Agile Development of Dialysis Check: Everyday Mobile Application for Dialysis Patients

Vladislavs Ignatjevs
AC40001 Honours Project
BSc (Hons) Applied Computing
University of Dundee, 2016
Supervisor: Dr. Keith Edwards

Abstract **-** *Chronic kidney disease (CKD) is an important public health issue. It describes abnormal kidney function and/or structure. Unlike 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 (RRT) for the rest of the individual's life. In the majority of cases, RRT 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) check patients' health by scanning their blood test results, (2) react immediately if health check failed, (3) 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 into the system.

The final solution is an Android mobile application sitting on the top of a SQL database back-end. The system is following REST architecture model with the help of Android-Volley [1], a powerful, unique networking library. A typical user of the system can interact with an application every day: before, during and after the 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 [2], there were 7411 new RRT patients in the UK in 2014. Apart from that, between 1980 and 2014, RRT incidence rate increased by more than 2 times and chances of RRT incident increase 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 2). 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 RRT, the amount of incidents is growing each year (Figure 1).

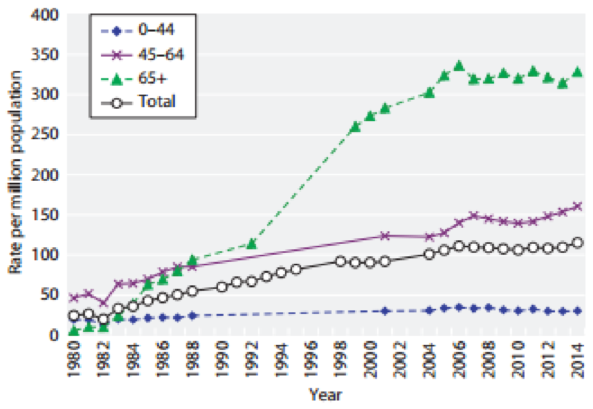


Figure 1*: RRT incidence rates between 1980 and 2014[1]*

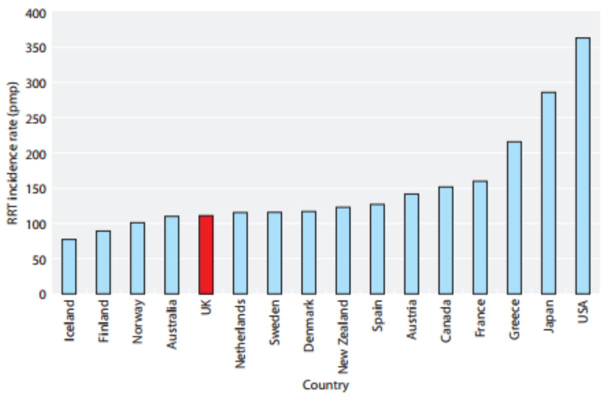


***Figure 2**: International comparison of RRT incidence rates in 2013(Data from USRDS)* [3]

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.) [4] 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.

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, assist through the treatment and in the aftercare. The aim of the project is to produce a mobile application to support daily RRT patient's needs.

This report describes the agile approach the author took to develop a mobile application for T.K.P.A.

The remainder of this document presents background research providing context for the project (Section 2), specification of the project (Section 3), individual project handling techniques (Section 4), the design, implementation and testing process for each phase of development (Sections 5-8), a functionality and testing overview of the final product (Section 9), and a conclusive discussion including student reflection and notions for future work (Section 10).

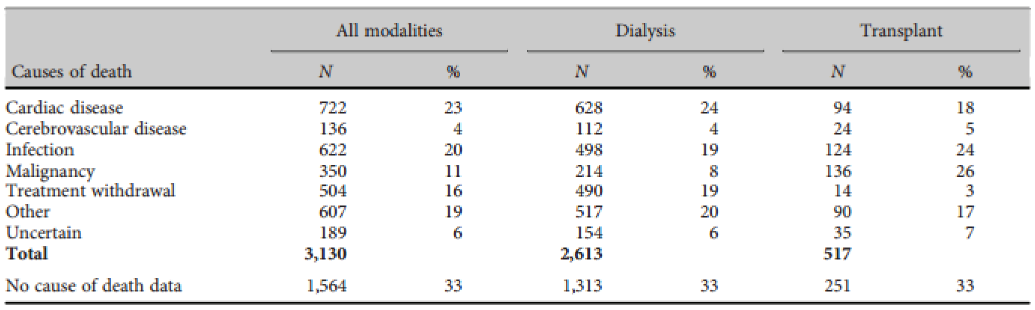
2 Background

2.1 Kidneys

The kidneys are two bean-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. [5]

2.2 Chronic kidney disease

Chronic Kidney disease (CKD) is an important public health issue. It can be defined as 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 2) [6]



	All modalities		Dialysis		Transplant	
Causes of death	*N*	%	*N*	%	*N*	%
Cardiac disease	722	23	628	24	94	18
Cerebrovascular disease	136	4	112	4	24	5
Infection	622	20	498	19	124	24
Malignancy	350	11	214	8	136	26
Treatment withdrawal	504	16	490	19	14	3
Other	607	19	517	20	90	17
Uncertain	189	6	154	6	35	7
Total	**3,130**		**2,613**		**517**	
No cause of death data	1,564	33	1,313	33	251	33

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

Stage[a]	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 2*: Stages of chronic kidney disease* [7]

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 [6], 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 (RRT). RRT can take a number of forms; kidney transplantation, haemodialysis or peritoneal dialysis. Receiving a kidney transplant can be a very challenging process. According to [9]:

- 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 does not 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.

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 Hemodialysis

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

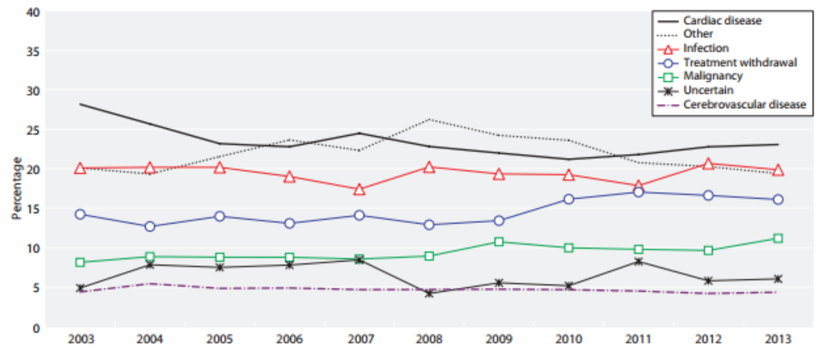


Figure 2*: Cause of death in RRT patients by year*

hours per treatment is essential. For new patients, following this strict schedule is one of the most challenging adjustments needing to be made [10]. 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

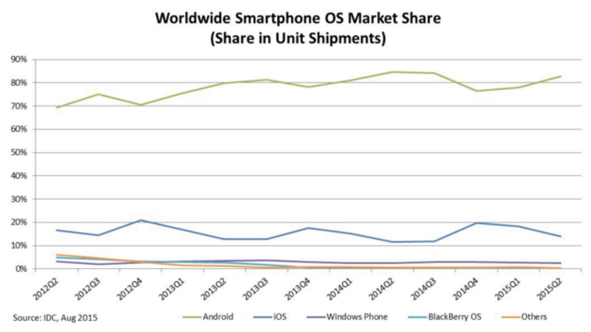
Currently, it is possible to access health check results from desktop computer using Patient View system [11], where registered patients can access their blood check results and view them in a grid or table format. NHS medical staff are using "eMed" system to input results into the system.

2.7 Market Research

The amount of mobile applications available on the market is very limited. According to National Kidney Foundation [12], 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 patients 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 [13] 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 [14], Android OS dominated the market in 2015 with 82.8% share in unit shipments (Figure 3). After taking this into consideration, the decision was made to develop the application for Android OS.



Period	Android	iOS	Windows Phone	BlackBerry OS	Others
2015Q2	82.8%	13.9%	2.6%	0.3%	0.4%
2014Q2	84.8%	11.6%	2.5%	0.5%	0.7%
2013Q2	79.8%	12.9%	3.4%	2.8%	1.2%
2012Q2	69.3%	16.6%	3.1%	4.9%	6.1%

Figure 3: *Worldwide Smartphone OS Market Share in 4 year period (2012-1015) (Share in Unit Shipments) [14]*

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 [15] and Caldicott Guardian ethical approval [16]. 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 had 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 evaluation 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. 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 helpful with medical aspects of the research, CKD and, more importantly, RRT.

3.2 Background

The client had around 60 patients, 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 were considered as an older age group patients. Apart from that, the client also 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 would 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 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 a 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
- 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
- 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 design to the very end of 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 RRT patients
- hand hygiene aspects for RRT patients
- RRT and CKD specific websites with useful information that can be used for the project [17],[18],[19],[20].

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 [21] and 12 principles [22] 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, Phase 2, 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 (Table 3).

Phase	Start	End	Length
0	Nov 2015	Dec 2015	4 weeks
1	Dec 2015	Mar 2016	6 weeks
2	Jan 2016	Apr 2016	6 weeks
3	Mar 2016	Apr 2016	6 weeks
4	Apr 2016	Apr 2016	1 week

***Table 3**: Projected development cycles in timetable*

The project cycle planned for 23 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.

Personas

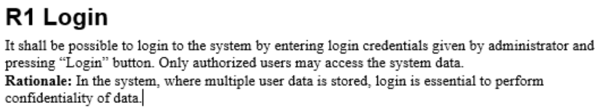
According to [23], the purpose of personas is to create reliable and realistic representations of your key audience segments for reference. These representations should be based on qualitative and some quantitative user research and web analytics. Creating personas would provide a significant benefit 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 H] was created, after the potential benefits personal could offer to the project were taken into consideration.

Use cases

According to [24], 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 I) helped student to negotiate which functions become requirements and are built.

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 J) was created for the project. 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 [25], 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.



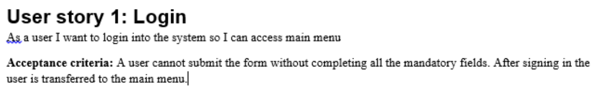
R1 Login

It shall be possible to login to the system by entering login credentials given by administrator and pressing "Login" button. Only authorized users may access the system data.
Rationale: In the system, where multiple user data is stored, login is essential to perform confidentiality of data.

***Figure 4:** 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 [26]. 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 K].



User story 1: Login

As a user I want to login into the system so I can access main menu

Acceptance criteria: A user cannot submit the form without completing all the mandatory fields. After signing in the user is transferred to the main menu.

***Figure 5**: User story*

Project backlog
According to [27], 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 does not work through the backlog at the product owner' space and the product owner is not 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 [28] 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 with progress marked per week. 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 [29], 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 product backlog (Appendix A) 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 developed using Android Studio and can be viewed in (Appendix O).

In order to generate a database schema, 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, project proposal document, user stories, use cases and class diagram were carefully studied. This lead into creation of database schema for the project.

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.

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 document created specially for this purpose (Appendix C).

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 [30], 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 or two-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.

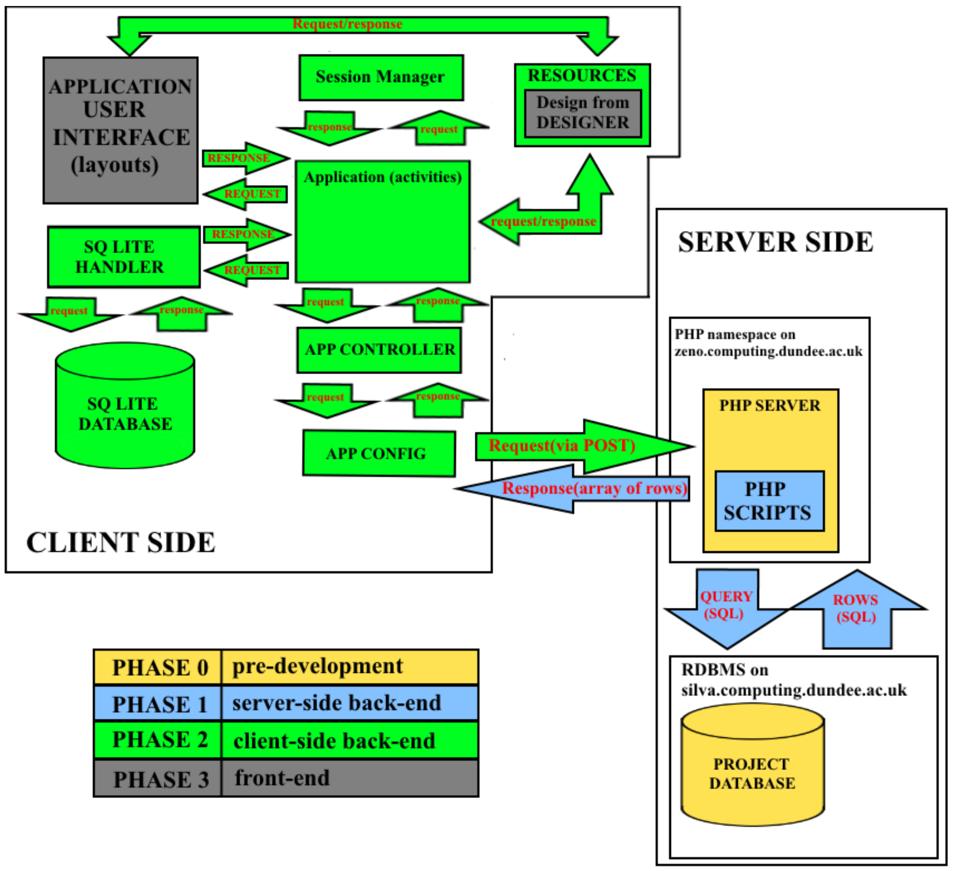
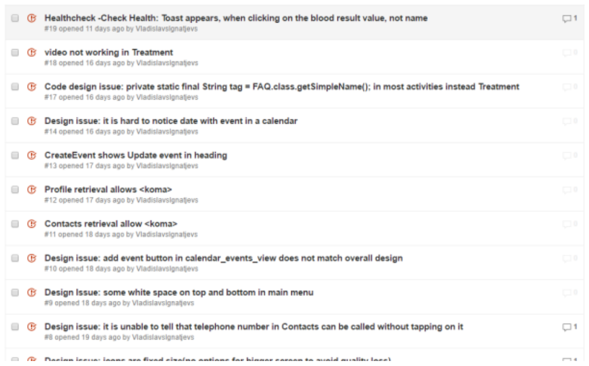


Figure 6: *Expected system architecture including project phase breakdown.*

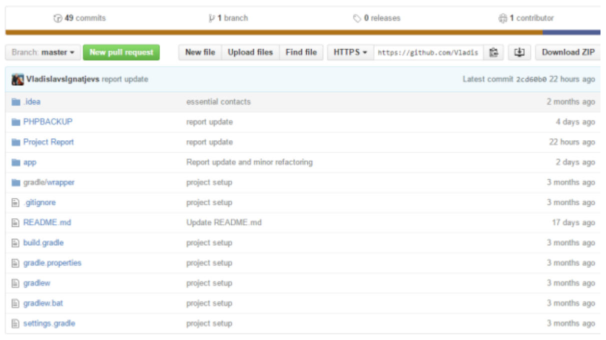
Version control system

According to [31], 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 (Figure 7). 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 [32].



Healthcheck -Check Health: Toast appears, when clicking on the blood result value, not name #19 opened 11 days ago by VladislavsIgnatjevs	1
video not working in Treatment #18 opened 16 days ago by VladislavsIgnatjevs	0
Code design issue: private static final String tag = FAQ.class.getSimpleName(); in most activities instead Treatment #17 opened 16 days ago by VladislavsIgnatjevs	0
Design issue: it is hard to notice date with event in a calendar #14 opened 16 days ago by VladislavsIgnatjevs	0
CreateEvent shows Update event in heading #13 opened 17 days ago by VladislavsIgnatjevs	0
Profile retrieval allows <koma> #12 opened 17 days ago by VladislavsIgnatjevs	0
Contacts retrieval allow <koma> #11 opened 18 days ago by VladislavsIgnatjevs	0
Design issue: add event button in calendar_events_view does not match overall design #10 opened 18 days ago by VladislavsIgnatjevs	0
Design Issue: some white space on top and bottom in main menu #9 opened 18 days ago by VladislavsIgnatjevs	0
Design issue: it is unable to tell that telephone number in Contacts can be called without tapping on it #8 opened 19 days ago by VladislavsIgnatjevs	1
Design issue: icons are fixed size(no options for bigger screen to avoid quality loss)	1

Figure 7: Use of issues on GitHub



49 commits | 1 branch | 0 releases | 1 contributor

Branch: master | New pull request | New file | Upload files | Find file | HTTPS | https://github.com/Vladis | Download ZIP

VladislavsIgnatjevs report update — Latest commit 2cd60b0 22 hours ago

.idea	essential contacts	2 months ago
PHPBACKUP	report update	4 days ago
Project Report	report update	22 hours ago
app	Report update and minor refactoring	2 days ago
gradle/wrapper	project setup	3 months ago
.gitignore	project setup	3 months ago
README.md	Update README.md	17 days ago
build.gradle	project setup	3 months ago
gradle.properties	project setup	3 months ago
gradlew	project setup	3 months ago
gradlew.bat	project setup	3 months ago
settings.gradle	project setup	3 months ago

Figure 8: Github Repository [32]

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 B). 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 [33] 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 RRT 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. An additional 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".

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 project management tools on Pivotal Tracker
7. Set up a project database (MySQL) with data created according to specification provided from the client and NHS staff during the meetings (blood 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:
    - DB constructor and Destructor
    - storeUser() – *function to store user in the database, check for new user to be unique and check for successful store*
    - getUserByEmailAndPassword() – *gets the user form the database by email and password*
    - isUserExisted() – *checks if the user exists or not by email*
    - hashSHA() – *encrypts password using Secure Hash Algorithm 1 (SHA-1) and returns encrypted password and salt*
    - checkhashSha() – *decrypts password from SHA-1 using salt and password*
    - getEssentialContacts() – *gets essential contacts from the database using user id*
    - getProfile() – *gets user profile from database using user id*
    - getFaq() – *gets FaQ(Frequently asked questions and answers) from the database in a loop*
    - getEvents() – *gets events assigned for the user, ordered by event start time descending, using user id in a loop*
    - createEvent() – *adds new event to the database by creating new row with events details, user id and then checks for successful store*
    - 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.*
    - 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
    - change_event.php
    - contacts_request.php
    - create_event.php
    - events_request.php
    - faq_request.php

- index.php
- login.php
- med_history_request.php
- profile_request.php
- register.php

Phase 2: *Back-end at client side*

1. Setting up the project in Android Studio
2. Configuring the project (minimum Android API, target API, 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
    a. Launcher
    b. LoginActivity
    c. RegisterActivity
    d. MainMenu
    e. Treatment
    f. Healthcheck
    g. HealthcheckCheckHealth
    h. Healthchecks_view_graphs
    i. CalendarEvents
    j. calendar_events_view
    k. CreateEvent
    l. ChangeEvent
    m. Contacts
    n. Profile
    o. FAQ
7. Performing white-box testing

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

1. Create layout for all the activities
2. Add resources:
    a. Video for washing hands tutorial
    b. Temporary drawables
3. Add Values:
    a. Colors.xml
    b. Dimens.xml
    c. Strings.xml
    d. Styles.xml
4. Request design from the designer
5. Add ”drawables” created by the designer 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*

1. Arrange meetings with participants, provided by the client (when the client resumes from the leave).
2. Arrange interviews with participants to perform design evaluation
3. Process results and make final changes for the project if needed

4.2 Projected system overview

Following the construction of the product backlog, a system overview diagram was sketched to guide the development process (Figure 6). 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, SQLite database for storing user data locally and SQLite 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 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 data needed for the served-side database was identified. After that, the database was created locally using MariaDB database and PHP interpreter provided with XAMPP [34], 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 9).

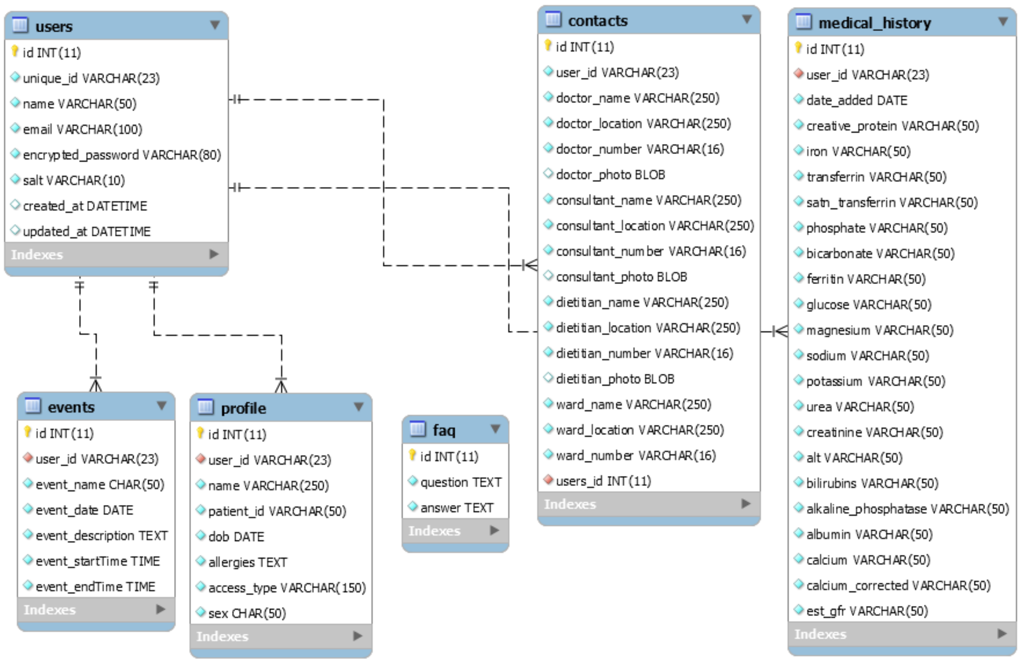


Figure 9: *E-R diagtam for the created database*

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 design evaluation was done. Apart from that, the setup time needed for presenting functionality of the application was significantly reduced.

5.2 Implementation and testing

For server-side database management, it was decided to use HeidiSQL [35], 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*

```
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* *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 and structures. 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.

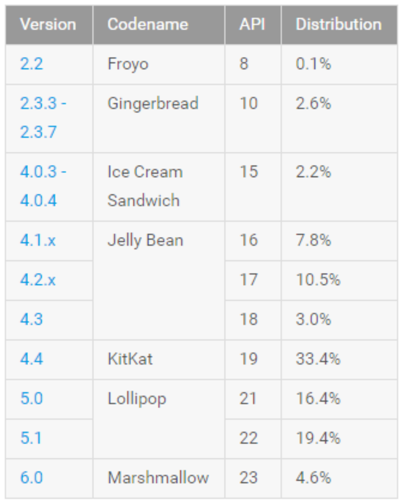
Testing

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 M) 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 [36] [37]. 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 (Figure 10): 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.

.



Version	Codename	API	Distribution
2.2	Froyo	8	0.1%
2.3.3 - 2.3.7	Gingerbread	10	2.6%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	2.2%
4.1.x	Jelly Bean	16	7.8%
4.2.x		17	10.5%
4.3		18	3.0%
4.4	KitKat	19	33.4%
5.0	Lollipop	21	16.4%
5.1		22	19.4%
6.0	Marshmallow	23	4.6%

Figure 10: *Relative number of devices running a given version of the Android platform (data collected using 7-day period ending on April 4, 2016) [37]*

6 Phase 1

6.1 Server-side back-end design

After initial meeting with the client the student made a decision to follow REST architecture (Figure 11) 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 [38].

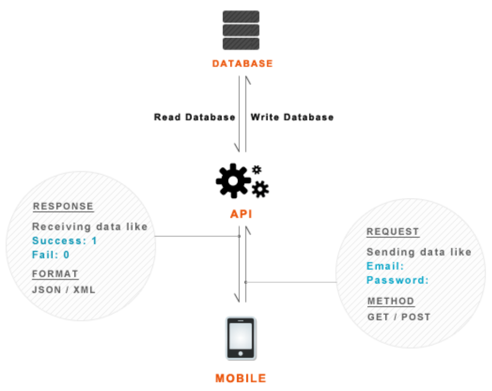


Figure 11: *REST Architecture*

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.

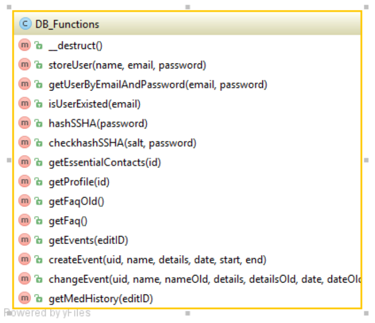

DB_Functions
__destruct()
storeUser(name, email, password)
getUserByEmailAndPassword(email, password)
isUserExisted(email)
hashSSHA(password)
checkhashSSHA(salt, password)
getEssentialContacts(id)
getProfile(id)
getFaqOld()
getFaq()
getEvents(editID)
createEvent(uid, name, details, date, start, end)
changeEvent(uid, name, nameOld, details, detailsOld, date, dateOl
getMedHistory(editID)
Powered by yFiles

Figure 12: *DB_Functions 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 [39].

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.

```
define("DB_HOST",
"silva.computing.dundee.ac.uk");
define("DB_USER", "renaldialysis");
define("DB_PASSWORD", "***********");
define("DB_DATABASE", "renaldialysisdb");
```

Note: *password removed for security reasons*

```
class DB_Connect {
private $conn;
// Connecting to database
public function connect() {
require_once 'include/Config.php';
// Connection
$this->conn = new mysqli(DB_HOST, DB_USER,
DB_PASSWORD, DB_DATABASE);
// returns database handler
return $this->conn;
 }
```

Code Snippets 2 & 3: *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.

```
require_once 'DB_Connect.php';
// connecting to database
$db = new Db_Connect();
$this->conn = $db->connect();
  }
// destructor
function __destruct() {}
/**
```

```
* Storing new user
* returns user details
*/
public function storeUser($name, $email,
$password) {
$uuid = uniqid('', true);
$hash = $this->hashSSHA($password);
$encrypted_password = $hash["encrypted"];
// encrypted password
$salt = $hash["salt"]; // salt
$stmt = $this->conn->prepare("INSERT INTO
users(unique_id, name, email, encrypted_password,
salt, created_at) VALUES(?, ?, ?, ?, ?, NOW())");
$stmt->bind_param("sssss", $uuid, $name, $email,
$encrypted_password, $salt);
$result = $stmt->execute();
$stmt->close();
```

Code Snippet 4: *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 5). Then, user details are inserted into "users" table in database by prepared statement with required variables bind as parameters. (Code Snippet 6) Apart from that, by using email, check for successful store is performed (Code Snippet 7).

```
$uuid = uniqid('', true);
```

Code Snippet 5: *Generating unique ID.*

```
$stmt = $this->conn->prepare("INSERT INTO
users(unique_id, name, email, encrypted_password,
salt, created_at) VALUES(?, ?, ?, ?, ?, NOW())");
$stmt->bind_param("sssss", $uuid, $name, $email,
$encrypted_password, $salt);
$result = $stmt->execute();
$stmt->close();
```

Code Snippet 6: *Inserting user details into database using prepared statement with bind variables as parameters.*

```
if ($result) {
$stmt = $this->conn->prepare("SELECT * FROM users
WHERE email = ?");
$stmt->bind_param("s", $email);
$stmt->execute();
$user = $stmt->get_result()->fetch_assoc();
$stmt->close();
return $user;
} else {
return false;
}
```

Code Snippet 7: *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 8).

```
if ($stmt->num_rows > 0) {
// user existed
$stmt->close();
return true;
} else {
// user not existed
$stmt->close();
return false;
}
```

Code Snippet 8: *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 9). Function returns encrypted password and salt in JSON object.

```
$salt = sha1(rand());
$salt = substr($salt, 0, 10);
$encrypted = base64_encode(sha1($password .
$salt, true) . $salt);
```

Code Snippet 9: *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 10). The function is used for user login.

```
$hash = base64_encode(sha1($password . $salt,
true) . $salt);
```

Code Snippet 10: *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 11).

```
while($response = $result->fetch_assoc())
{
$resp["question$c"] = $response["question"];
$resp["answer$c"]= $response["answer"];
$c++;
}
```

Code Snippet 11: *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.

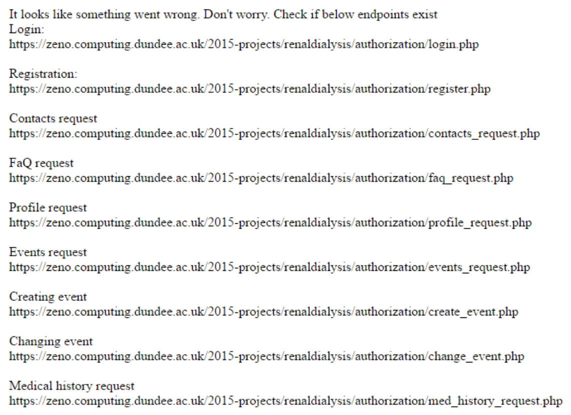
createEvent()
This function is used by create_event.php for inserting new event details into database. Prepared INSERT statement with bind variables (new event details) as parameters is used to insert new event into database. Event is also checked for successful store.

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, during server migration), this page can be used as a reference (Figure 13).



It looks like something went wrong. Don't worry. Check if below endpoints exist
Login:
https://zeno.computing.dundee.ac.uk/2015-projects/renaldialysis/authorization/login.php

Registration:
https://zeno.computing.dundee.ac.uk/2015-projects/renaldialysis/authorization/register.php

Contacts request
https://zeno.computing.dundee.ac.uk/2015-projects/renaldialysis/authorization/contacts_request.php

FaQ request
https://zeno.computing.dundee.ac.uk/2015-projects/renaldialysis/authorization/faq_request.php

Profile request
https://zeno.computing.dundee.ac.uk/2015-projects/renaldialysis/authorization/profile_request.php

Events request
https://zeno.computing.dundee.ac.uk/2015-projects/renaldialysis/authorization/events_request.php

Creating event
https://zeno.computing.dundee.ac.uk/2015-projects/renaldialysis/authorization/create_event.php

Changing event
https://zeno.computing.dundee.ac.uk/2015-projects/renaldialysis/authorization/change_event.php

Medical history request
https://zeno.computing.dundee.ac.uk/2015-projects/renaldialysis/authorization/med_history_request.php

Figure 13: *Guidance for the 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 12, 13).

```
if (isset($_POST['email']) &&
isset($_POST['password'])) {
// receiving the post params
$email = $_POST['email'];
$password = $_POST['password'];
```

Code Snippet 12: *Getting POST parameters*

```
} else {
// user is not found with the credentials
$response["error"] = TRUE;
$response["error_msg"] = "Login credentials are
wrong. Please try again!";
echo json_encode($response);
```

Code Snippet 13: *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 checks 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.

6.2.10 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.

6.2.11 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.

6.2.12 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.

6.2.13 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 14). 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 M) was created.

```
$event = $stmt->get_result()->fetch_assoc();
$stmt->close();
```

Code Snippet 14: *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 6)**.** 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 14).

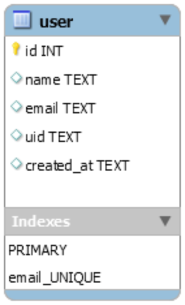


Figure 14: *E-R diagram for the client-side SQLite database*

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 [40]. Apart from that, each Activity in Android has a certain lifecycle. [41]. Class diagram for the client-side application was produced during its implementation (Figure 15).

Figure 15: *Class diagram for client-side application. Full size version available In Appendix L*

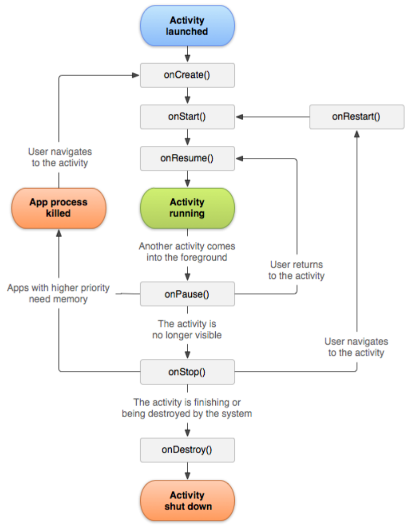

Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

Figure 16: *Android Activity lifecycle*

7.2 Implementation and testing

In order to build login and registration system, Volley library was used [1] along with SQLite. Please note that some of the code used for implementing client-side back-end was taken from [39].

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 (Code Snippet 15). 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.

```
<uses-permission
android:name="android.permission.GET_ACCOUNTS" />
<uses-permission
android:name="android.permission.READ_PROFILE" />
<uses-permission
android:name="android.permission.READ_CONTACTS"
/>
<uses-permission
android:name="android.permission.INTERNET" />
<uses-permission
android:name="android.permission.READ_CALENDAR"
/>
<uses-permission
android:name="android.permission.WRITE_CALENDAR"
/>
<uses-permission
android:name="android.permission.WRITE_EXTERNAL_S
TORAGE" />
<uses-permission
android:name="android.permission.READ_PHONE_STATE
" />
<uses-permission
android:name="android.permission.READ_EXTERNAL_ST
ORAGE" />
```

Code Snippet 15: *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 16).

```
//Server events request url
public static String EVENTS_REQUEST =
"https://zeno.computing.dundee.ac.uk/2015-
projects/renaldialysis/authorization/events_reque
st.php";
//Server create event url
public static String CREATE_EVENT =
"https://zeno.computing.dundee.ac.uk/2015-
projects/renaldialysis/authorization/create_event
.php";
```

Code Snippet 16: *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. For example, AppController was used to add POST request to request queue (Code Snippet 17).

```
AppController.getInstance().addToRequestQueue(str
Req, tag_string_req);
```

Code Snippet 17: *adding POST request to request queue.*

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 18).

```
public void setLogin(boolean isLoggedIn) {
editor.putBoolean(KEY_IS_LOGGED_IN, isLoggedIn);
// commit changes
editor.commit();
Log.d(TAG, "User login session modified!");
    }
public boolean isLoggedIn(){
return pref.getBoolean(KEY_IS_LOGGED_IN, false);
    }
```

Code Snippet 18: *code to check login status (Session Manager.java) [39]*

7.2.5 SQLiteHandler

This class was created to allow storing user data in SQLite database [Code Snippet 19]. In onCreate() method [41], 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 20]. For security reasons, all database rows are deleted when user signs out from the system [Code Snippet 21].

```
public void addUser(String name, String email,
String uid, String created_at) {
SQLiteDatabase db = this.getWritableDatabase();
ContentValues values = new ContentValues();
values.put(KEY_NAME, name); // Name
values.put(KEY_EMAIL, email); // Email
values.put(KEY_UID, uid); // Email
values.put(KEY_CREATED_AT, created_at); //
Created At
// Inserting Row
long id = db.insert(TABLE_USER, null, values);
db.close(); // Closing database connection
```

Code Snippet 19: *storing user data in SQLite database [39]*

```
HashMap<String, String> user = new
HashMap<String, String>();
String selectQuery = "SELECT  * FROM " +
TABLE_USER;
SQLiteDatabase db = this.getReadableDatabase();
Cursor cursor = db.rawQuery(selectQuery, null);
// Move to first row
cursor.moveToFirst();
if (cursor.getCount() > 0) {
user.put("name", cursor.getString(1));
user.put("email", cursor.getString(2));
user.put("uid", cursor.getString(3));
user.put("created_at", cursor.getString(4));
}
cursor.close();
```

Code Snippet 20: *fetching user data from SQLite database [39]*

```

```

```
SQLiteDatabase db = this.getWritableDatabase();
// Delete All Rows
db.delete(TABLE_USER, null, null);
db.close();
```

***Code Snippet 21:** wiping SQLite database [39]*

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 22). 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.

```
// Session manager
session = new
SessionManager(getApplicationContext());
// Check if user is already logged in or not
if (session.isLoggedIn()) {
// User is already logged in. Take him to main
menu
Intent intent = new Intent(LoginActivity.this,
MainMenu.class);
startActivity(intent);
finish();
}
```

***Code Snippet 22:** 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 23).

```
Toast.makeText(getApplicationContext(), "User
successfully registered. Try login now!",
Toast.LENGTH_LONG).show();
```

***Code Snippet 23:** 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 24). Then, videoView is used to allow displaying video (the guide to wash hands properly) in one of the layout parts (Code Snippet 25). 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 26).

```
public void onTick(long millisUntilFinished) {
long millis = millisUntilFinished;
String hms = String.format("%02d:%02d:%02d",
TimeUnit.MILLISECONDS.toHours(millis),
TimeUnit.MILLISECONDS.toMinutes(millis) -
TimeUnit.HOURS.toMinutes(TimeUnit.MILLISECONDS.to
Hours(millis)),

TimeUnit.MILLISECONDS.toSeconds(millis) -
TimeUnit.MINUTES.toSeconds(TimeUnit.MILLISECONDS.
toMinutes(millis)));
System.out.println(hms);
timeLeftValue.setText(hms + " left");
progressStart++;
progressBar.setProgress(progressStart);
```

***Code Snippet 24:** using OnTick() for timer with progress bar bind to it.*

```
//setting videoView
videoView =
(VideoView)findViewById(R.id.washHandsVideo);
Uri video1 = Uri.parse("android.resource://" +
getPackageName() + "/" +
R.raw.hand_washing_converted);
//setting washing hands video
videoView.setVideoURI(video1);
videoView.setMediaController(new
MediaController(this));
videoView.requestFocus();
```

***Code Snippet 25:** setting up videoView for playing video file in the application. Using getPackageName() to avoid hardcoded URIs.*

```
// get html strings
String fluid_text_html =
getString(R.string.treatment_prep_fluid_text_html
);
String food_text_html =
getString(R.string.treatment_prep_food_text_html)
;
//making spanned for displaying as html for those
strings
```

```
Spanned fluid_text_spanned =
Html.fromHtml(fluid_text_html); // used by
textViewFluidHtml
Spanned food_text_spanned =
Html.fromHtml(food_text_html); // used by
textViewFoodHtml
```

Code Snippet 26: *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 (Figure 17).

```
//raw output
{
    "error": false,
    "data": {
        "user_id":"56f14527e91210.87913842",
        "date_added":"2015-04-11",
        "creactive_protein":"14",
        "iron":"113"
    }
}
//cleaned output
user_id":"56f14527e91210.87913842,
date_added":"2015-04-11,
creactive_protein":"14,
iron":"113
```

Figure 17: *raw and cleaned JSON output*

After that, the data gets put into hash map for easy access (Code Snippet 27). The health gets checked by comparing blood test result values with the normal reading, received from NHS medical staff (Code Snippet 28). 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 29).

```
String[] pairs = savedResponse.split("\",\"");
for (int i = 0; i < pairs.length; i++) {
String pair = pairs[i];
String[] keyValue = pair.split("\":\"");
Log.d(tag, "keyvalue 0: " + keyValue[0] + "
keyvalue1 " + keyValue[1]);
med_history_map.put(keyValue[0], keyValue[1]);
```

Code Snippet 27: *Splitting cleaned response string into pairs and storing in hash map.*

```
alkaline_phosphataseValue =
med_history_map.get("alkaline_phosphatase" +
entryCount);
if (Double.parseDouble(alkaline_phosphataseValue)
> 130 ||
Double.parseDouble(alkaline_phosphataseValue) <
30)
//bold red
{
alkaline_phosphatase.setText(alkaline_phosphatase
Value + "U/L");
alkaline_phosphatase.setTypeface(Typeface.create(
alkaline_phosphatase.getTypeface(),
Typeface.BOLD));
```

Code Snippet 28: *Checking blood test results with normal values received from NHS staff.*

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

Code Snippet 29: *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 30) . 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.

```
Intent eventsView = new
Intent(CalendarEvents.this,
calendar_events_view.class);
eventsView.putExtra("date", date_month_year);

CalendarEvents.this.startActivity(eventsView);
```

Code Snippet 30: *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.

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

Code Snippet 31: *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 32).

```
OnTimeSetListener endTimeCallBack = new
OnTimeSetListener() {
public void onTimeSet(TimePicker view, int
hourOfDay, int minute) {
//adding millis for db format
dbFormatEndTime = (hourOfDay+":"+minute+":00");
//convert time to format that is best for UK
try {
SimpleDateFormat input = new
SimpleDateFormat("HH:mm:ss");
Date dt = input.parse(dbFormatEndTime);
SimpleDateFormat output = new
SimpleDateFormat("h:mm a");
String formattedTime = output.format(dt);
calendarFormatEndTime = formattedTime;
              } catch (ParseException e) {}
   eventEndTime.setText(calendarFormatEndTime);
}
```

Code Snippet 32: *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

This activity allows to see the list of patient's essential contacts. These contacts are requested by POST using requestEssentialContacts() method. Then, all the contacts and their details in a table (Name, number, location). It is possible to call essential contacts by pressing on the number (Code Snippet 33).

```
consultantsNumber.setOnClickListener(new
View.OnClickListener() {
@Override
public void onClick(View v) {
Intent caller = new Intent(Intent.ACTION_DIAL);

caller.setData(Uri.parse("tel:"+consNumber));
startActivity(caller);
}
});
```

Code Snippet 33: *Starting Caller Activity when table cell with a number is pressed.*

7.3.8 Profile

This activity was created to display user's personal details. Profile is requested from the server side by POST in requestProfile() method. The following personal details are added in the profile table: patients' name, date of birth, allergies (if any), dialysis access type and patient's hospital ID. The purpose of this profile is to use it as a reference, when filling "dialysis sheets" by patients. Using it as an "emergency medical card" can be another use of the profile.

7.3.9 FAQ

This activity was created to display the list of frequently asked questions and answers to these questions. This data is requested from the server side by POST and in requestFaq() method. In order to present questions in the correct order, qCount variable is used (Code Snippet 34).

```
for(int a=0; a<qCountInt; a++)
{
//getting questions from hashmap
quest = faq.get("question"+questionCount);
ans = faq.get("answer"+questionCount);
```

Code Snippet 34: *Using qCount to get questions from the hash map in a right order.*

7.4 Testing

In order to test client-side application, white-box testing was performed. Test plan was created according to procedure of white-box testing. In order to help with code testing, Log Cat messages were used along with TAG String (to indicate the Activity the output came from) (Code Snippet 35). The main use-cases were: making sure that the code compiles and builds with no errors, output from methods is correct, isLoggedIn() is used in every activity to check if the user is signed in or and if the user is not signed in, logoutUser() is called to wipe client-side database, set isLoggedIn flag to false and launch LoginActivity.

```
private static final String TAG =
calendar_events_view.class.getSimpleName();
Log.d(TAG, "key: " + entry.getKey() + " value " +
entry.getValue());
```

Code Snippet 35: *using Log with TAG for debugging.*

8 Phase 3

8.1 Client-side User Interface design

As mentioned earlier, it was decided to recruit designer to create the design for the user-interface (UI). Multiple meetings with designer were held to ensure the suitability of design. This meant, that user interface should be relatively easy to understand by the older people due to age group of potential users. In order to help the designer, application screens were sketched by the student using the pen and A4 size paper. This was done to show the designer approximate idea of how the application should look like (Figure 18). Followed by discussion of every sketch, they were given to the designer along with software requirements document, which was provided to give an inside look into functionality of the application. Apart from that, the following guidelines were provided during the meeting:

- Designer should follow Flat Design genre when creating user interface
- Application design should have as much icons as possible to make sure that users with minimal computer skills can use it
- Colors used in application should be moderate to allow long application use during treatments. Apart from that, there should be a good use of white background.
- Each Activity should have an action bar
- Main menu should be vertical with every item to have different background color to allow easier navigation

The designer had a complete freedom over the application design apart from these guidelines. There were multiple ways of how the application would benefit from this decision. First of all, the designer was studying Digital Interaction Design course, which meant that that designer was possible to meet the usability standards for the created application design. Then, the application would look more professional, since the designer had background experience in sketching medical applications and Android applications in overall.

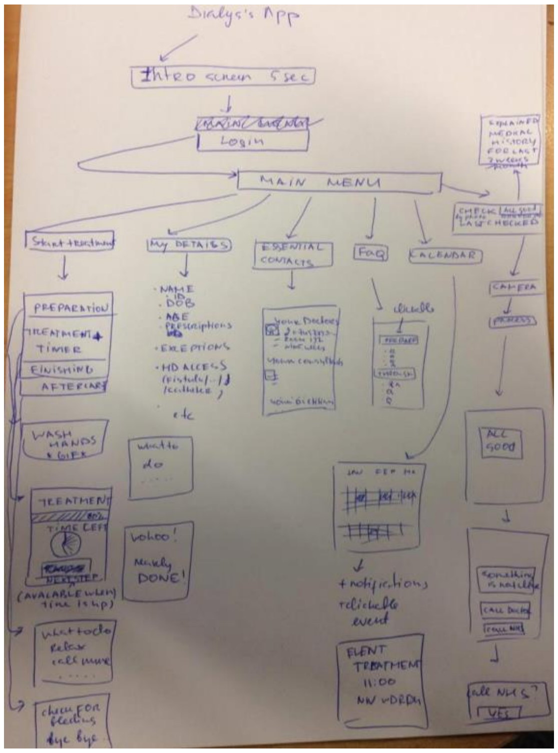
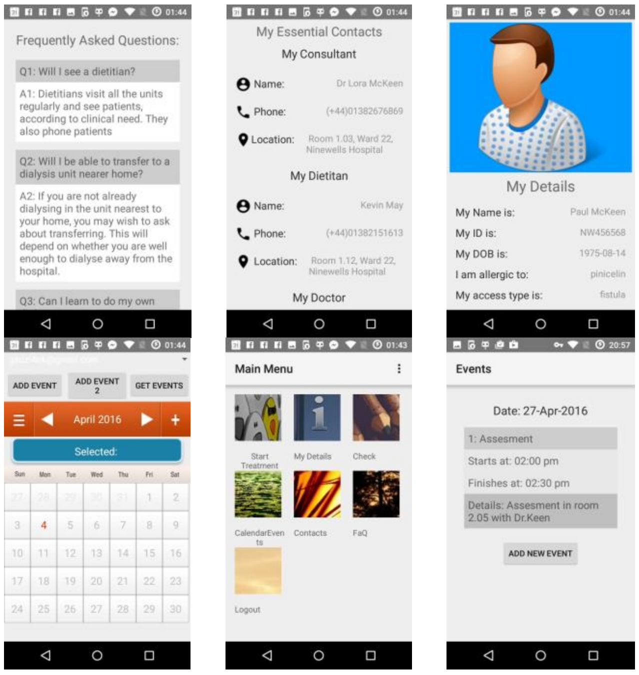


Figure 18: *Sketches of expected application screens with primitive flow chart made by student for the designer to give an idea of expected functionality.*

Meetings were usually followed by discussions by email/phone or on social media. It was ensured that every question raised by the designer was answered on time and all the additional information was provided. The rest of this section describes critical decisions that were made during design implementation phase.

Basic application design
In order to help during application implementation and testing, the basic design was created by the student during Phase 2 of the project (Figure 19).


Frequently Asked Questions:

Q1: Will I see a dietitian?

A1: Dietitians visit all the units regularly and see patients, according to clinical need. They also phone patients

Q2: Will I be able to transfer to a dialysis unit nearer home?

A2: If you are not already dialysing in the unit nearest to your home, you may wish to ask about transferring. This will depend on whether you are well enough to dialyse away from the hospital.

Q3: Can I learn to do my own

My Essential Contacts

My Consultant

Name: Dr Lora McKeen
Phone: (+44)01382676869
Location: Room 1.03, Ward 22, Ninewells Hospital

My Dietitan

Name: Kevin May
Phone: (+44)01382151613
Location: Room 1.12, Ward 22, Ninewells Hospital

My Doctor

My Details

My Name is: Paul McKeen
My ID is: NW456568
My DOB is: 1975-08-14
I am allergic to: pinicelin
My access type is: fistula

ADD EVENT | ADD EVENT 2 | GET EVENTS

April 2016

Selected:

Sun	Mon	Tue	Wed	Thu	Fri	Sat
27	28	29	30	31	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

Main Menu

Start Treatment | My Details | Check | CalendarEvents | Contacts | FaQ | Logout

Events

Date: 27-Apr-2016

1: Assesment
Starts at: 02:00 pm
Finishes at: 02:30 pm
Details: Assesment in room 2.05 with Dr.Keen

ADD NEW EVENT

Figure 19: *Examples of basic design created by the student.*

Early design sketches

Followed by early meetings with the designer, first sketches of the design were made and presented to the student.

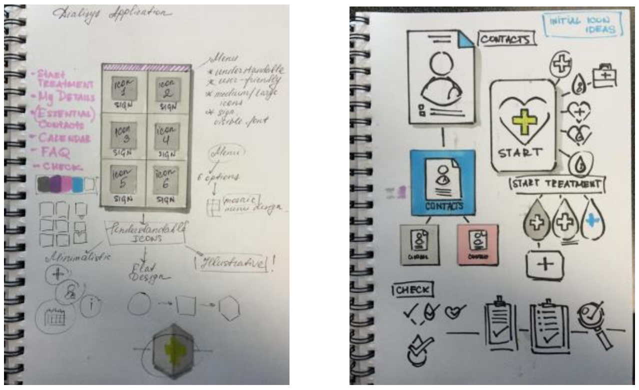


Figure 20: *First design sketches from the designer.*

1st design

Followed by the meeting, where first design sketches were presented, first design prototype was created (Figure 21). After multiple discussions, it was decided to change the design completely. This decision was offered by the designer due to her concerns about older patients being unable to understand certain parts of the created user interface. This lead into creating the second prototype (Figure 22). After discussion with the designer which lead into changing colors and some specific design elements, it was decided to accept second design. The following things were changed in the second prototype:

- Sign out button was added to the main menu
- Blue colour was set to be dominating along with white, to add more positive notes into the user experince

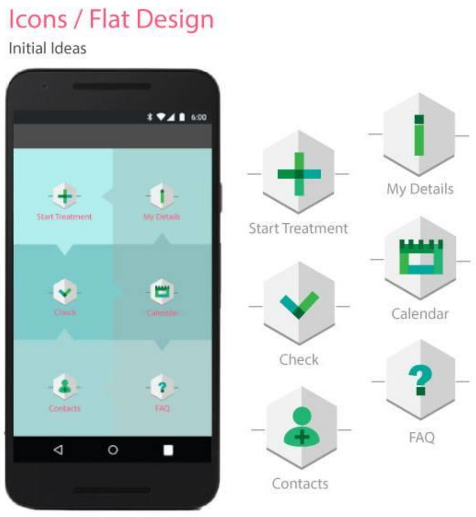


Figure 21: *First design prototype*

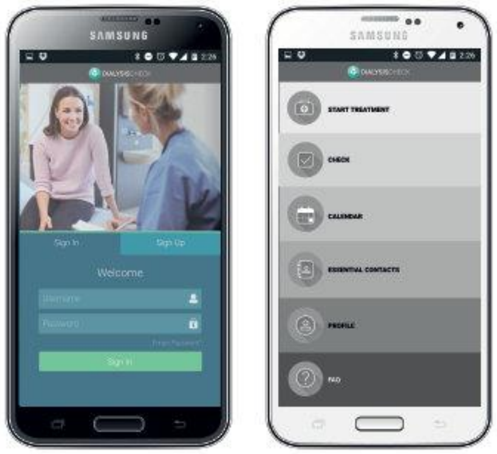


Figure 22: *Second design prototype*

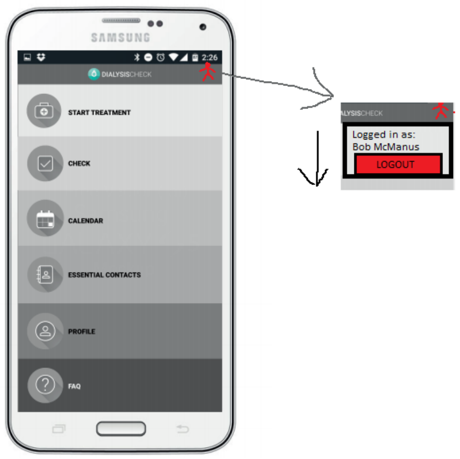


Figure 23: *Student sketch created during email conversation with designer to show one of the possible implementations of "Sign Out button"*

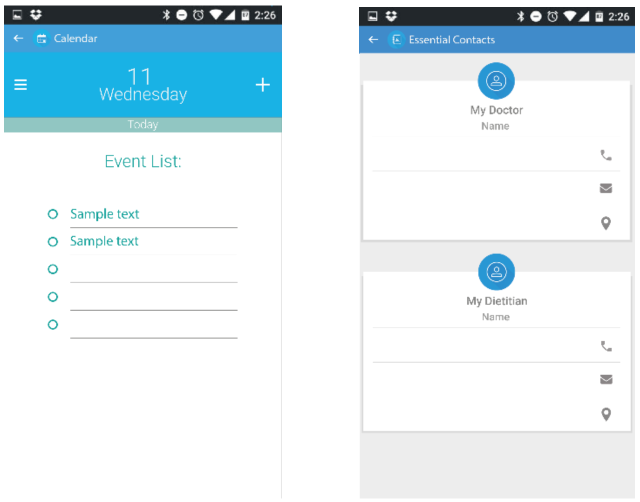


Figure 26: *Example of calendar_events_view and Contacts final design prototype*

Final design

The final design was provided in a zip file. It consisted of cut to size PNG images that were ready to be integrated into application. Full list of final design prototypes is available in (Appendix N).

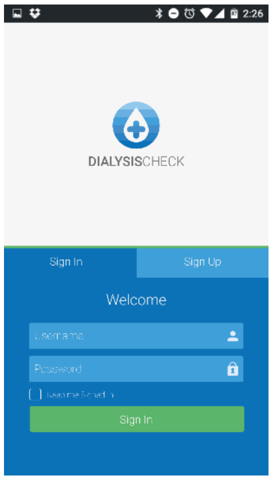


Figure 27: *Login screen final design prototype*

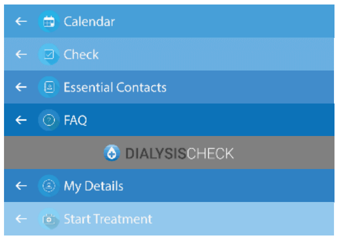


Figure 24: *Example of action bars final design prototype*

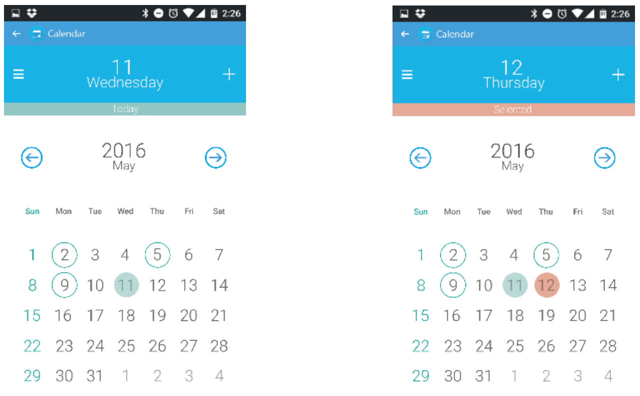


Figure 25: *Example of events calendar final design prototype*

Figure 28: *Application icon*

Design integration

Minor design elements were changed by the student during design integration due to implementation complications. The following elements were changed:

- In login screen, it was decided to place registration button under the login button
- "Remember me" checkbox was removed

- Email address line was removed in profile
- FaQ and events calendar activities were slightly changed.

Design integration in Android Studio

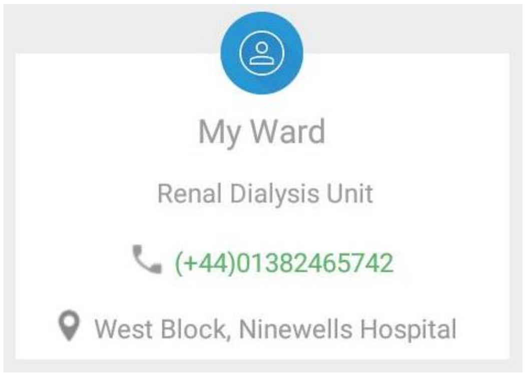
In Android, layout is implemented using layout XML vocabulary with each layout file having one root View or ViewGroup object element and additional layout objects as child elements. RelativeLayout was always used as a root element. However, implementation of some design ideas was very challenging and required additional research about available Android layout objects. For example, to allow certain UI elements to overlap each other in Contacts Activity, FrameLayout object was used. Apart from that, ScrollView was used to allow scrolling through the UI in certain activities. In order to switch between views in Treatment activity, ViewFlipper was used (Code Snippet 36). Good use of strings.xml file was provided to avoid hardcoded texts on a layout. The user interface would benefit from this decision in the future, when the client would require to change text in certain UI elements.

```
<ScrollView
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:gravity="center"
android:layout_marginTop="32dp">
```

Code Snippet 35: *Using scroll view*

```
<ViewFlipper
android:id="@+id/viewFlipper"
android:layout_width="fill_parent"
android:layout_height="fill_parent">
<!-- Intro part of treatment -->
startPreparation.setOnClickListener(new
View.OnClickListener() {
public void onClick(View view) {
vFlip.showNext();
} });
```

Code Snippet 36: *Using ViewFlipper to flip layout when next step button is pressed in Treatment Activity.*


My Ward

Renal Dialysis Unit

(+44)01382465742

West Block, Ninewells Hospital

Figure 29: *Example of overlapping layouts ("profile icon" ImageView overlaps TableLayout.*

```
//XML
<string
name="healthcheck_phosphate">Phosphate:</string>
//Code in activity
phosphate = (TextView)
this.findViewById(R.id.healthcheck_phosphate);
```

Code Snippet 37: *Example of using strings.xml*

Final application look

Followed by design integration, the following UI was produced for the application (please note that some of the application screens are not shown, see Appendix N for full UI).

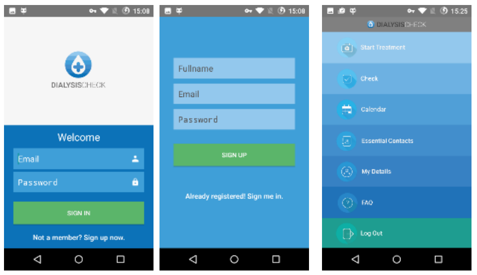


Figure 30: *Login/Registration screen and MainMenu*

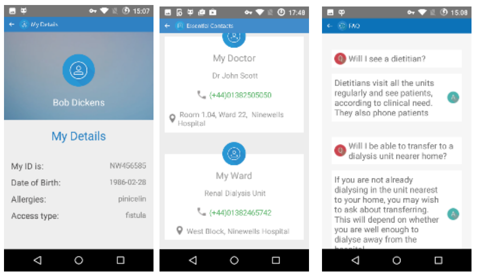


Figure 31: *Profile, Essential Contacts and Frequently Asked Questions*

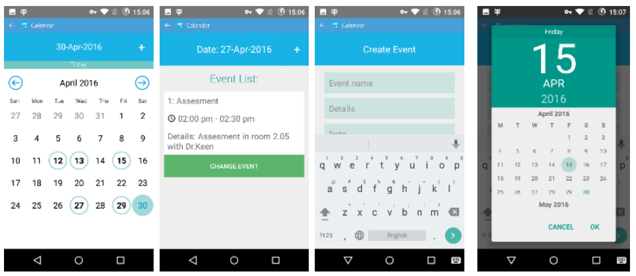


Figure 32: *Events Calendar, Events list for the date, Creating events, setting event date using DatePicker*

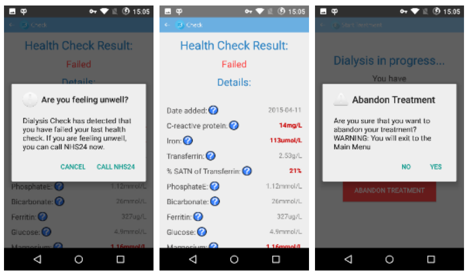
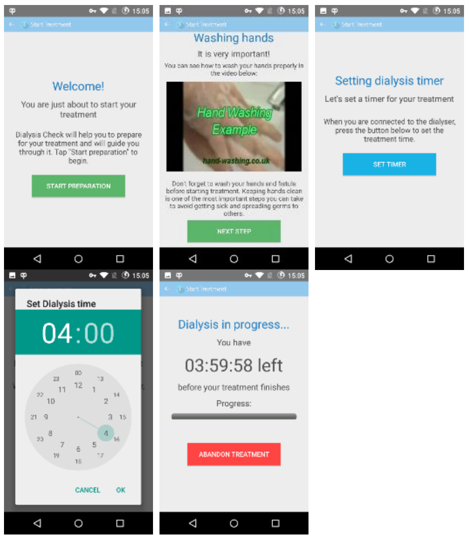


Figure 33: *Healtcheck with option to call NHS 24 if check fails*



Welcome!

You are just about to start your treatment

Dialysis Check will help you to prepare for your treatment and will guide you through it. Tap "Start preparation" to begin.

Washing hands

It is very important!

You can see how to wash your hands properly in the video below:

Don't forget to wash your hands and fistule before starting treatment. Keeping hands clean is one of the most important steps you can take to avoid getting sick and spreading germs to others.

Setting dialysis timer

Let's set a timer for your treatment

When you are connected to the dialyser, press the button below to set the treatment time.

Set Dialysis time

04:00

Dialysis in progress...

You have

03:59:58 left

before your treatment finishes

Figure 34: *Treatment (different stages)*

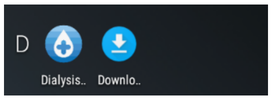


Figure 35: *Application look in Android main menu*

8.2 Design evaluation

In order to evaluate the design, design evaluation was carried out. Originally, it was planned to involve real patients into the process of design evaluation, but unfortunately, the client did not provide enough participants design evaluation. This happened due to the fact that the client resigned from the project for uncertain time. When the client returned to the project, it was too late to follow initial plan of design evaluation. However, the student managed to recruit final year computing students and NHS staff for design evaluation.

8.2.1 Participants

6 participants were recruited for design evaluation, 5 male and 1 female with an age range from 22 to 40 years of age. 5 of the participants were Dundee University School of Computing final year students, who had different knowledge about dialysis. They represented RRT patients with different patient experience. 1 of the participants was Renal Dialysis Unit staff nurse, who was considered as an expert, due to her extensive experience about the process of RRT. The evaluation was done in 3 stages, one participant at the time. In stage one, participants were given a project brief. Then, in stage two, participants were asked to use the application to carry out simple tasks. Finally, in stage 3, participants were asked to rate their user experince using system usability scale (SUS) [42]. After the design evaluation, participants were offered an opportunity to get research results by email.

Tasks

- Please login into the application using provided credentials
- Please locate events calendar and find the most recent event
- Please find the date of next treatment
- Please add new event to events calendar
- Please find the was back to apllication Home screen
- Please find the name, telephone number, location of dietitian in "Essential Contacts"
- Please call the dietitian from the application
- Please perform a health check and find the readings that failed it.

8.2.2 SUS

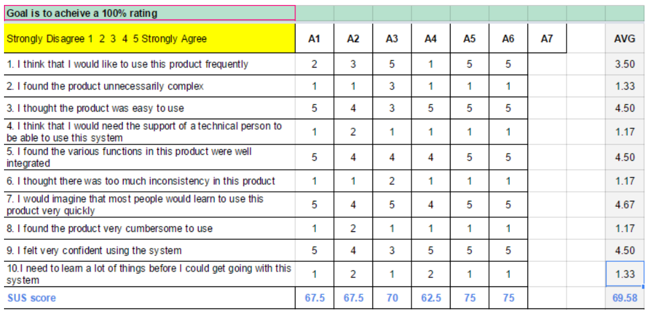
Standard system usability scale with Likert scale ranging from 1 to 5; 1 for "Strongly Disagree" and 5 for "Strongy Agree" was used for design evaluation

8.2.3 Analysis

For SUS questionnaires, guidlines from [42] were used to analyse the results. The participants' scores for each question were converted to a new number, added together and then multiplied by 2.5 to convert to original scores of 0-40 to 0-100 as the guidelines suggest. Score above 68 was considered above average and the score below 68 was considered below average. The results were calculating using the online SUS calculator [43].

8.2.4 Design evaluation results

Table 4 shows calculated SUS score, while Figure 36 shows the results in a chart. The usability tested for client-side application scored 69.58 points. According to slightly extended SUS scale by "Try My UI"[44], SUS with the result of 69.58 can be considered as a good result. Since real patients were not involved in a design evaluation process, it is hard to tell about results accuracy. The application could benefit from resetting design evaluation with real RRT patients involved in the process, as it was planned initially.



Goal is to acheive a 100% rating									
Strongly Disagree 1 2 3 4 5 Strongly Agree	A1	A2	A3	A4	A5	A6	A7		AVG
1. I think that I would like to use this product frequently	2	3	5	1	5	5			3.50
2. I found the product unnecessarily complex	1	1	3	1	1	1			1.33
3. I thought the product was easy to use	5	4	3	5	5	5			4.50
4. I think that I would need the support of a technical person to be able to use this system	1	2	1	1	1	1			1.17
5. I found the various functions in this product were well integrated	5	4	4	4	5	5			4.50
6. I thought there was too much inconsistency in this product	1	1	2	1	1	1			1.17
7. I would imagine that most people would learn to use this product very quickly	5	4	5	4	5	5			4.67
8. I found the product very cumbersome to use	1	2	1	1	1	1			1.17
9. I felt very confident using the system	5	4	3	5	5	5			4.50
10.I need to learn a lot of things before I could get going with this system	1	2	1	2	1	1			1.33
SUS score	67.5	67.5	70	62.5	75	75			69.58

Table 4: *SUS results in a table*

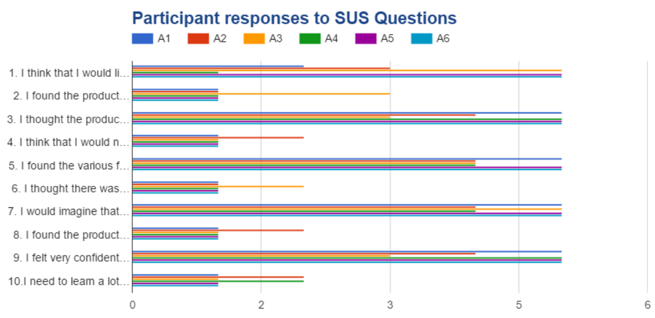


Figure 36: SUS results in a chart

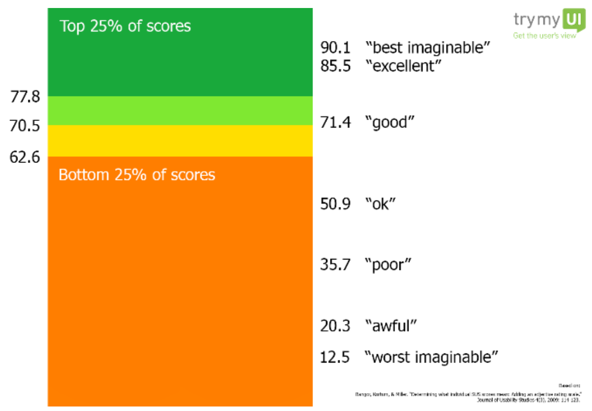


Figure 37: SUS scale by Try My UI [44]

9 Final Product Description

9.1 Mobile application

The client-side mobile application was developed using Android Development studio version 5.1. At the moment, the application provides an impressive list of features that can assist RRT patients in their day-to-day routine. Moreover, it provides patients with easy to use, but powerful tool to check their health at any time. In daily life, blood test result analysis can be very challenging for RRT patients, since some of the blood tests will always show higher values, than they are supposed to show. In such tools as PatientView, it is not possible to specify patient's diagnosys so that the value scope would adjust slightly. As the result, RRT patient will always fail certain blood tests. It would not matter if the result should be considered as normal or not, it will always indicate that there is a problem, which can cause additional confusion, or danger in the most unlikely event. For the patients, it can be very easy to get used to some of the blood test results appearing in "bold" on a result sheet. At the same time, it is really easy to get in a habit of not paying enough attention to the certain place on that result sheet, since the result will always fail anyway. Finally, in the most unlikely event, this can lead into patient's life being in risk. In the application, developed by the student, this aspect was discussed with medical staff before the implementation and as the result, certain blood tests were set for higher normal values to suit RRT patients' results. This means, that the health check will fail only if the results will get out of the scope designed for the RRT patients' blood test results. Apart from that, as both the client and medical staff mentioned during the meetings, starting on dialysis is very challenging for every new patient due to nutritional limitations and long lasting treatments. Treatment assistand that is included in the application, can potentially help the new patient to get used to RRT. This was one of the main reasons to create treatment assistant as such and include treatment preparation steps with handwashing, nutritional, fluid intake guides.

9.2 List of features included in the final product

9.2.1 Authentication and registration

It is required for the users to have a personal account in order to use the application. The account can be created easily, since only three fieds are required for registration. If the user decides to sign in with existing account, the welcome screen is shown as in Figure 38.

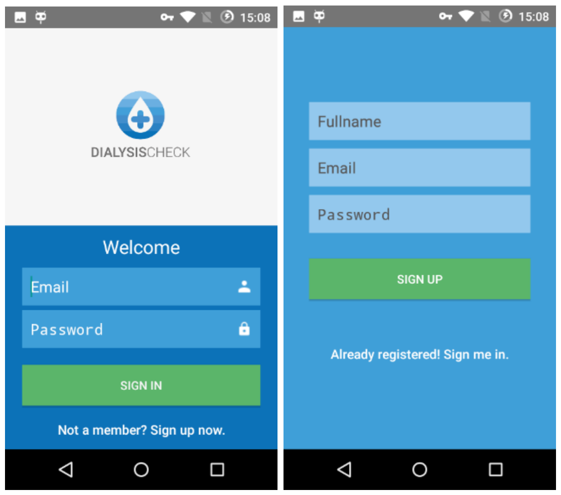
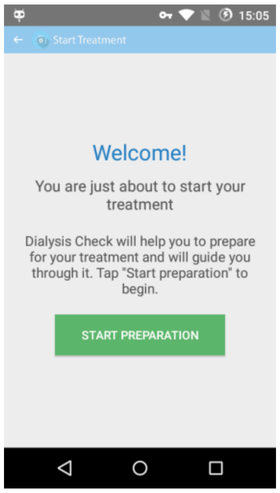


Figure 38: Login and Registration screens

9.2.2 Treatment Assistant

Treatment assistant prepares RRT patients for their treatment, helps them to get through it and helps in their aftercare (Figure 39).



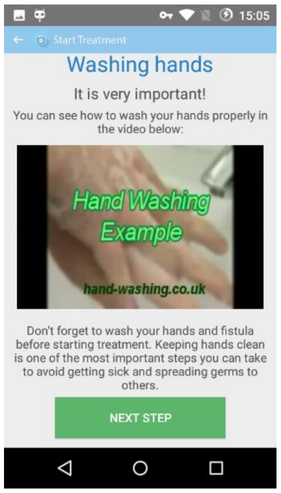
Start Treatment

Welcome!

You are just about to start your treatment

Dialysis Check will help you to prepare for your treatment and will guide you through it. Tap "Start preparation" to begin.

START PREPARATION



Start Treatment

Washing hands

It is very important!

You can see how to wash your hands properly in the video below:

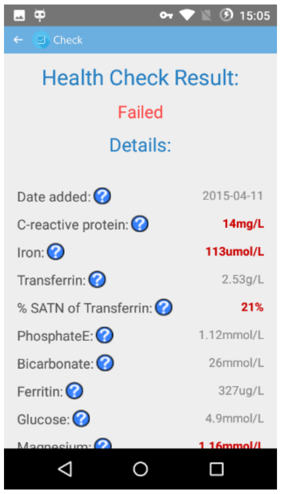
Don't forget to wash your hands and fistula before starting treatment. Keeping hands clean is one of the most important steps you can take to avoid getting sick and spreading germs to others.

NEXT STEP

Figure 39: *Treatment assistant*

9.2.3 Healthcheck

Healthcheck is designed to suit RRT patients blood results with ability to call NHS24 from application if required. It is possible to get an information about eny result by tapping a it. This produced the toast with information about the certain blood test. Question mark icons are used as an indication, that there is a possibility to get a guidance (Figure 40).



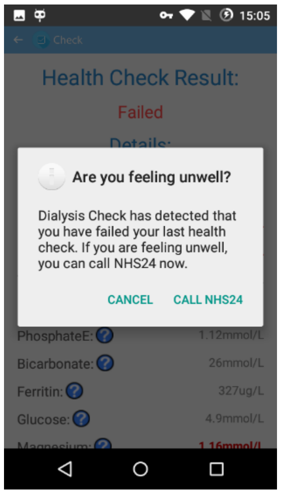
Check

Health Check Result:

Failed

Details:

Date added:	2015-04-11
C-reactive protein:	14mg/L
Iron:	113umol/L
Transferrin:	2.53g/L
% SATN of Transferrin:	21%
PhosphateE:	1.12mmol/L
Bicarbonate:	26mmol/L
Ferritin:	327ug/L
Glucose:	4.9mmol/L
Magnesium:	1.16mmol/L



Are you feeling unwell?

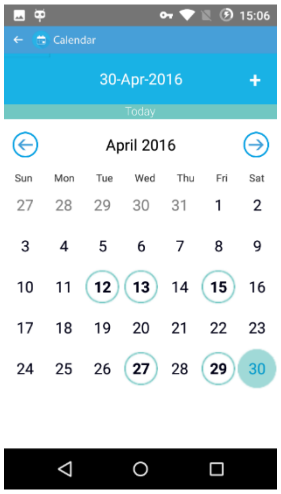
Dialysis Check has detected that you have failed your last health check. If you are feeling unwell, you can call NHS24 now.

CANCEL CALL NHS24

Figure 40: *Healthcheck*

9.2.4 Events Calendar

Events Calendar shows events created for the patient with an option to add or update personal events (Figure 41).



Calendar

30-Apr-2016

Today

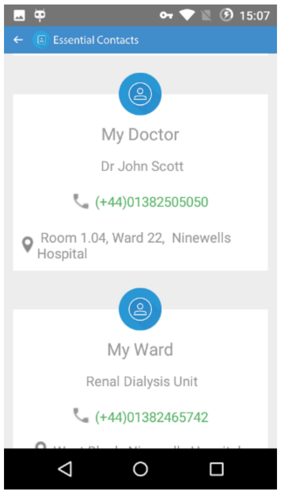
April 2016

Sun	Mon	Tue	Wed	Thu	Fri	Sat
27	28	29	30	31	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

Figure 41: *Events Calendar*

9.2.5 Essential Contacts

Essential Contacts is an easy way for the patients to get name, phone number and location of their consultant, doctor, dietitian and ward with an option to call the phone number by tapping on it (Figure 42).



Essential Contacts

My Doctor

Dr John Scott

(+44)01382505050

Room 1.04, Ward 22, Ninewells Hospital

My Ward

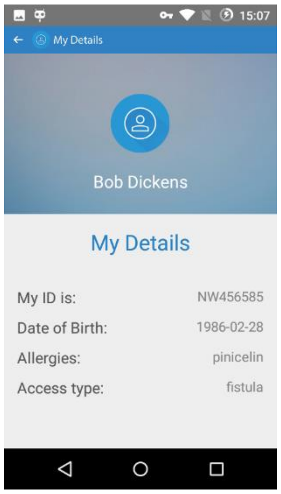
Renal Dialysis Unit

(+44)01382465742

Figure 42: *Essential Contacts*

9.2.6 My Details

My details is a personal profile with information about patient's name, hospital ID, allergies and access type for dialysis (Figure 43).



Bob Dickens

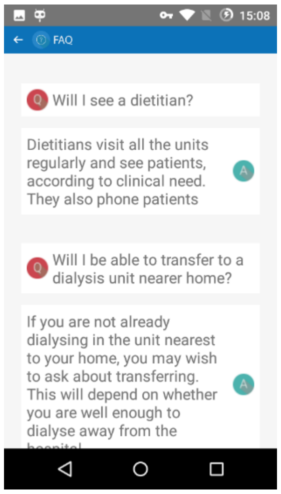
My Details

My ID is:	NW456585
Date of Birth:	1986-02-28
Allergies:	pinicelin
Access type:	fistula

Figure 43: *My Details*

9.2.7 FAQ

FAQ is the list of the most commonly asked questions about RRT with answers to these questions (Figure 44).



Will I see a dietitian?

Dietitians visit all the units regularly and see patients, according to clinical need. They also phone patients

Will I be able to transfer to a dialysis unit nearer home?

If you are not already dialysing in the unit nearest to your home, you may wish to ask about transferring. This will depend on whether you are well enough to dialyse away from the

Figure 44: *FAQ*

9.3 Requirements for starting the system

9.3.1 Client Side

With an appropriate setup the application can work on any Android phone that runs Android Jellybean 4.1 or any newer Android OS version. Since at the moment project API is deployed on Dundee University School of Computing servers, it is essential to set up Virtual Private Network (VPN) access to the School of Computing internal network for the application to work. Alternatively, API can be deployed on any other PHP server with minor changes made to endpoints in AppConfig class on client side Config.php on server-side. Similarly, if the database is deployed on own MySQL server, database connection configuration must be added in Config.php on server side.

9.4 API

For server-side, the variety of PHP scripts were written to accept requests in GET/POST methods, interact with database by inserting/fetching data from MySQL database and give response back in JSON. This was done in attempt to form a REST API.

9.5 Database

MySQL database was created to store the data required for the system to work. The database consists of 6 tables, each serving for its own purpose. The following tables were created:

- contacts – to store essential contacts' details for each user
- events – to store users' calendar events
- faq – to store frequently asked questions and answers
- medical_history – to store users' medical history
- profile – to store profile data for each user
- users – to store users' login data (please note that encrypted password and salt are stored instead of plain password for security reasons)

10 Summary and conclusions

The final product created during the project met the requirements, clients' needs and student's goals. The application runs smoothly and has the design, that was evaluated by users. The design achieved impressive results, when the usability was measured using System Usability Scale. Created system can be easily integrated into existing system, that has PHP and MySQL server. Such setup can be considered as very common due to its popularity, therefore the integration process should be ralatively easy. Apart from that, the market research has shown that such an application does not exist on the market yet. Due to availability of such tools as treatment assistant and health check that were specially designed for RRT patients, it is possible to assume that the health and daily routine of RRT patients across the world will gradually improve, when the application is released on the market. The project was very challenging for the student at certain stages of implementation, but seeing the system gradually improving and finally, working as planned, was highly motivating.

Mistakes

There were certain mistakes the student made during the project. Thease mistakes led into the project running not as smoothly as it was initially planeed. When the client left the project, there were no signs of the client willing to resume until the last weeks of the project. The mistake was made, when the student decided to wait for the client to resume in order to perform design evaluation. As the result, the design evaluation was moved to last days of the project. Then, decision to become a product owner could be made much earlier, than it happened. There was a possibility making these mistakes by creating a risk assement document on early stages of the project, where such risks could be predicted with the backup strategy defined.

Future Work

The project can benefit from the future improvements. As mentioned earlier, the created system can be added as an extension to other existing system, if the certain requirements are met. The system in current state still requires a tool to automatically insert results into database. In the best case scenario, the system will be integrated into NHS eMed or PatientView system to serve as a unique tool, designed specially for RRT patients. In this way, with additional security implementations, the system will be able to fetch patients blood test results as soon as they are submitted by the medical labarotory staff. At the same time, medical appointments for the patient could be fetched from the other database, as soon as they are added by the clinical staff. Of course, the system can be integrated into client’s new website, when it is finished, but it will require to recruit additional person, who will be responsible for collecting results from the hospital and adding them to the database using the website. Also, the system would benefit from adding a feature, that would show medical history in graphs, serving as a tool for clear visualisation of how the blood test results changed over time. Additionally, the system would benefit from adding “fistula check”, the feature that was described in interim report (Appendix D), but not implemented in the final system.

Acknowledgements

The author of this paper would like to thank the supervisor of the this project, Dr. Keith Edwards for providing support, feedback, advice and suggestions of improvement. The author also wants to thank Anna Morozova, the UI designer of the project who contributed large amount of her spare time to application UI design development. Additionally, the author wants to thank Lesley Adam, Donna Mcintosh, Kevin McGinty and Audrey McHugh for helping with understanding the process of RRT and providing advices from medical point of view. The author also wants to thank university teaching staff who have helped the author to develop the skills and experience required for this project

The author appreciates the help of the users who participated in the design evaluation process and also wants to thank all School of Computing staff, especially Anne Millar for giving positive notes during hard times and Mr Mahamadou Niakate setting up some of the system environment.

Finally, the author would like to say a special thanks to Anna Solovjova and Daniel Lennart for constant motivation and support during development of the project.

References
Bibliography

[1] mcxiaoke, "android-volley," GitHub, 2016. [Online]. Available: https://github.com/mcxiaoke/android-volley. Accessed: May 1, 2016.

[2] The Renal Association, "UK Renal Registry, 18th Annual Report," 2016. [Online]. Available: https://www.renalreg.org/wp-content/uploads/2015/01/web_book_07-04-16.pdf. Accessed: May 1, 2016.

[3] R. Saran, US Renal Data System 2015 Annual Data Report: Epidemiology of Kidney Disease in the United States., Y. Li and B. Robinson, Eds., 1st ed. In press, 2015.

[4] T. K. P. A., "Tayside Kidney Patients Association website,". [Online]. Available: http://www.tkpa.org.uk/. Accessed: Apr. 15, 2016.

[5] NHS, "Chronic kidney disease - NHS choices," Department of Health, 2016. [Online]. Available: http://www.nhs.uk/conditions/Kidney-disease-chronic/Pages/Introduction.aspx. Accessed: May 1, 2016.

[6] "Kidney disease: Key facts and figures," 2010. [Online]. Available: http://www.healthcheck.nhs.uk/document.php?o=81. Accessed: May 1, 2016.

[7] NICE, "Chronic kidney disease in adults: Assessment and management," NICE, 2014. [Online]. Available: https://www.nice.org.uk/guidance/cg182/chapter/1-recommendations. Accessed: May 1, 2016.

[8] NHS, "Kidney transplant - NHS choices," Department of Health, 2015. [Online]. Available: http://www.nhs.uk/Conditions/Kidney-transplant/Pages/Introduction.aspx. Accessed: May 1, 2016.

[9] R. Steenkamp, C. Shaw, and T. Feest, "UK renal registry 15th annual report: Chapter 5 survival and causes of death of UK adult patients on renal replacement therapy in 2011: National and centre-specific analyses," 2013. [Online]. Available: https://www.renalreg.org/wp-content/uploads/2014/09/Chapter_5.pdf. Accessed: May 1, 2016.

[10]"Hemodialys facts," 2016. [Online]. Available: http://www.niddk.nih.gov/health-information/health-topics/kidney-disease/hemodialysis/Pages/facts.aspx. Accessed: May 1, 2016.

[11]"PatientView website,". [Online]. Available: https://www.patientview.org/#/. Accessed: May 1, 2016.

[12]National Kidney Foundation, "App center," The National Kidney Foundation, 2016. [Online]. Available: https://www.kidney.org/apps. Accessed: May 1, 2016.

[13]"Mobile Friendly Websites," Google Developers, 2015. [Online]. Available: https://developers.google.com/webmasters/mobile-sites/get-started#key. Accessed: May 1, 2016.

[14]"IDC: Smartphone OS market share," www.idc.com, 2016. [Online]. Available: http://www.idc.com/prodserv/smartphone-os-market-share.jsp. Accessed: May 1, 2016.

[15]"Research ethics committees (RECs) - health research authority," Health Research Authority. [Online]. Available: http://www.hra.nhs.uk/about-the-hra/our-committees/research-ethics-committees-recs/. Accessed: May 1, 2016.

[16]Peter McKenzie, "NHS Tayside Information Governance: Caldicott approval procedure," 2010. [Online]. Available: http://www.tasc-research.org.uk/images/cmsimages/pdf/Caldicott%20Approval%20Procedure.pdf. Accessed: May 1, 2016.

[17]National Kidney Foundation, "The national kidney foundation," The National Kidney Foundation, 2016. [Online]. Available: https://www.kidney.org. Accessed: May 1, 2016.

[18]"Lab tests Online-UK," 2016. [Online]. Available: http://labtestsonline.org.uk/. Accessed: May 1, 2016.

[19]"Home - UK renal registry," UK Renal Registry, 2016. [Online]. Available: https://www.renalreg.org/. Accessed: May 1, 2016.

[20]"The Scottish renal registry," 2014. [Online]. Available: http://www.srr.scot.nhs.uk/. Accessed: May 1, 2016.

[21]"Manifesto for agile software development," 2001. [Online]. Available: http://agilemanifesto.org/iso/en/. Accessed: May 1, 2016.

[22]"Principles behind the agile manifesto,". [Online]. Available: http://agilemanifesto.org/principles.html. Accessed: May 1, 2016.

[23]"Personas," Department of Health and Human Services, 2013. [Online]. Available: http://www.usability.gov/how-to-and-tools/methods/personas.html. Accessed: May 1, 2016.

[24]"Use cases," Department of Health and Human Services, 2013. [Online]. Available: http://www.usability.gov/how-to-and-tools/methods/use-cases.html. Accessed: May 1, 2016.

[25]"Software requirements specification," 2016. [Online]. Available: http://www.chambers.com.au/glossary/rationale.php. Accessed: May 1, 2016.

[26]"User stories: An agile introduction,". [Online]. Available: http://www.agilemodeling.com/artifacts/userStory.htm. Accessed: May 1, 2016.

[27]"The product backlog: Your ultimate to-do list," Atlassian, 2016. [Online]. Available: https://www.atlassian.com/agile/backlogs. Accessed: May 1, 2016.

[28]"Renal Dialysis App on Pivotal Tracker,". [Online]. Available: https://www.pivotaltracker.com/n/projects/1523859. Accessed: May 1, 2016.

[29]"What is definition of done (DoD)?," 2016. [Online].Available: https://www.scrumalliance.org/community/articles/2008/september/what-is-definition-of-done-(dod). Accessed: May 1, 2016.

[30]"Scrum guides," 2014. [Online]. Available: http://www.scrumguides.org/scrum-guide.html. Accessed: May 1, 2016.

[31]"Git - about version control," 2nd ed. [Online]. Available: https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control. Accessed: May 1, 2016.

[32]V. Ignatjevs, "dialysisApp on GitHub,". [Online]. Available: https://github.com/VladislavsIgnatjevs/dialysisApp. Accessed: May 1, 2016.

[33]RenalDialysisApp, "RenalDialysisApp on Twitter," in Twitter, Twitter, 2016. [Online]. Available: https://twitter.com/dialysis_app. Accessed: May 1, 2016.

[34]"XAMPP by Apache friends,". [Online]. Available: https://www.apachefriends.org/index.html. Accessed: May 1, 2016.

[35]"HeidiSQL,". [Online]. Available: http://www.heidisql.com/. Accessed: May 1, 2016.

[36]Statista, "Distribution of Android operating systems used by Android phone owners in October 2015, by platform version," Statista, 2016. [Online]. Available: http://www.statista.com/statistics/271774/share-of-android-platforms-on-mobile-devices-with-android-os/. Accessed: May 1, 2016.

[37]"Android Platform Version Distribution,". [Online]. Available: http://developer.android.com/intl/ru/about/dashboards/index.html. Accessed: May 1, 2016.

[38]Roy Thomas, "Fielding Dissertation: CHAPTER 5: Representational state transfer (REST),". [Online]. Available: http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm. Accessed: May 1, 2016.

[39]R. Tamada, "Android Login and Registration with PHP, MySQL and SQLite," androidhive, 2012. [Online]. Available: http://www.androidhive.info/2012/01/android-login-and-registration-with-php-mysql-and-sqlite/. Accessed: May 1, 2016.

[40]"Android Activities,". [Online]. Available: http://developer.android.com/guide/components/activities.html. Accessed: May 1, 2016.

[41]"Android activity Lifecycle," www.javatpoint.com, 2011. [Online]. Available: http://www.javatpoint.com/android-life-cycle-of-activity. Accessed: May 1, 2016.

[42]"System usability scale (SUS)," Department of Health and Human Services, 2013. [Online]. Available: http://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html. Accessed: May 1, 2016.

[43]"Online SUS Calculator," Google Docs. [Online]. Available:https://docs.google.com/spreadsheets/d/1_DkSG52nWvcs5ah2dDWbKLOaNlQRHygj_ipAIwTliBo/edit#gid=1263174359. Accessed: May 1, 2016.

[44]"SUS: system usability scale," TryMyUi, 2016. [Online]. Available: http://www.trymyui.com/sus-system-usability-scale. Accessed: May 1, 2016.

Appendices

A. Development backlog
B. Log book
C. Meeting minutes
D. Mid-project progress report
E. Phase burndown chart
F. Poster
G. Project proposal
H. Personas
I. Use cases
J. Requirements specification
K. User Stories
L. Class Diagrams
M. Test plans
N. Design
O. Source code
P. Total burndown chart
Q. User guide