calendar, multiple arrays are used for defining the month in a calendar, day of the week and total number of days in a month. The number of the month in a year is extracted from “today” date. Current month is printed by printMonth() that gets passed to current month and day indexes. It is possible to change the month by using buttons at the calendar head. Events are requested from the server-side by POST using requestEvents() method. Days that have events assigned to them and “today” have different styling to offer the user an easy way to see the difference from other days in the calendar. All days in the calendar are buttons and if the users clicks on one, calendar\_events\_view Activity is started with current date passed to it using putExtra() (Code Snippet X) . This was done to show a list of events for the selected day. Apart from that, event creation functionality was implemented in the calendar. Additional methods were created for getting default Android calendar and adding events into it to allow notifying the user about upcoming events, but not used. Decision was made to let the client decide in the future if this option is required or not. This was done for the reason that users might be using default Android calendar for their personal events and would not want for additional events to appear in this calendar. The feature has a high potential in the future.

1. Intent eventsView = new Intent(CalendarEvents.this, calendar\_events\_view.class);
2. eventsView.putExtra("date", date\_month\_year);
3. CalendarEvents.this.startActivity(eventsView);

***Code Snippet X:*** *Passing current date to new Activity.* **7.3.4 calendar\_events\_view**This activity was created to show events list for the day. It uses date passed by CalendarEvents Activity to look for events with specific date before adding them in a table. These events are requested by POST from the server side using slightly changed requestEvents() method from the CalendarEvents Activity. Creating new events or changing event functionality is implemented in this Activity. When pressing “+” or “Change event”, corresponding Activities are started. If the event is getting changed, event details are passed to new activity.

1. String date = getIntent().getStringExtra("date");

***Code Snippet X:*** *Receiving date passed by calendar Activity.*

**7.3.5 CreateEvent**This Activity allows to create new event by POST using addNewEvent() method with event details passed to it. Android Date Picker and Time Picker are used to select the proper date format. This date is then converted to database format **(Code Snippet X**).

1. OnTimeSetListener endTimeCallBack = new OnTimeSetListener() {
2. public void onTimeSet(TimePicker view, int hourOfDay, int minute) {
3. //adding millis for db format
4. dbFormatEndTime = (hourOfDay+":"+minute+":00");
5. //convert time to format that is best for UK
6. try {
7. SimpleDateFormat input = new SimpleDateFormat("HH:mm:ss");
8. Date dt = input.parse(dbFormatEndTime);
9. SimpleDateFormat output = new SimpleDateFormat("h:mm a");
10. String formattedTime = output.format(dt);
11. calendarFormatEndTime = formattedTime;
12. } catch (ParseException e) {}
13. eventEndTime.setText(calendarFormatEndTime);
14. }

***Code Snippet X:*** *Selecting event time in TimePicker and converting it to DB format using Java DATE when user sets the time.*

**7.3.6 ChangeEvent**This Activity allows to update event in the database by POST using updateEvent() method with new event details passed to it. ChangeEvent is slightly changed CreateEvent. The only difference from CreateEvents is that Activity gets event details from the calendar\_events\_view and shows it to user in EditText layout objects.  **7.3.7 Contacts  
7.3.8 Profile  
7.3.9 FAQ**

**7.4 Testing  
White box, TAG, checking isLoggedIn**

**8 Phase 3**

**Design evaluation**

**DON’T FORGET GRAPHS AND HOW USERS WERE RELATED TO THE SYSTEM**

**Final Product Description**

**Novel Work**

**Results**

**Future Work**

**Acknowledgements**

References:

[1.1] https://www.renalreg.org/wp-content/uploads/2015/01/web\_book\_07-04-16.pdf

[1] http://www.nhs.uk/conditions/Kidney-disease-chronic/Pages/Introduction.aspx

[2] <http://www.healthcheck.nhs.uk/document.php?o=81>

[2.1] Saran R, Li Y, Robinson B, et al. US Renal Data System 2015 Annual Data Report: Epidemiology of Kidney Disease in the United States. Am J Kidney Dis, in press

[2.2] http://www.tkpa.org.uk/

[3] <https://www.nice.org.uk/guidance/cg182/chapter/1-recommendations>

[4] <http://patient.info/health/chronic-kidney-disease-leaflet>

[5] <http://www.nhs.uk/Conditions/Kidney-transplant/Pages/Introduction.aspx>

[6] <https://www.renalreg.org/wp-content/uploads/2014/09/Chapter_5.pdf>

[7] <http://www.niddk.nih.gov/health-information/health-topics/kidney-disease/hemodialysis/Pages/facts.aspx>

[8] <https://www.patientview.org/#/>

[9] https://www.kidney.org/apps

[10] NHS ethics committee

[REFERENCE IDC MARKET SHARE] http://www.idc.com/prodserv/smartphone-os-market-share.jsp

[11] http://www.tasc-research.org.uk/images/cmsimages/pdf/Caldicott%20Approval%20Procedure.pdf

[12] https://www.kidney.org

[13] http://labtestsonline.org.uk/

[14] <https://www.renalreg.org/>

[15] <http://www.srr.scot.nhs.uk/>

[16] <http://agilemanifesto.org/iso/en/>

[17] <http://agilemanifesto.org/principles.html>

[18] IEEE Std 100-1992

[19] <http://www.chambers.com.au/glossary/rationale.php>

[20] http://www.usability.gov/how-to-and-tools/methods/personas.html

[21] http://www.usability.gov/how-to-and-tools/methods/personas.html

[22] http://www.agilemodeling.com/artifacts/userStory.htm

[23] <http://www.usability.gov/how-to-and-tools/methods/use-cases.html>

[24] https://www.atlassian.com/agile/backlogs

[26] <http://www.tutorialspoint.com/dbms/dbms_data_schemas.htm>

[X1] <https://www.scrumalliance.org/community/articles/2008/september/what-is-definition-of-done-(dod)>

[X2] <https://www.apachefriends.org/index.html>

[X3] <http://www.heidisql.com/>

[X4] Statista: Distribution of Android operating systems used by Android phone owners in October 2015, by platform version [ONLINE] available at: http://www.statista.com/statistics/271774/share-of-android-platforms-on-mobile-devices-with-android-os/ [Last accessed on 28 January 2016]

[X5] Android Platform Version Distribution [ONLINE] available at: http://developer.android.com/intl/ru/about/dashboards/index.html [Last accessed on 28 January 2016]

[REFERENCE REST] http://www.ics.uci.edu/~fielding/pubs/dissertation/rest\_arch\_style.htm

[x8] <http://www.androidhive.info/2012/01/android-login-and-registration-with-php-mysql-and-sqlite/>

[X10] http://developer.android.com/guide/components/activities.html

[X9] http://www.javatpoint.com/android-life-cycle-of-activity