Design of an efficient administrator application

- For the experienced user

LUND UNIVERSITY
Campus Helsingborg

LTH School of Engineering at Campus Helsingborg
Department of Computer Science

Bachelor thesis:
Myad Tahajody
Eric Nilsson

LTH School of Engineering
Lund University
Box 882
SE-251 08 Helsingborg
Sweden

LTH Ingenjörshögskolan vid Campus Helsingborg
Lunds universitet
Box 882
251 08 Helsingborg

Printed in Sweden
Media-Tryck
Biblioteksdirektionen
Lunds universitet
Lund 2015

Abstract

Resurs Bank provides various online services to retail stores. Accessing the services require authentication. The retail employee privileges are managed through a web application by administrators at Resurs Bank. The application needs improvement since it is lacking in functionality. Among other issues, the administrator is not given any feedback when making changes which lead to insecurities when using the application. Another desired function is the ability to manage multiple users.

An evaluation was made through observation sessions combined with open-ended interview questions in order to establish the most common administrator tasks. The tasks were mapped in flowcharts and a mock-up application was presented to the Resurs Bank supervisors before implementing the most effective workflow.

By performing the common tasks with the new application the efficiency improvements were measured. An example of quantitative data measured was the amount of context switches between the mouse and keyboard. The experienced improvements were communicated through interviews with the administrators.

Some of the improvements found when comparing the legacy application with the new application were:

- Creating, adding and deleting n users to a specific store “costs” 49n + 29 units based on the obtained quantitative values. With the new application the corresponding cost is 16n + 5, an improvement of 68 % given a large amount of users.

- The qualitative improvements returned by the test sessions showed that the user interface was more intuitive. This was due to a self-explanatory interface with color-coded feedback, such as green for successful operations and red for deletion and errors.

Visual feedback was implemented in the new application through the use of a color-theme combined with clear messages and instant form-validation. Animations were added to reassure the user of application activity and any changes made.

Workflow was improved by implementing an efficient search-box, providing more functionality in each page, and adding the support for managing multiple users.

The new application was deployed at Resurs Bank alongside the legacy application. Some additional work needs to be done before replacing the legacy application. In conclusion, the results show an improvement in both user experience and application efficiency.

Keywords: GUI, interaction design, flowchart, observation test sessions, mock-ups

Sammanfattning

Resurs Bank tillhandahåller diverse online-tjänster till butiker. Tillgång till dessa tjänster kräver autentisering. Privilegier för butiksanställda hanteras via en webapplikation av administratörer på Resurs Bank. Denna applikation behöver förbättras, t.ex. ger gränssnittet ingen feedback när ändringar utförs. Administratörerna blir osäkra om ändringar sparas. Det är heller inte möjligt att hantera flera användare samtidigt, en funktion som är högt eftertraktad av administratörerna.

En utvärdering gjordes av administratörens vanliga arbetsuppgifter i form av observationssessioner med efterföljande intervjuer där öppna frågor ställdes. Utvärderingen stod till grund för att ta fram alternativa flödesdiagram och en mock-up applikation. Dessa presenterades för handledarna på Resurs Bank innan det mest effektiva användargränssnittet, utifrån administratörernas behov, implementerades.

Effektivitetsförbättringar uppmättes genom att en rad vanliga uppgifter utfördes och kvantitativa värden såsom antal kontextbyten mellan mus och tangentbord mättes för varje uppgift. Den upplevda effektivitetsförbättringen förmedlades av administratörerna genom användartesterna.

Ett par exempel på förbättringar som uppmättes vid jämförelse av det gamla verktyget med det nya var:

- Att skapa, lägga till och ta bort n användare till en specifik butik ”kostar” 49n + 29 enheter baserat på ett antal kvantitativa värden. Med det nya systemet är motsvarigheten 16n + 5, en förbättring som konvergerar mot 68 % vid ett stort antal användare.

- De kvalitativa förbättringar som förmedlades av testanvändarna var bl.a. att gränssnittet var mer intuitivt. Detta tack vare självförklarande användargränssnitt och återkoppling med hjälp av färger, såsom grönt för lyckade ändringar och rött för borttagning av data och felanvändning.

Gränssnittet förbättrades genom att använda en kombination av en genomgående färgkodning och tydliga meddelanden. Animationer lades till för att försäkra användaren om att ändringar utförs eller har utförts.

Arbetsflödet effektiviserades genom implementeringen av en förbättrad sökruta, ökad funktionalitet per sida och stöd för hantering av många användare.

Den nyutvecklade applikationen levererades till Resurs Bank och körs parallellt med den gamla applikationen. Ytterligare funktionalitet måste läggas till innan den gamla applikationen kan ersättas. Sammanfattningsvis visar resultaten en förbättring av både användarupplevelsen och applikationens effektivitet

Nyckelord: Användargränssnitt, interaktionsdesign, flödesdiagram, observation- och testsessioner, prototyp

Foreword

We would like to thank the developer team at Resurs Bank, especially Niclas Fredriksson, Markus Kruse, Pär Nilsson, and Vlado Palczynski, for the support and guidance we have received during the development phase of this project.

We would also like to thank our mentor Mats Lilja for the smiles and the brainstorming sessions and a special thanks to our examiner Christin Lindholm for the comprehensive feedback at LTH School of Engineering.

Myad Tahajody & Eric Nilsson

List of contents

1 Introduction

Resurs Bank is a bank specialized towards customer financing purchases through both in store and online shopping partners. With approximately 25 000 stores connected to their services, it´s one of the largest banks in the retail financing business in the Nordic region (Resurs Bank AB, 2015).

In order for the staff in stores to access Resurs Banks services, authentication is needed. The retail clerks are given roles with different privileges, i.e. an RBAC (Role Based Access Control) system. All stores, users and roles are managed by administrators at Resurs Bank. The database containing user privileges is managed through a web application – in this thesis referred to as *the legacy application* – which is built using plain JavaScript, HTML and minor CSS.

At the request of Resurs Bank, a new web application has been developed during the writing of this thesis. The AngularJS framework was used together with Twitter Bootstrap to develop the requested application, in this thesis referred to as *the AngularJS application* (see section 4 Technical background, for more information on the tools used).

1.1 Problem description

The interface of the legacy application does not give feedback when it should, e.g. when deleting or updating a user. There is no input field validation when entering information and there are no confirmation messages or visual cues as to whether changes made were successful, resulting in insecurities when using the legacy application.

Question 1: How can visual feedback be implemented to erase the experienced insecurities when using the legacy application?

Functionality is missing such as the possibility to manage many users at the same time. When editing multiple users and assigning attributes the administrator must exit each part of the legacy application to reload specific parts and switch back and forth between views. The non-consistent workflow within the legacy application must be re-evaluated so that the AngularJS application allows the administrators to work faster and more efficient. Figure 1 shows a deep, non-consistent workflow and a wide, consistent workflow. The deep workflow is common when mapping tasks in the legacy application.

Question 2: How can the workflow depth be reduced for a given task?

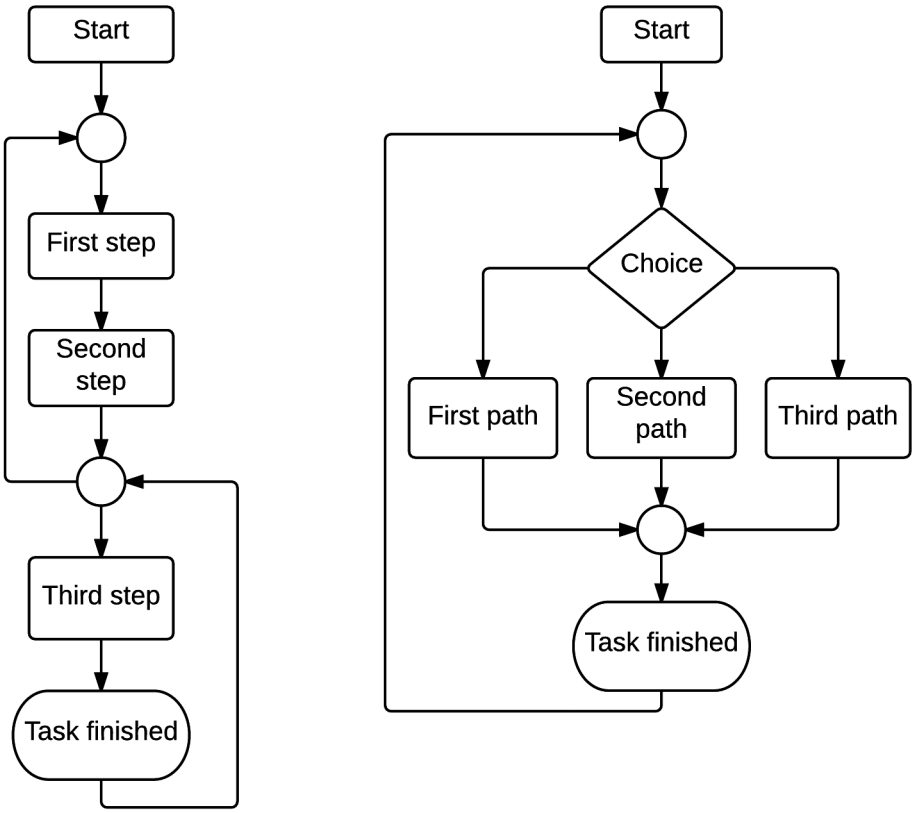

Figure 1: A deep workflow (left) with non-consistent start- and return point compared to a wide workflow (right) with a consistent start- and return point.

1.2 Purpose

Given the issues with the legacy application, the GUI (Graphical User Interface) has to be redesigned and give visual feedback when doing specific tasks. This has to be done with regards to the workflow efficiency, minimizing both the need to switch between keyboard and mouse, and extensive navigational operations in order to finish task.

The end-users of both the legacy- and AngularJS applications are experienced administrators, meaning they have a technical background with experience in user administration applications. The administrators should be able to perform their tasks in as few steps as possible, and they are imagined as self-learning and adaptive to new computer applications.

1.3 Scope

The scope of this thesis covers three main components:

1. Conducting observation tests of the legacy application (see section 2.2.1 Qualitative research, observation test sessions) with the goal to understand the main problems

of the application and constructing user stories for the AngularJS application (see appendix 8.6 User stories).

2. Prioritizing user stories to determine what functions to implement in the AngularJS application. The prioritized user stories to implement during the writing of this thesis are stories 1, 2, 5, and 7.

3. Observation tests and analysis of the AngularJS application to determine if the identified problems were resolved. Quantitative data was gathered (see section 3.2 Quantitative data) for both the legacy- and the AngularJS application. This data was used to determine any gain in efficiency (see section 5.3.3 Improvement comparison).

1.4 Limitations

User stories 3, 4, 6, and 8 were not implemented during the course of this project due to their lower priority, this was decided together with the project mentors at Resurs Bank.

Since the usability and the front-end interface is the priority during the development phase, any inefficiencies and limitations that exists server side are beyond the scope of this thesis.

Task completion time was not a measured variable in this thesis. During the observation test sessions the tester was urged to speak their mind and explain their behavior. This would lead to inaccurate task completion times being compared.

1.5 Expected result

The new workflow will have one initial choice branching out to a wide range of different functionalities. Rather than having a workflow tree with one single long branch representing the many steps of performing a task, the new workflow tree will have multiple shorter branches, resulting in fewer steps needed to perform a task. The new user interface will be more efficient and easier to use when performing the most common tasks and managing multiple users.

The AngularJS application will contain a fully functional store overview page as defined by user story 7. The new interface will also be more responsive than the legacy application interface as the user will receive feedback through visual confirmation cues on what has been executed or get descriptive error messages if the task could not be completed. When a task has been successfully completed, the administrator will be returned to the same default state to minimize any potential page hierarchy disorientation when using the application.

1.6 Terminology

AD	Microsoft's Active Directory service.
Administrator	A Resurs Bank employee using the External Authentication application.
AngularJS	A modular programming framework for HTML and JavaScript.
Context switches	When the user switches between using the mouse and the keyboard.
CSS	Cascading Style Sheets – is used for decorating HTML documents.
DOM	The document object model, i.e. representing objects in HTML.
GUI	The graphical user interface.
Legacy application	The web application used by Resurs Bank administrators prior to the completion of this thesis.
HTML	Hyper Text Markup Language – standard markup language for web pages.
Person	A user registered in the External Authentication database. Named 'user' instead of 'person' in the AngularJS application.
RBAC	Role based access control system, regulates access for users, e.g. in a corporation.
User	See person
Quantified task data	The parameters that were measured in this thesis i.e. context switches, actions, states, inputs and navigation choices.

2 Methodology

Upon starting the project, a Gantt chart was made to plan the different phases of the project (see appendix 8.1 Project plan, Figure 19). The contents and dependencies of the chart are made visible in Figure 2. Report writing spanned the entire project and was the ’fallback’ activity, should the development process come at a halt.

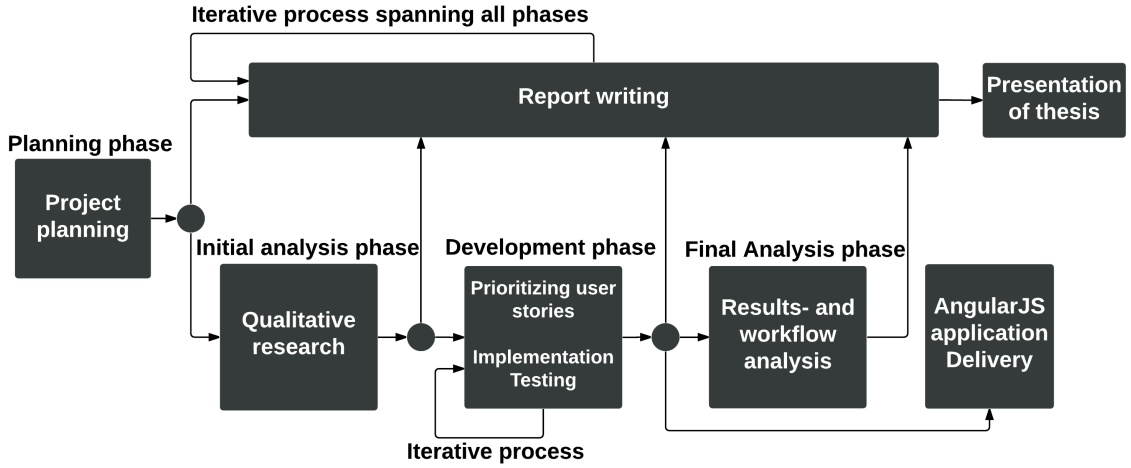

Figure 2: A flowchart showing how the phases of this project are related to each other.

2.1 Planning phase

The planning phase took place during the first week of the project and the goal was to set the ground rules.

2.1.1 Project planning

Since the project was done at Resurs Bank with several feedback meetings during the prototyping and development phase, a great number of changes to the application interface and functionality were expected. The goal was to quickly respond and adapt to change in order to deliver demos and working applications as soon as possible, with the customers’ wishes in mind.

This method of working iteratively in a project is known as Agile development (Kniberg & Skarin, 2010). Observation test sessions 1 and 2 (see section 5.1.1 Legacy application) were delayed when compared to the schedule, but the agile nature of the project enabled the authors to work on other topics while waiting for the availability of the observation test participants.

The Kanban model prescribes use of a board with example columns being ’Backlog’, ‘Bugs’, and ’In progress’. Trello, an online web application, was used for this purpose (see Figure 3).

The authors of this thesis placed the user stories or tasks on the different columns in which they belonged. The different tasks became visual with only a quick glance needed in order to see the work progress. The board was altered during development to adapt to the changes in the project.

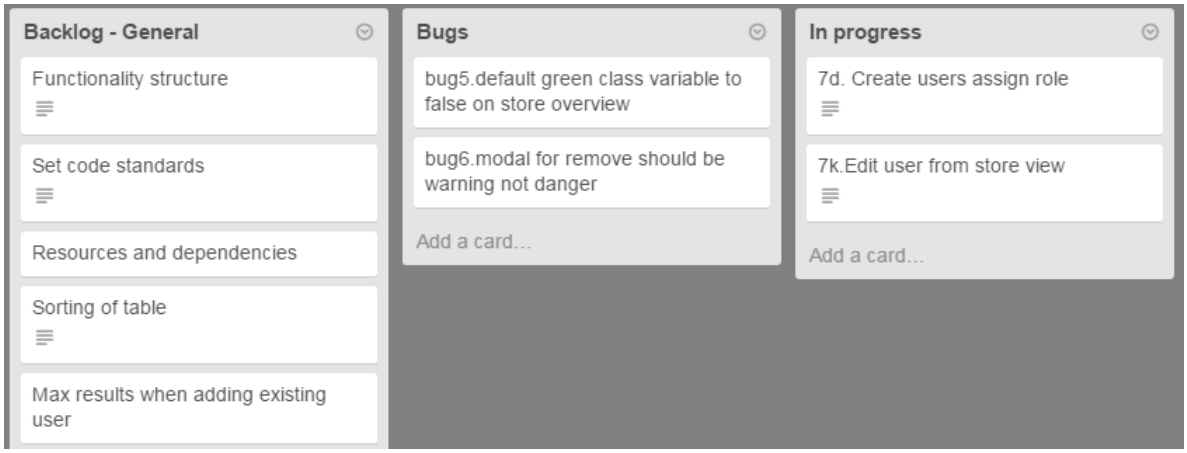

Figure 3: Some examples of the columns used during the development phase.

A WIP (work in progress) limit was set to a maximum of two cards per person. This limit prompted the need of a ‘Stuck’ column. If a task was delayed it was moved to the ’Stuck’ column and another card was quickly put into ’In progress’ giving the opportunity to stay productive throughout all project phases.

2.2 Initial analysis phase

The initial analysis phase set the ground for the development phase.

The flaws of the legacy application were found through interviewing the legacy application designer and understanding the main use of the application. Observation test sessions were held with experienced- and inexperienced testers. By introducing new testers insight was received on how to design an application with an obvious and self-explanatory *GUI*. Testers already familiar with the system gave information about the most common tasks and in what ways the legacy application could be improved.

Open-ended interview questions helped understand how the application is experienced. When mapping and analyzing the workflow in flowcharts any steps deemed unnecessary could be identified and removed. Quantifying task data gave measurable and comparable data (see section 2.4.1 Quantified task data).

Mock-ups were made in order to receive feedback (see appendix 8.5 Mock-up application). The feedback helped validate the functionality and design before implementing the AngularJS application.

Providing only quantitative– or qualitative data would leave doubts of the credibility of a thesis since one could choose only the type of data which validates the thesis (Silverman, 2001). Since this thesis takes both quantitative and qualitative data into consideration, the validity of the results will be strengthened.

2.2.1 Qualitative research

Prior to conducting the observation test sessions, research was made in order to determine the best practices. David Silverman claims in *Interpreting Qualitative Data* that different types of qualitative methods often are combined. No single method is sufficient to stand on its own (Silverman, 2001).

Contacting random users and having them perform tasks during a short amount of time could yield much less useful information than planning a proper test with a defined scope (Joseph S. Dumas, 1999). Gathering an authentic understanding of the user experience is usually the aim of qualitative research, and open-ended questions guarantee this authenticity (Silverman, 2001).

With this information in mind, the observation test sessions were combined with open-ended interview questions and audio-recording. If the participant asks: "*Is this where I can find the desired function?*" you could reply with: "*Where and how would you expect to find the desired function?*" Information is then immediately gathered by the observer while the participant is reflecting on what would be natural in the given application environment. If replied with a simple "*yes*" or "*no*" this valuable information would be lost (see appendix 8.3 Observation test session questions).

Observation test sessions

Observation test sessions help give insight as to what the tester perceives from the application and also helps determine the testers´ needs and expectations. This type of information is very valuable at any stage of developing an application and the process is widely used regardless of company size (Joseph S. Dumas, 1999).

As a test session guide one is required to be familiar with the product in order to ensure that the given tasks are performed correctly and to avoid time-consuming issues not related to completing the task at hand (Ingrid Ottersten, 2002) (Joseph S. Dumas, 1999).

Preparations were made before performing the tests. The authors of this thesis received training in using the application to order give proper guidance when the testers performed the tasks. A meeting room was booked and equipped with a computer to avoid distractions and disturb other employees at the office.

When defining what tasks to be performed during testing, the administrators were consulted in a short meeting (see appendix 8.2 Meeting with administrators). The most common tasks were found and listed together with the tasks that were perceived as taking the most effort to complete.

Testing could now be performed. This was done on separate sessions with one tester per session spanning a ~60 min period. Testers were given an explanation of the tasks before the start of each session and any questions were answered before testing so that the problems would be with the application itself, and not the tasks.

One of the authors of this thesis acted as an *observation session guide* continuously asking control questions to ensure that the tester was aware of what they were doing and to help

with their thinking out loud. The other author was the *observer*, taking notes of their thoughts regarding the application.

Open-ended interview questions

There are several guidelines to keep in mind when asking/answering questions as a session guide. Bias through leading questions and asking closed-ended questions is discouraged. The answers should only be what the user is thinking without any other party affecting their line of thought. Open-ended questions are preferred since the participant should constantly be reflecting on what they are experiencing. Responding to a question asked by the tester is best done with another question or a prompt encouraging the user to figure out the answer themselves (Joseph S. Dumas, 1999).

Some examples of appropriate questions and answers are as follows: "What part is the most confusing, and why?" instead of "Did you find this part confusing?" where the latter question is a leading closed-ended question. When asking what part is confusing and why, the user is forced to think out loud, identifying an experienced issue and reflect on why it was confusing. The answer can then be kept in mind when designing the new application (Ingrid Ottersten, 2002) (Joseph S. Dumas, 1999).

2.3 Development phase

After drafting the user stories a prioritization was made together with Resurs Bank. It was concluded that the goal of the project was to complete the *store overview page* (see section 1.3 Scope) with some added functionality supported by the back-end system.

2.3.1 User stories

User stories are a way of efficiently communicating software requirements to all parties in an IT project. They are used in agile software development processes and the technique aims to bridge the communication gap between project managers, developers, customers and company executives. Instead of writing complicated requirement documents early in the project that few people can understand, user stories are developed continuously throughout the project. This allows the project managers and developers to react to customer feedback after each demo session (Cohn, 2004).

The user stories in this thesis follow a specific pattern. There are three levels of hierarchy; the first level is *Epics* (1, 2, 3 …), the second level is *Detailed user stories* (a, b, c …) and the third is *Details/test cases* (i, ii, iii …). If any part of a user story was discarded, it was kept but written in "strike-through" font, see example in Figure 4.

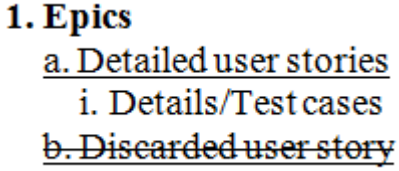

Figure 4: User story template

The epics are general user stories containing functionality that is too large and convoluted to implement all at once. In order to make them manageable they have been broken up into detailed user stories that can be prioritized according to any specific dependencies. Accompanying the detailed user stories are relevant parameters and specifications that are necessary when testing. This is why the Details/Test cases are often written as messages to the testers (Cohn, 2004), (Cowan, 2014), (Mountain Goat Software, 2015).

2.4 Final analysis phase

During the final analysis phase the same methods from section 2.2 Initial analysis phase, were used to analyze the qualitative data.

The flowcharts for both applications were analyzed and compared in order to determine any improvements. The quantitative data was processed according to the method presented in section 3.2 Quantitative data.

2.4.1 Quantified task data

Objective and quantifiable data was needed in order to show the improvements of the AngularJS application. The parameters in this section were measured for both applications and the result can be found in section 5.3 Quantified data.

Navigation choices

Navigation is one of the major usability problems listed in "The Scale of Misery" found in *Prioritizing Web Usability*. The focus is moved from actually using the web application when a user must think about how to navigate (Nielsen & Loranger, 2006). Multiple nested pages usually break the navigation within the application and confuses user (Krug, 2006).

Considering the importance of navigation, it was decided to measure the amount of navigational choices the user had to make in order to complete a given task.

Example: Alice is an administrator at Resurs Bank. She logs in to the legacy application and is met with a welcome screen containing three tabs in the header section: 'Persons', 'Stores', and 'Roles' (see Figure 7). Each time she switches between these tabs the 'Navigation' variable is incremented.

States

States represent an update of DOM elements on any given page. Since the variable is connected to navigation it would seem relevant to count the amount of states a user passes when completing a task.

Example: When Alice clicks on anything on the page, resulting in an update of the page content, a state is changed – incrementing the 'state' variable.

Context switches

When a power-user is using an application, they do not wish to switch between the mouse and keyboard often since it disturbs the working flow. The authors of this thesis find it

consistent with the principle of saving the user multiple steps whenever possible (Krug, 2006).

Example: When Alice is forced to switch between the keyboard and mouse in order to complete a specific step of her task, the ’context switch’ variable is incremented. When switching between input fields, e.g. first name, last name etc. and this is possible by using the ’tab’-key it is assumed that Alice does not use the mouse to highlight each input field since she is a ’power-user’ and wants to finish her tasks quickly.

Actions

For this thesis it was decided that an action performed with the keyboard is different than performing it with the mouse. Every click is an action, but if every keystroke counted as an action it would make counting actions useless. Filling out a form by tabbing to the next input field and using the ‘enter key’ all count as one action.

Example: Whenever Alice uses the mouse to click anywhere on the page the ’Actions’ variable is incremented. When she clicks on the first input field, fills out the form and clicks ’save’ this then counts as three actions. The first action is highlighting the first field in the form, the second is using the keyboard to fill out the form, and the third is clicking on the ’save’ button.

Input fields

The ’input fields’ variable represents the number of input fields that Alice needs to fill when performing a task.

2.5 Source criticism

The books used for the writing of this thesis have been located through the Campus Helsingborg library. By cross-referencing the contents, the reliability increased as they conveyed the same information. The books specifically address the interaction design and usability subjects, and even though some examples, e.g. *Graphical User Interface Design and Evaluation : A practical process,* are outdated the principles remain true. All books were published in either USA or UK except for *Användbarhet i Praktiken* which was published in Sweden.

The article *Kanban and Scrum* was given to the students during the ’year three project course’ by the computer engineering faculty and can therefore be regarded as valid source material. The article was published by C4Media on InfoQ.com with two authors and two commentators. It is mostly a comparison between Scrum and Kanban but thorough explanation of both project models made it an excellent source of information about Kanban.

Your Best Agile User Story by Alexander Cowen was found through online researching. The article set the basis for how the user stories were written and a similar approach could be found in the book written by Mike Cohn, author of *User Stories* and *User stories applied: for agile software development*. By comparing the findings the validity was confirmed. Mike Cohn is a founding member of the Agile- and Scrum Alliance and the author of other books regarding Agile development.

3 Analysis

The qualitative- and quantitative data collected during the initial- and final analysis phases was analyzed in the ways described in this section.

3.1 Qualitative data

With regards to the theories described in section 2.2.1 Qualitative research it was decided to use the combination of the three methods, with the audio-recordings being the fallback method in case the note-taker did not succeed in writing everything down. Since the sessions were relatively short and with one tester at a time it was decided not to transcribe the audio recordings. All relevant information from the session was written down since the testers clearly expressed their key concerns to the note-taker during- and after testing.

The interview notes were processed, clarified and presented to the Resurs Bank mentors at a meeting, during which a discussion was held as to how improvements could be made to the interface.

When the tester performed a task but the application did not respond as expected the thoughts were found through open-ended interview questions, such as "How do you feel when…(a given situation)".

The discussion set the foundation for coming up with the user stories which were designed to specifically improve on both functionality and the user interface. User stories 1 and -2 address the user interface, and user story 7 was the most prioritized user story functionality-wise during the development phase (see appendix 8.6 User stories).

3.2 Quantitative data

The authors of this thesis performed the tasks defined in sections 5.2.2 Legacy application and 5.2.3 AngularJS application and counted each occurrence of the variables described in section 2.4.1 Quantified task data.

General linear functions were calculated and are found in the 'Add', 'Delete', and 'Add and delete' columns in Table 3 and Table 4. The linear functions were summed up to create two overall functions, one for each application (see task total in Table 3 and Table 4). These overall functions were used to compare the legacy application to the AngularJS application. This comparison was visualized in Chart 1 and Chart 2.

The result of the quantified task data can be found in section 5.3 Quantified data result, and is structured as depicted in Figure 5.

$$a * n + b$$

Figure 5: A generalized linear function

Where a is the type of quantified task data, n is the number of users, and b is the number of quantified task data independent of the number of users.

4 Technical background

Various tools and applications have been used during this project. Some of the tools have been helpful when developing and planning and others when creating and analyzing workflows.

4.1 AngularJS

AngularJS is a JavaScript framework created by Google Inc. that allows for effective and modular development. It utilizes a binding mechanism between the view, what the user sees, and the model, where the data calculation is executed. This allows for real time view manipulation such as live field validation (see Figure 18) and committing changes without having to reload the webpage. AngularJS is well documented and the online community is large with many active developers.

The framework helped produce much functionality in a short amount of time, especially with asynchronous data requests, which were made simple using built-in services. The most prominent drawback was that it had a steep initial learning curve.

Website: https://www.angularjs.org/

4.2 Twitter Bootstrap

This is a free CSS theme used on websites to avoid having to create custom themes. Bootstrap is widely used and can be seen on major sites. It provides a simple and easy to use platform for creating user interfaces. The standardized visual components are intuitive for any user with clear colors and fonts.

Using Twitter Bootstrap during the development phase made it possible to focus on aspects of usability and functionality rather than graphical design and was easy to integrate with AngularJS. Because it is widely used, the application visuals might not be unique, but this was not a problem for Resurs Bank.

Website: http://www.getbootstrap.com/

4.3 Trello

Trello is a flexible organizational tool that enables the management of any given project (Trello Inc., 2015). It consists of columns containing cards, each of which can be moved between the columns. Each card can contain different types of information. The Kanban board was made possible with this tool and it has been of great help when organizing this project.

Website: http://www.trello.com

4.4 Lucidchart

This application was used to create all the flow diagrams and mock-ups used in this project. The tool is intuitive, versatile and easy to use. This resulted in fast and effective development of both detailed and extensive flowcharts and mock-ups. All Lucidchart documents can be accessed from the web through an account. They can also be worked on by multiple people and are updated in real time.

Website: https://www.lucidchart.com

4.5 GIT

GIT is a code version management system that enables for easy parallel programming. During development, the authors of this thesis used ’feature’-branches when programming new functionality and ’bug’-branches when fixing errors in the code. Each branch where based on the Trello board (see 4.3 Trello). All cards had an identifier which lay the basis for the GIT workflow, as all development branches were named according to the identifier, i.e. ’bug1/someBug’, and ’feature/7g.manageUserPassword’. This way of developing the application enabled the team to keep a ’master’-branch containing working code only, and merging feature-branches only when all tests in the user stories were successful.

Website: https://www.git-scm.com

5 Results

The results from the observation test sessions are presented in this section. The workflows are presented together with the GUI to show differences between the legacy- and AngularJS application. The results from the quantitative data are presented in tables and charts.

5.1 Observation test sessions

The tasks performed during the sessions simulated normal work routines. Audio was recorded during the entire session with the testers consent. An isolated room was reserved to minimize unforeseen disturbances and to make the tester more comfortable to speak her mind. A detached test environment was used to avoid contamination of the live database.

5.1.1 Legacy application

This section contains the results of the legacy application observation test sessions.

Session 1:

The test cases were executed by an experienced employee who has used the application many times before.

Date	2015-03-10
Tester	Malin, ~35/F
Test session guide	Myad Tahajody
Observer and note taker	Eric Nilsson
Duration	72 minutes

Findings:

Throughout the test session the tester showed clear signs of frustration and confusion. Even though she was an experienced user, many of the functionalities intended by the developer were used incorrectly. This indicates that the functionality is implemented in such a way that it is not intuitive. She often wondered “What is it doing now?” and “Why isn't it working?” due to lack of visual feedback from the application.

Listed below are the issues that occurred during the course of the test.

Issue 1	Confusion due to lack of feedback when assigning role and store to a user.
Explanation	The tester was uncertain whether the added role or store had been saved to the user automatically or that the *Update person* button needed to be pressed. Since the action does not provide any feedback the tester clicked it multiple times.
Occurrences	When updating a user with new roles or stores.
Recommendations	Provide feedback when information has been assigned.

Issue 2 When creating a new user there was a delay

Explanation After filling out the information fields correctly on the *Persons* view and pressing the *Create person* button there is sometimes a delay before the user is added to the list. This resulted in the user pressing the *Create person* button more than once causing the system to try to create multiple users with the same credentials. Since two or more users can't have the same user id only one user is created and the other creation attempt resulted in an error message.

Occurrences When adding a user.

Recommendations This can be avoided with some sort of progress bar indicating that something is loading.

Issue 3 When searching for a store to add to a user the tester clicked the *Add store* button with the intent to search, instead of pressing the enter key.

Explanation There was no *Search* button when searching for stores to add to a user, instead the tester had to press enter to commit the search. In other instances where you have to search for stores there is a search button.

Occurrences When adding a store to a user.

Recommendations Avoid inconsistencies when performing similar functionalities or tasks.

Issue 4 Unclear approach when removing a store/role from a user.

Explanation When in the user overview the roles and the stores that are assigned to the user are grouped together in a list under the collection name *Membership*. This resulted in two related problems.

1) It is difficult to decipher the list element type.

2) It is not intuitive that the list element needs to be highlighted before clicking the 'Delete membership' button.

Occurrences When removing a store or role from selected user.

Recommendations	1) Better categorizing of the different list element types to make them more distinguishable. 2) If no role/store is highlighted in the list the 'Delete membership' button is disabled.

Session 2:

The test cases were executed by an employee who has never used the legacy application before.

Date	2015-03-17
Tester	Margareta, ~40/F
Test session guide	Eric Nilsson
Observer and note taker	Myad Tahajody
Duration	60 minutes

Findings:

The main issues that arose from this session were those of confusion and uncertainty. Confirmation messages were expected, but not present, when updating information. Empty search results only showed a blank page, which was experienced as confusing.

The findings show the need to implement loading screens. Since asynchronous calls are being made, the data is not always updated immediately. The loading screens implemented block any interaction until the call is completed. It has a clear text and a spinning wheel, with the application background made darker. This clearly shows the user that the application is executing a task.

The lack of visual feedback was dealt with by highlighting the updated fields with a green fading color. This gives a visual confirmation that any changes were made and what data was updated.

Listed below are the issues that were experienced during the course of the test.

Issue 1	No feedback given when adding a user to a store.
Explanation	When a store is added, it shows in a list. There is no indication of whether the changes have been saved. *Update person* button was (wrongfully) pressed by the tester but no feedback or message was received. The tester navigated away from the user, searched for the store in order to confirm that the user was added.
Occurrences	Each time a store, user or role is updated with a member.
Recommendations	Give feedback through a confirmation message or a visual cue. Disable buttons that are not usable, or provide feedback upon clicking.

Issue 2	The list items in which membership information is shown were unclear.
Explanation	When a store, user or role is managed, a list of roles/stores/members is shown, but the list shows the information in an unedited AD format, I.e: “uid=test_t,ou=Persons,c=SE,dc=example,dc=org” This makes the information incomprehensible for the average application user.
Occurrences	For each list in a given store, user or role.
Recommendations	Use a standardized table for the membership information with clear column names and the content properly formatted.

Issue 3	Error handling was confusing when creating a new user.
Explanation	Because of a delay in when creating a user and updating the page the tester pressed *Create person* multiple times resulting in an error message that said “found another person with the same email”. The tester got the impression that the user already existed in the system and was unsure if the user shown in the list was the one newly created.
Occurrences	When adding a user and clicking *Create person* multiple times.
Recommendations	Disable the button or page when clicked with an explanatory text, e.g. “Person is being created, please wait”.

Issue 4	Empty page shown in *Persons*.
Explanation	Nothing is shown except for the sidebar with the search- and add functions when navigating to the *Persons* page. The tester was confused as to why the page was blank and thought it wasn't loaded properly.
Occurrences	When navigating to the *Persons* page.
Recommendations	Load all users into a list and present them when visiting the *Persons* page, or load only recently modified users, or show an empty list with a placeholder row.

5.1.2 AngularJS Application

This section contains the results of the AngularJS application observation test sessions.

Session 3:

The test cases were executed by the tester from test session 1.

Date	2015-05-12
Tester	Malin, ~35/F
Test session guide	Eric Nilsson
Observer and note taker	Myad Tahajody
Duration	40 minutes

This was the first time the tester had been in contact with the new application. The tasks defined for this test session were similar to the tasks when testing the legacy application with the end result being exactly the same. The differences were required due to an updated workflow.

Findings:

The tester could generally execute the tasks without any external guidance by the test session guide.

Listed below are the issues that were experienced during the course of the test.

Issue 1	The button that removed users from the current store was interpreted as the button that removed users from the system.
Explanation	Users can be removed from a store and still exist in the database. Instead of removing the user from the system, the tester removed the user from the store, failing the assignment.
Occurrences	Each time the tester was to delete a user from the system.
Recommendations	Changing the text on the button to *Remove from store* instead of *Remove user(s)* and making the button a different color than red. The *remove from store* and the *delete from system* could be located side by side on the same page making it obvious that the buttons do different things.

Session 4:

The test cases were executed by the same tester from test session 2. This was the first time she had been in contact with the new application.

Date	2015-05-12
Tester	Margareta, ~40/F
Test session guide	Eric Nilsson
Observer and note taker	Eric Nilsson
Duration	30 minutes

Findings:

The tester could perform the tasks except for issue 1 from session 3 (see 4.1.2). When performing the tasks the tester identified a solution to each task and committed to each solution with confidence.

The issues that were discussed in this session were the same as the ones from test session 3. The tester did not understand the difference between removing and deleting users.

The confusion from testing the AngularJS application may have occurred because the *remove from store* button was colored red and was named 'Remove user(s)'. The confirm dialogue that opened did state that the user was only being removed from the store and not the system but the tester did not take notice of this text and confirmed the dialogue. Since the tester is a power user she might have become over confident in her ability to use the legacy system and did not feel the need to read the confirm dialogue.

After sessions 3 and 4, minor changes were made to the interface. Figures 10 and 11 show the updated version, where *remove user* is colored yellow, and placed next to *delete user* which is colored red. The corresponding confirm-dialogues have the respective color, with explanatory text in the header.

5.2 GUI, flowcharts, and workflow analysis

In order to properly compare the different applications, the same tasks were analyzed and mapped to flowcharts which are shown in this section. Descriptions for the different flowchart elements can be found in Figure 6.

The tasks differ from the test sessions in that they test the way both applications handle adding and deleting multiple users which was reported as a common working routine by the administrators. The tasks are defined in the corresponding workflow analysis and slightly differ between the applications due to the difference in functionality (see sections 5.2.2 Legacy application and 5.2.3 AngularJS application).

The flowcharts also show the different possible actions from within the *stores view*. If a task is completed through many steps, the workflow is defined as deep. If many actions can be performed, the workflow is defined as wide.
Screenshots of both applications were taken and are meant to help visualize their distinct GUI.

5.2.1 Flowchart object descriptions

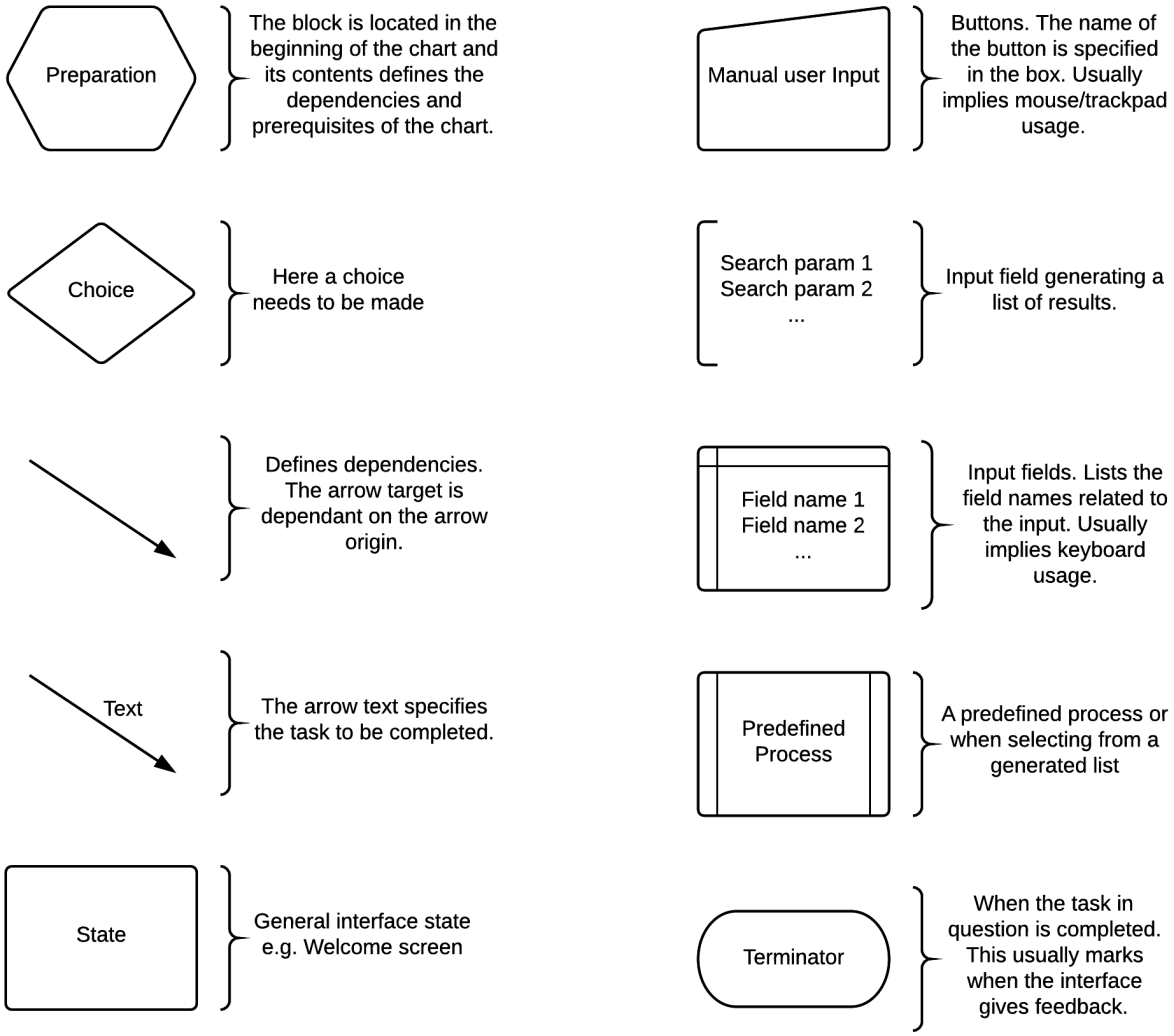

Figure 6: An explanation of the different symbols used in the flowcharts.

5.2.2 Legacy application

Preconditions: User has navigated to the application and logged in.

Task description:

1) Create two users using the information listed in the table below:

	Test user 1	Test user 2
First name	Eric	Myad
Last name	Test	Test
User id	EricTest	MyadTest
E-mail address	eric@test.se	myad@test.se
Password	*Generated*	*Generated*
Store	Test Store LTH	Test Store LTH
Role	ROLE_ADMIN	ROLE_ADMIN

Table 1: Test users to create in the legacy application.

2) Navigate to store overview for *Test Store LTH* and verify the number of users.

3) Delete the users created in step 1 from the system.

4) Navigate to store overview for *Test Store LTH* and verify the number of users.

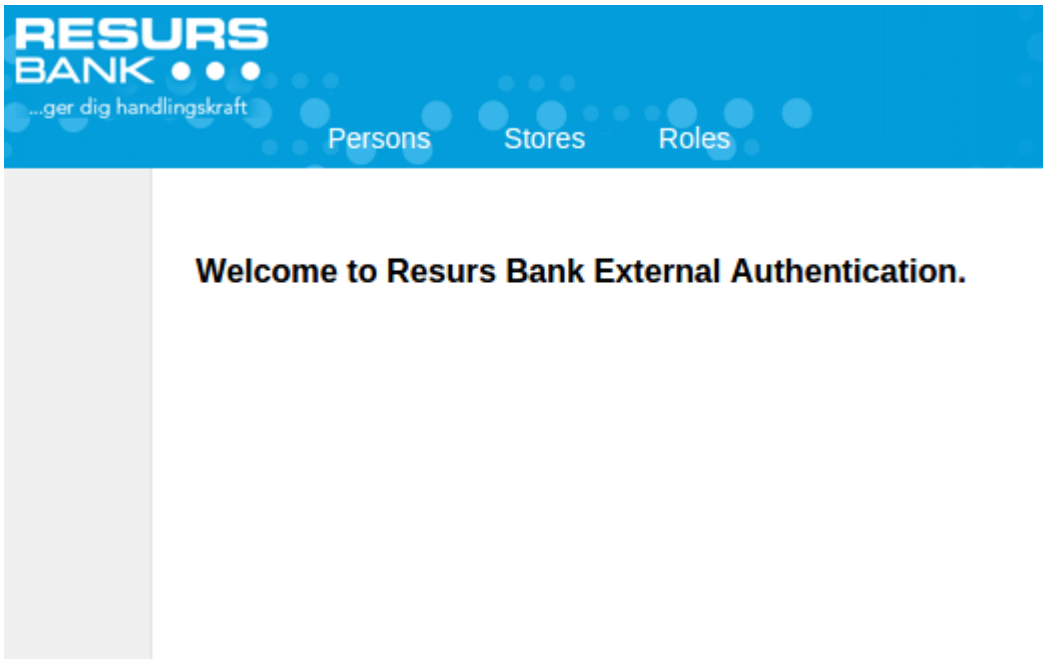

Figure 7: The welcome screen has no functionality, requiring an extra navigational choice before being able to work with the application.

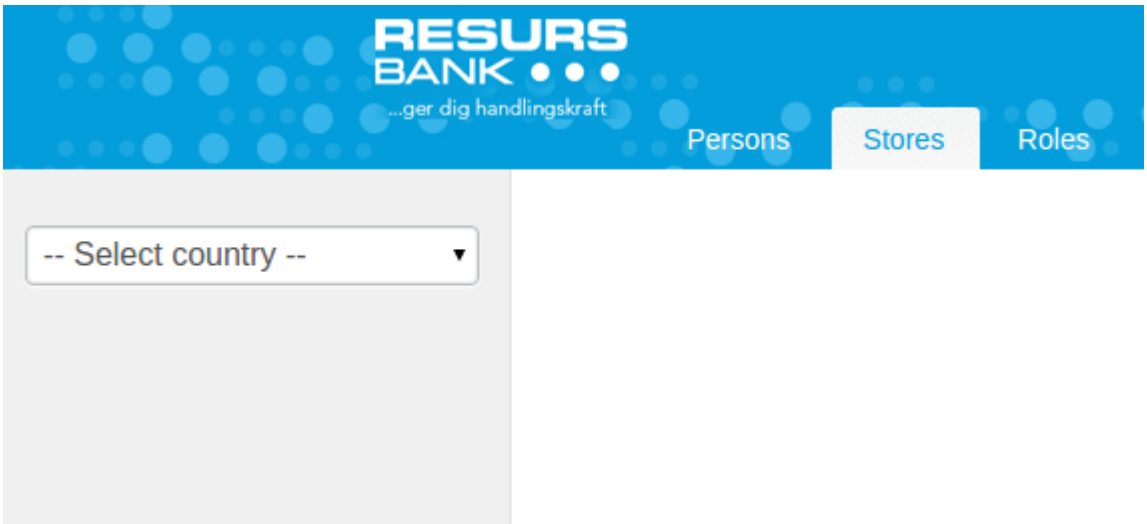

Figure 8: The obligatory country selection process adds another action to the workflow.

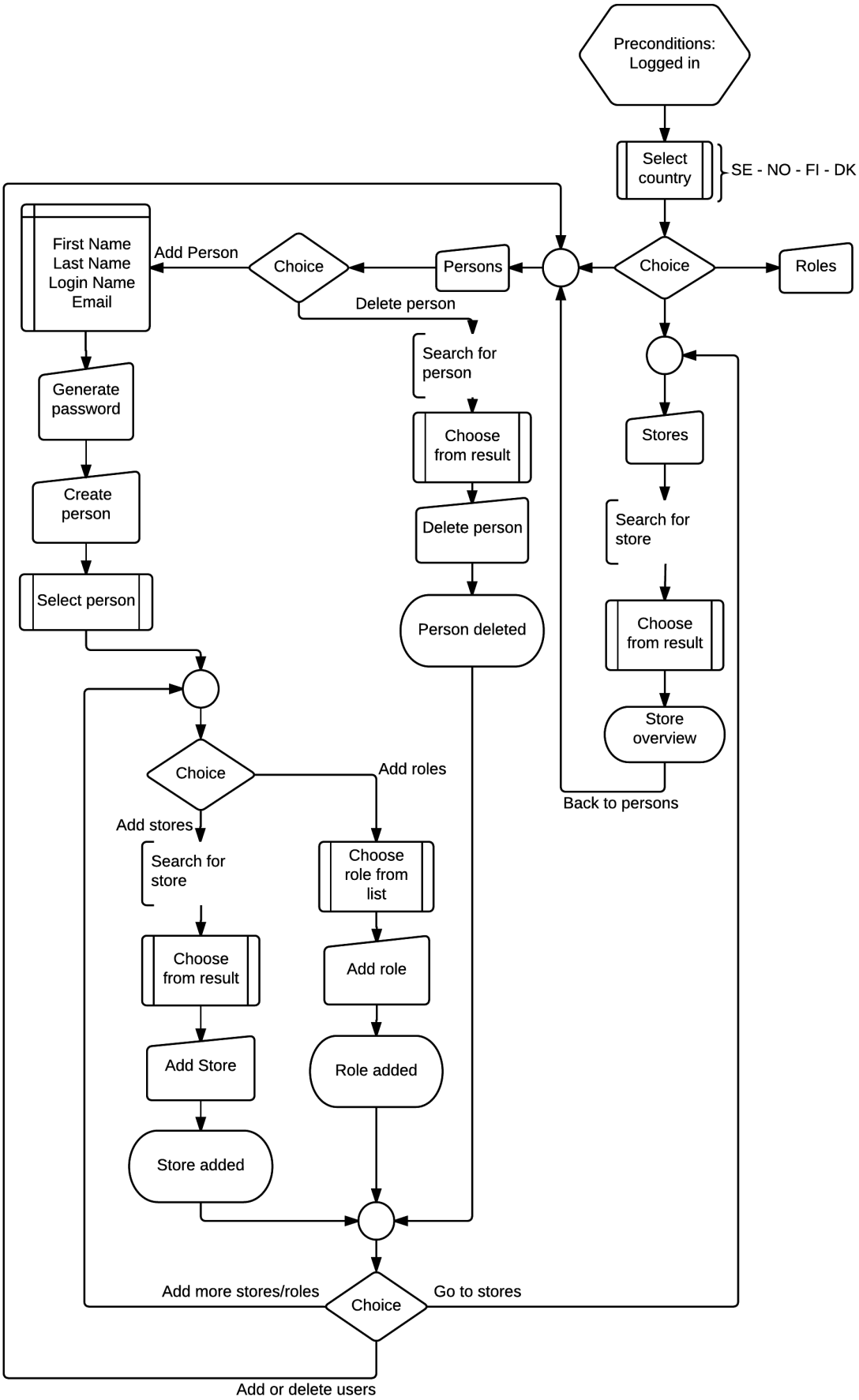

Figure 9: Completing the workflow analysis task (see 5.2.2 Task description).

Once the country is chosen in the session, the same country will be chosen by default when navigating between the *stores* and *persons* views. This was placed as the first step in the workflow to give a better understanding for this mechanism. The obligatory country choice was one of the steps the new application was going to get rid of (see Figure 8). The country dependency was a remnant from an old system constricting the external authentication application.

When navigating to a store, all memberships (stores and roles) are shown in a list which is presented in an AD domain name format (see Figure 10). A user is assigned to a role by adding them from a list, or to a store by first searching for the store, choosing the correct store from the list and then adding it to the user. The workflow is shown in Figure 11.

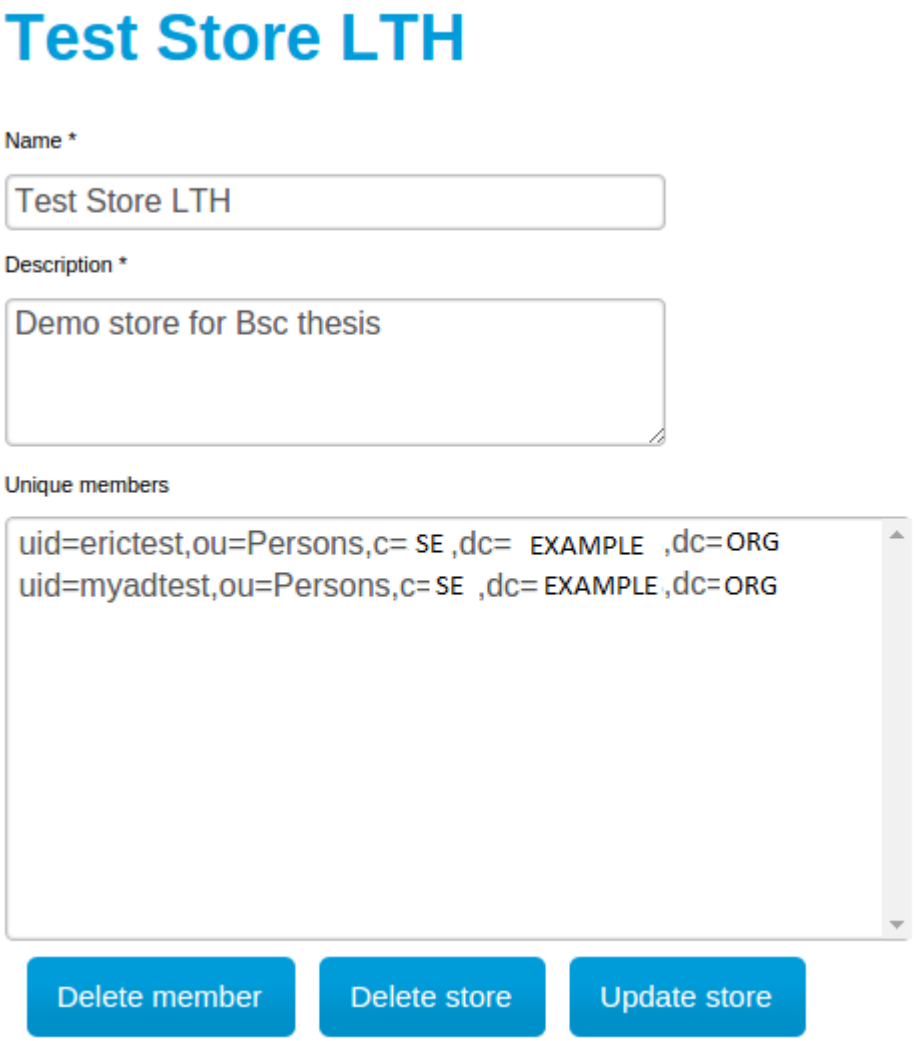

Figure 10: Store overview page. The AD formatted list of users is confusing and incomprehensible for the average user.

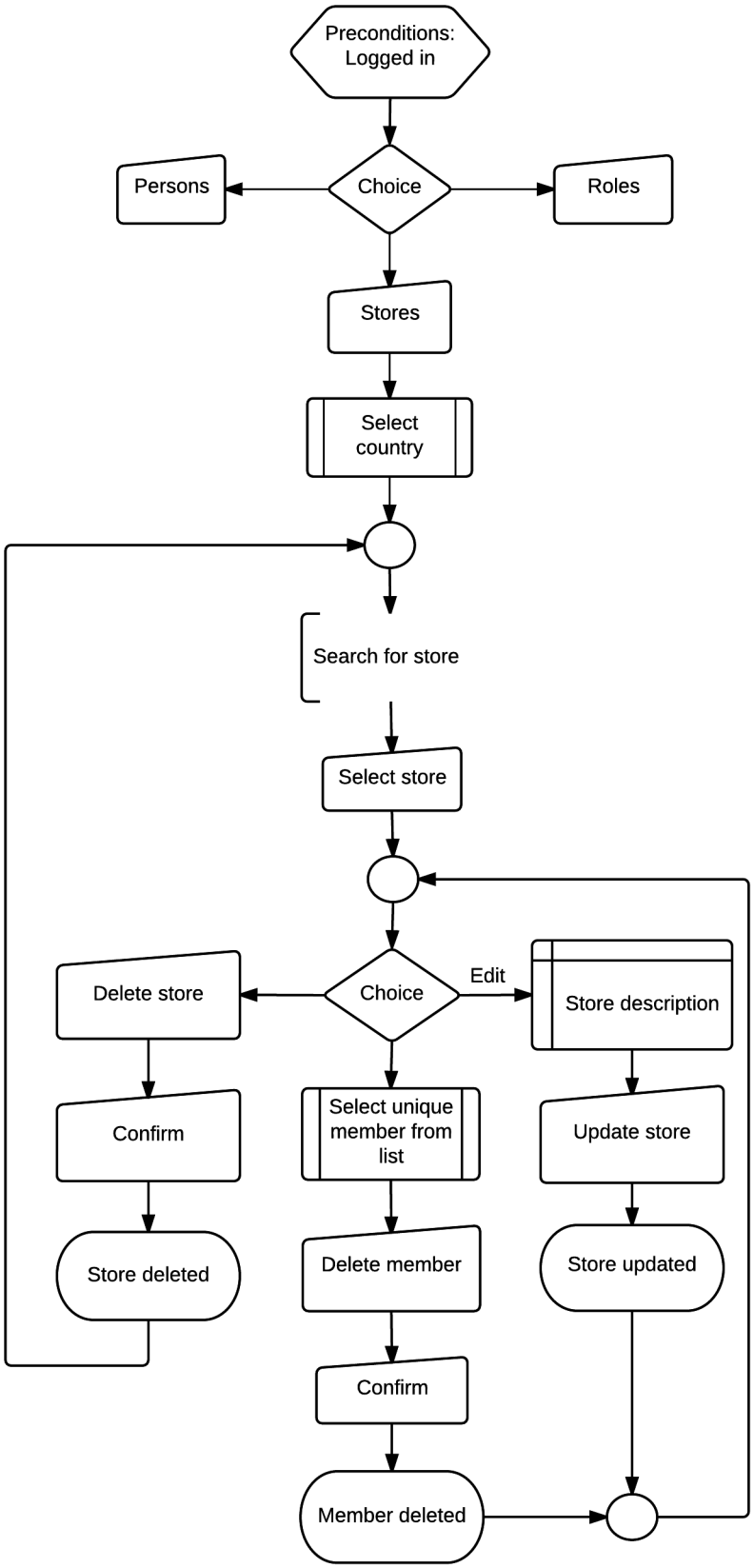

Figure 11: Reaching the store overview page in the legacy application and the available functions.

5.2.3 AngularJS application

Preconditions: User has navigated to the application and logged in.

Task description:

1) Navigate to the store *Test Store LTH*
2) Create two users using the information listed in the table below:

	Test user 1	Test user 2
First name	Eric	Myad
Last name	Test	Test
User id	EricTest	MyadTest
E-mail address	eric@test.se	myad@test.se
Password	*Generated*	*Generated*
Store	Test Store LTH	Test Store LTH
Role	ROLE_ADMIN	ROLE_ADMIN

Table 2: Test users to create in the AngularJS application.

3) Verify the number of users in *Test Store LTH*.
4) Delete the users created in step 2 from the system.
5) Verify the number of users in *Test Store LTH*.

Managing users from the store overview page is now possible. Given the extensive functionality in the store overview page the process of adding a new user to an existing store is done in fewer steps. The task workflow is realized by Figure 12.

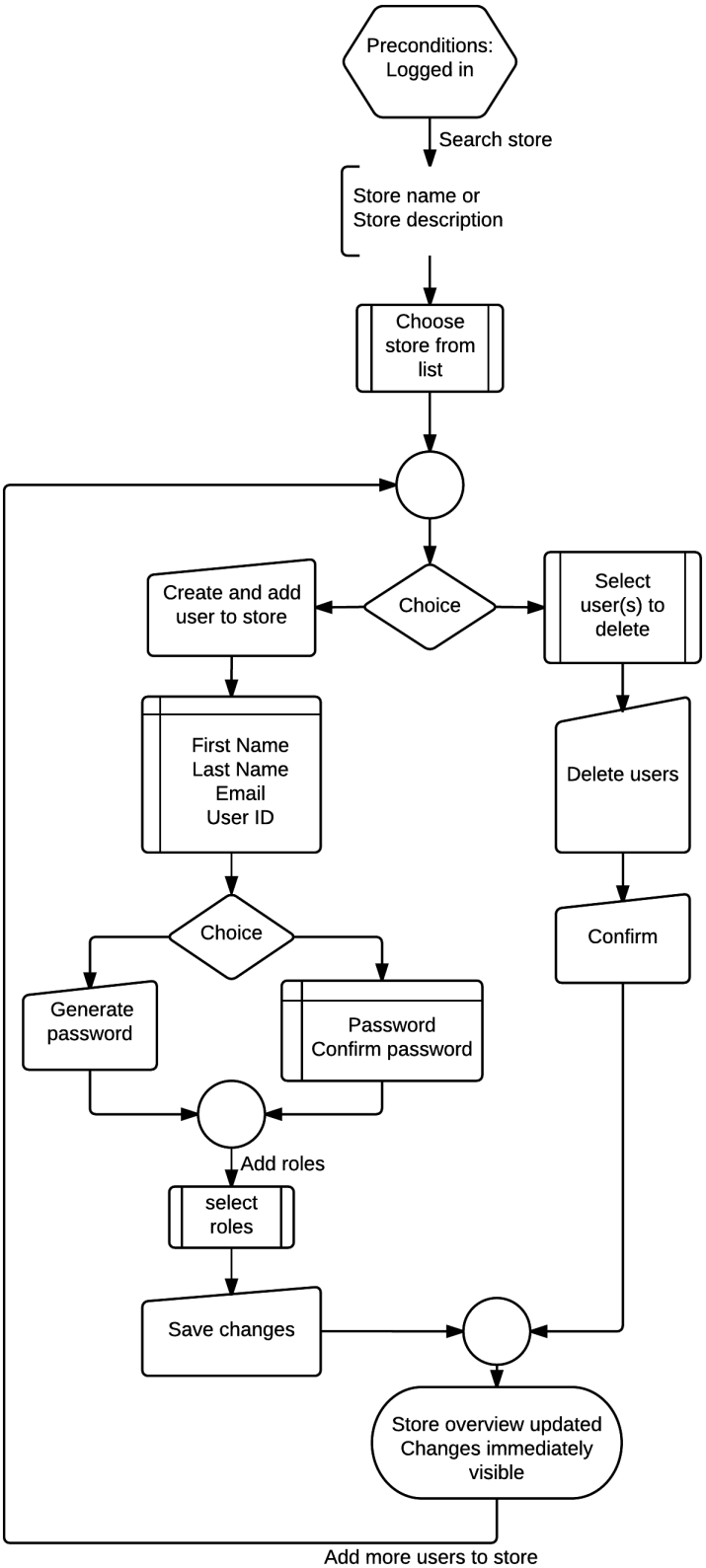

Figure 12: Task workflow for the AngularJS application.

Upon completing the task, the list of users on the store overview page is updated with a green highlight on the newly added row. The green highlight slowly fades after 1.6 seconds. This visual cue is a confirmation to the administrator that the changes made were successful.

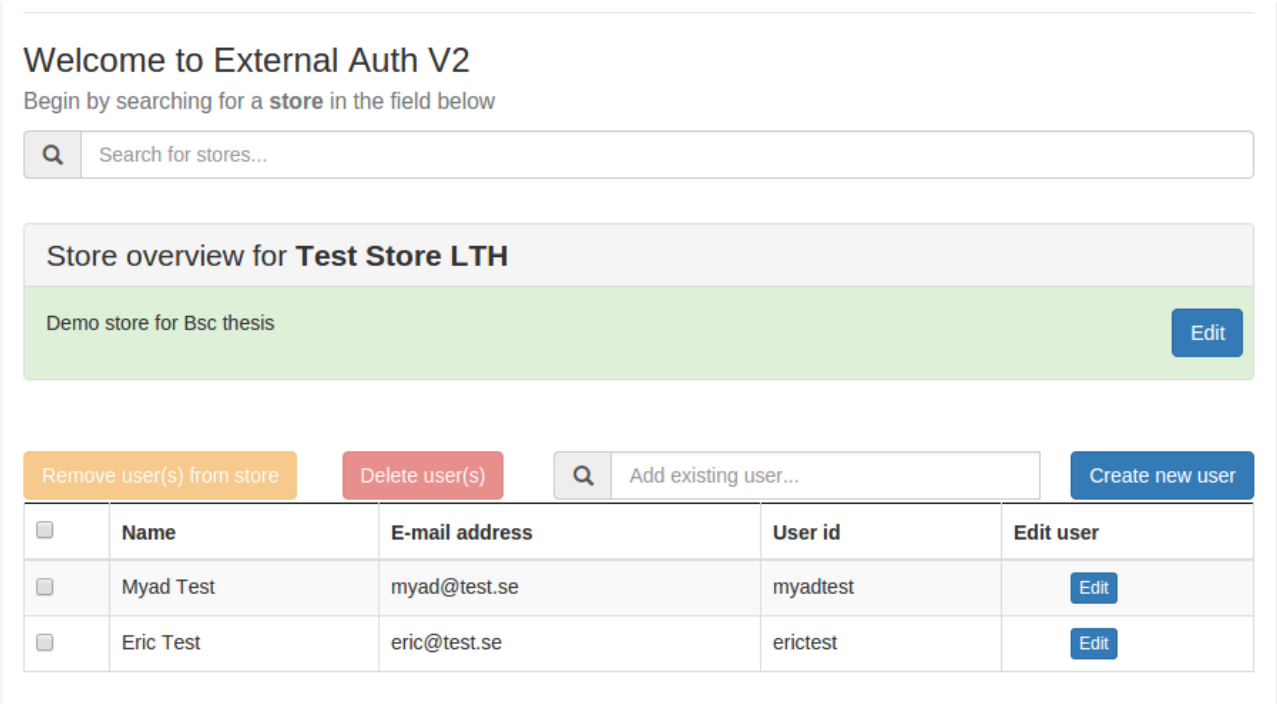

Figure 13: Store overview page with visual cues. The updated description field turns green and fades to white when changed. *Remove...* and *Delete…* buttons are disabled when no rows are selected.

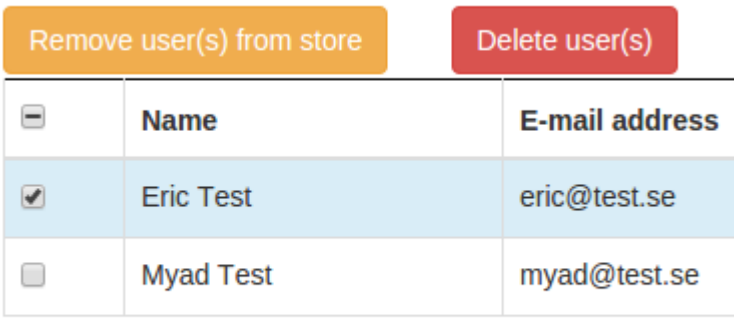

Figure 14: Visual cues are present throughout the AngularJS application, buttons activate upon selection and entire rows are color marked.

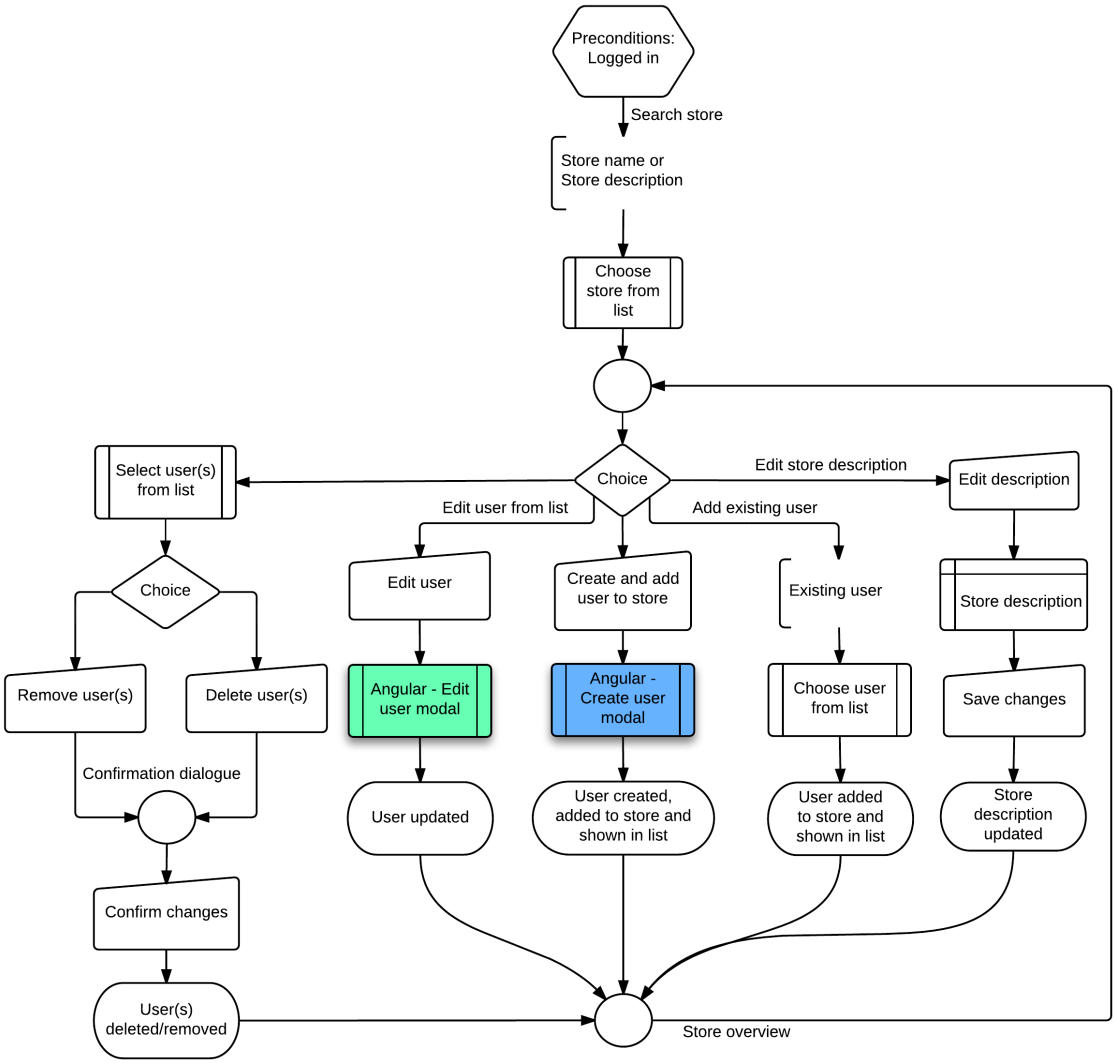

Figure 15: The functionality in the store overview. Compare to Figure 11

In addition to bypassing the country selection, a multitude of options are now available from within the store overview, such as removing/deleting multiple users or editing single users without leaving the page. The two *Angular modal* processes in Figure 15 are shown in Figure 16 and Figure 17:

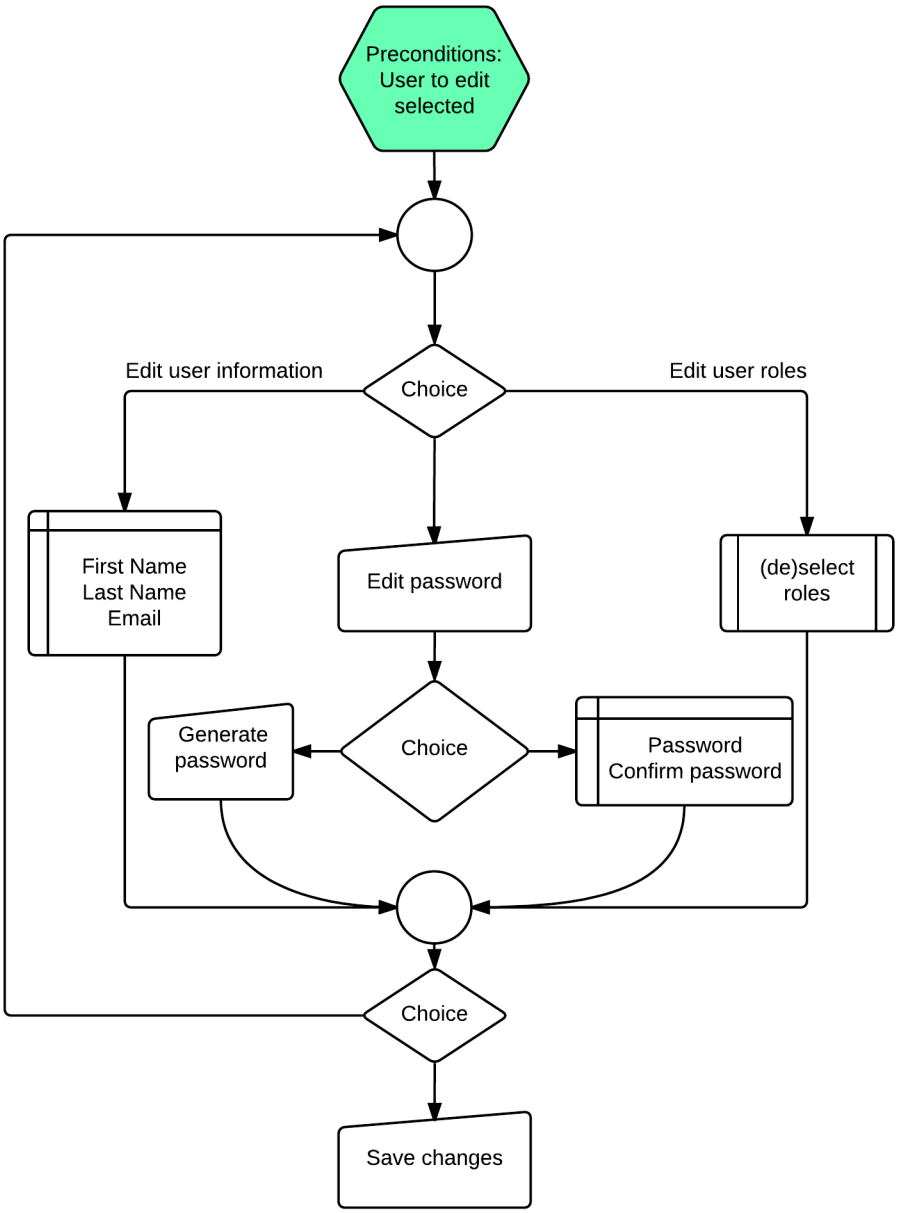

Figure 16: 'Edit user'-modal workflow

Editing a user is done in an arbitrary order, adding or removing roles is as easy as (de)selecting the roles from a list. A green shade is added to the selected rows to indicate selection. When filling out the password fields, instant validation is performed. The input field turns red showing messages such as: 'Passwords do not match' or 'Minimum length: 10 characters', and turns green when the complexity is fulfilled.

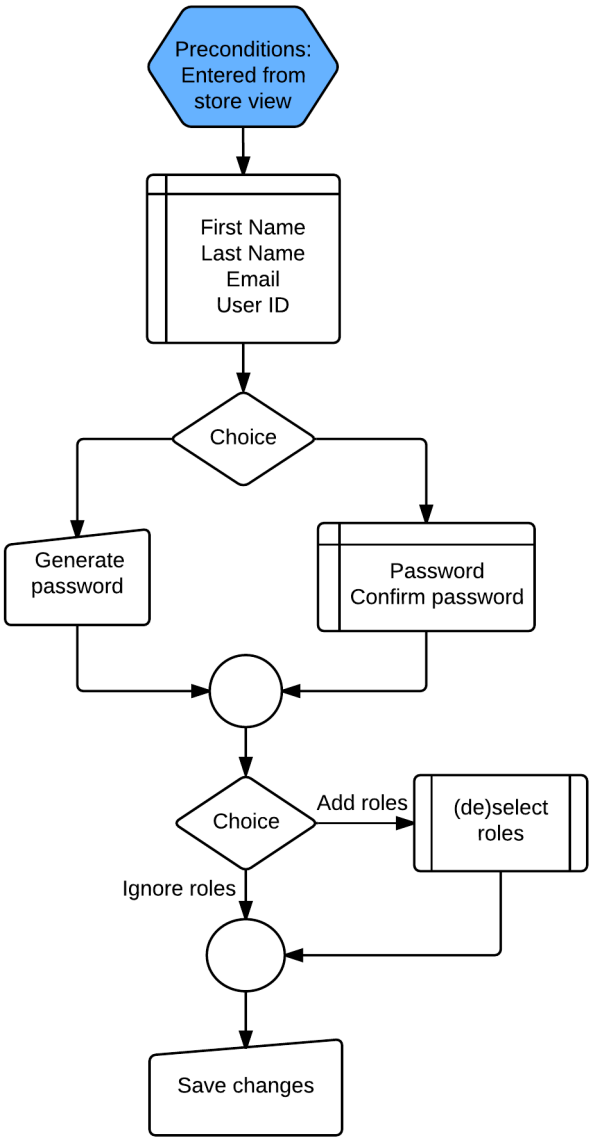

Figure 17: 'Create and add user'-modal

The *save changes* button will be deactivated as long as there are any issues with any of the input fields, which gives the administrator even more visual clues as to whether the input fields are valid or invalid.

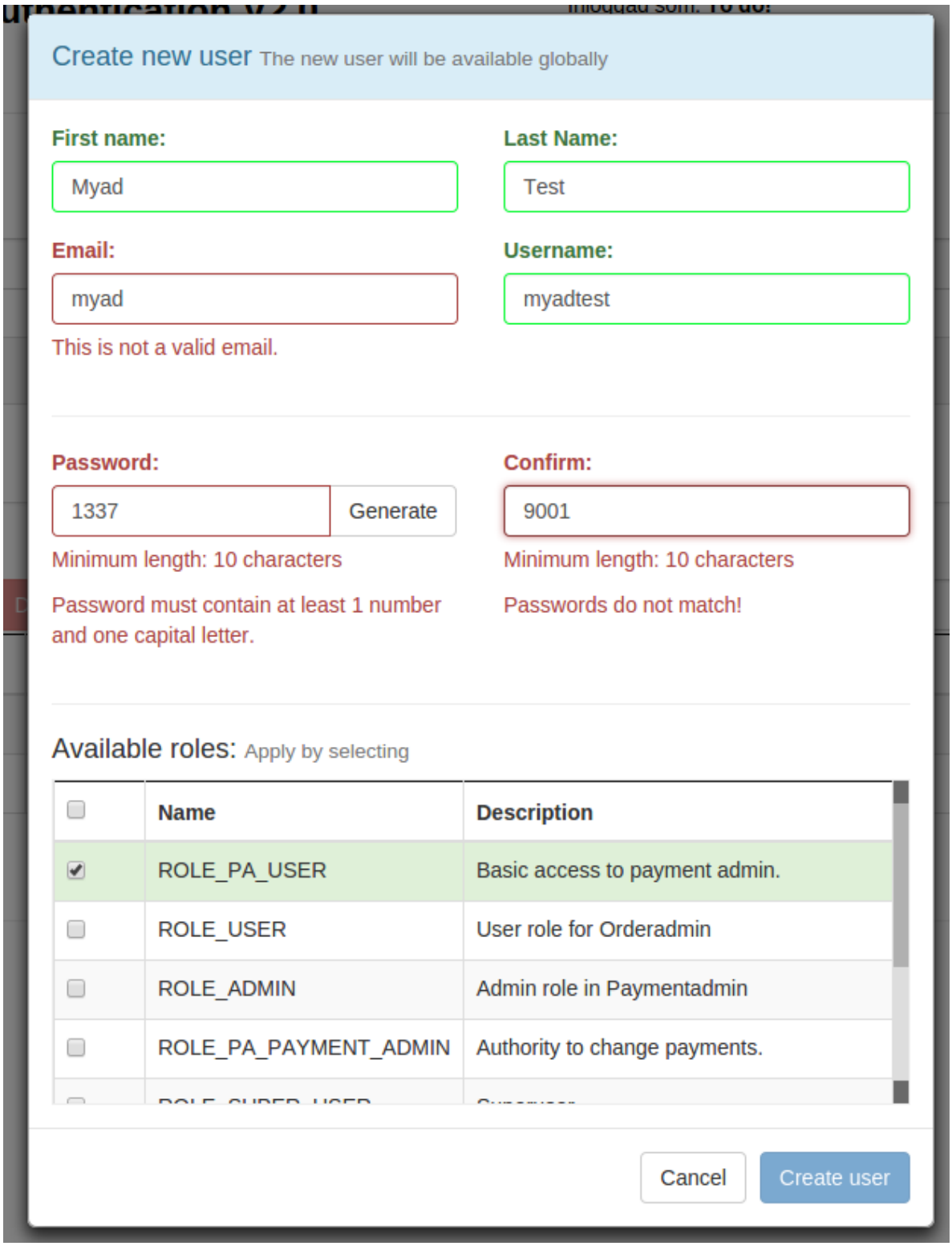

Figure 18: Field validation, error messages, and visual cues in the modals. The background is faded to highlight the active modal window.

The choices available and the live feedback are designed to be intuitive by using Bootstrap standard classes (green for success and red for error). By highlighting input fields on view changes the administrator can start typing immediately, keeping context switches to a minimal.

5.3 Quantified data result

Table 3 and Table 4 show the number of occurrences of the different parameters. They also show general functions for each parameter when adding and deleting users where n is the number of users. The cell in the bottom right shows the general sum of all parameters for the tasks in sections 5.2.2 Legacy application and 5.2.3 AngularJS application respectively.

5.3.1 Legacy application data

Task result:

	Occurrences	Add	Delete	Add + Delete
Context Switches	16	4n+2	2n+2	6n+2
Actions	44	11n+7	5n+5	17n+12
States	23	3n+5	5n+4	8n+9
Inputs	20	8n+1	n+1	9n+2
Navigation choices	5	n+1	3	n+4
Total	108	27n+16	13n+15	49n+29

Table 3: The data measured when performing the tasks from 5.2.2. The variable n represents the number of users.

5.3.2 AngularJS application data

Task result:

	Occurrences	Add	Delete	Add + Delete
Context Switches	4	2n	0	2n
Actions	15	4n	n+1 (alt. 3)	5n+2 (alt. 4n+3)
States	8	2n	2	2n+2
Inputs	17	7n	N	8n
Navigation choices	0	0	0	0
Total	44	15n	2n+3 (alt. n+5)	17n+4 (alt. 16n+5)

Table 4: The data measured when performing the tasks from 5.2.3. The variable n represents the number of users.

5.3.3 Improvement comparison

Chart 1 illustrates the relationship between the number of users and the corresponding total number of occurrences measured for both applications when completing the respective task (see task descriptions in sections 5.2.2 Legacy application and 5.2.3 AngularJS application). The number of occurrences measured contains the total number that at a minimum needs to be performed in order to complete the task.

The *AngularJS alt.* function is a special case where the administrator needs to delete all users from a store. This is done by checking the uppermost checkbox (see Figure 13, Figure 14).

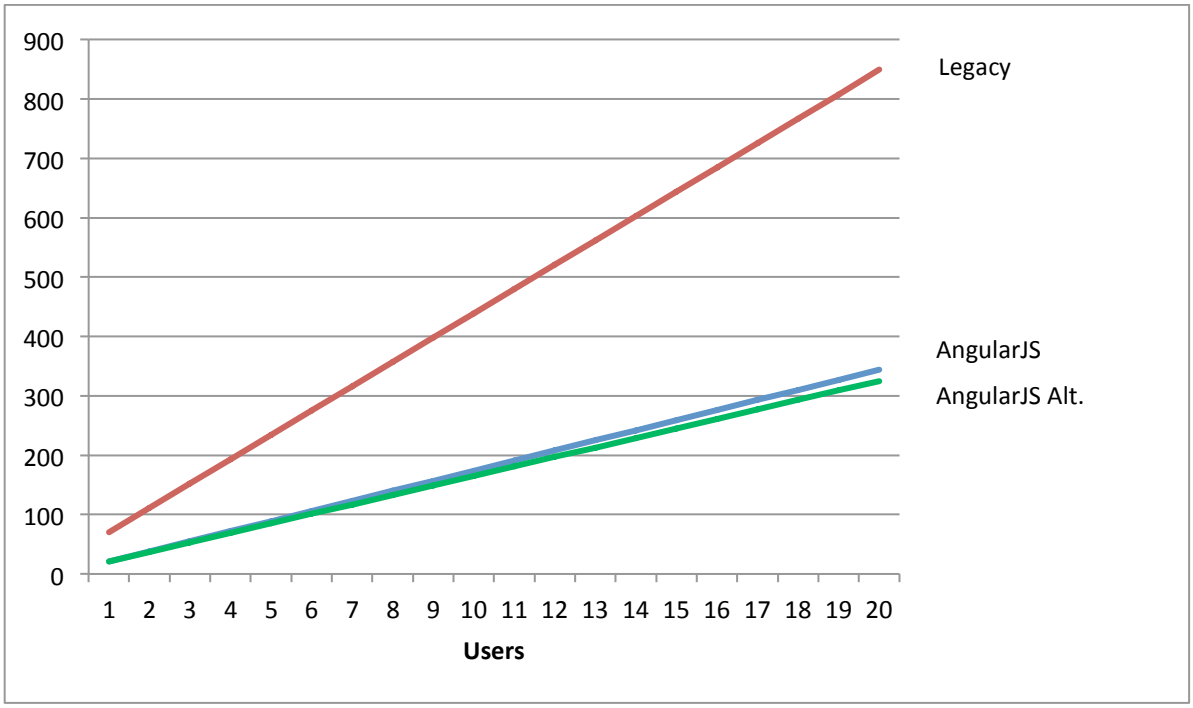

Chart 1: The relationship between the number of users and the total amount of measured occurrences of the parameters listed in the left-hand columns in tables 6 and 7.

Users	Legacy	AngularJS	AngularJS Alt.
1	70	21	21
2	111	38	37
3	152	55	53
4	193	72	69
5	234	89	85
…	…	…	…
10	439	174	165
15	644	259	245
20	849	344	325

Table 5: The quantified task data for 1 – 20 users. A function of the task total from tables 3 and 4.

The relative improvement between the legacy- and AngularJS applications converges to approximately 66 % for a large amount of users. When the amount of users is small (n < 4) the improvement is ~70 % or higher. The data can be found in appendix *8.7 Result analysis data*, Table 9.

$$\left(1 - \frac{AngularJS(n)}{Legacy(n)}\right) * 100 = \left(1 - \frac{17n + 4}{49n + 29}\right) * 100$$

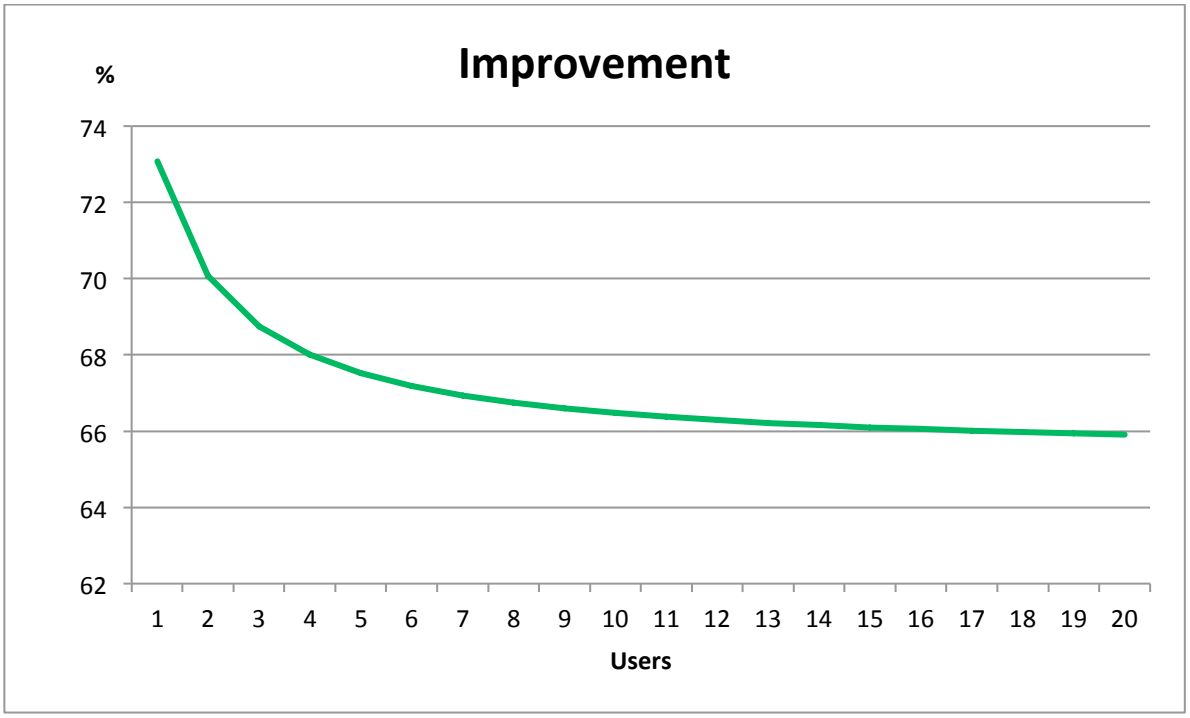

Chart 2: The relative improvement shown for 1 - 20 users.

6 Discussions and conclusions

This thesis aims to answer two questions, as stated in section 1.1 Problem description. Question 1 is answered in section 6.1 Interface improvements and question 2 in section 6.2 Workflow depth.

Question 1: How can visual feedback be implemented to erase the experienced insecurities when using the legacy application?

Question 2: How can the workflow depth be reduced for a given task?

6.1 Interface improvements

Messages that are difficult to understand are shown when submitting faulty information in the legacy application. Oftentimes no response is given at all. First, the user must submit the faulty information regardless of its validity. Second, they must correct the fields to the best of their ability. The *persons* view has many features but the lack of visual feedback when editing a user makes it necessary to double check the changes by reloading the user or navigating to the store. This was done during session 2 (see section 5.1.1 Legacy application – session 2), where the tester was insecure as to whether the changes were made.

The field validation is instant when filling out the input fields in the AngularJS application. This results in only submitting valid information, with the exception of user id and email. These must be validated by the back-end. The difference is that the fields will be marked red with a descriptive text, making the problem clear (see Figure 18).

A person that is colorblind will not get the full extent of the benefits that a standardized color theme provides. This is however compensated by accompanying the color indicators with clear descriptive messages (see Figure 18).

Instead of navigating to different pages/views, pop-up modals were used. This makes completing a task clear, since the modal header contains information about what the administrator is about to do. The modals are also color-coded according to the functionality type.

The background behind the modals was made darker to highlight the active modal. When faced with a critical error message, the administrator is not able to close the modal (i.e. pressing escape, or clicking outside the modal) without using the given buttons. This ensures that no erratic behavior from the application can occur and you will always be sure whether changes were made or dismissed. Green highlighting is used to mark successful changes, giving the user feedback through visual cues.

Search results are only displayed when the input field length exceeds one character without the need to press a dedicated search button or press the enter key. When the condition is unmet, the message: ‘At least **two** characters needed to search’ will show.

The results from test sessions 3 and 4 (see section 5.1.2 AngularJS Application) were clear and showed that the new interface is easy and straightforward. Even without the minor

visual cues added after the test sessions the administrators felt no insecurities when using the new application.

6.2 Workflow depth

The main goal for the AngularJS application was to have a wide and relatively shallow workflow regardless of which view the administrator is in. This is not always the case. Some tasks, e.g. creating a user, require many steps making the workflow deep, regardless of the initial width of the workflow. This is apparent in appendix 8.7, Chart 6 where the improvement is minimal, as the number of input fields required remain almost the same regardless of which application is being used.

One of the major issues with the legacy application was the need to navigate between *persons* and *stores* views when any corresponding update is needed. In Figure 9 it is shown that the only options available when creating a user are adding information such as name, email, and user name. In order to assign roles and/or stores, one must navigate to the user overview page after creating the user resulting in a deeper workflow. The AngularJS application solves the navigation problem by providing a list of users in each store view. It makes navigation obsolete when managing the users in a given store and the use of checkboxes enable the administrator to quickly remove or delete users. Any changes made lead back to the same state/view with the updated/added information, which is shown in Figure 15.

If the administrator needs to manage multiple users in the legacy application, many state-changes and actions are needed. They must navigate back and forth between each respective view, search for each user/store which makes it a tedious, time-consuming process. This is clearly shown in Chart 1 where the legacy application function (49n + 29) is larger initially and scales quicker than the AngularJS application function (17n + 4). Even when managing a few users, the difference is large.

6.3 Future development

In order to match the functionality of the legacy application and eventually replace it, additional functionality has to be implemented. The additional functionality is listed in descending order of relevance, as recommended by the authors of this thesis.

6.3.1 Universal search

Below the header in the AngularJS application there is a search field which currently only searches for stores. This could be transformed into a universal search where stores, roles, and users are searchable. The search results were planned to be grouped by category each containing a maximum of five results. When clicking on an item in the results the application would redirect to the respective overview page (see section 6.3.2 User overview page and 6.3.3 Role overview page).

6.3.2 User overview page

This page could resemble the *create user* modal (see Figure 18) in that it would display user information such as *first name, last name, email, user id* and the current roles the user is assigned to.
The ability to list and manage the stores that has the user assigned to it is also a desired function. It could feature a search field to locate stores to assign to the user. If the user is already assigned to one or more stores these stores will be checked and located next to the search results with a separation to distinguish the two.

6.3.3 Role overview page

From the role overview page the role description could be changed similar to the functionality found in the store overview page. There could be a list containing all the users with the current role but that list could be very long and take a long time to load making the application seem slow. This problem could be solved with a button that toggles the user list with the default state as off.

6.3.4 Send user credentials

When creating or updating a user the (updated) credentials could be sent to either a manager or the user themselves. The administrator could be given a choice to do this by checkboxes, or it could be implemented as an automatic task.

7 References

Cohn, M. (2004). *User Stories Applied: For Agile Software Development.* Redwood City, CA, USA: Addison Wesley Longman Publishing Co., Inc.

Cowan, A. (den 04 02 2014). *Your Best Agile User Story*. Hämtat från http://www.alexandercowan.com/best-agile-user-story/ den 26 03 2015

David, R.-P., & Alan, M. (1995). *Graphical User Interface Design and Evaluation : A practical process.* London: Pearson Education Limited.

Git. (2015). *Git*. Hämtat från https://git-scm.com/ den 25 05 2015

Google Inc. (2015). *AngularJS - Main page*. Hämtat från https://www.angularjs.org/ den 25 05 2015

Ingrid Ottersten, J. B. (2002). *Användbarhet i praktiken.* Studentlitteratur.

Jennifer, P., Yvonne, R., & Helen, S. (2002). *Interaction Design : Beyond Human-Computer Interaction.* New York: John Wiley & Sons, Inc.

Joseph S. Dumas, J. C. (1999). *A Practical Guide to Usability Testing.* Oregon: Intellect Books.

Kniberg, H., & Skarin, M. (2010). Kanban and Scrum - making the most of both. United States of America: C4Media, Publisher of InfoQ.com. Hämtat från InfoQ Enterprise Software Development.

Krug, S. (2006). *Don´t Make Me Think.* Berkeley: New Riders.

Lucid Software Inc. (2014). *Lucidchart - Tour*. Hämtat från https://lucidchart.com den 25 05 2015

Mountain Goat Software. (2015). *User Stories*. Hämtat från Mountain Goat Software: http://www.mountaingoatsoftware.com/agile/user-stories den 26 03 2015

Nielsen, J., & Loranger, H. (2006). *Prioritizing Web Usability.* Berkeley: New Riders.

Resurs Bank AB. (2015). *Om Resurs Bank: Kunder och Historik*. Hämtat från https://www.resursbank.se/om-resurs-bank/kunder-historik/ den 4 April 2015

Silverman, D. (2001). *Interpreting Qualitative Data.* London: SAGE Publications Ltd.

Trello Inc. (2015). *Trello - Main page*. Hämtat från http://www.trello.com den 25 05 2015

Twitter Inc. (2015). *Bootstrap - Main page*. Hämtat från http://www.getbootstrap.com/ den 25 05 2015

8 Appendices

8.1 Project plan

Aktivitet	Varakti	Start	Slut
Project planning	5 dagar	må 15-02-16	fr 15-02-20
Initial analysis phase	**14 dagar**	**ti 15-02-24**	**fr 15-03-13**
Qualitative research	**14 dagar**	**ti 15-02-24**	**fr 15-03-13**
Meeting w/ administrators	1 dag	to 15-02-26	to 15-02-26
Prototyping	5 dagar	to 15-02-26	on 15-03-04
Observation test sessions	**5 dagar**	**må 15-03-02**	**fr 15-03-06**
Legacy	1 dag	må 15-03-02	må 15-03-02
Prototypes	2 dagar	to 15-03-05	fr 15-03-06
Flowcharts	**14 dagar**	**ti 15-02-24**	**fr 15-03-13**
Legacy	1 dag	ti 15-02-24	ti 15-02-24
Prototype	5 dagar	må 15-03-09	fr 15-03-13
Improvements	1 dag	fr 15-03-13	fr 15-03-13
Development phase	**40 dagar**	**må 15-03-16**	**fr 15-05-08**
Prioritizing user stories	**1 dag**	**må 15-03-16**	**må 15-03-16**
Discussions			
Ranking most used functions			
Implementation	**35 dagar**	**ti 15-03-17**	**må 15-05-04**
Set code standards	1 dag	ti 15-03-17	ti 15-03-17
Resources and dependencies	1 dag	ti 15-03-17	ti 15-03-17
Testing	**5 dagar**	**må 15-05-04**	**fr 15-05-08**
Efficiency			
Browser support			
Final analysis phase	**5 dagar**	**må 15-05-11**	**fr 15-05-15**
Results- and workflow analysis	5 dagar	må 15-05-11	fr 15-05-15
First draft	4 dagar	må 15-05-18	to 15-05-21
Second draft	5 dagar	fr 15-05-22	to 15-05-28
Third draft	4 dagar	fr 15-05-29	on 15-06-03
AngularJS application delivery	0 dagar	fr 15-06-05	fr 15-06-05
Final thesis version	6 dagar	to 15-06-04	to 15-06-11
Presentation	1 dag	to 15-06-18	to 15-06-18

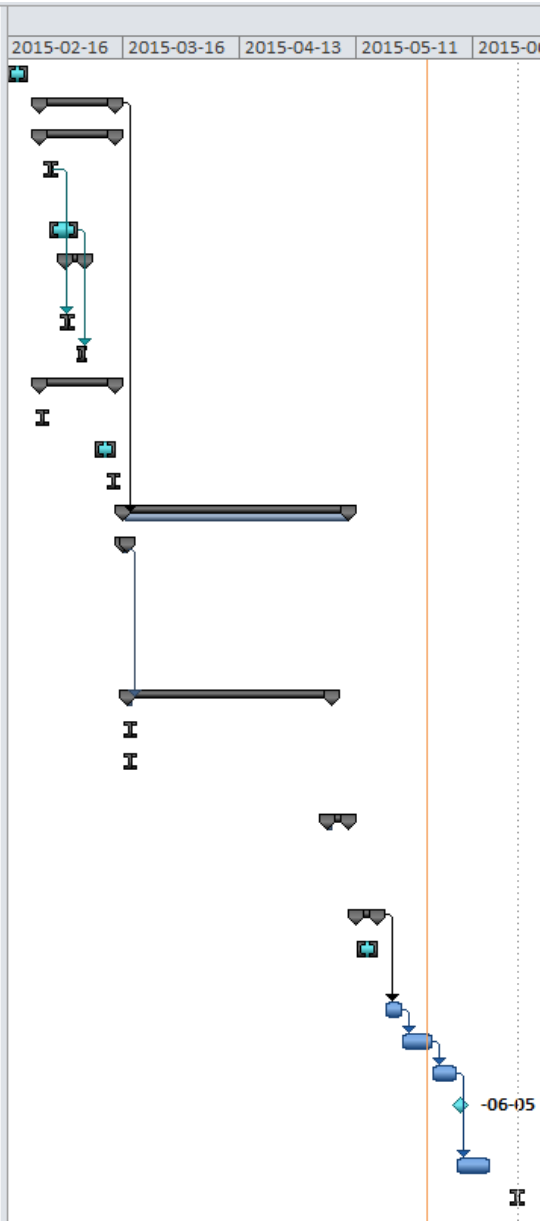

Figure 19: The complete Gantt-chart for this project.

8.2 Meeting with administrators

What are the most common tasks performed with this application? Please provide at least three examples:

1. Search for stores and/or users
2. Update user with assigned stores and/or roles.
3. Create stores and/or users

Which tasks are perceived as unnecessarily complex? Describe the steps needed for each task:
"Searching for stores or users is fairly straightforward, but in order to have the search available, one needs to be either in the *person* or *stores* view and select the country in which the user/store resides."

"In order to assign a user to a store, with the store overview open, one must change from the *stores* view to the *persons* view, find the user, assign the store and go back to *stores* in order to confirm the changes. This must be done for each new user in the store."

"In the 'membership overview' the information regarding the assigned users/stores/roles is incomprehensible. This makes it more difficult to manage and survey when there are many entries."

What extended functionalities are desired?
"A global search field to access all users and stores, without the need to navigate to each respective view, which shows results from all countries."

"The ability to manage users assigned to a store within the 'stores' view."

8.3 Observation test session questions

When starting with the task:
"How do you think you would best finish the task?"
Control questions:
"Where are we now in the application navigation?"
"What does this function do?"
"Please explain what you are thinking right now."
"What are you looking for, and why?"
"How do you feel when…(a given situation)."
"What do you think will happen if..."
After completing (or failing) the task:
"What response were you expecting from the application?"
"Please retell what happened when…"
"What would you change about the application?"
"Did any of the tasks feel unnecessarily complex?"

8.4 Observation test session tasks

Legacy application

1. Preparations:
Logged in using existing credentials.
Choose SE as country.
2. Create a user using the following information:

	Test user
First name	Eric
Last name	Test
User id	EricTest
E-mail address	eric@test.se
Password	*Generated using button*

Table 6: Test user information

3. Create a store using the following information:

	Test store
Name	Test Store LTH
Description	*Erics test butik*

Table 7: Test store information

4. Link the person with the user id *EricTest* to the store *Test Store LTH*.
5. Add the role *ROLE_ADMIN* to the person with the user id *EricTest*.
6. Delete the person with the user id *EricTest*:
7. Delete the store *Test Store LTH*

Test complete.

AngularJS application

The testing of the AngularJS application was done in a different manner. This is due to the added functionality, and some actions (i.e. choosing the country) are not required to achieve the same results.

1. Preparations:
   Logged in using existing credentials
2. Create a user using the following credentials

	Test user 1
First name	Eric
Last name	Test
User id	Eric Test
E-mail address	eric@test.se
Password	*Generated using button*
Store	Test Store LTH
Role	ROLE_ADMIN

Table 8: Test user information

3. Delete the user with the user id *EricTest* from the system.

Test complete.

8.5 Mock-up application

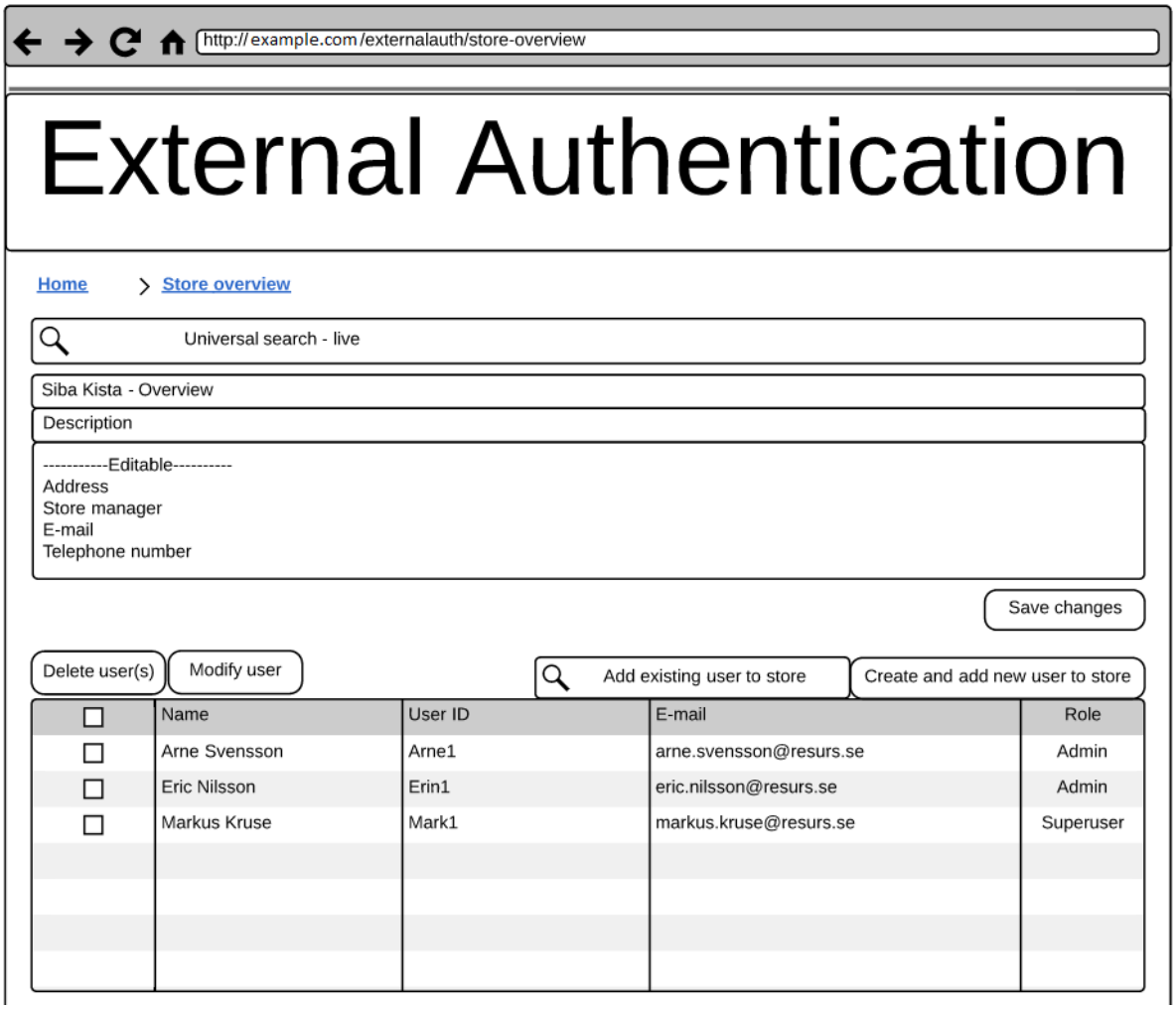

Figure 20: The store overview of the mock-up interface, created in Lucidchart.

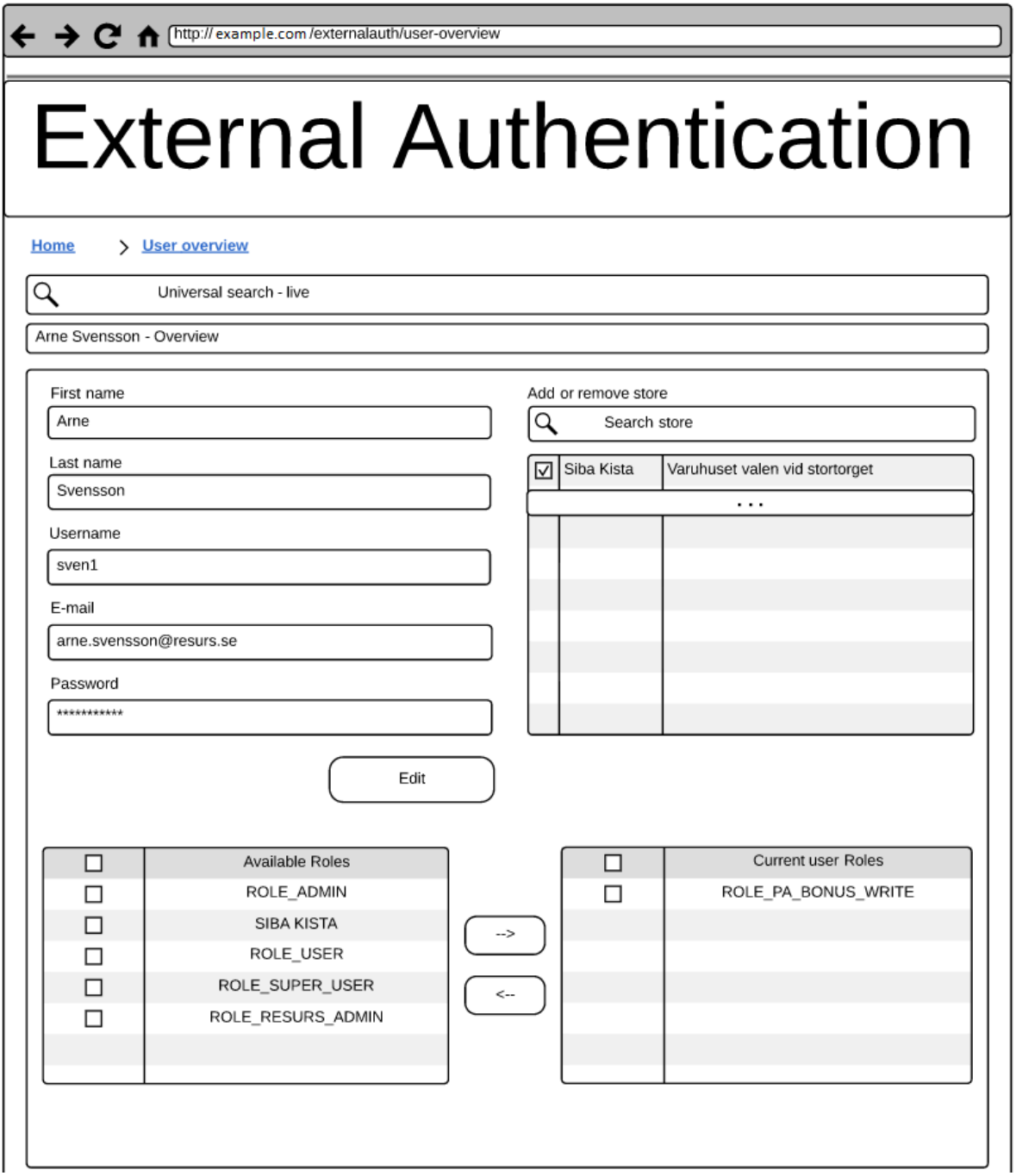

Figure 21: The mock-up for the user overview page. Note the two different ways of adding list items presented to Resurs Bank.

8.6 User stories

1. **As an administrator I want constructive feedback when managing users so that I can confirm the actions I have executed.**
    a. As an administrator I want live feedback regarding the validation on all the input fields when creating a new user and when editing an existing users information.

i. Make sure that a correct input gives feedback, possibly in the color green.
ii. Make sure that a incorrect input gives feedback, possibly in the color red.
iii. Make sure that empty input fields that are required only gives feedback when the page is submitted.

b. As an administrator I want feedback when a user has successfully been created.
   i. Make sure that a message stating something like *'The user has successfully been created and can be located using the universal search'* appears after the user has successfully been created.
   ii. Make sure that the input fields are emptied.

c. As an administrator I want feedback when a user has been removed.
   i. Make sure that a message stating *'The user has successfully been removed. The user has been removed from all the stores it was assigned to.'* appears after the user has successfully been removed.

2. As an administrator I want constructive feedback when managing stores so that I can confirm the actions I have executed.

a. As an administrator I want live feedback regarding the validation on all the input fields when creating a new store and when editing an existing stores information.
   i. Same test cases as 1.a.

b. As an administrator I want feedback when a store has successfully been created.
   i. Make sure that a message stating something like *'The store has successfully been created and can be located using the universal search'* appears after a store has successfully been created.
   ii. Make sure that the input fields are emptied.

c. As an administrator I want feedback when a store has been removed.
   i. Make sure a message stating something like '*The store has successfully been removed. No users are assigned to this store.'* appears after the store has been removed.

3. As an administrator I want to create stores so that I can assign users to them.

a. As an administrator, I want to be notified if the store that I want to add already exists.
   i. Make sure that the page returns a message notifying the user that the store already exists.

b. As an administrator I want to specify the name and the description of the store that I wish to create.
   i. Make sure the name field is between X1 and x2 characters and the description between Y1 and y2 characters.
   ii. The description box needs to contain information before the store can be created.

iii. Make sure that feedback is given in accordance with 2.b.

4. **As an administrator I want to create a user so that I can link it to one or multiple stores.**
    a. As an administrator I want to specify the first name, last name, user id, e-mail, (mobile phone number) and password (alt. See 4.b.) to the user that I want to create.
        i. Make sure that feedback is given in accordance with 1.a.
    b. As an administrator I want to have a button that generates a password so that I don't have to manually create one.
        i. Make sure the length of the generated password is longer than 8 characters.
        ii. Make sure that the password only can consist of (alphanumeric characters).
        iii. The generated password fills the password field and the password is legible.
    c. As an administrator I want to link the user to a store that already exists using the provided search field.
        i. Make sure the search field is using live search.
        ii. The search results are clickable and when a store is clicked the store is added to the list containing added stores.
    d. (As an administrator I want to create new stores and link it to the user I am creating)
    e. As an administrator I want the option to send the users credentials via e-mail or text message when creating the user.
        i. Make sure the option to exclude password is included..
5. **As an administrator I want to login using existing credentials so that I want to access the systems functionality.**
6. **As an administrator I want to use the one and only search function to access all users, stores and roles so that I can manage them.**
    a. As an administrator I want to access the user overview by searching for and selecting the user relevant to my task.
        i. The search string is matched against: First name, last name, user id, e-mail address.
        ii. The search results do not appear until the search string is longer than 2 characters.
    b. As an administrator I want to access the store overview by searching for and selecting the store relevant to my task.
        i. The search string is matched against: Store name, and store description.
        ii. The search results do not appear when the search string is shorter than 2 characters.
    c. As an administrator I want the search results categorized based on type so that I can easier locate what I am after.

i. The types which the search results should be categorized by are: Users and Stores.

7. As an administrator I want to manage a specific store.

~~a. As an administrator I want to change the name of the relevant store.~~

~~i. Make sure that the new name is between X1 and X2 characters.~~

ii. ~~Make sure that feedback is given in accordance with 2.a.~~

b. As an administrator I want to change the description of the relevant store.

i. Make sure that the new description is between 1 and 300 characters.

ii. Make sure that feedback is given in accordance with 2.a.

c. As an administrator I want to create users on the store overview page and automatically assign them to the relevant store so that I don't have to leave the store overview page.

i. The input information consists of the same types as in 4.a.

ii. A password can be generated using the button created for said task.

iii. Make sure that feedback is given in accordance with 1.a.

d. When I create a user on the store overview page, I want to assign roles to the new user.

~~i. Make sure that there is a search field to search for roles.~~

~~ii. Make sure that the search results are selectable and when one is selected it is added to the added roles list.~~

iii. Make sure that there are roles selectable in a table.

~~e. As an administrator I want the option to send the users credentials via e-mail or text message when creating the user.~~

~~i. Make sure the option to exclude password is included.~~

f. As an administrator I want to remove one or more users from the relevant store.

i. Make sure that the users in the list are selectable.

ii. Make sure that the delete button is clickable first after one or more users have been selected.

iii. Make sure that feedback is given in accordance with 1.c.

g. As an administrator I want to change the password of a specific user linked to the relevant store.

i. The password is not legible and cannot be edited. It can however be replaced by another password, manually created or generated by the button from 7.c.ii.

h. As an administrator I want to change the information on the users that is linked to the store so that I don't have to leave the store overview page.

i. After selecting the user the same information can be edited as in 4.a.

i. As an administrator I want to change the role of a user that is linked to the relevant store.

~~i. After the user is selected to be edited a search field for roles should appear.~~

ii. ~~The search results from the search field mentioned above can be selected and assigned to the user.~~
iii. The available role(s) are listed with checkboxes. Selecting them will add them to the user.
iv. The role(s) already assigned to the user can be deselected.

j. ~~As an administrator I want to remove a store from the system database.~~
   i. ~~The users linked to the store, which will be deleted, will remain in the database after the deletion is successful.~~
   ii. ~~Make sure that there is a button that deletes the store displayed on the store overview page.~~
   iii. ~~Make sure that there is a description next to the delete button stating the information given in (i.).~~
   iv. ~~Make sure that a dialogue box is displayed after pressing the delete store button to confirm the deletion.~~
   v. ~~Make sure that feedback is given in accordance with 2.c.~~

8. As an administrator I want to manage a specific user from a user overview page.

a. As an administrator I want to change the same user information as in 4.a.
b. As an administrator, when I change the user id, I want to know if the new user id already is in use.
c. As an administrator, when I change the user e-mail address, I want to know if the new e-mail address already is in use.
d. As an administrator I want to have a button that generates a password so that I don't have to manually create one.
   i. Make sure the length of the generated password is longer than 10 characters.
   ii. The generated password fills the password field and the password is legible.
e. As an administrator I want to link the user to a store that already exists using the provided search field.
   i. Make sure the search field is using live search.
   ii. The search results are clickable and when a store is clicked the store is added to the list containing added stores.
f. As an administrator I want the option to send the users credentials via e-mail or text message when creating the user.
   i. Make sure the option to exclude password is included.
g. As an administrator I want to remove users from the system database.
   i. When the user is deleted the link(s) to any assigned store(s) will be deleted as well and the user will no longer show up in the store overview for the store(s) the user was linked to prior to the deletion.
   ii. Make sure that there is a button that deletes the user displayed on the user overview page.

iii. Make sure that there is a description next to the delete button stating the information given in (i.).
iv. Make sure that a dialogue box is displayed after pressing the delete user button to confirmation the deletion.

8.7 Result analysis data

The charts in this section visualize the functions from tables 6 and 7.

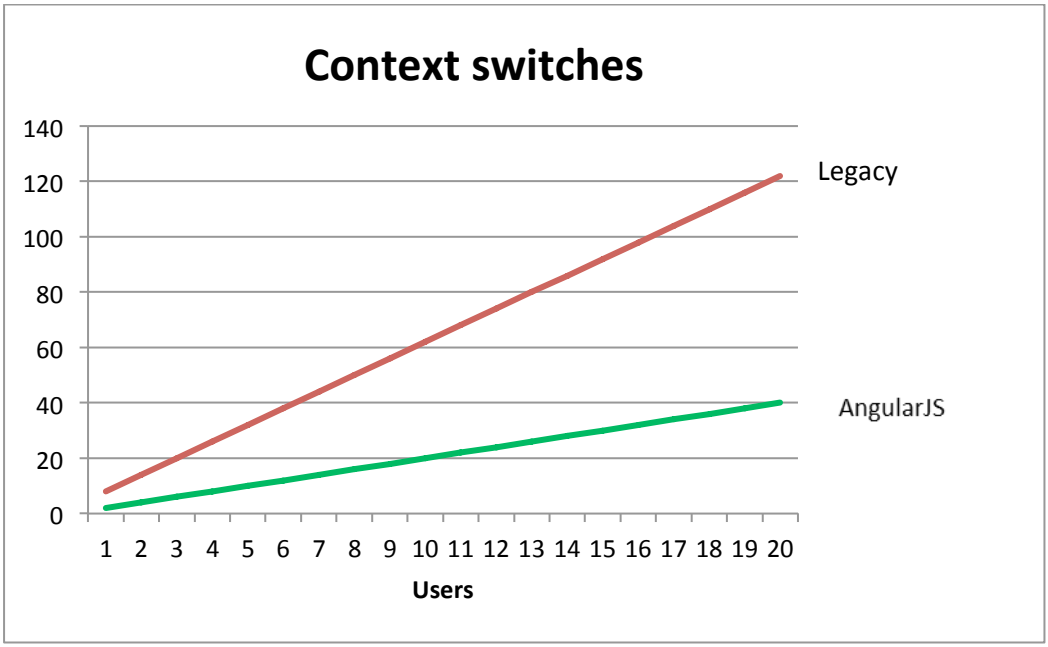

Chart 3: Comparison of the total number of context switches.

Users	Improvement
1	73,1
2	70,1
3	68,8
4	68
5	67,5
…	…
10	66,5
15	66,1
20	65,9

Table 9: The improvement percentage data explained in section 5.3.3, Chart 2.

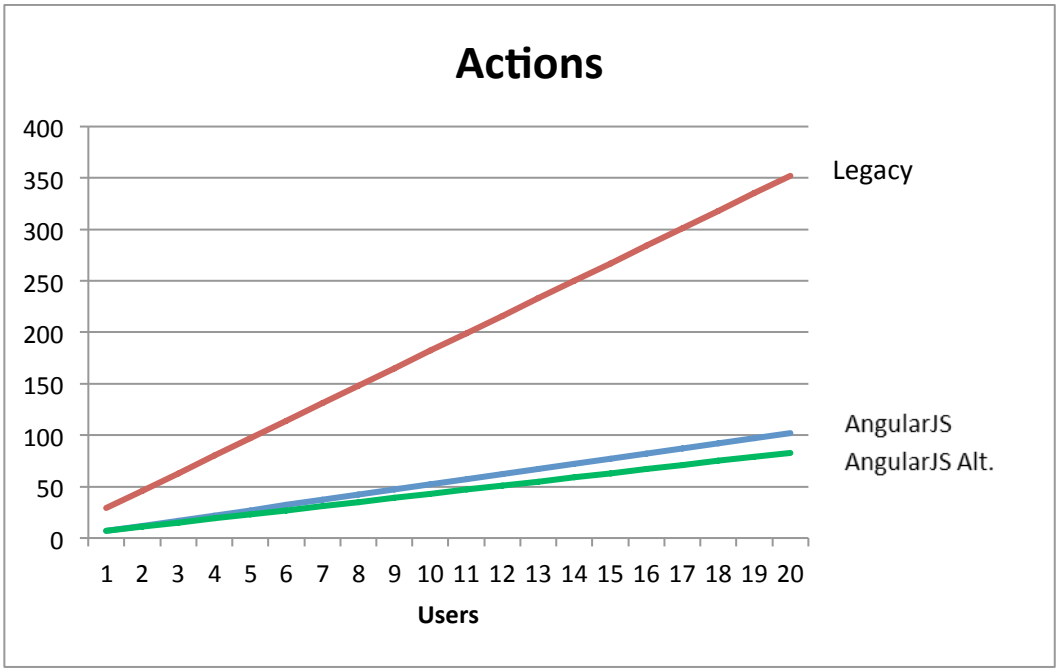

Chart 4: Total number of actions necessary for 1 – 20 users.

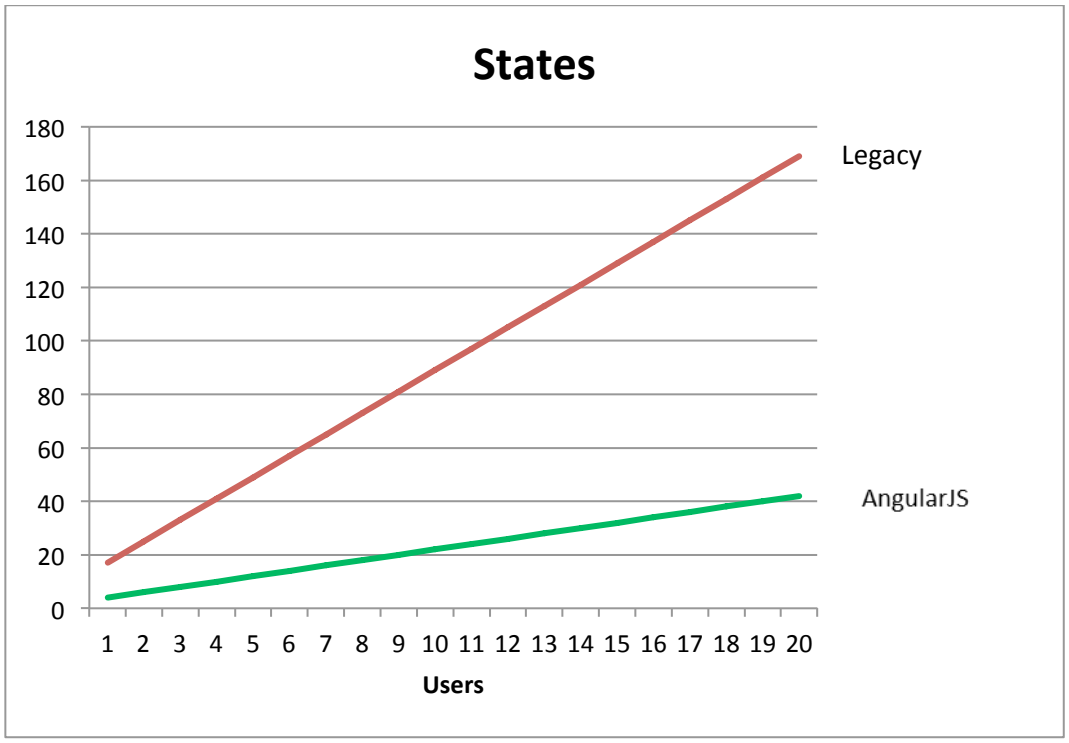

Chart 5: Total number of state-changes for 1 – 20 users.

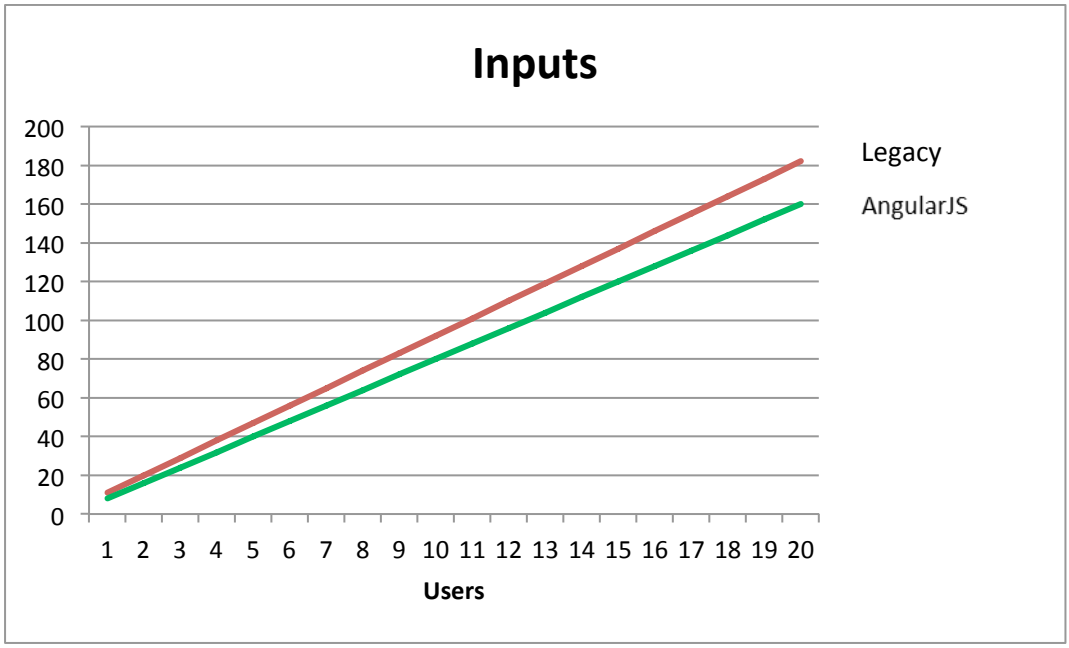

Chart 6: Total number of required inputs for 1 - 20 users.

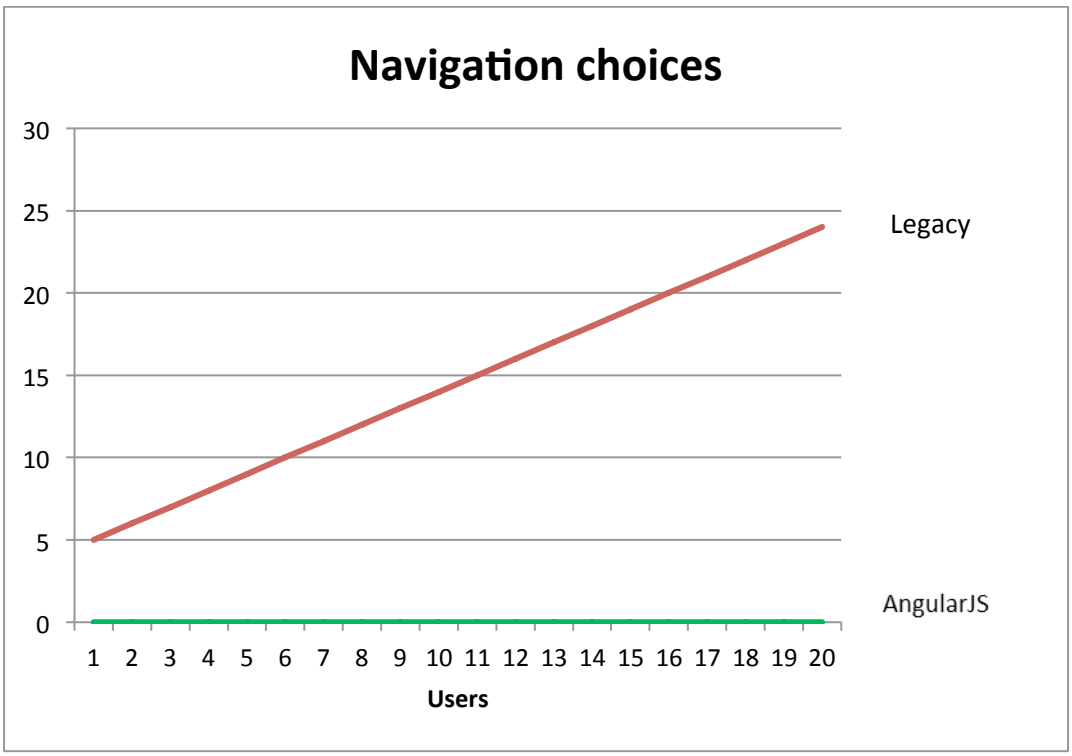

Chart 7: Total number of navigation choices. There is no need for navigation in the AngularJS application.

y