IoT HomeCare System Testing

Hristian Vitrychenko
Nikki Constancon
Juan du Preez
Gregory Austin
Marthinus Richter

October 13, 2017

1 Introduction

The testing report is designed to describe all tests that have been done on the system.

1.1 Purpose

The purpose of this document will be to list and describe all of the tests for the ReVA system, how they are carried out, what their purpose is and what subsystem and related module(s) they are related to.

1.2 Structure of the document

Each subsystem of ReVA will be addressed individually with the specific tests for each module concerned with that subsystem. The subsystems are:

- Real-Time Subsystem
  This includes the modules: Data Collection, Pub/Sub Server, Data Pull, and User Interface
- Data Storage Subsystem
  This includes the modules: Data Storage, Pub/Sub Server
- History/Statistics Subsystem
  This includes the modules: Data Storage, Statistics, Pub/Sub Server, and User Interface
- User Management Subsystem
  This includes the modules: User Management, User Interface
- Notification Subsystem
  This includes the modules: Pub/Sub Server, Notification, and User Interface
- Advice Subsystem
  This includes the modules: Advice and User Interface

1.3 Definitions, Acronyms, and Abbreviations

1.3.1 Acronyms

- **UI *(User Interface)***

  The means by which the user and a computer system interact, in particular, the use of input devices and software.

1.3.2 Definitions

- **Unit Test**

  Unit testing is a software development process in which the smallest testable parts of an application, called units, are individually and independently scrutinised for proper operation.

- **Integration Test**

  Find out if the units if integrated together will work without errors. For example, argument passing and data updation etc.

- **Functionality Test**

  Tests all functionalities of the software against the requirement.

- **Performance Test**

  This test proves how efficient the software is. It tests the effectiveness and average time taken by the software to do desired task. Performance testing is done by means of load testing and stress testing where the software is put under high user and data load under various environment conditions.

- **Alpha Testing**

  The team of developer themselves perform alpha testing by using the system as if it is being used in work environment. They try to find

out how user would react to some action in software and how the system should respond to inputs.

1.4 Additional Information

The code being tested is written by different developers, thus this document serves as a way in order to review the functionality and tests done, and to provide information about said tests on current functionality.

2 Real-time subsystem

Modules involved: Data Collection, Pub/Sub Server, Data Pull, and User Interface
Primary Actors: All users
Functionality: View real time data on patient(s)

2.0.1 Description of current tests

The data streaming consists of Raspberry pi's streaming to servers, and servers streaming that data out and the application displaying those data streams. We do this using the Nodejs package Zetta (which is already been tested) for the Data Collection and the Pub/Sub Server, and an open source Zetta-Starter-Android-Application developed to interface with the Zetta API to recieve the streams (DataPull) that's generated by the server which has also been tested. These have all been tested individually already, thus to create our own unit tests would be redundant.

The current tests are Integration Tests to see that data is in fact streaming from and to devices. You can see the tests in the three figures below.

2.0.2 Alpha tests:

Alpha Testing (testing it ourselves to potentially find problems) has been done each time new items or functionality has been added to the server or application. This is evidence of "eating our own cooking", and we have tried many different pathways that users would try in the application in order to make sure everything is working as it should.

2.0.3 Data Collection

Comment: At the moment the Pi collects data from mock devices. This data generated and streamed using the zetta architecture and therefore has already been tested with Unit tests thoroughly. (Zetta is an open source Nodejs project purposed for IoT)

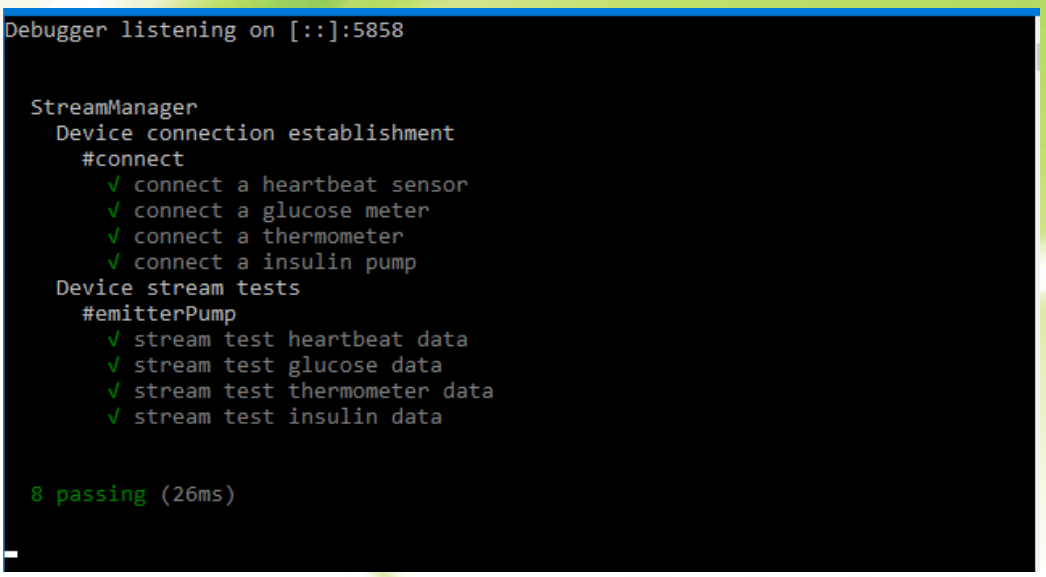


Figure 1: Mocha output of successful mock device streams

2.0.4 Pub/Sub Server

Comment: The Pub/Sub server uses zetta architecture as well and that's how it communicates with the pi (recieves streams being published) and any requesting devices looking to subscribe. Zetta already is established and has tests.

2.0.5 Data Pull

Comment: The way the data is pulled is by using the architecture of the open source Zetta-Starter-Android-Application developed to interface with the Zetta API that's generated by the server. This has also been tested, so testing is redundant.

2.0.6 User Interface

Comment: The user interface only displays data, and therefore it is easy to verify what is being displayed and that it is working as expected.

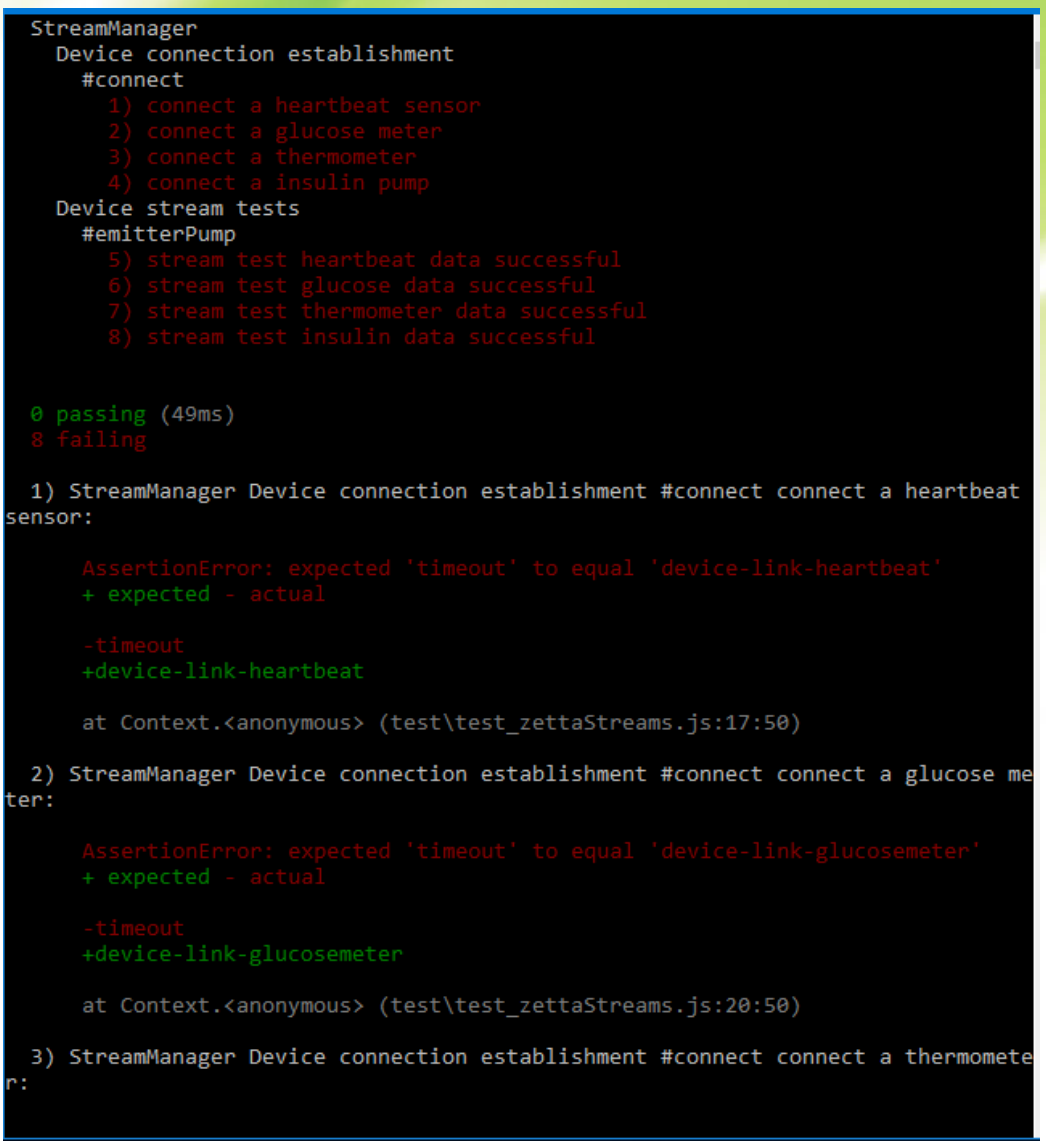


Figure 2: Mocha output (1 of 2) for unsuccessful mock device streams (due to being programmatic disabled)

3 Data Storage Subsystem

Modules involved: Data Storage, Pub/Sub Server
Functionality: Persist patient's data

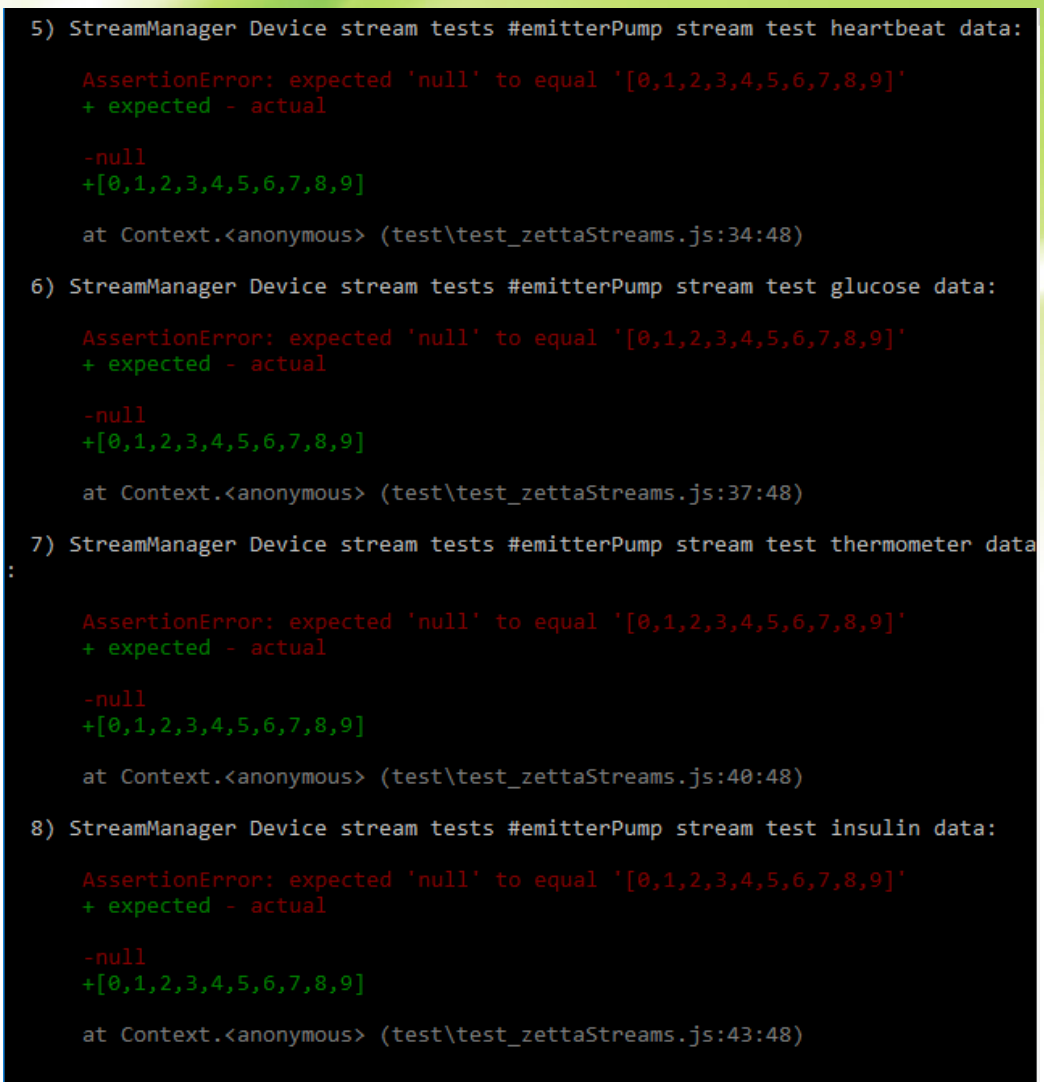


Figure 3: Mocha output (2 of 2) for unsuccessful mock device streams (due to being programmatic disabled)

Unit tests where executed to verify that the zetta kit was integrated seamlessly, allowing the system to pass timestamped real-time data to the associated Cassandra. Unit tests where also executed to ensure that obtained data is persisted correctly.

4 History/Statistics subsystem

Modules involved: Data Storage, Statistics, Pub/Sub Server, and User Interface
Functionality: View statistical data of patient(s)
Unit tests for each coupled module where executed. For statistics this included unit testing for verification of getting the right data from queries, as well as any mathematics done by the statistics module, e.g., min, max, avg, mean etc. The user-interface performed unit-testing to make sure that graphs are being shown as specified, or statistics about the patient are indeed correct. Other tests where included to test integration to make sure everything is communicating as expected. There was also be functionality testing to ensure that the History/Statistics subsystem meets the SRS accurately.

5 User Management Subsystem

Modules involved User Management, User Interface

5.0.1 User Management

Primary Actors: users, admin users
Functionality Create, Remove, Update or Delete items related to data storage, e.g., user info, users, user relationships etc.

5.0.2 Description of current tests

Precondition: The user manager is tested to ensure that arguments passed to the database manager are correctly formatted, i.e., that the json conforms to the database model.
Postcondition: Ensure that intended parameters are added to the database after successful query.
Invariant: Ensure that only intended parameters change within the related managers and database.
Example of when the CRUD test passes

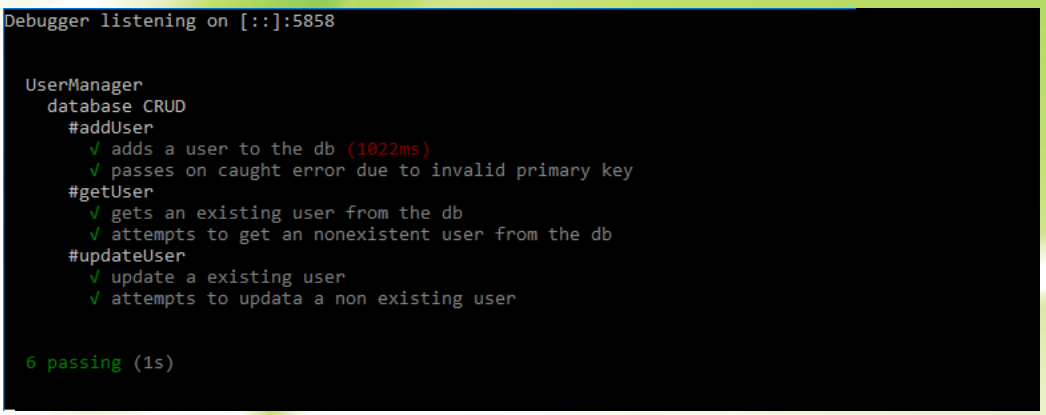


Figure 4: Mocha output of successful *User Manager* test

User Manager CRUD fail example, due to Cassandra database being down. Note error reports on the console is also piped into a rotating file. Thus systems users will be able to track down errors after they occur.

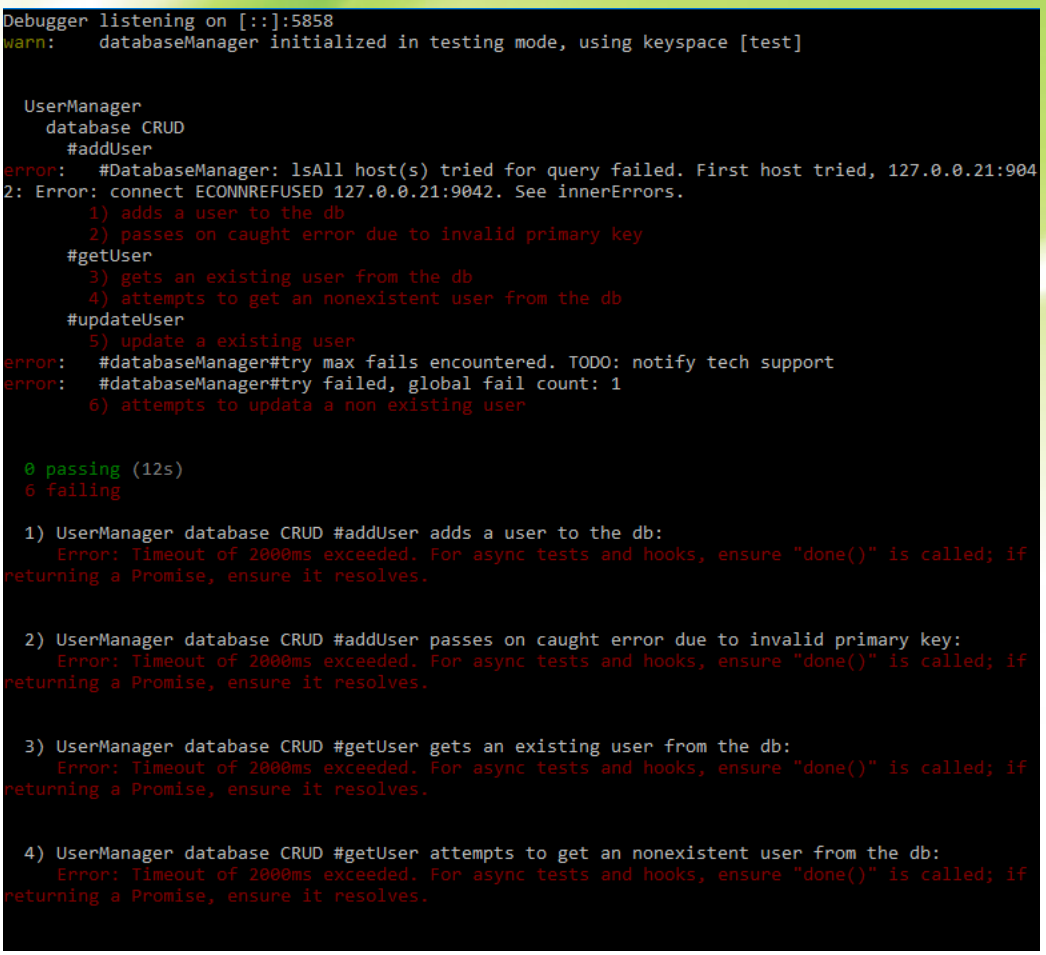


Figure 5: Mocha output of unsuccessful *User Manager* test, due to database connection failure

5.0.3 User Interface

Primary Actors: all users
Functionality Type in details and register to become a new user. There is only local validation and forms currently implemented. Client-server communication for registration and user management is not yet implemented.

5.0.4 Description of current tests

There are unit tests for each input that test the validation of those inputs. So there are tests for email, passwords, usernames etc. All to make sure that the validation is correct.
Integration tests to verify client-server communication was executed together with functionality tests to confirm the User Management subsystem adheres to the requirements. **Following is the code for each unit test to show what is tested.**

```
@Test
public void enterAgeTest() throws Exception
{
    int[] ages = {-3, 0, 6, 23, 65, 150, 4, 12, -4, -5};

    for(int i = 0; i < ages.length; i++)
    {
        if(ages[i] < 0 || ages[i] > 140)
        {
            throw new Exception();
        }
    }
}
```

```
@Test
public void verificationTestLoginDetails() throws Exception
{
    String[] emails = {"Bob@gmail.com", "Carl@yahoo.co.za", "lisa.com", "mina@hotmail", "tanith", "eric123@yknot.com", "trish@mail.cl", ".co.za@charles", "@.co.za", "co"};
    String[] passwords = {"123", "hello", "youaretheone", "IamGr00t", "hiiamatest", "IshouldWOrk", "iishouldnot", "ISHOULDNOTWORK?", "ThisIs123", "        "};

    for(int i = 0; i < emails.length; i++)
    {
        verificationTestLoginDetails(emails[i], passwords[i]);
    }
}
```

```
public void verificationTestLoginDetails(String email, String password) throws Exception
{
    String mail = email, pass = password;

    if(email.length() < 1 || password.length() < 6)
    {
        throw new Exception();
    }
    else if(!email.contains("@") || (!password.contains("1") && !password.contains("2") && !password.contains("3") && !password.contains("4") && !password.contains("5")
            && !password.contains("6") && !password.contains("7") && !password.contains("8") && !password.contains("9") && !password.contains("0")))
    {
        throw new Exception();
    }
    else if(!email.contains(".co.za") && !email.contains(".com") || password.toLowerCase().equals(password) || password.toUpperCase().equals(password))
    {
        throw new Exception();
    }
}
```

```
@Test
public void verificationTestRegPatient1() throws Exception
{
    String[] emails = {"Bob@gmail.com", "Carl@yahoo.co.za", "lisa.com", "mina@hotmail", "kanith", "eric123@yknot.com", "trish@mail.cl", ".co.za@charles", "@.co.za", "co"};
    String[] passwords = {"123", "hello", "youaretheone", "IamGr00t", "1hiiamatest", "IshouldWork", "1ishouldnot", "ISHOULDNOTWORK2", "ThisIs123", "      "};
    String[] confirmPass = {"123", "hello", "youaretheone", "IamGr00t", "1hiiamatest", "IshouldWOk", "1ishouldnot", "ISHOULDNOTWORK2", "ThisIs124", "      "};

    for(int i = 0; i < 10; i++)
    {
        verificationTestLoginDetails(emails[i], passwords[i]);

        if(!passwords[i].equals(confirmPass[i]))
        {
            throw new Exception();
        }
    }
}
```

6 Notification Subsystem

Modules involved: Pub/Sub Server, Notification, and User Interface
Unit tests for each individual module as well as integration tests to be sure messages are being passed properly between modules was executed.

7 Tests based on non-functional requirements

7.1 Usability study

One of the keys to accurate applicable data is that the users need to be relaxed. We planned to make the usability testing as relaxed as possible, to do this

- We had lollipops and toffees ready and waiting and as a reward for finishing the test.
- Clear instructions on how to get to the usability testing location, we meet with testers outside IT and took them to the venue.
- Get the user familiar with the environment.

```
@Test
public void pickUserNameTest() throws Exception
{
    String[] existing = {"carl123", "bob6", "Juan du Preez", "Jamie Bob", "George", "Sally", "Mick", "John", "1Address", "Bob"};
    String[] tested = {"carl13", "bb6", "Juan du Preez", "Jamie Bob", "George", "Sally", "Mik", "John", "1Address", "Bo"};

    for(int i = 0; i < existing.length; i++)
    {
        for(int j = 0; j < tested.length; j++)
        {
            if(existing[i].equals(tested[j]))
            {
                throw new Exception();
            }
        }
    }
}
```

- If they can't do something, make sure they know it'snot their fault.
- We reassure that the tests are completely confidential, and for permission to use the data generated during the test as part of our results (do this with a short consent form)
- To use no terms like usability testing to keep testers at ease.
- Tell them how long it will take.
- Although acting professional, some mild humour and conversation to relax participants.

We planned to not alter the test results by providing clues, suggesting directions or by reacting to things they say or do. Although taking 5 minutes for a task means the task is too hard (for our app).

The results will only be as good as the people you test, so we needed to choose people who at least, to some degree, had an interest in medical IoT and the tech. People were be asked if they were interested in the application, and because we dont know that many people, we picked people we knew (Convenience Sampling). Using our user needs we devised scenarios to carry out. **People tend to perform more naturally if youprovide them with scenariosrather than instructions** so we planned scenarios.

7.1.1 The script

Welcome, we are Loop, Im Greg, this is Nikki, Hris,and Juan. You were chosen because you have some interest in our app. Our app ReVA helps people see vitals of bedridden or ill people and allows you to do this on the

go as well as stats and alerts. You guys are here to test the app to see if its easy to use or terrible to use, youre not here to be tested, youre here to test the website. Were aiming for accurate data on the usability of our website, so just take it easy and use it as you would at home.

Before we begin make sure you sign the disclaimer at the top of the questionnaire. Youll find a small task sheet with tasks you have to do, try your best for each task, but if youre stuck for more than 5 minutes dont hesitate to ask for help. If you do get stuck that just means the website needs improving. If you are unsure if you completed a task or not, just ask us and well tell you if you did. After the tasks are done fill out the questionnaire. After youre done with that just wait for everyone to finish and then you can leave.

7.1.2 Process followed

We fetched the participants from outside the IT building, one participant was missing, however this was okay as we still had 6 participants. The chosen participants werent old or doctors, although they werent perfect, data of the navigation of the UI and UI impressions was still collected. We introduced ourselves and put the participants at ease with the script and lollipops and toffees. We gave each participant a task sheet and a phone with the app. We allowed them to begin whenever they felt ready.

The users took longer than expected. This was due to the fact that some tasks on the task sheet needed requests accepted on our side, but we were not doing that. Also some mock data for vitals wasnt streaming either so halfway through we had to tell them to use another patient for the task. We realised this was a mistake and shouldve been prepared for it. The one application on one specific phone had some particular changes at that time so it crashed on some pages. Which was not good for consistency of the test.

7.1.3 Tasks performed by users

- Task 1: **Your aunt nikki has recently been bedridden with a terminal illness, she has asked you to register for the app ReVA.** Cho-

sen because this is a common task users have to complete. Registering for the app is essential.

- **Task 2: Now that youre registered aunt Nikki has asked you to add her as a patient in the app, she sent you a message with her username nikki.** Chosen because adding patients is imperative for the purposes of ReVA.
- **Task 3: You need to see how your aunt is doing, check your aunts vitals.** Chosen because patient vitals is the core requirement of the application and seeing how easy it is to navigate here is important.
- **Task 4: You need to see your aunts average heart rate for the last day, find out what her average heart rate is.** Chosen to see that stats is easy to navigate to and that users can easily tell where statistics is.
- **Task 5: Your Grampa Greg has been overcome with illness, he too asks you to add him on the app ReVA his username is greg.** Chosen so that we could see how difficult it was to change between patients and the statistics of those patients.
- **Task 6: Have a look at Grampa Gregs body glucose graph for the last hour and try and get a closer look at the last 15 minutes.** Chosen so that we could see that people found the graphs easy to use and intuitive to find out.
- **Task 7: Leave the app as you normally would, you should get a notification warning you about one of your family members. Get back to the app through the notification.** Chosen to see that alerts are intuitive and show you when there is a problem.
- **Task 8: Notifications have started to annoy you, turn them off.** Chosen to see that this setting was easy to find and important because not all caretakers/subscribers want or need notifications.
- **Task 9: Your aunt Nikki has passed away, you decide to delete her off the app reva.** Chosen for user management sake and to see that it was easy to do.

7.1.4 Evaluation Methods

Survey
We used the PSSUQ because it was designed specifically for scenario-based usability studies (like ours), so some questions are more targeted, like I was able to complete the tasks and scenarios quickly using this system.. We used it to give us accurate data on how easy our system is to use/to navigate. The answering scale is also more complex, ranging from 1 to 7 instead of 1 to 5. This allows testers to give more nuanced responses to each question.

The end outcome is a score from 0 to 100, reflecting the overall usability of the website or app based on the respondents experience. PSSUQ also has 3 sub-scores, derived from subsets of the 16 questions, which reflect system usefulness, information quality, and interface quality. System Quality (the average of items 1-6), Information Quality (the average of items 7-12), and Interface Quality (the average of items 13-16)

Averaged survey results
Taking an average of the results give you a basic idea on how people perform on average. Although there can be outliers, this gives us nice easy to read data so that we can see where the problematic areas of the app are.

Interview with testers
We did a conversational interview after everyone was done with the usability test in order to illicit any comments, criticisms or advice about the application.

7.1.5 The results

Comments on the survey (only where there were comments)

User 2 comments: Some serious connectivity issues on my end, but the actual functionality of the system was highly intuitive. (Maybe could be more colourful) Also it crashed a number of times while using it (usually

Question	Description
1	Overall, I am satisfied with how easy it is to use this system
2	It was simple to use this system
3	I was able to complete the tasks and scenarios quickly using this system
4	I felt comfortable using this system
5	It was easy to learn to use this system
6	I believe I could become productive quickly using this system
7	The system gave error messages that clearly told me how to fix problems
8	Whenever I made a mistake using the system, I could recover easily and quickly
9	The information (such as online help, on-screen messages and other documentation) provided with this system was clear.
10	It was easy to find the information I needed.
11	The information was effective in helping me complete the tasks and scenarios.
12	The organization of information on the system screens was clear
13	The interface of this system was pleasant
14	I liked using the interface of this system
15	This system has all the functions and capabilities I expect it to have
16	Overall, I am satisfied with this system.

when loading/connecting to patient information).
User 3 comments: Possibly a link for cellphone number for those using the app. Also medical information on where the hospital is and contact numbers linked to if there is an emergency (automatic msgs/emails)
User 4 comments: Reva stopped working when I gave the wrong password. My connections had me confused when trying to add patients, but clicking it made it clear what it is.
User 6 comments: Very very nice UI and pleasant feedback when pushing buttons. Only one suggestion for enable/disable notification is have a slide lock thing just to help users not get confused between the enable/disable things.

Results of interview with the room:
There were five suggestions from the interview with the room:

- Prompt when adding a person asks for an email, but you can use

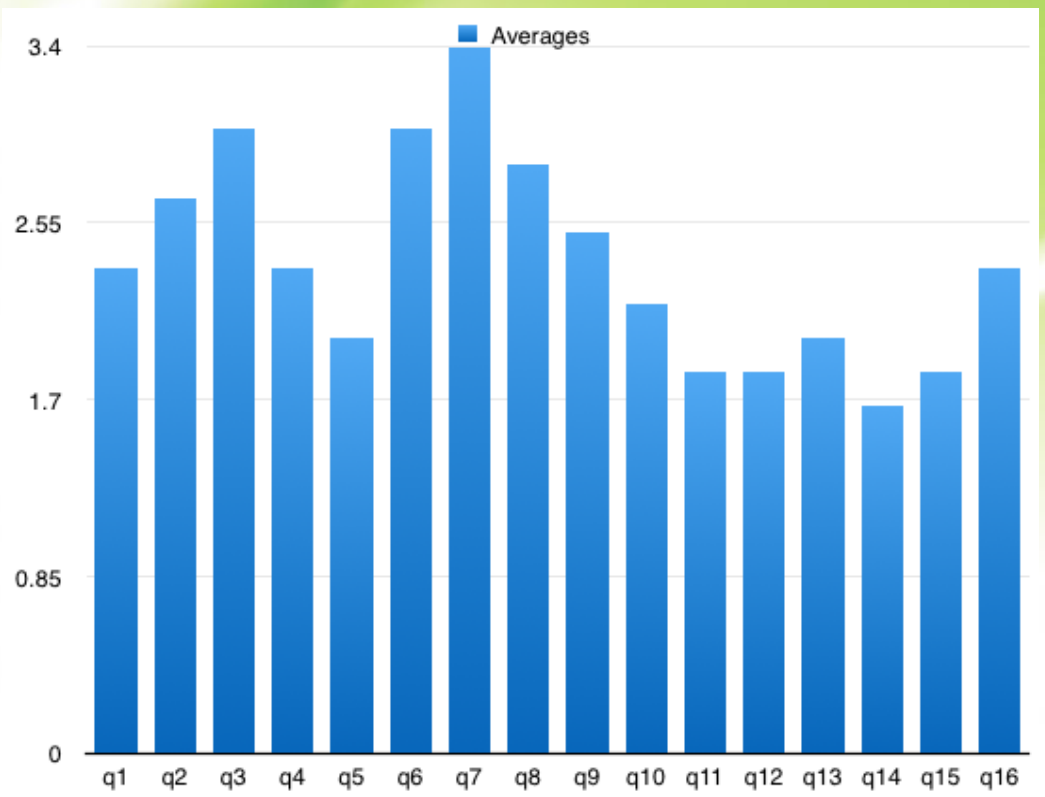
	User 1	User 2	User 3	User 4	User 5	User 6	avg
Question 1	3	3	2	1	3	2	2.33
Question 2	3	3	3	1	4	2	2.67
Question 3	4	5	3	1	4	1	3
Question 4	4	3	2	1	3	1	2.33
Question 5	4	2	2	1	2	1	2
Question 6	5	3	3	1	4	3	3
Question 7	5	5	3	2	2	na	3.4
Question 8	4	5	2	2	3	1	2.83
Question 9	1	5	1	1	6	1	2.5
Question 10	3	3	2	1	2	2	2.16
Question 11	3	1	2	1	3	1	1.83
Question 12	1	4	2	1	2	1	1.83
Question 13	1	3	1	3	3	1	2
Question 14	1	2	1	2	3	1	1.67
Question 15	3	1	1	1	3	2	1.83
Question 16	3	3	2	1	4	1	2.33

username as well

- My connections is a bad name for user management and should be called something like Patient management or Friends management or something
- Message when youre a new user, saying what to do
- Refresh for the patient vitals page
- Partition the vitals section better so that you can easily access what you want to see

7.1.6 Conclusions

Looking at the PSSUQ scores, the highest score was for the interface, however we got some lower scores for system quality and information qual-



ity. Looking at some of the lowest avg scores for some questions you can see that 3,6,7,8 are all the lowest. The first q3: I was able to complete the tasks and scenarios quickly using this system - the reason this may have been low was because of a mistake with the tasks and us being ready to serve that information or accept requests for specific users. It also could be because the interface is hard to navigate but in general people claimed it was easy to use so this couldve been a mistake on our part.

The next question, question 6 with the 2nd lowest avg score, is: I believe I could become productive quickly using this system - this has to do with productivity and how fast actions can be completed. There is no real productivity to the app, you just look at patient vitals, however this could be indicative of information overload because of many different vitals to look at it. A possible partitioning of the vitals and stats screen will be considered.

PSSUQ scores	User 1	User 2	User 3	User 4	User 5	User 6	Avg
System Quality (1-6)	59.5%	69.04%	78.5%	100%	66.67%	90.47%	77.36%
Information Quality (7-12)	68.57%	51.14%	82.85%	94.29%	65.71%	97.14%	76.61%
Interface Quality (13-16)	85.71%	82.14%	96.4%	89.28%	67.85%	96.43%	86.3%
Overall quality	71.26%	67.44%	85.92%	94.52%	67.74%	94.68%	80.09%

The lowest item score was q7: The system gave error messages that clearly told me how to fix problems - this means that some messages and items in the navigation were not named clearly. For example User 4 mentioned that My connections was a confusing name that could be related to networking. We will consider changing that to something else as well as messages for new users on how to add new patients and helpful messages like that for navigation in future iterations.

Another item with the 2nd lowest item score is q8: Whenever I made a mistake using the system, I could recover easily and quickly - we believe this has to do with interaction behaviour and the flow of screens but also has to do with the server not being up and the resulting confusion from sending a patient request, and not getting accepted and then even if they were accepted, the devices not being up. The message for devices not being up is also ambiguous as it just says the server is not online without describing which particular user it is about. These are considerations to take into account.

With the room interview, there was some valid feedback and all five points will be taken into consideration with further iterations of ReVA. Specifically things to do with error messages and tips or help messages.

Overall the feedback was positive with an overall score of 80.09 for the PSSUQ. This shows our system is indeed quite good, with some needed improvement mostly to do with informational messages, i.e., informational quality - this was our lowest score at 76.61. This means that we should focus on changing the informational messages with better error messages, better help messages and better state messages for when users navigate to somewhere by accident.

8 Platform compatibility test

We have tested the ReVA app on several android devices with different versions of android, totalling to 6. We have tested on:

- Samsung S8 (Nougat)
- Xiaomi MI5 (Marshmellow)
- Samsung Grand Neo (KitKat)
- Hauwei P8 Lite (Marshmellow)
- Samsung S4 (Lollipop)
- Samsung S3 Mini (KitKat)

This shows a wide variety of supported versions of the application. Which we have tested on and seen work empirically. We have even scrapped a few nice to haves of the newer development kits to support older versions like KitKat.