NTNU – Trondheim
Norwegian University of
Science and Technology

Generic HTML5 print-ad editor and PDF-generator

Adresseavisen

November 2014

Group 5:
Knut Sie Andersen
Linda Leidig
Paul Philip Mitchell
Erik Reimer
Nils Inge Rugsveen
Thea Marie Søgaard
Pernille Wangsholm

Adviser: PhD Candidate Gleb Sizov
Course responsible: Professor Jon Atle Gulla
Practical coordinator: Özlem Özgöbek

Abstract

This report was written during the course TDT4290 Customer Driven Project at the Norwegian University of Science and Technology, NTNU. The purpose of the report is to describe the development process and architecture of AdEditor, an advanced single page application for producing advertisements.The assignment was to renew the customer's advertisement system for editing and generating advertisements, as the current system has high server load and a static preview of the advertisement. The scope of our project assignment was restricted to editing and generating death notices, while facilitating possible extensions to include all advertisements the customer offers.

Together with the customer we were able to gather enough information about the problem and requirements to design a production ready application. Through several reviews of the prototype with the customer we converged on the final layout and functionality of the application.

We managed to develop a web application tailored specifically to the customer's needs. The system utilizes the client's computer resources for editing and generating death notices, and the preview is dynamically updated upon changes. In addition, we have designed a web service backend API for the customer to implement, which can be adapted to the customer's whole range of advertisements with few customizations.

List of Tables

List of Figures

1

Introduction

Our customer was Adresseavisen and they wanted us to make an editor for death notices in their paper. Their current editor was not dynamic, and was too slow. They also wanted to be able to export the ad from the editor to a PDF and a picture.

The customer wanted their new editor to be dynamic and run completely on client side. They wanted the editor to show the ad to the user in real time as it was being edited, instead of just when the user sends the ad to the server. They also wanted server communication to transmit less data, so that the application would only be speaking to the server when the user saved or approved the ad. In addition to approving the ad, the user should be able to save the ad, and continue the editing process later. When an ad was approved the application should generate and send the image, PDF, and HTML file representations of the ad to the server. When saving the ad, the application should send a representation of the ad that can be loaded at a later time. Concerning the content of the ad they wanted to have different templates for the user, similar to their current system. These templates included some rules on how the text fields within the ad were structured, and restrictions hat some fields may contain. The customer also wanted to send some information when initializing a new ad to automatically fill in some fields, such as the name of the deceased. The editor should be intuitive and responsive, and also be quite effective, as the users are going to become experts on it relatively fast. The customer explicitly wanted a right-click menu for some of the functionality. They also wanted user guidelines and tools for aiding the user in the editor itself, such as tooltips or popups.

Adresseavisen wanted us to use JavaScript by the conventions of well-structured and documented code. This was our main obstacle since none of us had worked with that technology beforehand, and only some group members knew what “clean code” entailed. To overcome this we had to do a lot of research on JavaScript itself and on which frameworks to use. We were using agile development, Scrum, which was also something most of us were using for the first time in our development careers. This proved quite the challenge when we neither has experience in Scrum, nor knew the process. This lead to the next challenge, which was organizing. With a new

work method and limited time to finish the project, developing an efficient team was crucial. We spent a lot of time at the beginning to figure out how to complete the project with these new aspects. We also needed to balance working on the editor and writing the report in order to get both parts done. Even with these big issues the spirit within the group was always high. We had a real customer with real problems and expectations, which we needed to solve and fulfill. This was a huge drive and we acquired experience on working in a real software development team. We also had the opportunity to see a project come to life, all the way from planning to realizing the finished product.

The rest of the report will be structured as follows: First we will address the aspects of planning and requirements. This part also includes the preliminary study, choice of lifecycle model, tools and technologies, and architecture. Second is the sprint documentation. Here all the sprints will have their own section, examining what we have done in each sprint and their respective results. At the end of the report, there will be an evaluation of the project.

2

Planning

2.1 Project plan

Name: Team Awesome SuperCool
Customer name: Adresseavisen
Partners: None
Background:
This project was given to us so that we could improve the ad-generator for Adresseavisen and replace Adobe's PDF generator server. We were instructed to use the newspaper's death notice as a template for the editor.

2.1.1 Goals

- Make the editor more flexible for the customer
- Reduce the time it takes to create a new ad by 10
- Make the process of adding new templates easier
- Convert to different technologies

2.1.2 General terms

Project duration: 12 weeks
Resources: 7 team members
Planned effort: 350 work hours per team member

Limitations

We made a summary of limitations that might affect our group in terms of workflow and development of the product:

- Unfamiliar technologies

- Little experience with team project work
- Time resources

Tool selections

We tried to carefully choose which tools to use, both for the project and the group organization. Some tools were replaced during the project, such as Trello and Github. Other tools were discovered when necessary, or as improvements to previously used tools. Some tools were also demanded by the customer for us to use. All of the tools listed below will be described in further detail in the tools and technology chapter 6. Tools we used:

- 6.1.1 HTML5
- 6.1.2 JavaScript
- 6.1.2 AngularJS
- 6.1.2 Karma
- 6.1.2 Jasmine
- 6.1.3 CSS3
- 6.1.3 Bootstrap
- 6.2.1 Git
- 6.2.1 Bitbucket
- 6.2.2 JIRA
- 6.2.3 Mailing list
- 6.2.4 Google Calendar
- 6.3.1 Google Drive
- 6.3.2 LaTeX
- 6.3.3 FluidUI
- 6.3.4 Microsoft Visio
- 6.4.1 NetBeans

Organizational requirements

- 4.2 Scrum
- JavaScript
- HTML5

2.2 Schedule of results

Sprint duration: 2 weeks (changed to one week duration later in the project)
Milestones:

- Week 39: Finish 1st sprint
- Week 41: Finish 2nd sprint
- Week 42: 17. October: Deliver preliminary report
- Week 43: Finish 3rd sprint
- Week 45: Finish 4th sprint

- Week 46: Finish 5th sprint
- Week 47:
  - Finish 6th sprint
  - November 20th: Final presentation

We chose to do our project as a scrum project, with sprints of 2 weeks. We were unsure about adding the documentation work to the backlog or to do the documentation separated, and when to stop working in sprints. We originally projected having 4 sprints of 2 weeks, and then stop working in sprints to focus on the final report and presentation. We eventually decided to continue working in sprints during the final weeks of the project due to the excellent project overview offered by our main organizational tool, JIRA 6.2.2. Because of this same reason, we decided to add all tasks, including documentation, to our sprint backlogs. This meant including all tasks regarding the final report and presentation, as well as pre-study, research and so forth. We ended up changing our initial plan to 3 sprints of 2 weeks, and then 3 shorter sprints of 1 week each.

2.3 Project organization

Following is an overview of our project roles and responsibilities. At first we divided all responsibilities strictly into their own roles, but we quickly discovered that our group worked best when we could mix the smaller roles into bigger and more substantial roles. This table is therefor organized by group member instead of a traditional role-only organization.

2.3.1 Project roles

Test Master and Validation: Thea
Choose and set up framework for testing
Make a test plan
Responsible for making sure that tests are made and followed through
Control Markup validation: Make sure the HTML code follows set syntax and grammatical rules

Table 2.1: Test Master and Validation role responsibilities

Release Manager and Git Master: Linda
Create a git cheat sheet so that all team members can look up things quickly
Introduce branching system
Help the team members with problems concerning git or JIRA
Merge the develop branch into the master branch after each sprint

Table 2.2: Release Manager and Git Master role responsibilities

Code Quality Manager and Development Leader: Erik
Responsible of having a overview of the code
Make sure all code is written in the state of clean code
Make sure all group members know how the code works and is able to contribute to the finished product

Table 2.3: Code Quality Manager and Development Leader role responsibilities

Customer Contact and QA Responsible: Pernille
Responsible for all contact between group, customer, and adviser
Call in meetings with Customer and adviser
Will distribute the meeting agenda to relevant parties before each meeting.
Will take the minute reports from all meetings, and send them to relevant parties
Will answer emails within 1 hour unless in another meeting or otherwise occupied
Responsible for making “routines for producing high quality internally” aka dividing teams etc
Responsible for making “routines for approval of phase documents” aka processing papers for delivery to customer and/or adviser
Insure the customer and adviser are up to date with our work

Table 2.4: Customer Contact and QA Responsible role responsibilities

Documentation Supervisor: Nils
Responsible for making the initial outline of the final report
Responsible for creating working documents for the team to work with
Organize and request documents needed in the project
Responsible for organizing the material in the final report

Table 2.5: Documentation Supervisor role responsibilities

Chief Graphical Designer: Paul
Responsible for rough sketches of the UI
Responsible for wireframes
High / low fidelity prototypes
Make sure we conform to specific UI standards

Table 2.6: Chief Graphical Designer role responsibilities

Master of Models: Knut
Responsible for UML
Focused on programming, since the models are not so much used in scrum

Table 2.7: Master of Models role responsibilities

2.3.2 Team members

Name	E-mail	Phone
Knut Sie Andersen	knutsieandersen@gmail.com	950 64 022
Linda Leidig	lindaml@stud.ntnu.no	944 30 340
Paul Philip Mitchell	paulpm@stud.ntnu.no	404 86 106
Erik Reimer	erikrei@stud.ntnu.no	993 35 964
Nils Inge Rugsveen	nilsiru@stud.ntnu.no	958 76 417
Thea Marie Søgaard	theams@stud.ntnu.no	915 56 396
Pernille Wangsholm	pernilwa@stud.ntnu.no	482 99 803

Table 2.8: Our team members and their contact information

2.3.3 Project sponsor

Name	E-mail	Phone
Asle Dragsten Liebech	asle.dragsten.liebech@adresseavisen.no	483 00 525
Hans Kristian Ormberg	hans.kristian.ormberg@adresseavisen.no	959 07 262

Table 2.9: Our project sponsors and their contact information

2.4 Version control procedures

See Git 6.2.1 and Google Drive 6.3.1 sections for information about our version control tools and procedures.

2.5 Test plan

2.6 Quality assurance

In this section we will be summing up the quality assurance for our group. This includes procedures, templates, and internal reports.

2.6.1 Internal organization

We decided early to set up and divide roles internally in our group. This was to make sure the group ran as smoothly as possible from the start, and that we could get started on our task early. Roles were defined and divided. Each member on our group was given one primary role, but we also made sure to have fall-backs in case of unforeseen circumstances. For instance we tended

to work together in the same room, so everyone was always up to date with what the other members of the group were working on. Programming was often done in pairs, so that at least two members were always familiar with the current work. Depending on the size of the role a member had been assigned, that member would also receive tasks delegated by another section in our project, specifically programming or documentation.

We set up the following times as team work hours, and made sure to have a group room booked.

- Mondays 12:00 - 18:00
- Tuesdays 08:00 - 16:00
- Fridays 09:00 - 13:00

This adds up to 18 hours a week where our team worked together. Not all team members could be present for the entire day on Mondays and Fridays, but we made sure to have a daily stand up each workday. Additionally, team members would work from home other days of the week.

2.6.2 Result approval

To assure overall quality in our work, all documents, code, and tests were reviewed by at least two set of eyes. For any code this was done not only through pair programming, but also woven into our version control. Member A would write code, and create a pull request. Member B would be assigned to review that code, and would on acceptance pass this through to the release manager who reviewed all code before merging. In a similar fashion, all documentation was assigned a reviewer, who on acceptance passed it on to the documentation supervisor, who in turn added it to the final report. Before any written material was sent out, be it meeting agendas, minutes, pre-delivery, or the final report itself, it was proof-read and overviewed by the QA responsible. This process was made easier by the group often working in the same room.

Each week both the customer and adviser was kept up to date with the progress of our project, and any feedback was distributed and discussed by the group.

2.6.3 Weekly adviser meeting

We had a weekly adviser meeting every Tuesday at 09:00 in our usual work room. We had agreed on this with our adviser during the first week of the project. Every Monday we would set the agenda for the meeting, and distribute this to our adviser along with any other relevant documents. Recurring documents were the minutes from the last meeting, minutes from other meetings that past week, a weekly status report of our work, and any sprint documents. We also gave our adviser access to our JIRA board, and would send him updates and progress graphs after each sprint. The minutes from each adviser meeting was based on the agenda, which was

again based on a suggested templates for adviser meetings. Minutes included the following information:

- Time and date
- Location
- Primary facilitator
- Time keeper
- Minute taker
- Attendees
- Approval of Agenda
- Comments on provided documents
- Comments on status report
- Planned work
- Actions, Clarifications
- Suggestions from adviser

We rotated meeting roles according to a list, all apart from the minute taker. As the QA responsible was also the customer and adviser contact, it seemed fitting for this team member to also be the minute taker in all meetings. This was a way to make sure all minutes were consistent, to benefit not only us, but also the adviser and customer. It was also important for us that the minutes were a comment-by-comment reflection of our meetings. During the first few weeks of the project we would send the weekly adviser meeting minutes within the same day as the meeting, but after a while our adviser stated that he preferred to get them along with all other documents, e.i the day before the next meeting.

2.6.4 Customer meeting

Our customer meetings were held every other week, on Tuesday 10:00 in our usual room. This was arranged with our customer, and would fit well with sprint progress. Each week, the day before a meeting we would send our customer the meeting agenda, along with relevant documentation. This would often include the weekly status report and sprint progress. We would also keep them updated on the graphical progress of the project. The customer had access to our JIRA project, and could view our progress at any time. Minutes for each meeting was written by the QA responsible and customer contact, as mentioned in the previous section. Minutes would be based on the meeting agenda, which was again based on a suggested template. Minutes included the following information:

- Time and date
- Location
- Primary facilitator
- Time keeper
- Minute taker

- Attendees
- Recap of the previous sprint
- Requested priority for items in next sprint
- Suggestions
- Feedback

Minutes would be reviewed and sent to the customer within two hours of the meeting. As with adviser meetings, it was important to us that the minutes included everything discussed during the meeting. Our customer was a team of two, and due to work both customer representatives could not always be present. It was therefore important that the minutes were thorough.

2.6.5 Sprint retrospective

After each sprint we held a sprint retrospective meeting. This meeting was a way for our group to go through what we had achieved in the sprint, and find lessons we wanted to carry on to the next sprint. It was an ideal way to discover small problems or inhibitors that were preventing us from doing our best work.

2.6.6 Templates

We built templates for some of our internal work. This included:

- Weekly adviser meeting agenda F.1
- Customer meeting agenda F.2
- Retrospective meeting agenda F.3
- Weekly status report F.4
- Textual use case F.5

2.6.7 Internal reports

During the project, our group kept a constant track of the work we were doing in daily scrum standups. During the first week of the project we had a formal work meeting, but we quickly figured that as our group was working together in the same room, it would be a time drain to have a formal meeting each week. Instead we focused on keeping each other in the loop. Our group also logged all working hours each day, and kept track of which field we were working with. Additionally we kept a detailed log in our JIRA board, to track how long each task was.

2.7 Coding conventions

See appendix A for a description of the coding conventions, and the task solution in section 8.2.3 for an explanation of the basis for these conventions.

2.8 Risk management

The following section is dedicated to explain how we handled risk scenarios during our project, and how these risks would be managed. Each risk had a probability scale and a severity scale ranging from one to five. These two scales were multiplied, resulting in an importance scale ranging from one to twenty-five. In addition, all risks had a consequence explaining how the risk would affect us if it occurred. Preventive measures are actions to be followed in order for the risk not to occur, while risk management measures are actions that need to be done if a risk occurs. The risk management document has been modified throughout the entire project to reflect the current experiences we had; probability and severity may have been changed for certain risks, while new risks were added as our experience grew. See appendix B for a full overview of our risk management.

3

Preliminary study

3.1 Problem space

The customer wanted to change their current solution for creating ads to make the editing process more fluid and lessen the load on the servers. Our project was restricted to creating an editor and generator of death notices, so that the funeral agencies can create death notices for the customer's newspaper, both the physical and online version. Our editor should build on the principles of their current solution, with the same rules for how a death notice should look and how the funeral agencies can customize a death notice. To make the editing process more fluid the preview of the death notice should be generated dynamically while the funeral agency edits the information, i.e. the funeral agency can observe the changes in real-time. To make the editing process less intensive on the customer's servers the work should be done by the funeral agencies' own computers. The customer wanted the death notice to be generated as an HTML file with corresponding styling, both for the online version and as a PDF file for the physical version. Additionally, the customer wanted an image of the finished death notice. To make this possible, as well as prepare for future implementations, the customer wanted the advertisement editor developed in JavaScript and HTML5. This left the choice of libraries or frameworks to the project team.

3.2 Existing solution

The solution as it is today lets the funeral agency select a deceased person registered with some basic personal information at Adresseavisen, and the template they wish to use for the death notice. The template is loaded into the editing application, with the available information on the deceased person automatically inserted into the correct text fields. The funeral agency can edit the predefined text fields as well as newly added, empty text fields. The funeral agency can add

and remove text fields and lines, and if the template allows, select a symbol from a predefined set of symbols. The user can also switch positions of text fields, lines and symbols according to rules set by the template. When the funeral agency wants to see a preview of the changes made, they press a button and a request to return a preview is sent to the server. The death notice is generated server-side each time this process is triggered to return a preview, which is a costly and time consuming process. When the user is satisfied with the death notice they can approve the state of the death notice to be put in the newspaper by pushing a button, and the generated death notice is stored at Adresseavisen's servers. The solution is built using a .NET framework and relies on Adobe InDesign to generate the advertisement in PDF [1].

3.3 Our solution

By using JavaScript as our programming language the computations involved in editing and generating the advertisement is done on the user's computer, thus reducing the server load according to the requirements. We chose to use the open-source JavaScript framework AngularJS for developing the application as it can seemingly fulfill all requirements. [2]. AngularJS provides a model-view-controller perspective in JavaScript which easily allows dynamic changes of the view according to the requirement. The advertisement editing process is initialized through an HTTP-post which triggers our program to get field contents and template rules from a JSON file and load them into the ad editor [16]. On the left side of the screen the funeral agency is then able to add or change fields, symbols and lines according to template rules. The user gets an instant preview of the current state of the death notice on the right. To save their progress or approve the death notice they can use the "Save ad"- or "Approve"-button, respectively.

To make the editing process more fluid and the user interface less imposing, we will implement a right-click function to edit font style and size. We believe this will go well with the users, as the main user group will be expert users on this system and prefer commands that saves them time and makes the UI cleaner. Additionally we want to make the graphical design with a conservative PC user in mind (considering the users are employees of funeral agencies), i.e. a simple WYSIWYG interface sized to fit the average computer screen resolutions and formats [36].

Because of the way we implement the loading and saving of advertisements, the customer's IT administrators will be able to make new templates with corresponding rules using the editor. To enable the customer to use the new templates they only need to store the template file on the server and enable the customer to choose the template from the software integrating our system.

3.4 Alternative solutions

We were free to choose framework and libraries of our own preference, and we compared a lot of frameworks to find what would work best for us. The main divide was between using a framework implementing a MVC perspective or not. Among these were the MVC frameworks AngularJS and Ember.js, and not implementing a MVC perspective were Bootstrap [6]. Along the way we considered using JQuery with the different frameworks as well, but in the end we have not utilized it because AngularJS covered all our needs[15]. Other than the technical aspects we considered several different graphical designs for prototype proposals.

Why not the alternative solutions?

We agreed that both Bootstrap and AngularJS would do the job. Because we all were familiar with the concept of MVC before the project started, we wanted a framework with an MVC perspective and therefore chose AngularJS. AngularJS provided a strong backend development framework, and for us it was important to easily and effortlessly manipulate data. For the same reasons we could have used Ember.js, but we were discouraged because of the many recent changes, which pollute online resources with old content and faulty examples. To realise our design prototypes, we used the HTML and CSS framework Bootstrap, as it offers predefined layout options and easily customizable HTML elements.

3.5 Existing solutions on the market

There exists very few similar solutions available to the public. The only existing solutions we could find during market research were provided by the online resources obitNow.com and Rip.ie [25][21].

ObitNow provided a service resembling our solution in many ways. The difference is that the obituary is written ”up-front” by the person concerned. By doing so, surviving family members do not need to write the obituary based only on what they think the deceased would want. Therefore, the actual usage for ObitNow’s service differs from our service by providing a document the surviving family members can use to write the obituary, whereas our solution is for Adresseavisen’s internal team to create a printable newspaper death notice.

Rip.ie provided a service related to our solution. However, their service was only available to funeral agencies. Therefore we did not have the opportunity to try it out.

Our project description also specified that the final, approved obituary was to be converted to a PDF file. We discussed this and concluded that we had to decide between using open-source software, proprietary software, or create our own PDF-generator. Because of the scope of our project and time limits, we decided to simply use open-source software. There are a lot of open-

source PDF-generators available; jsPDF and PDF.js are JavaScript libraries which generate PDF files client-side. There are many online PDF generators where you simply parse a URL and the server generates a PDF for you [17][22]. A client-side JavaScript solution would suit our needs better than online parsers, as we can then then specify what portions of our website needs to be converted.

When it came to market opportunities for our application, there were plenty. Every local newspaper includes obituaries, and their existing solution for writing them may be primitive or insufficient. As our application essentially is a general death notice-editor and not tailored to suit only Adresseavisen's needs, it may be a great asset for other newspapers as well. It is also developed to be easily extended to handle other types of ads such as car ads and real estate ads. For newspapers to be able to make use of the extended functionality, they need an IT department to understand our API and further develop and maintain our application.

4

Choice of lifecycle model

4.1 Waterfall model

The waterfall model is often considered the classic approach to a systems development life cycle. It is one of the most popular development frameworks, especially for large-scale projects with an equally large team. The development phases of the waterfall model are sequential, meaning when one phase is completed, reviewed, and verified, the next phase is initiated immediately. This also implies that the development is linear, meaning the phases need to be completed in a certain, final order[24].

In addition, when you are in any given phase in the project life cycle, there is no turning back – hence the name waterfall model, where water flows in only one direction. Development starts at the requirements stage, and moves through design, implementation, and verification before ending up in the maintenance phase.

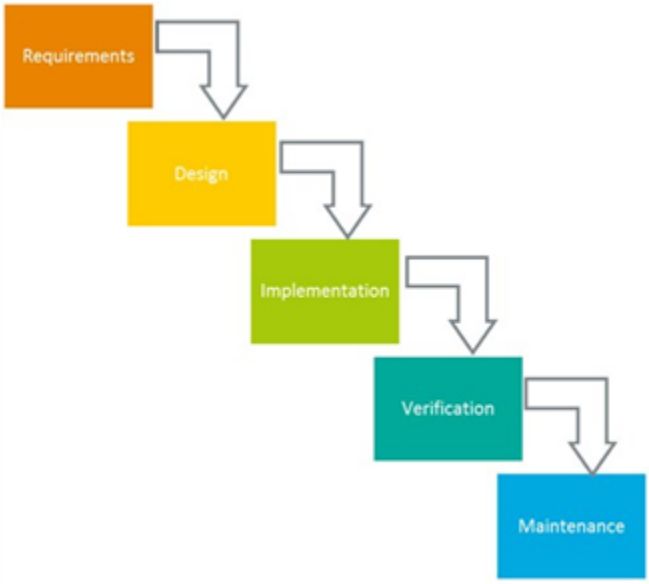


Figure 4.1: Waterfall model. Progress flows from top to bottom, like a cascading waterfall.

4.1.1 Advantages

Waterfall development is a great choice for large-scale teams because it allows greater managerial control over the development life cycle. For each stage of development, a schedule can be clearly defined with strict deadlines. In theory, if these deadlines are followed, the entire project can have a fairly precise deadline.

The model itself will progress linearly through discrete, well defined phases and is thus trivial to understand. Each stage in the development life cycle proceeds in a strict order, without overlapping or iterating.

Common practice when using the waterfall methodology is to invest 20-40% of the project schedule in requirements specification [35]. By investing this amount of time on requirements in the early stages of the project, it follows that every team member has an unambiguous idea of what the completed product will feature.

A final argument for the use of waterfall methodology is that it places emphasis on documentation as well as source code. If new team members are introduced to the project after the requirements phase, it should be easy for them to familiarize themselves only by reading the documentation. In other software development methodologies where documentation has a lower priority, important knowledge may be lost if a team member leaves before the project is complete.

4.1.2 Disadvantages

The major disadvantage of the waterfall methodology is that once the project is in its verification phase, the methodology states that you should not go back into a previous phase[28]. Therefore, if something was not properly defined or well thought out in the requirements or design phase, it would be very difficult to go back and change it.

When using the waterfall model, the team is restricted by when working software can be demonstrated to the customer. Usually, working software cannot be demonstrated until the very late stages of the life cycle.

For long and ongoing projects, where there is a high risk for requirement changes, the waterfall model is not very well suited. Because requirements are only specified in the very beginning of the project, new requirements are difficult to incorporate once the team has moved past the requirements phase.

A lot of time is invested in what some agile development methodologies would call unnecessary documentation.

4.2 Scrum

Scrum is a software development model where interdisciplinary teams develop products or projects in an iterative, incremental manner. Development is structured in cycles called *sprints*. These sprints usually have a length of roughly two weeks, and never more than four weeks. The sprints are *timeboxed*, meaning that they end on a specific date whether the work that was supposed to be finished within that date is completed or not. The sprint time period is never extended. Scrum teams commonly choose just one sprint time period and stick to this for all proceeding sprints. However, If the team significantly improves over the course of the project, they may choose a shorter sprint cycle for later sprints.

At the beginning of each sprint the team chooses certain elements from a prioritized list called the *product backlog*, which is a list of all the requirements from the *product owner*, and puts them in a *sprint backlog*. The team then collaboratively agrees on a goal they can demonstrate to the customer by the end of the sprint. The goal is concrete and should be completely solved - in other words, the goal should not solve the entire project, but rather concretize a single, complete module of the project.

The items in the sprint backlog are broken down into smaller subtasks. The team estimates the time required to finish each subtask in collaboration, and assign tasks to team members.

No new elements from the product backlog should be added to the sprint backlog during the current sprint. The team ideally meets every day for a short *stand up meeting* to inspect their progress and share what they have done since the last meeting. This includes any problems they

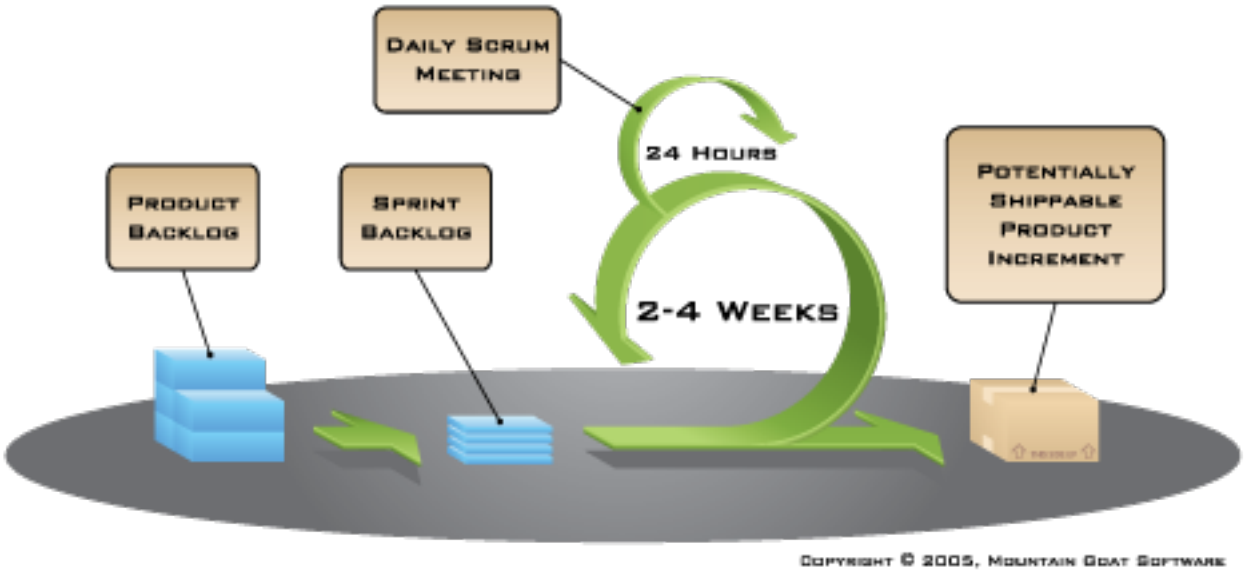


Figure 4.2: The scrum model. Elements from the product backlog are added to the sprint backlog, and then worked on for a sprint lasting 2-4 weeks with daily stand up meetings.

have had and what they are going to do. It also ensures that every team member are familiar with the current state of the project.

At the end of the sprint the team will discuss the sprint with *stakeholders* and demonstrates what has been developed. The stakeholders typically give feedback that can be incorporated by the team in the next sprint. In addition, the team organizes a *sprint retrospective*, where team members reflect on what went well and what did not go well in the last sprint, to improve collaboration and performance in the next sprint.

Role	**Main responsibilities**
Product Owner	Creates and manages the product backlog, a prioritized list of functions or features that are to be included in the product.
Scrum Master	Facilitates productivity by removing any impediments that a team may have to pursue their goal.
Scrum Team	Plans sprints by picking elements from the product backlog into a sprint backlog and develops the system.

4.2.1 Advantages

First and foremost, the scrum development model offers a great deal of flexibility and adaptiveness. As scrum works on such short cycles, changes to the requirements can be quickly incorporated in the next sprint. In addition, regular meetings with the customer ensure that there are no ambiguous details about the project. By providing such characteristics as flexibility, adap-

tiveness, and high customer involvement, the team is able to build only what is needed while still stimulating creativity by building products one piece at a time.

Scrum emphasize sharing of knowledge. Since teams are usually interdisciplinary where each team member specializes in one field, sharing knowledge is important. By doing so, if one team member suddenly can not show up for a meeting or leaves the project, another team member should be able to cover for his or her loss of work. The whole team will ideally have developed professionalism throughout a project, as other areas of expertise are added to their experience.

Projects with initially unclear requirements can still be successful when using scrum. As there is such a high amount of customer involvement, requirements are discussed back and forth throughout the entire project life time. As mentioned earlier, the short sprint cycles allow new or changed requirements to be included in later sprints. Mistakes can also be easily reverted - in contrast to the waterfall model where everything is tested in the testing stage, testing in scrum happens during a sprint. Therefore, if a test does not run successfully, it can be easily fixed during the next sprint.

4.2.2 Disadvantages

One disadvantage of scrum is the high amount of project organizational work. Most of the organizational work can easily be seen as a burden and a time consumer. Some examples include setting up the sprints, daily standup meetings, review meetings, and sprint retrospective. However, most of these scrum-specific artifacts are included in the scrum model to improve efficiency. In addition, scrum is only a development framework and teams are free to choose what elements to include in their own projects.

In our case, it was difficult to match individual student and customer schedules. Because scrum encourages the team to meet very often, we had to spend some time working out when we should meet both internally, with the customer, and with our adviser. Some members still had to leave during work hours and then come back later, which of course was not optimal. Scrum initially states that tasks are to be placed in one and only one sprint. However, tasks can still be incorrectly spread across several sprints due to improper estimation or breakdown. If the team has not invested a sufficient amount of hours into sharing knowledge and one or more team members decide to leave during the project, it can leave a huge impact on the rest of the team.

Another disadvantage we recognized during the project was the need to change data structures due to requirement changes. Scrum emphasize changes in requirements, but that also implies that architecture may need to be heavily modified during later stages in the project. This naturally takes up a lot of time, and may be one of the key aspects where the waterfall method is advantageous. If all the requirements were determined in the planning phase, there would

optimally be no need to modify our data structures or architecture later on.

4.3 Why did we choose Scrum?

We found it appropriate to utilize a process model based on agile development. We had several contributing factors to why we chose to use scrum as our development model. First and foremost, many of our team members were already familiar with the scrum framework from earlier projects. We did not want to spend too much time learning and trying out other process models, as the customer was eager to see progress very early in the project life cycle.

Our release manager and git responsible had particular expertise in scrum from earlier projects in Germany. She had knowledge about the artifacts that scrum involves and had procedures regarding managing sprint planning, retrospective, and such. We have saved a lot of time by using scrum because of her expertise. She served as a scrum mentor, guiding the rest of the team through how to properly use scrum for our project.

The initial requirements were not fully complete when we started the project. Our customer had an application that served the same purpose as our finished product would do, but it was written in another language and they wanted to upgrade it. As such, the customer was not completely clear as to which features and requirements our project should incorporate. By using scrum we were able to discuss and incorporate new or modified requirements as the project went on. This worked perfectly for us as there were some situations where we misinterpreted the customer's requirements and had to modify our application in the next sprint.

Another important contributing factor to why we chose scrum was that scrum allowed us to try out technology in a much better way than the waterfall model would have done. None of our team members were sufficiently familiar with HTML, CSS, JavaScript or AngularJS. It would have been impossible for us to design our entire application before actually implementing it, as the waterfall model states we should. By using scrum, we were allowed to experiment with AngularJS in particular, and we added functionality on the fly while we got a deeper understanding of the AngularJS architecture.

5

Requirement and specifications

5.1 Introduction

5.1.1 Purpose

The purpose of this requirement specification was to understand and clarify the problems that needed to be solved, and get a mutual understanding between our team and the customer about the details and implications of the problem. The reason to do this in writing is that in oral formulation there is often a mismatch between how the client and the supplier perceives the problem, and formulating the requirements to the solution in writing helps to discover any misunderstandings. In addition, both parties will have an informal agreement on what is to be done.

The intended audience for this application are workers at funeral agencies who wants to put death notices in the customers newspaper, as well as the editor and IT-department at our customer.

5.1.2 Scope

Our application ”adEditor” (working name) is an online advertisement editor made for our customer to help their customers create and generate advertisements for both the physical and digital issue of the newspaper, with the design of the ad in accordance with rules our customer set.

As a result of replacing their current solution with adEditor they will reduce both server load and the time their customers spend making the advertisements, thus making it a more viable option for both parties to use. In addition, adEditor introduces a dynamic generator, and the customer can instantly observe changes they make to the ad without requiering third-party software.

The assignment is restricted to developing the advertisement editor for death notices only,

but the application should be easy to extend to include editing and generation all advertisement options the newspaper offers.

5.1.3 Glossary

These words and expressions are recurring in the report. An understanding of these expressions elevates the reader's comprehension of the report, and should be read before proceeding.

- Browser: A computer program that is used to find and look at information on the Internet
- Add-ons: An extra part or device that can be added to something else to improve it
- Third-party (software): Of, relating to, or being software that is created by a vendor to be compatible with the products of another vendor
- Repository: One that contains or stores something nonmaterial
- Merge: To cause to combine, unite, or coalesce
- Merge conflicts: A difference that prevents agreement in merging
- DPI: Dots per inch, i.e. the average number of information points in an image
- Blob: A Blob object represents a file-like object of immutable, raw data
- Backlog: An accumulation of tasks unperformed
- Agile: Having a quick resourceful and adaptable character
- Modifiability: The condition of being able to change some parts while not changing other parts
- Accessibility: The degree to which a product or service is available to as many people as possible, including persons with impairments.
- Usability: The ease of use and learnability of an object
- Extensibility: The implementation takes future growth into consideration
- Getters: Functions used for receiving a specific set of information
- Directive (AngularJS): Behavioural marker for an HTML element
- ng-repeat; ng-bind: Inherent directive in AngularJS
- JSON (JavaScript Object Notation): Abstract file representation of JavaScript objects
- (Code) refactoring: Process of restructuring existing computer code without changing external behaviour
- Template: A document that has the basic format of something and that can be used many different times

5.1.4 Overview

The subsequent sections of the requirement specification consists of the products integration in user interfaces and the surrounding system, the functionality the product offers, a description of the intended user, constraints in regards to hardware, security, interface and reliability, and dependencies of the application. Last are bullet point lists of the specific functional and non-functional requirements.

5.2 Overall description

5.2.1 Product perspective

Software interfaces

The assignment is to make a web application in JavaScript to run in a web browser on the client-side computer system, with our customer only accessing a few controllers to start and save application states. Thus we have not been dependent on the customer to provide software or access to their systems. The customer's intention is to extend our application and in the future replace their current editor with our application as a part of their new system.

System interfaces

The application's interface with the customers system should consist of sending and receiving messages and files over HTTP. To initialize the system for the funeral agency the customer would send a message with information about what template to use and what information the customer has presented before editing this information further. When the funeral agency is finished we are to reply with the death notice represented in HTML (with CSS for styling), image format and as a PDF. We should also be able to load a saved ad for further editing.

User interface

The user interface should consist of a preview and some way of adding and removing text fields, lines and symbols, as well as moving them up and down. In addition there are buttons for "Save', "Cancel" and "Approve", for saving, canceling or approving the current state of the ad. The buttons are restricted by rules given by the template or given implicitly by the system, which prevents the user from making mistakes.

Site adaption requirements

As most of the changes we make from the current system is purely functional, we predict the customer will not need to spend much time adapting to the new software. As far as is practical we will try to conserve the work flow of the customer's current application, but removing the need for updating the preview manually.

5.2.2 Product functions

- Add
  - One-column text fields
  - Two-column text fields
  - Lines
  - Symbols
- Remove

  - Text field
  - Lines
  - Symbols
- Change
  - Order
  - Font style
  - Font size
- Cancel progress
- Save progress
- Approve death notice

To assist the user the application shall not allow the user to perform actions which leads to violating any rules set by the template. An overview over the rules can be found in table 8.3

Variables

The live variables are going to be stored in a local array in a service implemented through AngularJS, in which each element represents an element in the adEditor, such as a text field or a line. This array is passed as an argument to the functions manipulating or reading the array. By keeping the data in an array we should be able to easily keep track of and manipulate the order of the fields, and efficiently iterate through the data to create the preview.

5.2.3 User characteristics

The main user demographic of the application is the employees at funeral agencies, who will be using the application for generating approved death notices for the newspaper. Our two other user groups are the newspaper editor, who might need to change the ad to ensure correct spelling and styling for the newspaper, and the IT-department who are going to be the administrator of the application.

The employees of the funeral agencies are not going to use the application as often as the editor of the newspaper and do not have as much experience with computers as the IT-department, and thus should be our main concern when designing the application. We are going to preserve as much as possible of the current work flow in the graphical user interface of our application. We assume the immediate users of the software have experience with the current solution, and will have a low entry threshold.

We are going to make the assumption that many of the employees at funeral agencies are middle-aged and older with low technical expertise. Because of this we are going to implement a "what you see is what you get" editor, and focus on implementing good support for standard computer web browsers, ignoring tablets and smart phones unless we have sufficient time.

5.2.4 Constraints

Interfaces to other applications

The customer's main concern is the internal functionality, and they want a proof of concept and a prototype to build their own application upon. They have specified some requirements as to how our application is going to interact with their current system, as show in the figure below:

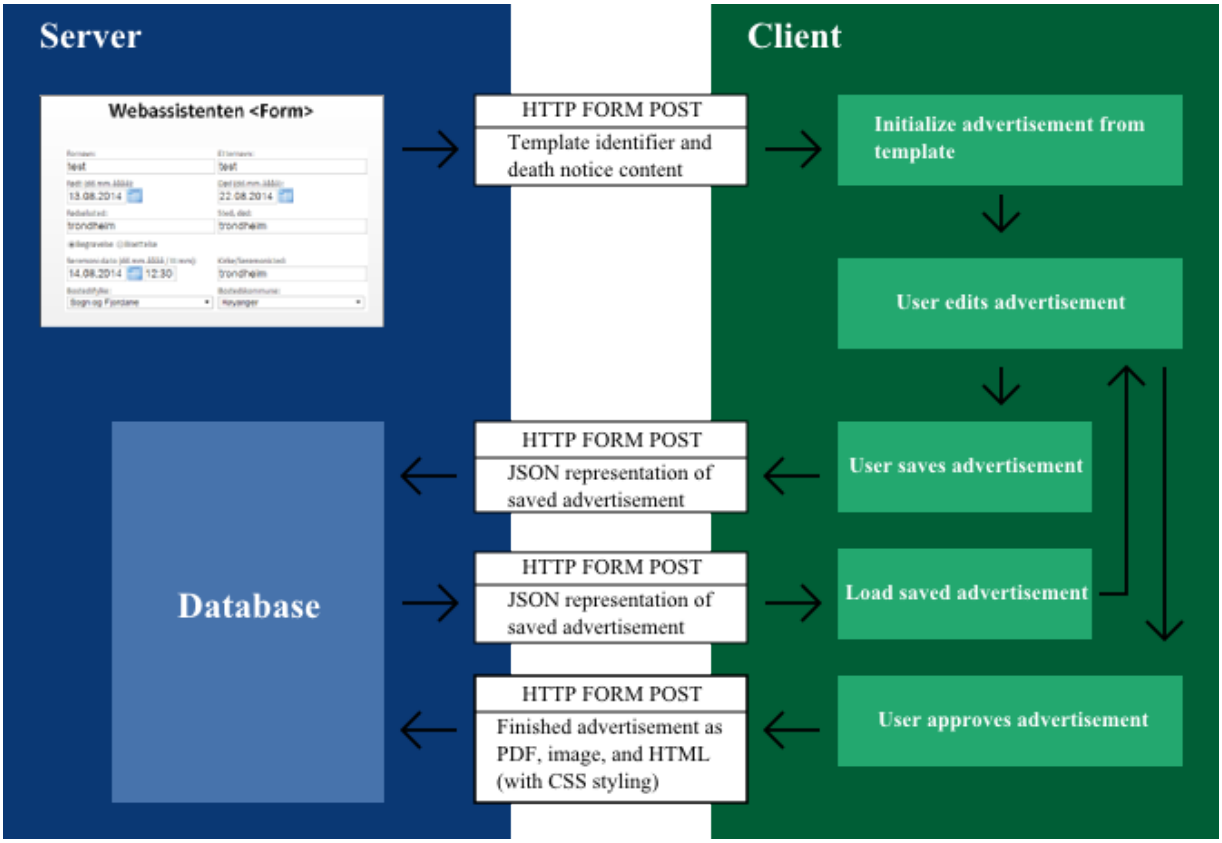


Figure 5.1: Communication interface requirements: How the client side application is going to interact with the customer's server

Hardware limitations

Because the solution relies on the clients computer to run, the concerns about performance are moved from our customers servers to our clients hardware and, to some degree, their internet connection. The application itself will not consist of data-heavy elements like images, and the text documents and scripts should be smaller than most websites and thus not a problem for the user. Our main concern regarding performance is minimizing memory and CPU usage at the client's computer by efficiently implementing data manipulation algorithms and data structures. In order to maintain instant feedback we need to avoid expensive operations.

Reliability requirements

The reliability of our software depends on the reliability of the technologies we use. As these are well-supported and widely used this should be no problem. However, we need to make sure the information in the application is preserved between different states of the application.

Safety and security considerations

The code is open source and the customer has no concerns about us working on it in a public space. The information the user submits is transferred using HTTP, which leaves the possibility for SQL injections to the server; preventing this is not part of our requirement. The local application data on the funeral agencies' computers could only be compromised if a hostile source broke into their computer, which cannot be prevented with our application. No security measures against manipulation of the code or it's internal data has been made, as this was not a part of the requirements. The reason for this is that the newspaper editor manually reviews all advertisement before they are released.

5.2.5 Assumptions and dependencies

Browser support

The technology we use (AngularJS and Bootstrap CSS) support these browser versions and newer, and everyone running any of them will be able to use our application without additional software.

- Internet Explorer 9
- Mozilla Firefox 4
- Opera 5
- Safari 11
- Chrome 30

5.3 List of functional requirements

- The editor has to take in a template that generates a default ad.
- The editor should be able to add: new text fields, and some approved images and lines.
- The editor should be able to change font and size
- The editing process has to be dynamic and client-side
- Possible to save ad for later approval
- Ad to be presented in HTML5
- Intuitive UI for customers
- General methods: Should be possible to make future customization/extension and easy to implement in existing software
- Generate PDF from HTML5 ad
- Convert HTML5 to image (300DPI, CMYK color profile) and image to PDF on approval

- Date stamps on generation, changes and approval
- Access restrictions
- Rules (we get this from Adresseavisen) restricting how the ad can be customized. This should be implemented in the template

5.4 List of non-functional requirements

- The layout has to look the same across all browsers
- Compatible with all browser versions listed in 5.2.5
- Documentation for the code
- Instant (or near instant) response time
- Modifiability
- Extensibility for other types of advertisements
- Effective to use
- Use few resources

5.4.1 Optionally

- If we have the time: Make a user guide for the funeral agency and producer
  - A few screenshots with a workflow description

6

Tools and technologies

Figure 6.1: Overview over our tools and their conceptual relationship

6.1 Languages and frameworks

6.1.1 HTML5

HTML5 is the latest form of HTML and introduces some new functionality that makes it easier to show web applications cross platform. HTML5 is a combination of XHTML and HTML in order to define a single markup language that can be written in either HTML or XHTML syntax. Some of the new features that were added to HTML5 were introduced to better allow handling of multimedia and graphical content.

Using this framework was not a hard decision given that it was a part of the requirement given to us by the customer. The request from the customer included generating an ad with corresponding HTML5 code. This led to our decision to make the whole ad editor in HTML5/JavaScript.

6.1.2 JavaScript

Given that we were supposed to make a web application with HTML as our base, the only real choice for us was to use JavaScript. JavaScript is widely used and contains the functionality we need. Since we only had to work on the client side of the process, we found that the framework AngularJS would suit our needs quite well.

AngularJS

AngularJS is a framework made by Google for developing client side applications. It makes use of the MVC pattern in an easy organized way, making the code more manageable. Even though it is a framework, it is still plain JavaScript, avoiding having to inherit properties from other sources. AngularJS introduces data bindings to the HTML which makes the web page update according to changes done to the data. Our customer wanted an editor that could change dynamically when different parts of it was updated, and this was one of the deciding factors for using AngularJS. AngularJS also include custom directives that give the user the ability to make their own HTML tags that can be reused and increases the readability of the code. So in addition, AngularJS is designed with testing in mind taking fully advantage of dependency injection.

Karma and Jasmine

As none of our group members had any experience with JavaScript, including testing in JavaScript, we had to do some research to learn about how to test our product. Read into what kind of testing frameworks are recommended, while also taking other JavaScript frameworks we are using into consideration. We looked into testing with the framework AngularJS, and figured that the AngularJS team of developers (Google) have also made a couple of test frameworks called Karma

and Protractor to work with Angular [29]. It is not recommended for us to use these together, as they provide different systems to run tests [30].

First and foremost we were looking at unit testing for our product, as the customer hhadas asked us to write only for the clients side of the application. They will implement the server side themselves, making the end-to-end testing excessive. Of Karma and Protractor, Karma is the one that fits the best for writing unit tests, while Protractor is mainly a tool for doing end-to-end testing. As we were only writing unit tests, Karma was the natural choice. While Karma is a framework for running tests on a local server using node.js with different config possibilities, we still need a framework to actually write the tests with. There are a lot of tutorials online on how to set up unit testing for AngularJS using Karma and Jasmine, so we decided to look closer into Jasmine as well [31]. As we were using NetBeans as our IDE, we looked at the compatibility between NetBeans and Jasmine [19]. After all of this research we decided to go for the Karma and Jasmine frameworks.

The setup process for these frameworks consisted of combining different tutorials to get it to support our project in a good way [4]. Node.js is a required part of Karma that every group member needed to install. We also chose to include the invisible browser PhantomJS, to avoid a browser pop-up every time we ran a test. The setup process was a bit harder than anticipated and also different for Mac and Windows.

Libraries

Html2Canvas

This library allows the user to take a “screenshot” of any part of HTML code. The library makes the canvas based on the information available in the HTML code and therefore may not be 100% accurate. The library is supported by all the main browsers, at least if the latest versions are used.

The reason behind using this library was to simplify the task of generating the image that is requested from the customer. It also helps generating the PDF and HTML code as we use the canvas generated by this library.

jsPDF

This library gives the user the ability to add HTML tags, text, and images to a PDF blob. So this library basically makes it possible to take any canvas. The library supports all the major browsers.

The problem that this library solved was generating the PDF that should replace the Adobe InDesign servers the customer use.

angular-drag-and-drop-lists

As normal AngularJS does not implement a way for drag and drop, an external library was necessary [7]. This library implements the drag and drop using the native HTML5 drag and drop

API. Therefore it is not usable on touch devices. It supports all modern browsers, but does not work for Internet Explorer 8 or lower.

The list of components in the editor view is generated by a ng-repeat which is an AngularJS element. We used this library to make the list of components draggable.

6.1.3 CSS3

Cascading Style Sheets (CSS) is a language used to describe the look and feel of an HTML/XHTML document. The primary intent is to separate the document content from the document presentation, such as layout, colors and fonts. CSS enables the author of a website to allow multiple HTML documents to share the same formatting through external style sheets, reducing the complexity and repetition of styles in each and every HTML document. CSS3 is the third iteration of the declarative stylesheet language.

CSS3 is a natural extension of HTML5, allowing the HTML documents to be styled with fonts, colors and layouts. Because our project description was to develop an HTML5 website, we had to use CSS3 in order to achieve the desired design.

Bootstrap

Bootstrap is an open-source front-end-framework developed by Twitter. It contains a combination of HTML5, CSS3 and JavaScript code designed to help build user interface components such as typography, forms, buttons and navigation elements. Since version 2.0 it supports responsive web design, meaning the layout of web pages adjusts dynamically based on the device used.

Why we used Bootstrap

Bootstrap offers a grid layout system that dynamically adjusts its size based on the resolution of the device the web page is viewed on. Because our application is intended for desktop use only, we did not take full advantage of the grid system, but we still implemented it in order to handle resolutions down to and including 1024x768. Button design is well covered when using Bootstrap; all you have to do as a web page author is to include the appropriate class to the associated button, and Bootstrap handles the rest through its pre-defined style sheets. This saved us a lot of time while still being modern and functional. In addition, we wanted to include easily recognizable icons so that the user maps actions to consequences simply by observing the icon. Bootstrap included an extensive list of icons that could be easily implemented.

6.2 Organizational tools

6.2.1 Git

Git is a distributed revision control system that distinguishes between remote and local repositories. It enables different people to work on the same code locally, while merging it automatically on the remote version. Only if it is not able to do this on its own, conflicts occur that have to be solved manually. For different features, different branches can be used so that the develop branch is always clean. Git keeps a log of changes and the code can be reverted to previous versions.

Git was an easy choice since most of us had experience working with it. There are different applications that do the same, but we did not consider them since Git is well known and documented. In addition, all team members had experience using it.

We built our branches in a way that helped us develop different functionalities at the same time. We had 3 layers of branches. On the top we had a master branch which was updated at the end of each sprint. On the next layer we had the develop branch that was updated during the development process. The third layer comprises many feature branches that are branched from the develop branch. Each feature branch covers a different functionality of the program and is merged back into the develop branch after the implementation of the feature is completed.

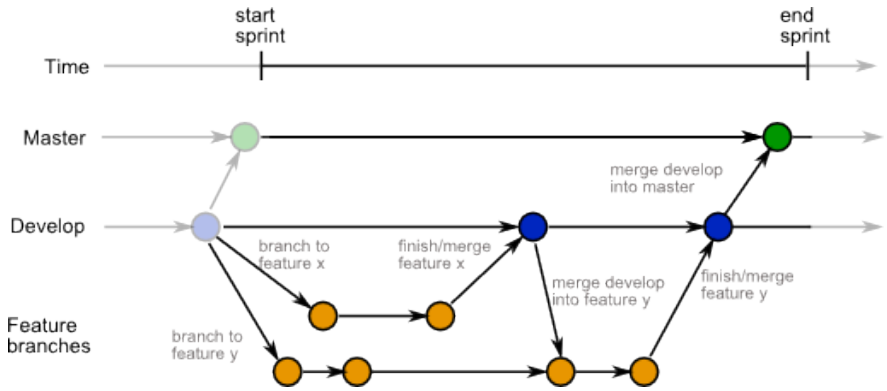


Figure 6.2: Git branch overview and conceptual illustration of our work process

Bitbucket

Bitbucket is a web-based hosting service that use Git. Bitbucket is integrated with Jira so that it is possible to create a branch directly from the Jira task. This gives an overview of branches, commits, and pull requests.

Using Bitbucket was very natural when we decided to work with Jira as it enabled linking tasks to branches. The possibility to create pull requests ensured that the code was reviewed before being added to the develop branch. Bitbucket gave a very good overview of the repository

of the project.

6.2.2 JIRA

JIRA is a licensed project organizing and issue tracking tool by Atlassian. It works together with Bitbucket and Agile, which are both also developed by Atlassian. In JIRA a project can be created, and will store issues assigned to users. It will organize them in either the product or sprint backlog if the project is an agile project. JIRA provides a well structured overview of the state and progress of the entire project by showing reports for each sprint including a burndown chart and a list of all tasks in the sprint divided into columns that show the progress of each task.

We found that this tool helps us immensely as we get a good overview while we work and an automatic generated burndown chart. It is very handy for delegating work and for planning each sprint. For each sprint we would take the tasks we were going to do based on user stories, and split them down into smaller subtasks with comments and estimated workload. We then assigned these and start working on the sprint. JIRA can export the sprint backlogs into excel sheets, which saved us a lot of work when creating the report.

We chose JIRA because of the generation of documents, charts, and the overview it gave us while working on the sprints. JIRA generated different IDs for all issues and gave us the opportunity to log work on each issue and then combine it in a burndown chart. It is not free, but the monthly fee was very small considering how much we used it. The customer was more than willing to sponsor us with this small fee.

6.2.3 Mailing list

Most communication between the customers, the adviser, and our group took place via e-mail. Therefore, we created a mailing list provided by NTNU.

This was very helpful to ensure the awareness of the team concerning the communication with both the customers and the adviser as only one team member was responsible for this contact.

6.2.4 Google Calendar

Google Calendar is another web tool published by Google to support group collaboration [8]. It permits creating calendars that can be shared and integrated into individual local calendars. Using Google Calendar to get an overview of scheduled meetings came as a natural extension to the other services from Google that we used, such as Google Drive. Since we had quite a lot of meetings, it was very practical that only one team member had to manage the calendar.

6.3 Documentation tools

6.3.1 Google Drive

Google Drive is a file storage and synchronization service that offer cloud storage, file sharing, and collaboration on documents [9]. The core functionality is accessed using a web browser, and is supported by all devices running a modern web browser. Google Drive implement tools for creating and editing known formats of text documents, spreadsheets and presentation slides. See [10] for a full list of native formats and additional information. Collaborators can see who are working on the same document as you, immediately observe their changes to the document, and view the history of all changes made. Google offers both free and premium plans, the difference being the storage capacity. In addition to the web browser application, the client can install a synchronization software enabling synchronization of files between the client's computer and Google Drive. In addition to providing a small selection of tools, Google Drive support installation of third-party applications from the Chrome Web Store [3]. With this users can extend the usability of the service by installing additional community software, which also gives rise to support of additional file formats not natively supported by Google Drive.

How we utilize Google Drive

Our project group have used Google Drive to a great extent in our work in document collaboration and file sharing. All working documents, except for the code, are shared with the group on Drive. This way we are able to read and edit each-others documents in real time. This saves us time, provides backup of our project files, and provides transparency of each member's work progress. Additionally, we are not reliant on all project members installing specific software to contribute to the documentation. Documents created with other software can also be imported to Google Drive. With the extensive storage space in the free plan we have not had the need to buy a premium plan, which was important for us when choosing a collaboration platform.

Alternatives

There are several other services similar to Google Drive, such as Microsoft OneDrive which also offer cloud synchronization of files and collaboration on documents through a web browser [18]. The reason for choosing Google Drive instead of OneDrive is our previous positive experiences with Google Drive, and the fact that all of us had accounts with Google. We also briefly considered using Dropbox, but Google Drive satisfied our needs better [5]. The main impediment of Dropbox is the reliance on third-party software for editing documents. This would make version control and simultaneous collaboration on documents difficult.

6.3.2 Latex

LaTeX is a document preparation system and markup language. It is popularly used in academia where advanced text formatting is required, such as in mathematics, statistics and computer science. It is written as abstractions of the TeX macro language, and uses the TeX typesetting program for formatting its output. Libraries provide collections of tools and labels developed for writing articles, books, presentation slides and more. One of the main differences compared to modern text editing tools is the separation of content and presentation. In LaTeX the content is written as plain text and formatted using labels. To view the text in a more presentable format, e.g. PDF, the text needs to be compiled, which can be done by one of the multiple available TeX distributions [34]. LaTeX is free software, and despite its age (initially released in 1984) it is still maintained. One of the goals of LaTeX and the TeX macro language is to provide longevity to the users, so that source files would produce the same output now as when LaTeX was released. LaTeX compilers are available on most operating systems, and the source files can be edited using any text editor.

How we utilize LaTeX

Our group use LaTeX for creating the project report, and the editing is done by the Documentation Supervisor. The working documents which all team members participate in resides in the shared folder in Google Drive (see 6.3.1 for more information), and this content is transferred to LaTeX at regular intervals. For editing and compiling our LaTeX project we use an online tool called ShareLaTeX [26]. This service provides an editor, file storage, a compiler and libraries for LaTeX, all residing on the servers without the need for third-party application. Private content is accessed by logging in through an internet browser, and can be created in the browser, linked from Dropbox, or uploaded from the computer. In addition to relieving our group of concerns with libraries and compilation, they provide templates and auto complete suggestions for labels. ShareLaTeX provide collaboration on projects, much like on Google Drive, although this requires premium access if more than two people wants to collaborate on a project. This was useful to our group, as it enabled the Quality Assurance responsible to read through and edit our report before final delivery.

Alternatives

We are required to submit our project report as PDF, and though there are many alternatives for creating PDFs on the market, most are expensive suites. Although Google Drive exports documents to PDF for free, we preferred LaTeX because of its consistency in the presentation of the content and the fluid editing process. Our choice was supported by our previous experience with LaTeX, the large community support, and free access to the tools.

6.3.3 FluidUI

FluidUI is an HTML5 user interface prototyping tool that allows UI designers to create horizontal or vertical interactive prototypes. It works by arranging pre-designed elements into a WYSIWIG (What You See Is What You Get) editor. It is normally used during the requirements stage of software development by enabling design-iterations and customer collaboration. Because it is an online tool, clients can be invited to a prototype project and write comments on a prototype and even propose modifications.

We used FluidUI to translate our initial paper prototype into an interactive prototype in order to receive accurate feedback from the customer about what they really wanted. It has been an on-going process, and the interactive prototype has developed throughout the entire project lifetime based on new functionality and customer feedback. Because FluidUI only offers ten pages for free accounts, we had to start by creating a rather horizontal prototype - once the customer was pleased with that, we divided this into several vertical prototypes, emphasizing on small details instead of the whole picture. Both the customer and our group have been very pleased with the positive experiences FluidUI has brought us, enabling us to rapidly build our application based on a feedback-based interactive prototype.

6.3.4 Microsoft Visio

Microsoft Visio is a diagram and vector graphics tool, and is a part of the Microsoft Office package. This is a licensed tool that is free for students at NTNU. Within Visio there are loads of different templates and layouts for different diagrams. There is everything from basic electrical and organizational charts to office layouts.

Why we used Microsoft Visio

There are many basic diagram tools on the internet for free.Some of our team members have tried a couple of these tools with varying results, both in terms of looks and functionality. Visio was then recommended to us by another student, and after giving it a quick look we saw it was very easy to work with. It is also well documented.

6.4 Other tools

6.4.1 NetBeans

NetBeans is an open-source integrated development environment (IDE) supporting a wide range of programming languages including Java, C/C++, JavaScript and HTML5. This development

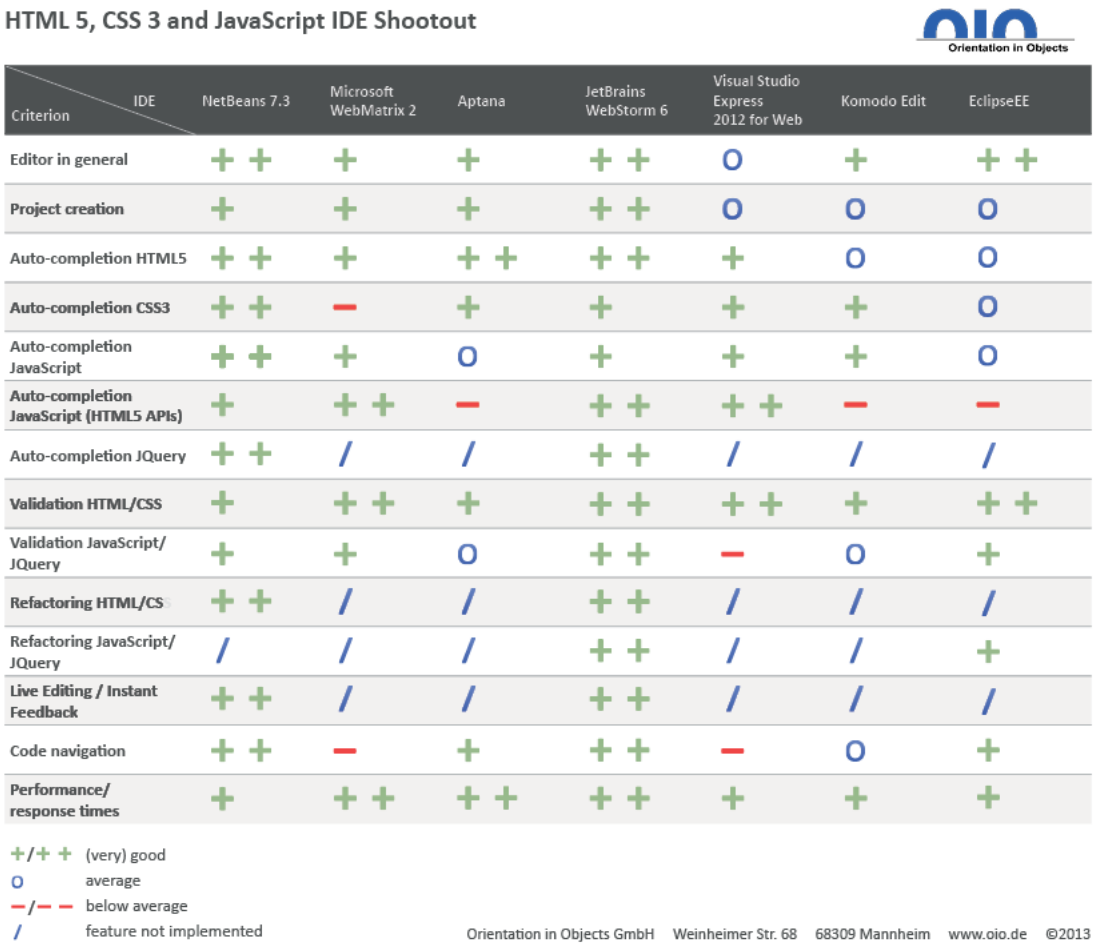
tool also offers integration with Git and support for many of the major JavaScript frameworks and libraries, including AngularJS.

Why we chose NetBeans

In the initial phases of the project, we decided that we should all use the same development tool when working. The reasoning behind this decision was that none of us were familiar with the applicable technologies that were required. By using the same IDE we were able to help each other when we had any technical issues with our development tool, and we could all recommend new features we discovered along the way. In addition, we were given the opportunity to gain experience in a development environment none of us had tried before.

There are, however, countless IDE's supporting languages and technologies aimed at web development. We did some research in the initial phases of the project to narrow down what development tool would suit us best. The figure below best summarize our choice of IDE by comparing the advantages and disadvantages of the major IDE's for HTML, CSS and JavaScript. By reviewing the comparison, it is clear that JetBrains' WebStorm offers the best support for the applicable technologies. However, by the time we started our project, WebStorm was proprietary software that had to be paid for (it is now offered to students for free through the JetBrains Student Development Pack). Because of this, we decided to use the second best supported tool, namely NetBeans.

NetBeans has been an excellent choice to suit our needs. We especially appreciate the Git integration, the continuous and well-implemented auto-completion for HTML, CSS and JavaScript, and the easy-to-use code navigation.



HTML 5, CSS 3 and JavaScript IDE Shootout

Orientation in Objects

Criterion \ IDE	NetBeans 7.3	Microsoft WebMatrix 2	Aptana	JetBrains WebStorm 6	Visual Studio Express 2012 for Web	Komodo Edit	EclipseEE
Editor in general	+ +	+	+	+ +	o	+	+ +
Project creation	+	+	+	+ +	o	o	o
Auto-completion HTML5	+ +	+	+ +	+ +	+	o	o
Auto-completion CSS3	+ +	–	+	+	+	+	o
Auto-completion JavaScript	+ +	+	o	+	+	+	o
Auto-completion JavaScript (HTML5 APIs)	+	+ +	–	+ +	+ +	–	–
Auto-completion JQuery	+ +	/	/	+ +	/	/	/
Validation HTML/CSS	+	+ +	+	+ +	+ +	+	+ +
Validation JavaScript/ JQuery	+	+	o	+ +	–	o	+
Refactoring HTML/CS	+ +	/	/	+ +	/	/	/
Refactoring JavaScript/ JQuery	/	/	/	+ +	/	/	+
Live Editing / Instant Feedback	+ +	/	/	+ +	/	/	/
Code navigation	+ +	–	+	+ +	–	o	+
Performance/ response times	+	+ +	+ +	+ +	+	+	+

+/+ + (very) good
o average
–/– – below average
/ feature not implemented

Orientation in Objects GmbH Weinheimer Str. 68 68309 Mannheim www.oio.de ©2013

Figure 6.3: Comparison of popular IDE's supporting HTML, CSS and JavaScript development

7

Architecture

7.1 Introduction

This project's goal has been to replace the existing system for generating ads at Adresseavisa. The main quality attribute that has been identified was the modifiability aspect. The customer wanted a system that responded quickly and replaces their current need of Indesign servers to convert their ads to PDF. The solution that was to be replaced was undesirably slow showing how the ad would look after the user was finished editing it. Therefore, automatic viewing of the preview, and the preview being updated in real-time according to modifications in the editing section is something that has been expressed by the customer to be desirable functionality.

The solution made has taken these two primary functions into account to produce what the customers wanted. However, since they only wanted the client side part of the system, there has been no concern regarding the server aspect of the situation.

7.2 Explanation of AngularJS expressions

Controller: A controller is a collection of methods and variables that cano directly connect with the html code. By using controllers and referencing them in the html code you can trigger different methods.
Service: A service is a grouping of methods and variables that can be injected into other services or controllers. By default they implement the singleton pattern.
Directives: A directive is a custom HTML tag or attribute which AngularJS enables us to create and read. A directive is one of the core functions supplied by AngularJS, which is why the framework is considered so powerful.

7.3 Architectural Drivers/Architecturally Significant Requirements (ASR's)

Functional requirements

For this project we were supposed to make an ad editor that would replace Adresseavisen's old ad editor for funeral ads. Designing a solution with extensibility in mind, specifically being able to use our system for all types of advertisements offered by Adresseavisen, affects the architecture. In their old system the editor does not update the preview of the ad with every change done, so performance in this regard will be part of the architecturally significant requirements.

Quality requirements

Since the main focus of this project was to make a new extendable ad editor, the main quality attribute of the project was modifiability. By discussing the attribute of usability, the customer did not specify this to be of high concern.

Business requirements

This project's lifespan was 12 weeks. This is not a lot of time to make a perfect system so the customer has to expect to be working on this after receiving our work. A drive for this project is to eliminate the Indesign server that Adresseavisa is currently spending money on to convert the ads to PDF. In order for this project to have increased value compared to the previous system, it is important that the PDF generation is present.

Technical requirements

This project is written in Javascript/Angular, CSS, and HTML5. In addition there are some libraries, such as html2canvas, jspdf, and angular-drag-and-drop-lists, that have made the implementation of some key aspects of the solution easier (including conversion to PDF). None of the participating members of the group had any experience with any of the technologies that were required by the customer.

7.4 Stakeholders and concerns

The stakeholders in this project are identified as; developers, the customers, the adviser, the end user, and the examiner of this project.

Stakeholders	Description
End user	The end users are funeral agents who will use this system to generate advertisements for the newspaper. Their concerns include a flexible system that with experience will make the job of generating ads fast and easy.
Developer	The developers are the members of group 5. Their concerns included making the system with the specifications defined by the customers. In addition they wanted to produce a well-designed solution for the project that would reflect well on them during evaluation.
Customer	The customers are developers from Adresseavisen. Their concerns include that the end result will be a base for a new system that they can extend so that they may be able to avoid paying for the Indesign servers. In addition they want the solution to be more responsive than the previous.
Adviser	The adviser's concerns regarding this project include if there should ever be a conflict with any of the group members or the customer, that he/she manages the conflict and facilitates productivity and group dynamics.
Examinator	The final report will be evaluated by the examiner. His or her concerns include how well documented the final report is and the overall execution of all the aspects of how the project was managed and completed.

7.5 Selection of architectural views (viewpoint)

Logic view

For the logic view of our solution we have decided to show the overall structure. This is to clarify to the customers and the developers how the system works. The modelling method we have chosen to use is the standard class diagram form UML. Since the solution is written in Javascript there are no real classes. However, as we use the angular extension that implements the MVC pattern this makes it possible to group parts of the code into different sections and scripts that together to show how everything is dependent on each other. The notation used is the following.

- Class: used to show the different classes' states. They are divided into three parts:
  - Name: the name of the class
  - Attributes: the attributes in the class
  - Methods: the methods in the class
- Stippled lines: Used to show which classes are dependent on each other.

Process view

We have chosen to look closer at how the final interaction of the different components in the editor will work when it is fully integrated with the customer's server. This is done by using UML's sequence diagrams. The notation consists of the following.

- Blocks are used to show the different files and users.
- Arrows indicate which method is called from the different files
- Stippled line is a time line
- Green box shows when the different files are active or waiting

Develop view

The develop view shows how the different files/classes are related. This gives the developer an understanding of how the code is structured and where he/she should make changes or add new classes to keep the consistency intact. The notation consists of the following.

- package: used to indicate packages to give an overview of the class structure. They are divided into two parts:
  - Name: the name of the package
  - Attributes: the classes contained within the package.
- arrows: used to show which packages have connections.

7.6 Architectural tactics

Given that modifiability is our architectural significance requirement, it is very important that the solution is extendable when delivered to the customer. We therefor decided to make the data structure as flexible as possible. Another tactic to make the modifiability easier was to do unit testing. This way, when refactoring parts of the code or adding new functionality, the core essence would not be destroyed. Another factor, which was also a requirement from the customer, was that we adapt the clean code approach when programming.

Since the editor that we created was written in JavaScript with the AngularJS framework, we had the opportunity to use dependency injection. This would make sure that the methods used in the various controllers, filters, services, and directives would not modify properties or variables they should not reach. In addition we use private properties and getters to access data from different "classes". By doing this we ensure that it is easy to track where a data modification has been made and makes the job of debugging simpler.

7.7 Architectural design patterns

Since AngularJS implements the model/view/controller-design (MVC pattern) it was natural that this pattern was used for this project. This pattern helped increase the modifiability of the overall solution.

The way this pattern is implemented in AngularJS automatically implements the observer pattern, giving us the advantage of being able to rapidly update multiple views of the same dataset. In addition a variation of the singleton pattern is used by angular for defining services,

ensuring that there is never multiple places where data is stored. We also used a variation of the factory pattern, which was utilized when generating the different fields required to edit the ad. By utilizing these patterns, the modifiability of the overall design of the architecture greatly increased.

Regarding better performance when editing the ad, we automatically get this feature from the MVC pattern. Since the preview of the editor and the editing part of the editor listens to the same data, the immediate response that the customer was looking for will be achieved.

MVC pattern

The MVC pattern divides the structure of the application into three different parts. The model is the core part where all the data is stored. The controller is the part that updates the data inside of the model. Lastly, the view is where the data is displayed. When a change is done to the data in the model, the view is immediately notified of the change done and updates accordingly.

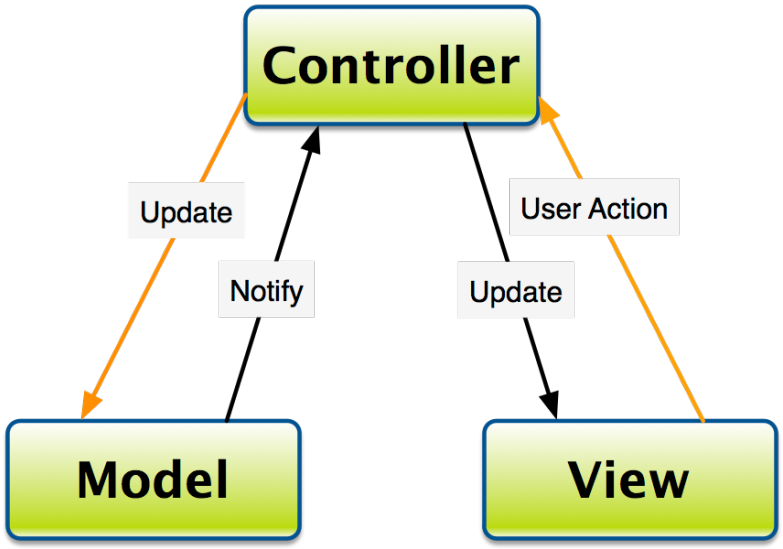


Figure 7.1: Illustration of how the MVC pattern works

Observer pattern

The observer pattern has one task. That is to make sure that when a property is updated in one part of the code, then if there are any objects that are listening for updates on this particular property, it should notify the object that is listening. This pattern is part of the core aspect of MVC.

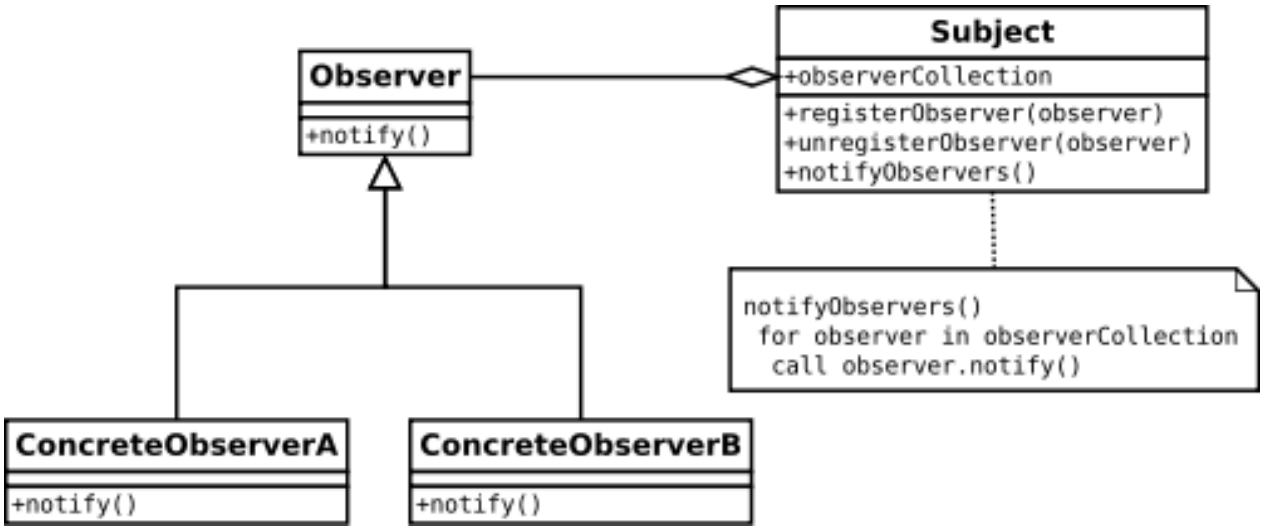


Figure 7.2: UML class diagram describing the observer pattern

Factory pattern

The factory pattern increases modifiability by defining a core object. This core object is what the program needs to be able to handle and change. This way, when generating different instances of this core object as long as it contains the core functionality, the program is not concerned what kind of object it is. This way we can generate many different objects and still have them working as if no new object was inserted into the equation.

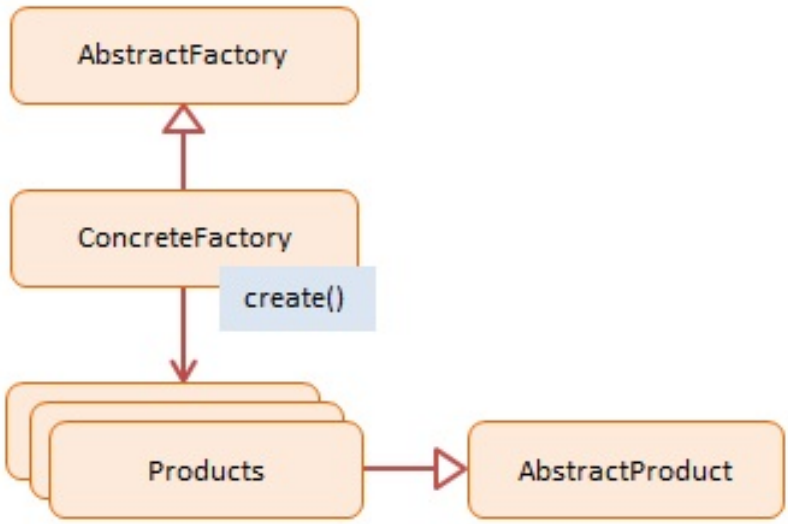


Figure 7.3: General illustration of how the factory pattern works

Singleton pattern

The singleton pattern is one of the simplest patterns. The only job for this pattern is to ensure

that there never exists more than one instance of a given object.

7.8 Views

Logic view

The logic view provides an overview of the build up of the solution and the injected dependencies that the angular solution contains. This figure shows how everything is structured.

Figure 7.4: Class diagram stating the interaction of the services and controllers

Process view

The sequence diagrams shown below give an impression of how the system will be used when saving an ad, or displaying the rules as an admin, creating a template, and approving the ad. These sequence diagrams make it easier to understand the system's integrity.

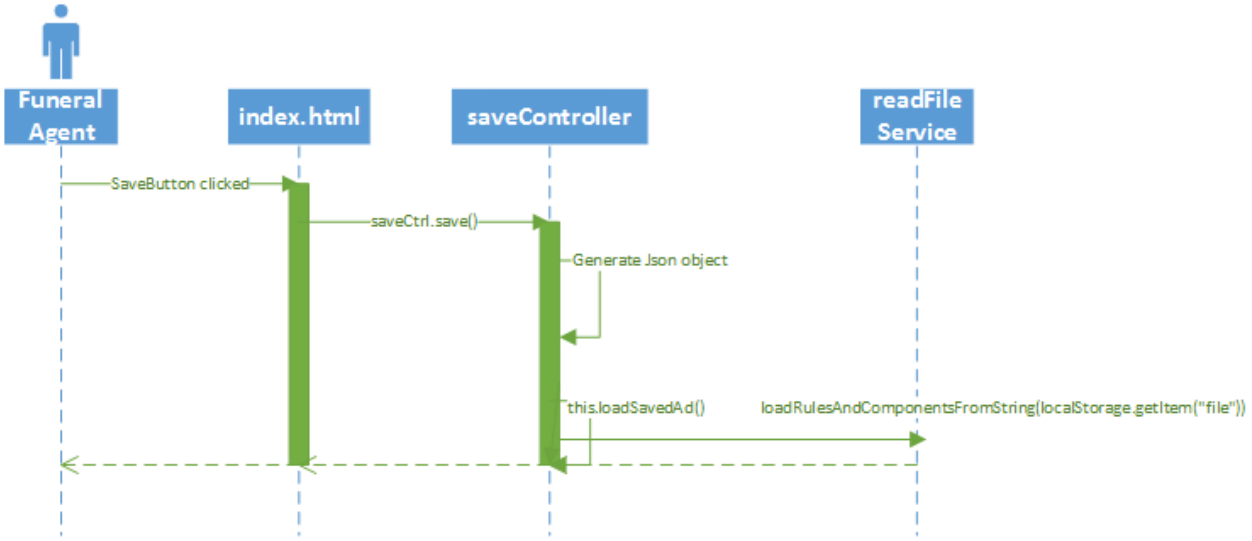


Figure 7.5: Sequence diagram for saving the ad

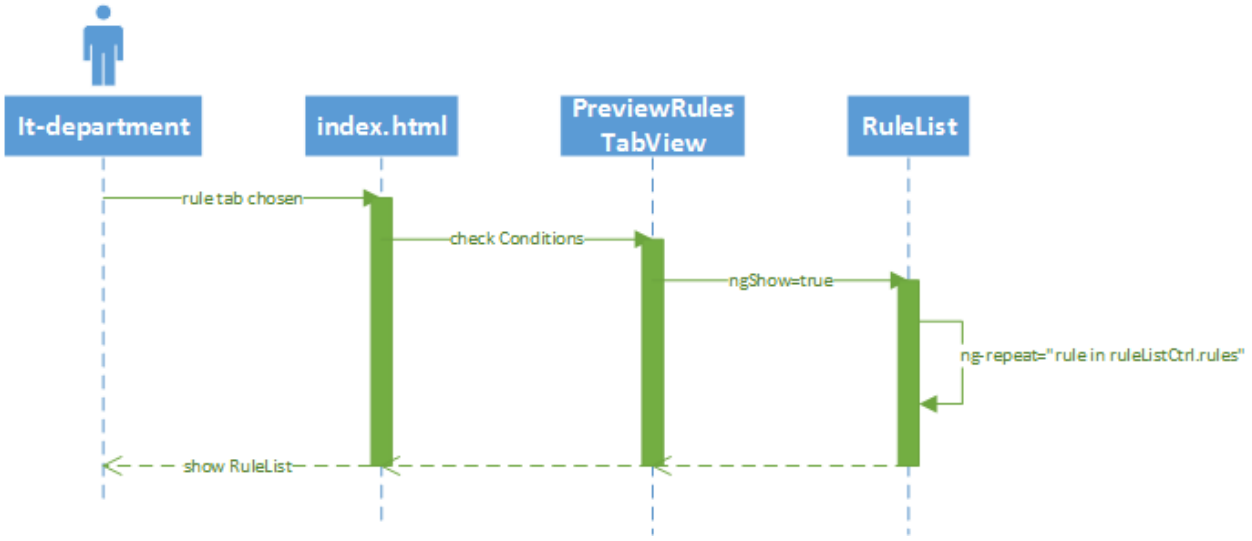


Figure 7.6: Sequence diagram for displaying the rules

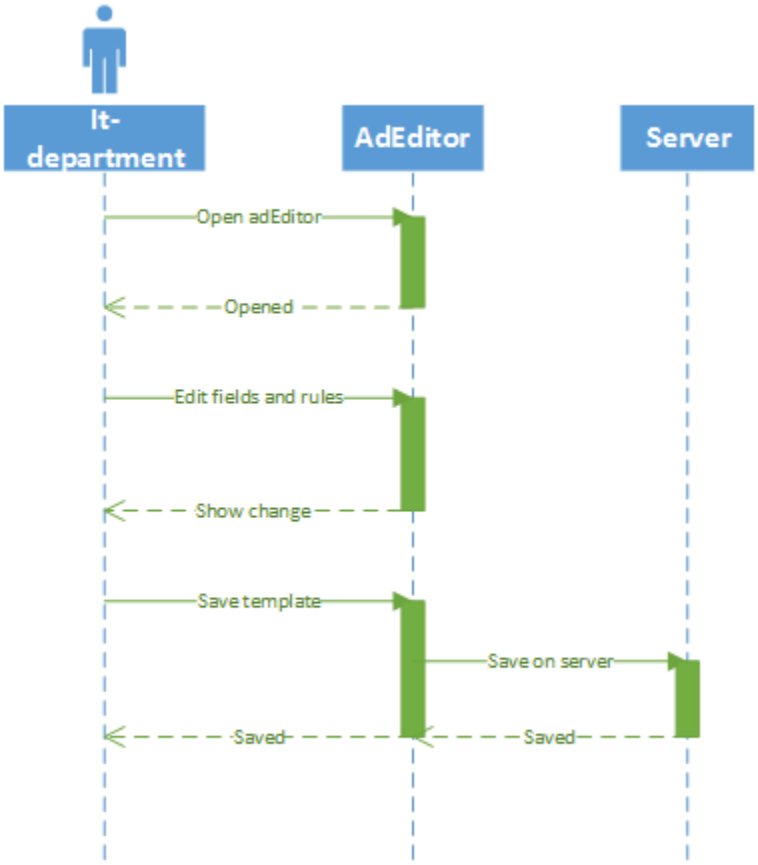


Figure 7.7: Sequence diagram for creating a template

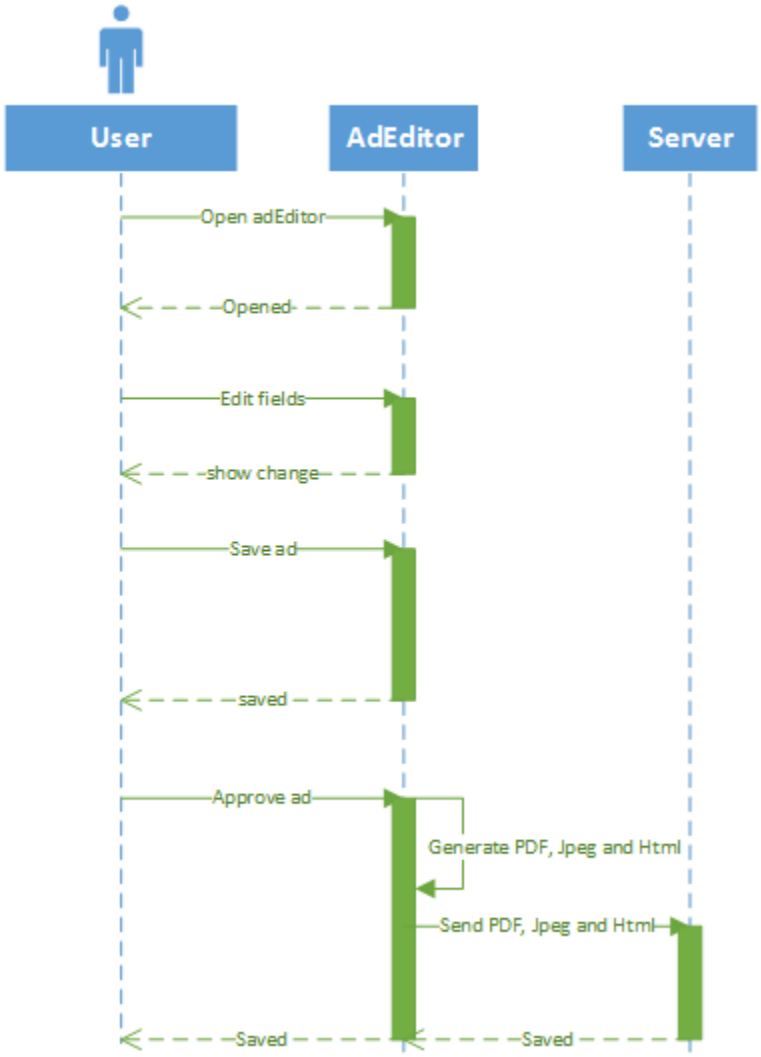


Figure 7.8: Sequence diagram for approving the ad

Development view

The development view shows how the code and it's files or "classes" are divided into different folders to better get an understanding of how the code is structured. The packages are as follows; services, controllers, and directives. In addition there are some files that are not included in any of these folders. This is because they are placed in the layer above the other files and does not fit in any of them.

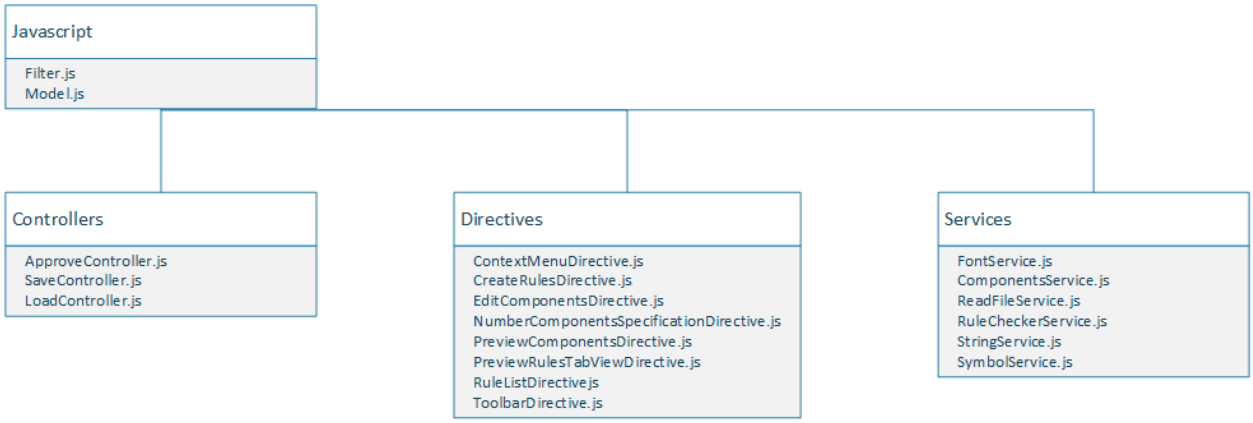


Figure 7.9: Package diagram for showing the file structure

7.9 Architectural rationale

We think that by using the MVC pattern we satisfy the goals and requirements set by the customer. This is based on the fact that MVC introduces the possibility to easier structure javascript. In addition, the MVC pattern supplies the solution with increased modifiability by making it possible to share data and listen for updates. Another increase in modifiability that angular provides is the introduction of directives that can behave as smaller view objects that can easily be put together to make a complete view.

To make sure that there is no confusion when building further on the solution, the use of unit testing and adaptation of clean code should simplify the process of taking over the code from the customer's perspective.

8

Sprints

8.1 Sprint 1

8.1.1 Planning

Before the sprint we sat down with our customer, formulated the product backlog, and assigned priority ranking to the stories. Then, when we knew what stories to prioritize, we broke down the tasks into smaller sub-tasks and assigned estimations of realization effort to each sub-task making up the main task. We assigned the tasks together as a group to give everyone tasks they wanted to work on, and come to an agreement as to the allocation. The only exempted tasks were the programming tasks, which were given to the development leader to delegate to the rest of the team. In JIRA, progress on each sub-task was reported to the team member responsible for that part of the project.

8.1.2 Sprint backlog and estimation of realization effort

Key	**Summary**	**Original estimate (h)**	**Description**
ADE-1	Create outline for the final report	10	
ADE-2	Set up coding conventions	2	
ADE-3	Explore testing		
ADE-4	Setting up the basic editor P:10		

ADE-5	ADE-4 Create CSS for a static ad (like a pre-study on the CSS out-line)	10	
ADE-7	Preview for ad	30	As a user I want to see a preview of the ad so that I can see what it will look like at the moment. The preview should update in real time.
ADE-9	ADE-4 set up basic MVC	4	
ADE-10	Textfields		As a user I can edit textfields so that I can insert text into my ad.
ADE-11	ADE-10 Enable text fields in the editor	1	
ADE-12	ADE-10 Update the content of the textfield also in the preview	3	
ADE-13	ADE-10 Change the type of the text field	5	i.e one or two columns
ADE-14	Lines	10	As a user I can insert lines in the ad so that it looks nicer.
ADE-15	ADE-14 Enable lines in the editor	2	
ADE-16	ADE-14 Update the line information in the pre-view	2	
ADE-17	Symbols	15	As a user I can have symbols in my ad which I can chose from a list so that it looks nicer.
ADE-18	ADE-10 Change the font	5	
ADE-19	ADE-10 Define cus-tomers fonts in CSS	2	
ADE-20	ADE-17 Enable sym-bols in the editor	2	
ADE-21	ADE-17 Update the symbols in the preview	2	

ADE-22	ADE-17 Let the user chose from a list of symbols (drop-down)	5	
ADE-23	Create a prototype		
ADE-25	Pre-study / Self-study		
ADE-26	ADE-25 Do the JS Tutorial	5	
ADE-27	ADE-25 Do the AngularJS tutorial	4	
ADE-39	ADE-23 paper prototype pernille	10	
ADE-40	ADE-23 paper prototype and interactive prototype paul	30	
ADE-41	ADE-23 paper prototype Linda	2	
ADE-42	ADE-23 send it to the customer to get feedback	2	
ADE-43	ADE-3 Install frameworks	3	
ADE-44	ADE-3 Gather information	8	
ADE-45	ADE-3 Test componentController	8	
ADE-47	Write requirement document	5	Finish the draft of requirements section as future sprints depend on this. May be subject to change during the project
ADE-64	ADE-4 Structure the files	5	Do research about how to best structure the documents and files in our project.
ADE-65	ADE-3 Make cheat sheet	5	
ADE-66	ADE-25 jQuery Tutorial	3	

ADE-68	ADE-23 refine interactive prototype according to the customer preferences	6	
ADE-70	Contact with customer and adviser	10	
ADE-71	ADE-25 JavaScript Tutorial	2	
ADE-75	ADE-25 inform about sprint retrospective	1	
ADE-107	ADE-3 Explore testing Pernille	2	
ADE-108	ADE-3 Explore testing Knut	2	
ADE-109	ADE-3 Explore testing Paul	2	
ADE-110	ADE-3 Explore testing	2	
ADE-111	ADE-3 Explore testing Nils	2	
ADE-112	ADE-3 Explore testing Linda	2	

Table 8.1: Tasks and subtasks assigned to this sprint, the corresponding estimated realization effort, and an optional description of the task

Planned meetings for this sprint

- Customer meetings
  - 24.09.2014
- Adviser meetings
  - 16.09.2014
  - 23.09.2014
- Work meetings
  - 30.09.2014

8.1.3 Solutions to sprint backlog items

ADE-1 Create outline for final report

Using the compendium we found recommended content for the final report, and were able to

make a tentative outline of the final report in Latex and document hiearchy in Google Drive. By doing this, all team members could add their assigned documentation tasks to the final report by inserting the document in the right place in the document hierarchy. The documentation supervisor was assigned to transfer content to Latex when needed. By finishing the outline for the final report we also have the Table of Contents for the pre-delivery of final report ready.

ADE-2 Set up coding conventions

Due to optimistic work effort estimations the team member assigned to this task was not able to do any formal work on this task. To be transferred to sprint 2.

ADE-3 Explore testing

Due to optimistic work effort estimations the team member assigned to this task, and thus members related to sub-tasks of this task, were not able to finish all the research for this task. To be completed in sprint 2.

ADE-4 Set up the basic editor

We finished a conceptual model representing the most basic story of adding text fields on the left side of the screen, and the possibility to move these text fields up or down relative to each other. To accomplish this we decided to make use of services in AngularJS. By using a service we could fully benefit from the MVC that AngularJS provides and inject our data through it. We stored the data in a list with objects where we specified which type each object should represent. Then by using directives, we iterate through the list, and based on the amount of objects in the data structure, add the right components to the view. This was done by using the built in ‘’ng-repeat” tag. Thanks to AngularJS’ MVC, the view is automatically updated according to changes done to the fields. All of this was tied together by having a controller that makes the connection with the HTML page.

ADE-7 Preview for ad

The preview on the right side of the screen is a low fidelity presentation of the given fields on the left side of the screen. It is updated dynamically with any changes made to the editor on the left side of the screen. This was done by injecting the service with the data structure to the controller that manages the preview. To make the preview show a lower fidelity we defined a new directive based on the same data, and would show the data the right way. So far the only styling implemented is a black outlining and centering of the content.

ADE-10 Text fields

Added the option of changing the content of the text fields by using ‘’ng-bind”. Implemented the customer’s fonts so that the death notice is initialized with one of the approved fonts, and

can only be changed to one of the other approved fonts. It is also possible to change between one- and two-column text fields.

ADE-14 Lines

Made it possible to add lines to the content, and enabled these lines in the preview. This was made possible by extending the directives for the editor and the preview to show the lines. Both directives with details how to show the lines, the one for the editor and the one for the preview.

ADE-17 Symbols

Made it possible to add symbols from a list of predefined symbols to the content, and enabled the preview of said symbols. To access the symbols we made a service that contains all the paths to all the different images stored in a list. Then, by injecting this to the controller for showing the preview and the editing part of the editor, and using a value indicating the position in the list, the preview knows which image to apply in the preview.

ADE-23 Create a prototype

We created both a paper prototype and an interactive prototype to show the customer, for feedback on how to design the user interface. We repeated this process until the customer was satisfied with the prototype.

ADE-47 Write requirement document

Research about the IEEE Recommended Practice for Software Requirements Specifications [13] provided outline for the requirement section. Information for this section was discussed during several meetings internally, with customer, and with adviser and ready to be filled into the outline.

8.1.4 Burndown chart

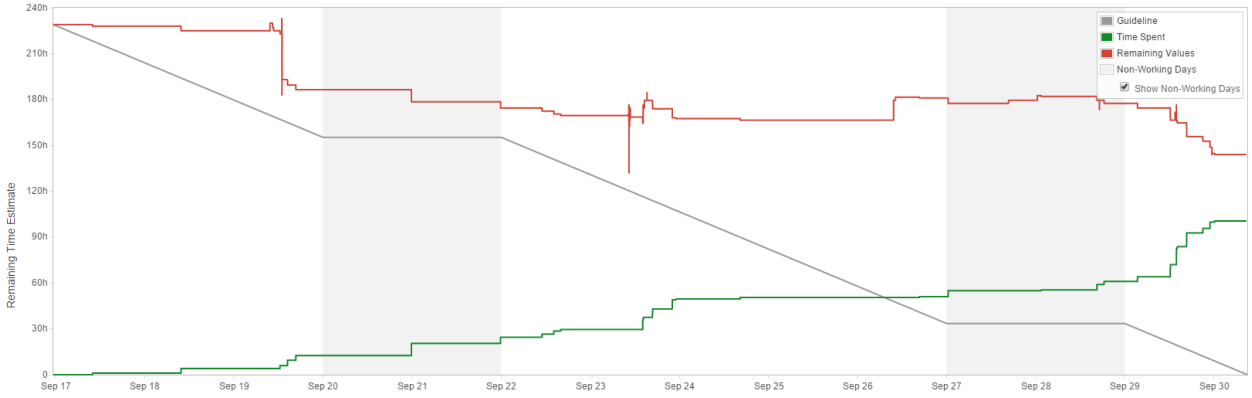


Figure 8.1: Correlation between work hours estimated and work hours done in sprint 1

The burndown graph illustrates the correlation between the total work effort estimated and total effort logged in JIRA, during this sprint (see subsec:jira for a detailed explanation about JIRA and which tasks are included). The large vertical changes in the estimation signify a change to the backlog, either adding or removing a task, or changing an estimate. Where the change in logged effort differs from the change in remaining estimation, a task has demanded more or less effort to realize than estimated. The distance between the red graph and the x-axis at the end of the sprint is the sum of the remaining estimate for tasks that were not finished, and the overestimation of tasks that were completed.

As shown by the graph, the effort needed to solve tasks were generally underestimated. In addition, the remaining time estimate demonstrates that the workload committed for this sprint was too great. The vertical changes to the estimates during the sprint which were immediately undone, were mistakes which we were not able to undo in the chart.

8.1.5 Customer feedback

The customer reported that they were happy with how we performed during our meetings with them. Regarding our prototype the customer suggested some changes in layout and functionality, but reported that they were very happy with the prototype and thought it looked nice. They were happy about our progress as well, and said we were further ahead with the project than anticipated. All in all they thought the application looked exciting, and we were excited that the customer liked our work.

The customer has been introduced to JIRA in accordance with our transition from Trello to JIRA, and commented that JIRA looked great. Lastly, their impression about our group dynamics were good and we seemed to be working together very well.

8.1.6 Sprint evaluation

Introduction

At the end of the sprint we had a retrospective meeting, discussing our experiences from this sprint. This evaluation is compiled from the meeting minutes, and contain our expectations for the retrospective meeting, important events, discussion about the outcomes of the sprint, achieved and prospective improvements, and conclusion to the meeting.

Expectations from meeting

Everyone had different expectations for this meeting. To sum up we expected to get a clear picture of our accomplishments, map what the next sprint's main points were going to be, solve problems regarding group dynamics and improve for the next sprint, and discover possible unknown problems or areas of improvement.

Important events

Together we brainstormed the events we found most memorable or had the strongest impact on the progress of the project. Afterwards we discussed and categorized the events. The list with categorizations is found in appendix D.1.

- Spent time getting familiar with AngularJS
- Setup the basic editor
- Made paper and interactive prototype of the application
- Had difficulties scheduling regular meeting with customer
- Knut was sick the most of the sprint
- One team member was several hours late two times
- Linda went diving for a week
- Moved project organization from Trello to JIRA, combined with Bitbucket
- Overflow of documents in Google Drive in need of organizing
- Outlined final report
- Got feedback from the customer, and they were satisfied with our progress
- Our adviser was satisfied with our progress
- Due to complexity of testing environment these tasks were moved to sprint 2
- Team members were not too familiar with Git, and we spent more time than expected solving problems with Git

Discussion

Disappointments

At the start of the sprint we had difficulties scheduling regular meetings and receiving feedback from our customer. This was mainly due to vacation and illness both on ours and the customer's side, and made us uncertain which direction to take the design of our prototype. However, this issue was sorted out quickly. The group was disappointed to have one of the team members show up some hours late a couple of times, and in accordance with our rules that person had to make cake.

Upturns

Some tasks went better than expected this sprint: AngularJS turned out to be a very good choice for this project, and easier to learn and faster in development than expected. The transition from Trello to JIRA was complicated at first, but we agreed that JIRA was superior and more practical for our project. After a couple of days the team felt comfortable using it; team members were handling their main responsibility of the project very well.

Surprises

Only one big surprise was noted, being that team members did not seem as interested in coding as anticipated. Everyone had different reasons for not getting as involved as they wanted to, and this is planned to change in sprint 2. Although it presents a risk concerning unanticipated absence that not everyone has knowledge about the code, the basis for the application was made efficiently with few code conflicts.

Future

- Start doing
  - Get involved in coding
  - Get involved in documentation
  - Refactor document organization
  - Have more breaks and make them more defined, so we can be more concentrated when working
  - Respect and not interrupt the person who got the word in discussions
- Stop doing
  - Joking around in work meeting and early morning work
  - Talk at the same time
- Continue doing
  - The responsibility assignments
  - Group organization

Conclusion to meeting

We had a turbulent first sprint, but the whole team was invested in the project and wanted a happy and productive work environment, thus we always get to an agreement when there are deviating opinions. By organizing our group we did indeed get more work done and everyone knew who to ask for help on a particular subject. The retrospective meeting was good for the working environment, and all members understood what could be improved and are set for improving the teamwork in the next sprint.

8.1.7 Summary

We learned that we needed to break down the main tasks into even smaller tasks, ideally tasks smaller than 4 hours estimated realization effort, because some of the initially large tasks needed to be moved to the next sprint, which is not ideal. In addition, we needed to distribute the tasks immediately instead of leaving several tasks for, i.e. the development leader, to distribute on their own. By distributing the tasks immediately, every single group member would be able to influence the estimations, leading to a more well refined overall estimation. We discovered a mismatch between the estimated effort and the real effort, where we estimated more hours than we needed for those tasks we finished, and had too many tasks to finish with the current team disposition, partly due to unforeseen illness. Tasks that went unfinished in this sprint will be transferred to sprint 2.

We did a lot of work in this period, including some unanticipated work transitioning from Trello to JIRA, but in the end it worked out nicely and we have concluded that it would benefit us in the long run. Any disagreements discovered during the course of the sprint were discussed and dealt with, and we have defined clearer communication with the customer.

In the upcoming sprint we expect the team members to be involved in all aspects of the production, and we will work on group dynamics to perform better as a team and utilize the available time better.

8.2 Sprint 2

8.2.1 Planning

At the beginning of the sprint we had a group meeting for creating the sprint backlog. We extracted the stories with the highest priority from the product backlog, and made tasks and subtasks to cover the stories' requirements. We did this until we had filled our work quota for this sprint, and then we assigned the tasks. According to what we learned in the previous sprint we estimated the work effort and assigned every task immediately, with some exceptions: Due to

the documentation supervisor being away at the end of the planning meeting we did not divide the documentation tasks until a day later.

One of the goals we made in the retrospect of sprint 1 was for all members to participate in all parts of the project. To some degree we accomplished this goal, but we decided to prioritize the programming for this sprint to meet as many of the customer's requirements as possible. To do this some members were assigned more programming than others and no documentation. Since the documentation tasks were assigned later than the others, they were used to fill in the differences in planned work effort between the team members, and were not evenly distributed. The tasks we did not finish in sprint 1 were transferred to this sprint.

8.2.2 Sprint backlog and estimation of realization effort

Key	**Summary**	**Original Estimate (h)**	**Description**
ADE-2	Set up coding conventions	2	
ADE-3	Explore testing		
ADE-24	Create estimate diagram of Sprint 1	5	
ADE-29	save the ad P:8		As a funeral agency I can save the ad so that it is delivered to adresseavisa.
ADE-30	Load template / ad P:8		As a funeral agency I want to select which template I use. P:8 Aa a user I can load saved ads and templates so that I can change them.
ADE-33	Textfields P:6		As a funeral agency I can add text fields so that it is more flexible and customizable.
ADE-34	Delete fields P:6		As a funeral agency I can delete text fields so that it is more flexible and customizable.
ADE-43	ADE-3 Install frameworks	3	
ADE-44	ADE-3 Gather information	8	
ADE-45	ADE-3 Test componentController	10	
ADE-48	Write sprint 1 document		Write documentation for the first sprint

ADE-49	Write choice of lifecycle model document		Write why we chose agile development and scrum instead of waterfall and others. This is going to be a part of the pre-delivery
ADE-50	Write pre-study document		Research and write pre-study document for pre-delivery of final report
ADE-65	ADE-3 Make cheat sheet	5	
ADE-72	Pre-study / Self-study Sprint 3		
ADE-73	ADE-72 JavaScript Tutorial	6	
ADE-74	ADE-72 AngularJS Tutorial	5	
ADE-77	ADE-29 save file format locally	8	Add a save button and store the current state of the ad locally
ADE-79	ADE-29 add a save button	0.167	add a save button
ADE-80	Find a way to formulate the rules		As an IT department I can define rules for the templates so that I make sure that the ad will not be messed up totally.
ADE-81	ADE-80 How to check the rules	6	
ADE-82	ADE-80 How to enforce the rules	6	
ADE-83	ADE-80 create a small set of rules	1	like a before b
ADE-84	Create file format for the template / ad		As a IT department I can store templates so that I can provide them for the funeral agency. As a funeral agency I can store the ads.
ADE-85	ADE-84 Research file format and create one template	2	especially JSON
ADE-86	ADE-84 Fields format	1	
ADE-87	ADE-84 Rules Format	4	
ADE-88	ADE-30 create load button	0.167	

ADE-89	ADE-30 display local folder	2	
ADE-90	ADE-30 load file from local folder	8	
ADE-91	ADE-84 Field description format	1	
ADE-92	Interactive prototype		
ADE-93	ADE-92 update the existing prototype	2	
ADE-94	ADE-92 improve the interactivity	4	
ADE-96	ADE-34 Hide the field and make it invisible	2	
ADE-97	ADE-34 deal with the hidden field	5	Store the field somehow and make sure that it is considered in enforcing the rules.
ADE-98	ADE-34 add the delete button left to the field description	0.167	
ADE-99	Field description		As a funeral agency I can see and edit the field description so that I know what is supposed to be in the field.
ADE-100	ADE-99 add a generic description as a textfield (editable (for now)) to every field when the field is added	2	
ADE-101	ADE-99 add a default value	1	e.g. "added #2"
ADE-102	ADE-33 enable \n and consider it in the preview	4	
ADE-103	ADE-33 implement different kinds of textfields	6.5	single line multi line two columns
ADE-104	ADE-33 change font by right click	4	

ADE-105	ADE-33 change font size by right click	3	
ADE-106	refactor documents in google drive	3	identify unnecessary documents –> trash folder reorganize folder structure one folder with files at have to be edited a lot (e.g. phase log)
ADE-107	ADE-3 Explore testing Pernille	2	
ADE-108	ADE-3 Explore testing Knut	2	
ADE-109	ADE-3 Explore testing Paul	2	
ADE-110	ADE-3 Explore testing	2	
ADE-111	ADE-3 Explore testing Nils	2	
ADE-112	ADE-3 Explore testing Linda	2	
ADE-113	ADE-72 Angular tutorial	2.5	
ADE-114	ADE-50 Describe situation and solution of today	1	
ADE-115	ADE-50 Describe the wanted situation and possible solutions	3	Compare benefits and outcomes of different alternative solutions
ADE-116	ADE-50 Export total scrum backlog to document	1	
ADE-117	ADE-50 Market research	3	Find competing solutions and if they could replace our software/we could implement their software, possible market opportunities for this software
ADE-118	ADE-49 Waterfall advantages	2	Short description of the waterfall model (and others, if needed), and it's advantages and disadvantages
ADE-119	ADE-49 Scrum advantages	2	Short description of the model, and it's advantages and disadvantages.

ADE-120	ADE-49 Comparison of (dis)advantages, conclusion	2	Compare advantages and disadvantages of discussed models, and conclude why we chose scrum (or did we in reality implement a scrumish model?)
ADE-121	ADE-48 Sprint backlog/specification	2	List tasks and priority together with the estimated work effort, and list our planned meetings for this sprint
ADE-122	ADE-48 Short description of solutions to sprint backlog	3	
ADE-123	ADE-48 Customer feedback	1	Sum up what little feedback we have received in this sprint
ADE-124	ADE-48 Sprint evaluation	2	Write evaluation of this sprint, main points being what we discovered during our evaluation meeting on Tuesday.
ADE-125	ADE-80 enable the creation of rules in the editor	3	This is mainly for testing purposes, so that the rules and their enforcement can be tested even if we cannot load an ad yet.
ADE-126	Contact with customer and adviser	15	Contact with customer and adviser, and overview of all attached documents
ADE-155	ADE-34 consider available fields	3	add the hidden component to the list of available fields without destroying its characteristics
ADE-168	ADE-3 create example test	4	

Table 8.2: Tasks and subtasks assigned to this sprint, the corresponding estimated realization effort, and an optional description of the task

Planned meetings for this sprint

- Customer meetings
  - 02.10.2014
  - 14.10.2014
- Adviser meetings
  - 07.10.2014
  - 14.10.2014
- Work retrospective meeting
  - 14.10.2014

8.2.3 Solutions to sprint backlog items

ADE-2 Set up coding conventions

The code quality manager outlined coding conventions for the rest of the team to follow. Conventions related directly to programming were based on principles from "clean code" and letter conventions known from, among others, Java and JavaScript development. Some project specific conventions related to naming and referencing of files and dependencies were also defined. See appendix [A] for details about the coding conventions.

ADE-3 Explore testing

Due to the complexity of the testing environments, the test leader had not yet been able to write a working test on our project environment. The testing frameworks to be used, had been selected and were working. In addition an example test had been written, and was working. The requirements of the testing environment was not yet fully understood by our team, but we assume the dependencies specific to this project prevented the tests from working. Because of these impediments all of the team members had not been able to complete their subtasks for exploring the testing environment. We would discuss further actions in our sprint 3 planning meeting.

ADE-24 Create estimate diagram of Sprint 1

We created a spreadsheet for our work estimations for sprint 1, but soon decided to abolish this task in future sprints. We argued that JIRA was better suited for this task and included other useful functionality, making the spreadsheet redundant. To make the customer and adviser able to view our estimations and track our progress, we invited them to our JIRA project.

ADE-29 Save the ad

With HTML5 a specification for "Web Storage" was introduced, which implemented local and persistent storage of key-value pairs. By combining all the necessary data that was used for displaying the ad, mainly the list with the data objects, we used JSON's stringify method to convert it to valid JSON format. We then used this local storage to save the objects, and planned to pass these fields to the customer when they needed to save an advertisement on their servers.

ADE-30 Load template/ad

By adding a form for uploading a template file and using a directive that reads the file's data, it was possible to add the data to a variable. The directive was not going to be used by the customer, because all of the data should be sent from their server and not loaded from a local file on the client's computer. Then by using JSON's parse method to extract the data out of the file

it was added to the right lists in the service updating the view with a ready made set of different components.

ADE-33 Textfields

To signify insertion of new lines (by pressing the return key) in the text fields we add an escape character to the text in our object. To preview line breaks and multiple spaces (HTML strips away line breaks and whitespace by default) we used one of the built-in AngularJS filters. The filter exchanges the new line escape characters with HTML line breaks, and multiple spaces with the HTML whitespace escape character.

By clicking the right mouse button inside a text field an event was fired, which make a menu show up in the position of the mouse. The menu contained options for changing the font style and font size to a predefined set of values, and applied to the text in the clicked text field.

We planned to implement support for two-column text fields, as required by our customer, but were not able to do it during this sprint. This subtask has been transferred to sprint 3.

ADE-34 Delete fields

As a result of underestimation of other tasks, we did not have the time to do this task. It has been transferred to sprint 3.

ADE-80 Find a way to formulate the rules

Formulating the rules was done by translating the rules decided upon by the customer, to a language we could apply to our application. The rules formulated are described in the table below. We added a subtask to this task for implementing the enforcement of rules, but due to unforeseen complexity we spent more time than estimated finishing this task. The set of rules implied many possible combinations, and the system needed to accept these dynamically. We found a solution by combining rules implemented as fields in the data objects with a separate object containing rules. When discussing how to best implement the rules, an unforeseen problem arose. At this point, when you wanted to switch place of two components you would have to press a button: One for moving up, and one for moving down. This lead to the complexity of evaluating if the buttons should be shown or not. Due to this complexity and discussion with the customers, we decided to remove the buttons, and implement drag and drop as a substitute.To implement drag-and-drop functionality we needed to adapt the rule enforcement, which was done in sprint 3.

Rule	Explanation
A at position X	Component A has to be at position X, where X is a number in the list.
A not at position X	Component A is not allowed to be at position X, where X is a number in the list.
A before B	Component A has to be in front of component B, i.e. the index of component A in the list of components has to be smaller than the index of component B.
A directly before B	Component A has to be directly in front of component B, i.e. the index of component A in the list of components has to be one smaller than the index of component B.
A not directly before B	Component A is not allowed to be directly before component B, i.e. the index of component A in the list of components must not be one smaller than the index of component B.
A is bottom	Component A has to be at the last position of the list of components.
A not bottom	Component A must not be at the last position of the list of components.

Table 8.3: Available rules for the template creation

ADE-84 Create file format for the template/ad

We decided to use JSON as file format for the templates and saved advertisements. The data objects were trivial to save in JSON format, as JSON supports JavaScript objects. Because of the solution to the task "ADE-80 Find a way to formulate the rules" it was natural to save the rules in the same file as the data object. Some of the rules were saved in the data objects and some in a separate object for rules. As part of this task, the assignee made an example template for the outline of fields and rules.

ADE-92 Interactive prototype

We changed our existing prototype according to customer feedback and improved the interactivity.

ADE-99 Field description

The customer wanted each field in the editor to have a description in order for the user to easily identify the purpose of the field. This was implemented as a text field in each advertisement field, and was supposed to be editable to tailor to the users needs. After feedback from the customer it was decided that each field should have a predefined description, which was implemented before the end of the sprint.

ADE-106 Refactor documents in Google Drive

In the evaluation of sprint 1 we decided to reorganize our files in Google Drive. Because of the

accumulation of documents it was getting difficult locating the different documents. We tjerefor removed unnecessary files and changed the document hierarchy.

8.2.4 Burndown chart

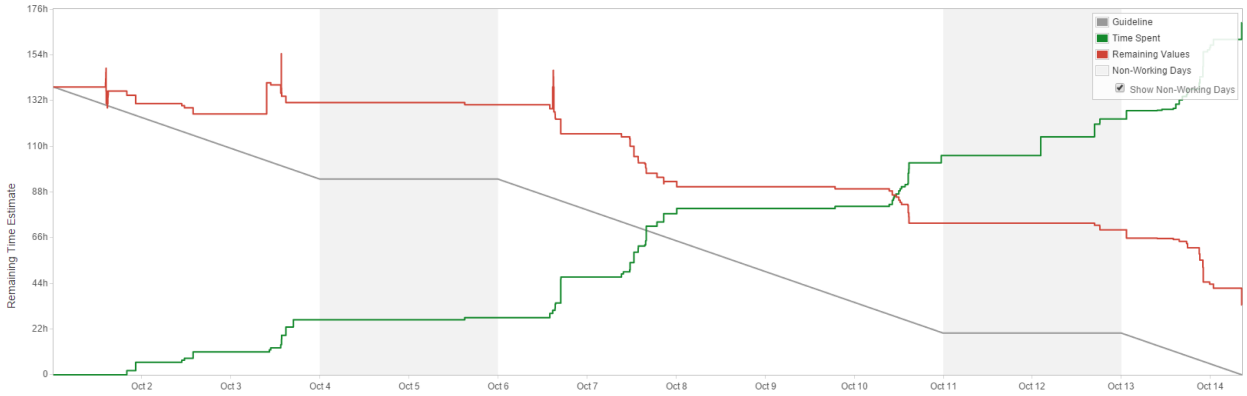


Figure 8.2: Correlation between work hours estimated and work hours done in sprint 2

For a general explanation of the burndown chart, see subsection 8.1.4.

As shown by the graph, the effort needed to solve some tasks were greatly underestimated, evident when comparing the total time spent with the initial remaining estimate. Not all tasks were finished due to the underestimations, which was evident by the remaining estimate at the end of the sprint. The vertical changes to the estimates during the sprint which were immediately undone, were mistakes which we were not able to undo in the chart.

8.2.5 Customer feedback

The customer was satisfied with how we expressed the rules they had specified. They would send us the rules regarding the double death notice (where information about two deceased persons is displayed in a single obituary) as well, as this is not covered completely by our current rules. They would also give notice if they wanted us to implement any additional rules. The customer expressed the need for the rules to be flexible rather than to be restricting, in regard to letting the funeral agencies make the changes they wanted.

During our customer meeting we told the customers about our impediment with JavaScript testing. They have no experience with the technology, but emphasized that they want us to implement testing in the application. We would take this into consideration when planning the next sprint. Together with the customer we reviewed the product backlog, in which we changed the priority of the story “ADE-37 Create templates” from priority 1 to 3. The story with the highest priority, “ADE-139 save the add as pdf”, they emphasized should be our main concern during

the next sprint. Completing this task would result in nearly fulfilling all basic requirements given to us from the customer. Optional requirements and nonessential functionality, they noted, were of no concern until we finish the hard requirements.

For the next presentation the customer would like to see the application with styling, and the current mock symbols switched with the images they had provided. They also requested that we put the provided images in sorted folders dependent on categories, and used religious and nonreligious as an example. In the conclusion of the meeting they expressed satisfaction with our work so far.

8.2.6 Sprint evaluation

Introduction

At the end of the sprint we had a retrospective meeting, discussing our experiences from this sprint. This evaluation was compiled from the meeting minutes, and contained our expectations for the retrospective meeting, important events, discussion about the outcomes of the sprint, achieved and prospective improvements, and conclusion to the meeting.

Expectations from meeting

Now that we had the experience of a retrospective meeting and we improved in some respects from the last time, our expectations were a little different as well:

- Going to be fun and productive
- Reflect on last sprint to help us improve for the upcoming sprint
- If something is bothering someone, they will bring it up
- Clearly define what we did in this last sprint

But as before, the main points are to identify our accomplishments and improve the team dynamics in the next sprint.

Important events

We brainstormed the events individually, and discussed and categorized the events together afterwards. This list contains our combined lists of events, and may at some points be contradictory as a result of different experiences. The list with categorizations is found in appendix D.2

- Documentation
  - Sprint 1
  - Choice of lifecycle model
  - Preliminary study
- Clarification from customer regarding
  - Rules

  - Images and fonts
  - Implementation of
    * Right-click menu
    * Save and load functionality
    * Rule enforcement
    * JSON file format

- Pizza night (social gathering)
- Customer and adviser meetings
- Needed to reschedule customer meeting
- Came closer to writing test
- The adviser suggested to make our own company
- Some productive work meetings
- Knut got a warning for being late last friday

Discussion

Disappointments

The adviser told us during one of the meetings our project might not be complicated enough. Our team members interpreted this in different ways, some for the worse. We decided that, in the end, there was nothing we could do but do our best with the given task.

The whole team agreed that the first social gathering we held was really fun, and we should do this or some similar social activity again. Some members are sad that other's plans, like going out or working, have prevented us from doing it yet.

Upturns

We were happy the entire group took the project seriously, and that everyone completed their tasks in a timely manner. The team agreed that our pizza event was enjoyable and should be repeated, only with more candy. Also, everyone was happy that the adviser seems pleased with our work.

Surprises

We expected the tasks involving testing and implementation of rules to be hard, but it surprised us how hard it actually turned out to be.

Future

In addition to discussing points of improvement, we discussed if we achieved any of the goals we set in this section in the previous retrospective meeting. The points we felt we achieved are listed here:

- Everyone should get involved in coding
- Refactor document organization

- Have one person controlling the discussion
- The roles, or areas of responsibility
- Being organized

Items we felt we improved, and want to continue improving:

- Everyone should get involved in documentation
- Define beginning and end of breaks
- Stop joking around in work meeting and early morning work
- Stop talking at the same time

This is our new list of improvements, also containing the items we want to continue improving:

- Start doing
  - Everyone should get involved in documentation
  - Define beginning and end of breaks
  - Bring earphones (for individual work)
  - Add UML tasks for our Master of Models
  - Add tasks to review all documentation
  - Possible to stay home and work if announcement is given (on mondays and tuesdays it is only
  - possible to leave early)
  - Mondays 1200-1330: Reserved for preparing documents to meetings on Tuesday
  - Update JIRA before the end of the sprint
  - Prepare better questions for the adviser
  - Include the adviser where we can benefit from it
- Stop doing
  - Joking around in work meetings
  - Talk at the same time
  - Forget logging work in the phase log
- Continue doing
  - Involve everyone in the code
  - Controlled discussions
  - Roles and areas of responsibility
  - Being organized
  - Social gatherings

Conclusion to meeting

Everyone were pleased with the progress we had made, and were happy about where we are in the project. We had improved as a team during this sprint, and everyone go along very well. So far we had not had any big discussions or fights, which makes for a good work environment. Because of the friendly environment we like to work together and we had scheduled several work sessions each week. During this sprint we have had great progress, which we agreed came as a result of our focus and well defined responsibilities.

8.2.7 Summary

The task assignment went better in this sprint than the previous. Since we assigned all tasks immediately we felt more organized and confident regarding what work we needed to do in this sprint. The estimations were better in this sprint, but we underestimated the effort of creating a testing environment, and we added an important task which took a lot more time than anticipated. The result was that we needed to transfer tasks to the next sprint this time as well, but in total we did put in a little more work effort than we estimated.

With the implementation of right-click menus, save and load functionality, and enforcement of rules, the application has come a long way in fulfilling the requirements set by the customer. Both the customer, the adviser, and the team were happy with the progress during this sprint. As it was now the customer could only create templates by manually writing a JSON file specifying the template, but the customer noted this solution was acceptable on their part. We would hopefully be able to implement creation of templates using the application, but this was not a prioritized task at the moment. After we improved the prototype we made a commitment to this as the final graphical design, and we should be able to start styling the application in the upcoming sprint. Because we would continue to add functionality to the application, we did not expect to be finished with the styling until near the end of the project.

There had been no surprises considering the team disposition in this sprint, with no illness or travel preventing the progress of the project. The only impediment not directly related to the programming was that one person experienced computer problems the last day of the sprint, but he was able to fix it in a couple of hours in parallel with some other work. The rules were more complicated than expected, but due to the importance of this functionality it was prioritized in favor of other tasks. We were able to achieve our goals considering the rules, and the customer was satisfied with our solution. Although the sprint backlog was not finished according to plan, we are satisfied with the accomplished work. The testing was still a problem, and since the customer wanted us to implement a working test environment we would concentrate more of our resources on this task in the next sprint. During the sprint some of the team members fell behind schedule due to other school activities, but this would be solved without any impact on the end result.

In the next sprint we expected to improve our focus further to be more efficient in our work, and make our breaks more defined. Hopefully we would be able to arrange a social gathering to compensate for lack of socializing during work and reward us for our effort. We wanted to involve more of the team members in the documentation, and we would begin reviewing our documentation continuously to be better prepared for the final report delivery. Up till now we had experienced some problems regarding late logging of work, both in our phase log and in JIRA, and we wanted to improve this so we were always up to date and ready for delivery of documents prior to meetings. In case we had any problems we would involve our adviser earlier,

in order to make use of his experience to solve the problems efficiently.

8.3 Sprint 3

8.3.1 Planning

Before the beginning of the sprint we had a meeting with the customer, and as a result we made some decisions about what tasks to prioritize during this sprint: As the customer provided more template specifications, we had to extend our rules and the rule enforcement to cover these scenarios as well. It was decided that this task, in addition to implementing the file generation of the approved advertisement, was top priority this sprint. Next we changed the priority of task *ADE-37 Create templates* from 1 to 3, which did not affect this sprint's backlog. While we began planning the interface for interacting with the customer's servers, the final decision was postponed to the next meeting, because the customer was unsure of their needs. In the meantime we agreed to keep in contact by e-mail to discuss new information regarding this issue.

During the meeting we also discussed the issue of testing the code. Despite our efforts implementing testing of the code, we had not been able to fully realize this yet. The customer explicitly noted, though, that they still want this as a part of the final delivery. To ensure their satisfaction we would involve more resources in testing during this sprint.

With a clear picture of our customer's needs and wishes we were able to assess the difficulty of each story. While extracting stories and tasks from the product backlog we created sub-tasks, estimated the work effort and assigned them to the project members. We repeated this process until we had filled our work quota for this sprint. This sprint meeting all team members were present during the entire meeting, and thus we were able to estimate and assign all tasks for this sprint immediately.

8.3.2 Sprint backlog and estimation of realization effort

Key	**Summary**	**Original Estimate**	**Description**
ADE-3	Explore testing		
ADE-33	Textfields P:6		As a funeral agency I can add text fields so that it is more flexible and customizable.
ADE-34	Delete fields P:6		As a funeral agency I can delete text fields so that it is more flexible and customizable.
ADE-43	ADE-3 Install frameworks	3	

ADE-44	ADE-3 Gather information	8	
ADE-45	ADE-3 Test componentController	10	
ADE-46	Pre-delivery of final report		Write table of contents, and gather abstract, introduction, pre-study, choice-of-lifecycle model and table of contents in final report. Proof-read and deliver to our adviser for external examination
ADE-51	Write abstract document	2	For pre-delivery report: Do your own research on the topic, but what I found was: "In short, everything goes in the Abstract. Its purpose is to provide a summary of the whole report or thesis. In this context, it is similar to the Conclusions chapter, except that the Abstract gives the individual chapters more even weighting and is typically much shorter overall."
ADE-52	Write introduction document	2	For pre-delivery document: Briefly outline problem and motivation for the project, summarize the existing solution, write the objective/task that the customer wants us to accomplish, and describe shortly the outline of the report
ADE-53	Write sprint 2 document		
ADE-55	Write tools and technology document		
ADE-65	ADE-3 Make cheat sheet	5	
ADE-72	Pre-study / Self-study Sprint 3		
ADE-73	ADE-72 JavaScript Tutorial	6	
ADE-74	ADE-72 AngularJS Tutorial	5	

ADE-96	ADE-34 Hide the field and make it invisible	2	
ADE-97	ADE-34 deal with the hidden field	5	Make sure that the drag and drop still works.
ADE-98	ADE-34 add the delete button left to the field description	0.167	
ADE-102	ADE-33 enable and consider it in the preview	4	
ADE-103	ADE-33 implement different kinds of textfields	6.5	single line multi line two columns
ADE-104	ADE-33 change font by right click	4	
ADE-105	ADE-33 change font size by right click	3	
ADE-107	ADE-3 Explore testing Pernille	2	
ADE-108	ADE-3 Explore testing Knut	2	
ADE-109	ADE-3 Explore testing Paul	2	
ADE-110	ADE-3 Explore testing	2	
ADE-111	ADE-3 Explore testing Nils	2	
ADE-112	ADE-3 Explore testing Linda	2	
ADE-113	ADE-72 Angular tutorial	2.5	
ADE-127	ADE-53 Sprint/backlog specification	2	List tasks and priority together with the estimated work effort, and list our planned meetings for this sprint
ADE-128	ADE-53 Short description of solutions to sprint backlog	3	Just give the overall picture and describe the solution to the stories/main tasks

ADE-129	ADE-53 Customer feedback	1	Gather all customer feedback we received and put it in full sentences
ADE-130	ADE-53 Sprint evaluation	4	Use what we learned from the retrospective meeting to put together an evaluation of the sprint (see the temp project report file if you need a suggestion on the outline)
ADE-131	ADE-53 Planning	1	Write a short summary of the planning process for this sprint (how, when and who divided tasks, any main reponsibilities, etc)
ADE-132	ADE-53 Summary of sprint	2	What we have learned, what we could do better in the upcoming sprint, status of the project and motivation of the group
ADE-133	ADE-53 Add documents to final report	1	Copy-pasta, proof-read and edit the design for the final report
ADE-134	Implement the rules		
ADE-135	ADE-134 research about drag-and- drop	2	
ADE-136	ADE-134 implement with drag and drop	10	
ADE-138	Save the ad as html5		As a user I can save the ad as html5
ADE-139	Save the ad as pdf		As a user I want to save the ad as pdf.
ADE-140	ADE-55 HTML5	1.5	Describe HTML5 and how we use it
ADE-141	ADE-55 Javascript & AngularJS	1.5	Describe Javascript, AngularJS and how we use it and utilize the framework's functionality in the application
ADE-142	ADE-55 Karma & Jasmine	1.5	Describe these frameworks and how they apply a testing environment to Javascript
ADE-143	ADE-55 CSS3 & Bootstrap library	1.5	Describe how CSS works and how we utilize the bootstrap library for our pretty, pretty design to-be
ADE-144	ADE-55 Bitbucket & Github	1.5	Describe Github functionality and how Bitbucket adds to that functionality
ADE-145	ADE-55 JIRA	1.5	Describe Jira and how we use it in relation to this project. What are the gains?

ADE-146	ADE-55 Google Drive	1.5	Describe the functionality og Google Drive shortly and how we utilize it specifically for our project
ADE-147	ADE-55 Latex	1.5	Describe Latex and how we utilize it in our project
ADE-148	ADE-55 FluidUI	1.5	Describe FluidUI and how it helped us improve understanding between us and the customer.
ADE-149	ADE-55 Netbeans	1.5	Describe Netbeans and what tools it offers (remember to mention Git functionality), and how these tools helps us realize our application
ADE-150	Rules for available fields to add		
ADE-151	ADE-150 implement the adding of components so that it can be picked from a list	3	what are the necessary components and their characteristics? implement it: - add components at correct position - let the user choose from a list of components to add
ADE-152	ADE-150 add it to the json file	1	
ADE-153	ADE-150 add it to the loading	1	
ADE-154	ADE-150 add it to the saving	1	
ADE-155	ADE-34 consider available fields	3	add the hidden component to the list of available fields without destroying its characteristics
ADE-156	ADE-46 review the entire pre-delivery	5	
ADE-157	ADE-46 edit the latex document	2	
ADE-158	ADE-134 rethink the rules (double ad)	1	
ADE-159	Save the ad as an image		As a user I want to store the final ad as an image.
ADE-160	ADE-138 write parser for get the components into a single html that looks like the preview	10	

ADE-161	ADE-138 research about the best way to do that	2	
ADE-164	ADE-139 research what is the best way to do it	2	
ADE-165	ADE-139 implement the saving as pdf	10	
ADE-166	ADE-159 research what is the best way to do it	2	
ADE-167	ADE-159 implement the saving as an image	8	
ADE-168	ADE-3 create example test	4	
ADE-169	Beautify the editor		
ADE-170	ADE-169 entire container	1	
ADE-171	ADE-169 tool bar	1	
ADE-172	ADE-169 editor container	1	
ADE-173	ADE-169 preview container	1	
ADE-174	ADE-169 header bar (logo...)	1	
ADE-175	ADE-169 Button design	1	
ADE-176	ADE-169 text design	1	
ADE-177	ADE-169 create group logo	1	
ADE-178	ADE-169 css for individual edit components	1	
ADE-179	ADE-169 right-click menu	1	
ADE-180	ADE-134 research about considering blocks in the drag and drop	4	

ADE-181	Customer and Adviser Contact	15	
ADE-182	ADE-150 datastructure in javascript and json	3	also ensure that the new component is added at the correct position
ADE-183	Get familiar with modeling (UML) in javascript		
ADE-184	ADE-183 research about uml in javascript	3	
ADE-185	ADE-183 start with modeling the editor	2	
ADE-186	Review Sprint 1 document	3	
ADE-187	ADE-53 Review Sprint 2 document	4	
ADE-188	ADE-55 Review 5 tools	1.5	Jira, Bitbucket & Github, Google Drive, Latex, FluidUI
ADE-189	ADE-55 review 5 tools	1.5	HTML5, Javascript & AngularJS, Karma & Jasmine, CSS3 & Bootstrap library, Netbeans

Table 8.4: Tasks and subtasks assigned to this sprint, the corresponding estimated realization effort, and an optional description of the task

Planned meetings for this sprint

- Adviser
  - 21.10.2014
  - 28.10.2014
- Customer
  - 28.10.2014
- Retrospective
  - 28.10.2014

8.3.3 Solutions to sprint backlog items

ADE-3 Explore testing

During this sprint we were able to successfully test controllers in our code using the Karma and Jasmine testing framework. The test master had created guidelines for installing the frameworks, and the tests currently implemented provide guidelines to writing new tests. Because of

our progress with testing and the guidelines provided by the test master, the other team members were able to explore the testing environment as well. During the next sprint we would cover more of the code with tests.

ADE-33 Textfields

One subtask of this task remained from last sprint, *ADE-103 Implement different kinds of text fields*. This was done by adding a description field to each component and displaying it beside the component in the editor window, as well as adding the right HTML to the two directives that display the textfield components.

ADE-34 Delete fields

This entire task was moved from sprint 2 to this sprint. Now it was possible to hide components required by the template by clicking the button ''Deaktiver" next to the component. This would in effect remove the component from the preview, but it would still be available in the editor area, only not editable. To activate the component again you just click the button called ''Aktiver" where the ''Deaktiver" button used to be. The components added by the client could only be deleted. When implementing this functionality we had to refactor the drag and drop functionality to make it compatible with hidden component.

ADE-46 Pre-delivery of final report

Some of the content for the pre-delivery report was already finished before this sprint, while several of the project members participated on finishing the rest. When all the content was ready it was transferred to LaTeX and compiled, and the quality assurance responsible was assigned to proof-read the document before delivery.

ADE-134 Implement the rules

After researching drag and drop functionality in AngularJS, a suitable library called ''angular-drag-and-drop-lists" implementing this service was found. Still a lot of work was required to make it enforce the rules correctly with this new functionality. As there are rules restricting components from being put at certain positions, we needed to prevent the user from dropping a component where it was not allowed to be. After this was finished, we removed the buttons for moving components up and down. Later we considered the rules for the double death notice, and to implement these rules effectively, some changes were made to the data structures of our application. Lastly we did some research about how to move blocks of components, as some components may be connected by rules. We began implementing this, but we would discuss this feature with the customer at the next meeting.

ADE-138 Save the advertisement as HTML5

Due to custom labels and the use of AngularJS directives for dynamically generating the HTML web page we could not simply replicate our code as it was on another web page. To save the advertisement as a HTML file, we first planned to write a generator reading information from our JavaScript data arrays and writing an HTML file. Later we suggested an alternative solution for the customer: To make a canvas object into an HTML file. With HTML5, a new feature called canvas was introduced, with the possibility to both draw and store graphics displayed on the web page [12]. This suggestion, for all intents and purposes, turned out to satisfy the requirements just as well. This way we could extract the content as a canvas element using a third-party library and store it in an HTML file, to be presented as a canvas again. We encountered problems while trying to figure out how to send the HTML file to the customer without testing it with a server. We managed to generate and download a default file (with no file type), containing an HTML string that represents the advertisement. If you manually changed the file name to end with ''.html", the downloaded file would work perfectly. This was not an ideal solution, but the HTML string could be used to implement a better solution later on, either by us or the customer.

ADE-139 Save the advertisement as PDF

By utilizing a library called ''jspdf" it was easier than expected to create PDFs locally: Using functions in this library and the canvas of the advertisement as basis for the PDF, we spent less time than estimated finishing this task. This library was made so that it was easy to make a PDF blob with different attributes added to it. In our case, since we parse the HTML code in the preview side of the editor into a canvas, all that was needed to make the PDF was to inject the image generated from *ADE-159* as an image. As of then the specifications of the image in the PDF was wrong, but we discussed this with the customer. As soon as we knew the image dimensions, resolution, and padding we were able to finish the PDF creation.

ADE-150 Rules for available fields to add

We needed to prevent the user from adding components not allowed by the template, so that only the components allowed by the template were displayed in the list of components. When a component was added, it was inserted at the first viable position from the bottom of the death notice. In order to implement this functionality some of the existing rules needed to be refactored as well. Lastly we added this option to the template rules, and adapted the loading and saving of advertisements to the new functionality.

ADE-159 Save the advertisement as an image

To make an image we had to find a way to grab the HTML on the preview side and convert it to some kind of image. To accomplish this we found a library called ''html2canvas" that parsed HTML code and took the HTML code and by parsing it, adds it to an HTML5 canvas element

(see task *ADE-138 Save the advertisement as HTML5* for more information about canvas). The canvas element contained an inbuilt method called ''toDataURL" which converts the data to an URL string that can be viewed in any browser. By creating a canvas element using the preview content we could save the canvas directly to an image file by built-in JavaScript functions.

ADE-169 Beautify the editor

This task included styling all the finished content on the web page, and creating logos. By using the Bootstrap CSS library we could do a great part of the styling without editing the CSS files, and thus saved a lot of time both researching and writing. By referencing elements in the HTML code to classes and identifiers in the CSS library, the styling was applied to the web page content. After making some alterations to adapt the styling to our needs, we were finished styling the content implemented thus far.

ADE-183 Get familiar with modeling (UML) in JavaScript

This task was not finished in this sprint. To be transferred to next sprint.

ADE-189 Review 5 tools

The review of the tools and technologies documentation was not finished during this sprint. The remaining tasks were transferred to the next sprint.

8.3.4 Burndown chart

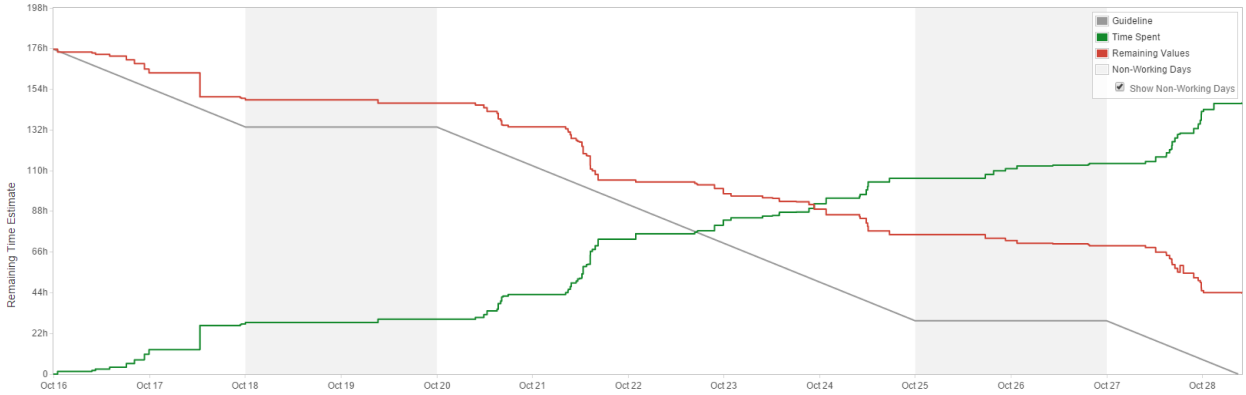


Figure 8.3: Correlation between work hours estimated and work hours done in sprint 3

For a general explanation of the burndown chart, see subsection 8.1.4.

As shown by the graph, some tasks were underestimated this sprint, and the time spent on the tasks did not correspond to the workload committed during the planning.

8.3.5 Customer feedback

Because only one of the customer representatives had the opportunity to attend the sprint meeting, we only had feedback from him. In total, the customer was impressed with our progress and how the application looked. Our application styling only worked on a specific range of screen resolutions, but this range satisfied the customer's needs. He was pleased with the drag and drop functionality to move fields around, and also how you could now add components by description from a list of allowed components. When moving a field to an illegal position the field would flickers, but the customer noted this was acceptable behaviour. In the future he would want us to make the mouse pointer dynamically give feedback as to where you are allowed to move the field.

As of now we were able to generate PDFs from the advertisement data, but the dimensions and resolution needed to be changed according to the customer's requirements. He provided a copy of the image generated so we could view the specifications needed. We were also able to generate the HTML file for the advertisement locally, but after discussing the matter we decided together with the customer that we were going to leave the HTML file generation to the servers.

Because of our drag and drop functionality we had to consider new enforcement of rules, and as such we discovered a scenario where it would be nice to introduce a new rule not explicitly stated by the customer before. Other than this we had implemented all the rules, and the customer was satisfied with how they prevent the client from reaching illegal states.

8.3.6 Sprint evaluation

Introduction

At the end of the sprint we had a retrospective meeting, discussing our experiences from this sprint. This evaluation was compiled from the meeting minutes, and contains our expectations for the retrospective meeting, important events, discussion about the outcomes of the sprint, achieved and prospective improvements, and conclusion to the meeting. We have changed one element from the usual setup: Instead of listing important events together as a group, we brainstormed events individually and categorized them collectively afterwards. Due to illness Pernille was not able to attend in person, but she attended by video call.

Expectations for the meeting

In turn each team member stated their expectations for the meeting, and we compiled this common list of expectations

- Discover any disagreements or issues in our group dynamic
- Sum up the results of the sprint
- Find points of improvement

- Get a clear picture of the end of the project
- Get the push we need to finish the project, and to do it as friends
- Find out what we did better in this sprint
- Find out how to become even more productive

Important events

We brainstormed the events individually, and discussed and categorized the events together afterwards. This list contains our combined lists of events, and may at some points be contradictory as a result of different experiences. The list with categorizations is found in appendix D.3

- What worked well?
  - The tasks were easier than expected
  - We had a very productive Tuesday
  - Estimations were spot on for the most part
  - The customer and the adviser seem satisfied
  - Testing
  - PDF-generation
  - Documentation
  - Communication
  - Meetings
  - Design
  - Good cooperation
  - Inclusion of everyone in the documentation
  - Arrived on time and fit for work
  - The breaks were more defined and effective
  - Less joking around, i.e. better focus
  - Excellent results in all areas
  - Everyone ready with relevant information when needed in meetings
  - Everyone works on what they are good at
  - Got tests running
- What did not work well?
  - Lack of backend
  - Late customer feedback due to illness and vacation
  - Late start on HTML file generation
  - Bad temporary solution on the server side for HTML file generation
  - No social gathering
  - Hobbies interfered
  - Too much other school work
  - Finished tasks others were dependent on late
  - People not finishing their tasks
  - People not showing up on time
- What should we start doing?
  - More testing
  - Focus on documentation
  - Design
  - Finish programming
  - Presentation
  - Finish work that others depend on earlier
  - Assign the tasks more carefully

Discussion

Disappointments

We were disappointed we did not have any social gatherings this sprint. We had planned to do a sushi night, but due to work both related and unrelated to this project several members could not attend, and therefore we postponed it.

Before one of the adviser meetings some members went to have a break and did not come back on time, and as a result we had to start the meeting without the whole group. We had neglected doing all of our daily standups, and some members missed this as part of our group dynamics. Although we had resumed our routine with daily standups before this meeting, we agreed to do this as often as possible in the future.

Upturns

One of the members brought cake, which was surprisingly good. We were happy that the testing was working, and that the work estimations were very good this sprint. It was uplifting and motivational to get praise from both the adviser and the customer in the meetings before this retrospective meeting.

Surprises

Some were surprised that the cake was good. Otherwise we were surprised that the estimations were so accurate, and that generating PDFs was easier than expected. The last retrospective meeting we noted we were uncomfortable about something the adviser said, and we were surprised that the first thing the adviser wanted to talk about the next meeting was this issue. He assured us this was nothing to be concerned about.

Expected

We expected our group dynamics to be better than the previous sprint, and we agreed that we had accomplished this.

Future

We discussed our points of improvements, and which points from the previous meeting we had achieved during this sprint. Our achieved points of improvement are listed here:

- Everyone should get involved in coding
- Everyone should get involved in documentation
- Have one person controlling the discussion
- The roles, or areas of responsibility
- Being organized

- Define beginning and end of breaks
- Bring earphones (for individual work)
- Add tasks to review all documentation
- Update JIRA before the end of the sprint
- Mondays 1200-1330: Reserved for preparing documents to meetings on Tuesday

Points we partially improved on, and want to continue improving:

- Forget logging work in the phase log
- Stop joking around in work meeting and early morning work
- Stop talking at the same time

This is our new list of improvements, including the points we want to continue improving:

- Start doing
  - Add UML tasks for our Master of Models
  - Prioritize testing
  - Daily standups
  - Finish work that others depend on early
- Stop doing
  - Joking around in work meetings
  - Talk at the same time
  - Forget logging work in the phase log
  - Coming in after time, but before late limit
- Continue doing
  - Controlled discussions
  - Roles and areas of responsibility
  - Being organized
  - Social gatherings
  - Define beginning and end of breaks
  - Everyone should get involved in documentation
  - Add tasks to review all documentation
  - Bring earphones (for individual work)
  - Update Jira before the end of the sprint
  - Mondays 1200-1330: Reserved for preparing documents to meetings on Tuesday

Conclusion to meeting

We all felt like we had improved a lot as a team since the first sprint, and we were happy with our group. It was still an issue that people joked around while working, but we think this is due to our friendly and enjoyable work environment. To get a social outlet we would try to have a social gathering the next week, and hopefully this would reduce our need to socialize during work. Concerning our individual organization there was room for improvement, but we agreed that if we could not finish our work, we should at least lessen the impact on the rest of the team. Our punctuality has been a recurring problem, with a bigger impact this sprint, and we want to improve this for the rest of the project.

8.3.7 Summary

During this sprint we implemented the remaining rules required by the customer. We also implemented drag and drop functionality for moving the components in the editor area, which had a big impact on the functionality for enforcing rules. As this task had a very uncertain scope we were worried we would estimate the work effort poorly, but it turned out to be a good estimation. In order to move components bound together by relational rules, we had to consider implementing containers to group components. Together with the customer we decided this would be nice to have, and that we should implement it during the next sprint. To prevent the user from trying to add components not permitted by the rules, we made the editor hide components not permitted from the list of components to add.

Another priority this sprint was generating the image, PDF and HTML files representing the approved advertisement. To generate the image we utilized an element introduced in HTML5, the *canvas* element. Using a JavaScript library it was possible to extract an area of the web page and store it as a canvas, and in turn store this canvas locally as an image file. The PDF generation turned out to be easier than expected by utilizing the canvas as basis for generating the PDF, while the HTML file generation we found was impractical to implement on the client side. Together with the customer we agreed to send the data files for the advertisement to the customer's servers to generate the HTML file there. We had received the specifications for the image and PDF from the customer in order to perfect the output during the next sprint.

Spending additional resources on testing this sprint we finally made successful tests, well before the end of the sprint. Because of the timely success the whole team were able to explore testing as planned, and in the upcoming sprints we would write tests for the already finished code. Regarding the styling of the editor, we had spent a lot of resources this sprint finishing it for all the available functionality. This has made the application more user friendly, and notably more appealing. Our work effort estimations this sprint were good and therefore most of the tasks were finished, while the unfinished tasks would be transferred to the next sprint.

During our customer meeting the customer told us he was impressed with our progress, both during this sprint and in total during the project. He liked both the styling and the new functionality we presented to him. The customer was pleased with how we had solved most of the tasks, but he wanted us to change the mouse pointer when dragging components to give feedback to the user. During our retrospective meeting we agreed that we had improved as a team, but we could improve in regards to focus, punctuality, and personal organization.

8.4 Sprint 4

8.4.1 Planning

At the end of the previous sprint we had a customer meeting and a retrospective meeting in order to map our results from that sprint, and prepare for this next sprint. Because we soon would reach the end of the project period, we needed to consider our work strategy for the remaining time, including how long our sprints were going to be. We had approximately three weeks left, so in order to rapidly adapt to problems and changes, we decided to have three, week long sprints.

As we now knew how to write tests for our code, we decided to assign a good amount of resources to this during the sprint. Additionally , we wanted to implement blocks of components for moving components related by a rule, and finish the styling of the application. We would need to research handling of requests from the server, consider the integration of our application in the customer's current system, and facilitate possible expansions of the application for the customer.

As it was now the customer could only create new templates by writing the template file specifications manually, but we would make it a priority to create a tool to abstract this. The customer wanted to be able to add images as new symbols for the death notices, and would like the process to be as simple as possible. As our solution was right then, you could add new symbols by adding a couple of lines of code for each image, but we would try to load symbols from the image folder dynamically. The customer also wanted guidelines for the application, but together with them we decided to write guidelines for the system closer to the end of the project, when the functionality and the user interface would be fixed. To satisfy the requirements we needed to correct the scale and resolution of the image and PDF generated from the approved advertisement, and we planned to do it during this sprint.

8.4.2 Sprint backlog and estimation of realization effort

Key	**Summary**	**Original Estimate**	**Description**
ADE-37	Create templates P:3		As an IT department I can create new templates so that the customers can use the templates.
ADE-54	Write sprint 3 document		
ADE-137	Write planning document		

ADE-183	Get familiar with modeling (UML) in javascript		
ADE-184	ADE-183 research about uml in javascript	3	
ADE-185	ADE-183 start with modeling the editor	2	
ADE-190	Tools and technologies document		
ADE-191	Consider Blocks		
ADE-193	Styling		
ADE-194	post, get requests for image, html, pdf and json		
ADE-195	Test RuleCheckerService	4	
ADE-198	ADE-190 review 5 tools (see description)	1,5	HTML5, Javascript & AngularJS, Karma & Jasmine, CSS3 & Bootstrap library, Netbeans
ADE-199	ADE-190 put everything in latex	1,5	
ADE-200	Read through compendium	3	Do we miss something?
ADE-202	Presentation Outline	2	
ADE-203	Customer and Adviser Contact	8	
ADE-206	ADE-194 implement approve button functionality	2	
ADE-207	ADE-194 research	3	
ADE-208	ADE-194 setting up handling of getting a new template with name...	5	create a second page for that
ADE-209	ADE-194 sending the saved ad as json	2	

ADE-210	ADE-193 button placement	0,5	
ADE-211	ADE-193 styling of inactivated components	1	gray overlay, red border
ADE-212	ADE-54 Sprint/backlog specification	0,5	List tasks and priority together with the estimated work effort, and list our planned meetings for this sprint
ADE-213	ADE-54 Short description of solutions to sprint backlog	3,5	Just give the overall picture and describe the solution to the stories/main tasks
ADE-214	ADE-54 Customer feedback	1	Gather all customer feedback we received and put it in full sentences
ADE-215	ADE-54 Sprint evaluation	2,5	Use what we learned from the retrospective meeting to put together an evaluation of the sprint (see the temp project report file if you need a suggestion on the outline)
ADE-216	ADE-54 Planning	1	Write a short summary of the planning process for this sprint (how, when and who divided tasks, any main reponsibilities, etc)
ADE-217	ADE-54 Summary of sprint	2	What we have learned, what we could do better in the upcoming sprint, status of the project and motivation of the group
ADE-218	ADE-54 Add documents to final report	1,5	
ADE-219	ADE-54 Review sprint 3 document	2,5	
ADE-220	ADE-54 Testing document	1	Describe for each test what parts of the code they test
ADE-221	representation that a component is drag able i.e. if it is not locked		
ADE-222	ADE-221 make locked fields not drag able	0,167	

ADE-223	ADE-221 add locked variable and change it when creating isLocked and isBottom rules	0,167	
ADE-224	ADE-221 change the appearance of the cursor when it is moved over not locked components to indicate that they are drag able	0,5	
ADE-225	ADE-221 create a css class that is applied to the components depending on locked or not	0,333	
ADE-226	ADE-37 let the editor know who is the user	1	
ADE-227	ADE-37 show a list with the rules that exist	1	
ADE-229	ADE-37 show the html components that are necessary for creating the rules and adjust the selections so that it is clear which component is meant	0,5	
ADE-230	ADE-37 styling of the toolbar and the rule list	4	
ADE-231	ADE-191 consider it in drag'n drop	3	
ADE-232	ADE-191 consider it in saving	1	
ADE-233	ADE-191 make a block visible	4	

ADE-234	ADE-137 Check project plan document and complete it	0,5	Use project plan, roles and goals document as basis to organize and review it for the report
ADE-235	ADE-137 Write group organization document	0,333	Use what we already have and organize and review it for the report
ADE-236	ADE-191 consider it in ordering	1	
ADE-237	ADE-137 Write a document for templates	0,5	Reference screenshots placed in the appendix
ADE-238	ADE-193 margins of the images	0,5	
ADE-239	ADE-193 margins on top and bottom of the ad	0,5	
ADE-240	ADE-193 clean up css	3	
ADE-241	ADE-137 Reference version control tools	0,25	Add references to Tools & tech to relevant tools
ADE-242	ADE-193 change the line	0,5	
ADE-243	ADE-137 Sum up our Quality assurance	2	See appendix A7 in the compendium
ADE-249	ADE-137 List partners	0,25	(For appendix) List customer's and adviser's name and contact info, optionally anything other that may be useful. Most of it should be found in the compendium in assignment description or adviser description
ADE-250	ADE-137 Review risks	0,333	Use risk document as basis
ADE-252	ADE-137 add it to the report	3	
ADE-253	Verify html files	1	
ADE-254	ADE-137 coding conventions	0,5	Review coding conventions (add reference to AngularJS, CSS, HTML conventions), and add it to the document in an organized way
ADE-255	ADE-194 set up basic server	3	

ADE-279	ADE-194 get familiar with REST API	2	
ADE-324	ADE-194 add a flag into the post that says if it is a template or not	1	
ADE-325	ADE-194 get the user type as a flag from the server	1	

Table 8.5: Tasks and subtasks assigned to this sprint, the corresponding estimated realization effort, and an optional description of the task

Planned meetings for this sprint

- Adviser
  - 04.11.2014
- Customer
  - 04.11.2014
- Retrospective
  - 04.11.2014

8.4.3 Solutions to sprint backlog items

ADE-37 Create templates

As only the administrators of the application should be able to create templates, we needed to control the user's role, i.e. IT administrator or funeral agent. The role was going to be received as a flag in the request from the server, and set as a local variable we could check later. If the user was an administrator, we showed the rule creation tool. For each component you could add a rule, and depending on the rule, create a relation to another component or position.

ADE-183 Get familiar with modeling (UML) in JavaScript

This task was transferred from the last sprint. We researched how to document our JavaScript code, and decided we should do it using models from the UML (Unified Modelling Language, see [32]). We considered what models were relevant for the report along the way, but we created some use case diagrams during this sprint. We would iterate on this task during the next sprints.

ADE-191 Consider blocks

To move components implicitly connected by a rule, we needed to rethink the data structure for

the array of components. The solution was to wrap all components in containers, where connected components share a container. To make this work we needed to edit the drag and drop, saving, ordering, hiding, and deletion functionalities. This was a hard task to estimate, and as a result we underestimated it by six hours.

ADE-193 Styling

Not all the elements of the application were styled last sprint, and with new functionality there were several subtasks for styling and correction: Button placement, styling of inactive components, margins of the images, margins on top and bottom of the advertisement, and editing of the lines in the preview. As before we used the Bootstrap library, and adapted the styling to fit our specific application. The task of cleaning up the CSS code was transferred to the next sprint.

ADE-194 Post and get requests for image, HTML, PDF and JSON

Except for setting up a server environment to test our solution with, the subtasks of this task were not finished. These would be transferred to a later sprint.

ADE-195 Test RuleCheckerService

This task was not finished during this sprint, and will be transferred to the next sprint.

ADE-221 Representation that a component is draggable, i.e. if it is not locked

In order to show the user if a component was draggable or not, we made the mouse pointer indicate draggability when hovering over a movable component. Also, we made it impossible to drag a locked component (before it was possible to drag it, only not drop it in a new position). To do this we added a new variable to the component object. For the user to easily identify if a component is locked or unlocked, we have applied different stylings for them.

ADE-253 Verify HTML files

To ensure the validity of our HTML code we controlled it using an online tool called Validator.nu [33]. We were unable to run the test without any warnings, because of the directives provided by angular and the custom directives that angular made it possible for us to define ourselves. But other than the warnings concerning angular elements, our HTML code was valid.

8.4.4 Burndown chart

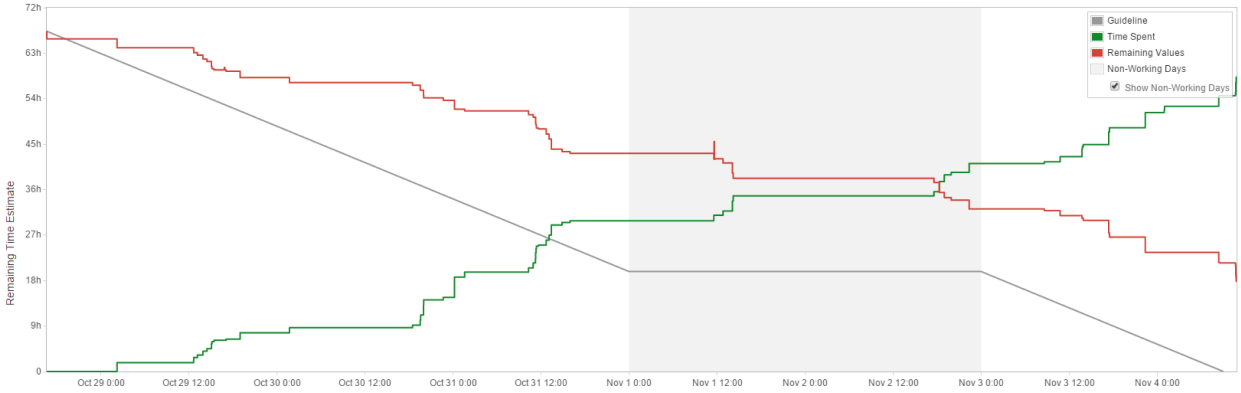


Figure 8.4: Correlation between work hours estimated and work hours done in sprint 4

For a general explanation of the burndown chart, see subsection 8.1.4.

As shown by the graph, several tasks were underestimated this sprint, and the time spent on the tasks did not correspond to the workload committed to during the planning.

8.4.5 Customer feedback

In the customer meeting at the end of the sprint we presented our progress and results, which the customers were happy with. They noted that they would like an extra indicator if a component was locked or unlocked, but other than that they were satisfied with the look and feel of the application, and did not want us to change it otherwise.

Concerning future sprints, they required correct image and PDF generation to be the main priority. As we did not write the tests we had planned, they noted that high code coverage of tests will not be critical, but it is still something they would like to have. We asked them about the language of the user interface, as it is in English and they requested it to be in Norwegian, but they told us this was not a high priority to change. They would like us to provide a guide to setup dynamic image loading of symbols for their server side implementation, which we will take into consideration when planning the next sprint.

8.4.6 Sprint evaluation

Introduction

At the end of the sprint we had a retrospective meeting, discussing our experiences from this sprint. This evaluation was compiled from the meeting minutes, and contains our expectations for the retrospective meeting, important events, discussion about the outcomes of the sprint, achieved and prospective improvements, and conclusion to the meeting.

Expectations from meeting

In turn each team member stated their expectations for the meeting, and we compiled this common list of expectations:

- Reflect on how this sprint went, with special regards to it being shorter than usual
- Gain better sprint planning knowledge for the next sprint
- Uncover what our problems were this sprint
- Figure out what was difficult in this sprint, since we seemed to be improving until this last sprint

Important events

We brainstormed the events individually, and discussed and categorized the events together afterwards. This list contain our combined lists of events, and may at some points be contradictory as a result of different experiences. The list with categorizations is found in appendix D.4

- What worked well?
  - Sushi/social gathering
  - Adviser and customer seem happy
  - Verify HTML (task)
  - Documentation
  - Good communication with both adviser and customer
  - Group roles
  - Adapting to implement unplanned features
  - Improved skills
  - Good reviewing of documents
- What did not work well?
  - We had to setup a server
  - Estimations were worse than last time
  - Not enough time for this project due to other work
  - Categorize symbols
  - Unclear focus/priority on features
  - Not very precise feedback from customer
  - Difficulty implementing blocks
  - Received tasks too late
  - Spent too much time on this project
  - Did not finish all tasks
  - Illness
- What should we start doing?
  - Testing
  - More documentation
  - Less programming
  - Stop adding features
  - Enjoy life more, i.e. lessen workload
  - Create the presentation
  - Include everyone in writing and reviewing documentation
  - Refactor styling of the report
  - Improve personal organizing
  - Not commit to too much work during sprint planning
  - List all the remaining tasks for the project
  - Stop interrupting each other

Discussion

Disappointments

The members that did not finish their tasks were disappointed about this.

Upturns

We were happy about our group and that we still get along so well. The social gathering was very fun.

Surprises

Some of us was surprised they did not get feedback that was useful for their tasks.

Expected

Some of us expected the feedback from the customer to be of little help to their tasks. Also, some expected that we had too much work planned, rightfully, and that we would spend too much time planning compared to work directly related to the tasks at hand.

Future

We discussed our points of improvement, and planned improvements from the previous meeting we had achieved during this sprint. These are as follows:

- Add UML tasks for our Master of Models
- Stop coming in after time, but before late limit
- Bring earphones for focused individual work

This is our new list of improvements, including the elements we want to continue improving:

- Start doing
  - Prioritize testing
  - Daily standups
  - Finish work that others depend on early
  - Update JIRA regularly
  - Get an overview of all tasks that have to be done in the scope of this project
- Stop doing
  - Joking around in work meetings
  - Talk at the same time
  - Forget logging work in the phase log
- Continue doing
  - Controlled discussions
  - Roles and areas of responsibility
  - Being organized
  - Social gatherings
  - Define beginning and end of breaks
  - Everyone should get involved in documentation
  - Add tasks to review all documentation
  - Mondays 1200-1330: Reserved for preparing documents to meetings on Tuesday

Conclusion to meeting

Although this sprint was short, it was an eventful sprint with mixed experiences. We felt, for the first time, we managed a sprint worse than the previous one. Some of our earlier improvements as a team had regressed, while we made fewer new accomplishments. Still we agreed we did better than the first sprint, and all things considered, we felt good about our group dynamics. While this sprint went bad, we would use the experience we had gained and try to improve in the next sprint.

8.4.7 Summary

After this sprint we had mixed experiences with the shorter sprint format. We did a poor job estimating the tasks, and several of the team members accepted more work than they were able to finish. This was somewhat a consequence of other work and more project management, but nothing that could not be planned for, and needed to be considered during the next sprint planning. Testing was one of the tasks that were not finished, although we decided this was a prioritized task in the beginning of the sprint. The customer was informed about this, and although they would like testing to be a part of the product delivery, they asserted that it is not critical. We will use our experience from this sprint to improve in the next one.

We adapted very well to new features we wanted to implement during the sprint. Administrators now had access to functionality for creating templates in the editor, which would relieve the customer of a lot of manual work. As we implemented blocks for containing several components, all the functionality for editing advertisements was now completed. We researched dynamic addition of new symbols, but found this was not possible to do from the client side application. The customer had requested a guide for setting this up on the server side instead, if we have the time for it.

Although we were not able to correct the image and PDF generation according to the requirements during this sprint, the customer was very satisfied with our progress and results. Besides one visual element they want us to keep the user interface as it is.

8.5 Sprint 5

8.5.1 Planning

Although the last sprint went poorly in regard to planning and organization, this sprint was a week long as well, because there were only two weeks left and it facilitated the awareness of the deadlines. We applied our experiences from the last sprint, and planned effectively while ensuring that no team members were assigned more work than their schedule allowed. As the project was nearing the end, we decided to make a general outline for the next and last sprint

as well, in order to get an overview of the remaining work. This provided a guideline for us to assess how much work we needed to do during this sprint in order to finish on time. In addition, we were able to decide which tasks were realizable and which were not.

Before creating the sprint backlog we had a meeting with the customer in order to get feedback on our progress, and request priority for the backlog. For the customer, the most important task to get done this period was to correct the dimension and resolution of the generated image and PDF files. According to their feedback, we decided to add an extra indicators to the fields to show if a component was locked, so that the user did not need to move the mouse to the field to see if it is movable. Additionally, we planned to improve the styling and design of the template creation tool during this sprint.

Together with the customer, we assessed that other tasks were more important than testing the API with a mock server. Therefore, we planned to create a more thorough documentation for it instead. Although the customer said they could do the translation of the user interface from English to Norwegian themselves, we found resources to do it during this sprint. The guidelines for the customer would be written in the next sprint when all the functionality was implemented.

Encouraged by the customer, we decided to allocate more resources for documentation and preparation of the presentation for the project during these last sprints. This affected the amount of resources dedicated to programming, but none of the required functionalities were neglected.

8.5.2 Sprint backlog and estimation of realization effort

Key	**Summary**	**Original Estimate**	**Description**
ADE-37	Create templates P:3		As an IT department I can create new templates so that the customers can use the templates.
ADE-54	Write sprint 3 document		
ADE-56	Write sprint 4 document		
ADE-137	Write planning document		
ADE-193	Styling		
ADE-195	Test RuleCheckerService	4	

ADE-196	Test ComponentsService	9.5	
ADE-210	ADE-193 button placement	0.5	
ADE-211	ADE-193 styling of inactivated components	1	gray overlay, red border
ADE-212	ADE-54 Sprint/backlog specification	0.5	List tasks and priority together with the estimated work effort, and list our planned meetings for this sprint
ADE-213	ADE-54 Short description of solutions to sprint backlog	3.5	Just give the overall picture and describe the solution to the stories/main tasks
ADE-214	ADE-54 Customer feedback	1	Gather all customer feedback we received and put it in full sentences
ADE-215	ADE-54 Sprint evaluation	2.5	Use what we learned from the retrospective meeting to put together an evaluation of the sprint (see the temp project report file if you need a suggestion on the outline)
ADE-216	ADE-54 Planning	1	Write a short summary of the planning process for this sprint (how, when and who divided tasks, any main reponsibilities, etc)
ADE-217	ADE-54 Summary of sprint	2	What we have learned, what we could do better in the upcoming sprint, status of the project and motivation of the group
ADE-218	ADE-54 Add documents to final report	1.5	
ADE-219	ADE-54 Review sprint 3 document	2.5	
ADE-220	ADE-54 Testing document	1	Describe for each test what parts of the code they test
ADE-226	ADE-37 let the editor know who is the user	1	
ADE-227	ADE-37 show a list with the rules that exist	1	

ADE-229	ADE-37 show the html components that are necessary for creating the rules and adjust the selections so that it is clear which component is meant	0.5	
ADE-230	ADE-37 styling of the toolbar and the rule list	4	
ADE-234	ADE-137 Check project plan document and complete it	0.5	Use project plan, roles and goals document as basis to organize and review it for the report
ADE-235	ADE-137 Write group organization document	0.333	Use what we already have and organize and review it for the report
ADE-237	ADE-137 Write a document for templates	0.5	Reference screenshots placed in the appendix
ADE-238	ADE-193 margins of the images	0.5	
ADE-239	ADE-193 margins on top and bottom of the ad	0.5	
ADE-240	ADE-193 clean up css	3	
ADE-241	ADE-137 Reference version control tools	0.25	Add references to Tools & tech to relevant tools
ADE-242	ADE-193 change the line	0.5	
ADE-243	ADE-137 Sum up our Quality assurance	2	See appendix A7 in the compendium
ADE-249	ADE-137 List partners	0.25	(For appendix) List customer's and adviser's name and contact info, optionally anything other that may be useful. Most of it should be found in the compendium in assignment description or adviser description
ADE-250	ADE-137 Review risks	0.333	Use risk document as basis
ADE-252	ADE-137 add it to the report	3	

ADE-254	ADE-137 coding conventions	0.5	Review coding conventions (add reference to AngularJS, CSS, HTML conventions), and add it to the document in an organized way
ADE-256	Write remaining tools		Google Calendar, UML tool (Knut), figure for Git (Linda), Server (mongodb, nodejs, crest)
ADE-257	Fix the rightclick when the template contains a block	1.5	
ADE-258	put all strings in one service		
ADE-259	indicate dragablility	1	
ADE-261	fix the appearance of the image and the pdf		
ADE-262	ADE-56 Sprint/backlog specification	0.25	
ADE-264	ADE-56 Short description of solutions to sprint backlog	2	
ADE-266	ADE-56 Customer feedback	0.5	
ADE-267	ADE-56 Sprint evaluation	1	
ADE-268	ADE-56 Planning	0.5	
ADE-269	ADE-56 Summary of sprint	1	
ADE-270	ADE-56 Add documents to final report	0.75	
ADE-271	ADE-258 translate the strings to Norwegian	0.333	
ADE-272	ADE-56 Review sprint 4 documents	1	
ADE-273	Presentation		
ADE-275	Make and improve figures for the report		
ADE-277	Test smaller components	18	

ADE-278	Add table of figure and tables	0.5	
ADE-280	ADE-261 image	3	
ADE-281	ADE-261 pdf	1	
ADE-315	ADE-256 Google Calendar	0.5	
ADE-316	ADE-256 Argo UML	0.5	
ADE-317	ADE-256 figure for branching system	0.333	
ADE-318	ADE-273 create editor demo Linda	2.5	
ADE-319	ADE-273 contact customer for old screenshots and the problems of its usage	0.5	
ADE-320	ADE-273 take screenshots of the old versions of our editor	2	
ADE-321	ADE-273 create slides for each sprint (concerning impediments mainly)	5	
ADE-322	ADE-275 fix stuff we do stuff they do	0.5	
ADE-323	ADE-275 what figures are necessary	3	
ADE-328	create uml diagrams for the architecture		
ADE-329	ADE-328 class diagram	6	
ADE-330	ADE-328 use cases	2.5	
ADE-331	ADE-328 sequence diagram	6	
ADE-332	customer and adviser contact	8	
ADE-333	ADE-258 create the service	2	

ADE-338	ADE-256 review	0.5	
ADE-348	Create rules only if the current state allows it i.e. rules evaluate to true	0.5	
ADE-349	ADE-256 Mailing List	0.333	
ADE-353	Rename ImageService to SymbolService	0.5	
ADE-378	ADE-273 finish presentation	5	
ADE-386	ADE-273 create editor demo Erik	2.5	
ADE-387	ADE-273 think about features to present in the editor demo	2.5	

Table 8.6: Tasks and subtasks assigned to this sprint, the corresponding estimated realization effort, and an optional description of the task

Planned meetings for this sprint

- Adviser
  - 11.11.2014
- Customer
  - 11.11.2014
- Retrospective
  - 11.11.2014

8.5.3 Solutions to sprint backlog items

ADE-37 Create templates

The subtask *ADE-230 Styling of the toolbar and the rule list* was transferred from the last sprint, while the rest of the subtasks were finished in sprint 4.

The styling of the rule creation toolbar was improved, and now flowed with the rest of the application. The list of rules created, and the preview of the advertisement were put in different tabs so the user could change to the most relevant view when creating rules.

ADE-193 Styling

The subtask *ADE-240 Clean up CSS* was transferred from the last sprint, but was not finished during this sprint. Thus it will be transferred to the next sprint.

ADE-257 Fix the right-click when the template contains a block

After implementing blocks for the components, a bug occurred when using the right-click menu in the components. More specifically, the right-click menu did not disappear after making a choice or clicking somewhere else in the editor. The cause was that the original data structure was renamed, and the old name was given to the new data structure implementing the blocks, while the reference in the right-click directive had not been changed accordingly. When changing the reference to point at the data structure without blocks, the right-click menu behaved as intended again.

ADE-258 Put all strings in one service

In order to translate the interface from English to Norwegian, instead of hardcoding the solution, a service translating the strings was made. This also made it possible for the customer to change all texts in the user interface from one location in the code, and provided the basis for a possible extension of the application to implement several languages.

ADE-259 Indicate draggability

The customer wanted an additional indicator to show the user if a component was locked before hovering over it with the mouse. The indicator was implemented as a small lock icon in all locked components. Only some changes to the CSS styling was required to implement this, as the CSS classes differentiating locked and unlocked components was already made.

ADE-261 Fix the appearance of the image and the PDF

The symbol shown in the preview was restricted from scaling in order to be saved with the correct dimensions and size for the image. When generating the PDF file, the width and height can be provided as arguments in millimeters to get the wanted dimensions of the output. To find these dimensions, the height and the width of the advertisement were fetched as a number of pixels from the canvas generated with the html2canvas library, which were translated to the required size in millimeters citehtml2canvas. The resolution of the generated advertisement was not corrected during this sprint, and would be transferred to the next sprint.

ADE-328 Create UML diagrams for the architecture

In order to improve the comprehension and illustrate the structure of the application for the report, models of our solution were made. Class diagrams and use case diagrams were made in this sprint, and sequence diagrams were created in the next sprint. The use case diagrams could be found in section C, while the class diagram task was reopened during sprint 6 and is not included here.

ADE-348 Create rules only if the current state allows it, i.e. rules evaluate to true
In order to prevent the user from creating rules that contradict the current state of the editor, a rule is validated before it gets added to the list of rules. If it validates to false, the user gets a notification that the rule could not be created. The method for validating the state was already implemented to check if a component could be moved to a given position, and was reused to solve this task.

8.5.4 Burndown graph

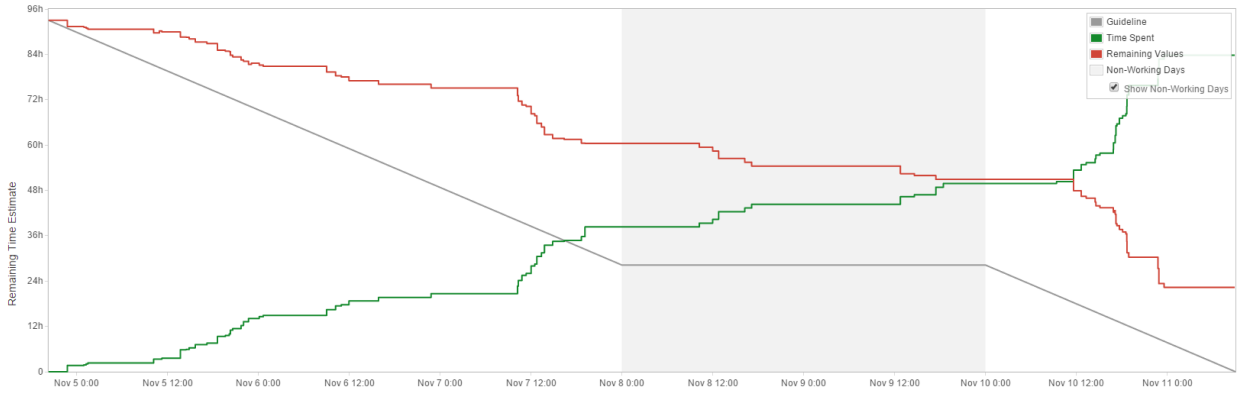


Figure 8.5: Correlation between work hours estimated and work hours done in sprint 5

For a general explanation of the burndown chart, see subsection 8.1.4.

As shown by the graph, several tasks were underestimated this sprint. Furthermore, the entire time spent on the tasks did not correspond to the workload committed to during the planning.

8.5.5 Customer feedback

The customer was happy that the editor produced images and PDF files with correct dimensions for the advertisement, but emphasized that the resolution of the advertisement should be our highest priority to correct. Additionally, they want us to create the template for the double death notice, as this is part of their current application. The changes made to the user interface, i.e. Norwegian translation, tabs for the rule creation mode, and lock icon, were satisfying. We suggested to write more tests and clean up the code, which was also appreciated by the customer.

8.5.6 Sprint evaluation

Introduction

At the end of the sprint we had a retrospective meeting, discussing our experiences from this

sprint. This evaluation was compiled from the meeting minutes, and contains our expectations for the retrospective meeting, important events, discussion about the outcomes of the sprint, achieved and prospective improvements, and conclusion to the meeting.

Important events

We brainstormed the events individually, and discussed and categorized the events together afterwards. This list contains our combined lists of events, and may at some points be contradictory as a result of different experiences. The list with categorizations is found in appendix D.5.

- What worked well?
  - Testing
  - Project report
  - Modelling of the application
  - Planning
  - Daily standups
  - Lot of work done
  - Presentation
  - Done with work early in the sprint
  - The feedback from the adviser on the report was clear and extensive
  - Helping each other
  - Teamwork
  - Progress of the project
- What did not work well?
  - Image generation more complex than expected
  - Too much time spent on planning, i.e. less time for other work
  - People overslept and came late to meetings
  - Failed at estimation and workload
  - Unrelated work interfering with the project work
  - A lot of work left to do
  - Mailing list not used enough
  - Task assignment
  - Preparation for meetings
- What should we start doing?
  - Go to sleep earlier
  - Stop adding tasks to backlog
  - Focus on finishing the project
  - Finish programming
  - Document technical aspects of the application
  - Do more of the sprint planning individually
  - Get the work done as early as possible
  - Work more for the rest of the project

Discussion

Disappointments

We were disappointed that we did not get to set up the server for testing the communication interface of our application. Since the project was nearing the end and we had limited time left, we were disheartened to see that some of the tasks planned for this sprint were not finished. Some team members were disappointed that the documents had not been reviewed as well as expected. The group was sad that the project would be over soon.

Upturns

The presence of the second adviser during the previous adviser meeting was helpful as we got a third opinion on our work, from a new point of view. We were happy that the feedback we received for the report was so detailed, and that the testing was going along well. We were delighted to see that all tasks other team members were dependent on were finished early in this sprint, so that there was enough time left to finish the dependent tasks. The customer showed understanding for the importance of the documentation and presentation, and encouraged us to assign more resources to this. It was also pleasing to get an early start on the presentation.

Surprises

The complexity of the image generation was surprising. In addition to being surprised at the detail of the feedback on the report, we were surprised about the differences in the advisers' feedback on our work.

Expected

The greater workload and focus on documentation needed for this sprint was expected by the team. The good start on the testing was predicted, as well as the progress of the documentation. We expected the feedback from the adviser, regarding the documentation, to point out the lack of technical explanation of the application, as for example the architecture document was missing.

Future

We discussed our points of improvement, and which planned improvements from the previous meeting we have achieved during this sprint. These are as follows:

- Use a scrum master for daily standups
- Prioritize Testing
- Roles
- Being organized
- Define beginning and end of breaks
- Get an overview of all tasks that have to be done in the scope of this project
- Documentation involvement
- Add review documentation

- Monday 12-13:30 look over documents that have to be handed in to the adviser
- Social gatherings

This is our new list of improvements, including the elements we want to continue improving:

- Start doing
  - Better reviewing of documents
- Stop doing
  - Joking around in work meetings
  - Talk at the same time
- Continue doing
  - Controlled discussions
  - Log work done in the phase log
  - Daily standups
  - Finish work that others depend on early
  - Update JIRA regularly

Conclusion to meeting

Concerning the planning and the teamwork, we managed this sprint much better than the last sprint, and we would try to benefit from this experience in the next sprint. Everyone seemed satisfied with the progress on the documentation and presentation, while the final functionalities of the application were more complex to implement than expected. With the extensive feedback from the advisers we had a guideline for the work to be done before the end of the project. Although we experienced some leisurely behavior regarding personal organization, we were confident about the outcome of the project.

8.5.7 Summary

The one week long sprint went much better this time, mostly due to better planning. One of the main points when planning this sprint was to not commit to more work than one could finish, and we finished most of the tasks. Planning took more time than expected, which limited the time available for other tasks in this sprint, but this may have been beneficial for our progress in the end. The group dynamics had improved, with more cooperation on finishing tasks, and our efficiency had improved as a result.

The difficulties with the PDF and image generation persisted, but other than that, the work on the application was progressing well, and the customer was happy with the functionality we had implemented. In our planning we considered all the remaining work for the project, and even with the tasks not finished in this sprint, we were positive we would finish successfully. The adviser provided extensive feedback on the project report, which was valuable for both planning the next sprint, and the end result of the documentation.

8.6 Sprint 6

8.6.1 Planning

This sprint was the last sprint of the project lasting until the project presentation. Most of the tasks for this sprint were already planned during the last sprint planning, but with the feedback from the adviser and the customer, and greater insight into the remaining work, more tasks were created.

Since we were getting close to the end of the project, this sprint focused on the documentation and presentation of the project. The customer encouraged us to prioritize this to meet the requirements for the project delivery, and this affected the resource allocation during this sprint. Some programming tasks still needed to be done, mainly the correct generation of PDF and image files. Some more tests were also thought to be written, and we planned to perform manual tests to verify that our solution satisfies the use cases. User guidelines for both the customer, as administrators of the application, and the users of the editor were scheduled for this sprint. The feedback from the adviser emphasized the lack of technical documentation of our application, which therefore was the main focus for the documentation in this sprint. We planned to review the entire report at the end of the project.

Tasks that supplied others got delivery deadlines to make sure all tasks could be finished during the sprint. Programming and modelling tasks had their deadline Friday evening, testing and documentation tasks Monday evening, while finishing the report had the end of the sprint as deadline.

8.6.2 Sprint backlog and estimation of realization effort

Key	Summary	Original Estimate	Description
ADE-28	Finish final report		
ADE-36	User guidelines for customer P:2		As a funeral agency I can read the user guidelines so that I know how the editor works.
ADE-57	Write architecture document		refer to architecture part in the user guidelines
ADE-61	Write sprint 5 document		
ADE-62	Write evaluation document		
ADE-63	Final report delivery		

ADE-193	Styling	2	
ADE-196	Test ComponentsService	9.5	
ADE-210	ADE-193 button placement	0.5	
ADE-211	ADE-193 styling of inactivated components	1	gray overlay, red border
ADE-238	ADE-193 margins of the images	0.5	
ADE-239	ADE-193 margins on top and bottom of the ad	0.5	
ADE-240	ADE-193 clean up css	3	
ADE-242	ADE-193 change the line	0.5	
ADE-261	fix the appearance of the image and the pdf		
ADE-263	ADE-36 install and run testing	0.5	
ADE-273	Presentation		
ADE-276	User guidelines for the user i.e. funeral agency		
ADE-277	Test smaller components	18	
ADE-280	ADE-261 image	3	
ADE-281	ADE-261 pdf	1	
ADE-282	ADE-36 describe the architecture using the uml diagrams	3	
ADE-283	ADE-276 loading templates and ads	1.5	
ADE-284	ADE-276 rearrange textfields	1.5	
ADE-285	ADE-276 add and delete / inactive text fields	1.5	

ADE-286	ADE-276 approve ad	1.5	
ADE-288	ADE-57 describe architecture using uml diagrams (MVC, patterns)	5	
ADE-291	ADE-61 Sprint/backlog specification	0.25	
ADE-292	ADE-61 Short description of solutions to sprint backlog	2	
ADE-293	ADE-61 Customer feedback	0.5	
ADE-294	ADE-61 Sprint evaluation	1	
ADE-295	ADE-61 Planning	0.5	
ADE-296	ADE-61 Summary of sprint	1	
ADE-297	ADE-61 Review sprint 5 document	1	
ADE-299	ADE-61 Add documents to final report	0.75	
ADE-301	Write Test document		
ADE-302	ADE-301 which unit tests do we have	4	
ADE-303	ADE-301 code coverage of unit tests	1	
ADE-304	ADE-301 describe manual tests in the report and add them to the appendix	1	
ADE-305	ADE-62 The internal process and results	0.75	1.The internal process and results: How have you worked together as a team? What have you done well? What have you not done so well? What would you have done differently? Conflicts that arose and how these were handled? Did you reach the project goals? What did you learn?

ADE-306	ADE-63 printing 2 times	1	
ADE-308	ADE-62 customer and the project task	0.75	2.The customer and the project task: How was the communication with the customer? How did you experience the project assignment?
ADE-309	ADE-62 The advisers	0.75	3.The advisers: How was communication with the advisers? Was the supervision good enough? How could the course be improved to next year?
ADE-310	ADE-62 Further work	0.75	4.Further work: Give an estimate for how much effort that is necessary to complete the product / project.
ADE-311	ADE-62 Suggestions for improvement	0.75	5.Suggestions for improvement. What is missing to make this course better for both students, customers and advisers. Finish document
ADE-312	ADE-28 read through the entire document	10	also focus on the tense that is used and the way we address ourselves i.e. our group ("our group", "the group"). It should be consistent
ADE-313	ADE-28 presentation of entire structure of the report	4	
ADE-318	ADE-273 create editor demo Linda	2.5	
ADE-319	ADE-273 contact customer for old screenshots and the problems of its usage	0.5	
ADE-320	ADE-273 take screenshots of the old versions of our editor	2	
ADE-321	ADE-273 create slides for each sprint (concerning impediments mainly)	5	
ADE-327	ADE-57 Put UML diagrams in the document	2	

ADE-328	create uml diagrams for the architecture		
ADE-329	ADE-328 class diagram	6	
ADE-330	ADE-328 use cases	2.5	
ADE-331	ADE-328 sequence diagram	6	
ADE-335	Write sprint 6 document		
ADE-336	Customer and Adviser Contact	5	
ADE-337	practice presentation		
ADE-339	ADE-335 Sprint/backlog specification	0.25	
ADE-340	ADE-335 Short description of solutions to sprint backlog	2	
ADE-343	ADE-335 Planning	0.5	
ADE-344	ADE-335 Summary of sprint	1	
ADE-345	ADE-335 Review sprint 6 document	1	
ADE-347	ADE-335 Add documents to final report	0.75	
ADE-350	ad templates		
ADE-351	ADE-350 create double death notice template	0.5	
ADE-352	ADE-350 update font in the existing template	0.333	
ADE-354	Burndown charts for the report		
ADE-355	ADE-354 Create burndown charts	0.5	One burndown chart for each sprint
ADE-356	ADE-354 Add to the report	0.5	With caption, and make sure it shows up on list of figures
ADE-357	phase log in the report		

ADE-358	ADE-36 describe api plan	3	
ADE-360	Add remaining documents to report		
ADE-361	ADE-360 Test document	0.5	
ADE-362	ADE-360 Evaluation document	0.25	
ADE-363	ADE-360 Architecture document	1	
ADE-364	Find solution for change of user type	0.5	
ADE-365	delete server related stuff	0.333	
ADE-366	Manual testing		
ADE-367	ADE-366 design manual tests	3	
ADE-368	ADE-366 perform manual tests	1	
ADE-369	ADE-357 add phase log to the appendix of the report	0.5	
ADE-370	ADE-357 create a graph from the phase log	1	
ADE-371	refactor model.js injections	1	
ADE-372	ADE-337 Knut	2	
ADE-373	ADE-337 Pernille	2	
ADE-374	ADE-337 Paul	2	
ADE-375	ADE-337 Linda	2	
ADE-376	ADE-337 Erik	2	
ADE-377	Take care of Glebs feedback		
ADE-378	ADE-273 finish presentation	5	
ADE-379	ADE-36 Review	1	

ADE-380	ADE-276 review	1	
ADE-381	ADE-377 make sure it is technically written (and why which framework...)	3	
ADE-382	ADE-377 go through Glebs comments in the document	5	
ADE-383	copy latex documents	0.5	
ADE-384	put all sprints in one chapter	0.75	
ADE-385	ADE-62 prepare overall retrospective	0.75	
ADE-386	ADE-273 create editor demo Erik	2.5	
ADE-387	ADE-273 think about features to present in the editor demo	2.5	
ADE-388	Fix template creation view		
ADE-389	ADE-388 save correct available number of components	1	
ADE-390	ADE-388 read number of components correctly	1	
ADE-391	ADE-388 create tab for specify the available number of components	1	
ADE-392	ADE-388 apply the specified number of available components to the data	1	

Table 8.7: Tasks and subtasks assigned to this sprint, the corresponding estimated realization effort, and an optional description of the task

Planned meetings for this sprint

- Adviser
  - 14.11.2014
- Retrospective
  - 18.11.2014

8.6.3 Solutions to sprint backlog items

ADE-193 Styling

The subtask *ADE-240 Clean up CSS* was transferred from sprint 5.

The CSS code was refactored and organized according to logical relationship, which made it more readable and quicker to modify for the customer.

ADE-261 Fix the appearance of the image and the PDF

To correct the DPI of the PDF and image files, a workaround using the current solution was made. At first the possibility of replacing the ''toDataUrl" function with ''toDataUrlHd" was researched, which would produce a canvas with higher DPI. The research revealed that this function is only supported by HTML5.1, which is not supported by any public browsers yet. The workaround was to scale the canvas up to a resolution great enough to support the required DPI, and then scale the canvas down to correct size while retaining the number of pixels. This new approach generated a canvas with the same amount of information as with the previous method, but contained a higher number of pixels, in result yielding the required DPI. Another workaround, using a hidden preview with higher resolution for the canvas generation, was researched; with the limited time this was not implemented during the project. There is room for improvement regarding this task.

ADE-328 Create UML diagrams for the architecture

The subtask *ADE-331 Sequence diagram* was transferred from sprint 5. In addition, subtask *ADE-329 Class diagram* was reopened to change the class diagram previously made according to the changes in the code.

To increase the comprehension of the architecture document and illustrate the design of the application, sequence and class diagrams were made in this sprint. The sequence diagrams were made to illustrate the typical work process of the two user roles in the application, i.e. the system administrator and the funeral agency. These diagrams also show when interaction with the customer's servers happen. The sequence diagrams can be found in section 7.8. The class diagram needed to be changed due to refactorization of the code. The final class diagram can be found in section 7.4.

ADE-350 Ad templates

The existing template for a normal death notice had to be changed by specifying the font and font size. In addition, the customer requested a template for the double death notice (a death notice for two deceased persons), as this was part of their current system. When it was created using the template creation tool, some errors handling the rules were discovered. To fix these bugs task *ADE-388 Fix template creation view* was created and added to the sprint as this was the last sprint 8.6.3. After fixing the bugs and adding the missing parts, the second template could be created.

ADE-364 Find solution for change of user type

Usually the user role is specified by the request from the server initializing the application, but for demonstration purposes, an option to change the user role within the application was required. This problem was solved by creating an additional button in the top right corner of the editor, which toggled the boolean variable ''templateView" to change the user role. The view is automatically updated to present the correct view when changing the role.

ADE-371 Refactor model.js injections

In order to improve the structure of the code and make it more readable, files injected into ''model.js" were refactored. The result was a restructuring of ''index.html", and moving the content of ''componentController" to ''editComponentsDirective", ''previewComponentsDirective", ''ToolbarDirective", and ''Filter". Injections and tests that were affected by the refactorization were modified as well to adapt to the new structure.

ADE-388 Fix template creation view

In order to show and change the correct number of available components specified by the template, variables for the maximum number of components of the single types were introduced in the templates and application logic. Since there were several different types of components, a separate tab for specifying the maximum number of each type of component was made. If the maximum number of available components is unlimited, it is set to ''-1" which is the default value. Lastly, styling was applied to the new content.

8.6.4 Burndown graph

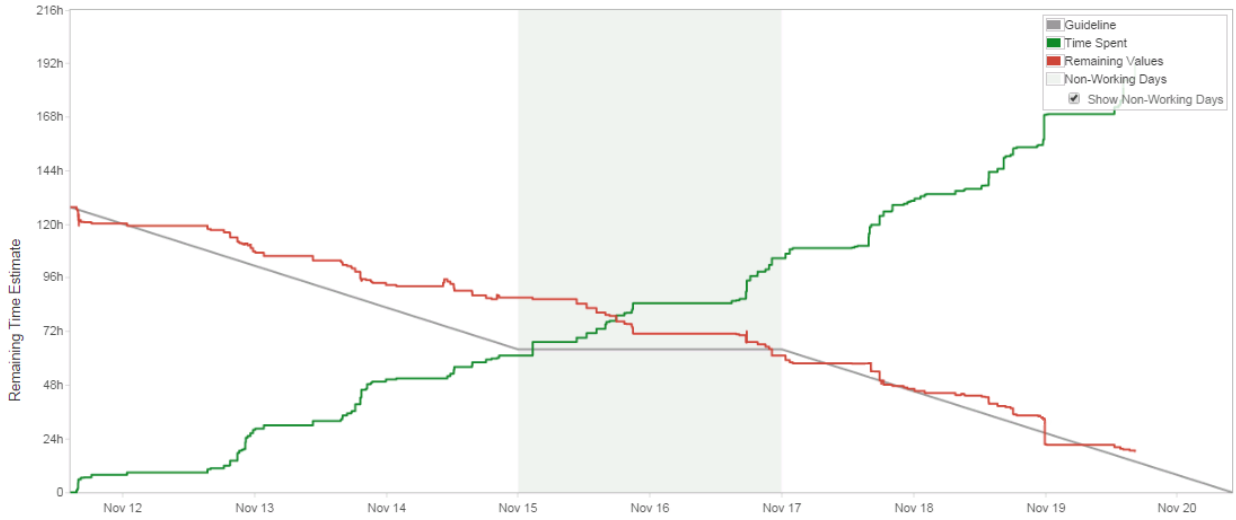


Figure 8.6: Correlation between work hours estimated and work hours done in sprint 6

For a general explanation of the burndown chart, see subsection 8.1.4.

Initially this sprint was estimated to a high amount of work effort, and in addition to this more tasks were added, signified by the increase in estimation. The vertical changes to the estimates during the sprint which were immediately undone, were mistakes which we were not able to undo in the chart. As seen at the end of the graph representing time spent, the work effort realized were larger than the initial estimated work effort, which tells us that several of the tasks were underestimated. This sprint lasted until the representation, so the chart presented here only includes the work completed until the printing of the report.

8.6.5 Summary

During this sprint our main focus was to finish the documentation and presentation, while fulfilling all the requirements for the application. Generating the image and PDF file for the advertisement proved to be complex to do from the client's browser, and our solution was not ideal; while the files generated had the correct DPI and dimension, the image quality of the advertisement was imperfect. The templates made were corrected according to the customer's requirements, and those missing were made. In order to demonstrate the different user roles to the customer, a toggle button to change roles were added, though this would not be a part of the final application.

Despite the staggering amount of work to be done, with even more tasks that were added during the sprint, all tasks were finished. More unit tests were made, and manual tests have

been performed on the finished application. User guidelines for both the customer and the funeral agency have been written, as well as a description of the architecture.

During a meeting with the adviser the group received feedback on the presentation, which proved most helpful in improving it and preparing for the final demonstration.

9

Testing

For this project it has been advantageous for us to use unit testing during our development, mainly because the customer wanted to expand the application later on. They had specifically requested some unit tests to be implemented. When expanding an application, it is a great tool to have unit testing that will, hopefully, fail if some of the existing functionality gets compromised. Some of the functionalities covered by the user interaction is hard to test with unit testing. Therefore, we developed manual tests as well. We chose to do both white box testing and black box testing.

9.1 White box testing

White box testing is testing a program where the structure and the expected outcome from the different functionalities are well known. White box testing is to test beyond the user interface and into the logic of the system [14].

Our white box testing consisted of unit tests, which we chose to include in this project. Due to the fact that the programming language in the project is JavaScript, with the use of the AngularJS framework, it was natural to choose karma and jasmine for our testing (see tools and technologies in section 6 for more information). We wrote unit tests covering the most important parts of our code. At the beginning of the project, we agreed that group members should write the tests for the code they made themselves. The problem was that by the time we were able to set up the test framework and run some tests, the coding was well underway. We managed to begin the tests, and the group members with the best overview of the code were able to start on the testing. We soon had implemented about 80 tests for our application. Even though we changed parts of the logic during the project, we made the tests work, and used them to find irregularities in our changes.

9.2 Black box testing

Black box testing includes testing a program without knowing the inner structure of it. Because some of the functionalities covered by the user interaction is hard to test with unit testing, we chose to do manual testing. The use cases were our starting point for the development of the manual tests C.

Please do these tests in the same order as they are written and while performing the tests make sure that all changes of the components appear in the edit section as well as in the preview.

ID	Test name	User	Execution	Expected result	Result
1.0	Create Template	IT-department	Open the editor as an admin.	View 3 tabs in the preview section of the editor and the possibility to make new rules	True
1.1	Lock element to the top	IT-department	Add a name component and lock it to the top of the ad.	A rule added to the rules list called “A in position X 0”. The lock symbol appears on the component and when hovering over the component the cursor looks normal, the component is not draggable, the delete button text changes to “Deaktiver”.	True

1.2	Create a block	IT-department	Add a date of birth component. Make sure that the name component is directly before the date of birth.	A rule added to the rules list called “A rett før B 0 1”. The border between the two components disappeared, both components have a lock icon and the cursor looks normal, the block is not draggable, the delete button texts change to “Deaktiver”.	True
1.3	Add two components	IT-department	Add a symbol component, add a poem component	The poem component is directly under the symbol component at the bottom.	True
1.4	Make sure a component is not at the bottom	IT-department	Create a rule that the symbol component must not be at the bottom	A rule added called “A er ikke nederst”, no lock symbol appears, when hovering over the component the cursor changes to the drag’n drop indicator, the component is draggable, the delete button text changes to “Deaktiver”.	True
			Try to drag the symbol component to the bottom	It is not possible to drag the symbol component to the bottom. It jumps back to its previous position	True

1.5	Make sure a component is at the bottom	IT-department	Lock the poem component to the bottom of the ad.	A rule added to the rules list called “A er nederst”, the lock symbol appears on the component and when hovering over the component the cursor looks normal, the component is not draggable, the delete button text changes to “Deaktiver”.	True
1.6	Lock a component to a position	IT-department	Add a family component.	Check that the component is added at position 3 i.e. on top of the bottom field	True
			Try to drag the family component to the bottom	It is not possible to drag the family component to the bottom. It jumps back to its previous position (3)	True
			Lock it to position 3.	A rule added called “4 i posisjon 3”, the lock symbol appears on the component and when hovering over the component the cursor looks normal, the component is not draggable, the delete button text changes to “Deaktiver”.	True

1.7	Prevent a component from a certain position	IT-department	Add another family component, add a line component	Check that the family component is added at position 4, and the line component at position 5.	True
			Make sure the family component never ends up at position 5.	A rule added to the rules list called “5 ikke i posisjon 5”, the delete button text changes to “Deaktiver”	True
			Try dragging the family component to position 5	It is not possible to drag the family component to position 5. It jumps back to its previous position (4)	True
1.8	Make sure a component is not before another	IT-department	Add a family two column component	A family two column component is added on top of the poem	True
			Make sure the family two column component is never above the line component	A rule added to the rules list called “7 ikke rett før 6”, the delete button texts of both components change to “Deaktiver”	True
			Try to drag the family two column component somewhere over the line component	It is not possible to drag the family two column component over the line component. It jumps back to its previous position (7)	

			Try to drag the line component under the family two column component	It is not possible to drag the line component under the family two column component. It jumps back to its previous position (6)	True
1.9	MaxNumber of available components	IT-department	assign the following numbers to the maxNumber of components: Fødselsdato -1 Forklaring 1 Familie 2 Familie, to kolonner 1 Informasjon om begravelse -1 Informasjon om gaver -1 Linje 2 Navn 3 Sted og Dato 10 Vers -1 Innledning -1 Symbol 1 Tekstfelt 0 Tekstfelt, to kolonner 0	the following available components should be in the selection where components can be added: Fødselsdato Forklaring Informasjon om begravelse Informasjon om gaver Linje Navn Sted og Dato Vers Innledning	True
			add an explanation component	the component type in the selection list disappears	True
			delete the explanation component	the component type appears in the selection list	True
1.1	Save template	IT-department	Click the save button to save the template	The local storage contains the saved ad	True

1.11	Load the saved template	IT-department	Click on load saved ad to load the ad from the local storage	Everything should be in the same way as before. Only the ids of the components increase. (check also the rule list)	True
2.0	Load from json file	IT-department	save the json string from the local storage in a local file, reload the page and load the saved template	The ad should look exactly like it was before (Maybe take screenshots to verify it)	True
3.0	Choose template	Funeral Agent	Open the editor as a regular user(funeral agent) and choose the previously made template	Open the previous ad, with no possibility to add or change rules	True
4.0	Create Ad from template	Funeral Agent			
4.1	Rearrange Fields	Funeral Agent	Try to move the fields from the template around.	The symbol, second family, line, and family two column components should indicate movability. Still, it should not be possible to move them anywhere. The other components should show the lock	True

4.2	Add a new component	Funeral Agent	Add a component containing time and place of death	A component is added on top of the poem.	True
4.3	Move the new component	Funeral Agent	Try to move the new component around	The new component can only be moved on top of the line and the family two column component	True
4.4	Delete and hide components	Funeral Agent	Hide the poem and delete the time and place of death component	The poem disappears from the preview and is not editable, it is covered by gray stripes, the delete button text changes to “Activate” and the button becomes green. The time and place component should be gone completely from both preview and editor.	True
4.5	Activate component	Funeral Agent	Press the activate button of the poem component	The poem is editable again and should reappear in the preview, the delete button text changes to “Deactivate” and the button becomes red again	True
4.6	Change the font	Funeral Agent	right-click on a component with a textfield	the right-click menu appears	True
			change the font	the font changes both in the edit section and the preview	True
			click somewhere on a component	the right-click menu disappears	True

4.7	Change the font size	Funeral Agent	right-click on a component with a textfield	the right-click menu appears	True
			change the font size	the font size changes both in the edit section and the preview	True
			click somewhere on a component	the right-click menu disappears	True
5.0	View ad	Funeral Agent, Editor	View ad in preview window	Ad is visible and representable in the preview window	True
6.0	Approve ad	Editor	Push the approve button	Should download an html and pdf file in your browser	True
			Load the double death notice template and push the approve button	Check that the content of the downloaded files covers the entire ad	True
7.0	Blocks	IT-department	restart the editor as admin		
7.1			create the following components: symbol, name, date of birth, explanation, poem, information about funeral, information about gifts		
7.2	create a block	IT-department	make the symbol be directly before the name	the border between the two components disappears, the delete buttons show “Deactivate”	True

7.3	drag the block down	IT-department	move the block under the date of birth	Both of the components should be moved at the same time	True
			fix date of birth to the top	A rule added to the rules list called “2 in position 0”. The lock symbol appears on the component and when hovering over the component the cursor looks normal, the component is not draggable, the delete button text changes to “Deaktiver”.	True
7.4	add locked component to block	IT-department	make the date of birth be directly before the symbol	the block consists of all three components i.e. no border in between, all three components have the locked symbol and are not draggable	True
7.5	move block up	IT-department	make the information about funeral be directly before the information about gifts	the rule is added to the rule list, the border disappears, the delete button text changes to “Deactivate”	True
			move the created block on top of the poem	moving works	True
7.6	move single component along another block	IT-department	move the explanation to the bottom	moving works	True

7.7	move block along another block upwards	IT-department	make the poem be directly before explanation	the rule is added to the rule list, the border disappears, the delete button text changes to “Deactivate”	True
			move the block from the bottom in between the two other blocks	moving works	True
7.8	move block along another block downwards	IT-department	move the block from the middle to the bottom	moving works	True

Table 9.1: Manual testing table

10

Evaluation

10.1 Team dynamics

Our group consisted of seven people, one German, and six Norwegian students. Some of the members knew each other beforehand, but mostly the group was randomly put together to include seven strangers who were supposed to work closely together for the duration of 12 weeks. An important way our group chose to discuss our work was by having regular retrospective meetings at the end of each sprint, to reflect on different issues regarding the past sprint. At the end of the project, we had a conclusive retrospective for the entire project. The results of this overall retrospective are presented in this chapter.

10.1.1 Self evaluation

During the retrospective we reflected on what went well, and what did not go so well. Everyone contributed with their own opinions and it resulted in this table 10.1.

What worked well?	What did not work so well?
We were really well organized during the work process	We were not as organized in the beginning of the project as we were in the later stages
Roles and responsibilities worked especially well	We did not always work as efficiently as we would have liked
We had good internal communication in the group	We had issues with punctuality
Using Scrum worked really well	We had an uneven division of workload
Group dynamic was good	We struggled with a general lack of task completion
Everyone helped each other out if we felt the need to allocate extra resources for specific tasks	Non-efficient way of making estimations in the beginning of each sprint
A lot of face-to-face work meetings where we were able to discuss and help each other.	One role was really small - which lead to problems like wrong distribution of workload
Team buildings, i.e. social gatherings for the entire group, were a lot of fun and helped the group to get even more familiar with each other	We should have had a main project leader instead of democracy
We have reflected on our work after each sprint in a retrospective meeting which uncovered unexpected problems as well as strengths of the group, and helped to improve future sprints	Should have included more group members in the bigger parts of coding
We did really well with learning all the new tools and technologies	We did not really have any additional time to spend on this project because we have other courses in addition to this course, and it is hard to prioritize different courses
Everyone was motivated for the project from the beginning until the presentation.	We sometimes make jokes on each others expenses
Everyone in the group enjoyed each other's company quite a lot	
Rotation on the distribution of roles for different meetings so that everyone could learn and improve him-/herself	

Table 10.1: Self evaluation table

10.1.2 Potential organizational and structural improvements

Our group worked quite well from the beginning, but there were still issues to be discovered in the retrospective. Such discoveries were of great value, to avoid similar problems in future projects. The group discussed elements that could have improved our organisation, result, or

other aspects of this project.

We spent some time in the beginning of the project, just discussing how to organise the project. We did not have any defined roles, but when we finally decided to define roles and assign them to the group members, we saw that it eased the distribution of the tasks. We still wanted to include the entire group on most of the different elements of the project like programming or documentation, but if there was a question about a specific topic, the contact person was known. In a project of this size, roles should be defined in the early stages of the project. The roles could be changed or expanded during the project. The group experienced some problems with the scope of one of the roles as its scope was too small. The role should probably have been better defined and should have included more responsibility.

This project was challenging for our group members, due to the fact that most of our technologies were unknown to us. Therefore, 13% of the entire time spent on the project were dedicated to study the different technologies E.1. To give feedback to the customer about how well their requirements would be realised through the different technologies was difficult. The estimations of our first tasks were also hard to make, because of our lack of experience both in the technologies and the estimation process. We should probably have been more thorough during preliminary study and research before giving feedback and making estimations. For instance, we might have been able to realize that we needed to implement a server to make the application optimal. However, because the customer did not want a server, we did not see the need until the very end of the project.

We also had a problem with interrupting each other during discussions. People got very involved and wanted to share their ideas, which caused interruptions. This was something we worked on during the entire project e.g. by introducing a “talking mitten”, and we improved, but sometimes we still forgot. The group should have been stricter on this from the beginning, also by having dedicated time for general discussions etc.

The group also struggled with including more group members to most aspects of the project. During the project, people got more specialised to different areas, and it was easy to just delegate tasks to people who were already familiar with the topic. This made the estimations easier and far smaller, but in the long run, only a few group members had a good overview of the code, the documentation, the testing, and other areas. The testing also started too late in the project, but the group managed to catch up on the testing during the project and produced a significant number of unit tests and manual tests.

The group started by using different tools for the group organisation and code version control, but ended up using JIRA and bitbucket. We spent a significant amount of time switching to the final organisation tools, that would have been saved if we had known the benefits of the tools we ended up using. Even if we considered using JIRA from the beginning of the project, we neglected it due to the small fee. Later we decided to use it anyway, and the customer gladly

offered to cover the fee. The groups limited experience with JIRA also made us neglect the potential of some of the functionalities provided by this organisational tool. The use of epics would have helped us categorize the tasks in categories like programming or documentation. It could have given us a greater overview of the amount of work that was put into different aspects of the project. We also should have created issues for meetings and other administrative obligations we had during the course as we have spent a lot of time on that which is not reflected in the burndown charts. If we had used epics and created tasks more consequently, our phase log would have been unnecessary. The different categories for logging hours in the phase log were too general and should have been more defined.

Still, using JIRA also made us spend too much time on the bureaucracy of the project, spending more than half a workday between each sprint on the retrospective and sprint planning. Especially with regards to the fact that we had limited time resources, we felt that the planning before each sprint was too insufficient. This is something we could have reflected on and discussed more during the project.

10.1.3 Group conflicts

Conflicts may be seen as an important aspect of group projects, because conflicts often trigger discussions which may lead to increased productivity. However, our group has experienced very few actual conflicts. We have all enjoyed each other's company and have been satisfied with how often we have met in person to work. Because of this tight group bond, group members may have been hesitant to discuss issues during sprints, in fear of damaging relationships with other members or the team spirit itself. In order to resolve unspoken issues during sprints, we held sprint retrospective meetings at the end of each sprint. The retrospective meetings allocated dedicated time to express issues the members might have kept to themselves. The meetings were held in order to improve the group work, not to blame each other. This has been a great tool to uncover and resolve hidden problems during the course of this project.

In the very beginning of the project, during the group dynamics session hosted by the course staff, we defined some ground rules on which we all agreed. Our late policy was among these. During this project we had some difficulties showing up on time for our meetings. By being five or ten minutes late, the rest of the group had to wait to start our daily standup meeting, which wasted their time. By discussing and refining our ground rules thoroughly, we were nearly able to solve this issue completely.

Sometimes, tasks were made more complicated than expected, without the other team members knowing about it. When we were defining and implementing the rules, we discovered many edge cases that had to be taken care of. For instance, the drag and drop of the components. The more work that was spent on implementing it, the more work got obvious to complete the task. By spending a lot more time on a single task than the original estimate, other tasks allocated

to the same person were lower prioritized than what they should have been. In some cases, the lower prioritized tasks had to be moved to a different sprint. We handled this issue by discussing problems we had with our tasks in the group, which allowed us to further improve estimations for the proceeding sprint, ultimately resulting in tasks not having to be expanded as often as previously.

All in all, our conflict level has been at a minimum, resulting in a very pleasant environment to work in. However, in retrospect we thought that a moderately increased conflict level would have aided us in increasing productivity by sparking discussions that may have led us in slightly different directions.

10.1.4 Project goals

In the early stages of the project we determined our personal and project-related goals. Discussing our goals in plenary during the group dynamics session resulted in the following list:

Gain real customer-driven development experience with a larger group Improve programming skills Learn to develop software efficiently

These goals were shared between all team members, which resulted in good cooperation and an agreement of what we wanted to achieve during the project. By being assigned to a real customer, and developing a real application, we have gained a tremendous amount of software development experience. In addition, because we were forced to explore technologies none of us had worked with before, such as JavaScript with AngularJS, HTML5, Karma, and Jasmine, we have improved our programming skills significantly.

However, regarding project-related goals, we felt we did not reach our full potential. Some of the requirements were only partly fulfilled, such as the correct dimensions and resolutions of the final PDF-file. In addition, we realized that some aspects took longer than estimated. In our final project retrospective, we came to the conclusion that these issues may have been solved if we had spent more time on preliminary study. Nevertheless, we are overall satisfied with how we have achieved all of our goals.

10.1.5 Learning outcome

As mentioned in subsection 10.1.4 a huge part of what we have learned was on the technical side by learning e.g. JavaScript or HTML5. Furthermore, we learned a lot on the group dynamics side. Some group members learned that it is important to be responsible for the assigned work, while others learned that there is also a life outside of work. It is important to have good team members that work well together in order to be able to create a good product and have a productive process. We have learned how to communicate better and that being aware of each other's problems improves the entire group performance. In addition, we learned how to

appreciate each other, which has a positive impact on the group dynamics.

10.2 Adviser

During this project we were given an adviser from the institute to give us regular feedback on our work and process, and make sure we were experiencing progress. Our experiences with the adviser were mostly positive, and our adviser showed up prepared to our weekly meetings. The group sent him all of the documentation progress each week, which turned out to be a significant amount of pages to go through. Still, our adviser managed to read through them and give constructive feedback on them. He also responded to our correspondence in a timely fashion, which made it easy to communicate. He always showed up on time, and we had a nice, informal tone during our meetings, which made it easy to ask even the stupid questions. In spite of all this, the group sometimes missed diverse feedback which could have helped us improving our work. The feedback was sometimes a bit too general and the negative feedback was lacking. To be able to improve our work, it would have been nice to get feedback on what we did not do so well. We also missed feedback on the process itself, with the meeting being focused on the previously delivered documentation. The documentation naturally included information on the progress but the feedback on this might not always have felt sufficient. We could also have been pushed to organize ourselves better. Maybe this could have been achieved by moving the time of the group dynamic workshop to the very beginning of the course. It took place after we had already come up with our own group dynamic and organization, without any supervision. We also felt that it was not optimal that the adviser was sent to a conference for our last week working on the project. Luckily he offered to give us an extra meeting before he left, where we could go through the presentation with him. His feedback on this was of great value to our presentation.

10.3 Customer relations

Customer relations were one of the core aspects of this course because it helped the students get an insight of how real software development is done. We maintained a professional and friendly relationship with our customer throughout the entire project. Every other week, we had a meeting with the customer where we demonstrated the requirements we had implemented since the last meeting. The customer supplied us with feedback regarding the functionalities and stated what needed to be modified or improved. In addition, because we arranged the customer meetings to be right before our sprint planning, we were able to properly select items from the product backlog to put in the sprint backlog together with our customer. During our final project retrospective, we all discussed how we felt the customer interaction was during the

course of this project.

10.3.1 Customer evaluation

What worked well?	What did not work so well?
Customer seemed very positive to our project, and showed enthusiasm throughout the entire duration	Initially, defining rules for customer communication was difficult e.g. response time
Customer was very agreeable to our solution for the application	It had a major impact on our project when our customers were sick or busy. We were not always able to receive concrete feedback on our application, and had to allocate extra resources in order to reschedule meetings.
We have been very well prepared for customer meetings. All documents needed for every meeting has been supplied prior to all customer meetings, such that there was no disambiguation as to what the purpose of the meeting was	It was sometimes difficult to receive precise feedback, especially regarding negative feedback which would have been of much more help to us than the customer might have thought.
We had regular meetings with the customer, which enabled us to rapidly and incrementally improve our application	We could have asked more open-ended questions in order to trigger a descriptive and informative response from the customer.
The relationship between the group and the customer was professional but friendly	
Communication with the customer went well. There was always an informal atmosphere during our meetings, which allowed everyone to say what they wanted to say, resulting in informative meetings.	
Customer replied fairly quickly to email	
Customer knew that a huge part of this project was documentation, so they understood that we needed quite a lot of time writing this report.	

When the customer had questions during meetings, the correct team member, based on role and what category the question was in, answered the question. This reflected very positively on us as a team with structured organization.	
We were good at showing progress every other week.	
We listened carefully to our customer when they told us what they wanted, and we prioritized the backlog items together with the customer in a professional manner	

Table 10.2: Customer evaluation table

Overall, our group has been very satisfied with the customer we were assigned to. The initial requirements and application were perhaps somewhat unclear. However, that meant that we were given more room to come up with incremental, prototypical solutions to requirements during each sprint. The project itself was enjoyable and manageable simply because of the fact that the customer showed so much enthusiasm for our progress. We felt that they were interested in what we were developing and wanted to actually use our solution, which ensured a good meeting atmosphere and working relationship. This was also highly motivating for us as a team. Throughout the entire project, they gave us essential feedback on the implementation and requirements. Despite feeling that some of their feedback was rather unclear, we were still able to improve our solution incrementally, and they always seemed satisfied with what they were shown. In addition, we think that having two customer contacts instead of just one has increased the amount of feedback we were given, which we are very pleased with.

10.4 Further improvements on the application for the customer

The following improvements are either something we did not have the time or technologies to finish, or they were simply not part of our task. These are optional for the customer themselves to implement.

- The image, PDF, and HTML generation has to be rethought, as the technology for implementing a satisfying solution for these functionalities is not yet available for the client side.

- The deletion of rules should be added to complete the template creation for the IT-department user. This was specified as a functionality that we did not need to prioritize.
- More unit tests should be implemented to cover both new, and existing functionalities. Integration tests, which are not possible without a server, should be included. We also recommend to add some sort of code coverage check.
- The following rule enforcements may also be added
  - When a component is dragged across a locked component, the drop is not possible as this would mean a change of the position of the locked component. A solution would be that the component on the other side of the locked component would swap around the locked component.
  - The maximum number of components of a type can be specified to be less than the number of components added in the template, which does not make sense. It needs to be forced that only valid values can be entered.
- Implement a server connection (GET, POST) to, among other features, be able to
  - Dynamically load additional symbols
  - Categorize the symbols in folders and browse through them
  - Implement a login functionality with different user roles
- Improve dependency relations in the code.

These are possible extensions for our application

- It should be easy to run the editor in different languages
- It should be extendable to other types of ads, and to pamphlets of several pages
- If some security is implemented the project could be commercialized. As the editor is supposed to be integrated into an already secure system by Adresseavisen, we were told that the security aspect could be neglected.

10.5 Risk assessment

Risk 1: Lack of time

During the project we faced one week where the group members had a lot of deadlines in other courses. Therefore, we could not spend as much time on this project as we wanted. We solved this by shifting tasks to the next sprint and spent more time during the next week.

Risk 3: Sickness

There was no particular person being more sick than others, and when a person was too sick to go to school, he/she was still able to work by remote communication. Furthermore, there was at least one team member who was able to take over his/her tasks.

Risk 9 : Lack of Competence

We were faced with a number of unfamiliar tools and technologies. The problems were that we had to spend a lot of time on pre-study and it was hard to make the estimations properly.

Risk 10: Computer trouble leading to technical problems preventing team members from doing the assigned work
We had a couple of computers crashing or malfunctioning. Once, the group member had to buy a new computer. This happened too fast for it to really have any negative effect on our work flow. Because all of our work is saved remotely, no data loss occurred. We also had an episode where one of our members lost all of his computer settings. He managed to find a solution quickly and other than the loss of work, no major impediments occurred.

10.6 Course evaluation

10.6.1 Summary of evaluation

The group started with quite a bit of scepticism to HTML5 and especially .NET. We had also been given the impression that we had to use Microsoft Visual Studios, and we were more than eager to find a replacement for this that was suitable for everyone. We ended up enjoying working with new, future relevant technologies like HTML5 and JavaScript. It was of great advantage to have some general knowledge of these technologies for future work and projects. We also found our task to be well defined and the project seemed clear to us very early on. One impediment we experienced was about implementing a server. Firstly, the customer asked us to implement some sort of mock server to test out application. The customer then changed their minds about the server and wanted the editor to be plainly running on the client side. They wanted to do the server implementation themselves, when integrating the editor into their existing system. Here, we probably lost out on some great server experiences, and possibilities to test the application. The group is quite satisfied with being able to actually produce a finished application, that might be used in production.

10.6.2 Suggestions for course improvements

Here are some aspects we felt is worth to have a look at, in terms of improving the course.

- The compendium should be updated. We found and asked about a lot of outdated stuff from the compendium, which the adviser repeatedly told us was not important anymore.
- Take a look at the time requirements, the compendium estimates 14 weeks, but we only had 12 weeks on this project.
- The course has an overly focus on the report, while we expected and wanted to focus on the customers' wishes. This is also what the name of the course is indicating.
- We were lucky to have a visit by a second adviser once during an adviser meeting. This was of great value to our group, and also, it seemed, to our adviser.
- Maybe not having too many groups per adviser.

- We had a workshop on group dynamics included in this course, but we felt it took place too late.There were a lot of great group dynamic discussions and values discussed, but this workshop happened about a month after the beginning of the project, and therefore we think it lost a lot of value to the groups.

Appendix A

Coding conventions

A.1 Procedure when adding a new JavaScript file

- Add the name of the app in the `Module.js` file
- Reference the JavaScript file in the `index.html` file.

A.2 Syntax conventions

- Always use lower case at the start of function and variable names
- Each method should only contain one layer of logic
- Should you wonder if your method is complete: Try reading only the method and see if you can understand what it does just from those few lines of code
- All variable names should be expressive
- The modules should start with an uppercase letter
- The service or controller should start with a lowercase letter

A.3 File naming conventions

- Every service, controller and directive should have the file name end with ”service”, ”controller” or ”directive”, respectively
- The file names should start with an uppercase letter
- When finished with a function, write an unit test that covers the code
- All file names should be expressive

If none of the rules above states explicitly how to write the code we use Google’s styleguide for AngularJS [11].

Appendix B

Risk management plan

Risk ID	1
Risk name	Lack of time
Risk consequence	Not being able to finish project completely.
Probability	4
Severity	5
Importance	20
Preventive measures	Keep a well-maintained overview of how much time we spend on each part of the project and do not fall behind. Since we have not worked together before and Scrum is relatively new to us, time estimation errors will occur – therefore, we must set a safe time frame on every part of the project.
Risk management measures	Decide on which parts to skip and which parts to keep. Core functionalities are of course prioritized.

Table B.1: Risk ID 1

Risk ID	2
Risk name	Wrong focus on either report or implementation
Risk consequence	Ending up with either an unfinished report or implementation.
Probability	3
Severity	5
Importance	15
Preventive measures	Keep track of person-hours used in every aspect of the project to see whether we are out of balance or not.
Risk management measures	Properly shift focus to the previously low prioritized task in order to restore balance.

Table B.2: Risk ID 2

Risk ID	3
Risk name	Sickness
Risk consequence	Less team members contributing to work resource pool
Probability	4
Severity	3
Importance	12
Preventive measures	Wear warm clothes, wash your hands often, eat properly and exercise.
Risk management measures	Provide the sick person with information about what is going on in the project and the sick person should try to work from home. If it lasts long the person should visit a doctor, and worst case receive a sick leave.

Table B.3: Risk ID 3

Risk ID	4
Risk name	Misunderstanding the customer
Risk consequence	Decreased time resources to finish project as the team works on something that the customer does not want to have.
Probability	4
Severity	3
Importance	12
Preventive measures	Keep the communication clear and simple. Frequently update the customer via mail and establish common understanding that mail communication is low-threshold. Do not hesitate to ask them questions. Have effective meetings with the customer and ask questions frequently so that no aspect of the system is ambiguous
Risk management measures	Go through the plans, documents, and sketches together and try to find further misunderstandings. Take actions to share all plans and documents, keep communication open between team members and agree on one solution. If an activity has already begun based on misunderstandings, we have to look into how much can be used and how much must be discarded.

Table B.4: Risk ID 4

Risk ID	5
Risk name	Adviser missing a meeting
Risk consequence	Missing information about core parts of the project or system.
Probability	3
Severity	4
Importance	12
Preventive measures	Be clear on communication with the adviser and frequently maintain communication.
Risk management measures	Report it to the course coordinator, and get confirmation on future appointments with adviser to avoid confusion or forgetfulness.

Table B.5: Risk ID 5

Risk ID	6
Risk name	Arguments and/or disagreements
Risk consequence	Decreased time resources to finish project.
Probability	5
Severity	2
Importance	10
Preventive measures	Have clear and focused communication and be nice to each other. Arguments will surely arise no matter what, so our risk management measures are probably more important in this case.
Risk management measures	Settle argument or disagreement in collaboration with the rest of the group. If arguments persists, the scrum master must initiate a group conversation.

Table B.6: Risk ID 6

Risk ID	7
Risk name	Loose parts or all work due to human or technical issues.
Risk consequence	Need to re-do parts or all work.
Probability	2
Severity	5
Importance	10
Preventive measures	Use version control for code, store documents in the cloud.
Risk management measures	Since we use version control for code and store documents in the cloud, if no one mistakenly stores locally, everything should be fine.

Table B.7: Risk ID 7

Risk ID	8
Risk name	Internal misunderstandings
Risk consequence	Decreased time resources to finish project.
Probability	3
Severity	3
Importance	9
Preventive measures	Keep the communication clear and simple. Use the Facebook group and mailing list frequently. Have effective meetings with the customer and ask questions frequently so that no aspect of the system is ambiguous.
Risk management measures	Go through the plans together and try to find further misunderstandings. Take actions to share all plans, keep communication open between team members and agree on one solution. If an activity has already begun based on misunderstandings, we have to look into how much can be used and how much must be discarded.

Table B.8: Risk ID 8

Risk ID	9
Risk name	Lack of competence
Risk consequence	Decreased time resources to focus on actual project work, increased time working on pre study.
Probability	4
Severity	2
Importance	8
Preventive measures	There are no preventive measures to do here either, other than perhaps discussing other alternative technologies that everyone in the group already knows. It is important that we have enough time in the beginning of the project to read enough about the technologies we are going to use.
Risk management measures	Provide documentation, books, videos or similar to the person lacking competence on a technology or organizational paradigm. Read and learn satisfactory amounts before set deadline.

Table B.9: Risk ID 9

Risk ID	10
Risk name	Computer trouble leading to technical problems preventing team members from doing the assigned work
Risk consequence	Decreased time resources to finish project.
Probability	2
Severity	3
Importance	6
Preventive measures	There's not much to do to prevent this – only common computer knowledge applies, e.g. maintain your computer, don't visit malicious websites etc. Save all work on the web, regularly.
Risk management measures	Find a replacement, i.e. computer lab to work on. Get the computer fixed or borrow a computer.

Table B.10: Risk ID 10

Risk ID	11
Risk name	Customer makes changes to project requirements
Risk consequence	Decreased time resources to finish project due to having to re-estimate workload.
Probability	3
Severity	2
Importance	6
Preventive measures	Use an appropriate and effective process model, such as Scrum. It is important to stay in regular contact with the customer.
Risk management measures	Make necessary changes to the project requirements in order to meet the customers needs and calculate new time estimates.

Table B.11: Risk ID 11

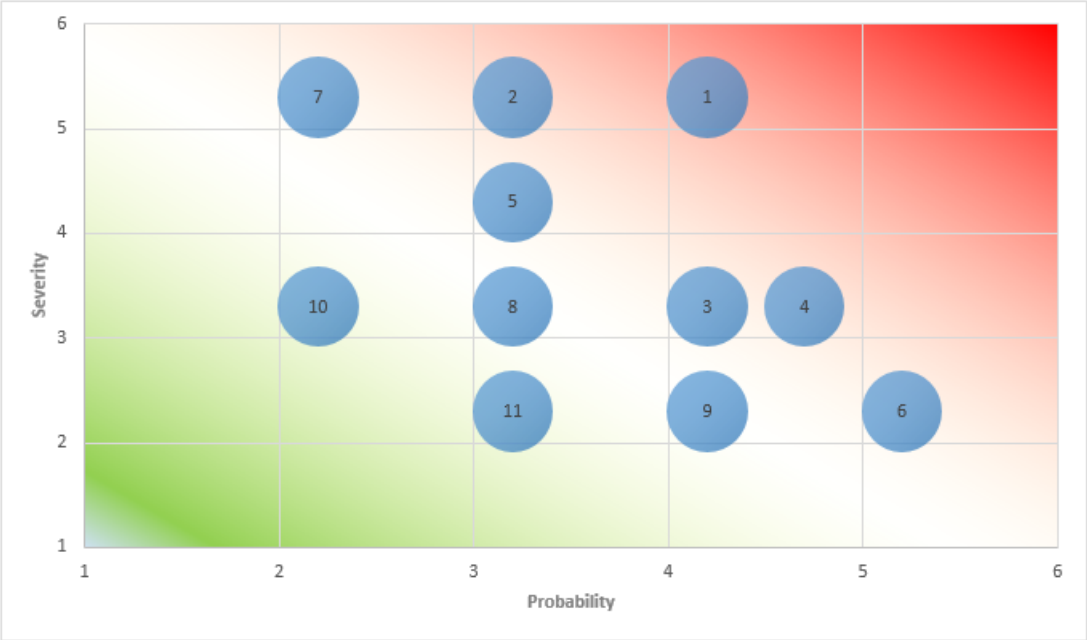
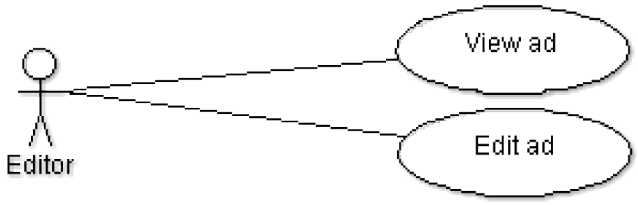
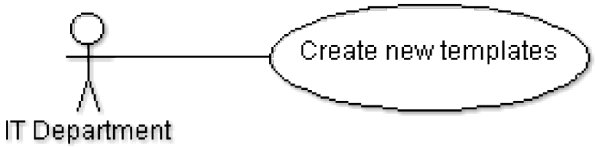


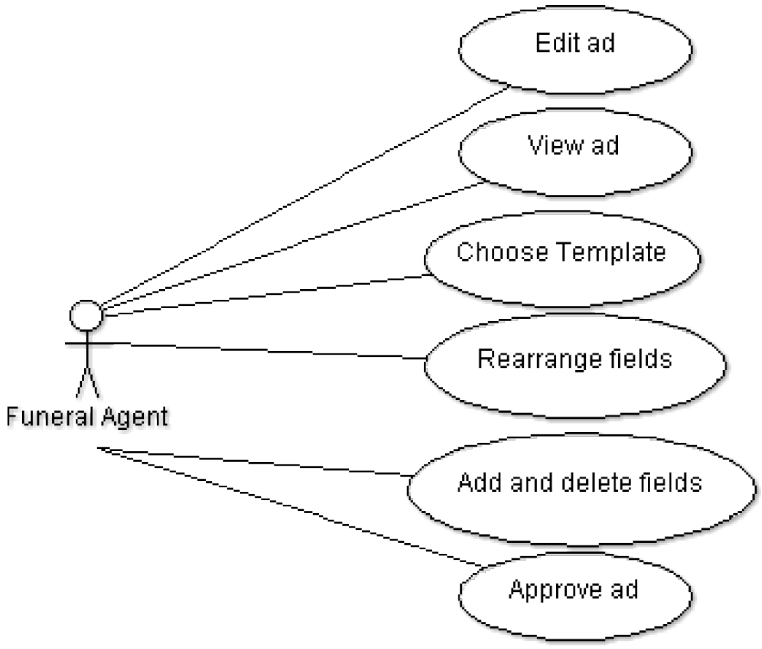
Figure B.1: Dispersal of risks

Appendix C

Use case diagrams

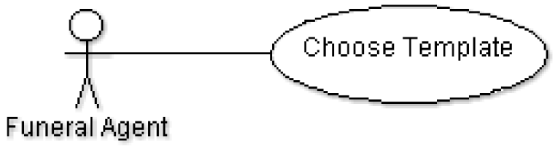




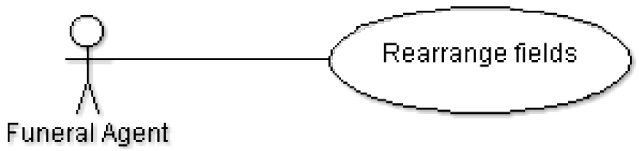


Element	Description
Use case name	Edit ad
Goal	The user wants to be able to edit an ad
Summary	The user wants to change something in the ad and save it
Preconditions	The user has to be logged into Adressavisen's servers
Postconditions	The ad is changed
Flow of events	1. The user opens an ad with the ad editor 2. Then changes the ad
Exceptions	None

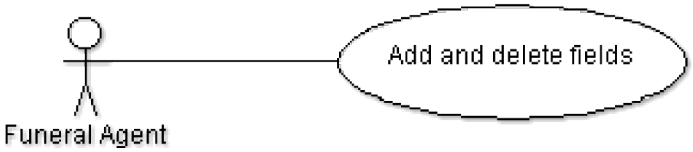
Element	Description
Use case name	View ad
Goal	The user wants to be able to view ad
Summary	The user wants to view the ad to check it
Preconditions	The user has to be logged into Adressavisen's servers
Postconditions	None
Flow of events	1. The user opens an ad with the ad editor
Exceptions	None



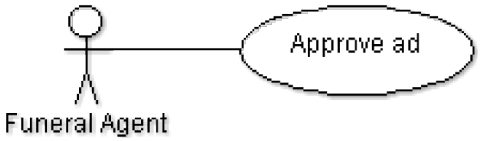
Element	Description
Use case name	Choose template
Goal	The user wants to be able to choose a template to use for the ad
Summary	The user wants to choose template for different ads
Preconditions	The user has to be logged into Adressavisen's servers
Postconditions	None
Flow of events	1. The user opens the ad editor 2. Choose a file 3. Load file
Exceptions	None



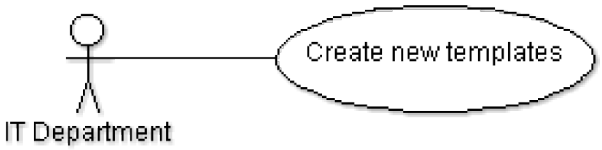
Element	Description
Use case name	Rearrange fields
Goal	The user must be able to rearrange the fields in the ad
Summary	The user must be able to rearrange the fields in the ad according to the rules set in the template
Preconditions	The user must have opened an ad in the ad editor
Postconditions	Text fields in the ad have changed position
Flow of events	1. The user drags one field to another position
Exceptions	Where the rules specify it



Element	Description
Use case name	Add and delete fields
Goal	The user should have the possibility to add and delete fields
Summary	The user should be able to add and delete fields according to the rules specified by the template
Preconditions	The user must have opened an ad in the ad editor
Postconditions	None
Flow of events	1. The user opens an ad with the ad editor
Exceptions	None



Element	Description
Use case name	Approve ad
Goal	The user should be able to approve the ad
Summary	The user should be able to approve the ad after editing it, approving it for the paper
Preconditions	The user must have opened an ad in the ad editor
Postconditions	The ad is approved
Flow of events	1. The user press the approve button
Exceptions	None



Element	**Description**
Use case name	Create new templates
Goal	The user must be able to create new templates
Summary	The user must be able to create new templates with rules and layout for later use
Preconditions	None
Postconditions	A template is created
Flow of events	1. The user opens the ad editor 2. Then adds all the text fields they want the template to contain 3. Then apply the order and rules for the template 4. Finishing by saving the template
Exceptions	None

Appendix D

Important sprint events tables

D.1 Sprint 1

- Getting familiar with AngularJS
- Testing
- Setting up basic editor
- Moving from Trello to JIRA and Bitbucket
- A lot of documents eg. for logging the work time
- Git trouble
- Prototype
- Satisfied customer
- Insufficient communication with customer
- Knut sick
- Paul too late
- Outlining documents for final report
- Satisfied adviser

Groups of ideas or events that we see as related. The points that are multi-colored belong to all the indicated groups.

- Technical issues
- Communication
- Feedback
- Programming
- Course framework

D.2 Sprint 2

- Documentation
  - Sprint 1
  - Choice of lifecycle model
  - Preliminary study
- Clarification from customer regarding
  - Rules
  - Images and fonts
- Implementation of
  - Right-click menu
  - Save and load functionality
  - Rule enforcement
  - JSON file format
- Pizza night (social gathering)
- Customer and advisor meetings
- Needed to reschedule customer meeting
- Came closer to writing test
- The advisor suggested to make our own company
- Some productive work meetings
- Knut got a warning for being late last friday

Groups of ideas or events that we see as related. The points that are multi-colored belong to all the indicated groups.

- Productivity
- Customer contact
- Communication
- Team building and teamwork
- Progress

D.3 Sprint 3

Name	What worked well?	What did not work well?	What should we start doing?
Erik	• The tasks were easier than expected • We had a very productive Tuesday • Estimations were spot on for the most part • The customer and the advisor seem satisfied with us (impressed)		• Knut should continue making cake • More testing
Thea	• Testing • PDF-generation • Documentation • Communication • Estimation	• Lack of backend • Late customer feedback • Late start on HTML generation, caused by late customer feedback • A bad temporary solution on the server side for the HTML generation	• More testing • Documentation • Design
Knut	• Meetings • Design • Cake (it was relatively good)	• No sushi • Other projects and fotball combined with this project	• Sushi • Testing • Finish up programming • Presentation
Paul	• Good cooperation • Everyone was included in documentation • Great feedback from customer and advisor • Arrived on time and fit for work	• Slightly unclear response from the customer	• Testing • Focus on documentation (both writing and reviewing internal documents and other reports)
Nils	• The breaks were more defined and effective • Less joking around, i.e. better focus • Excellent results in all areas	• Too much other school work, so didn't finish one of my tasks • Was not able to add documentation to	• Finish work that others depend on earlier

	• Everyone ready with relevant info when needed in meetings	Latex until last day because they were finished late (goes for me as well)	
Linda	• Estimation • Everyone works on what they are good at • Got tests running • Styling	• One of the members did not finish some of his tasks	• Sushi • Assign the tasks more carefully • Concentrating on my other subject and master thesis topic search
Pernille	• Good estimation	• People don't show up on time	• Sushi

Groups of ideas or events that we see as related

- Planning
- Socialization
- Customer & advisor relations
- Working environment
- Programming
- Documentation

D.4 Sprint 4

Name	What worked well?	What did not work well?	What should we start doing?
Erik	• Sushi • Cake • Gleb and customer seem happy	• We had to set up a server • Estimation was worse than last time • I did not have enough time for this course	• TESTING!!!!!!!!!!
Thea	• Verify HTML • Documentation (as always)	• Estimation of the work load • Lack of backend • Categorize symbols	• More documentation • Less programming • Stop adding features • Enjoy life more
Knut	• Good communication with both advisor and customer • Sushi • Cake • Role	• Lot of work outside of project	• Not add new features to the ad editor • Making the presentation
Paul	• Sushi • Adapting to implement new features	• Unclear focus on features • Not very precise feedback from customer	• Include everyone in writing / reviewing
Nils	• Social gathering • Getting better at LaTeX • Good reviewing of documents	• Had a little too much work to do	• Look at the styling of the report • Better personal organizing
Linda	• Sushi • Got my tasks done	• Too much to do • Stupid blocks! • Got task concerning the server too late • Worked too much on the project	• Do a really good sprint planning and not commit to too much. Probably too little and then we can still expand it • List all tasks that

			have to be done in the backlog
Pernille	• Documentation • Finished all my tasks • Customer seems happy	• Overall we did not finish all tasks • Got sick. Again.	• Testing • Presentation • Stop interrupting each other

Groups of ideas or events that we see as related. The points that are multi-colored belong to all the indicated groups

- team work
- sprint planning
- Impediments
- customer and advisor
- work

D.5 Sprint 5

Name	What worked well?	What did not work well?	What should we start doing?
Erik	• We finally started the testing • Nils was awesome with the report • Almost finished the entire sprint • "Lots" of models where made	• The image-generating was more complex than expected • No cake	• More Cake
Thea	• Testing • Better planning • People worked alot • Daily standup	• Planning includes too much details, and was kind of a waste of time • People oversleep	• Sleep more • Finish everything • Stop adding new stuff to do
Knut	• Got started with UML • Presentation getting along good	• Failed at estimation and workload	• Finishing up • Done with programming
Paul	• Tests! • A lot of work done on the report • Clear feedback from advisor on the report	• I did not have time to do much before sunday • Still have a lot to do (especially after advisor meeting)	• Document technical aspects in report (architecture etc)
Nils	• All work done early • Progress with the report • Helping each other • Daily standups • Got lots of feedback	• Spent too much time planning • Have a lot left to do	• Make sprint tasks and estimations individually/in pairs, and assign together • Running start on the sprint
Linda	• Tasks went very well so that I was able to help others in addition • Good teamwork e.g. doing tasks together remotely • Sprint planning improved! • Finally have a better idea about testing	• Little bit afraid that we get not everything done • Don't really use mailing list • Presentation demo task assignment	• Work more to get everything done - just one week left!

Pernille	• Well underway with the report • Well underway with presentation • Everything is coming along well • Finally started properly with testing	• Half the group was late today • We've been busy with other things • Short sprint, without the tuesday so there is a lot of work on few hours • Need to be better prepared for the documents we're talking about	• Cake

Groups of ideas or events that we see as related. The points that are multi-colored belong to all the indicated groups

- Planning
- Testing
- Report & Presentation
- Work behaviour
- Work progress

Appendix E

Phase log charts

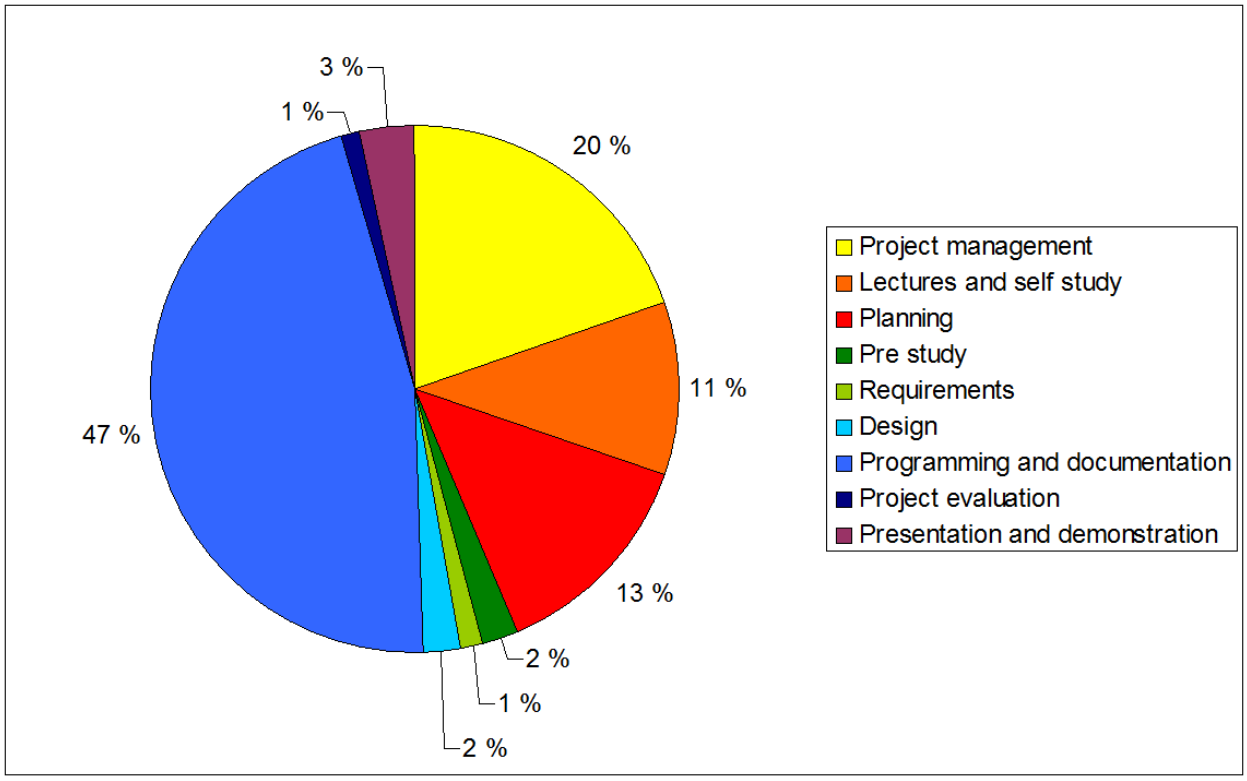


Figure E.1: The diagram shows the work distribution, categorized by phase

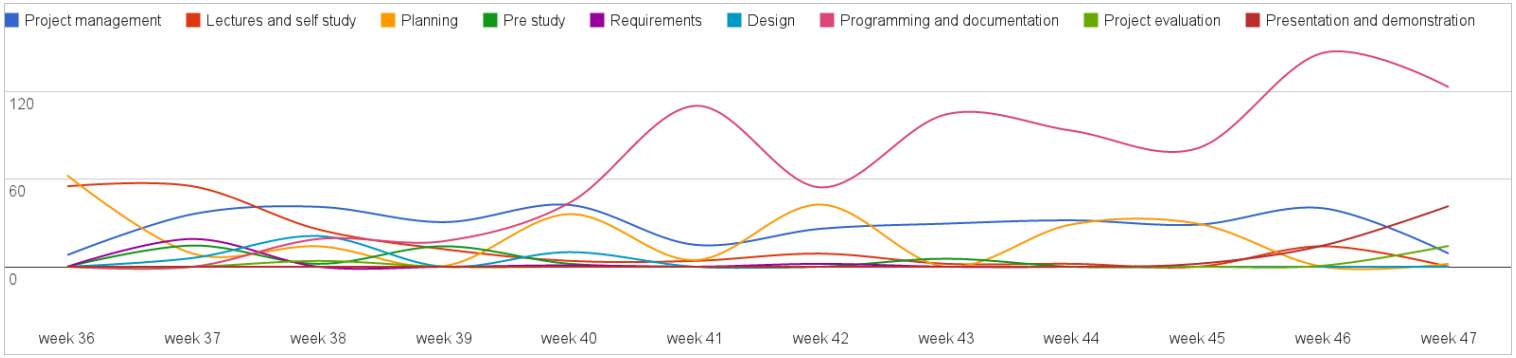


Figure E.2: The weekly work effort throughout the project, categorized by phase

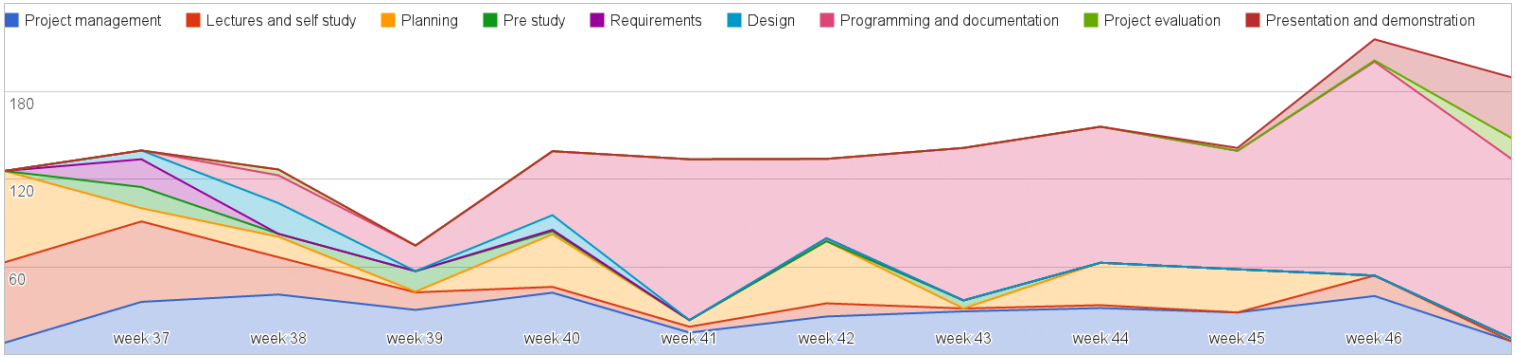


Figure E.3: The accumulated weekly work effort throughout the project, categorized by phase

Appendix F

Templates

F.1 Weekly Advisor Meeting Agenda

2014-MM-dd Weekly Adviser Meeting Agenda

0. General Information

When and Where	Roles
Date: <Weekday, dd.MM.2014>	**Primary Facilitator:** <Name>
Start: <hh:mm>	**Time Keeper:** <Name>
End: <hh:mm>	**Minute Taker:** <Names>
Room: <room number>	**Participants:** <Names> **Missing:** <Names>

1. Approval of agenda

2. Approval of minutes of meeting from last advisor meeting

3. Comments on provided documents

4. Approval of the status report, which may be structured as follows:

- **4.1 Summary**
-
- **4.2 Work done in this period**
  - **Status of the documents that are being created**
  - **Meetings**
  - **Other activities**

- **4.3 Problems – what is interfering with the progress or taking resources? Problems are often risks that have taken effect.**

- **4.4 Planning of work for the next period**
  - **Meetings**
  - **Activities**
  - **4.5 Other**

5. Review/approval of attached phase documents

6. Other issues

F.2 Customer Meeting Agenda

<date> Customer Meeting Agenda

0. General Information

When and Where	Roles
Date: <weekday>, <date>	**Primary Facilitator:** <name>
Start: 10:00	**Time Keeper:** <name>
End: 11:00	**Minute Taker:** Pernille
Room: ITV 157	**Participants:** Knut, Erik, Linda, Nils Inge, Pernille, Thea, Paul, Hans Kristian (customer), Asle (customer) **Missing:**

1. Purpose of the Meeting

-

2. Discussion

2.1.

-

3. Organisation

3.1.

4. Summary

4.1. New Action Items

4.2. Meeting Critique

F.3 Retrospective Meeting Agenda

Retrospective Sprint <number>

0 General Information

When and Where	Roles
Date: <weekday>, <date>	**Primary Facilitator:** <name>
Start: <start time>	**Time Keeper:** <name>
End: <end time>	**Minute Taker:** Pernille
Room: ITV 157	**Participants:** Erik, Linda, Nils, Paul, Thea, Knut, Pernille **Missing:**

1 Prelude: Setting the Stage

1.1 Kerth's Prime Directive

Don't play the blame game

1.2 Ground Rules

- No talking or interruptions when others are talking
- Talking stick

1.3 Check in Lite

What do we expect out of the meeting?

-

2 Past: Gather Data

2.1 Remote Brainstorm

Name	What worked well?	What did not work well?	What should we start doing?
Erik			
Thea			
Knut			
Paul			
Nils			
Linda			
Pernille			

3 Past: Generate Insights

3.1 Clustering

Group ideas / events that seem related and name the clusters

-
-
-
-
-

3.2 Discussion

What was expected? What was surprising? What makes happy / uncomfortable?

- Expected
  -
- Surprised
  -
- Happy
  -
- Sad
  -

4 Future: Decide what to do

Start doing	**Stop doing**	**Continue doing**

Accomplished:

-

5 Conclusion: Thank each other

“[name], I thank you for [...] because [...].”

Everyone is thankful to everyone

F.4 Weekly Status Report

Weekly status report

Group 5

Week XX

Summary

Activities performed the last week

Activity	Phase	Status	Comments
		Finished/in progress	

Registered work effort (see phase log for details)

Phase	Hours
Phase 1	xx
Phase 2	
Total	Sum

Planned activities for this week

Activity	Phase	Comments

Impediments

To do… we need to solve the problem regarding...

F.5 Textual Use Case

Use case ID	UseCaseID
Use case name	The name of the use case
Scope	The scope of the use case
Description	The goal of the use case and sources of requirement
Preconditions	The preconditions that must be met before the use case can begin
Assumptions	The conditions that must be met for the use case to terminate successfully
Steps	The steps between the actor and the system that are necessary to achieve the goal
Variations	The different variations of the flow of the use case

Example

Use case ID	UC-1.21A
Use case name	The name of the use case
Scope	Evaluation and file
Description	Add file(s) on revision of evaluation
Preconditions	1. Running system 2. An existing evaluation
Assumptions	1. There exist an evaluation group 2. There exist an evaluation set 3. There exist an evaluation
Steps	1. Select a revision of an evaluation 2. Click edit 3. Select add file 4. Select a file from your device (A new link add file will occur below the previous one) 5. Jump to step 2 as many times as wanted 6. Click save
Variations	(a) If no evaluation group exists, create one. Then create an evaluatio set with and an evaluation, and do step 1 again

	(b) If no evaluation set exists, create one with and evaluation. Then d step 1 again (c) If no evaluation exists, create one and do step 1 again

Appendix G

Group agreement contract

Group agreement for group 5 - TDT4290 Customer Driven Project

1. Attitude and behavior

1.1. I will take responsibility for my own actions.

1.2. I will do the work that is allocated to me during a specific time period. There will never be occasions where only one or a few team members do all the work nor will there be occasions when one team member does none of the work.

1.3. There will be dedicated feedback sessions for both positive and negative feedback regarding either:

1.3.1. programming, or
1.3.2. social behavior

I will attend these feedback sessions and must be open for constructive criticism during this time and for the remainder of the project.

2. Meeting time, absence and duties

2.1. Both time and room of meetings must be agreed upon unanimously within the team.

2.2. In the possible event that I will be unable to attend a meeting, I will make it my personal responsibility to read up on all notes that the team has made during that meeting.

2.3. Similarly, in the event that a team member is unable to attend a meeting, it is also the teams responsibility to inform the absentee about any important information and decisions that were made.

2.4. If I am late to a meeting three (3) times without warning or one (1) time not showing up without mentioning the absence the day before, I must bring one (1) cake to the next meeting, as well as make a public apology to the group.

(Late is hereby defined as ten (10) minutes past scheduled meeting time. A meeting scheduled at 09:00 must be considered to have an actual start time at 09:15 due to room reservation policies.)

3. Project rules

3.1. Each member of the team will agree on a single solution to a problem before it is submitted, committed or finalized. In the event where there is no unanimous agreement, a discussion must be initialized until every team member agrees. If no agreement can be reached, the team will hold a democratic vote on which solution to utilize.

3.2. I will do everything in my capabilities to assist my team members understand all concepts, aspects, technologies and issues.

3.3. If I do not understand a concept, technology or solution, or I have issues with my delegated work, I will not hesitate to ask my team members for assistance as soon as possible.

3.4. I will communicate with my team members about any concerns I may have about organizational or managerial structures.

3.5. As a group, we agree that the maximum allowed time before an e-mail, from either the supervisor, customer or team, is responded to is twenty-four (24) hours.

3.6. Any and all logs and/or minute reports from meetings with either the

3.6.1. customer,
3.6.2. supervisor or
3.6.3. team

must be prepared before the next meeting.

I hereby certify that I have thoroughly read this contract and that I will abide by it. I am signing this contract at my own free will, and have initialed each of the above statements because I agree with it, and am willing to adhere to each clause.

Date: 09.09.2014 **Place:** Trondheim

Signatures

Erik Reimer

Pernille Wangsholm

Paul Philip Mitchell

Knut Sie Andersen

Nils Inge Rugsveen

Linda Leidig

Thea Marie Søgaard

Appendix H

Brukermanual for Adresseavisens AdEditor

Innholdsfortegnelse

H.i Brukermanual for Adresseavisens AdEditor

AdEditor er en dynamisk, klient-side WYSIWIG (What You See Is What You Get) editor for å lage dødsannonser. Editoren lar brukeren lage helt nye dødsannonser eller velge en av flere maler som han/hun deretter fyller ut. Alle endringer brukeren gjør i en annonse vises i sanntid gjennom en egen forhåndsvisning.

Dette dokumentet er en håndbok for AdEditor, og her står alt brukeren trenger å vite for å kunne bruke applikasjonen. Fremgangsmåtene og rekkefølgen til stegene blir beskrevet ved hjelp av nummer-symbol som ser slik ut:

H.ii Systemkrav

Brukeren av webapplikasjonen må kjøre applikasjonen i en av følgende nettlesere:

- Internet Explorer versjon 9 eller høyere
- Mozilla Firefox versjon 4 eller høyere
- Opera versjon 5 eller høyere
- Safari versjon 11 eller høyere
- Chrome versjon 30 eller høyere

I tillegg bør brukeren benytte en skjerm med oppløsning på 1280x720 piksler eller høyere. Lavere oppløsning enn dette støttes, men vil ikke være optimalt.

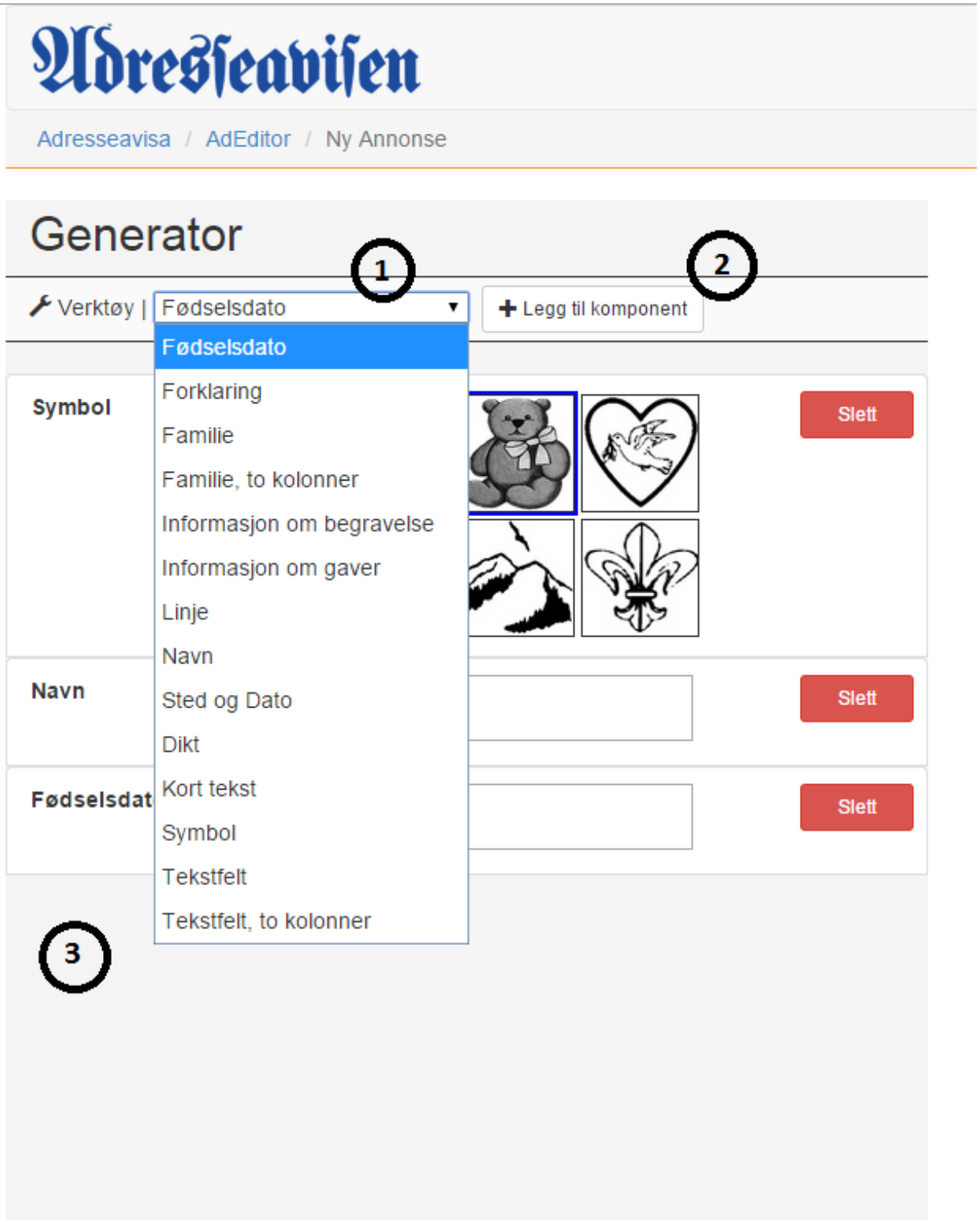
H.1 Oversikt

Brukeren vil bli presentert med følgende skjerm når webapplikasjonen startes.

Applikasjonen er delt inn i to hoveddeler: generator (1) og forhåndsvisning (2). I generatorvinduet finnes det en verktøyboks (3). Her kan brukeren legge til nye komponenter i sin annonse. Det finnes en knapperad nederst på siden. Brukeren kan velge en av flere forhåndsdefinerte maler ved å trykke på "Choose File" (4), og deretter trykke på "Last Opp"-knappen (5). I tillegg har brukeren mulighet til å lagre sin annonse i nåværende tilstand ved å trykke på "Lagre"-knappen (7). Brukeren kan på et hvilket som helst tidspunkt trykke på "Last Opp Lagret Annonse"-knappen (6) for å komme tilbake til den lagrede annonsen. Når brukeren er fornøyd med hvordan annonsen ser ut i forhåndsvisningen (2), kan han/hun trykke på "Godkjenn"-knappen (8) for å sende annonsen til Adresseavisens database for vurdering. Alle komponenter i dette bildet vil bli grundigere forklart i kommende avsnitt

H.2 Legge til nye komponenter

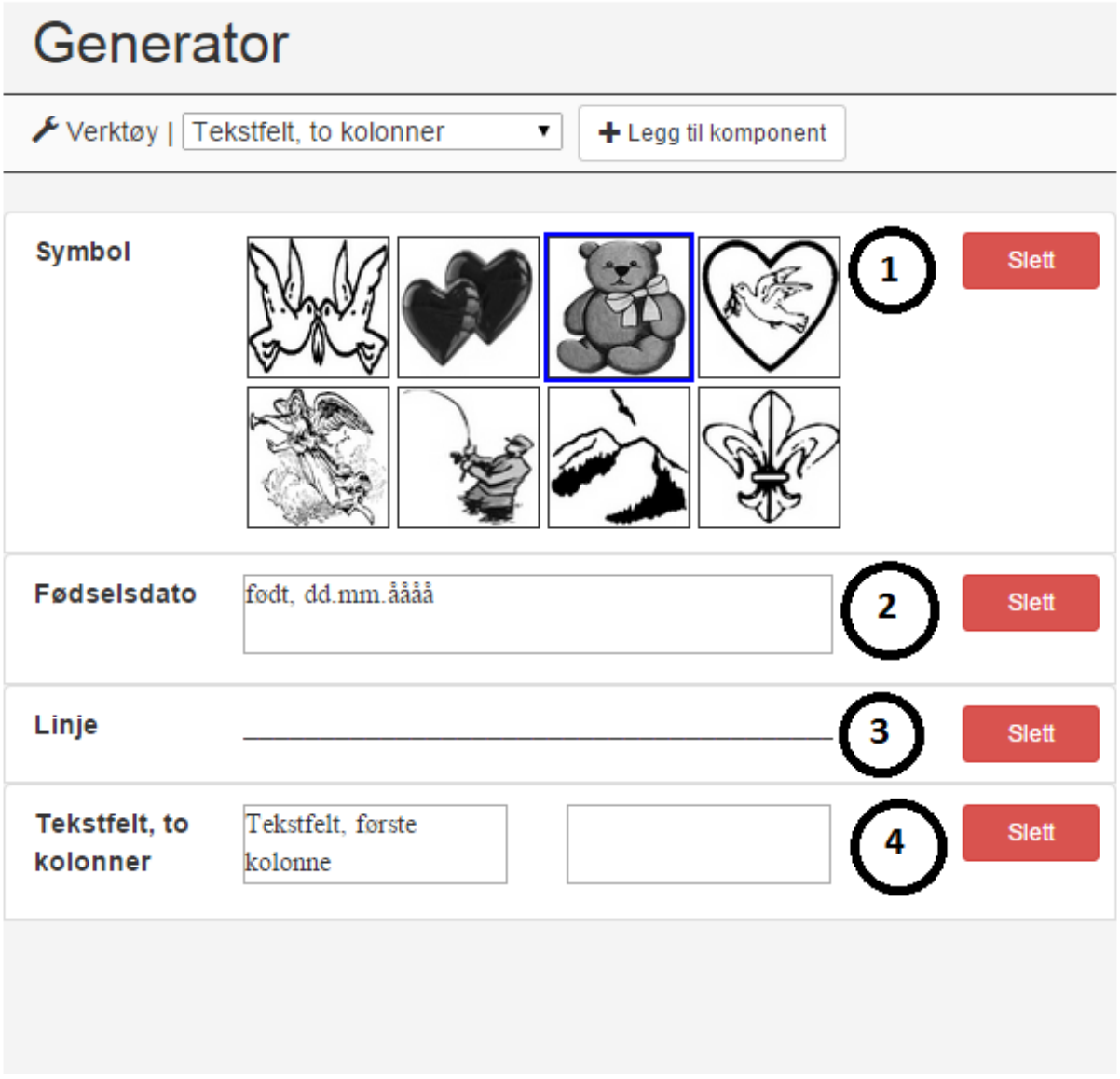
I de kommende bildene vil kun generator-vinduet til applikasjonen vises.



Nye komponenter kan legges til i annonsen ved å trykke på nedtrekkslisten (1) som viser de ulike komponentene brukeren kan legge til. Når brukeren har valgt ønsket komponent, må han/hun trykke på “Legg til komponent”-knappen (2) for å gjennomføre valget. Komponenten

vil da plasseres i listen over komponenter i generator-vinduet. Plasseringen til komponenten vil være nederst som standard (3), men dersom regler fra en valgt mal gjelder, vil plasseringen til komponenten bli satt i henhold til reglene som er spesifisert. Innholdet i komponenten vil umiddelbart vises ferdig formattert i forhåndsvisningen på høyre side.

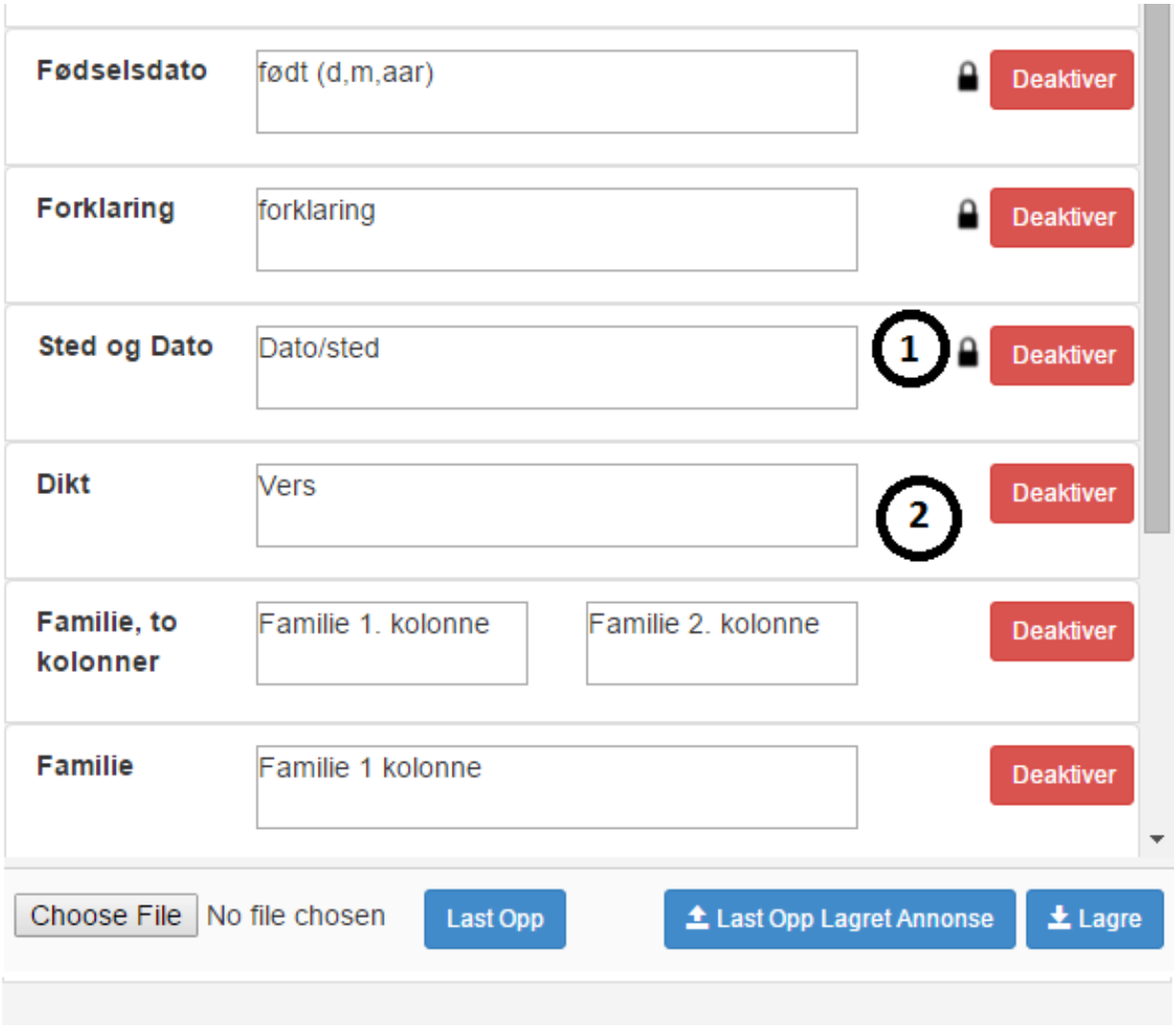
H.2.1 Ulike typer komponenter



Det finnes fire ulike typer komponenter: symbol (1), én-kolonne tekstfelt (2), linje (3) og to-kolonne tekstfelt (4). Etter at brukeren har lagt til en symbol-komponent ved å bruke nedtrekkslisten, kan symbolet som skal brukes spesifiseres. Dette gjøres ved å klikke på ønsket symbol. Den

blå rammen rundt et symbol indikerer at dette er det aktive symbolet. Innholdet i én-kolonne tekstfelter blir midtstilt i forhåndsvisningen som standard. Linjer er seperatorer som er ment for å separere en annonse fra en annen. Derfor er det en margin fra en linje til neste komponent i forhåndsvisningen. Innholdet i hver kolonne i to-kolonne tekstfelter blir midtstilt hver for seg i forhåndsvisningen som standard. To-kolonne komponentene er ment å brukes for å for eksempel liste nære familiemedlemmer.

H.3 Omorganisere komponenter

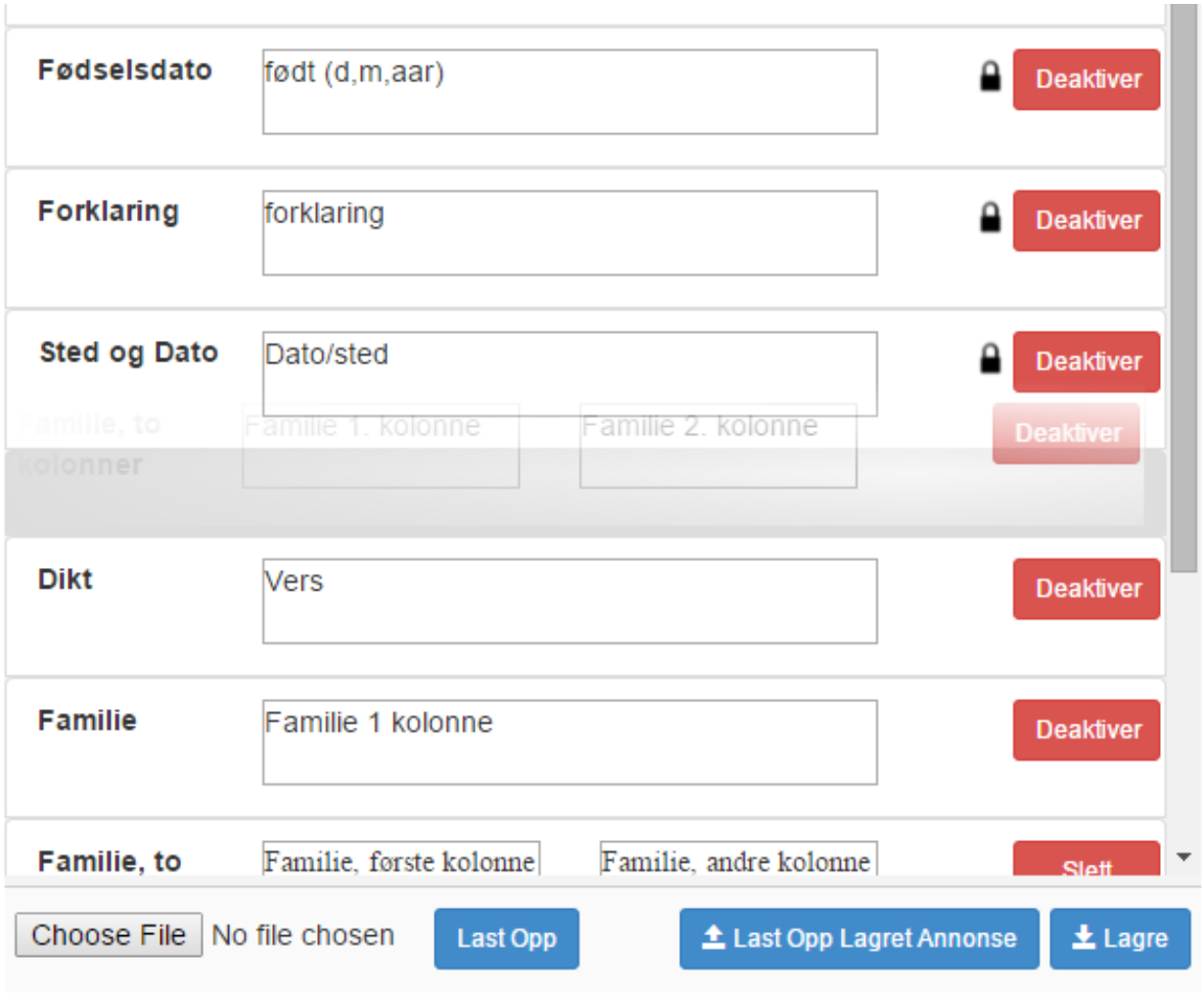


Komponentene i generator-vinduet kan omorganiseres for å endre rekkefølgen i forhåndsvisningen. Hvis brukeren lager en ny annonse uten å bruke en mal, står han/hun fritt til å organisere komponentene slik han/hun vil. På den andre siden, dersom brukeren velger én av de

forhåndsdefinerte malene, vil det foreligge restriksjoner på organiseringen av feltene. For å indikere at en komponent ikke kan flyttes, brukes et låse-ikon (1). Komponenter som er flyttbare (2) vil ikke vise dette låse-ikonet. I tillegg vil musepekeren se slik ut

når brukeren holder pekeren over en ikke-flyttbar komponent, mens den endrer seg til

dersom komponenten er flyttbar. Flyttbare komponenter kan enkelt flyttes ved å klikke og dra komponenten opp eller ned, slik som vist under.



Her blir "Familie, to-kolonner" komponenten, som tidligere var rett under "Dikt", dratt og flyttet til rett over "Dikt". Et grått rektangel indikerer at komponenten kan flyttes hit.

H.4 Slette, aktivere og deaktivere komponenter

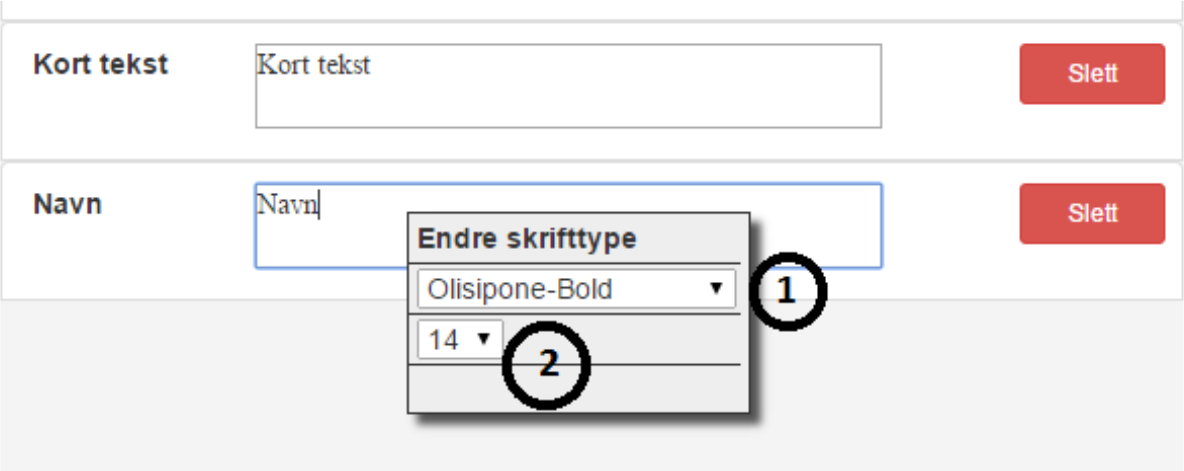
Avhengig av om en komponent kommer fra en mal eller om brukeren har lagt den til på egenhånd, vil knappen på høyre side endre seg både visuelt og atferdsmessig.

Dersom brukeren laster inn en forhåndsdefinert mal, vil alle komponentene som følger med ha en "Deaktiver"-knapp (1). Komponenter som brukeren selv legger til, vil ha en "Slett"-knapp (2).

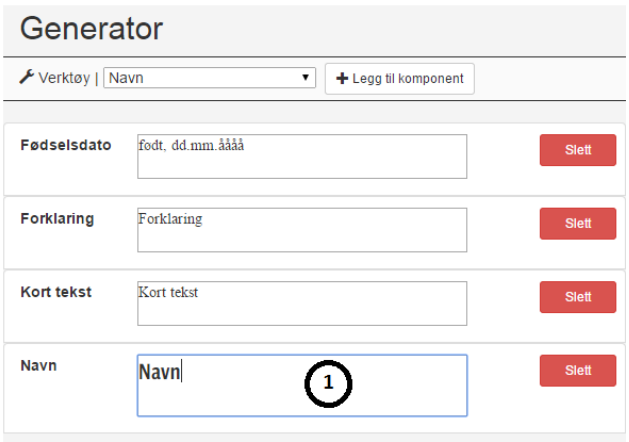
Dersom brukeren ønsker, kan en komponent fra malen "gjemmes" ved å klikke "Deaktiver"-knappen. Innholdet til komponenten vil gjemmes, høyden vil minke, bakgrunnen vil endres til stripete grå og knappen vil bli grønn og endres til "Aktiver" for å indikere at dette feltet er deaktivert. Deaktiverte felter vil også fjernes fra forhåndsvisningen. Dersom brukeren nå klikker "Aktiver" vil komponenten igjen komme til syne på riktig plass, både i generator-vinduet og forhåndsvisningen.

Komponenter som brukeren selv har lagt til vil ikke kunne aktiveres eller deaktiveres, kun slettes. Dersom brukeren klikker på "Slett"-knappen vil komponenten fjernes fra både generator-vinduet og forhåndsvisningen, og dersom brukeren vil ha komponenten allikevel, må den legges til på nytt ved hjelp av nedtrekkslisten i verktøyboksen.

H.5 Endre skrifttype og skriftstørrelse



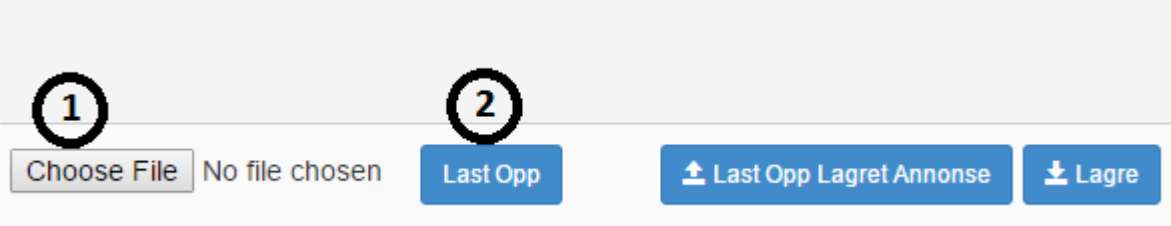
Skrifttypen og skriftstørrelsen til en komponent kan enkelt endres ved å høyreklikke på komponenten. Brukeren vil da få opp en meny som erstatter den standard høyreklikk-menyen til nettleseren. I menyen kan brukeren endre skrifttype ved å velge et av elementene i den første nedtrekkslisten (1), eller skriftstørrelse ved å velge et av elementene i den andre nedtrekkslisten (2).



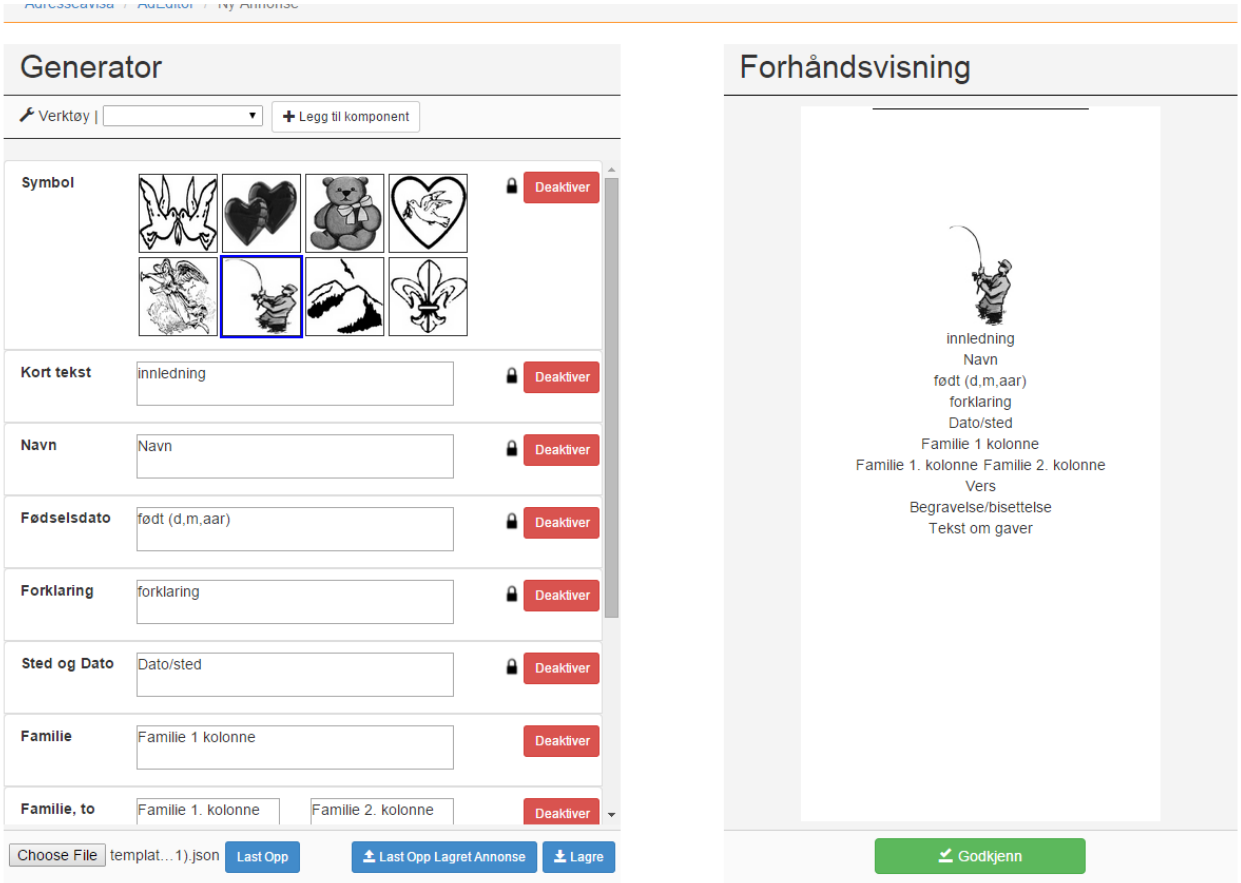
I dette bildet er skrifttypen og skriftstørrelsen til "Navn" endret til henholdsvis "Olisipone Head Demi" og 20. Både innholdet i komponenten i generator-vinduet (1) og forhåndsvisningen (2) oppdateres i sanntid for å speile endringene som blir gjort på skriften.

H.6 Laste inn forhåndsdefinert mal

En av hovedfunksjonene til AdEditor er forhåndsdefinerte maler. Malene inneholder et antall komponenter som har visse regler knyttet til seg. Hensikten med malene er at brukeren kan raskt lage en annonse uten å måtte tenke på normer og regler knyttet til dødsannonser.



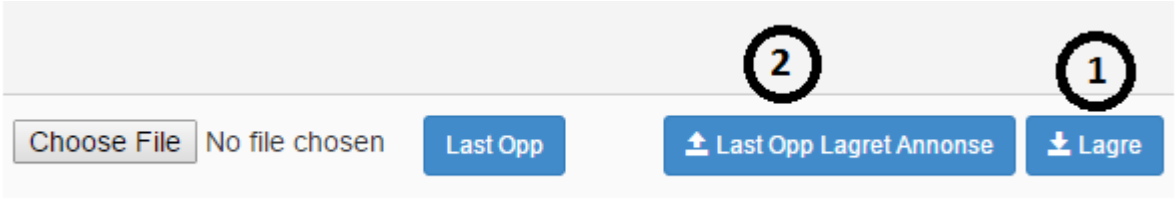
Brukeren kan laste inn en mal ved å trykke på "Choose File"-knappen (1). Brukeren vil da bli presentert alle tilgjengelige maler som Adresseavisen allerede har definert. Han/hun kan så velge den malen som ønskes. Når en mal har blitt valgt, trykker brukeren på "Last Opp"-knappen (2) for å laste inn komponentene i applikasjonen.



I skjermbildet ovenfor har brukeren lastet inn en forhåndsdefinert mal. Alle komponenter som hører til i malen vises i applikasjonen, og alle regler knyttet til komponentene er håndhevet. Brukeren kan, hvis ønskelig, legge til flere komponenter ved å benytte seg av verktøyboksen.

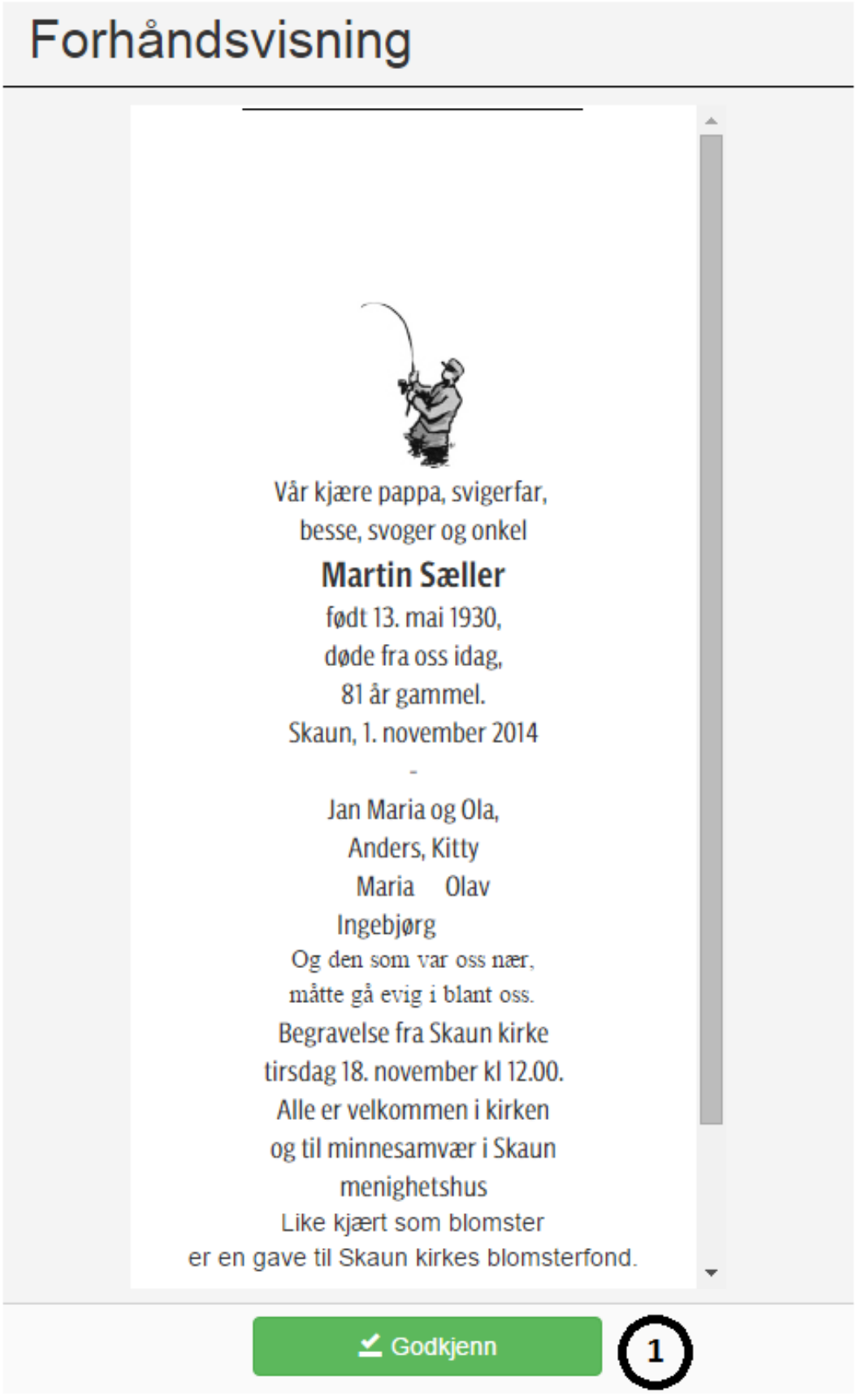
H.7 Lagre annonser og laste inn lagrede annonser

I tillegg til å kunne laste inn forhåndsdefinerte maler, har brukeren også mulighet til å laste inn tidligere iterasjoner av sine annonser.



Brukeren kan på et hvilket som helst tidspunkt klikke på “Lagre”-knappen (1) i generator-vinduet. Annonsen vil da bli lagret på Adresseavisens servere, i nøyaktig den tilstanden annonsen er i når brukeren trykker på “Lagre”-knappen. Dersom brukeren så gjør noen endringer på annonsen, men bestemmer seg for å forkaste disse endringene, kan han/hun trykke på “Last Opp Lagret Annonse”-knappen (2) for å komme tilbake til samme annonsetilstand som da “Lagre”-knappen ble trykket.

H.8 Fullføre annonsen



Når brukeren er fornøyd med annonsen, trykker han/hun på “Godkjenn”-knappen (1). Dødsannonsen vil da bli sendt til Adresseavisens servere i korrekte formater. Annonsen vil deretter bli vurdert av Adresseavisen, og dersom alt er som det skal, blir annonsen printet i avisen innen noen dager.

Appendix I

Brukermanual for IT-avdelingen til bruk for Adresseavisens AdEditor

Innholdsfortegnelse

I.i Brukermanual for IT-avdelingen til bruk for Adresseavisens AdEditor

AdEditor er en dynamisk, klient-side WYSIWIG (What You See Is What You Get) editor for å lage dødsannonser. Editoren lar brukeren lage helt nye dødsannonser eller velge en av flere maler som han/hun deretter fyller ut. Alle endringer brukeren gjør i en annonse vises i sanntid gjennom en egen forhåndsvisning. IT-avdelingens ansatte har muligheten til å lage nye maler i editoren. IT-avdelingens ansatte vil i dette dokumentet forkortes til IT-ansatte.

Dette dokumentet er en håndbok for AdEditor, og her står alt brukeren trenger å vite for å kunne bruke applikasjonen. Fremgangsmåtene og rekkefølgen til stegene blir beskrevet ved hjelp av nummer-symbol som ser slik ut:

I.ii Systemkrav

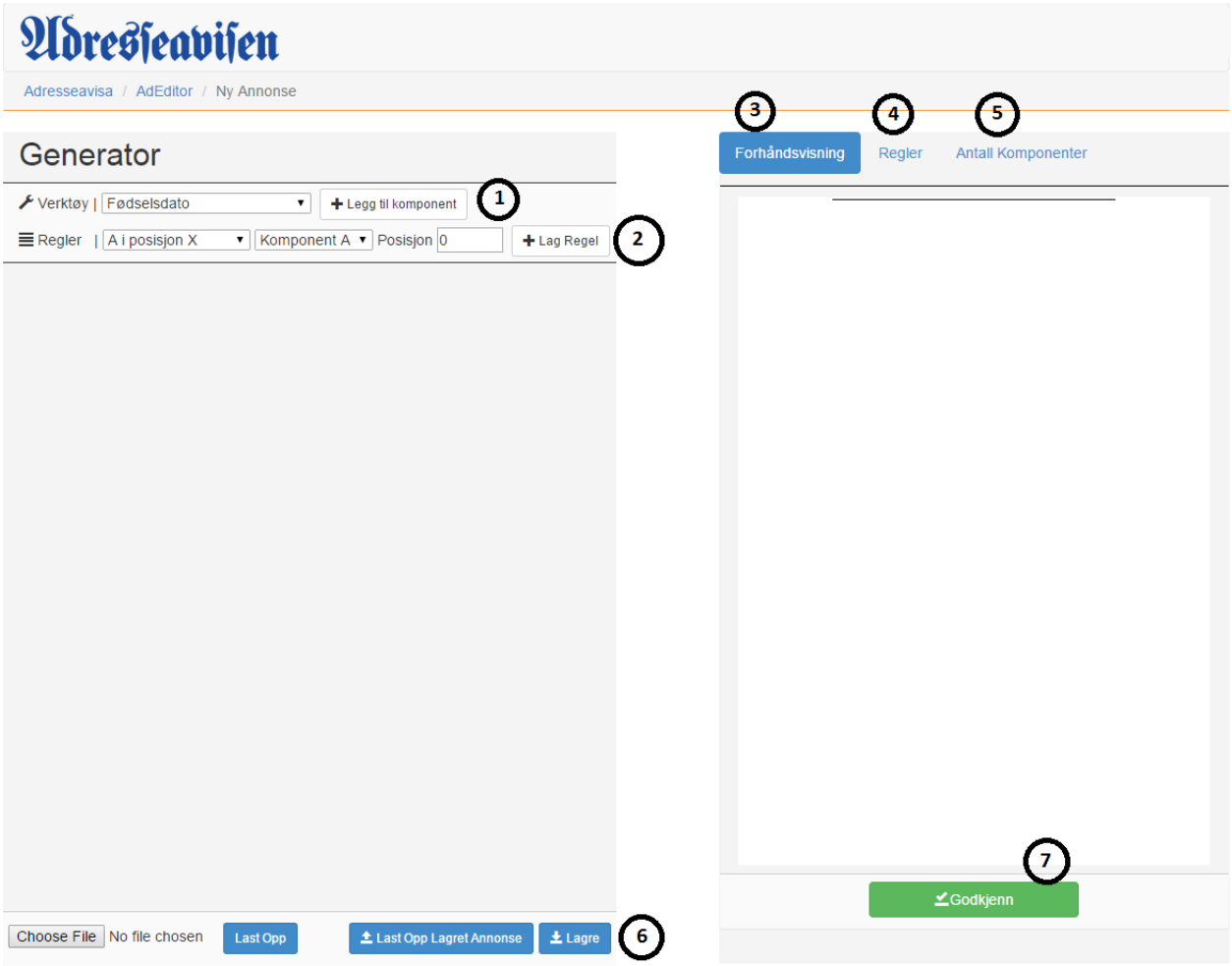
Brukeren av webapplikasjonen må kjøre applikasjonen i en av følgende nettlesere:

- Internet Explorer versjon 9 eller høyere
- Mozilla Firefox versjon 4 eller høyere
- Opera versjon 5 eller høyere
- Safari versjon 11 eller høyere
- Chrome versjon 30 eller høyere

I tillegg bør brukeren benytte en skjerm med oppløsning på 1280x720 piksler eller høyere. Lavere oppløsning enn dette støttes, men vil ikke være optimalt.

I.1 Oversikt

IT-ansatte vil bli presentert med følgende skjerm når webapplikasjonen startes.

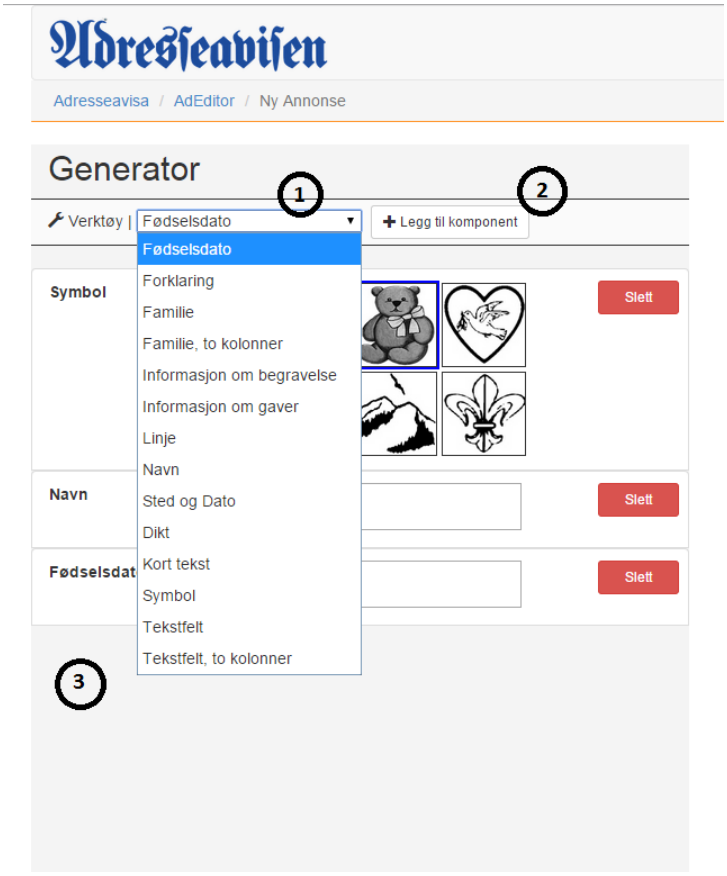


Applikasjonen er delt inn i to hoveddeler: generator-vinduet på venstre side og et fanebasert vindu på høyre side som inneholder forhåndsvisning (3), regler (4) og antall komponenter (5). I generatorvinduet finnes det en verktøyboks (1) og en regelboks (2). Her kan brukeren legge til nye komponenter eller regler i sin annonse. Det finnes en knapperad nederst på siden (6). IT-ansatte kan velge en av flere forhåndsdefinerte maler ved å trykke på "Choose File", og deretter trykke på "Last Opp"-knappen. Deretter kan man gjøre endringer på malen for å skape en ny mal. I tillegg har IT-ansatte mulighet til å lagre sin annonse i nåværende tilstand ved å trykke på "Lagre"-knappen. Han/hun kan på et hvilket som helst tidspunkt trykke på "Last Opp Lagret Annonse"-knappen for å komme tilbake til den lagrede annonsen. Når IT-ansatte er fornøyd med hvordan annonsen eller malen ser ut i forhåndsvisningen (3), kan han/hun trykke på "Godkjenn"-knappen (7) for å sende malen til Adreseavisens database slik at sluttbrukere kan ta malen i bruk. Alle gjeldende regler som IT-ansatte har definert for en gitt mal vises i "Regler"-fanen (4), mens spesifikasjoner for antall tillatte komponenter for denne malen vises i "Antall Komponenter"-fanen (5). Alle komponenter i dette bildet vil bli grundigere forklart i

kommende avsnitt.

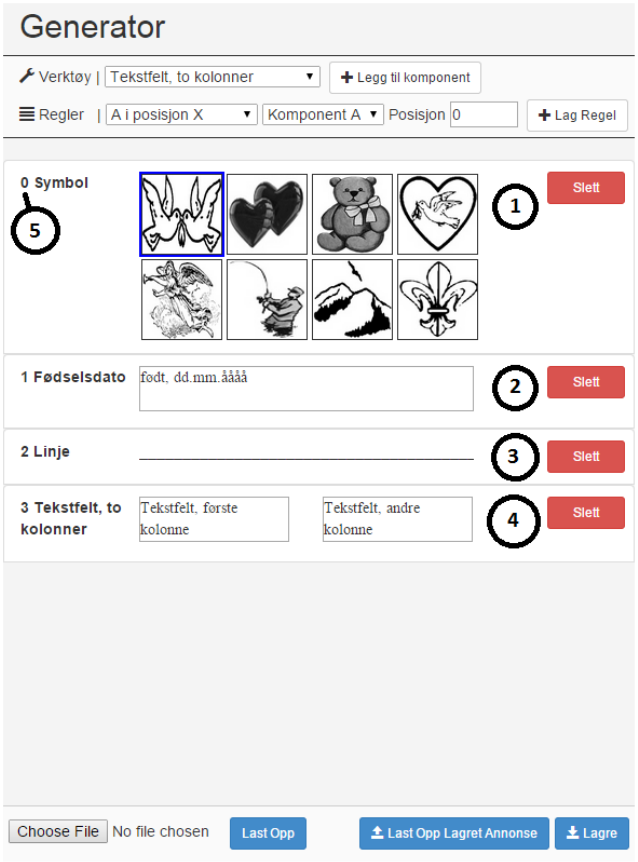
I.2 Legge til nye komponenter

I de kommende bildene vil kun generator-vinduet til applikasjonen vises.



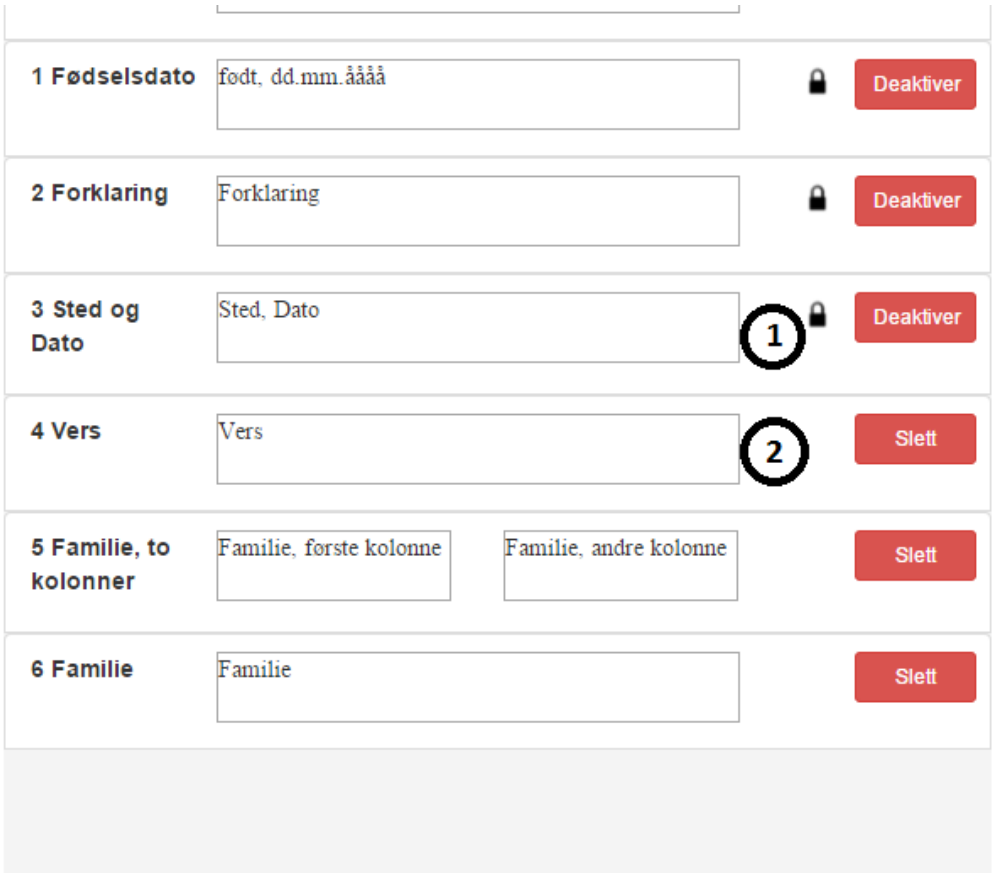
Nye komponenter kan legges til i annonsen ved å trykke på nedtrekkslisten (1) som viser de ulike komponentene som kan legges til. Når brukeren har valgt ønsket komponent, må han/hun trykke på "Legg til komponent"-knappen (2) for å gjennomføre valget. Komponenten vil da plasseres i listen over komponenter i generator-vinduet. Plasseringen til komponenten vil være nederst som standard (3), men dersom regler fra en valgt mal gjelder, vil plasseringen til komponenten bli satt i henhold til reglene som er spesifisert. Innholdet i komponenten vil umiddelbart vises ferdig formattert i "Forhåndsvisning"-fanen på høyre side.

I.2.1 Ulike typer komponenter



Det finnes fire ulike typer komponenter: symbol (1), én-kolonne tekstfelt (2), linje (3) og to-kolonne tekstfelt (4). Etter at brukeren har lagt til en symbol-komponent ved å bruke nedtrekkslisten, kan symbolet som skal brukes spesifiseres. Dette gjøres ved å klikke på ønsket symbol. Den blå rammen rundt et symbol indikerer at dette er det aktive symbolet. Innholdet i én-kolonne tekstfelter blir midtstilt i forhåndsvisningen som standard. Linjer er seperatorer som er ment for å separere en annonse fra en annen. Derfor er det en margin fra en linje til neste komponent i forhåndsvisningen. Innholdet i hver kolonne i to-kolonne tekstfelter blir midtstilt hver for seg i forhåndsvisningen som standard. To-kolonne komponentene er ment å brukes for å for eksempel liste nære familiemedlemmer. Alle komponenter som legges til vil tilegnes en unik ID (5) ut ifra når komponenten ble lagt til.

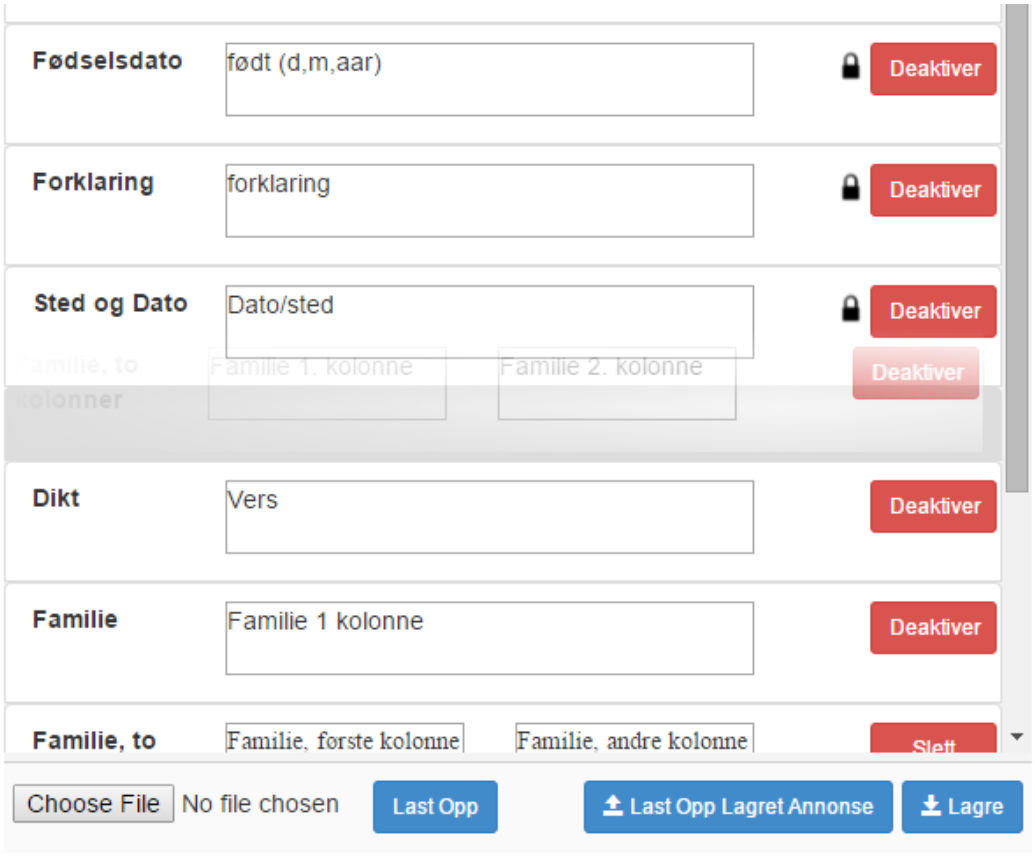
I.3 Omorganisere komponenter



Komponentene i generator-vinduet kan omorganiseres for å endre rekkefølgen i forhåndsvisningen. Hvis brukeren lager en ny annonse uten å bruke en mal, står han/hun fritt til å organisere komponentene slik han/hun vil. På den andre siden, dersom brukeren velger én av de forhåndsdefinerte malene, vil det foreligge restriksjoner på organiseringen av feltene. For å indikere at en komponent ikke kan flyttes, brukes et låse-ikon (1). Komponenter som er flyttbare (2) vil ikke vise dette låse-ikonet. I tillegg vil musepekeren se slik ut

når brukeren holder pekeren over en ikke-flyttbar komponent, mens den endrer seg til

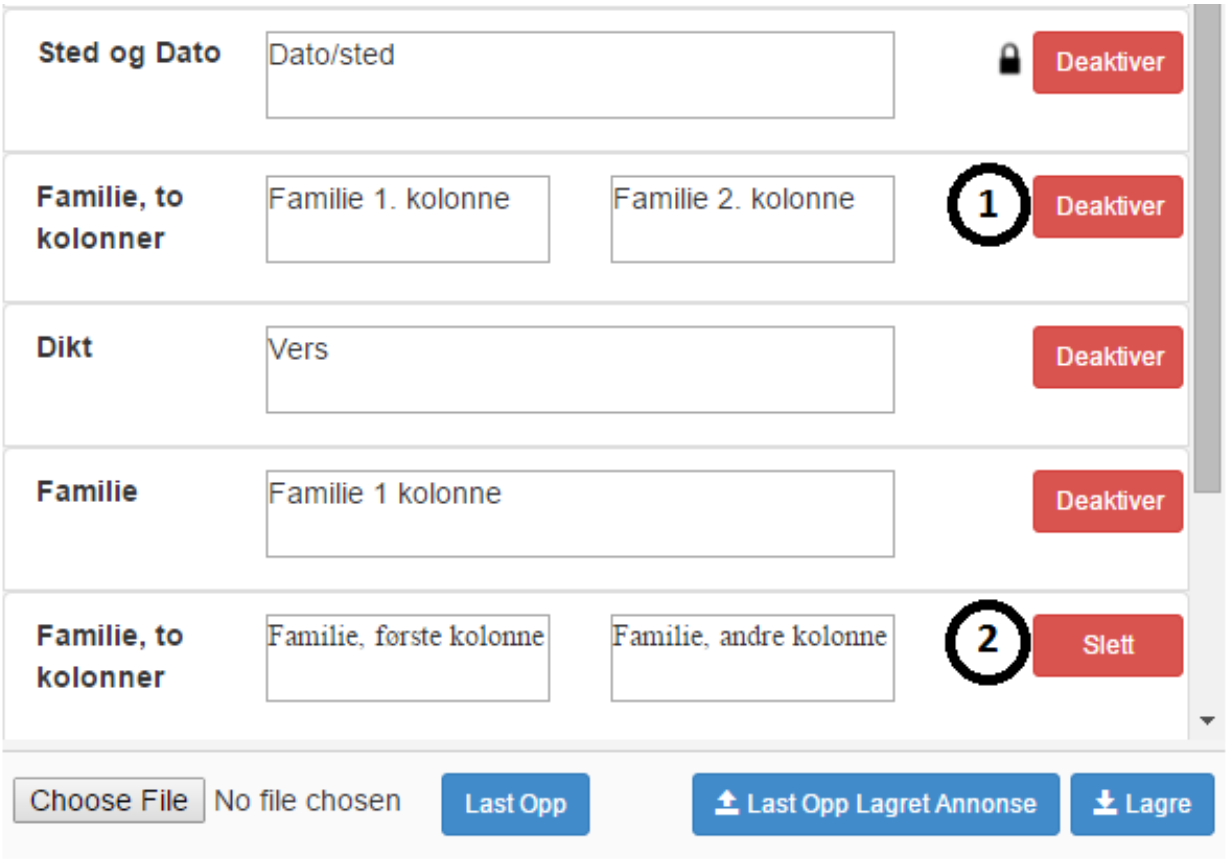
dersom komponenten er flyttbar. Flyttbare komponenter kan enkelt flyttes ved å klikke og dra komponenten opp eller ned, slik som vist under.



Her blir "Familie, to-kolonner" komponenten, som tidligere var rett under "Dikt", dratt og flyttet til rett over "Dikt". Et grått rektangel indikerer at komponenten kan flyttes hit.

I.4 Slette, aktivere og deaktivere komponenter

Avhengig av om en komponent kommer fra en mal eller om brukeren har lagt den til på egenhånd, vil knappen på høyre side endre seg både visuelt og atferdsmessig



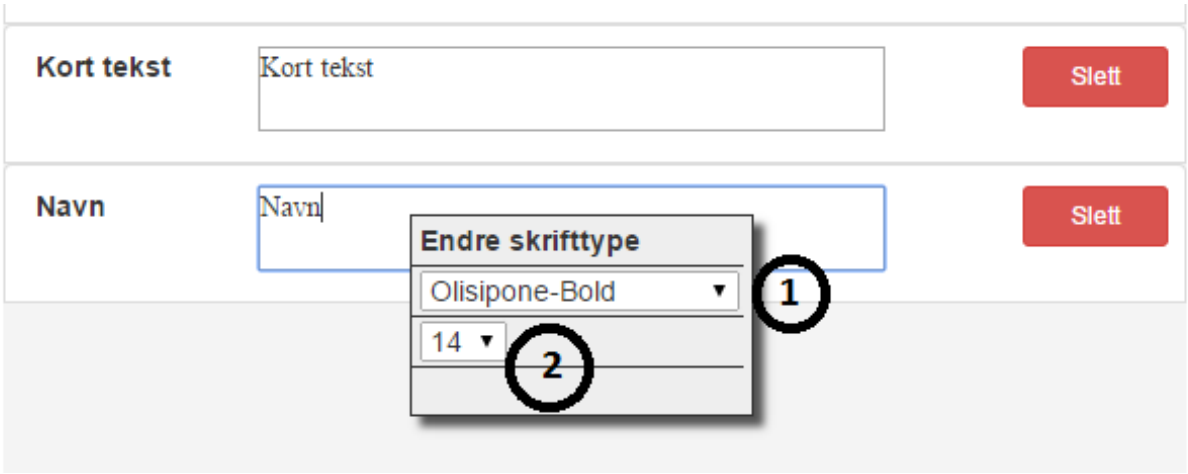
Dersom brukeren laster inn en forhåndsdefinert mal, vil alle komponentene som følger med ha en "Deaktiver"-knapp (1). Komponenter som brukeren selv legger til, vil ha en "Slett"-knapp (2).

Dersom brukeren ønsker, kan en komponent fra malen "gjemmes" ved å klikke "Deaktiver"-knappen. Innholdet til komponenten vil gjemmes, høyden vil minke, bakgrunnen vil endres til stripete grå og knappen vil bli grønn og endres til "Aktiver" for å indikere at dette feltet er deaktivert. Deaktiverte felter vil også fjernes fra forhåndsvisningen. Dersom brukeren nå klikker

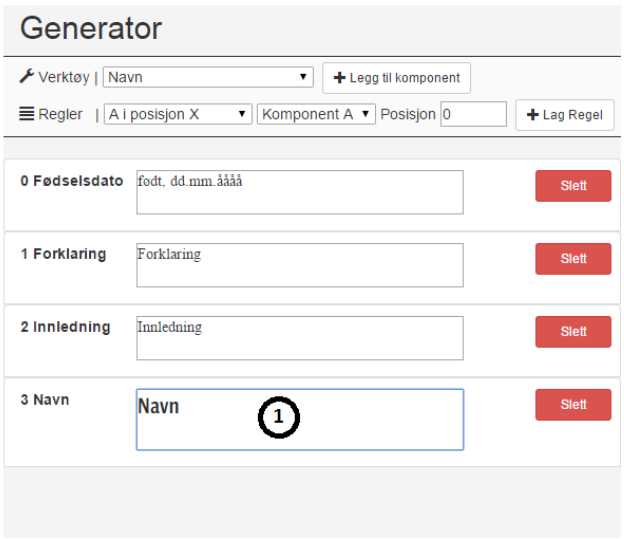
"Aktiver" vil komponenten igjen komme til syne på riktig plass, både i generator-vinduet og forhåndsvisningen.

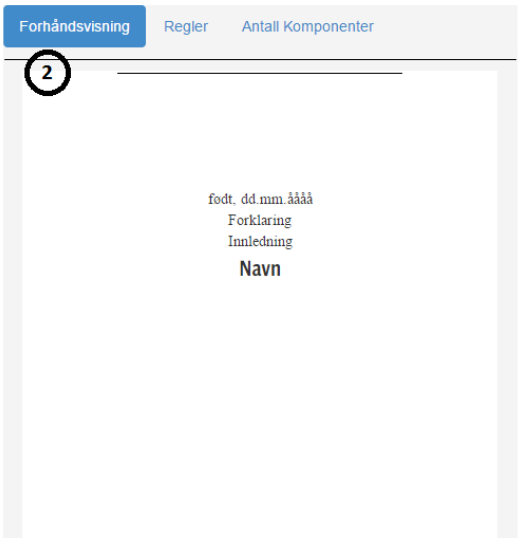
Komponenter som brukeren selv har lagt til vil ikke kunne aktiveres eller deaktiveres, kun slettes. Dersom brukeren klikker på "Slett"-knappen vil komponenten fjernes fra både generator-vinduet og forhåndsvisningen, og dersom brukeren vil ha komponenten allikevel, må den legges til på nytt ved hjelp av nedtrekkslisten i verktøyboksen.

I.5 Endre skrifttype og skriftstørrelse



Skrifttypen og skriftstørrelsen til en komponent kan enkelt endres ved å høyreklikke på komponenten. Brukeren vil da få opp en meny som erstatter den standard høyreklikk-menyen til nettleseren. I menyen kan brukeren endre skrifttype ved å velge et av elementene i den første nedtrekkslisten (1), eller skriftstørrelse ved å velge et av elementene i den andre nedtrekkslisten (2).

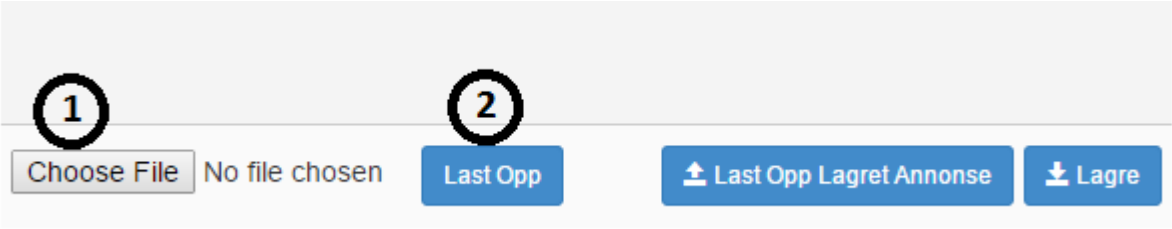




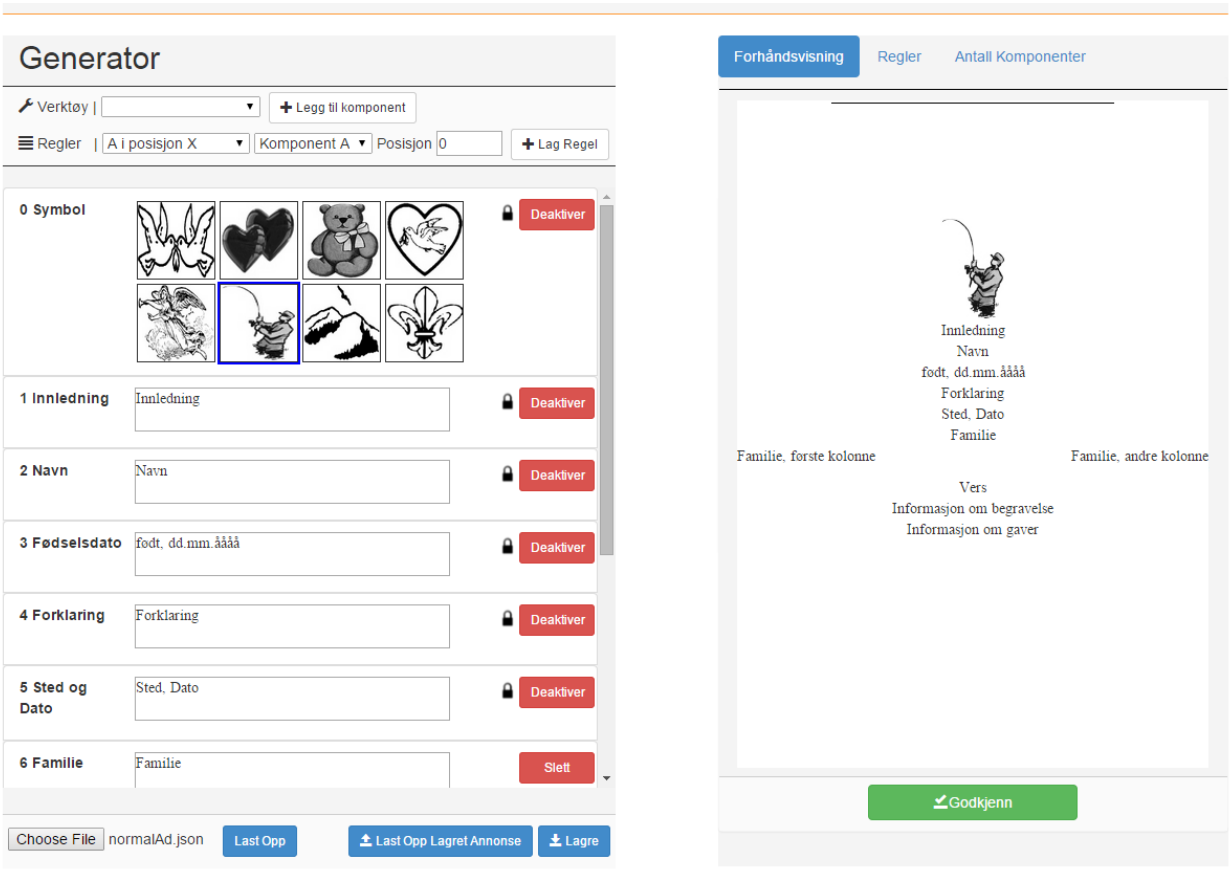
I dette bildet er skrifttypen og skriftstørrelsen til "Navn" endret til henholdsvis "Olisipone Head Demi" og 20. Både innholdet i komponenten i generator-vinduet (1) og forhåndsvisningen (2) oppdateres i sanntid for å speile endringene som blir gjort på skriften.

I.6 Laste inn forhåndsdefinert mal

En av hovedfunksjonene til AdEditor er forhåndsdefinerte maler. Malene inneholder et antall komponenter som har visse regler knyttet til seg. Hensikten med malene er at brukeren kan raskt lage en annonse uten å måtte tenke på normer og regler knyttet til dødsannonser.



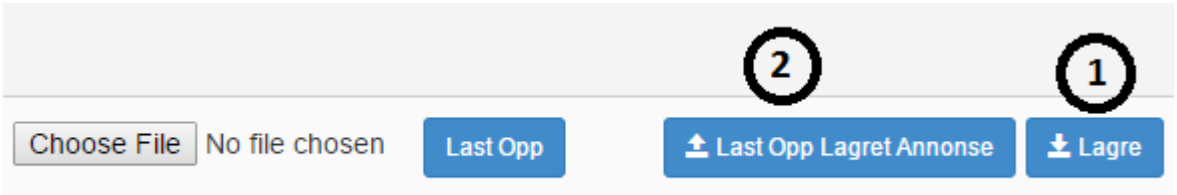
Brukeren kan laste inn en mal ved å trykke på "Choose File"-knappen (1). Brukeren vil da bli presentert alle tilgjengelige maler som Adresseavisen allerede har definert. Han/hun kan så velge den malen som ønskes. Når en mal har blitt valgt, trykker brukeren på "Last Opp"-knappen (2) for å laste inn komponentene i applikasjonen.



I skjermbildet ovenfor har brukeren lastet inn en forhåndsdefinert mal. Alle komponenter som hører til i malen vises i applikasjonen, og alle regler knyttet til komponentene er håndhevet. Brukeren kan, hvis ønskelig, legge til flere komponenter ved å benytte seg av verktøyboksen.

I.7 Lagre annonser og laste inn lagrede annonser

I tillegg til å kunne laste inn forhåndsdefinerte maler, har brukeren også mulighet til å laste inn tidligere iterasjoner av sine annonser.

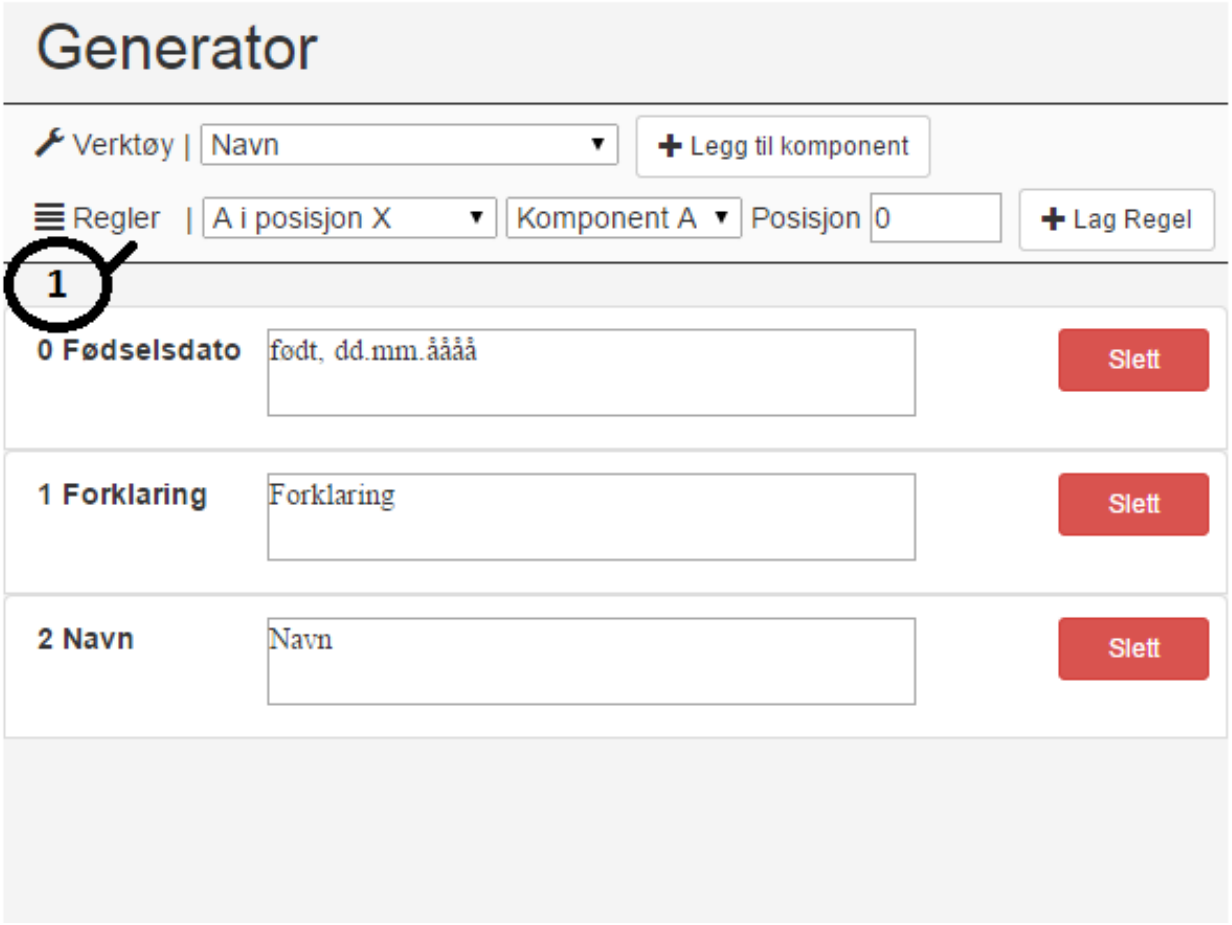


Brukeren kan på et hvilket som helst tidspunkt klikke på “Lagre”-knappen (1) i generator-

vinduet. Annonsen vil da bli lagret på Adresseavisens servere, i nøyaktig den tilstanden annonsen er i når brukeren trykker på "Lagre"-knappen. Dersom brukeren så gjør noen endringer på annonsen, men bestemmer seg for å forkaste disse endringene, kan han/hun trykke på "Last Opp Lagret Annonse"-knappen (2) for å komme tilbake til samme annonsetilstand som da "Lagre"-knappen ble trykket.

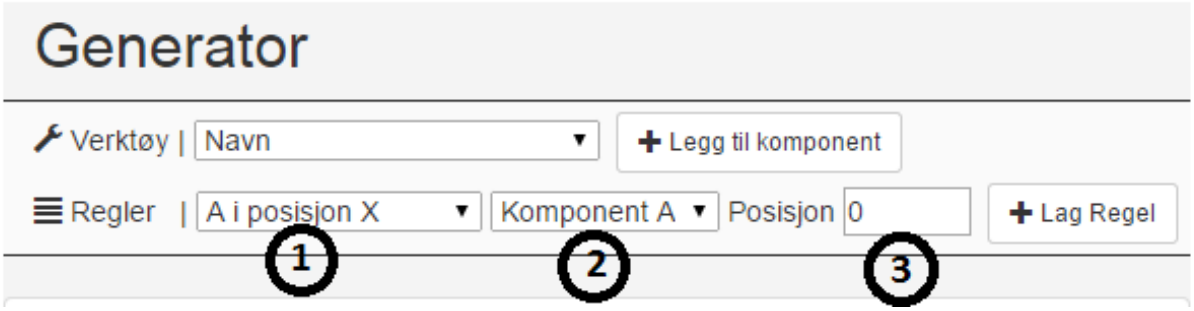
I.8 Definere regler for komponenter

IT-ansatte har mulighet til å definere regler for komponentene i maler for å begrense hva sluttbrukeren kan gjøre, slik at feil kan forhindres før sluttbrukeren sender inn annonser til vurdering.



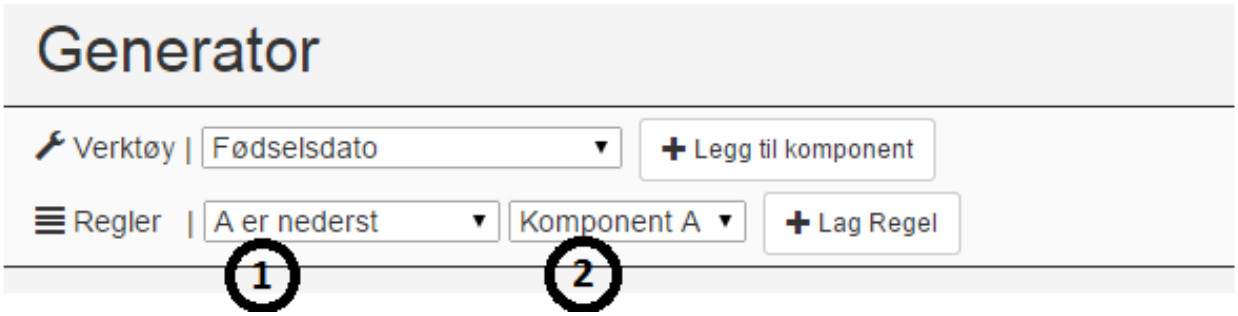
Regler defineres ved å benytte seg av regelboksen (1) øverst i generator-vinduet. Regelboksen endres basert på hva slags type regel IT-ansatte vil legge til. Det finnes tre forskjellige former for regler: regler som sier at en komponent skal være på en spesifikk plass, regler som sier at én

komponent må være før eller etter en annen komponent, og regler som sier at en komponent må enten være plassert øverst eller nederst.



Slik ser regelboksen ut når valgt regel i nedtrekkslisten (1) er av typen én komponent på en spesifikk plass. Den andre nedtrekkslisten (2) inneholder alle ID-ene til komponentene som er lagt til. Her velger man hvilken komponent man ønsker å definere en regel for, og velger deretter hvilken posisjon komponenten skal være i ved å spesifisere et tall i posisjon-boksen (3).

Slik ser regelboksen ut når valgt regel i nedtrekkslisten (1) er av typen én komponent må være før eller etter en annen komponent. Den andre og tredje nedtrekkslisten (2 og 3) inneholder alle ID-ene til komponentene som er lagt til. I den andre nedtrekkslisten (2) velger man hvilken komponent som skal være før eller etter en annen komponent (som velges i den tredje nedtrekkslisten (3)).



Slik ser regelboksen ut når valgt regel i nedtrekkslisten (1) er av typen en komponent må være nederst eller øverst. Komponenten som skal være øverst eller nederst velges i den andre nedtrekkslisten (2).

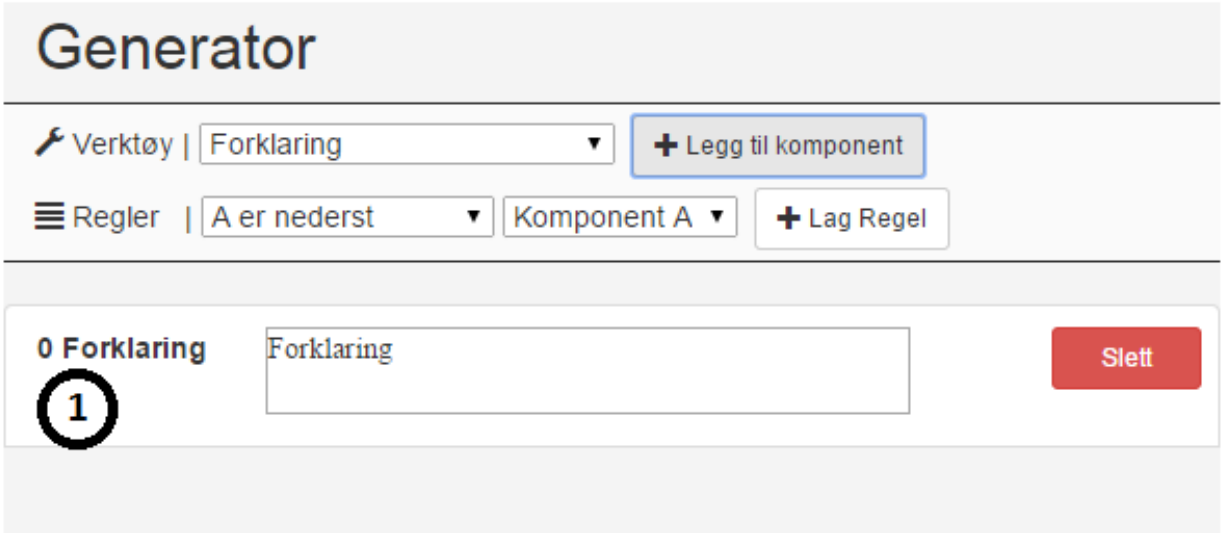
I.8.1 Ulike typer regler

Alle mulige regler er oppsummert i følgende tabell.

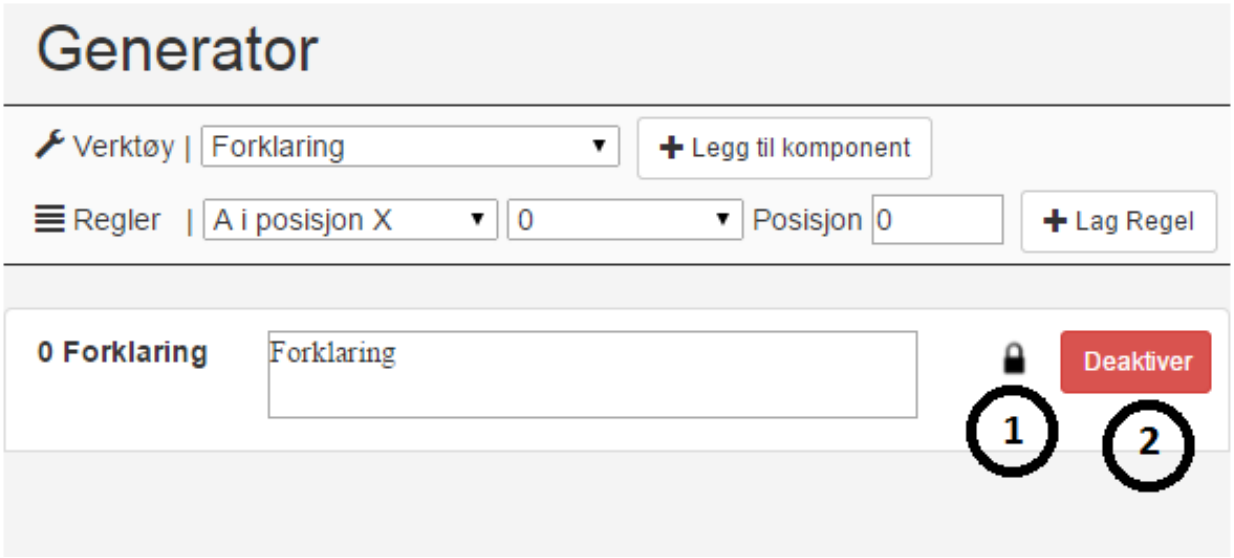
Rule	**Explanation**
A at position X	Component A has to be at position X, where X is a number in the list.
A not at position X	Component A is not allowed to be at position X, where X is a number in the list.
A before B	Component A has to be in front of component B, i.e. the index of component A in the list of components has to be smaller than the index of component B.
A directly before B	Component A has to be directly in front of component B, i.e. the index of component A in the list of components has to be one smaller than the index of component B.
A not directly before B	Component A is not allowed to be directly before component B, i.e. the index of component A in the list of components must not be one smaller than the index of component B.
A is bottom	Component A has to be at the last position of the list of components.
A not bottom	Component A must not be at the last position of the list of components.

I.8.2 Hvordan komponenter endrer seg basert på definerte regler

I tillegg til at regelboksen endres ut i fra hva slags type regel som er valgt, vil også komponenter endre seg visuelt når regler er definert for de aktuelle komponentene.



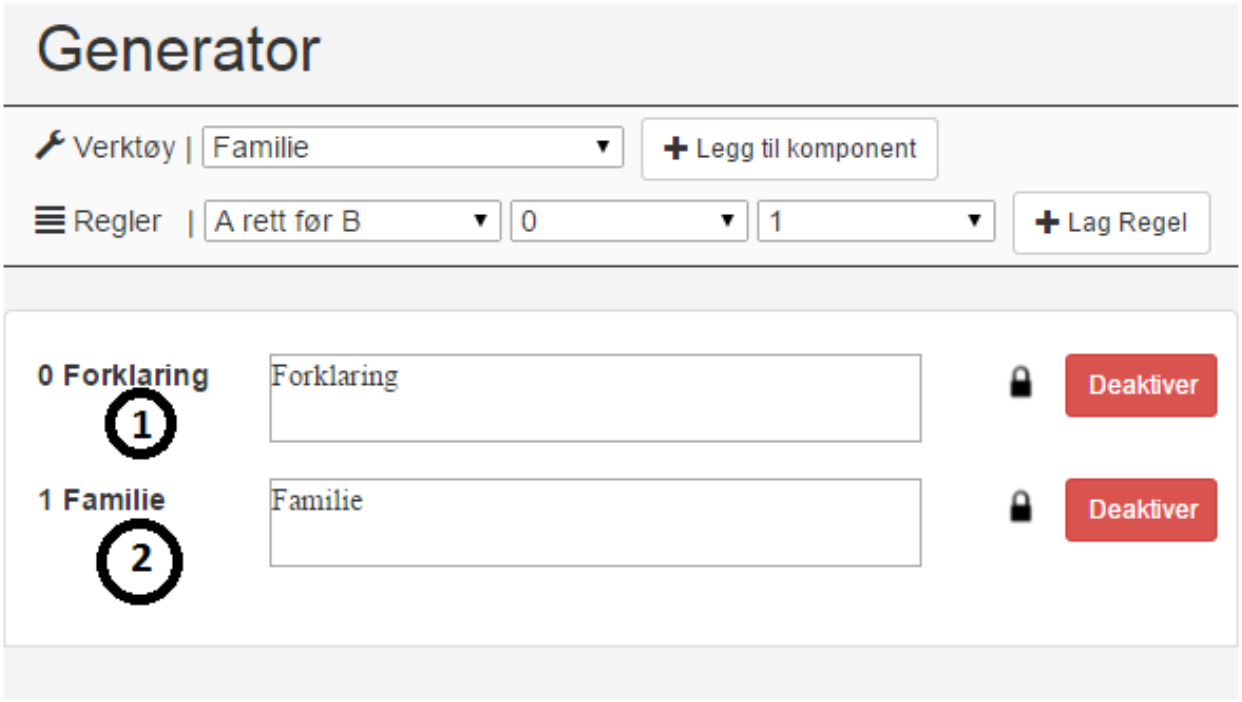
I eksempelet ovenfor er det ikke definert noen regler for "Forklaring"-komponenten (1). Dersom det legges til en regel om at denne komponenten alltid skal være i posisjon 0, vil komponenten se slik ut:



Når regelen om at "Forklaring"-komponenten alltid skal være i posisjon 0 er definert, betyr det også at komponenten aldri kan skifte posisjon. Komponenter som ikke er flyttbare, slik som den ovenfor, vil vise et låse-ikon (1) for å indikere at den er låst. I tillegg, siden det er spesifisert at "Forklaring"-komponenten alltid må være i en viss posisjon, impliserer det at komponenten alltid må eksistere for å opprettholde indeksen til listen. "Slett"-knappen vil dermed endres til å bli "Deaktiver" (2), siden denne komponenten må inkluderes i malen som sluttbrukeren tar i

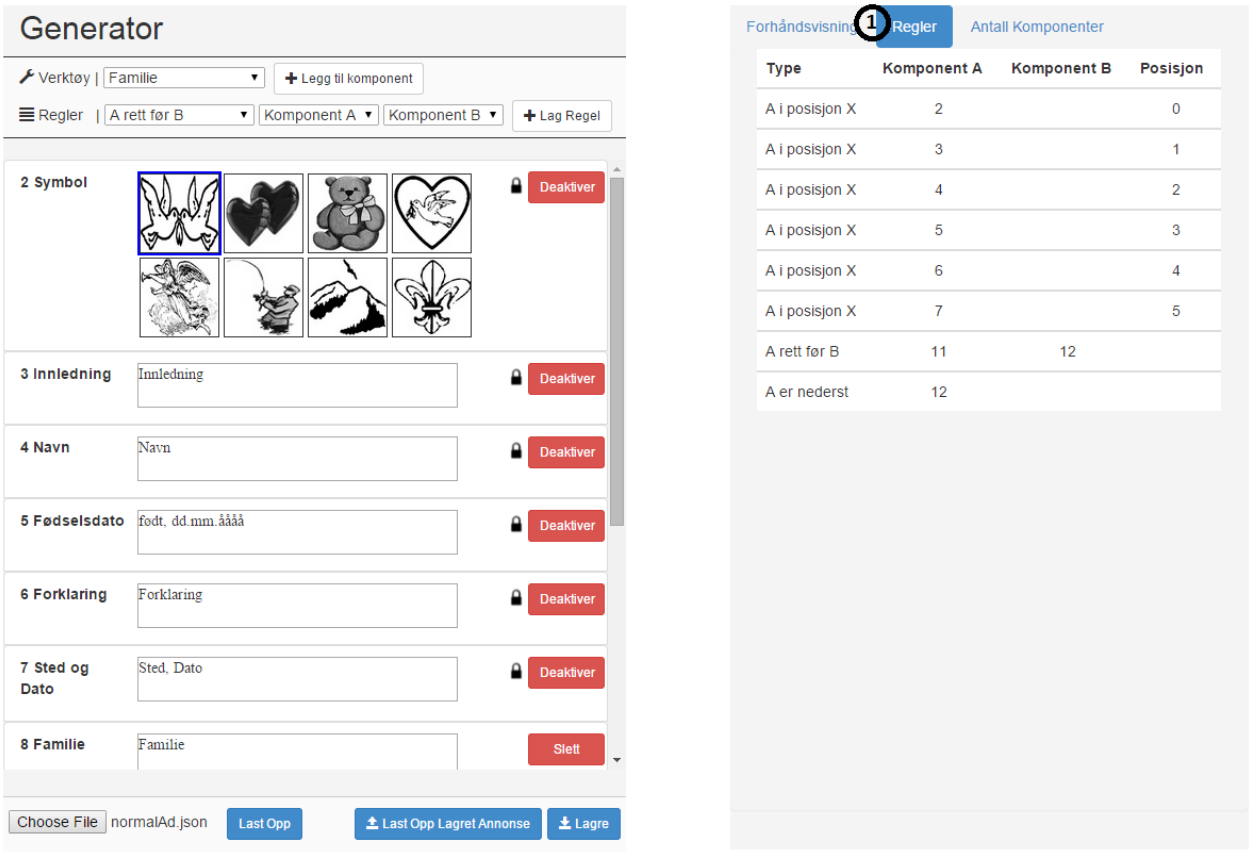
bruk. Sluttbrukeren har ikke mulighet til å slette komponenten, kun "gjemme" den ved å klikke på "Deaktiver"-knappen. Mer om deaktivering av komponenter i seksjon 4.

I tillegg, dersom man spesifiserer at en komponent A alltid må være rett før en komponent B, impliserer det at hvis man flytter komponent B, så må komponent A flyttes sammen med B. For å gjøre dette klart for sluttbrukeren blir komponenter som har direkte tilknytning til hverandre, ved for eksempel å definere "A rett før B" regelen, gruppert sammen.



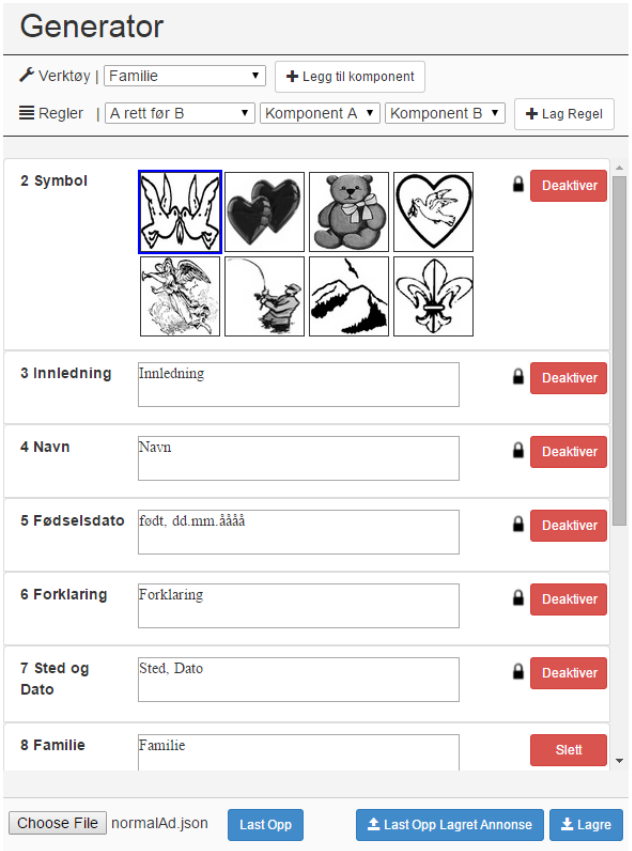
Her er det definert en regel om at "Forklaring"-komponenten skal være rett før "Familie"-komponenten. Begge komponentene vil da grupperes i en blokk, der skillelinjen mellom de to komponentene er fjernet. Siden de er gruppert i en blokk, vil man flytte hele blokken dersom man forsøker å flytte den opp eller ned.

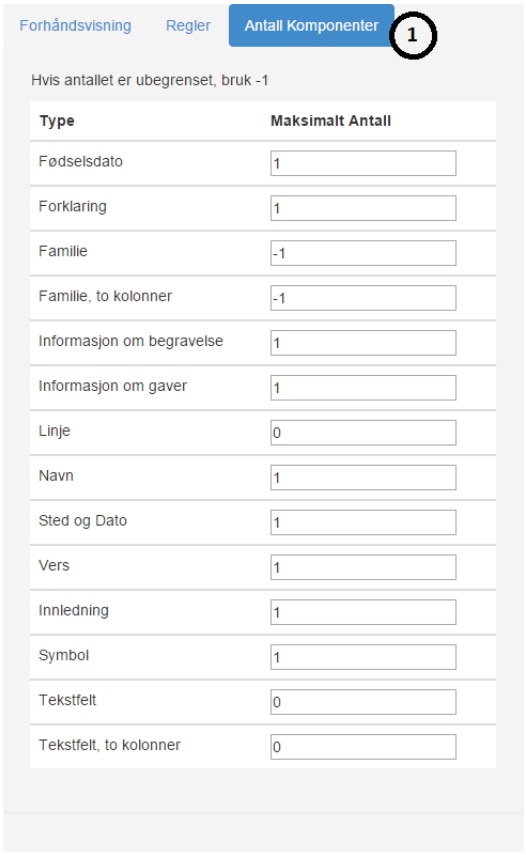
I.9 ”Regler”-fanen



Alle regler som er definert for en annonse vises i “Regler”-fanen (1). Her vises en tabell over alle reglene, der kolonnene består av “Type”, “Komponent A”, “Komponent B” og “Posisjon”. Dette gjør det enkelt for IT-ansatte å holde en oversikt over hvilke regler som allerede er definert.

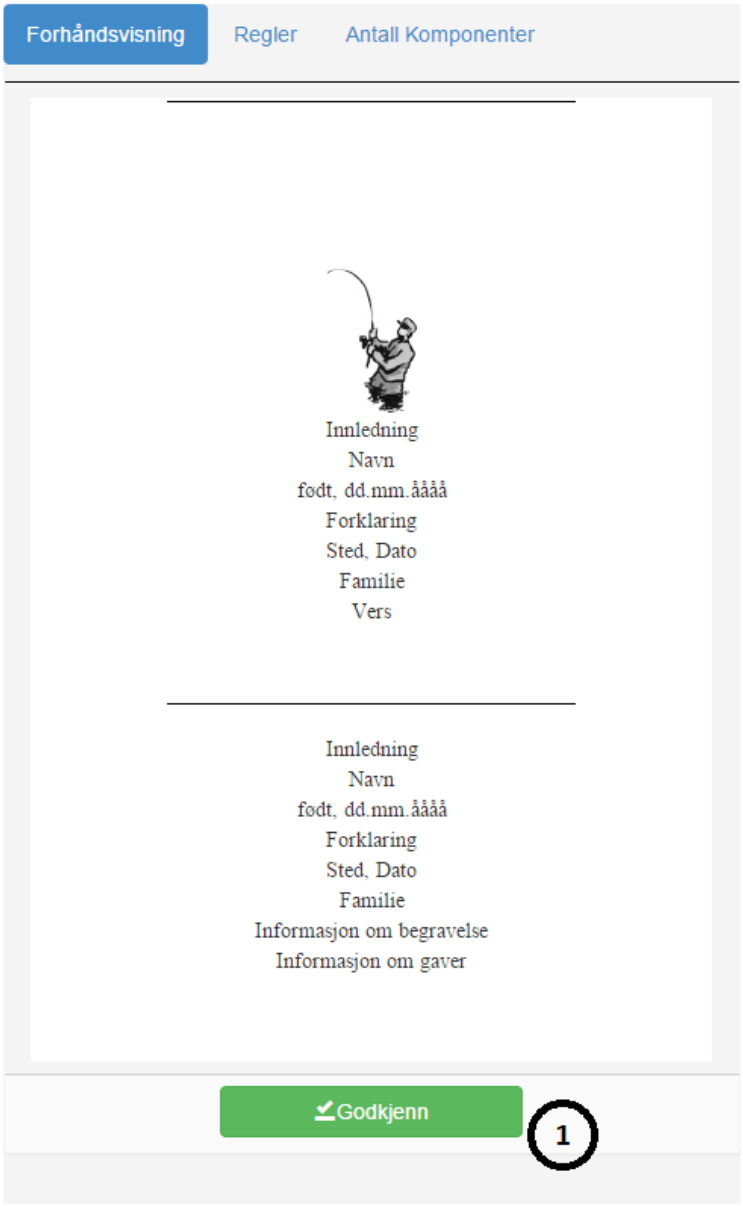
I.10 ”Antall komponenter”-fanen





For å begrense sluttbrukeren i antall komponenter som kan legges til en forhåndsdefinert mal, har IT-ansatte muligheten til å spesifisere antall komponenter som kan være i annonsen ved å henvende seg til “Antall Komponenter”-fanen. En verdi på -1 i et felt betyr at det ikke er noen grense for hvor mange komponenter av denne typen som kan legges til. Andre positive heltallsverdier sier at det kan eksakt være dét antallet komponenter i annonsen. Denne funksjonaliteten er spesielt nyttig dersom man for eksempel skal lage en dobbel-mal, der man kan ha to annonser i en. I så tilfelle kan IT-ansatte spesifisere at det for eksempel må være kun to navn i annonsen.

I.11 Fullføre annonsen



Når man som IT-ansatt er ferdig med annonsen, trykker man på "Godkjenn"-knappen (1). Malen man har definert vil da bli sendt til Adresseavisens servere i korrekte formater. Malen kan da brukes av sluttbrukere.

Appendix J

Technical description for the customer

J.1 Description of the program in detail (architecture)

Services

ComponentsService

The core design of our solution is the componentsService file. This is the file that contains the different components and manages what data they consist of. Also the adding and moving of components is organized here. This service consists of four arrays that control the manipulation of the components in the editor. These are the following: components, linearizedComponents, orderOfComponents, and blocks. The components and linearizedComponents are basically the same list. However, the difference is that in the components list the blocks are included. These blocks have values that are not stored in the saving and loading of the editor, which is why there are two lists for almost the same thing. The linearizedComponents list is also the list that is being used to show the preview of the ad. The orderOfComponents list is the list that specifies the order in which the components should be shown. The reason behind this list is that when one checks if a move is valid or not, all the data from the linearizedComponents list will not be copied. Instead the orderOfComponents is manipulated and the rules are checked according to the new order. The blocks list is a list with numbers indicating if a component is inside a block. The default value is -1, indicating that the component is not part of a block. Otherwise the value is the block id.

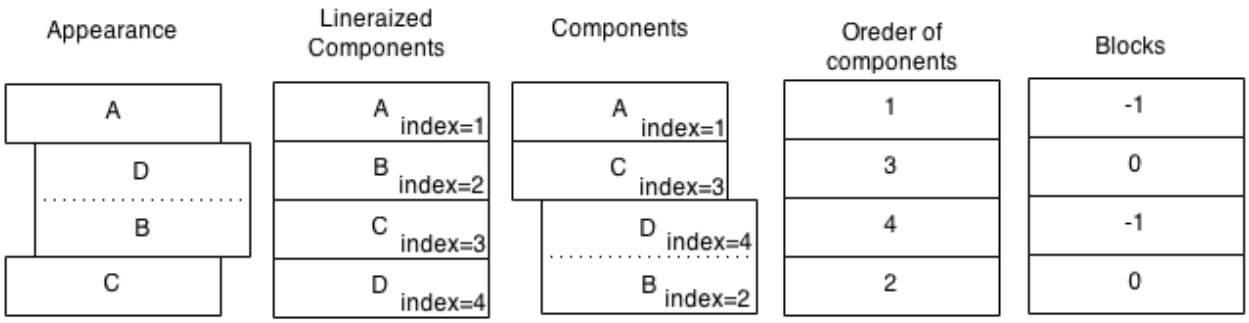


Figure J.1: Visualization example of the structure of the lists

As mentioned earlier, the componentsService create all the fields in the view. Therefore, it also depends on the fontService. That way, when a new component that should contain text is generated, it gets a valid font from this service.

All the components share some common properties. These are the following: id, type, viewType, description, isGeneratedFromTemplate, componentListIndex, and dragged. In addition they need the functions setActiveAttribute, and setDraggableAttribute. The property isGeneratedFromTemplate indicates if the component should be deletable or just be deactivated. The componentListIndex gives us the index of where the position of the component is stored in the orderOfComponents list.

The ComponentsService also includes a list containing how many components of each type is allowed to be in a given template. This is done through the availableComponents object. This object includes the name of every component as keys. The most important part of this objects is the specification of the maxNumber and the usedNumber. If the value of maxNumber is -1, the specific type of component can be used unlimited. Otherwise the type of component can appear only as often as maxNumber specifies. UsedNumber holds the number of existing components of a specific type. If usedNumber is equal to maxNumber, the type of component is not shown in the selection for adding components of specific types.

The moving of components is handled by the function changeOrderBy() which gets the placeholderPosition from the drag and drop library i.e. the position over which the dragged component is hovered. It is important to note that the placeholderPosition is intuitive when the component is moved to smaller indices. If it is moved for example from position five to position two, the placeholderPosition is two. But if the component is moved to higher indices e.g. from position one to position 6, the placeholderPosition is 7 as the dragged component remains at position one until it is dropped.

RuleCheckerService

The ruleCheckerService is responsible for organizing the different rule types and their validation as well as the existing rules. The different types of rules are stored in the availableRules object

that is used when the rule types are listed like in the selection for creating rules. For an overview of rule types see table 8.3.

If a rule from the type “A at position X” or “A is bottom” is created, the component A becomes locked. The rule “A is directly before B” leads either to the creation of a new block or the manipulation of an old block which happens in the componentsService.

If a rule is created, it gets validated before being added to the rulesList array where all existing rules are stored. Each rule object has a function checkRule() that validates if the rule applies given a specific order of components. When the function checkRules() is called, the checkRule() function of the single existing rules are called.

When a drag and drop event happens, the dropability is determined by calling the function checkManupulationOfComponentsListWithComponentSearch(). First the dragged component is identified, then the order of components is manipulated by the componentsService which gets the position of the placeholder of the drag and drop i.e. where the dragged component is hovered. For this manipulated order all rules are checked. If false is returned, the component is not dropable at this placeholder position.

Model

The model file is the file that ties the javascript together with the index.html file. It is this module that contains all of the controllers and services, and where the dependency injection starts. This file also has a controller for setting the role of the adEditor user. If the value is “admin”, the parts of the editor enabling the creation of rules are visible.

FontService

The fontService is just a collection of the different approved fonts and the font sizes that can be applied. The data for how big the fonts are is not specified in this class but in the stylesheet (CSS file).

SymbolService

The symbolService is just a collection of the different paths to the different images and it contains a getter for accessing the images.

StringService

The stringService is the class which contains the translation to norwegian and the wording of the different fields in the HTML code.

FileReaderService

The fileReaderService contains two methods. The first one, readFilesToComponents, is a test

method for trying to work with gets and posts. The second method, loadRulesAndComponentsFromString, inputs a string in the form of a JSON object and extracts data from it, given that the syntax is correct.

Controllers

ApproveController

The approveController contains the methods to create all the files or their string counterparts that should be sent to the server when the approve button is pressed. It is also part of the mechanism that changes parts of the stylesheet so that the whole preview gets taken into consideration when making the different parts. To get the dpi as high as the customer wanted, the canvas gets generated with the library htmlToCanvas and gets scaled up and converted to an image. Because the customer wanted an image with 300 dpi and the conversion of the canvas resulted in an image of 96 dpi we had to make some adjustments. To compensate for this we scaled the canvas up to a bigger image (3.125 times bigger), so when it is downscaled in the pdf conversion it gets the proper quality. If the customer want to get higher quality of the finished image, they should consider two things: Either make the preview bigger, or wait until html5.1 is released. If they make it bigger, the generated image will be several times bigger than the image they want, and they can then downscale it to an acceptable size and have an overall better quality of the image. If they wait for html5.1, the release will include a new way to convert canvas to an image url without the restriction of maximum 96dpi.

LoadController

This controller handles the way the local file gets uploaded and calls the right FileReaderService to load the JSON object into the editor. This controller is useless when the server is installed and should then be removed.

SaveController

This controller takes the components in the componentsService, and rules from the ruleCheckerService and generates a JSON object string that is valid to load from in the editor. Currently the string is saved in the local browser storage. This gets unnecessary as soon as the server is used.

Directives

All of the directives have HTML files accompanying them. When defining a directive, you often need a controller. The controller that is accompanying the directive is usually located in the same file as the directive. By assigning this controller to the directive, you can access the methods stored in the controller in the HTML file.

CreateRulesDirective

The createRulesDirective is the javascript behind the bar that shows up when the user is the admin. This means that it includes the controller that handles the different actions for creating rules using the ruleCheckerService.

EditComponentsDirective

The editComponentsDirective is the directive that includes the editComponentController. This controller handles all the interactions with the user, when the user does changes to the components in the edit view. This is also where the contextMenu is used to get the popup that gives the user the possibility to change the font and size of the text in the editor.

ContextMenuDirective

This directive is a representation of the menu that shows up in the editor when the user right-clicks on a text area. The way it gets access to the data is through the ngRightClick directive that resides in the EditComponentDirective.js file.

NumberComponentsSpecificationDirective

This directive is used to show and set the maximum amount of components from different types when creating a template.

PreviewComponentsDirective

This directive organize the single components in the way the ad will look like in the printed form.

PreviewRulesTabViewDirective

This directive is the common directive for showing the right side of the adEditor including the different tabs when the user is an admin and otherwise only the preview. It uses the panelController for the tab organization.

RuleListDirective

This directive shows a table of the existing rules, and is only shown when the user is an admin.

ToolbarDirective

This directive is the collection of the methods that the user can access through the toolbar in the editor. The controller here is the one that calls the methods in componentsService to add new components to the view.

The HTML code

The main file for the HTML code is the index.html, where all the js files are referenced and the website gets classified as an angular site. The ng-app tag is the one tying the view to the model that again includes the controllers.

To loop through every component in the list, the angularJS tag ng-repeat is used. This loops through every component in the list. It is used both in the edit-components directive HTML file and in the preview-components HTML file. The preview-components HTML file is added through the preview-rules-tab-view HTML.

J.1.1 Improvements and extensions

The editor has room for improvements as there are features we did not have the time or technologies to finish, or it simply was not part of our task. When integrating the editor in your system, we suggest to add or improve the following:

- The image, PDF, and HTML generation has to be rethought, as the technology for implementing a satisfying solution for these functionalities is not yet available for the client side.
- The deletion of rules should be added to complete the template creation for the IT-department user.
- More unit tests should be implemented to cover both new, and existing functionalities. Integration tests, which are not possible without a server, should be included. We also recommend to add some sort of code coverage check.
- The following rule enforcements may also be added
  - When a component is dragged across a locked component, the drop is not possible as this would mean a change of the position of the locked component. A solution would be that the component on the other side of the locked component would swap around the locked component.
  - The maximum number of components of a type can be specified lower than the number of components added in the template, which does not make sense. It needs to be forced that only valid values can be entered.
- Implement a server connection (GET, POST) to, among other features, be able to
  - Dynamically load additional symbols
  - Categorize the symbols in folders and browse through them
  - Implement a login functionality with different user roles
- Improve dependency relations in the code.

These are possible extensions for our application It should be easy to run the editor in different languages It should be extendable to other types of ads, and to pamphlets of several pages If some security is implemented the project could be commercialized. As the editor is supposed to be integrated into an already secure system by Adresseavisen, we were told that the security aspect could be neglected.

J.2 Server API

The way we see the server interacting with our solution, contains the following: When loading the editor there are a number of fields that need to follow the instruction, when saving the editor it should send the ad to the server, and when approving the ad it should send the image, pdf, hmlt, and ad to the server.

Loading

The GET method should include:

- A JSON object containing the rules, template and the saved ad.
- A field for the deceased's name, date of birth, and date of death.
- A field containing the role of the user (admin vs. user).

To load the add you need to call the method loadRulesAndComponentsFromString, with the JSON object as the parameter, that lies in the ReadFileService. This will ensure that the ad loads with the right data. Since we haven't implemented a server, we haven't specified anywhere in the code where to handel the GET method, so the customer will have to implement this. This is the case for all the properties, except the role, sendt with the get. The property to set and to get the right role is the one in the controller userInformationController, which is located at the Modul.js file.

Regarding the POST's we see two places that should trigger a response like this. When the user saves the add and when the user approves it.

Saving

The save POST method should send only the JSON object generated through the save function. The file you are interested in is the jsonFile variable in that method. In addition it should send the role of the user, so the server can verify if it is a new template or a saved ad.

Approving

The approve POST method should send the JSON file as mentioned above in addition it should send the PDF, image, and html. All of these are generated when the button is clicked, however what you need to do is to put each of them inside the post and send that to the server. The image is a URL string generated by the method cerate image. This string is currently stored in the local storage of the browser, but by sending it to the server you will be able to handle it there to get the image out of it. The PDF now generates a download file that saves to the local machine. What needs to be done is to remove the PDF file generation on the client side and send the PDF blob that gets created. Last the html is just in string form and should be sent to the server and saved as html and it will open in any browser.

The create image method resides in the file ApproveController.

Right now there are some things that should be removed after setting up the API. The load button and input field should be removed, as they stand now they work with any browser except internet explorer, they are to be removed and therefore does not pose a problem. The file loadController can also then be safely removed from the solution.

Appendix K

How to set up frameworks for running and writing unit tests

K.1 Installation

Make sure you run all the installation commands as administrator.

Install node.js from:
http://nodejs.org/ [20]

Install Karma by running this command in your Windows command line or your unix terminal:
npm install -g karma

Install karma command line interface
npm install -g karma-cli

Install Kama and Jasmine plugins
npm install karma-jasmine karma-chrome-launcher –save-dev
npm install jasmine-node -g

Install Phantom.js by either:
npm install -g phantomjs
http://phantomjs.org/ [23]

For Windows you need to make sure the karma and node are added to the system variables. Right-click on my computer, select advanced system settings and choose environment variables. In the system variables list. Select the variable named PATH and make sure the path

to your new installations are there, or else you need to add them.

To verify the installations: test by running the commands “node -v” and “karma –version”. It will hopefully print something similar to this:

node v0.10.xx

karma version 0.12.xx

K.2 Set it up in Netbeans

1. Make sure you have the karma.config.js for this project in the files tab in the navigator window. It should be included with the code.
2. Right-click the project node in the Projects tab (still in the navigator window) and choose Properties in the popup menu.
3. Select JavaScript Testing category in the Categories pane of the Project Properties window.
4. Select Karma in the Testing Provider drop-down list. Click OK.
5. Open the Project Properties window again and select Karma under the JavaScript Testing category in the Categories pane.
6. Specify the location of your Karma installation and the karma.config.js file.

Example locations in Windows: Karma:

```
C:\Users\username\AppData\Roaming\npm\karma.cmd
```

Config file:

```
C:\Users\username\Documents\NetBeansProjects\AdEditor\AdEditorProject\karma.conf.js
```

7. When you click OK you can see that a Karma node appears under the project node in the Projects tab in the navigation window. You right-click the Karma node and start and stop the Karma server and set the configuration file in the popup menu.
8. When you click Start the Karma server starts and a browser window opens that displays the status of the server.
9. Right-click the Karma node and choose Set Configuration > karma.conf.js to confirm that the correct configuration file is selected.
10. Right-click the project node in the Projects window and choose Test.
11. When you choose Test the test runner runs the unit tests on the files. The IDE opens the Test Results window and displays the results of the test.
12. Tests are located in the test folder of the project. You may have to define the path to the tests somewhere in the properties of the project. This depends on your OS and your netbeans version.

Some possible issues:

Karma issue (on Mac) problem description and solution [27].

K.3 Run tests once from command line (Windows) or terminal (Unix)

If you wants to run the tests without having to set up netbeans, or you are using a different kind of IDE, you can run tests using commands.

Navigate to the project folder, and into the AdEditorProject folder(where karma.config.js is located. Run this command and the tests should run once: karma start karma.config.js

Bibliography

[1] Adobe indesign wikipedia article. http://en.wikipedia.org/wiki/Adobe_InDesign.

[2] Angularjs homepage. https://angularjs.org/.

[3] Chrome web store. https://chrome.google.com/webstore/category/collection/drive_apps.

[4] Different tutorials. http://www.ng-newsletter.com/advent2013/#!/day/19.

[5] Dropbox vs drive comparison. http://www.trustedreviews.com/opinions/google-drive-vs-icloud-drive-vs-dropbox-vs-onedrive.

[6] Ember.js homepage. http://emberjs.com/.

[7] Github repository for angular-drag-and-drop-lists. https://github.com/marceljuenemann/angular-drag-and-drop-lists.

[8] Google calendar. https://www.google.com/calendar.

[9] Google drive. https://drive.google.com.

[10] Google drive formats. http://en.wikipedia.org/wiki/Google_Drive.

[11] Google's angularjs styleguide. http://google-styleguide.googlecode.com/svn/trunk/angularjs-google-style.html.

[12] Html5 canvas. http://www.w3schools.com/html/html5_canvas.asp.

[13] Ieee recommended practice for software requirements specifications. http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=720574.

[14] Information about white box testing. http://softwaretestingfundamentals.com/white-box-testing/.

[15] Jquery homepage. http://jquery.com/.

[16] Json wikipedia article. http://en.wikipedia.org/wiki/JSON.

[17] jspdf homepage at parallax. https://parall.ax/products/jspdf.

[18] Microsoft one drive. https://onedrive.live.com/.

[19] Netbeans compatibility with jasmine. https://netbeans.org/kb/docs/webclient/HTML5-js-support.HTML#karmatests.

[20] Nodejs homepage. http://nodejs.org/.

[21] obitnow.com homepage. http://www.obitnow.com.

[22] Pdf.js homepage. http://mozilla.github.io/pdf.js/.

[23] Phantomjs homepage. http://phantomjs.org/.

[24] Princeton on waterfall model. http://www.princeton.edu/~achaney/tmve/wiki100k/docs/Waterfall_model.html.

[25] Rip.ie homepage. http://www.rip.ie/.

[26] Sharelatex. https://www.sharelatex.com/.

[27] Solution to karma issue on mac. https://github.com/karma-runner/karma/issues/1034.

[28] Techtarget on waterfall model. http://searchsoftwarequality.techtarget.com/definition/waterfall-model.

[29] Testing in angularjs. http://stackoverflow.com/questions/300855/JavaScript-unit-test-tools-for-tdd.

[30] Testing using both karma and protractor. http://stackoverflow.com/questions/17070522/can-protractor-and-karma-be-used-together.

[31] Tutorial for using jasmine. http://nathanleclaire.com/blog/2013/12/13/how-to-unit-test-controllers-in-AngularJS-without-setting-your-hair-on-fire/.

[32] Uml homepage. http://www.uml.org/.

[33] Validator.nu homepage. http://validator.nu/.

[34] Wikipedia latex. http://en.wikipedia.org/wiki/LaTeX.

[35] Wikipedia on waterfall model. http://en.wikipedia.org/wiki/Waterfall_model.

[36] Wysiwyg wikipedia article. http://en.wikipedia.org/wiki/WYSIWYG.