

# CHALMERS



## Touch interface in the healthcare industry

Converting a desktop application to support touch interactions in the healthcare industry, with focus on interaction design.

*Master of Science Thesis in Interaction Design*

Tommy Davidsson

Joachim Haglund

Department of Applied Information Technology  
CHALMERS UNIVERSITY OF TECHNOLOGY

Gothenburg, Sweden, 2013

Report No. 2013:011

ISSN: 1651-4769



*Touch interface in the healthcare industry*  
*Converting a desktop application to support*  
*touch interactions in the healthcare industry,*  
*with focus on interaction design*

*Tommy Davidsson*  
*Joachim Haglund*

Touch interface in the healthcare industry

Converting a desktop application to support  
touch interactions in the healthcare industry,  
with focus on interaction design

© Tommy Davidsson, Joachim Haglund 2013.

Department of Applied Information Technology  
CHALMERS UNIVERSITY OF TECHNOLOGY  
SE-412 96 Gothenburg  
Sweden

Telephone

Tommy: +46 706 520329

Joachim: +46 737 845179

# ***Preface***

This report is the final outcome of a master thesis; the final examination for a Master of Science in Interaction Design at Chalmers University of Technology in Gothenburg, Sweden. It was carried out at the Unite Solutions R & D at Ascom Wireless Solutions during spring-summer of 2013. We would like to thank everybody that contributed with their time, energy and knowledge, without whom this outcome would have been impossible. Special thanks to our supervisors at Ascom; Anna Engström and Jan Bentzer.

We would also like to thank our supervisor at Chalmers University of Technology; Olof Torgersson for valuable input and support.

Last but not least we would like to thank the nurses that user tested our prototype for their contribution and valuable feedback on our design.

KEYWORDS: NUI, Touch Interface, Healthcare, Alarm



## ***Abstract***

This master thesis was carried out at Ascom Wireless Solutions. Ascom Wireless Solutions (Company within the Swiss Ascom group) is a world leading solution provider with comprehensive technological know-how in Mission-Critical Communication. The aim of this master thesis was to find out how to adapt a desktop user interface under development to support touch/gestural interaction in the healthcare. What should be done, what should not be done, and how could a prototype look like? It was addressed by analyzing wireframes and the first stable version of the View interface, researching guidelines for touch interfaces and similar projects and constructing one high fidelity prototype that was as near as possible (interface wise not technical) to the real version of View.

## CONTENTS

1	Introduction.....	11
1.1	Purpose.....	11
1.2	Problem .....	11
1.3	Delimitations .....	12
1.4	Expected Results.....	13
1.5	Critical Systems .....	13
2	Theory .....	15
2.1	Using standards is important.....	15
2.2	Long distance and natural user interfaces.....	25
2.3	Screen size .....	28
3	Background.....	29
4	Method.....	32
4.1	Literature study .....	32
4.2	Brainstorming .....	33
4.3	Sketching .....	33
4.4	Mockups .....	34
4.5	Prototyping.....	34
4.6	Interactive prototype.....	34
4.7	Ascom User testing.....	34
4.8	Qualitative user test .....	35
4.9	Selection .....	35
4.10	Methods not used .....	36
5	Design Process.....	37
5.1	Pre Study.....	37
5.2	Brainstorming .....	38
5.3	Sketching .....	39



5.3.1	Define form factor, posture, and input methods.....	39
5.3.2	Define functional and data elements .....	39
5.3.3	Determine functional groups and hierarchy .....	41
5.3.4	Sketch the interaction framework .....	43
5.3.5	Construct key path scenarios .....	46
5.3.6	Check designs with validation scenarios .....	46
5.4	Mockups .....	47
5.5	First Prototype .....	48
5.5.1	Key Guidelines Followed .....	50
5.5.2	Windows 8.....	51
5.6	Ascom Future evening (testing).....	52
5.6.1	Preparations .....	52
5.6.2	Test persons .....	53
5.6.3	Tasks.....	56
5.6.4	Outcomes from the user tests.....	56
5.6.5	User Test Statistics .....	60
5.7	Evaluate .....	62
5.8	Prototype (html) .....	63
5.9	Testing interactive prototype 2 (At Uddevalla hospital) .....	67
5.10	Outcomes from the second user test .....	70
6	Final Result .....	74
7	Discussion .....	84
7.1	Methodology .....	84
7.2	Results .....	85
7.3	possible improvements /future research .....	90
8	Conclusion .....	92
9	References.....	96

10	Appendix .....	98
10.1	Appendix A – System Usability Scale (first usability test) .....	98
10.2	Appendix B – First Usability Test Tasks (In Swedish) .....	99
10.3	Appendix C - Second Usability Test Tasks (In Swedish) .....	100
10.4	Appendix D - System Database Diagram .....	101
10.5	Appendix E - System Site Map .....	102
10.6	Appendix F - First User test detailed statistics.....	103

## 1 INTRODUCTION

Touch interfaces are becoming more common in today's society, you now meet them everywhere, at the food store, in cars, in the pumps at the gas station and not at least in smart phones and in computers, it seems like they are here to stay. With this in mind more products need to consider if they should be developed for both touch and ordinary desktop environment, also referred to as Wimp (Windows Icons Menus Pointers). In this project we created a prototype for how such a healthcare Wimp interface could be converted to be adapted to touch/gestural interactions.

### 1.1 PURPOSE

Ascom Wireless Solutions has developed a middle ware product with several computer applications, a family called Unite (developed with among others Microsoft Silverlight) which are developed to facilitate nurses work in the health care industry. The company's application that we worked with was called Unite View which is an application to visualize all the alarms that are coming from several systems<sup>1</sup>, which in turn is connected to equipment in the healthcare connected to individual patients, alarm buttons found in hospital bedrooms as well as toilets. An increase in heart rate coming from a heart monitor is sent not only to the nurses' personal dect<sup>2</sup> phones, but also to the View application, which in turn visualizes this on a large touch screen hanging in the ward corridor. Unite View is supposed to act as a "second in line" guarantee that nurses notice and receive the alarms coming from patients. When we entered the project it was in its early stage and Ascom had just released its first stable version (Q1) with limited functions, and it was here that Ascom asked us to develop a concept prototype for how their equivalence of their View application could look like if adapted to a touch/gestural interactions, the current being a standard Wimp interface.

### 1.2 PROBLEM

How should a traditional desktop interface be translated into a large screen touch/gestural interface in a healthcare context?

---

<sup>1</sup> The systems supported are among others: **MMG, Cardiomax – Philips/Spacelabs interface, Mindray, Rauland Borg, Telligence.**

<sup>2</sup> Digital Enhanced Cordless Telecommunications (DECT™) is the ETSI standard for short-range cordless communications, which can be adapted for many applications and can be used over unlicensed frequency allocations world-wide.

### 1.3 DELIMITATIONS

Ascom Unite View is supposed to be a generic product, i.e. there is supposed to be one version of View, but it should support different ways of working with it. For example in the United States, alarms aren't necessarily directly directed to individual nurse's dect phones, but there is a so called "war room" where there are nurses that take the alarms from the individual patients medical equipment, and here we are talking about a person sitting down in front of a computer and interacting with it. Developing a touch interface here is possible but the person using it would risk getting the so called gorilla arm from using it (gorilla arm is a term often used to describe the appearance of a user with sore and cramped arms after interacting for a long time with a touch screen). In other departments such as ICUs (Intensive Care Units), View is supposed to be an easy access terminal in the corridor to get an overview of all the alarms going off. We are delimiting us to not take into consideration how easy the program is to "customize". We are also delimiting us to not look into how such a war room view application could look like but rather the one hanging in an ICU ward corridor.

In the healthcare there exist a lot of policies regarding **patient sensitive information**. This wasn't much researched in our project, we had some discussion about how we should act on this matter, but nothing concrete was presented. One suggestion of usage is that View is hanging in a ward corridor where ordinary people/relatives will pass by and here it's not desirable to show patient sensitive data without any kind of protection. In the end our application shows some patient sensitive data, like comments made from other nurses on a patient, we show the personnel info such as their working title, email, phone number etc. which if it were a real application should not be present or protected with password/fingerprint/gesture/RFID/Bluetooth in order to unlock this information.


Our project is involving the software only and not any hardware, e.g. what screen to choose, although optimal screen size is discussed. Later on in our project we considered experimenting with some kind of Bluetooth/RFID technology in order for the screen to unlock patient sensitive data while approaching the screen and not having to enter a password to post comments but this was skipped. Neither are we considering practical issues here like for example whether nurses should be able to interact with gloves or that the screen would need to be liquid proof in order to be able to disinfect it and wipe it clean of all bacteria gathering on the screen. Another technical limitation that was made was to use fake data for the final prototype. Programming it to communicate with an actual MMG system would be extremely complex and take a large amount of time, for something that was not really in the project description to do. So in order to be able to get incoming alarms etc. the decision was made to use faked data somehow. When the

decision was made to use HTML5 for the final prototype, naturally using a database and PHP for the fake data was the first choice. Other alternatives such as socketing in Windows 8 were considered, however the decision was made to stick with MySQL/PHP to distribute more time for the designing of the actual interface. Another thing that we are not taking into account that is a crucial thing in mobile devices is caching data to speed up the loading of interfaces since we assume that the application will be permanently connected by wire to the internet (when using Fass).

#### 1.4 EXPECTED RESULTS

The expected result of this thesis was a set of guidelines to follow when converting a desktop application into a large screen touch interface, specifically in a healthcare context. The expected results also included which of the currently existing guidelines for touch interfaces that should be applied, and which that should be avoided.

Additionally we expected to produce a full working prototype of our converted touch version of View. This was done in two steps, the first was to construct a high fidelity prototype of what the system could look like, and the second was to actually implement this with a convenient programming language expecting the prototype to take fake data as input. The first goal was that the data should be real in that sense that it should look the same as real data coming from medical systems, but later on in the project it was skipped in order to save development time and the alarms had to be manually inserted into the MySQL database in order for them to appear in the prototype. In order to facilitate this, a simple PHP input page was created.



Priority:  Bed:  Description:  Type:

**Figure 1: Showing the alarm input page.**

#### 1.5 CRITICAL SYSTEMS

The real View application is not categorized as a safety-critical system<sup>3</sup> itself; it's merely a display device to facilitate alarm awareness at a ward. With this in mind system failure might still result in personal injury or even loss of life if nurses are not made aware of alarms (although this is only a second in line safety net, together with the alarms coming to the nurses Dect phones). One of the important properties in this case is the

---

<sup>3</sup> Safety-critical systems are those systems whose failure could result in loss of life, significant property damage, or damage to the environment. John C. Knight : Safety Critical Systems

application's dependability<sup>4</sup> and the View software reliability<sup>5</sup> the so called MTBF (meantime between failures), it must work always. This was something that was reflected upon but wasn't taken into consideration when developing the real prototype since the project as far as we were concerned only involved what was visible in the interface and some minor bugs/errors were tolerated. This also meant that we could use whatever development technology/environment we wanted and didn't have to adapt to using trusted methods and technology that is required for developing a critical system (an application run through a browser might not be such a good idea in this case). Neither is the hardware reliability or security considered in this project. The only security we considered is the encryption of passwords inside the database. Still early on in the project three weeks to fully test the prototype to get rid of bugs in it were planned.

---

<sup>4</sup> Probability that a computer or other system will perform its intended functions in its specified environment without significant degradation.

<sup>5</sup> Ability of a computer program to perform its intended functions and operations in a system's environment, without experiencing failure (system crash).

## 2 THEORY

In this chapter theory of what makes a good touch interface will be presented. A big part of the text and images in this chapter is strongly influenced/taken from the iOS and the Android guidelines (some is directly copied).

### 2.1 USING STANDARDS IS IMPORTANT.

Touch interfaces differ from ordinary wimp interfaces and the reason is simple, how the user interacts with the system (using mouse & keyboard vs. gestures). "One step forward, two steps back" as Norman & Nielsen wrote in their article about gestural interfaces (D. Norman, J. Nielsen, 2010). While well-established standards have been developed for wimp interfaces for a decade or more it isn't until lately that well written standards for mobile touch interfaces have been set (yet slightly different for each platform). Before this, and still in some cases, the developers totally ignored some of the most fundamental elements when designing a user friendly interface (Consistency/Discoverability/Visibility). Visibility being one of the most obvious early problems, since the user interacts with gestures and those interactions are "invisible". One might argue that standard guidelines should have been designed from the beginning. As it is now iOS/Android/Windows Phone/Windows 8 all have slightly different guidelines on how to design/interact with interactable objects. A problem that occurs with this is that users of View all have different mobile operating systems and therefore have a slightly different experience in using touch interfaces. For example closing an application in Windows 8 differs from closing an application in Android/iOS. According to the prestudy made in the project there still doesn't exist any major design guidelines for larger touch screen interfaces (50" and above), which is unfortunate because this would facilitate for the users. Windows 8 was the closest, however users have used Android and iOS for a longer time than Windows 8, iPhone was released in 2007 and Windows 8 was released in October 2012. Naturally users have more experience in using these touch interfaces. The drawback that these interfaces have is that they are designed for smaller screens and as View is going to run on a larger screen, some of these guidelines/gestures might not be well suited.

Most touch interfaces in use in today's society are mobile devices (Smartphones/tablets) and especially there are two big dominant Operating Systems on the market; iOS and Android. According to the International Data Corporation (IDC website, 2013), Android and iOS combine for 92.3% of All Smartphone/Tablet Operating System Shipments in the First Quarter 2013. Naturally guidelines from these OS manufactures are important to study to see how they suggest that a touch interface should be designed. Below are some key guidelines that were picked up and used in the high fidelity prototypes.

Regarding the naming of menus and explanation texts in the application, it's extremely important to follow certain guidelines in order to make it easy for the user to understand what the different actions they take do. There shouldn't exist any uncertainty in the user because they do not understand the texts in the system. The following bullet list is directly quoted from Android Designer guidelines (Android website, 2013).

- ***“Keep it brief.*** *Be concise, simple and precise. Start with a 30 character limit (including spaces), and don't use more unless absolutely necessary.*
- ***Keep it simple.*** *Pretend you're speaking to someone who's smart and competent, but doesn't know technical jargon and may not speak English very well. Use short words, active verbs, and common nouns. Use terminology that you make sure your users understand. In an application designed especially for the healthcare such typical healthcare “language” could be appreciated by the users.*
- ***Be friendly.*** *Use contractions. Talk directly to the reader using second person (“you”). If your text doesn't read the way you'd say it in casual conversation, it's probably not the way you should write it. Don't be abrupt or annoying and make the user feel safe, happy and energized.*
- ***Put the most important thing first.*** *The first two words (around 11 characters, including spaces) should include at least a taste of the most important information in the string. If they don't, start over.*
- ***Describe only what's necessary, and no more.*** *Don't try to explain subtle differences. They will be lost on most users.*
- ***Avoid repetition.*** *If a significant term gets repeated within a screen or block of text, find a way to use it just once.*
- ***Keep it brief.*** *Use short phrases with simple words. People are likely to skip sentences if they're long.”*

When comparing the user's interaction with a desktop application and the interaction with a mobile device there exists one big difference, namely the time spent with the application. Whenever the desktop user can sit around and spend some time with the interface and tolerate some waiting there should exist no waiting in a mobile device, the user uses it in short bursts and wants instant access to whatever information they desire at the time. There is a big similarity with this and the users (Nurses/Charge



Nurses/Physicians etc.) of the Unite View Touch interface that is supposed to be hanging in the hospital corridor. Alarms from patients might come suddenly and so the nurse will have to run off and help that patient and totally forgot what they were doing at the moment. With this in mind it's crucial that the users can perform necessary tasks fast since often the user will only have a few seconds to perform it. The guidelines suggest that the developer should make it easy for the user to make choices for example use a picker rather than have the user manually type in what they desire, because it's easier for people to select an item from a list than to type words.

This obvious difference between a desktop application and a mobile application which also includes the View application means that the system should be responsive. According to Stephen Woods a non-responsive touch interface is the biggest thing that could spoil it and making the users perceive the interface as “broken” (S. Woods 2013). This is essentially important when coming to what the Apple touch interface guidelines call direct manipulation (Apple design guidelines, 2013). The user interface feels broken when direct manipulation isn't “direct” meaning that there exists lag in the system. In other words it's crucial that there is a minimum of lag in the high fidelity prototypes wherever direct manipulation should be present, one example being swiping between pages. However according to Stephen Woods the interface doesn't have to be fast it just has to be responsive and this is a big difference. He brings up one example with the old TiVo TV box, there were a lot of complains about it but none about it being slow despite it could take some time before the user pressed the play button and the box starting playing the content, and this was due to the familiar TiVo beep-boop sound. So immediate feedback (users expect it) is crucial in touch interfaces because the user aren't just pointing and clicking and waiting for something to happen (like in a wimp interface), the user uses gestures and for the usability's sake the system should not wait until the user has finished the gesture before the system starts to operate. Having instant feedback that tells the user that the system is listening and heard the request and is processing the user's request is important to make the interaction natural. If the user swipes the page the whole page should follow the users' finger and not wait until the gesture is complete, same thing goes for other gestures such as pinch to zoom etc. If an action takes some seconds to perform then there should appear a notification box telling the user that the system is processing the request, and if possible also a text to explain what is going on.

Other types of feedback are haptic/tactile feedback. For example in Android when a user press and hold down the finger against the touch screen the phone vibrates slightly to tell the user that it heard the users' request. Audible feedback like in the example with TiVo could also be used to give the user feedback; although not usable in this case, there is already too much noise in the healthcare. Visual feedback however and animations could be usable to give the user direct feedback and enhance the feeling of direct

manipulation. However in this type of application where it's a tool to aid in the healthcare industry, animations should not be overused and should be subtle to give a professional impression to the user. They could (if implemented wrong) slow down the user interface and distract the users from the main task which, in this case, is to keep an eye on all active alarms on a ward. They should also not get in the way of the users tasks or slow them down. They should also be consistent throughout the application to not confuse the user.

With the above in mind (also pointed out in both Android and iOS the guidelines) the **“Only show what I need when I need it”** (Android design guidelines, 2013). guideline is very important. The user will only interact with the system in short bursts and must have a clear overview of all information and hide all unnecessary information that is of minor importance at the time.

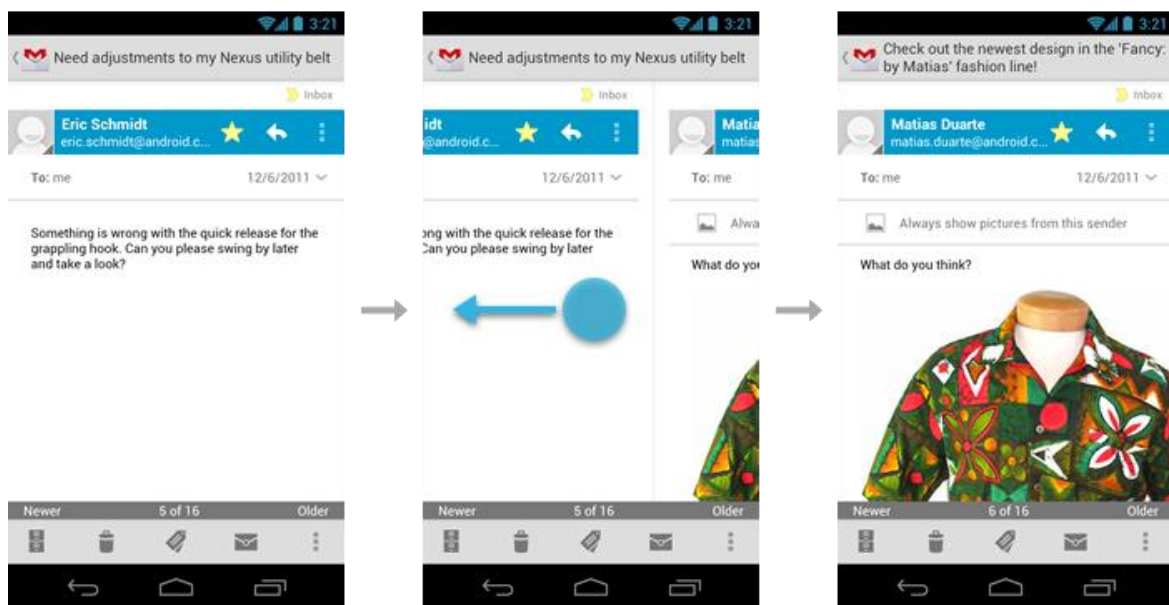


Figure 2: showing an example of direct manipulation. Here it could be noted that the page is following the user's finger as he does the swipe gesture. The image is taken from Android design guidelines <http://developer.android.com/design/patterns/swipe-views.html>.

People get overwhelmed when they see too much at once and both Android and iOS recommends breaking down tasks and information into small chunks and hiding information that isn't essential at the moment in order to facilitate for the user. Continuing on this theme, **“Pictures are faster than words”** (Android design guidelines, 2013) is another guideline that make the application easier to navigate and easier to perceive all information, through extensive use of pictures in the application whenever possible. Instead of explaining an idea to a user by having a long text, draw a picture instead, it gets people's attention easier and can be much more efficient than words.

Navigation is crucial in every application, if the users can't see a way to navigate around they will get frustrated and give up. In order to facilitate for the users building a mental model inside their heads the application should always make sure that the design principle **"I should always know where I am"** (Android design guidelines, 2013) is followed. Ways of doing this is providing some sort of menu highlighting in which tab/view the user currently is. Combine this with the demand that tasks should be done fast and one understands that there should be a minimum of navigation/minimum excise to get to desired information. There should always exist a physical menu for the user to navigate through or buttons to perform different actions, **"gestures should only be used to navigate/performing tasks as a shortcut"** (Android design guidelines, 2013) as very few people will figure them out right away. There should always be a simple way of getting to a desired view or performing desired task even if it means some extra taps. Simple gestures allow users to focus on the experience and the content and not the interaction. A common way to implement navigation in touch interfaces is to make use of a tab bar, it gives the user easy access to different views/subtasks or modes. This could be customized to fit the theme of the application however it should always be consistent.



Figure 3: Showing an example of a tab bar. The image is taken from Apple design guidelines.

<https://developer.apple.com/library/ios/documentation/userexperience/conceptual/mobilehig/UIElementGuidelines/UIElementGuidelines.html>

One of the most important aspects of a touch interface is making interactable objects big enough for the user to be able to interact with them. When comparing ordinary wimp interfaces with touch interfaces it's obvious that everything is much bigger on a touch interface, that's because the finger is a very clumsy device whereas the mouse is very accurate. Having this in mind it's crucial not to make the buttons too small but **"using the recommended touch target sizes"** (Android design guidelines, 2013). Different touch Operating systems have different sizes but they are all roughly the same, for example Android have 48 dp (around 7-10 mm) as the recommended touch target size on screen elements and Apple has 44 x 44 points as the comfortable minimum size of a tappable UI element. In some cases even this isn't enough but larger buttons are needed, for example in educational apps. Regarding points and dp this is not the same as pixels on a regular screen. On an ordinary iPhone for example one point is equal to one pixel but on a retina (high resolution) display one point is equal to two pixels (Apple designer guidelines, 2013). However the whole interactive object doesn't need to be this big as the recommended minimum size; as long as the clickable area is big enough, the visible button could be smaller than the recommended size.



Figure 4: Showing Android minimum target. The image is taken from Android design guidelines.  
<http://developer.android.com/design/style/metrics-grids.html>

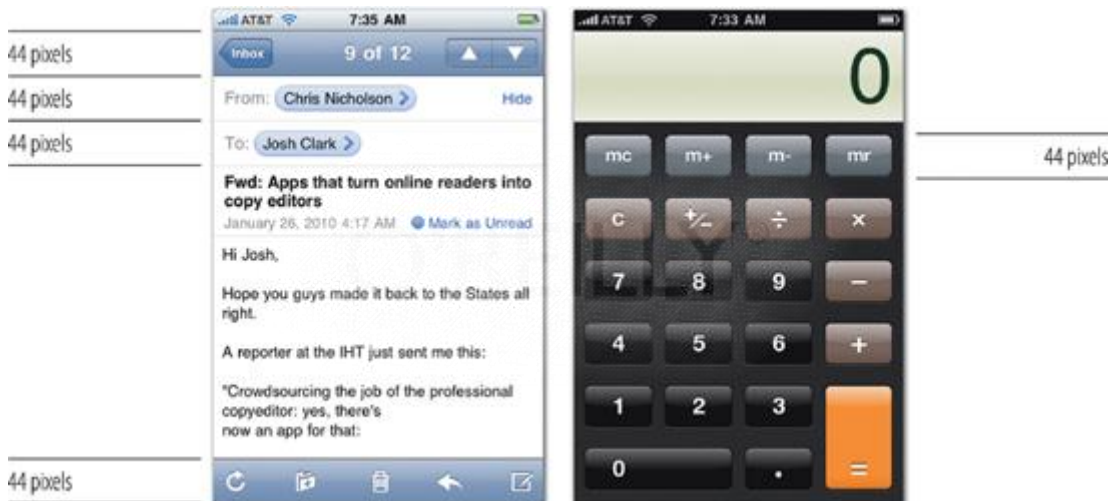


Figure 5: Showing iOS minimum interactive areas. The image is taken from Oreilly website.  
<http://answers.oreilly.com/topic/1691-what-is-the-optimal-size-of-an-iphone-touch-target/>

An obvious part of every user interface is its environment conventions. Every operating has its conventions, for example all wimp interfaces have buttons which could be pressed, windows have close buttons in order to make them disappear etc. Touch interfaces are no different, they have their conventions and since Apple made the touch interface popular by introducing iPhone in 2007 they have put the major standards that need to be followed in every touch interface in order to facilitate for the user learning it.

One classic convention is the Pinch to zoom; another is the swipe gesture which means to navigate to another view within the same information level.

When developing a new touch interface users will bring these conventions from their mobile touch interfaces so preserving these is crucial in order to facilitate the user's major transition from a small touch screen to a large one. This is why consistency within the app but also from other touch applications is also important. It enables the users to transfer their knowledge already gained from other applications and shortens the time it takes to learn the application, as described by Apple (Apple design guidelines, 2013). Further Apple explains how consistency means following standards for touch interfaces, but also that as already mentioned that the application should be consistent within itself as well as earlier versions of the application. Questions developers could ask themselves are for example; do the same icons mean the same things in the applications, can users

predict what will happen when performing a task etc. Other techniques of shortening the time it takes to learn an interface is to make use of well-known metaphors. One classic example that is explained in About Face (Cooper, Reinmann, Cronin, 2007), is the folder metaphor. People know that they can put things in a real folder and that's why people tend to try to put things in a virtual folder, however they should not be dragged too far, the metaphors should be obvious and well known in order to work best. Mobile apps solve this by bringing a set of already pre-defined icons that means certain things and are widely used in most applications, for example an icon of a floppy disk means save and a magnifying glass means search. Imitating these icons could be a smart move to shorten the time it takes to learn a new interface and not coming up with new ones or redesigning standard icons (think twice before doing this) unless it makes the task easier for the user. If creating new icons is a necessity make sure to use them in a consistent way throughout the user interface.

The below conventions (figure 6 and 7) are examples taken from Android guidelines (Android design guidelines, 2013) and are widely used in most touch interfaces, although names of the same gestures could vary between Android & iOS. For example a gesture is labeled pinch to zoom in iOS and pinch close in Android but it is the same. Apart from these conventions custom gestures could be implemented however there should be a good reason for this and it's normally not recommended by Apple unless it's obvious what gesture to do. For example if there is an old fashion telephone then the users might figure out right away to use their finger in a circular way of inputting the numbers to dial a number. On larger screens multi finger touch interaction gestures are more feasible than on small mobile screens due to the larger interactable space (Apple design guidelines, 2013).

Showing text in the user interface is quite normal and so is inputting text where convenient. When inputting text, a text view should be used. It is described as a rectangle of any height that enables auto scrolling when the text content gets too big for the text view. It should also enable a keyboard to appear whenever the user taps inside the text view (Apple design guidelines, 2013).



Figure 6: Taken from Android design guidelines.  
<http://developer.android.com/design/patterns/gestures.html>

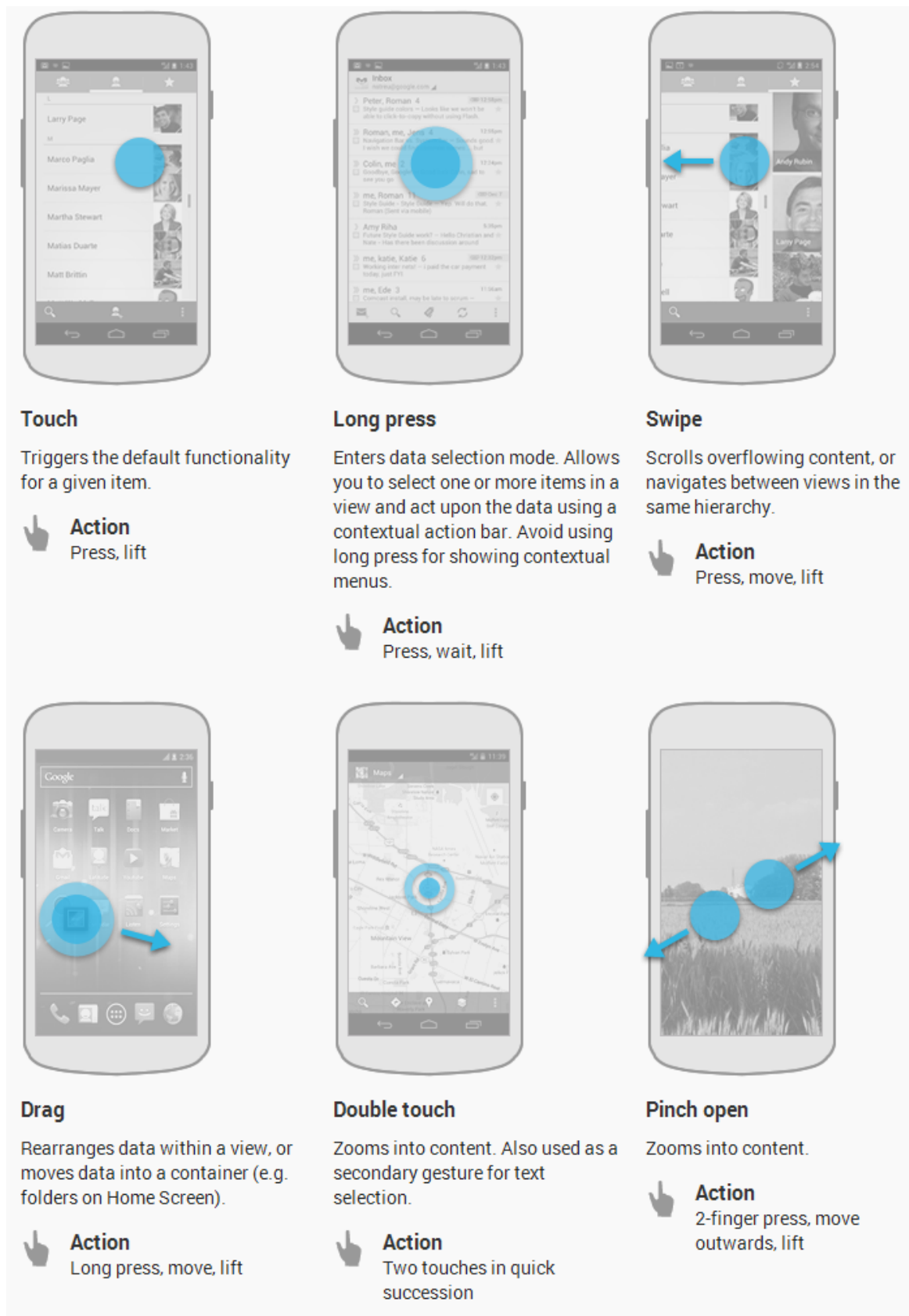


Figure 7: Taken from Android design guidelines.

<http://developer.android.com/design/patterns/gestures.html>

In a touch specialized application where the users don't have much time to interact with the application, the user interface should focus on the primary task according to Apple (Apple design guidelines, 2013). In other words less important features of the program should be dropped to make the users turn their full attention to the main feature. However according to Apple, this doesn't mean that other features couldn't be present but it means that the developers should always ask themselves is this critical information or functionality at the moment or could it be placed somewhere else or dropped altogether. A big mistake developers could do is to look at a larger touch interface and bring back old functionality from a wimp interface without first redesign it to better fit a touch interface. In this project only the most relevant features will be implemented in the prototype and other minor features is not covered in this application, also because of practical limitations.

Using the company's brand in an interface is an important aspect to remind the user of the company's identity. It works best when it's subtle and understated and not in the way of the user.

If there are a lot of data in the interface, for example a lot of beds and a lot of personnel then a search function could be in order together with a filter function. However a few guidelines should be followed if implemented. The bullet list below is taken directly from Apple (Apple design guidelines, 2013):

- *"Use auto complete while the user types, this decreases the amount of excise for the user.*
- *Use an optimized database in order to shorten the search time, if a search is taking too long time this creates a negative impression on the user.*
- *If the search is taking more than a second or two provide an information box telling the user that a search is ongoing and the progress state so the user knows that the search has not stalled. Also provide a possibility for the user to abort the search like a cancel button.*
- *Only display a search bar where search is possible, like for example at the top of a list and not bottom. Also only provide a search function in the main interface if it's a key feature" like for example in Spotify or iTunes.*

A user interface should be easy to use and a touch interface is no different. A user of such an interface has according to Apple neither the time or the desire to read through a lot of help text, they just want to get started using the application, so one should try to avoid using popup windows at the start of the application unless it's crucial for the user to understand the user interface (Apple design guidelines, 2013). A touch interface should be especially easy to use; however in this case users will probably get some sort of demonstration of the interface. It's supposed to be a helpful tool in the healthcare so



most functionality should not be compromised for the sake of usability but try to find a balance between them. Still the interaction with the interface should be intuitive and all helpful texts/images should be placed in the back but still easy to find and pop up whenever the user desires it by for example pushing a button.

Developing for tablets is another story compared to developing for phones. In mobile interfaces there is a constant challenge to get everything to fit the small screen which in turn leads to creating multiple pages, one for each navigation level in the information hierarchy. With a larger screen comes the possibility to flatten the information hierarchy by at least one level by making use of what Apple call “Split Views” (Apple design guidelines, 2013). This makes it easier for the user to get more information in one view instead of having to do unnecessary navigation which in turn creates excise for the user. Still the “less is more” rule is still in practice, only show the most important information in the different split views. The standard is that the navigation part is in the left pane and that the detailed information is in the right pane. One example is the mail application on both Android (Gmail) and iPad, where the inbox is displayed in the left pane and the detailed mail is displayed in the right pane. Some guidelines (the bullet list below) regarding the split view taken directly from Apple (Apple design guidelines, 2013):

- *“Don’t make the right pane narrower than the left pane*
- *Don’t display a navigation list in both panes at the same time this makes it harder for the user to know the relationship between the panes.*
- *A marker in the left pane should be visible to indicate for the user to know where he is in the navigational hierarchy.”*



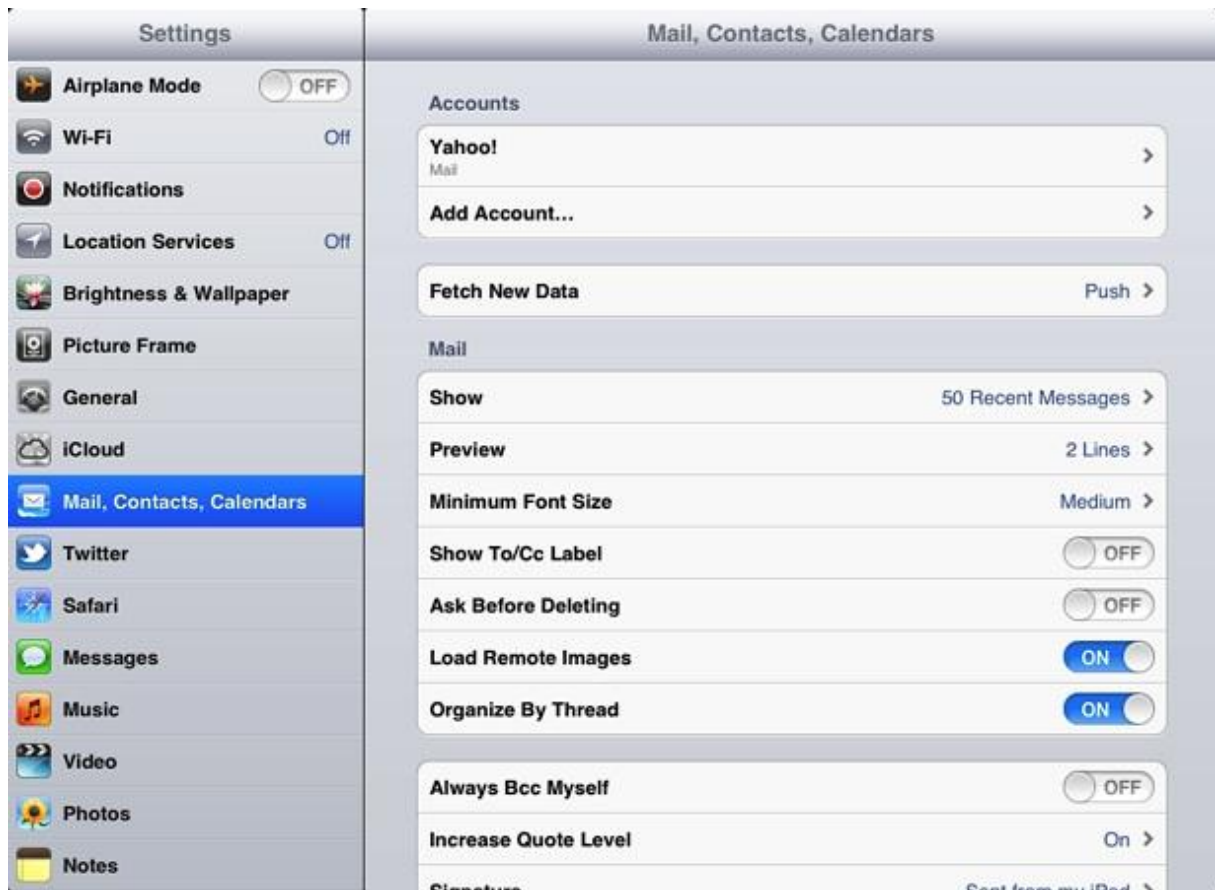


Figure 8: Showing an example of a split view on an iPad. The image is taken from Apple design guidelines. <https://developer.apple.com/library/ios/documentation/userexperience/conceptual/mobilehig/UIElementGuidelines/UIElementGuidelines.html>

## 2.2 LONG DISTANCE AND NATURAL USER INTERFACES

When designing an interface for a TV or a bigger screen, it is commonly referred to as a 10-feet interface, based on the distance from the screen that the user will be interacting with it. In this context the normal desktop applications are referred to as 2-feet interfaces, as the users are typically at a distance of 2 feet or less from the screen. An average touch interface, typically Smartphone, would have an even shorter average distance and, if referred to in this context, would probably be referred to as a 1-foot interface or maybe even on the inch scale. This creates difficulties for designers when the product combines two opposite extremes, large screen and touch. On the one hand, it should be designed as 10-feet UI for users passing by and/or viewing the application in the hallway. But on the other hand it should also be designed as a 1-foot UI for the users that approach the screen to interact with the interface through the touch screen. Representatives of the Customers even proposed requirements that the users should be able to view the application from an 8 meter (25 feet) distance, which needed to be investigated if it was at all viable to do. To solve this problem, a big part of the theory

study involved searching for similar projects, and research done on the subject. However, although the concept of touch screens is not new, until recently it has only been common on smaller screens, so the area of applications specifically made for both long distance viewing and close up interacting was very slim. As there was not any viable research done on this specific problem that was easily accessible, instead separate research for TV interfaces (10-foot UI) and tablet/Smartphone interfaces was studied extensively, and then conclusions were drawn as to what would be applicable.

A very good video of a presentation regarding the designing of a 10-foot interface was found during the literature study (Cardan, Wojogbe, Kralyevich, 2006). This provided a good basic understanding of what makes a 10-foot interface so different compared to a normal 2-foot interface as they refer to it. For example how long distance interfaces need to exclude excessive information and focus on the essentials etc.

During the research process, a review of YouTube XL was stumbled upon. Although this was actually a blog post, and not any scientific research, it was a very interesting example. Google is a very dominant player in the software industry, so analyzing their products is by no means wasted time.

Both information visualization and interaction flow had clearly been altered to optimize YouTube XL for large screens rather than desktop computers. Only essential information is displayed, and the navigation interactions are focused in one place optimized for a TV remote using up/down left/right input. Further examples of relevant products can be found in the background chapter.

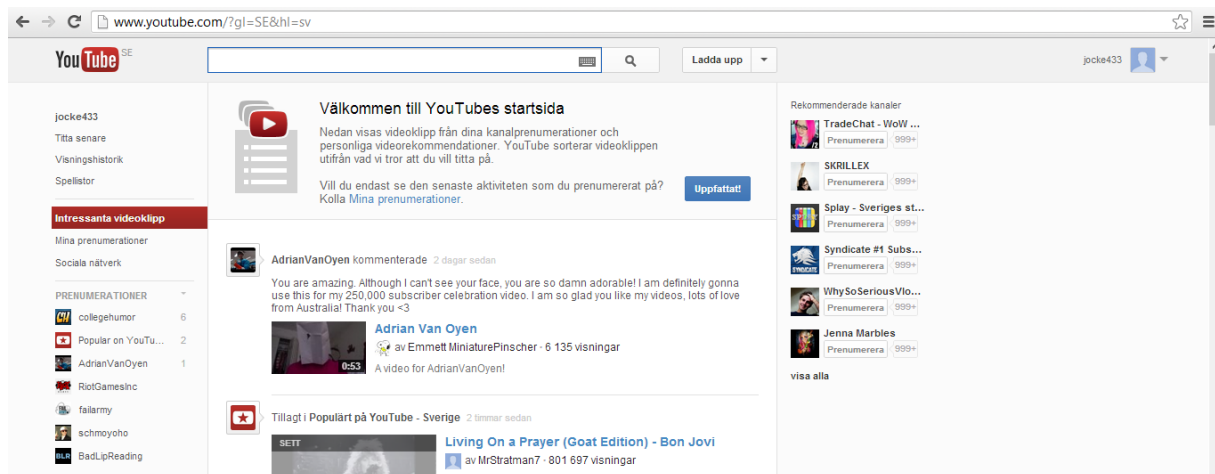


Figure 9: Showing a screenshot of ordinary YouTube.

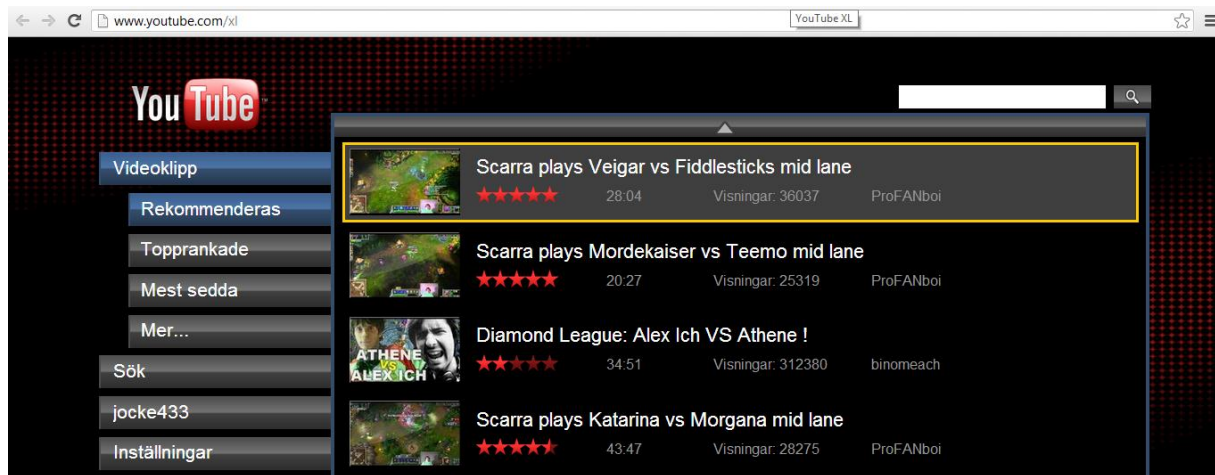


Figure 10: Showing a screenshot of YouTube XL.

One feature that was frequently seen in TV and DVD interfaces, was way they kept track of position in lists so the users could easily see where in the applications environment they were. This was often done by displaying some kind of frame around the “active” element or highlighting it with a different background color. In some cases a new window/field with additional information about the selected object, and it would then be linked with the object in question by having connected frames, or same highlighted background color etc. Over all these types of interfaces went out of their way to avoid displaying any excessive information that had not specifically been requested through interactions made by the user, which is something that was taken into consideration during the design process for this project.

TV interfaces were relevant due to the large screen platform, however having a remote as input made them less relevant for this project. A common term when talking about touch interfaces is Natural User Interface (NUI), and much research is being performed on this concept today. Results of such research were highly relevant for the project, and especially research involving Bill Buxton (principal researcher for Microsoft Research) was used. A video of Bill Buxton was published<sup>6</sup>, which capture the concept of the work being done on NUI very well. This video describes the results of studies done on simple writing with pen and paper, and to paraphrase from Bill Buxton's paper (Buxton, 2010); “The question should not be if the user prefer to use right or left hand for a task, but rather the division of labor between the two hands”. By letting the users reposition the paper with one hand while writing with the other, an interface takes a long leap towards feeling natural, which is the goal of NUI research. Having a Natural user interface is something that all touch interfaces should strive towards, and many of the guidelines that exists for different operating systems today, are made for just that. This was

<sup>6</sup> [http://www.youtube.com/watch?v=NcdrfacG\\_y4](http://www.youtube.com/watch?v=NcdrfacG_y4)

something that needed to be considered in this project, beyond the scope of existing guidelines, to create an interface that would feel natural to interact with.

### 2.3 SCREEN SIZE

Regarding the optimal screen size of the final application, the customer requested that the interface should be visible on a viewing distance of about eight meters, which would correspond to a screen size of at least 70 " screen (Myhometheater webpage 2013). THX requires that the eight back row seats in a theater have at least a 26 degree viewing angle (THX webpage 2013). Using these requirements a screen of about 170" would be needed to fill the requirements for an eight meter distance view. On the other hand the user would stand on a distance of about two feet when interacting with it so it's not feasible if the screen is too large at this point. Based on this, the preliminary belief was that a screen size of 50-70" would be optimal. This ensured that the application could be viewed from a relatively long distance, while not affecting the natural interface interaction negatively. However user testing would be required to be able to say with certainty, which was not possible during the project since the screen the users tested on was a 27" screen. The original View is optimized to be run at a resolution of 1280x800, however to get a more detailed looking application and because most new touch screens support full HD resolution the 1920x1080 (no need to support old screens that doesn't support touch) is to prefer when developing a new application. It's important to think about the preferred development resolution. Viewing an application developed to support 1280x1024 in full HD (1920x1080) makes the buttons smaller and makes them harder to hit in a touch interface. Also it has been demonstrated that viewing content on a display that occupies a greater visual angle (also referred to as field of view), increases the feeling of presence which is one parameter of how the user experiences the user interface (Lombard, Reich, Grabe, 2000). The monitor used in this project (Acer 27" LED T272HLbmidz) had a vertical and horizontal viewing angle of 178° (almost as perfect as can be, a user will never see the screen from a greater angle than 180° since its hanging on the wall) which fulfilled this requirement.

### 3 BACKGROUND

Many applications and products were studied in the research phase. As the project interface would be a unique mix of a desktop/touch/TV application, products from each of these separate genres was studied. One of the biggest influences of course being the original desktop view application, due to the scope of the project, as well as the administrative version called *Assign*. Other health care software applications were also researched but the relevance proved to be slim.

However one application developed by Cetrea<sup>7</sup>, was found during the project. It was used at a hospital at the regional hospital Viborg. It turned out to be somewhat relevant for our work. Here the user combined brief touch inputs with keyboard and mouse (as can be seen on figure 11). There are different versions of the application for example it could be used in surgical/patient ward and in emergency care. A video was found how this worked in practice and what was interesting was the time it took for the nurses to learn the interface. According to the video the nurses wanted to run the application in “live” mode after only two days of usage and also that the efficiency on the ward had gone up following the introduction of the application<sup>8</sup>. This was a strong indication that touch interfaces are possible to use in a healthcare context.



Figure 11: Showing the Cetrea “hybrid” touch interface, taken from the Cetrea website  
<http://www.cetrea.com/index.php/en/>

---

<sup>7</sup> <http://www.cetrea.com/index.php/en/>

<sup>8</sup> <http://www.youtube.com/watch?v=fqmozl39T9o>



More interesting related products were TV applications, or so called “10 feet UI’s”. Applications like Growl 10-foot style<sup>9</sup>, Windows media center<sup>10</sup>, YouTube XL (as previously mentioned in the theory chapter), and several average DVD menu interfaces and game consoles. Especially information visualization was considered when studying these types of applications, what information was typically displayed/not displayed on a 10-feet UI, how big the font of important/less important information was and which color combinations that were used.

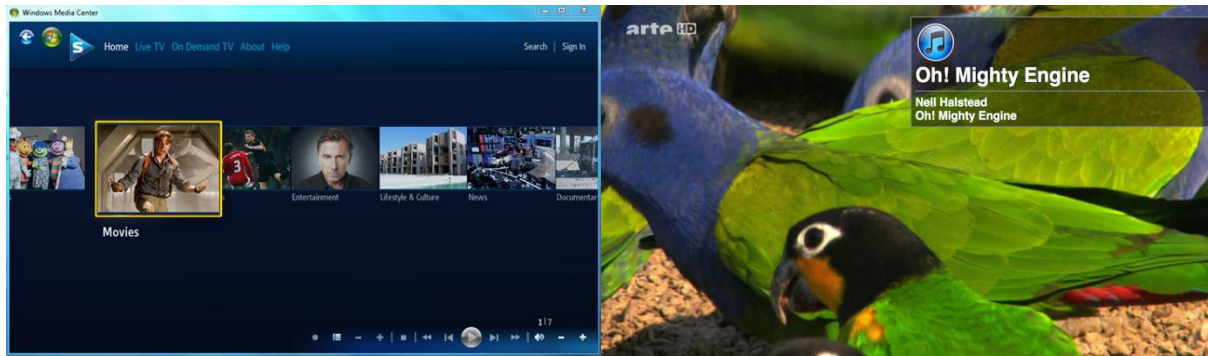


Figure 12: Showing a screenshot of Windows Media Center and an image of Growl 10-foot interface taken from <http://www.easyclasspage.de/mac/seite-16.html>



Figure 13: Showing the Playstation 3 interface, image taken from hexus.net [http://img.hexus.net/v2/gaming/ps3linux/pslinux\\_text\\_1.jpg](http://img.hexus.net/v2/gaming/ps3linux/pslinux_text_1.jpg)

<sup>9</sup><http://www.easyclasspage.de/mac/seite-16.html>

<sup>10</sup><http://windows.microsoft.com/sv-se/windows7/products/features/windows-media-center>

As for touch applications, there was a long list of applications (Smartphone/tablets and windows 8) that were looked into. Several mail applications were studied (Gmail, mail for iPhone/iPad, GO SMS Pro) for the purpose of understanding how to display long lists of information in viable ways. Games and other applications with many different ways of interaction through gestures and specific touch inputs were studied (Angry Birds, Cut the rope etc.) to learn how to hint or introduce these unique gestures in question. Finally touch applications with multiple views, to understand the visual architecture of such an application, and how the navigation could be handled.

## 4 METHOD

In this chapter we will describe the methods that we used in the project.

### 4.1 LITERATURE STUDY

At the beginning of the project an extensive literature study was made. The goal of an extensive literature study is to broaden the domain knowledge, and to get a better understanding of the problem. This study can be done in many different ways depending on the project, for example what to search for and which sources to use. In this section it will be described how the literature study was made for this project.

In this project the goal of the literature study was to find information about designing/converting an existing WIMP interface to a touch/natural interface. Additional information about designing interfaces for large screens was needed, as the general idea of a touch interface is that it is based on a Smartphone or tablet device, and in that case a lot of consideration goes into optimizing to fit information on the limited screen size. Ascom's local archives of information<sup>11</sup> was a big resource for the literature study, mostly regarding similar products from the Ascom Unite family which were in some way connected to View, but also user studies and research made by the company. This also included upcoming versions of View, including seminars with the several potential clients and other relevant speakers.

One of the methods used throughout the project for finding relevant literature was to go back and check what courses that had already been taken and see if they had some relevant course literature. A lot of books had been bought throughout the studies and some were used in this project. Among others used were Software Engineering 8 (Sommerville, 2007) and About Face 3 (Cooper, Reinmann, Cronin 2007). With these books as a starting point, related articles and research could be found, by searching for author or referenced sources.

Since the task was to convert an existing wimp interface under development into a touch interface, the course taken in mobile computing was in focus. Much literature about how to design a touch interface was taught there, among others the Android and iOS guidelines. Another early method for finding relevant literature was to check Adlibris.com website under course literature and see if there were any relevant literature. By doing so the book "Building Touch Interfaces with HTML5" was found and bought.

---

<sup>11</sup> Due to corporate privacy reasons, these cannot be listed



Various search engines were used, as well as Chalmers library database and Ascom itself had a vast collection of relevant information that was used. A few of the keywords and phrases used were; **touch interface, natural user interface, touch interface for large screens, TV interface, touch interface in the health care, windows 8 guidelines, converting wimp to touch, converting desktop application to touch, converting interface to touch, design guidelines**. As information was found, further keywords and phrases to search for were found with it, such as 10-foot interface and other more specific terms.

#### 4.2 BRAINSTORMING

As the work in this project is based on an already existing product, the focus of the brainstorming was to come up with additional functionality enabled by the touch input, and/or functionality and situations that could occur which would need the system to work differently due to the additional interaction possibilities. Also which functionality should be removed due to the requirement that it should be fast and easy to use in a stressful environment. The method was carried out according to how Alan Cooper describes it in About Face (Cooper, Reinmann, Cronin 2007). The only persons present at the brainstorming session were the two project members.

#### 4.3 SKETCHING

In the early stages of the design, several iterations of sketches were made in order to come up with a basic design for the interface, and get a sense of how the application could be interacted with. Starting with “the rectangle phase” as Alan Cooper calls it (Cooper, Reinmann, Cronin 2007), sketching containers and marking areas for objects in each view, finishing with detailed sketches which could be used as basis for the design of the first prototype. This method was used according to Alan Cooper's 6 steps of “Designing the interaction framework” (Cooper, Reinmann, Cronin 2007);

- Define form factor, posture, and input methods
- Define functional and data elements
- Determine functional groups and hierarchy
- Sketch the interaction framework
- Construct key path scenarios
- Check designs with validation scenarios

#### 4.4 MOCKUPS

After the sketching phase, mockups of each screen of the interface were made using the Balsamiq mockups tool<sup>12</sup>. A mockup is a screen blueprint of the actual interface and is made for the purpose getting user feedback if the page layout is suitable. It usually lacks color or graphics since it focuses on what a screen does not how it looks like. Also a mockup does not support any interaction.

#### 4.5 PROTOTYPING

Prototyping has been a big part of this project, and was a great way to get feedback on the design. Prototypes are often divided into two categories; Low fidelity and High fidelity (lofi and hifi). Fidelity refers to how interactive and realistic the prototype is. A lofi prototype generally consists of small parts of the program and not much effort is made to make it look good. This can be all from handmade sketches used with wizard of Oz technique (usability net, 2006), to fake feedback and interactions, to paper prototypes or even a very simple digital prototype. Hifi prototypes are typically an early version of the product, but not necessarily including all parts of the product. In this project the first real prototype was a hifi prototype. Sketching was used as a tool to visualize ideas, and some lofi sketch prototypes were made, but this was only an aid to creating the digital prototype, and was never intended for user testing, hence the first real prototype was the digital hifi prototype.

#### 4.6 INTERACTIVE PROTOTYPE

A high fidelity prototype was developed in a rather early stage of the project. This was due to the annual user testing night on Ascom called “future evening” (framtidskväll), which was an opportunity that could not be missed. The first interactive prototype was made with linked pictures, where each picture represented a view. For the users testing the application it would have felt very real, and been more or less indistinguishable from a real application.

#### 4.7 ASCOM USER TESTING

User testing is a critical tool for the design process in order to create a useful application. In this case there was an opportunity to test the prototype on a group of users that could potentially use the application in their future work, which made it even better. To quote an interaction design employee from Ascom “It's one thing if our application could be used by persons over 30 years old with a Master of Science degree within computer science, another thing if it could be used by an average person”.

---

<sup>12</sup> <http://balsamiq.com/products/mockups/>

The tests were made with one test person at a time. The test person, one facilitator and one observer were present, the observer also video recorded the interaction with the screen, for more depth analysis. The test person was presented with the touch screen showing the prototype and also the native windows 7 keyboard with its docking state in “float”. He was encouraged to “think loud” (Lewis, Reiman, 1994) while interacting with the interface and was then given some tasks to perform.

The user group consisted of six professional nurses, and after a brief welcome and dinner they spent two hours testing potential future Ascom products. During this time a 15 minute session was done with each nurse, in which they performed a set of tasks on the developed prototype, and then answered a questionnaire about it and the tasks. Each session was video recorded in order to really capture the interactions the users did or tried to do, and which functionality they were expecting. Additionally the think aloud-protocol was used, in which the user’s had to verbalize what they were thinking as they complete the tasks.

The test person was asked to perform five different tasks (See Appendix B) and afterwards he was asked to fill in a questionnaire (See Appendix A) providing us with valuable feedback of what he/she thought of the system.

#### 4.8 QUALITATIVE USER TEST

Qualitative user testing was used in the project. User testing is used to test new interfaces to find flaws in the design and one usually distinguish qualitative user research and quantitative user research. In qualitative user testing a fewer number of user testers are used and more time is spent with each tester, in return they each give a lot more detailed feedback than a quantitative user research does. Developing user interfaces is a social science, every person is different, having different knowledge and behaves differently compared to studying nature laws where one can predict what will happen. Qualitative user testing is often done with “face-to-face” interviews with the user, compared to the user just filling out a form, which creates a wider understanding of the user’s behavior/emotions/reactions (Cooper, Reinmann, Cronin 2007).

#### 4.9 SELECTION

Ascom had a solid collection of documentation on user testing that they have done when they were out in the healthcare industry, “shadowing” nurses to see how their everyday

life were going about, to get user input on how the view application should look like. We used some of this information in our prototype to get inspiration.

#### 4.10 METHODS NOT USED

This section describes some of the methods that, despite the fact they were relevant for our project, was not used in the end. Information about these methods can be found in About Face 3 (Cooper, Reinmann, Cronin 2007).

**Quantitative user research** was not used in the project. The reason for this was because a lot of user research was available about the potential users, their behavior/needs etc. The desired result of this was direct deeper feedback on the first hifi prototype and how the users interacted with it. This wasn't something that could be gotten from a form; it was needed to see the users actually using the interface in order to find out flaws in the design and how gestures would work on a larger screen. There was also lack of time to conduct quantitative user research since getting a hifi prototype ready for the annual Ascom "Future Night" was a high priority. Max information from each user tester was desirable.

**Personas** were not constructed during the design process; they were still used and were made available to us from a master thesis written by Linnea Fogelmark (Fogelmark, 2008).

**Shadowing** nurses was not a method used in this project. Although invaluable where we as developers don't know much about future users, Ascom had a big collection of information from when they had shadowed nurses that could be used.

## 5 DESIGN PROCESS

Here the design process is described in depth. In general it went through four main stages the first being the prestudy/brainstorming phases where the current view was analyzed and sketches of the prototype were made. Later on the first interactive high fidelity prototype was made with Justinmind Prototyper<sup>13</sup> and tested at the Ascom future evening and was updated after the comments gotten from the test subjects. A more realistic prototype was made from scratch using HTML/PHP/CSS JavaScript/JQuery, and then tested, this time visiting a hospital in Uddevalla and getting valuable feedback. Finally another final revision of the prototype was made.

### 5.1 PRE STUDY

One part of the job was to investigate what type of screen to choose to develop on. At first we planned to go for a touch screen overlay and buy the led screen separately and in this way get the preferred screen size to develop on immediately (~ 50 " according to our research). We had other alternatives such as renting a Microsoft table or a large touch screen. Both these turned out to not be viable options for various reasons, but the most obvious was the price. We settled for an Acer T272HLbmidz 27" touch screen monitor<sup>14</sup>, and although it was not the perfect size according to our pre-study it had to do. As the resolution is the same, it will not matter for developing purposes, but for testing it a full size screen would be optimal, a compromise we had to make.

Ascom followed a standard for usability in medical equipment's called ISO-62366 and also ISO-60601-1-8 for all colors in the product so we followed some of these when developing our prototype for all alarms.

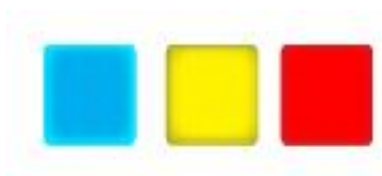


Figure 14: Showing an image from Ascom displaying the colors for the different alarm levels.

According to a report written by Linnea Fogelmark (mentioned earlier) that made her master thesis at Ascom, hospital staff could notice patterns in patients alarm history for

---

<sup>13</sup><http://www.justinmind.com/>

<sup>14</sup><http://www.acercomputer.se/ac/sv/SE/content/model/UM.HT2EE.001>

about eight hours back, indicating that in some sense heart attacks could be alerted in advance by looking at a patient's heart rate history for eight hours back. That's why we propose that the alarm history should be showing all alarms for the past ten hours to get a buffer. When presenting this to some user testers (Nurses) they proposed going even further saving up to about two-three days of alarm history claiming even this could be of great help in the healthcare although this is not researched. One problem with this with today's technology is that the history of the hospital life support machines would have to be reset manually every time a patient is checked out and a new patient is placed in this machine in order to avoid displaying the old patients alarm history in the new one. However this wasn't something that we took into consideration when designing our prototype. When researching the web among others, the NUI Group (NUI Group webpage 2013) was stumbled upon (this was also a tip we got from our supervisors at Ascom). From here interesting articles and reports could be found, among others a report about Multi Touch Technologies describing how to practically implement touch interfaces.

During our prestudy we came across a healthcare application made by Cetrea that was a hybrid application with both touch and mouse/keyboard as input. While this was highly relevant due to the fact that this was a large touch screen interface it was very different in that sense that it was optimized for longer use and not easily visible from a longer distance. It was partly designed as a desktop application with much more information cluttering the interface as well as narrow spaces, one example being the drop down lists being designed very much like a desktop application.

## 5.2 BRAINSTORMING

When the pre study was completed, we had a brainstorming session. As there was a first working version of view for desktops (Q1 version), we already had the groundwork of the design done, so the focus with the brainstorming was to come up with ideas on how to improve the system with the added interaction possibility from a touch screen. Using personas from Linnea Fogelmark report (Fogelmark, 2008) we started writing down scenarios that might occur, and situations where the application could assist them. A very early idea that we found very simple, but yet extremely helpful, was to integrate Fass (Fass webpage, 2013 ). We assumed that many hospitals have terminals where the staff can search for medication and find out detailed information about it, so why not let them do this directly in our application (This was later confirmed in the user tests that it's used in some of the hospitals).

Another very early and simple idea was to show the staff and information about them. An example of a scenario we came up with was that a user wanted to contact a specific nurse, or just wanted to know whom the responsible nurses on the ward were. So called runners (nurses that work on different wards at the same time) might be on shift and

have a hard time keeping track of all co-workers. Additionally we wanted some way of showing which nurses were working on the current shift as well. Several ideas of how to show this were brainstormed, however none of them were actually used, as another way of displaying them emerged later in the design process.

Considering the actual alarms, an idea emerged to have different views for showing alarms. This was inspired by another product called Assign, which is used to assign nurses to patients/beds. By letting the user switch to Bed view, the application could show additional information about that bed/patient, and by letting the user log in, even confidential information could be displayed. The idea of confidential information was however put on ice, due to the fact that there was not a viable solution for identification. By letting the user log in to show the information, it could potentially be accessed by someone unauthorized if the user suddenly had to run to an ongoing alarm for example. The design still supports this functionality though, but until a viable authorization is found, it remains disabled.

We also wanted to evolve the standard Listview, so that it took advantage of the added potential of touch interaction possibilities. So an idea was born, about letting the user show advanced information about each ongoing alarm by tapping on it in the alarm list. This concept changed several times during the design process, but this was the original idea to follow the split screen guideline to flatten the information hierarchy to make the interface easier to overview.

### 5.3 SKETCHING

The next step we took after the brainstorming was to start sketching, or rather to start *designing the interaction framework* as according to chapter 7 in About Face (Cooper, Reinmann, Cronin 2007).

#### 5.3.1 DEFINE FORM FACTOR, POSTURE, AND INPUT METHODS

First step in this method is to define form, posture and input. Our application will be a touch application viewed in very high resolution on a 42" or larger screen. The application will mostly be used in short sessions and possibly under time pressure. The input method will be the touch screen with several possible gestures along with an on-screen keyboard.

#### 5.3.2 DEFINE FUNCTIONAL AND DATA ELEMENTS

The second step in this method is to define elements, functional and data.

### **The data elements that we identified at this point were:**

- active alarms
- accepted alarms
- alarm details
- beds
- staff (nurses and other titles)
- wards
- comments
- patient information
- Listview
- Bedview
- Staff view (later on renamed to personnel view)
- Fass view (integrated browser)

### **The functional elements that we identified were:**

Main (universal for all views):

- navigating to another view (bed, Fass, list, staff)
- viewing all active alarms

List view:

- selecting an alarm from the list of active alarms
- closing detailed information (returning to default list)

Staff view:

- selecting a ward to display nurses
- selecting a nurse to display detailed information

Fass view:

- toggle the on-screen keyboard



- input string to search for

Bedview:

- selecting a bed
- closing detailed view(returning to default Bedview)
- toggle on-screen keyboard
- input text as comment
- authenticate (log in)
- input username and password
- post comment (save the text from input)
- remove comment

### 5.3.3 DETERMINE FUNCTIONAL GROUPS AND HIERARCHY

At this stage we divided the interface into several groups/containers. First the main page was divided into three groups; “Main-mid” where the active view would be displayed, “Main-top” for the navigational tools and “Main-left” for a miniature alarm list. Then there were the four potential active views; Listview, Bedview, Fassview and Nurseview. Further, Listview and Bedview were divided into Listview, Listview detailed, Bedview and Bedview detailed.

Functional elements colored in blue and data elements in black.

Main-mid container:

- active view (bed/list/Fass/staff)

Main-top container:

- navigating to another view (bed, Fass, list, staff)

Main-left container:

- viewing all active alarms
- active alarms

Listview:

- selecting an alarm from the list of active alarms
- active alarms
- accepted alarms

Listview detailed:

- selecting an alarm from the list of active alarms
- closing detailed information (returning to default list)
- active alarms
- accepted alarms
- alarm details

Bedview:

- selecting a bed
- authenticate (log in)
- input username and password
- beds
- wards
- active alarms

Bedview detailed:

- closing detailed view(returning to default Bedview)
- toggle on-screen keyboard
- input text as comment
- post comment (save the text from input)
- remove comment
- comments
- patient information
- active alarms
- accepted alarms

- alarm details

Nurseview:

- [selecting a ward to display nurses](#)
- wards

Wards (contains):

- [selecting a nurse to display detailed information](#)
- staff (nurses and other titles)

---

#### 5.3.4 SKETCH THE INTERACTION FRAMEWORK

The fourth step is where we started with the actual sketching. The sketching is an iterative process in itself, starting with very rough sketches marking out where things should be placed. This first iteration is referred to as the “rectangle phase” in About Face. At this point in the design process, we had not yet decided which platform the final prototype would be made in, but we knew that it was going to be either and Android/Windows 8 or a web/desktop application. So the choice was between HTML, Java (Android), Python, or Java/HTML (Windows 8). But since we knew it was going to be a touch application we decided to look into guidelines for both Windows 8 and Android, and we also included iOS even though Ascom did not want an iOS application. iOS and android are optimized for small screens (phone/tablet) so some of the guidelines (far from all) were not suited for a big screen interface; hence the main source regarding the menu interaction became Windows 8. One good example of this is the top and right menu, which was designed according to the standards for Windows 8 applications. Even though we were not sure it was going to be a Windows 8 application, this was a good way to place the navigational tools because it saved a lot of space, and as the alarms should be seen from a long distance, this was a very important decision.

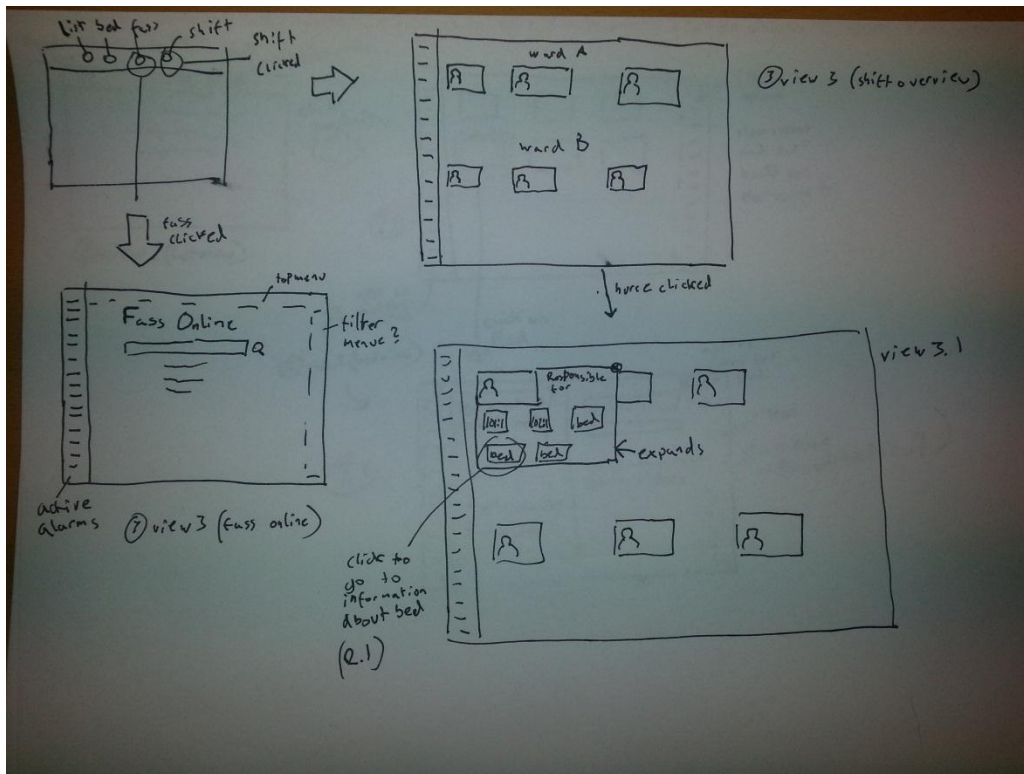


Figure 15: Showing a sketch of the navigation inspired by Windows 8 guide lines.

The main function of the application was to display alarms, and with the added functionality, the application would sometimes be displaying a view that did not display all or any alarms. Fass online or staff view would not display any alarms at all for example. With this in mind, the design choice was made to add a list on the left side, where all active alarms (sorted by priority) would always be shown, regardless of which view/state the application was currently in. This list, however, was not designed to be visible from a long distance, but rather as a 2-feet feature. The reason for this was that the user would be standing within arm length of the screen interacting with the application when this particular list would be needed. The default view when the application was idle would be Listview which was designed to be visible from a longer distance.

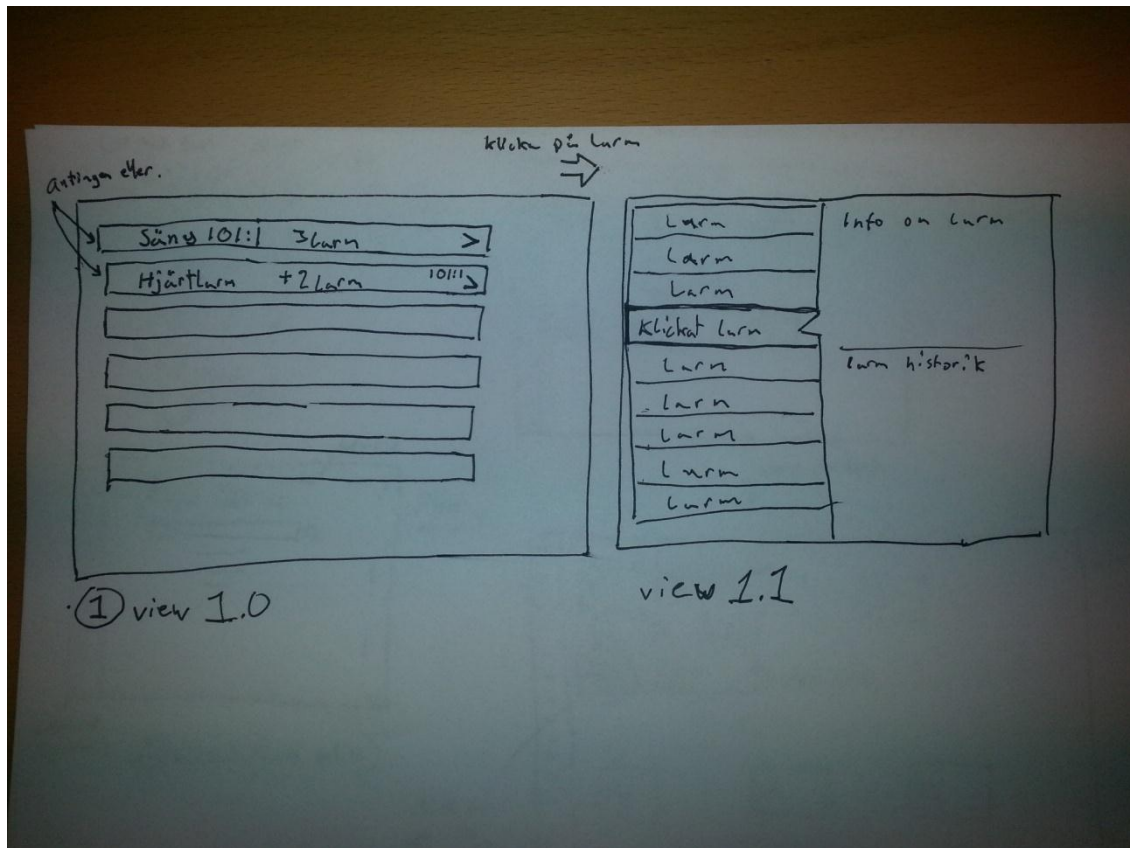


Figure 16: Showing an early sketch of Listview.

Listview's design was inspired by Android/iOS standards, as seen in figure 13. When an alarm in the list is clicked, a detailed view with more information about this alarm would appear to the right. This detailed view would be connected to the clicked alarm, as seen in figure 13. If another alarm is clicked, the detailed information would change and it would become connected with that alarm instead. This concept originated in interactive TV systems, such as TiVo and DVD's where the user was navigating through the up/down left/right buttons on a TV remote, and was later adopted by Smartphones and smart TV's. As our application is little of both, it would fit right in.

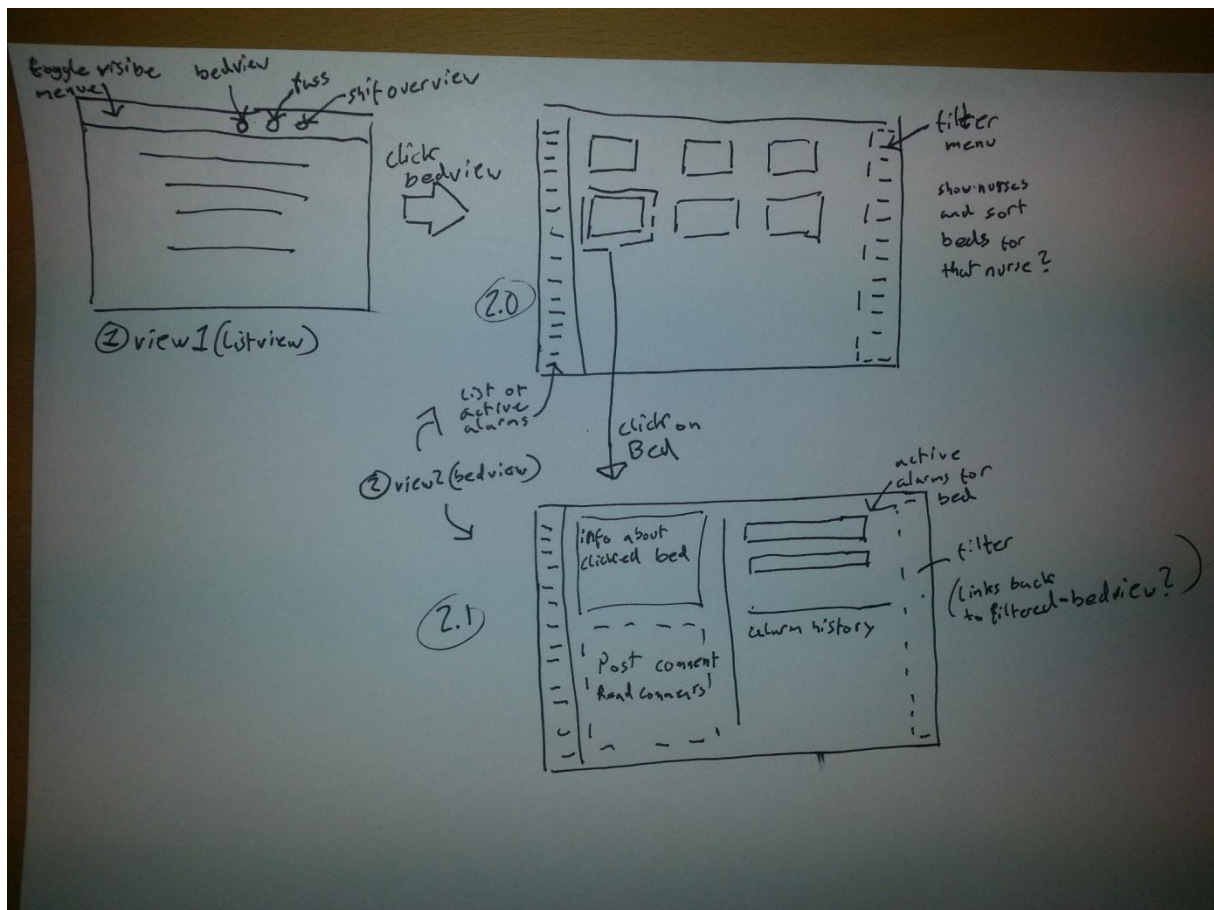


Figure 17: Showing an early sketch of Bedview and detailed Bedview.

### 5.3.5 CONSTRUCT KEY PATH SCENARIOS

Key path scenarios were constructed in parallel with sketching the interaction framework. We sketched a rough design of a view, and then we constructed the key path scenarios for the view in question, then reevaluated and sketched a more detailed version including the key path scenarios, which is shown in figure 12-14 above.

### 5.3.6 CHECK DESIGNS WITH VALIDATION SCENARIOS

Validation scenarios were done verbally, as a session where we tried to poke holes in the design. One design flaw that was discovered during this process was the patient information on detailed Bedview. The user was supposed to log in to access confidential information about the patients, however if a high priority alarm came in while the user was logged in, and the user needed to run to take it, the information would be available for the next user. And as this is an application that might be placed in public areas, unauthorized users might get access to information they are not allowed to see. So we put this functionality on ice (metaphorically) until we would find a viable solution for

authorization. Instead we let the user authorize in order to post or delete a comment. This action would only authorize the user for that specific action and would not be kept logged in.

## 5.4 MOCKUPS

Mockups were created with Balsamiq Mockups as an aid to further visualize the interface and were used together with sketches when designing the hifi prototype. Below are two of them.

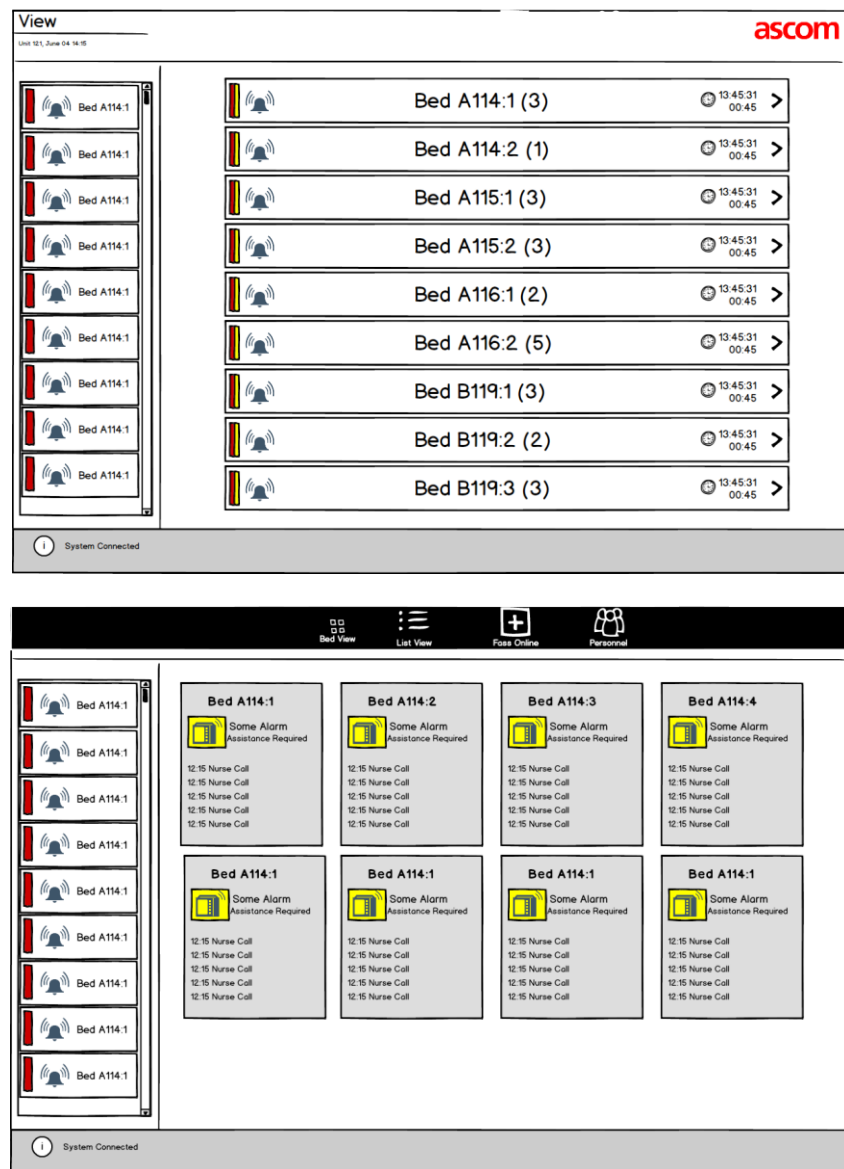


Figure 18: Showing two examples of mockups made in Balsamiq Mockups. First being the main Listview and the second being Bedview with the tab bar visible.



## 5.5 FIRST PROTOTYPE

We were offered the chance to participate in Ascom's "future evening", which is an evening dedicated to user testing. Here several nurses come to visit and as the name hints, they user test potential future products. As these nurses are part of the actual intended user group, this was an incredible opportunity for us to make some sort of user testing. In order to get as much useful information as possible out of this evening, we started developing a hifi prototype very early, without creating any lofi prototypes first. It was pretty clear to us early on in the design process that since it should be quick to perform tasks, the users were often under stress and people under stress are more likely to perform errors (Sommerville, 2007). Additionally, since it's kind of (not technically) a critical system, the operator reliability (how likely it is that a user performs an error while using the software) should be very high. We spent some time discussing what software to use in order to get a hifi prototype done in the short time we had, and we were considering several different prototyping tools for this;

Napkee<sup>15</sup> was one of the applications we considered, and it is working side by side with Balsamiq<sup>16</sup>, already used in the project to create mockups. But since this is rather lofi prototyping, this made us exclude it. The only reason to consider Balsamiq was the possibility to let Napkee convert our prototype into interactive HTML code, which later could be modified.

Invisionapp<sup>17</sup> was another consideration. In this application the starting point was creating an image, and then using image maps<sup>18</sup> to link different parts of the image to another image etc. The main problem with this application was the limitation of only 1 project per user, and the only interactions the user could do were point and click, hence this application was of no use since the task stated was to develop an interface with which the user should be able to use more gestures than just the tap gesture.

---

<sup>15</sup><http://www.napkee.com>

<sup>16</sup><http://balsamiq.com/>

<sup>17</sup><http://www.invisionapp.com>

<sup>18</sup>[http://www.w3schools.com/tags/tag\\_map.asp](http://www.w3schools.com/tags/tag_map.asp)



Flairbuilder<sup>19</sup> and Hotgloo<sup>20</sup> were two other applications considered, however with a very short trial period (only 15 days), together with the same argument as Invisionapp regarding interaction possibilities; they had to be excluded as well.

Finally the application of choice was found, an application that suited the purpose and needs very well. An application with a 30 day trial and very easy to use, yet extensive functionality called Justinmind<sup>21</sup>. It was able to create components and data tables on each screen in addition to just placing image maps on a picture. The most important functionality of this application though, was the compatibility with touch gestures. Justinmind is also able to export the project to HTML code which was a great feature for us to be able to use our prototype after the trial was exceeded.

In order to simplify the development process we early on established that the resolution to go for was the full HD resolution, 1920\*1080 and not trying to make an adaptable interface that would support multiple resolutions.

Another major issue that we faced was how big different objects should be on the screen since most documentation we encountered was regarding mobile/tablet interfaces (11"), or in the case of Windows 8 the standard desktop screen sizes (~11-24"), and what we designed for would be a much larger interface (~50"). So what we did was that we followed the guidelines for previously mentioned platforms when coming to different sizes of objects and then scaled them up a little less than proportional so that something on an 11 inch tablet that would be say 10 mm, would be something like 30 mm on the big screen.

Also some mobile guidelines we followed while developing our prototype were contradictory among different operating systems, for example iOS suggested that every list item should have an arrow pointing in a direction suggesting that there is "more" content while Android said that there should be none. When this happened we just went with what we personally thought was the most intuitive and in this case we went with the iOS guideline and added arrows to our list items in Listview.

---

<sup>19</sup><http://www.flairbuilder.com/>

<sup>20</sup><http://www.hotgloo.com/tour>

<sup>21</sup><http://www.justinmind.com/>



Figure 19: Showing the arrow on a list item, following the iOS guidelines and not the Android.

### 5.5.1 KEY GUIDELINES FOLLOWED

Below are some key guidelines taken from different mobile operating systems that we have followed during our development.

- **I should always know where I am;** was a guideline that we tried to follow and implement in our application. It means that we as developers should give people confidence that they know their way around. Transitions were used where it was possible to implement in our design, for example when you close detailed Bedview and when you open a list item in Listview.



Figure 20: Showing our implementation of "I should always know where I am" and in this case the user is in Listview.

- A **split view** was used in Listview to make it easier for the user to overview all the alarms.

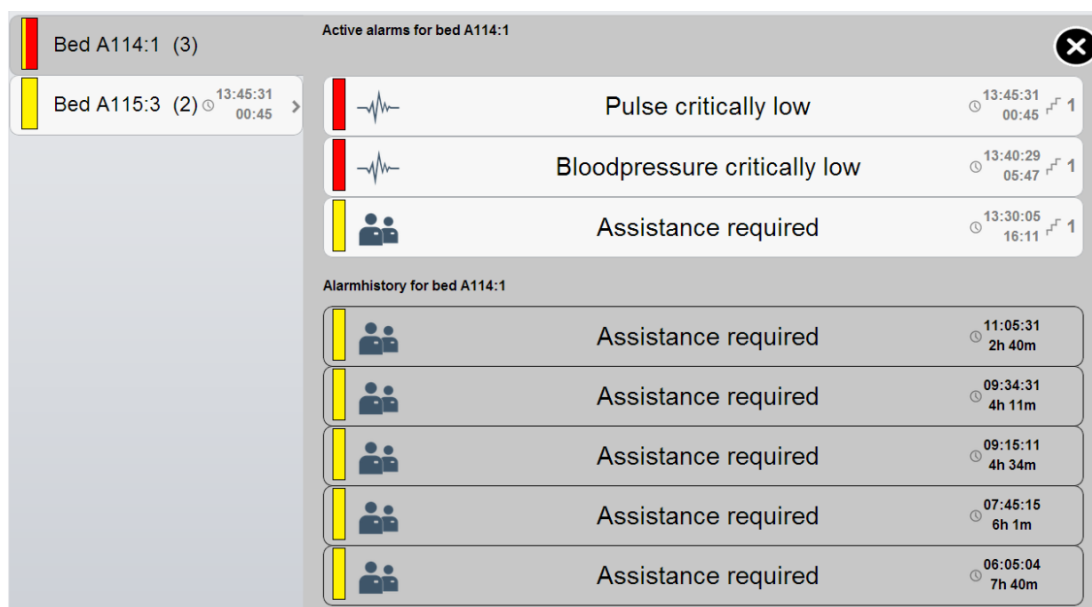


Figure 21: Showing the first version of the split view.

### 5.5.2 WINDOWS 8

The following were some Windows 8 key guidelines used in this stage of the design process.

#### NAVIGATION DESIGN

Nurse Menu and top menu are inspired by Windows 8 navigation design guidelines. The following bullet points are taken directly from Windows guidelines (Windows design guidelines, 2013):

- *“Swiping from the bottom or top edge of the screen reveals the navigation and command app bars.”*
- *“Swiping from the right edge of the screen reveals the charms that expose system commands.”*

Originally the nurse-menu was the bottom bar, however to use the space of the screen more effectively we decided to replace the right side menu, containing system commands, with the nurse-menu. If the decision would be made to implement the application as a commercial Windows 8 app, the nurse-menu would have to be converted to bottom bar though, to follow the Windows 8 standard **“Navigating with the edge swipe”**.

#### TOUCH AND CLICK FEEDBACK

Windows 8 has a very good visual feedback<sup>22</sup> standard for touch applications, where the object shrinks in size 5% for 1 second when activated (tap), which should most definitely be used if this application were to be implemented as a Windows 8 app. JavaScript for this is included in the developing tools for Windows 8 so it would be easily done. For our first prototype this was never implemented due to the technical limits of the software (Justinmind), but in the final prototype this was attempted. Due to technical difficulties and time limitations we had to remove the feature though, but it is suggested as part of the design.

---

<sup>22</sup><http://content1.catalog.video.msn.com/e2/ds/b2430d2d-73e7-4b23-9537-36b4d63b7365.mp4>

## 5.6 ASCOM FUTURE EVENING (TESTING)

This was the evening that extensive testing of the first hifi prototype was conducted and how it went is described below. The key research questions that we wanted to get answers to were the below:

- Is the system easily understood?
- Is the system showing enough information for the test users to be able to perform their tasks?

The participants for this event consisted of users in the health care with mixed experience of computer/touch interfaces and experience in working on different wards. The iterative prototyping process was used, first creating sketches, mockups and then created a hifi prototype, tested it and then redesigned the parts that weren't good.

### 5.6.1 PREPARATIONS

A prototype was constructed with the software called Justinmind Prototyper, as described in the previous chapter, and was based on screenshots taken from the original View application constructed by Ascom Wireless solutions. This was done to enhance coherency (because people using corridor view should also be able to use the WIMP version of View), but also since a lot of effort has already been made to discover how this application best would be constructed. Employees had shadowed nurses in their daily work to find out how they perform their tasks, color codes must be followed in order to categorize it as a health care equipment (ISO Standard 60601-1-8)<sup>23</sup> in United States.

---

<sup>23</sup> IEC 60601-1-8 Ed. 1.0 b:2005

Medical electrical equipment

- Part 1-8: General requirements for safety

- Collateral Standard: General requirements

- Part 1-8: General requirements for safety

- Collateral Standard: General requirements equipment and medical electrical systems

Standard IEC 60601-1-8, 2006

[http://admin.altran.it/fileadmin/medias/IT.altran.it/Images/Publication/TechnologyReview/Technology\\_Review\\_n.8\\_Ottobre\\_2012\\_P.Sessa.pdf](http://admin.altran.it/fileadmin/medias/IT.altran.it/Images/Publication/TechnologyReview/Technology_Review_n.8_Ottobre_2012_P.Sessa.pdf)

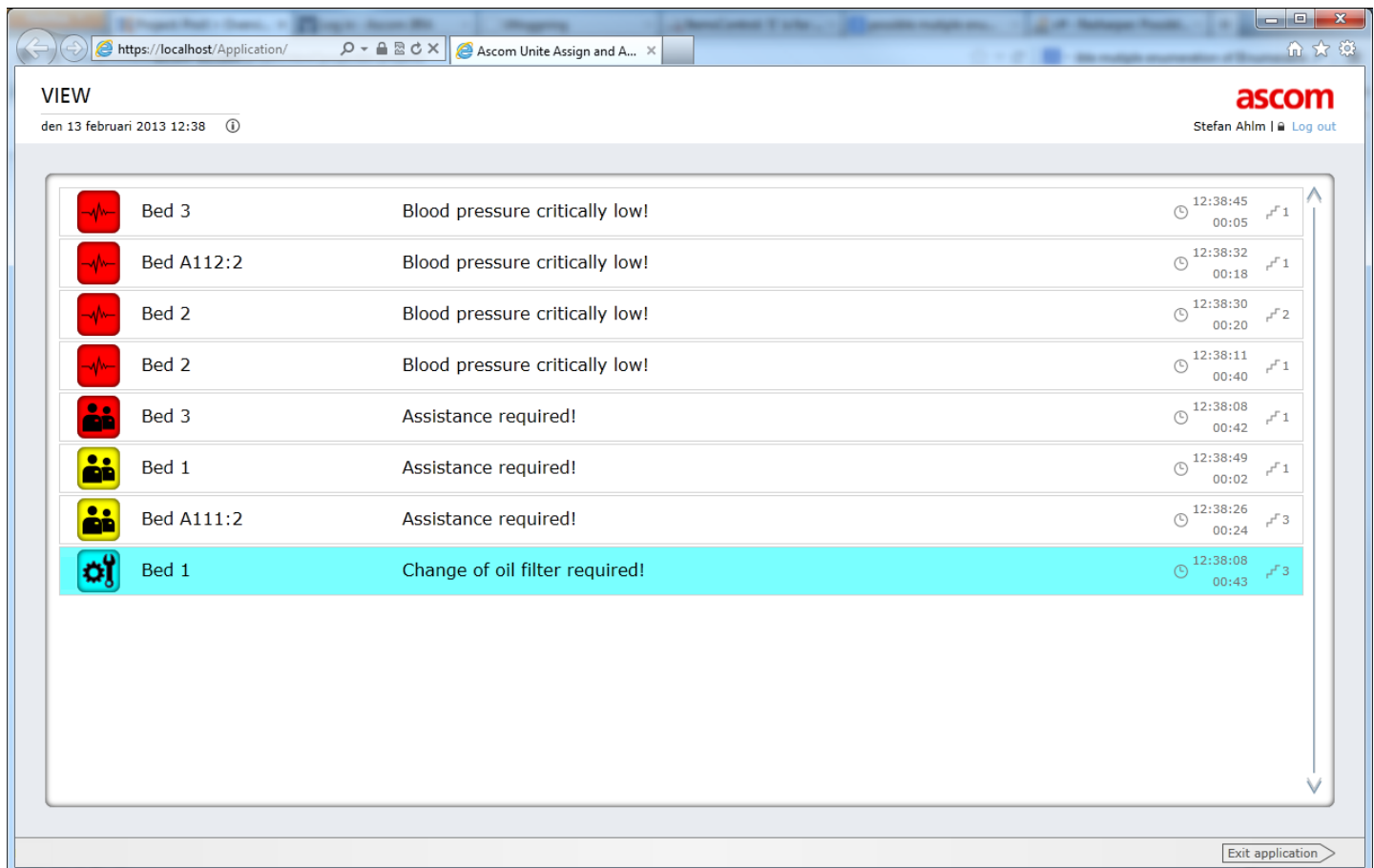


Figure 22: Showing an image of the Desktop View (Quarter 1)

### 5.6.2 TEST PERSONS

The test was performed by six test persons working in the Swedish health care and was designed to evaluate the physical interaction with the interface. The test persons had the following characteristics.

Test person A	
Age	49
Gender	Female
Profession	Specialist Nurse within psychiatry
Working Experience	7 years

<b>Average patients daily.</b>	4
<b>Phones Used daily</b>	iPhone
<b>Test person B</b>	
<b>Age</b>	49
<b>Gender</b>	Female
<b>Profession</b>	Specialist Nurse within psychiatry, forensic open care unit.
<b>Working Experience</b>	16 years
<b>Average patients daily.</b>	5
<b>Phones Used daily</b>	iPhone/Samsung

<b>Test person C</b>	
<b>Age</b>	29
<b>Gender</b>	Male
<b>Profession</b>	Nurse
<b>Working Experience</b>	1 year
<b>Average patients daily.</b>	10
<b>Phones Used daily</b>	Samsung S3

Test person D	
Age	58
Gender	Female
Profession	Charge nurse
Working Experience	35 years
Average patients daily.	2
Phones Used daily	iPhone

Test person E	
Age	41
Gender	Female
Profession	IVA-Nurse
Working Experience	17 years
Average patients daily.	2
Phones Used daily	Samsung S3

Test person F	
Age	25
Gender	Male

<b>Profession</b>	Homecare coordinator
<b>Working Experience</b>	1 year
<b>Average patients daily.</b>	10
<b>Phones Used daily</b>	Samsung S2

**Table 1: Showing the most important user characteristics. Additional Characteristics could be read in Appendix F**

### 5.6.3 TASKS

Below are the tasks that the test persons were asked to perform.

1. Add a comment on a patient
2. View alarm history of a patient
3. Check who is responsible for a patient
4. Check who is working on ward A and B
5. Do a search on Fass

### 5.6.4 OUTCOMES FROM THE USER TESTS

This section describes the findings and recommendations in the first usability test.

All in all six persons working in the health care as nurses tested our prototype, all with different technical expertise and working background. For example some works at the ICU which was exactly where this application would be handy where there are a lot of alarms. Other wards that the test persons had worked at were psychiatric ward, home care and emergency care, and some comments reflected their current working environment pretty well. For example the person that worked in the home care thought that it would be convenient to see where the staff currently was on the ward. He has as a job to visit sick people in their homes and help them out, so that can explain why he had that as a high priority. A problem with this feature would be that this would imply that maps would have to be generated over each ward (or even the whole hospital since staff could transport patients to surgery, “x-ray” etc.). This could be simplified by only showing what rooms are on the ward (as it currently is in Bedview) and when a nurse has accepted an alarm with his/her dect phone the system is showing an icon of that staff that has accepted the alarm over the bed. In addition a text could occur under “staff on shift menu” saying if a person has accepted an alarm and what bed they are at.



A back button, to go to the previous view, or a home button to go to the home screen was directly requested by two testers, and indirectly by another so this is something we really needed to consider. Originally we wanted the Listview as a default/home view that the application would go to if idle, however a real home screen with links to the 4 main views was probably a good idea, even if the idle state would still be Listview.

In general the most difficult tasks, during the testing, proved to be finding the top and right menu for the first time. Once the user found both the menus, the remaining tasks were completed with few difficulties. Some of the users did not find the top menu, because the “slide down button” for it was the same color as the frame of the screen (black), so it was mistaken as part of the physical screen, and not a software object. Additional hinting might be needed here to improve the guessability.

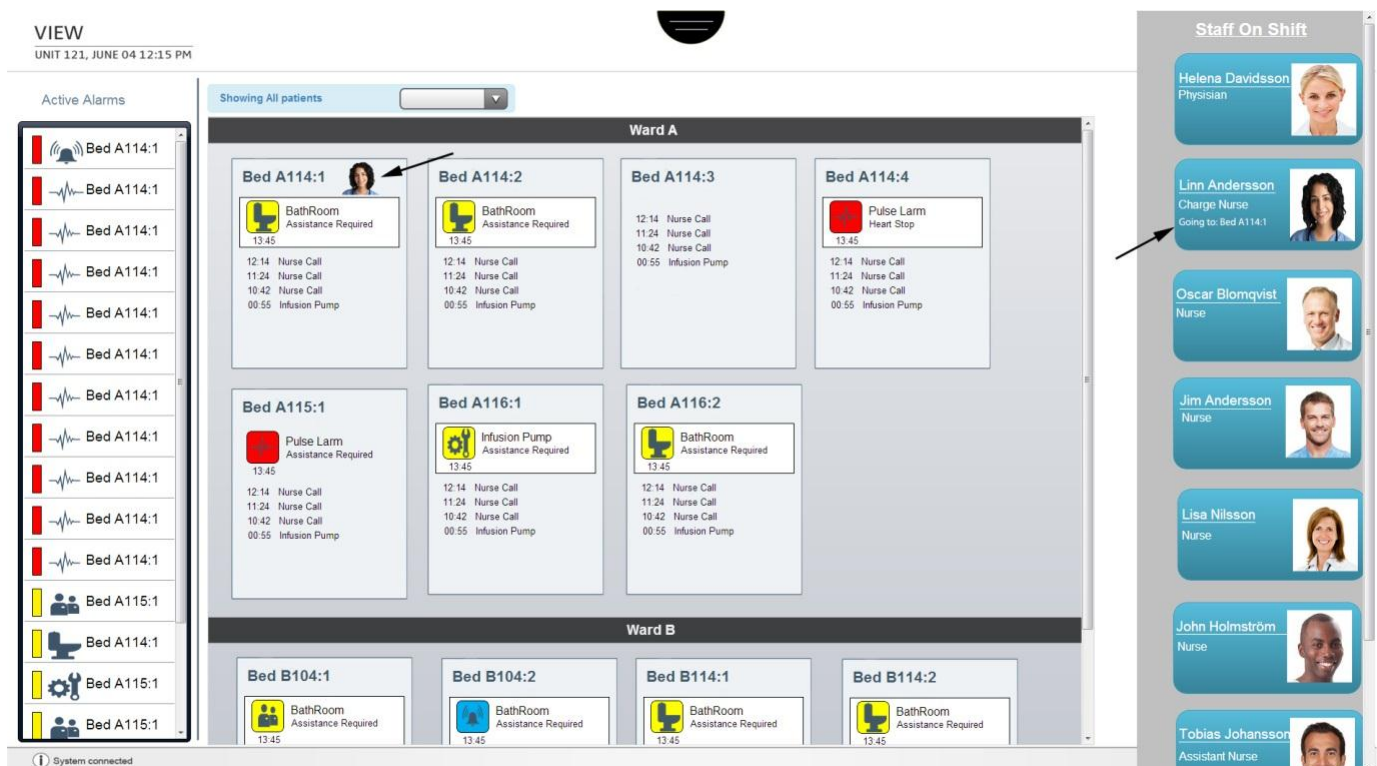


Figure 23: Showing an improved version where the system shows what nurse has taken what call.



**Figure 24:** Showing the menu button that “melted” in to the screen.

Interaction with scrollable object was very interesting. In an average wimp application or web page, the scrolling is done by dragging the scrollbar or clicking the up/down arrows on the scrollbar (in addition to pg\_down and pg\_up buttons on the keyboard or scroll wheels). In a touch interface naturally you scroll by sliding the actual component you want to scroll, however many of the users interacted with the scrollable objects as if it was a wimp application, by trying to slide the scrollbar or click the down/up buttons on the scrollbar. Most likely this occurred because the scrollbars we used were visually identical to those of a default webpage or wimp application, and not of an Android/iPhone standard. Generally the scrollbars in touch interfaces are very small and do not have any down/up buttons; hence they are merely acting as indicators of how far down/up you have scrolled, and not as directly interactive objects. In many applications the scrollbar only appear when the user scrolls the page and then disappears.

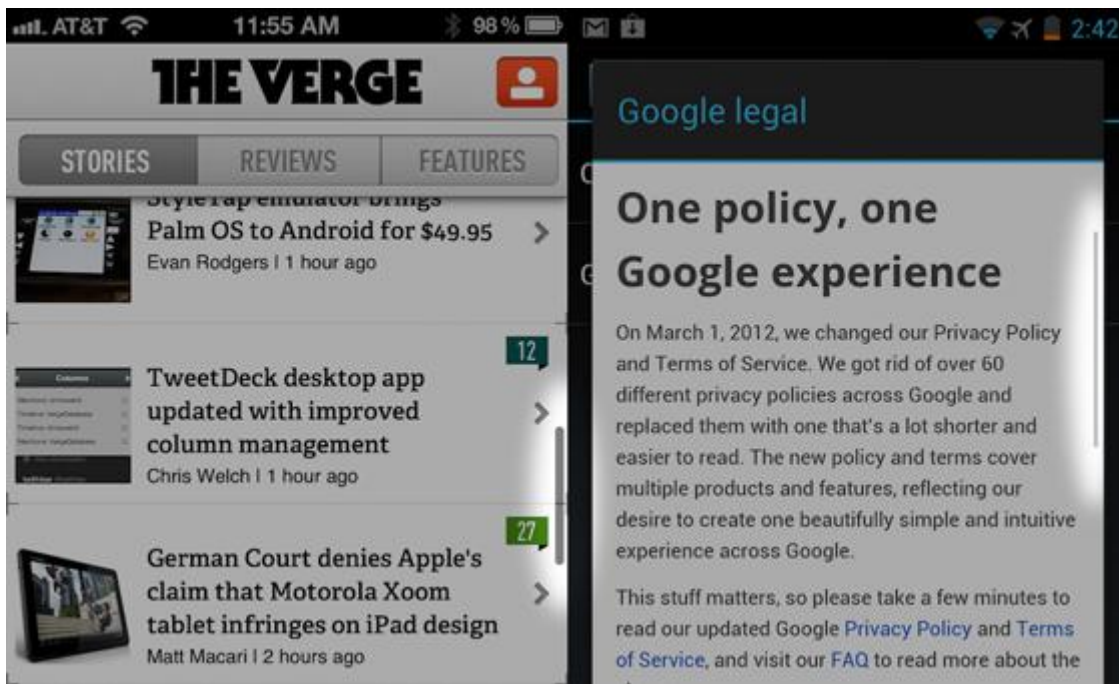


Figure 25: Showing how scroll bars are used in mobile applications, hinting the user that the user can scroll the entire content instead of trying to scroll dragging the scrollbar. The image is taken from the Verge website. <http://www.theverge.com/2012/7/17/3165237/intellectual-property-apple-patents-disappearing-vertical-scroll-bars>

Interacting with the native virtual keyboard in Windows 7 went as expected, it took some time for users to print the comment and some spelling mistakes were easily made. Not surprisingly some users said that a real keyboard would have been better. Other comments were that the keyboard would be better if opened on request. There were some problems with achieving this in our prototype; the web browser that best displayed our prototype didn't have support for enabling the native virtual keyboard on demand as other browsers did (for example Internet Explorer 9). So that's why we had to have the keyboard already brought up for the users and having it floating around the screen so it wouldn't cover any vital parts of our user interface. In the real prototype this would be taken care of so that the keyboard is displayed whenever the user presses a text input field (Like figure 23).

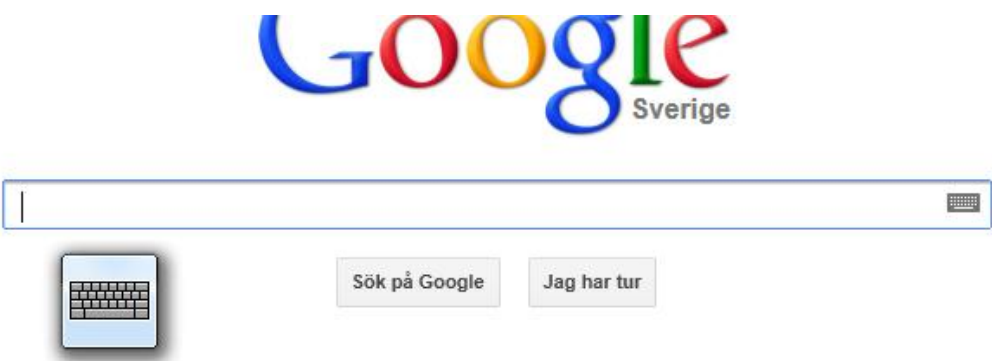


Figure 26: Showing how a web browser that supports touch acts when it senses that a user has pressed a text field with a touch screen (Not how Chromium Portable web browser behaved in our user test).

Additional results from the user testing were that it lacked Static visual hinting in its places. For example half of the test persons (3/6) didn't find either top or right menu button (one didn't find neither), this resulted in us improving the static visual hinting on both menu buttons but also on the Bedview buttons, making them more 3d-like by adding drop shadows to them. This is a generally large drawback with touch interfaces, since you don't have a mouse you can't have cursor hinting in the interface.

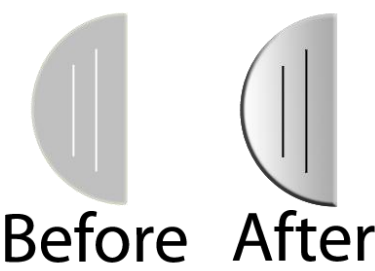


Figure 27: Showing the added the static visual hinting.

5.6.5 USER TEST STATISTICS

Below are the statistics that were gotten from the first user test. After the user had tested the hifi they were asked to fill out a questionnaire providing us with additional information. The testers are labeled A-F in the table below and the higher the number the better except for question 2 where lower is better.

Questions	
Q 1	I find this application useful in the health care:
Q 2	I find this application unnecessary complex:

<b>Q 3</b>	I find this application easy to use:
<b>Q 4</b>	I would need assistance in order to use this application:
<b>Q 5</b>	The menus were easy to find:
<b>Q 6</b>	The naming of the menus were good, I immediately knew where I would end up if clicking on a menu button:
<b>Q 7</b>	I imagine that people would learn this application rather quickly:
<b>Q 8</b>	The virtual keyboard was easy to use:
<b>Q 9</b>	Additional features you would like to see in the application:

**Table 2: Showing the questions that were given to the user testers.**

Q nr \rate	1	2	3	4	5
F 1				BFCA	DE
F 2	DF	B	CA	E	
F 3				BFCEA	D
F 4	DB		FC	A	E
F 5			BF	CEA	D
F 6	E		B	FA	DC
F 7				BCA	DFE
F 8			D	BFCA	E
F 9	C: "I would like to be able to 'click back' to previous page"				
	B:" I would like to know where people are located"				

**Table 3: Showing the answers from user testers.**

From the statistics it can be read that most people would find this application usable in the healthcare, everyone answer either a four or a five out of five possible. In question

four two people answered that they thought people would need assistance to use the application but pointed out that this would only be necessary in the beginning and that it probably would be easy to use after that.

## 5.7 EVALUATE

When the future evening had been completed, and results reviewed, we made some design changes based on the feedback we had gotten from the test, as described in previous chapter. When this was completed, we started to evaluate the application to decide how the final prototype should be implemented, and what major changes and decisions needed to be done before we started to implement the final prototype. A very large portion of the interaction with the prototype consisted of point and click, which made it feel like it was just an average desktop application being run on a touch screen. While it was partially due to the users not being familiar with touch interfaces, we also felt like we needed to include more possible touch gestures. We decided that this should be done redundantly though, so that there was the possibility to interact through the gestures, but not exclusively so. Navigation menus, for example, should be displayed both by swiping from the edge of the screen according to Windows 8 guidelines, but also by tapping the flap.

As for how to implement the final prototype, the first thing we considered was Windows 8. A Windows 8 application could be implemented both through HTML and Java, which were both languages that we were comfortable with. Python was an alternative we considered as well, as it had recently started to support touch inputs and gestures. However compared to html and java there was a lack of experience on our part, and due to touch support being a relatively new feature for it, the web support would be very limited if we ran into technical difficulties, which eventually made us exclude it as an alternative. Ascom also made it clear that they would not consider making an iOS application, so that platform was excluded as well. It all came down to three alternatives in the end; Windows 8 application (Java/HTML), Android application (“Java”), or platform independent HTML/JavaScript application. While Windows 8 seemed like a good idea for a real commercial version, it was very strict when it came to the interface design, and developing for Windows 8 would not let us configure the interface to differ from the default structure and visual appearance. So in order to stay neutral, and develop a concept design that could be used for both Android, Windows 8 and/or neither, we chose to develop the final prototype as a platform independent html application. This would give us more freedom to explore alternative designs, whether or not they fit the Windows 8 design requirements.

## 5.8 PROTOTYPE (HTML)

When we first started to develop the real prototype the normal software process was used according to the software process chapter in Software Engineering 8 (Sommerville, 2007).

1. *Software Specification*: Ascom had a document in which there were some clearly stated specifications that should be in the software. Also some specifications were taken from meetings with Ascom clients, like for example that the most important information should be seen from a distance of about eight meters. Other specifications were brainstormed by us as developers to see if they were desirable in the real product, like for example to include Fass into the system.
2. *Software Development*: When starting working on the final prototype we didn't have any workflow/dataflow or activity model since we already had done a technical prototype and knew exactly how our prototype should look. We used the *iterative development* approach together in some sense with *component-based software engineering* (Sommerville, 2007) when developing the real prototype. The reason for choosing the iterative development was because we didn't know exactly how to develop our prototype, we knew that we wanted to make a website but because there are so many solutions to a problem (for example making a sliding effect in JQuery) we chose one effect, evaluated it and changed it if it didn't satisfy our needs. We also made use of the *component-based software engineering*, instead of writing every touch interaction from scratch in JavaScript which would occupy too much of our time we used finished freeware libraries and included them in our design.
3. *Software Validation*: This phase was in a way combined with number four below. We wrote the user tests in a way so that they would test almost every aspect of the program and then we could find out potential errors and correct them. Although much more tests than just test the interaction could be done, like for example test the mean time before failure (MTBF) we skipped this in our prototype because this would take too much time to test and the gain would be too little since this prototype was only going to be used for demo purposes. However according to Stephen Woods (Woods, 2013) the most problematic thing that could spoil a touch interface is it not being responsive, it's crucial that it's responding rapidly for users to perceive the interface as good, in other words it's crucial that the prototype's code is executed fast. That's why we spent some time trying to go through our technical prototype's JavaScript code and tried to speed up it where possible. We used his advice to test our code on JSPerf.com



4. *Software evolution*: As mentioned above this phase was combined with software validation. When users tested it the second time at Uddevalla hospital we got a lot of feedback on our design that we had to act upon and make some changes in our design to be more user friendly.

When we first planned on how to implement our prototype interface we got a lot of information about different implementation techniques from the paper Multi Touch Technologies (NUI group Authors, 2009). The paper suggested using the Script Language Python and it seemed at the time as a good idea giving us the ability to access the extensive graphics library Open GL7 which we have used already in a previous course called Computer Graphics. However the decision fell on going with HTML for the simple reason for simplicity, using it with an ordinary web browser on any platform.

As already mentioned we chose to implement our final prototype in HTML/PHP/CSS/JavaScript/JQuery/MySQL using Dreamweaver<sup>24</sup> since we had experience in that environment in previous projects. This development environment enabled us to quickly develop a prototype with HTML with some extensions (jQuery) that could be run from a web browser without any further dependencies on any other software. Another major contribution for choosing this way of implementing the prototype was because early on in the project during one of the seminars at Ascom we found out that they were not keen on “locking themselves” into too much technology that they don’t have that much control over. Our implementation is built running Wamp Server<sup>25</sup>, which in turn uses Apache web server, having a MySQL database providing the prototype with data, everything being freeware and no royalty would have to be paid. This solution enabled us to quickly develop a realistic prototype although some compromises had to be done in order to save development time. One major compromise that we did was that we only implemented our prototype to fully work in Google Chrome web browser.

- Android/iOS guidelines were followed in the case of being able to use the swipe gesture to quickly navigate between detail views (in this case the detail Bedviews). Contextual information was implemented in the detailed Bedview that informed the user about the relative list position of the currently visible item. In addition transition between the views is done as the user performs the swipe gesture and is not waiting for the gesture to complete and then transition between the detailed Bedviews.

---

<sup>24</sup><http://www.adobe.com/products/dreamweaver.html>

<sup>25</sup><http://www.wampserver.com/en/>



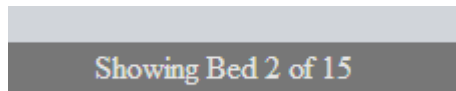


Figure 28: Showing contextual hinting.

- Our home button screen is inspired by the actual Smartphones home button (not visually but practically). One click on it and you see all the things that are available in the interface in order to simplify for the user, it was also requested by users in the first user test at the Ascom user testing evening.



Figure 29: Showing the home button implemented in the second hifi prototype.

We followed the Pictures are faster than words guideline to some extent, i.e. **“Consider using pictures to explain ideas in the application, they get people’s attention and can be much more efficient than words”** (Android design guidelines, 2013). In our application pictures of employees are used frequently throughout the application, for example in the dropdown menu in Bedview because people have much easier to remember faces than they have to remember names. We used a spinner dropdown in Bedview since users are switching between views within the same dataset (showing different beds depending on who is responsible for that bed), just as Android guidelines recommend.

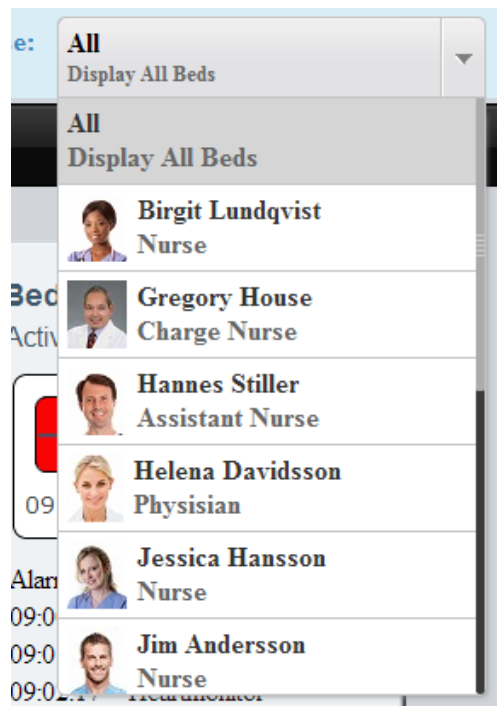


Figure 30: Showing a picture of the dropdown list used in Bedview.

- Confirming and acknowledging actions that a user performs is an important part of a user interface, they are supposed to ensure the user what just happened and not leave the user questioning if the application really performed the task that (s)/he asked it to do. However they should be used when really needed and the right type of notification should appear. That's why we only interrupt the user when really needed i.e. creating an acknowledging dialog for when for example the user tries to remove a comment and just use a small notification dialog that the comment really was removed in the Bedview after the user has removed it.

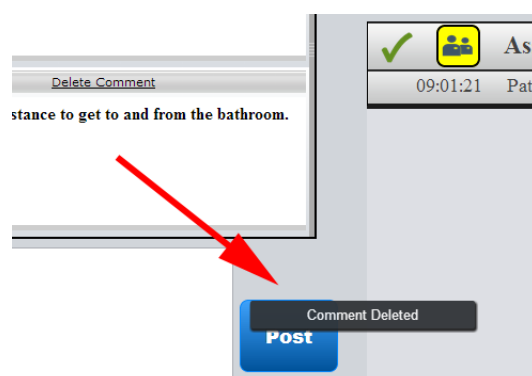


Figure 31: Showing an example of system feedback in the prototype.

In Listview we decided to make some changes as well. We had realized that it was not apparent which alarm elements represented alarm history and which represented

ongoing alarms, so in an attempt to make this clearer, we put a little shade on alarm history, making them slightly darker. Additionally we added a green border on the elements in alarm history to suggest that they had been taken. In the same manner we added a red border to the ongoing alarms, as red to our knowledge is universally associated with danger or attention. While on the subject of colors, we also changed the background color of detailed list view and its corresponding bed element, to a light blue color. The reason for this was to get more contrast to the other bed elements and the background of list view, to more clearly display which bed the detailed alarm list belongs to, without having a negative effect on visibility, focus or flow. As a response to the confusion regarding escalation chain, we added a picture of the corresponding nurse next to the icon and a number representing escalation chain, which hopefully would clarify to some extent. Even though this is not a feature that must be immediately apparent, this was deemed a necessary addition, as it would also inform the user who the nurse, corresponding to the number in the escalation chain, is.

#### 5.9 TESTING INTERACTIVE PROTOTYPE 2 (AT UDDEVALLA HOSPITAL)

As the final prototype was fully developed a final user test was needed to further find out potential flaws in our design. This time we wanted to conduct a user test on an actual hospital, this because we wanted our user test to be as efficient as possible, we realized that gathering a lot of nurses at the same time outside such an environment would be difficult. On a hospital it would be easy to find user testers and they would be on the same location, rather than to try to have the testers come to us which would be a challenge. Also we wanted them to test our prototype in an environment that they were comfortable with. Contact was taken with several hospitals in "Västra Götalands regionen/Hallands Län" but many wards said they didn't have time due to vacations, but that we were welcome back in autumn (which was not relevant since the project was to be complete at autumn). Eventually though, we found a hospital with the time and personnel available for user testing. An appointment was made with a head of department at an acute orthopedic ward at the Uddevalla hospital. A small booth was put up and then the testers would come in one at a time whenever they had some spare time, to test the interface. This time though only notes were taken and the interaction with the screen was not video recorded; neither were the test users asked to fill out a form what they thought about the prototype.



**Figure 32: Showing the "monter" set up at orthopedic ward.**

In total nine persons user tested our prototype divided among six nurses, two assistant nurses and one charge nurse (As seen in table 4). When we designed the user test some key areas were of interest listed in a bullet list below (The complete test can be seen in Appendix C). Since we figured that the user testers would be stressed and only spend some short time each with the interface we designed one long test and then we would adapt this long test to shorten it down so it would fit all testers, regardless of how much time they would have.

- How easy it is to find the different views (Bedview, Listview etc.)
- Are the main menus easy to find (Top/nurse menu)
- Find the views with more detailed information (Bedview, Listview). Find out who is responsible for each bed, find and post comments, and more detailed information about an alarm.
- Is it clear when a new alarm arrives?
- Is the information from the alarms clear? Is it obvious which alarms that are active and which that have been taken?
- Is it obvious how to show a certain nurse's patients?

Test person 1	
Working Title	Assistant Nurse
Age	20
Test person 2	
Working Title	Assistant Nurse
Age	22
Test person 3	
Working Title	Nurse
Age	25
Test person 4	
Working Title	Nurse
Age	27
Test person 5	
Working Title	Nurse
Age	28
Test person 6	
Working Title	Nurse
Age	34
Test person 7	
Working Title	Nurse
Age	39
Test person 8	
Working Title	Nurse
Age	43
Test person 9	
Working Title	Charge Nurse
Age	43

Table 4: Showing the most important user characteristics in the second user test.

### 5.10 OUTCOMES FROM THE SECOND USER TEST

This section describes the findings and recommendations from the second usability test.

Few found the “swipe between beds” functionality. But since this was a shortcut this wasn’t considered that big of a deal, as long as test persons could access the information from another place (Close the current detailed Bedview and open another one). What was more worrying was that some test subjects (the older ones) had some minor trouble finding the main menus at first as soon as they were faced with our interface for the first time. Some persons also thought a small tutorial would come in handy (like are present in apps in cell phones). Hinting interactions had been studied earlier in the project by (among other things) studying a report written by Sus Lundgren and Martin Hjulström about gesture hinting (Lundgren, Hjulström, 2011). In short they bring up the problem today that a lot of interactions in mobile devices are hidden and that they need to be discovered by the user. They had an interesting way of solving it by using a fix set of symbols to highlight to the user what gestures were available. Although perhaps not feasible in this project, we understood from the user tests that some kind of gesture hinting would have to be implemented in the prototype, mainly to find the menus in the prototype.

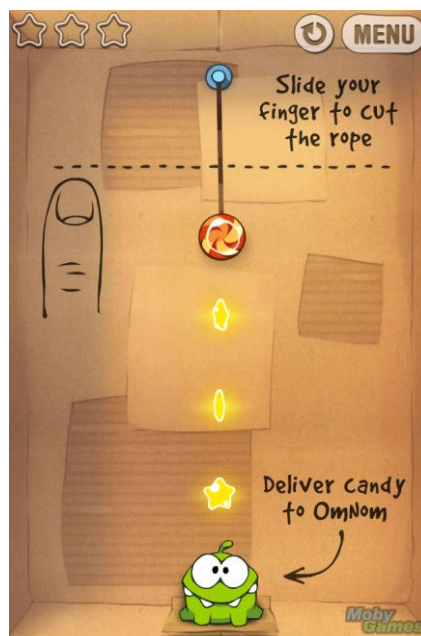
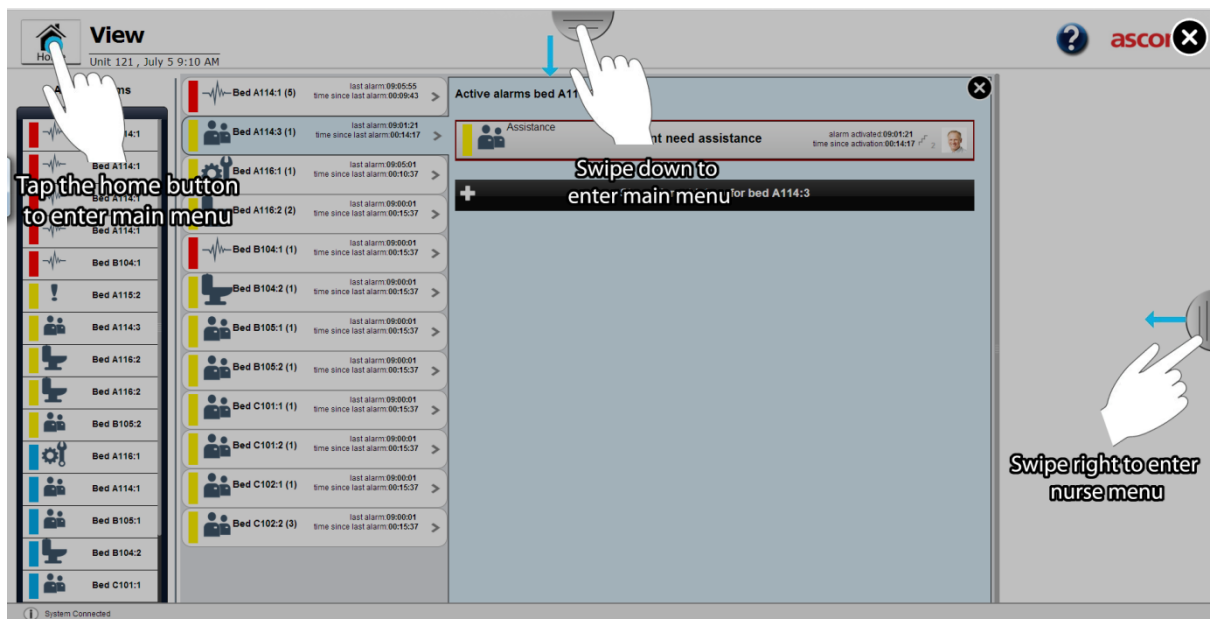


Figure 33: Showing gesture hinting in the mobile game called cut the rope. The image is taken from Moby Games website. <http://www.mobygames.com/game/iphone/cut-the-rope/screenshots/gameShotId,499867/>

Our response to this was to simply implement a button labeled with a question mark (?) symbol on it that when clicked opened up an overlay explaining where the main menus were (Figure 34). The reason for only showing the main menus was because they were the only things that some people had trouble finding at the beginning, as soon as they had found the menu for the first time they later could accomplish all tasks that they were given without any trouble. The reason for choosing to visualize this “tutorial” by pressing a button in the interface and not having it appear “by itself” like it does in cell phones is because you here have different users using the interface and not everybody is using the interface for the first time and it would be annoying if it would appear for everyone, even those that already master the interaction with the prototype.













**Figure 34:** Showing the help screen that appears after the user presses the question mark button in the main interface.

In Listview, despite the efforts to make active and taken alarms stand out from each other, i.e. mark active alarms with a red border and accepted one's with a green one, test persons still misinterpreted which were active and which were not (Figure 35). Many test persons even thought that all alarms, both active and inactive (alarm history) were active. This was something that had to be dealt with, and our solution to that problem was to let the user click a row to expand and show alarm history (Figure 36& 37).



## Active alarms bed A114:1

	Heartmonitor	<b>bloodpressure critically high</b>	alarm activated:09:05:55 time since activation:07:49:24	1	
	Heartmonitor	<b>Pulse critically high</b>	alarm activated:09:05:07 time since activation:07:50:12	1	
	Heartmonitor	<b>bloodpressure critically low</b>	alarm activated:09:02:17 time since activation:07:53:02	3	
	Heartmonitor	<b>Pulse critically low</b>	alarm activated:09:01:35 time since activation:07:53:44	3	
	Assistance	<b>Patient need assistance</b>	alarm activated:09:00:01 time since activation:07:55:18	2	

## Alarm history for bed A114:1













	Heartmonitor	<b>bloodpressure critically high</b>	alarm activated:09:05:55	1	
	Heartmonitor	<b>Pulse critically high</b>	alarm activated:09:05:07	1	
	Heartmonitor	<b>bloodpressure critically low</b>	alarm activated:09:02:17	3	
	Heartmonitor	<b>Pulse critically low</b>	alarm activated:09:01:35	3	
	Assistance	<b>Patient need assistance</b>	alarm activated:09:00:01	2	

Figure 35: Showing how we first tried to solve the issue of visually separating active from inactive alarms which turned out to not be that successful.

	Assistance	<b>Patient need assistance</b>	alarm activated:09:01:21 time since activation:00:12:43	2	
---	------------	--------------------------------	--	---	---



Show alarm history for bed A114:3

Figure 36: Our solution to try to separate active from inactive alarms. Here the user must push a label in order to show the alarm history.



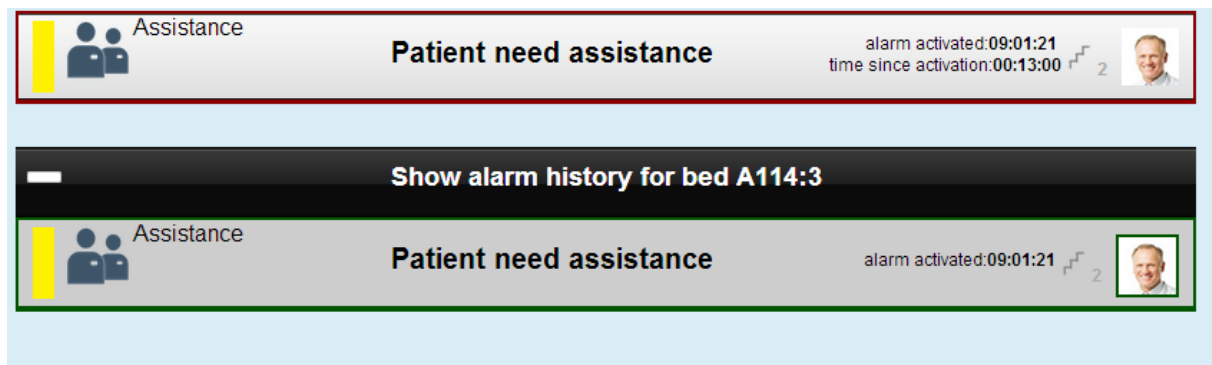


Figure 37: The user has pushed the "Expand" button and the system shows all accepted alarms.

Our hypotheses that Fass is widely used when the healthcare turned out to be true in our user tests. Good feedback was given that Fass was implemented into the system as it currently was widely used within the ward that the tests were performed on. However less good feedback was given to the visualization of a new alarm. We implemented a new alarm to blink one time (about two seconds) which proved to be too short, that's why we prolonged this time so that every new alarm would blink for about 10 seconds instead.

As a final test we checked how far away you could see the text in Listview when not being expanded, since one of the requirements from the potential clients was that it should be visible from a distance of about eight meters. Both of us could easily spot the text and see the symbols but as it should suit both younger and older persons and preferably should be read by persons without glasses we asked some fellow colleagues, one in their mid-40's and the other in their early 50's and they both could spot the color and the first one had no trouble reading the text and spotting the symbols. The other had some minor trouble but at a distance of about six meters it could be read. Another one at age 54 and was farsighted and could read the text at about four meters (which is the minimum distance that an alarm should be seen according to ISO 60601-1-8) without any glasses.

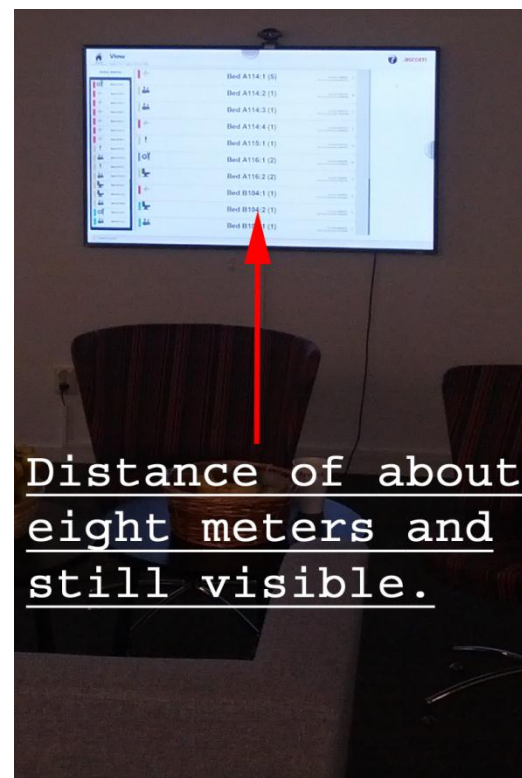


Figure 38: Showing the test made of how far one could see the interface.

## 6 FINAL RESULT



Figure 39: Showing the small alarm list.

The final design of the concept prototype, mainly consists of 4 views; Listview, Bedview, Fass and Personnel. The former two is where alarms and information about the alarms are viewed and interacted with, Listview being the default view and the most similar to the desktop version. As the names implies, the layout of Listview is a list of all the ongoing alarms while the layout of Bedview is bed by bed grouped in wards. The Fass view is a direct implementation of Fass online into our application, to allow access to a big medicinal database online. And finally the Personnel view displays the staff of the hospital in the wing/part that the application has been installed for.

As Listview is the default view, the application will automatically return to this view if it is left idle (not being interacted with for a time of two minutes but this is an arbitrary estimated number without any research behind it). However as the main purpose of the application is still to display ongoing alarms which are high priority and time sensitive, the user needs to be informed about ongoing alarms and alerted to new incoming alarms, even when interacting with the application. Therefore the design decision was made to add a smaller alarm list, which would be constantly visible regardless of the state of the application. This alarm list was given little less than  $\frac{1}{8}$  of the total screen space in order to be able to display the most vital information about each ongoing alarm.

In order to get the users attention if a new alarm was activated, the new alarm would flash red for 10 seconds (same thing should occur if multiple alarms are going off simultaneously for each alarm). By simply tapping any of the alarms in the list, the user would be taken to detailed information about that alarm, and all

other ongoing alarms and alarms within the last 8 hours for that patient, in Listview. The expanded Listview is directly inspired from a split view, widely used in tablets (Figure 40) and as previously mentioned it changes back to an ordinary list when idle (Figure 42).

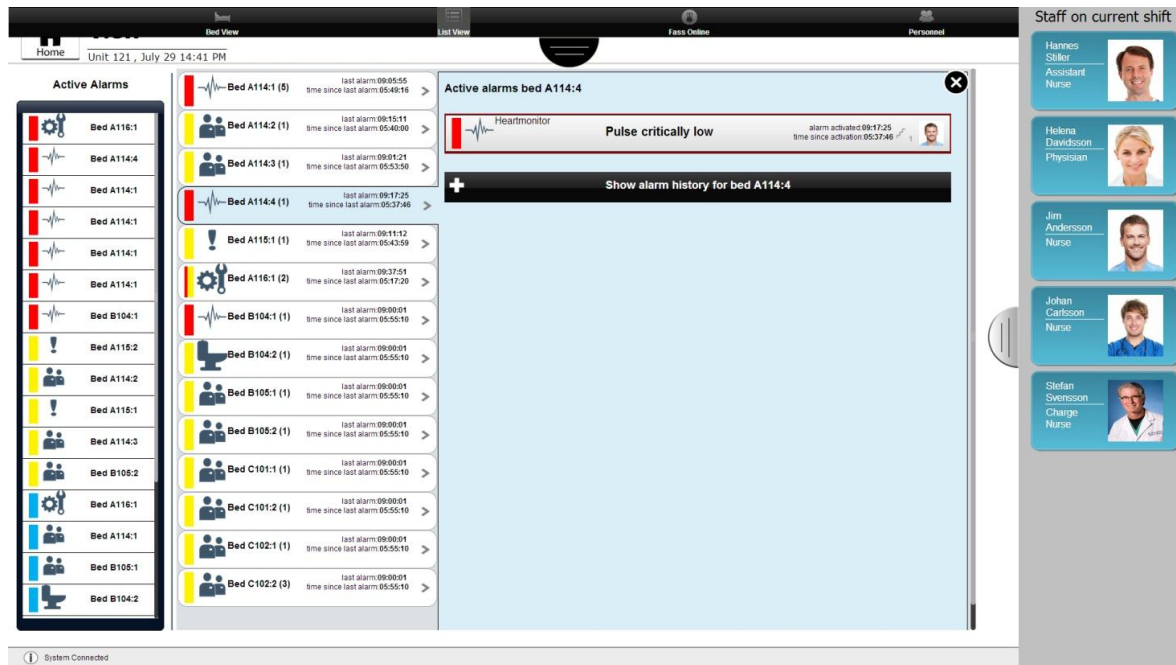


Figure 40: Showing all main menus in the interface.

The main navigational tools for our application can be found in two menus or bars as we like to refer to them. This top bar and right bar can be displayed by sliding them out from the top and right edge of the screen respectively. The decision to hide them by default was made in order to distribute as much space as possible to display the alarms, and to simplify the interface in idle state. If this were an application that would be used by only one person (like an app on a mobile phone) then these menus would be out the first time the application was launched, to make themselves known to the user just as Android guidelines recommend, and they would still be there every time the application was launched until the user hides them. But in this case the application will be used by users with different knowledge about the application and to always have them out in idle state was something we considered would annoy the greater part of the users of the application.

Focus in the application should be on the information about the alarms, and displaying too much information at once might overwhelm the user and lead to delayed reaction or alarms going unnoticed in a worst case scenario. Through the top bar the user can

navigate between the four main views, each view is represented by an **icon** and the currently active view is marked to indicate the state of the application. With the right bar, also referred to as the nurse menu, the user can see all the nurses that are currently on shift. Each nurse is represented with a blue “button” containing a picture of them, along with name and title (head nurse, assistant nurse etc.). By tapping one of these buttons, the user will navigate to Bedview displaying the beds that the corresponding nurse is responsible for. The concept of these navigation bars is based on Windows 8 design guidelines/restrictions. In all Windows 8 applications there is a top/bottom menu. These two menus are linked together and are both displayed if the user either slides from the top edge and down or bottom edge and up. A similar menu can be found on the right side, toggled by sliding from the right edge, and this includes system operations such as go to windows start menu or search. As our vertical space is very limited, compared to our horizontal space, and the alarms are considered critical information, we did not want to use the bottom menu. At the top of the page there was a small amount of space assigned to non-critical information, so having a slidable menu at the top did not affect the design negatively in any way. However with vertical space being the bottleneck for displaying critical information, we felt that decreasing it even further was not an option. Therefore the decision was made to replace the system operations menu on the right edge, with our second navigation menu, granting us much more needed vertical space, and occupying a small part of the horizontal space. As opposed to vertical space, we could spare some horizontal space without affecting the amount of information displayed. So in order to avoid the right menu overlapping critical information in the alarm view, we decreased the width of the alarms, leaving an empty space where the right menu could slide out without covering any critical information.

The default view, as mentioned earlier, is Listview. Coherency with the desktop version is something we have strived for in our design, so this view is very similar to the desktop version of View that we had as a starting point. Alarms are represented by a list element, containing a colored stripe representing the priority levels, a symbol representing the type of alarm along with a text describing the type, a text description of the alarm followed by time for the alarm and how long since the alarm started, and finally an icon resembling stairs with a number next to it representing the current position in the escalation chain. The last part was often confusing during the user tests, so additionally a picture element was added, displaying the nurse corresponding to the escalation level. The big difference between Listview and the original desktop View (Q1 version) is the possibility of interaction. Instead of displaying every single alarm with one list element, they are grouped by which bed they are originating from. So each list element is representing all the ongoing alarms for 1 specific bed and by tapping on a list element, a detailed view will become visible, displaying all the alarms for this bed, both ongoing and alarm history for the last ten hours. The list elements in the detailed view are

represented as described above just as in the original application, with the exception of alarm history where escalation level is no longer necessary. In order to avoid confusion the ongoing alarms were marked with a red border to get attention, while the alarm history was marked with a darker green color. Alarm history is also hidden by default in a list which can be expanded and collapsed by tapping a header element. This was designed post user testing, as we found that the alarm elements were so similar that they were easily confused with one another and it was not always clear that not all of the alarms were ongoing.

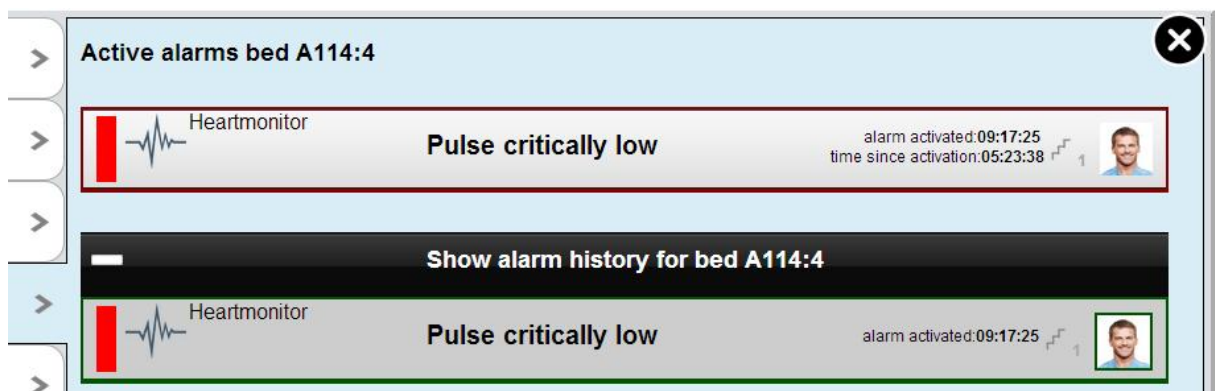


Figure 41

The list elements representing beds, or all alarms for a specific bed, looks different compared to the detailed alarm elements. As they represent several alarms, the colored stripe representing priority has been divided into three parts displaying the color of the three highest priority alarms from this bed. For example if only one alarm is active on that bed and it has priority three (serious) then the whole bar becomes red. But say that there exist two priority three alarms and two priority two alarms, then  $\frac{2}{3}$  of the bar becomes red and  $\frac{1}{3}$  becomes yellow since we write out the three first alarms and all alarms are sorted after priority with priority three first in the list. The text describing the alarm is replaced by a description of the bed (location and number) along with a number within parentheses representing the amount of active alarms on this bed. The escalation chain elements have been removed as well, replaced by a small arrow to indicate that additional information can be displayed (According to iOS guidelines).

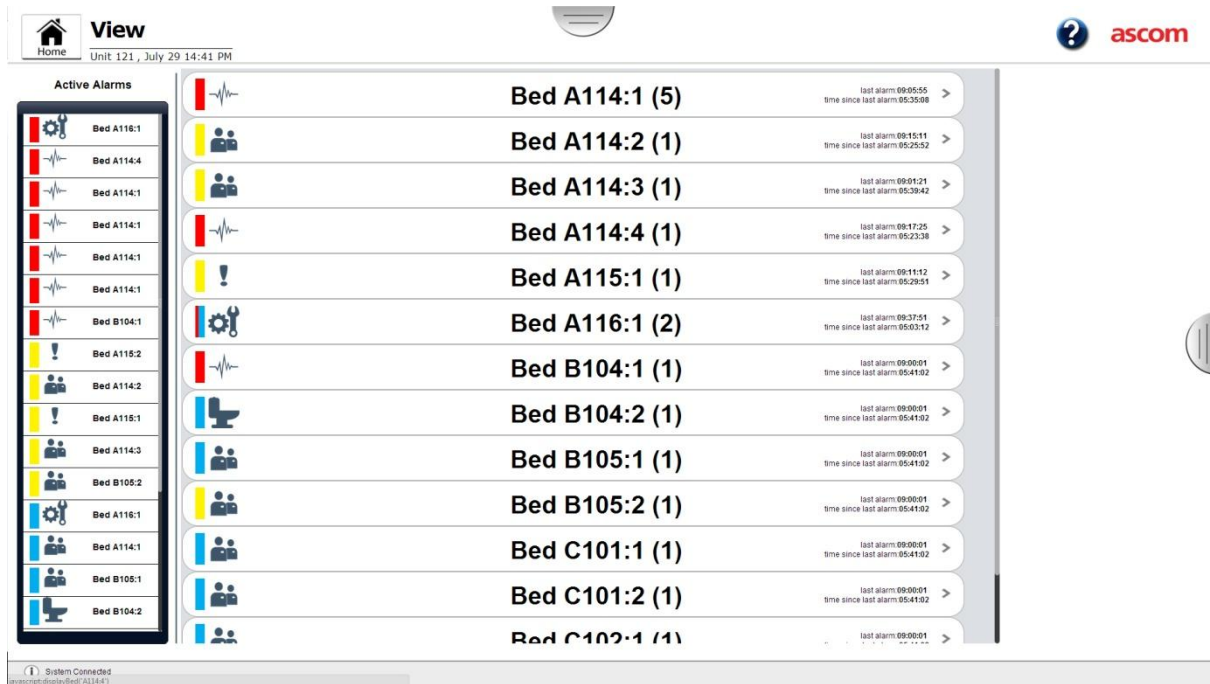


Figure 42: Showing the default idle Listview.

When the user taps on one of the elements representing a bed, all the list elements will shrink to  $\frac{1}{3}$  of the size, and at the same time the detailed list slides in from the right. The element that was tapped will change color and visually merge with the detailed list, so that the user can easily understand which bed the detailed list belongs to. The list representing all beds will remain to the left of the detailed view in order for the user to easily be able to swap between them. This design is inspired by TV interfaces, and various tablet applications (called split views) where a large amount of information is displayed, such as mail applications. The feature where the bed, which the detailed information belongs to, is of a different color in particular is an important feature for TV/DVD interfaces where the user has no mouse input, but have to navigate through up/down left/right arrows on the remote. This design is then critical for the user to be able to understand the current location in the list, and with touch input being “clumsy” or inaccurate in many cases, this was deemed as an important design feature.



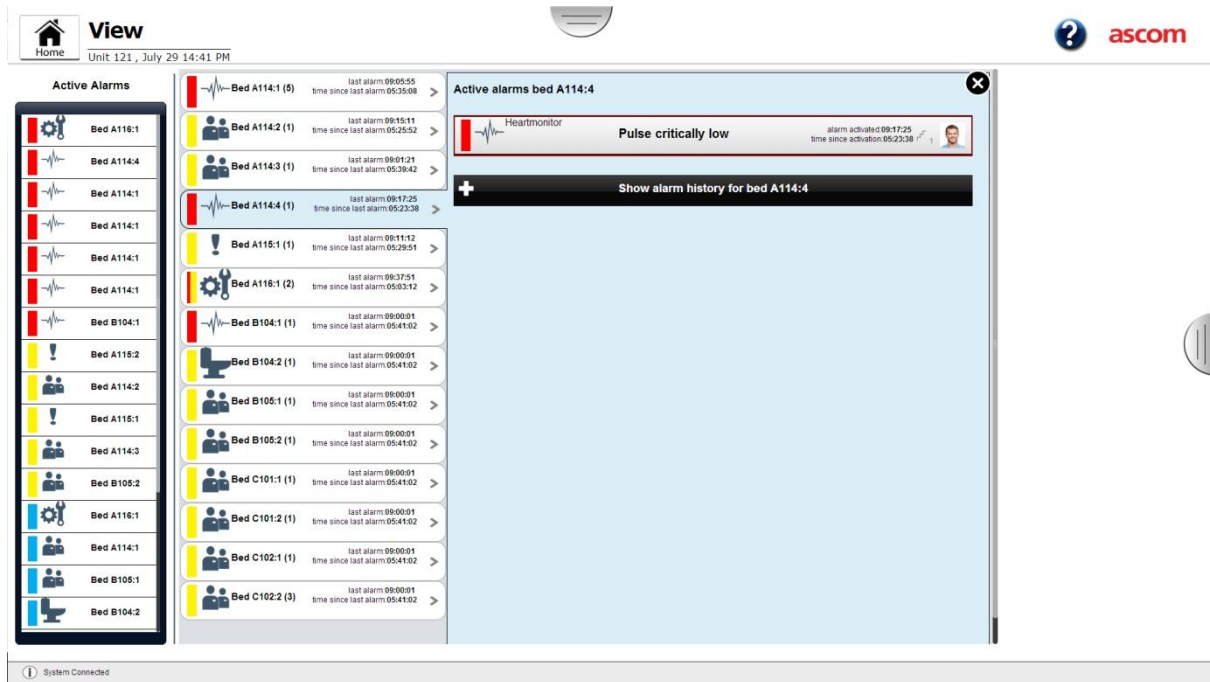


Figure 43: Showing detailed Listview.

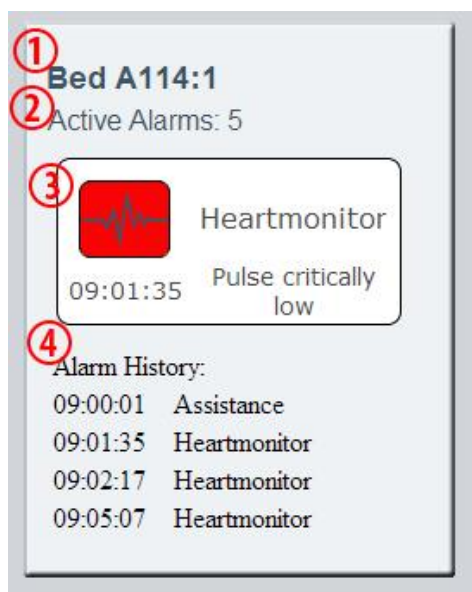
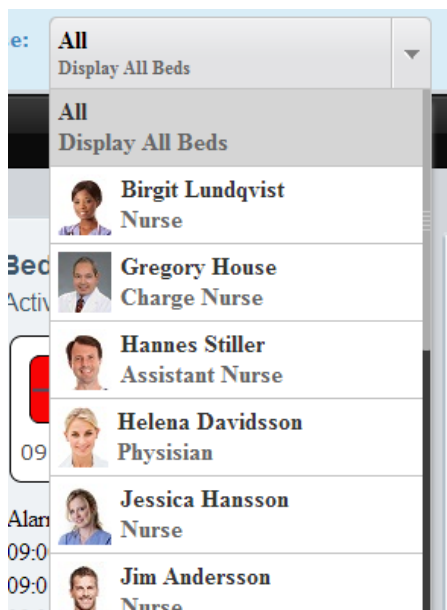


Figure 44: Showing how a bed is represented in Bedview.

The alternative view for displaying alarms is called Bedview. The concept of this view was inspired by the application called “Unite Assign” where the user would assign nurses to be responsible for specific beds. The purpose of Bedview was to give the user an alternative perspective of the alarms, and to allow for more interaction and additional information. While Listview is designed to be visible from a distance, Bedview is mainly considered a 1-foot interface, or short distance interactive view. Therefore the alarm elements may be smaller, and we were able to include other elements such as comments. The general layout of Bedview is of a grid format, with four elements per row and grouped by wards. Each ward is contained in a collapse/expand panel, just as alarm history in detailed Listview, and when expanded beds are ordered by number. Each bed is represented by a 250 by (300-320 depending on alarm history) pixel

rectangle containing bed number (1), the amount of active alarms (2), an alarm element for the highest priority ongoing alarm on this bed (3), and a short list of alarm history (4) with a cap of the four latest taken alarms (although there may be more accepted alarms). This is designed to give the user a quick overview of each bed, and by tapping on any of the bed elements, the detailed view for the chosen bed will be displayed. Additionally there is a drop down list at the top of Bedview (Figure 45), with which the

user can choose to filter by nurse. The list contains elements representing all the nurses responsible for one or several of the beds, and contains the name of the nurse, work description and a small picture of the nurse. If one of these nurses is chosen, only the beds that he or she is responsible for will be displayed. In our prototype one bed have three responsible nurses, the first responsible nurse is the nurse that an alarm for that bed is sent to first in the escalation chain. However in a real situation one bed might have a different number of responsible nurses and if more than three maybe a spinner should be used to display who is responsible for a specific bed. Pictures of the nurses are used both here and in other parts of the application, as users have much easier to remember faces than names, inspired by the android guideline **“Pictures are faster than words”**. In our prototype in order to minimize excise we set a min-height of the drop down list so that it will show a minimum of 13 employees without having to scroll in the list. Styling of an ordinary dropdown turned out to be quite tricky in CSS. We didn’t want the default height of a dropdown element since it had quite low height and turned out to be hard to select on a touch screen, and in addition pictures couldn’t be inserted here (It worked in Firefox but we didn’t develop for this browser). The solution turned out to make use of a free custom dropdown menu called msDropDown<sup>26</sup>. With this we now could customize how our dropdown should look like and in turn made the rows bigger in order to easily select them.



**Figure 45: Showing the personnel dropdown. Here the rows are much higher than the ordinary dropdown element in HTML in order to make it more user friendly in a touch environment.**

The detailed Bedview (Figure 46), which is displayed when a bed has been tapped, contains all the ongoing alarms and the alarm history for the bed just as in Listview. However they have been granted a lot less space in this view, to make room for the comment section (4) and the information panel (3). Originally the idea was that the user would be required to login, in order to display the detailed view, and that there would be confidential information displayed in the information panel if the user was authorized (3). Confidential information in this case could be if the patient had any allergies, what the main cause were for the patient being on the ward etc. However we have yet to find a solution that solves the problem of an unauthorized user accessing the information if the application was left logged in due to an emergency. Due to this the information panel will remain empty

<sup>26</sup><http://www.marghoobsuleman.com/jquery-image-dropdown>



except for the pictures representing nurses responsible for the bed, but we chose to still include it in the design, to visualize how it would look when this issue would be solved.

For the comment section the user will have to authenticate though, using a personal code to post and to remove comments. All comments will still be visible, but in order to remove a comment the user must input the personal code of the user that posted it. Through this method the application will not be logged in to an account at any point, but instead authenticate each time an action is made that requires authentication. A similar method was considered for the information panel by letting the user authenticate to get a popup window with the confidential information, but it was finally decided that no risks should be taken with sensitive information like this, and it should not be included until a viable authentication method is found.

While in the detailed Bedview (Figure 46), the user can easily navigate between beds by swiping left or right. This feature was designed according to Android guidelines that the users should be able to swipe between information on detail level. Also contextual information was implemented in the detailed Bedview that informed the user about the relative list position of the currently visible item (5). In addition transition between the views is done as the user performs the swipe gesture and is not waiting for the gesture to complete and then transition between the detailed Bed views. As additional hinting for this feature, two semi-transparent arrows were added at the right and left edge (6). It is the thought that these arrows should be clickable, unfortunately they are not in our prototype due to a bug. But some younger user testers (mid 20's) tried to tap them before trying to swipe the whole screen which was a good sign that the indication worked.

The 3rd view, called Fass (Figure 47) is not designed in this project. It is a remote website embedded into our application which displays Fass online mobile. Unfortunately though the site is labeled "mobile" the site is far from adapted to a touch screen (every element proved hard to hit due to their small size) which was found out in our user tests. Still we included it since this is an extremely helpful tool for nurses to look up information about any medicine, which is used on a daily basis according to some users we asked.

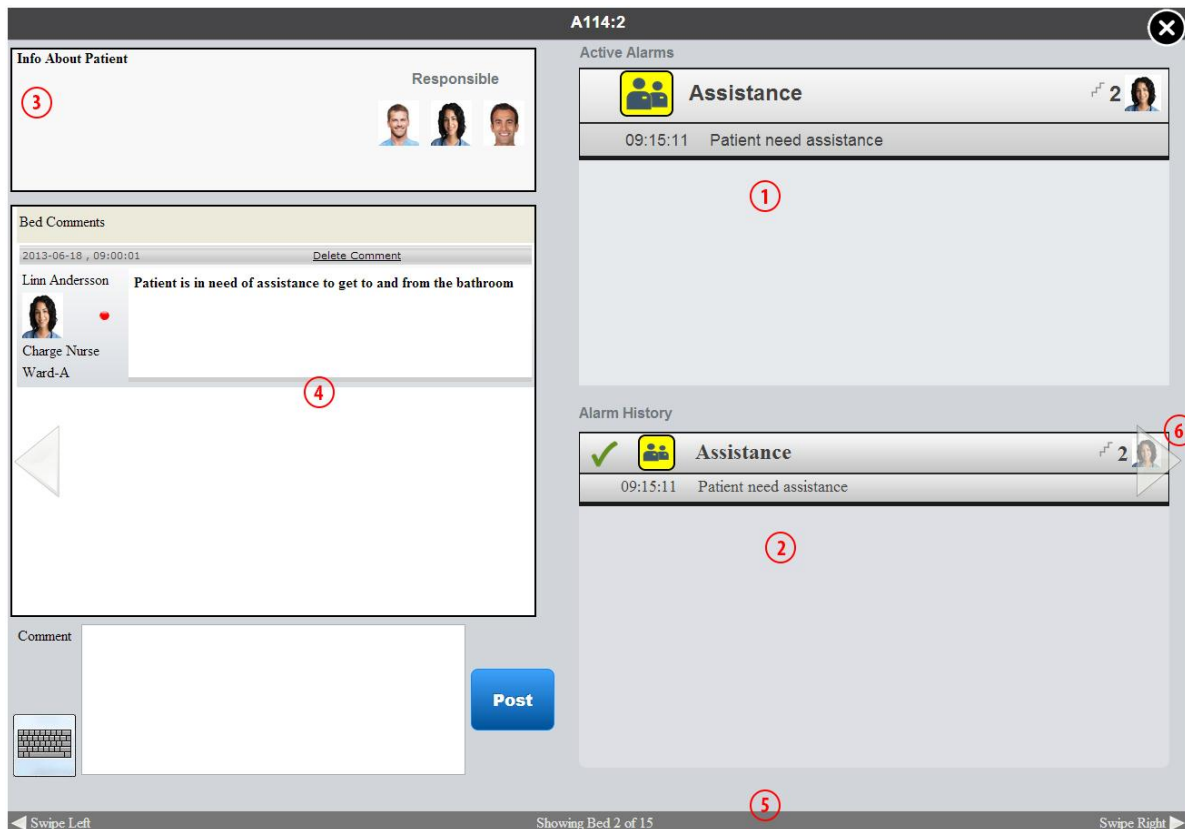
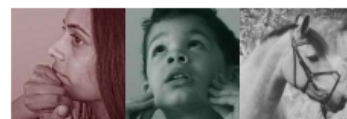


Figure 46: Showing detailed Bedview.

nobil.FASS.se 

**Välj ingång**  
[FASS för allmänheten](#)  
[FASS för förskrivare](#)  
[FASS om djurläkemedel](#)



[FASS.se](#)  
 (webbversion)

Figure 47: Showing an image of the embedded version of mobile Fass. As seen in the image standard links are used and proved to be very hard to hit.

The last main view in our application is called Personnel View (Figure 48). In this view the user can overview all the personnel working on the ward(s) that the application is implemented for, in two ways. The first way to view them is grouped by ward in expandable panels, just as in Bedview, the second is as a list ordered by name. The user

can switch in between these two views through tabs at the top of the page. In addition to be able to navigate by pushing the tabs the user can also swipe the screen in order to change view and it also has a “snap back” feature to tell the user that there is no more content as he tries to swipe when there are no more content, as suggested by Android guidelines. In both cases each nurse is represented by a picture of them along with name, ward, title, email, phone number, and date and time for when their next shift will start.

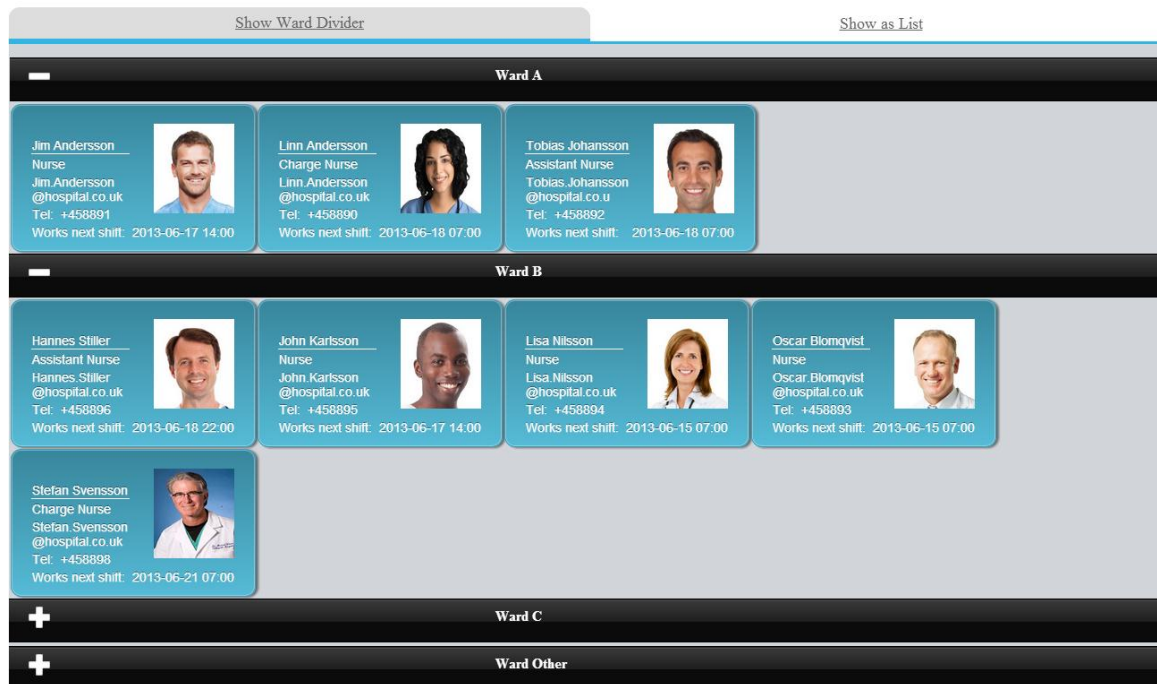


Figure 48: Showing final version of Personnel View.

## 7 DISCUSSION

In this chapter there will be discussion about methodology and results, what was good/bad. In addition we will discuss future improvements on the prototype and some future research that could be conducted.

We struggled a lot with the literature study in the beginning, as we could barely find any information at all about touch interfaces on large screens such as 50" TV. Whether this was due to our research methods or the fact that it is a new field is uncertain. Regardless the solution was eventually to focus on touch interfaces and large screen interfaces separately, and then draw conclusions from both. This turned out to bring some unexpected problems for us. For example some gestures used in touch interfaces are designed for small screens.

### 7.1 METHODOLOGY

What was good and gave us more user feedback was the recording of the first user tests. By analyzing the video sequences a second time some things could be studied more in detail such as exactly what the user struggled with and if the user tried to tap elements that weren't supposed to be "tappable".

In the situation that we were in, with a user test in a near future, going straight on and constructing a high instead of a low fidelity prototype turned out to be a good strategy. Despite the fact that we could not spend as much time in the early stages of the design, we were very satisfied with our first prototype, much thanks to a good prestudy and luck with finding great software to develop it (Justinmind). We wanted to see the interaction with a touch interface first hand and faking this type of interaction was something that we thought was hard to get right. Thanks to a good first high fidelity prototype we got a lot of valuable feedback from the users testing it. This feedback was later used in our second prototype. However due to the relatively short amount of time spent with each user tester, we had to guide them at the beginning telling them where they were in order for the user test to not take up to much time. This wasn't optimal but it was the best solution in order to get the most out of each user tester. The problem with this was that we could not fully observe how the user would find all features in this prototype on their own.

Another thing that in retrospect would have been interesting to test, is whether or not the application was displaying too much information. We asked the users during the first user test if there were any unnecessary functionality, however not specifically about the amount of information. On the contrary though, this would be difficult to test in a short test session, but rather feedback from usage over a longer period of time would be most helpful.

During the brainstorming session we two were the only present ones. It could be argued if there should have been any other person present as well.

## 7.2 RESULTS

In general the most interesting result that we got from our research was that people were not keen on trying to perform other gesture input then tap inputs. Especially few tried to swipe the top and nurse menu in order to bring it forward, all just tapped the button to bring it forward. One could imagine otherwise since more and more people are getting used to the NUI's and have interacted with them for a few years now. On the other hand most NUI's that people have interacted with have been small ones (up to about 11 "), and in our testing environment (27") the area to swipe is much larger. So in one point of view it makes less sense to implement swipe gestures in a larger interface where a tap gesture is sufficient. But we believe as more and more people get used to using Windows 8, the will to try to explore more touch interactions with the application will increase. It's like when Steve Jobs first introduced the iPhone in 2007, he introduces the revolutionary interaction pattern "Slide To Unlock"<sup>27</sup> and back then the audience gasps because they had never seen anything like that. Today it's one of the most well-known interaction patterns in mobile touch technology and we believe that Windows 8 will encourage the will to try to explore more touch possibilities in future applications.

The interface could have benefitted from a lot more touch feedback to let the user know that the system is listening, this is critical in a touch interface according to Stephen Woods as mentioned earlier. During the testing we noticed several test persons tapping on objects that were just static information, and not meant to be interacted with, and situations like this could be decreased and less confusing with more touch feedback. We could have used more color and illumination to respond to touches, reinforce the resulting behaviors of gestures, and indicate what actions are enabled and disabled. One critical example is one thing that we tried to solve but didn't find a good solution to is when a user scrolls to the end of a list in our interface, in a normal touch interface the interface should lighten up at the bottom of the list to highlight to the user that they have reached the end of the list. We wanted to implement this feature but couldn't find a good technical solution to this in HTML, but it is suggested as part of the design.

---

<sup>27</sup><http://www.youtube.com/watch?v=9hUlxyE2Ns8&t=15m30s>

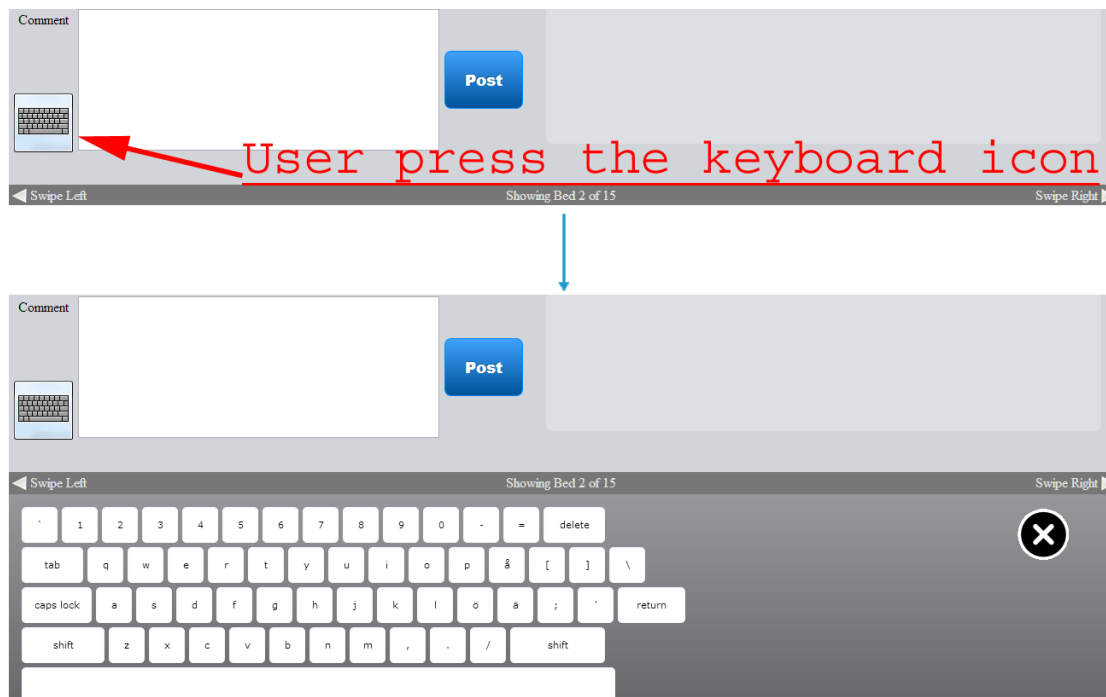


Figure 49: Showing an example of visual feedback that the user has reached the end. The image is taken from Android design guidelines. <http://developer.android.com/design/style/touch-feedback.html>

Another interesting topic that we discussed frequently during the project was the idea of tracking personnel. It was something that was talked about in the meetings with potential clients, and suggested by test persons in both user tests; however the main problem with this idea remains the same. Tracking personnel relies completely on them checking in and checking out wherever they go, and as we have been informed by several sources both from shadowing and experience from test persons; nurses often check in when they take an alarm or enter the room where there is an ongoing alarm, but often forget to check out. This would lead to very much false data, and if the feature would not be reliable it would not be useful. So short of tracking all nurses via GPS, this tracking feature could not be implemented in a reliable way. What could be done, if a tracking system was really needed or wanted, would be to be forthcoming with the limitation to the user, by instead adding a “last known location” attribute to each nurse. The user would then know that this is not an assumed position, but rather the last time that the system recognized where this nurse in question was located. This would eliminate the problem of false data, but it would also make the feature a long way from real time tracking, which is what we interpreted the user tester’s suggestions/requests to be.

We implemented our own keyboard that wasn’t that great to be honest so we ended up using Windows 8 native keyboard. The only drawback with the native keyboard is that its currently covering the text area where you post your comment (it has to be moved manually every time it opens), we tried to solve this by adding our implemented keyboard so that whenever the user pressed the keyboard symbol (or should also but not in our prototype pressed the text field) the keyboard would pop up just as in a mobile device. We were pleased with the aesthetics but not the interaction with the

keyboard so we skipped using it and still went with using Windows 8 native keyboard in our application although we kept the keyboard in our prototype just as a demo of how we thought it would look like in a finished product.



**Figure 50: Showing how we implemented our own virtual keyboard in the application.**

Continuing on this comment topic, one should have had a smart way of highlighting, cut/copy text like in a mobile OS in an easy way, like for example if a nurse has commented that a patient is taking a special medicine and it is hard to spell, then it should be easy to select that part of the text and then paste it in Fass. Although it's possible using the virtual keyboard the user needs four inputs of doing it, first double tap the text to be selected, press Ctrl+C then enter Fass view and press Ctrl+V and this is too many steps and not many people would realize that. This option wouldn't even let the user select multiple words but only single words or a single line (depending on if the user taps two or three times on the screen). Here it would be convenient with a contextual action bar that would pop up as soon as the user highlights text like in mobile devices providing the user with actions to be made, copy/cut/paste etc.

Also what we found out while doing our user tests was that it went rather slow to type on a virtual keyboard on screen, so one thing that is used frequently in mobile devices is some form of auto complete, and it would be extremely convenient to use it in this form of interface due to the fact as already stated typing on a virtual keyboard on a large touch screen is pretty hard. One way of solving this is that since this is supposed to be a complement to nurses phones comments could be made in their phones and then be shown in the large touch interface.

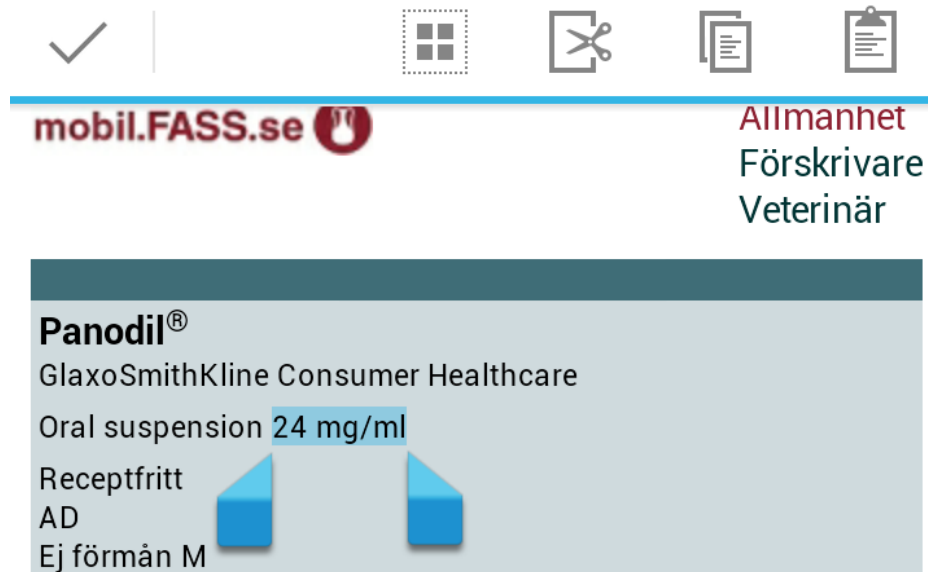


Figure 51: Showing a picture of a contextual action bar that appears when the user taps and holds down on a selected piece of text.



Figure 52: Picture of how it looks if the user has copied a piece of text and then "tap and hold" on another text field.

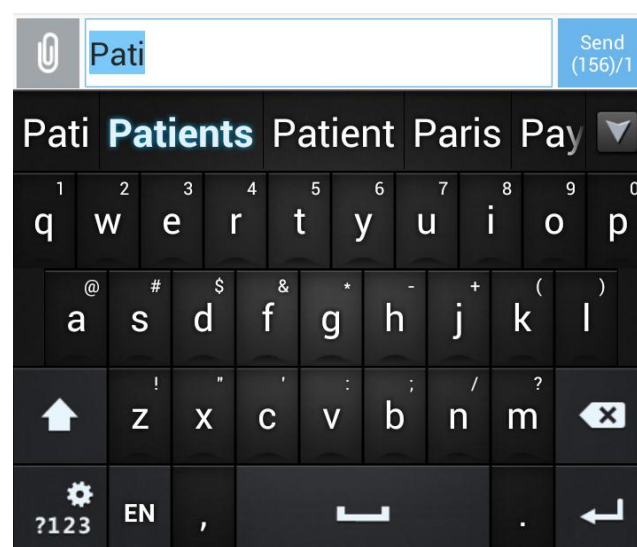
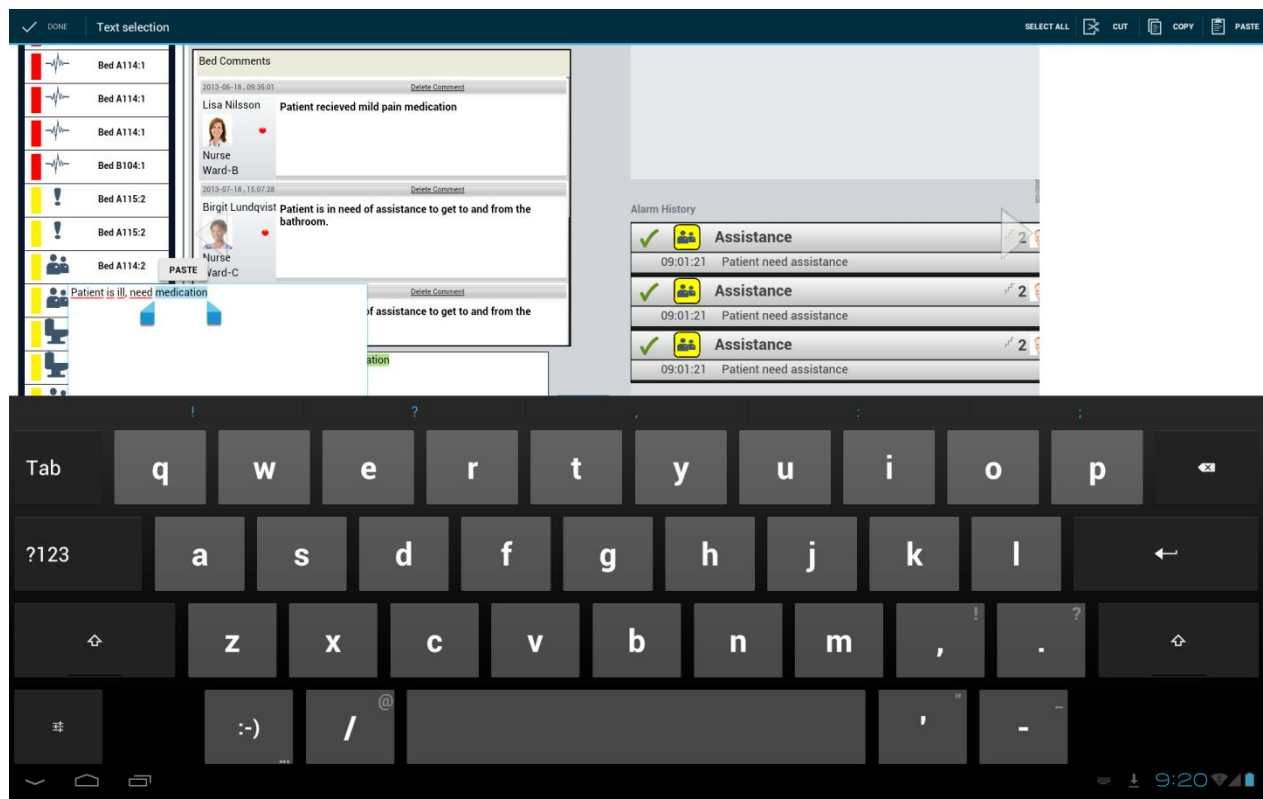


Figure 53: Showing an example of how auto complete is used in an SMS application.



We did some testing late on in the project to run our prototype on a new fast Android emulator called AndroidWindows<sup>28</sup> (Figure 51) to see how it visually would look if running in an Android environment and on a big screen (tested on a 27 " screen). Unfortunately there was some trouble installing the chrome web browser for the emulator so the native web browser was used which didn't make our application look that good and the touch screen didn't work either (mouse were used to simulate gestures). Still we got a pretty good opinion on how our application could look if used in an Android environment. Especially how the keyboard would behave and the contextual action bars were of major interest.



**Figure 54:** Showing how our application look if run in an Android environment on a 27" screen. It can be seen that the keyboard is pretty large and that the contextual action bar is accessible by using the "tap and hold" gesture on an element.

We had some problems in our application that easily could be solved if the application was a personal one and not an application that were used by many users. For example some people had a hard time finding the upper (top) and left (nurse) menu. In Android they have a similar problem, there they have something called a drawer (The main menu in the Android mobile app for Facebook has one for example) and its acting similar in

<sup>28</sup><http://www.socketeq.com/>

that way that you swipe from the left (or right) and a menu comes out just as ours, and its hidden to the user. In their guidelines it is suggested that the first time the user run the app it should start with the drawer in a state where its opened to highlight to the user that its “there”. In our case we have many different persons using our application and always showing these menus whenever the application runs is not feasible.

### 7.3 POSSIBLE IMPROVEMENTS /FUTURE RESEARCH

One question that one might ask is how well gestures that are specially designed for a smaller touch screen adapt to larger screens. As one can imagine, gestures that work well on smaller screens might not work that great on larger ones. This is a key question that should be researched before implementing new gestures. Another area of research might be if multi touch interactions should be present since they are better suited on larger screens than smaller. In our application we didn't find any suitable multi touch interactions and therefore they were not implemented.

One improvement that could be made is implementing swipe between top-level screens since we are kind of using tabs in our prototype's top menu (but not quite since our menu is inspired from both Windows 8 and Android). This is also a suggested guideline in Android that navigation between tabs should be possible with the swipe gesture. This was tried late in the project but we made a big mistake from the beginning when we started to implement our HTML prototype, namely instead of using only divs<sup>29</sup> and using SSI's (Server Side Includes)<sup>30</sup> we used an IFrame<sup>31</sup>. This resulted in all kinds of problems later on in the project, one of them making us unable to make use of the swipe view package that we used for detail Bed view and Personnel view, since in every single detailed view there was in a single div which was a requirement for the swipe expansion to work.

In our prototype a new alarm will be blinking with a highlighted color for 10 seconds. This is an arbitrary length estimated after second user test where we confirmed that 2 seconds was to short time, so it needs to be further user tested. We assumed that having all the ongoing alarms blink would cause stress, confusion or frustration, which is why we added this time limit, but again this is our assumption and need to be confirmed by user tests.

---

<sup>29</sup>[http://www.w3schools.com/tags/tag\\_div.asp](http://www.w3schools.com/tags/tag_div.asp)

<sup>30</sup><http://httpd.apache.org/docs/current/howto/ssi.html>

<sup>31</sup>[http://www.w3schools.com/tags/tag\\_iframe.asp](http://www.w3schools.com/tags/tag_iframe.asp)

One issue that one might see in our prototype is that we have redundant menu's enabling the same navigation options: top menu and the home button. In future revisions instead of showing top menu when the user swipes from the top, it could act more like a notification menu covering parts of the top screen, for example notifying the users whenever a new test result is ready. Navigation between views could still be made through swiping left/right and pushing the home button.

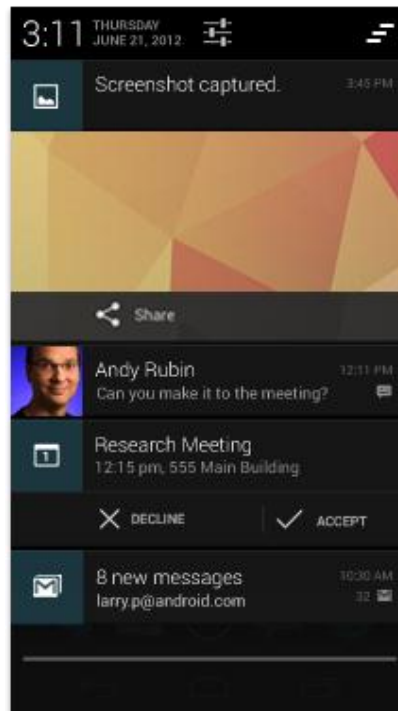


Figure 55: Showing an example of how notifications are displayed in Android.

The image is taken from Android design

guidelines.<http://developer.android.com/design/get-started/ui-overview.html>

## 8 CONCLUSION

When developing a touch application, it's important to follow the standards and guidelines for the target platform. For touch screens of a 50" size with the purpose of both being interactive and being viewed from a long distance, there is not a specific set of guidelines to follow. Both Android iOS and Windows 8 guidelines can be adaptable for this task, however not all of the guidelines applies. Additionally when designing an interface for this healthcare context it is important to be aware that users are working in a stressful environment. Hence the tasks performed with the interface need to be executed quickly.

In this chapter we will go through some of the most important guidelines, what should be considered and what should NOT be done.

### **Standards that should be followed:**

**Use an appropriate size of buttons;** fingers are a clumsy pointing device and making intractable objects too small will make them difficult to tap. The objects in the prototype are about three times the recommended size in mobile operating systems.

**Do not add too much functionality;** the users will not have time or the will to complete much work in front of a screen in a corridor. It's a tiresome position for the user and promoting extensive interaction could lead to the "Gorilla arm syndrome".

**Carefully consider gestures before implementing them;** not all touch gestures are fit for a large screen in this position. Swiping across the screen becomes a completely different interaction on a 50" screen compared to a smartphone. In this project the user testing could only be done on a 27" screen, so further testing on a 50-60" screen should be conducted before accurate conclusions could be made. For example if it's at all convenient including swipe in the real version.

**The application should be responsive;** touch feedback and low delay on interactions are very important for all types of touch interfaces and delay is very easily noticed.

**Be aware of the grouping of objects;** 50" is a big workspace, which makes it all the more important that frequent operations are placed close together. Fitts' law<sup>32</sup> is important concept to consider. The suggested screen size is 50-60", any bigger and it will become difficult to interact with, however if the screen is smaller, the visibility from a distance becomes compromised.

---

<sup>32</sup> <http://sixrevisions.com/usabilityaccessibility/improving-usability-with-fitts-law/>

**Pictures are faster than words;** Images gets the users attention and are faster recognized than text. Additionally if the image is interactive, it creates a larger visible area to tap with the finger as opposed to a linked text, which makes interaction smoother. So using pictures/images whenever possible is encouraged.

We believe that making the whole application visible from a distance of eight meters for all employees is unrealistic. What we suggest is to design the default view, and/or the “idle” state, to be visible at a distance of four (minimum viewing distance according to ISO 60601-1-8) to eight meters (depending on if the users have some kind of impaired vision). The rest of the interface should be designed as a two feet interface. Making the alarm list elements 70 pixels high with a font-size 39 was sufficient for a visibility of up to eight meters (whiteout impaired vision).

If the application will have multiple views, alternatively that the alarms are hidden in any of the applications states, make sure that new incoming alarms are still visible for the user. Preferably all ongoing alarms should to some extent be visible regardless of which state the application is in, as this is the highest priority of the application.

Some kind of introduction feature was requested by several users in the test conducted in the project. This should be easily spotted in the interface and cover the most basic functions. In our case we simply implemented it as a button labeled with a question mark.

A back/home button was requested by several users and should be in the interface, so that the users always can find a way to get “home”.

**Design scrollbars to fit their purpose;** as opposed to desktop applications, scrollbars are only meant to be an indication of the location in the list, not an interactive object to navigate through the list with. Therefore they should be slimmer than the average desktop scrollbar, and without the up/down button at the top and bottom. It could also be considered to have the scrollbars fade out when the list is not interacted with, however in our case we chose to let the scrollbars remain visible to further visualize that the list is in fact scrollable.

**Visual static hinting;** is critical in a touch interface, since users can’t hover over any objects and see which are clickable.

Especially for descriptions or other form of texts, the following Android guidelines are important (Android designer guidelines, 2013):

- **“Keep it brief.** Be concise, simple and precise. Start with a 30 character limit (including spaces), and don't use more unless absolutely necessary. People are likely to skip sentences if they're long.

- **Keep it simple.** Pretend you're speaking to someone who's smart and competent, but doesn't know technical jargon and may not speak English very well. Use short words, active verbs, and common nouns. Use terminology that you make sure your users understand. In an application designed especially for the healthcare such typical healthcare "language" could be appreciated by the users.
- **Put the most important thing first.** The first two words (around 11 characters, including spaces) should include at least a taste of the most important information in the string. If they don't, start over
- **Describe only what's necessary, and no more.** Don't try to explain subtle differences. They will be lost on most users.
- **Avoid repetition.** If a significant term gets repeated within a screen or block of text, find a way to use it just once."

The "Only show what I need when I need it" guideline is very important. The user will only interact with the system in short bursts and must have a clear overview of all information and hide all unnecessary information that is of minor importance at the time. A big mistake developers could do is to look at a larger touch interface and bring back old functionality from a wimp interface without first redesign it to better fit a touch interface.

## Standards that should not be followed:

### Personalization/adaptation

This is a public application, not a personal application, so any guidelines regarding customization or adapting to fit the individual user should be disregarded. The application should be standardized to fit all the users, not customizable to fit an individual user.

### Android/iOS specific UI

Unless the application is created as an Android or iOS application, the specific look of the corresponding OS should not be used (Color schemes etc.). The application should have coherency with other View applications, not necessarily with other Android applications.

### Size and form of objects

Specific size and form regulations and guidelines for buttons, icons and other objects, are optimized for smaller screens, especially Android/iOS. In some cases they can be applied, but in general such guidelines should be considered with much skepticism.



## 9 REFERENCES

- John C. Knight, Safety Critical Systems: Challenges and Directions; John C. Knight  
<http://www.cs.virginia.edu/~jck/publications/knight.state.of.the.art.summary.pdf>
- Donald A. Norman, Jakob Nielsen (2010) Gestural Interfaces: A step Backward In Usability  
<http://www.cse.chalmers.se/research/group/idc/ituniv/courses/12/mc/NormanNielsen.pdf>
- IDC website (2013) <http://www.idc.com/getdoc.jsp?containerId=prUS24108913>
- Android designer guidelines (2013) <http://developer.android.com/design/index.html>
- Stephen Woods (2013) Building Touch Interfaces with HTML5 - Develop and Design; Speed up your site and create amazing user experiences
- Apple design guidelines (2013)  
[https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/Introduction/Introduction.html#//apple\\_ref/doc/uid/TP40006556-CH1-SW1](https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/Introduction/Introduction.html#//apple_ref/doc/uid/TP40006556-CH1-SW1)
- Ken Hinckley, Michel Pahud, and Bill Buxton (2010) Direct Display Interaction via Simultaneous Pen + Multi-touch Input <http://www.billbuxton.com/SID10%2038-2.pdf>
- Matthew Lombard, Robert D. Reich, Maria E. Grabe (2000)  
 Presence and Television; The role of Screen Size  
<http://www.wilcoxlab.yorku.ca/PresencePapers/Lombardetal2000.pdf>
- Myhometheater webpage 2013  
<http://myhometheater.homestead.com/viewingdistancecalculator.html>
- THX webpage 2013 <http://www.thx.com/consumer/home-entertainment/home-theater/hdvtv-set-up>
- Usability net 2006 <http://www.usabilitynet.org/tools/wizard.htm>
- Clayton Lewis and John Rieman (1994) Task-Centered User Interface Design: A Practical Introduction.  
[http://grouplab.cpsc.ucalgary.ca/saul/hci\\_topics/tcsd-book/chap-1\\_v-1.html](http://grouplab.cpsc.ucalgary.ca/saul/hci_topics/tcsd-book/chap-1_v-1.html)
- Linnea Fogelmark (2008) Nurse Communication Assistance - User-centred Design in Healthcare Context
- NUI Group webpage (2013) <http://nuigroup.com/go/lite>
- Fass webpage (2013) <http://www.fass.se>
- Ian Sommerville (2007): Software Engineering 8
- Windows design guidelines (2013)  
<http://msdn.microsoft.com/en-us/library/windows/apps/hh761500.aspx>



NUI Group Authors [Community Release] (2009) Multi Touch Technologies 1st edition  
[http://nuicode.com/attachments/download/115/Multi-Touch\\_Technologies\\_v1.01.pdf](http://nuicode.com/attachments/download/115/Multi-Touch_Technologies_v1.01.pdf)

Sus Lundgren, Martin Hjulström (2011) Alchemy: Dynamic Gesture Hinting for Mobile Devices  
[http://www.cse.chalmers.se/research/group/idc/ituniv/courses/12/mc/gesture\\_hinting\\_camera\\_ready.pdf](http://www.cse.chalmers.se/research/group/idc/ituniv/courses/12/mc/gesture_hinting_camera_ready.pdf)

Richard Cardan, Kate Wojogbe, Brian Kralyevich (2006) The Digital Home: Designing for the Ten-Foot User Interface  
<http://channel9.msdn.com/Events/MIX/MIX06/BTB029>

## 10 APPENDIX

### 10.1 APPENDIX A – SYSTEM USABILITY SCALE (FIRST USABILITY TEST)

**Instructions:**

For each of the following statement, mark one box that best describe your reaction to the product *today*.

1. **Jag finner denna produkt högst användbar i vården**  
Håller inte med | 1 | 2 | 3 | 4 | 5 | Håller med
2. **Jag finner denna produkt onödigt komplex**  
Håller inte med | 1 | 2 | 3 | 4 | 5 | Håller med
3. **Jag finner denna produkt användarvänlig**  
Håller inte med | 1 | 2 | 3 | 4 | 5 | Håller med
4. **Jag skulle behöva assistans för att använda denna produkt**  
Håller inte med | 1 | 2 | 3 | 4 | 5 | Håller med
5. **Menyerna i produkten var lätta att hitta/få fram.**  
Håller inte med | 1 | 2 | 3 | 4 | 5 | Håller med
6. **Meny namnsättningen var bra, jag visste direkt vart jag kom när jag klickade på olika knappar i menyn.**  
Håller inte med | 1 | 2 | 3 | 4 | 5 | Håller med
7. **Jag föreställer mig att personer skulle lära sig denna produkt snabbt.**  
Håller inte med | 1 | 2 | 3 | 4 | 5 | Håller med
8. **Det Virtuella Tangentbordet var lätt att använda**  
Håller inte med | 1 | 2 | 3 | 4 | 5 | Håller med
9. **Finns det onödiga funktioner som du aldrig skulle använda dig utav?**
10. **Finns det några ytterligare funktioner som du skulle vilja se i nyss testade produkt?**
11. **Övriga kommentarer**

## 10.2 APPENDIX B – FIRST USABILITY TEST TASKS (IN SWEDISH)

### 1. Alarm Historik

Navigera till Listview och visa alarm historik för säng A114:1

### 2. Lägg till kommentar på patient

Lägg till kommentar på någon utav Linn Anderssons patienter spelar ingen roll vilken.

### 3. Kontrollera att kommentaren sparades

- a) Navigera tillbaka till samma patient som du skrev en kommentar på i första uppgiften. (Om du navigerade via top menyn så försök att gå via högermenyn i detta steg eller vice versa)
- b) Ta reda på vilka utöver Linn Andersson som också är ansvariga för patienten i fråga.

### 4. Kontrollera vilka som jobbar på avdelning

Ta reda på vilka som jobbar på avdelning A respektive B

### 5. Fass

Navigera till Fass och gör en sökning på Panodil.

## 10.3 APPENDIX C - SECOND USABILITY TEST TASKS (IN SWEDISH)

**Test 1 (långt)**

1. Den vyn som du befinner dig på nu kallas för "Listview".

a) Försök att hitta och navigera till vyn som kallas "Bedview".

b) Ta reda på vilka som jobbar just nu, välj därefter en av dessa sköterskor och visa enbart denna personens patienter.

2.

a) Välj någon utav sängarna/patienterna och ta reda på vilka 3 sköterskor som är ansvariga för personen ifråga.

b) Lägg en kommentar på patienten/sängen, ditt lösenord är 5.

c) navigera på enklast möjliga sätt till nästa säng

3. Navigera tillbaka till Listview.

a) ta reda på hur många aktiva larm som finns på säng A114:2

b) Ett nytt larm kommer nu att komma in, ta reda på vilken säng det är och vad larmet innebär.

c) Förklara vad (du tror att) all information på larmet innebär.

4. Du behöver ha information om en medicin (för enkelhetens skull säger vi Panodil). Navigera till Fass Online, och ta reda på vilka doser och förpackningar som säljs receptfritt.

5. Ta reda på vilka som jobbar på avdelning A, utöver de som är på shift just nu.

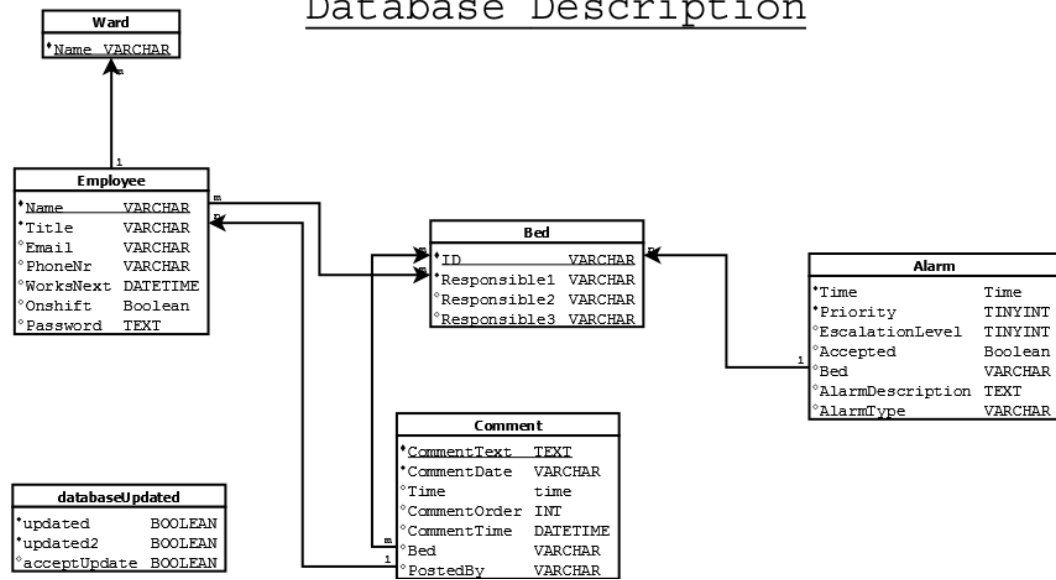
**Test 2 (kort)**

Välj 2 slumpvalda uppgifter utav 1-5 ovan baserat på vad tidigare test personer har haft svårigheter med.

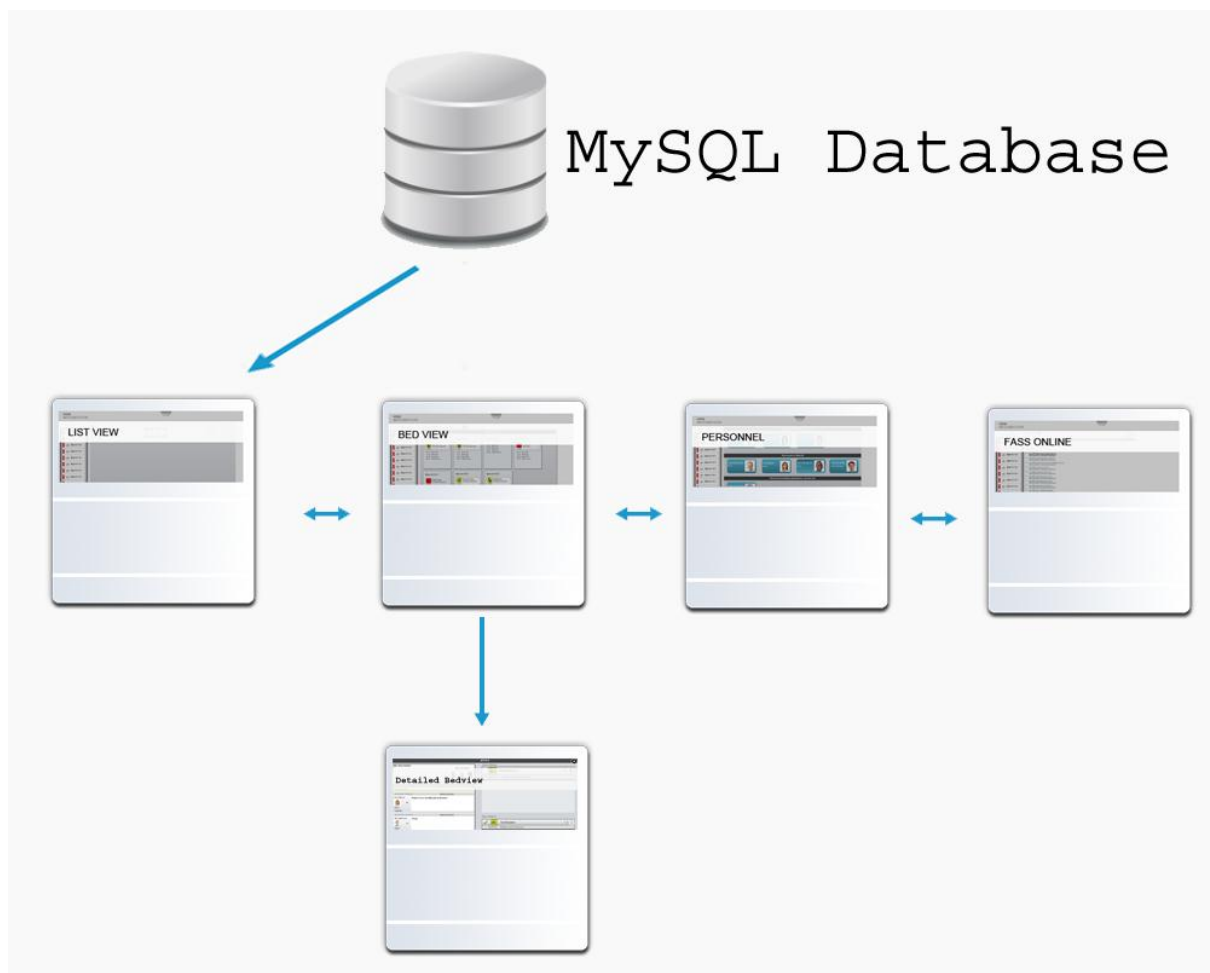
**Test 3 (kortast)**

Utforska applikationen och "tänk högt" beskriv vad du ser och hur du tänker. Försök att hitta så mycket funktionalitet och information som möjligt.

## 10.4 APPENDIX D - SYSTEM DATABASE DIAGRAM

Database Description

## 10.5 APPENDIX E - SYSTEM SITE MAP



## 10.6 APPENDIX F - FIRST USER TEST DETAILED STATISTICS

Fyll i ditt ID-nummer för kvällen:

c

E

B

f

A

D

**1. Hur gammal är du?**

49

49

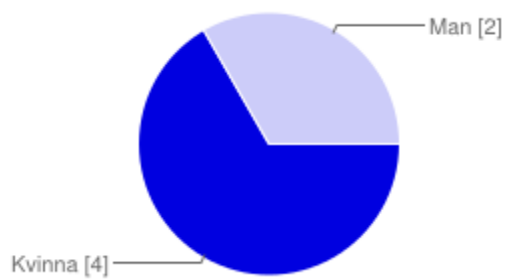
29

58

41

25

**2. Vad är ditt kön?**



Kvinna    **4**    67 %

Man        **2**    33 %

**3. Vad har du för utbildning?**

Sjuksköterskeprogrammet (kandidat) Specialistutbildning Psykiatri (magister)

Specialistsjuksköterska i psykiatri Omvårdnadshandledare

Sjuksköterska

iva sköterska, magisterutbildad

sjuksköterskeutbildning samt vidareutbildning inom intensivvård

Undersköterska

**4. Vad är din arbetstitel?**

Specialistsjuksköterska inom psykiatri

Specialistsjuksköterska i psykiatri Rätt psykiatrisk öppenvård

Sjuksköterska

sektionsledare

IVA-sjuksköterska

Samordnare på hemtjänsten

**5. Hur länge har du jobbat som sjuksköterska?**

7 år

16 år

1 år

35 år

17 år

1 år

**6. Vilken är din nuvarande arbetsplats?**

Tillnyktringsenheten, boende, Social Resursförvaltning, Göteborgs kommun

Rågårdens samt Järntorget's rättspsykiatriska öppenvård

Hemsjukvården Västra Göteborg

niva SU

CIVA, Sahlgrenska

Kållereds omvårdnadsteam



**7. Om du arbetar på en avdelning, hur många patienter finns det i genomsnitt på denna avdelning?**

5

18

16 lva-patienter och 10 postop-patienter

**8. Hur många olika patienter arbetar du med under en genomsnittlig arbetsdag?**

4

5

10

2

2

10

**9. Vad har du för arbetsuppgifter på din nuvarande arbetsplats?**

Ta hand om berusade personer, Blodprovstagnig (drogtest) Urinprov (drogtest), Omvårdnad av hemlösa personer

Fungera som stöd i utslussningen mellan slutenvården och öppenvården

Medicindelning, injektioner, såromläggning, arbetsledare för hemtjänst, åker svarar på larm från hemtjänst. Sektionsledare forskare och kliniskt verksam

Patientomvårdnad, läkemedelsutdelning, dokumentation, övervaka patient.

Samordnar mellan olika parter för att ge god omsorg för våra vårdtagare

**10. Vilka olika typer av larm hanteras på din arbetsplats?**

Skyddslarm (assistans, polis)

Överfallslarm D 62 ( sambanstelefoner) För kommunikation inom huset.

Telefonsamtal.

patientlarm monitorlarminfusionslarm

Monitor, respirator, infusionspumpar, dialysapparat

Trygghetsjour

**11. Hur upplever du mängden larm under ett normalt arbetspass?**

0-2

Hög de dagar jag jobbar på Rågårdens, slutenvården. Näst intill obefintlig de dagar jag är på öppenvården.

En person har "larmtelefonen" vilket innebär allt ifrån 0-50 samtal för ett pass.

det larmar mkt speciellt respiratorlarm

mycket till extremt 0

### **12. Har du tidigare erfarenhet av användning av Ascoms produkter?**

Ja, Psykiatri SU/ Östra sjukhuset

Jobbat på IVA anesthesi östra sjukhuset, med en del monitorer/ övervakningssystem.

Ja, larmklockor på sjukhus. Telekom-kommunikation med talfunktion på äldreboende.

ja, telefoner

Telefoner

nej

### **13. Vilken typ av mobiltelefon använder du privat?**

iPhone

iPhone och Samsung i jobbet

Samsung S3

iPhone

Samsung S3

Samsung s2

### **14. Använder du en smartphone med touchskärm eller Tablet (såsom iPad) ?**

Nej

iPad hemma. Smartphone för mail i jobbet

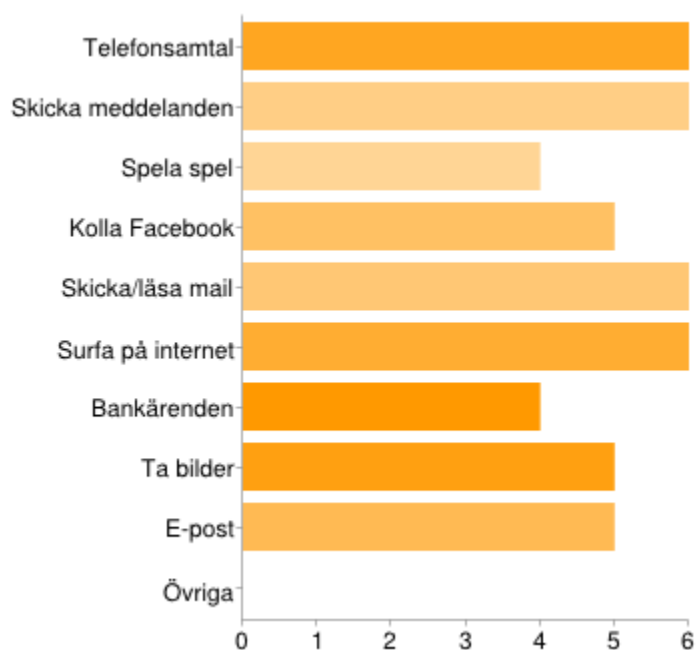
Hemma (och i jobbet för att söka information på t ex Fass.)

Smartphone, min egen

Samsung S3 och iPad

Hemma

### 15. Vad använder du vanligtvis din mobiltelefon till?



Telefonsamtal      **6**      13 %

Skickameddelanden      **6**      13 %

Spelaspel      **4**      9 %

Kolla Facebook      **5**      11 %

Skicka/läsa mail      **6**      13 %

Surfapå internet      **6**      13 %

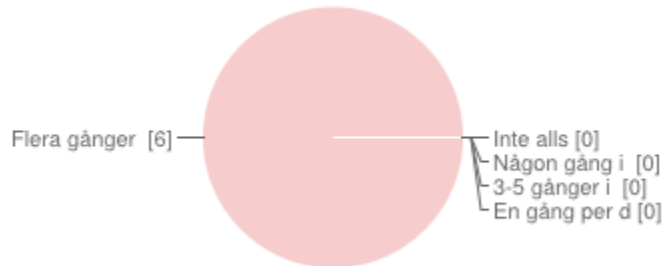
Bankärenden      **4**      9 %

Ta bilder      **5**      11 %

E-post      **5**      11 %

Övriga                      0      0 %

### 16. Hur ofta använder du internet?



Inte alls                      0      0 %

Någon gång i veckan      0      0 %

3-5 gånger i veckan      0      0 %

En gång per dag            0      0 %

Fler gånger per dag        6      100 %

### 17. Vilka datorprogram använder du hemma?

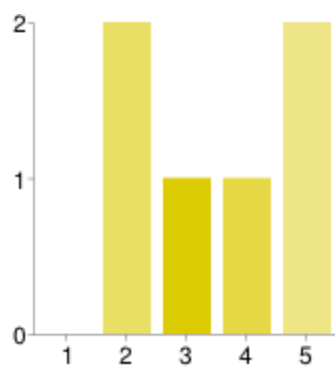
Windows

Word Internet explorer

Bildbehandling, ordbehandling, spel, + lite allt möjligt.

windows, Internet explorer, Microsoft Office

spel, bild och ordbehandlingsprogram

**18. Hur intresserad är du av teknik?**

1    **0**    0 %

2    **2**    33 %

3    **1**    17 %

4    **1**    17 %

5    **2**    33 %