NTNU
Det skapende universitet

Rammeverk for presentasjon av hendelsesforløp

Steffen Eriksen
Vegard Lundberg Holter

Master i datateknologi	
Innlevert:	desember 2015
Hovedveileder:	Svein-Olaf Hvasshovd, IDI
Medveileder:	Rolv Bræk, ITEM

Norges teknisk-naturvitenskapelige universitet
Institutt for datateknikk og informasjonsvitenskap

NTNU – Trondheim
Norwegian University of
Science and Technology

Map Framework for Event Presentation

Steffen Eriksen & Vegard Lundberg Holter

December 2015

MASTER THESIS

Department of Telematics &

Department of Computer and Information Science

Norwegian University of Science and Technology

Supervisor 1: Professor Svein-Olaf Hvasshovd

Supervisor 2: Professor Rolv Bræk

Preface

The work featured in this thesis has been performed as collaboration between Vegard Lundberg Holter and Steffen Eriksen at the Department of Computer and Information Science (IDI) and the Department of Telematics (ITEM), respectively, during the Autumn of 2015. The thesis has been submitted to the Norwegian University of Science and Technology (NTNU), as partial fulfilment of the study programs "Master in Computer Science" and "Master in Communication Technology", and as part of the courses "TDT4900 Computer Science, Master's Thesis" and "TTM4905 Telematics, Master's Thesis". The thesis supervisors has been IDI Professor Svein-Olaf Hvasshovd and ITEM Professor Rolv Bræk.

Trondheim, 2015-12-15

Vegard L. Holter

Vegard Lundberg Holter

Steffen Eriksen

Steffen Eriksen

Acknowledgment

We would primarily like to thank our supervisors Professor Svein-Olaf Hvasshovd and Professor Rolv Bræk for their persistent guidance and commenting during our work. We would also like to thank Marius Ekerholt at Dogu AS for lending us a web server used for testing and implementation. Lastly, we would like to thank Torben Sørensen, Remi Allegre, Kristoffer Øseth, Alice Holm and Marianne Evjen for taking part in a user test essential to the thesis' result.

Summary

Literary plots often evolve around a series of geographical locations. However, the plot often fail to provide precise details on locations, which may leave the reader uncertain of where the events exactly unfolded. Readers often have to rely on third party services in order to get more precise geographic information for a given plot. Third party look ups can be time consuming, yield inaccurate results and might lead to loss of the storyline.

The objective for this thesis has been to investigate the technical feasibility and desirable functionality for a service that combines existing literature with supplementary maps and photos. The service is intended to provide readers with an enriched reading experience, that precisely communicates locations in map context. Such a service is yet to be developed.

The thesis' objective has been approached using prototyping in combination with case study and user testing. A case study, consisting of creating a case for the prototype based on an existing storyline, is used as the basis for validation of the prototype and foundation for user testing. The storyline featured the participation of heavy German cruiser Blücher during the German attack on Oslo, Norway in World War 2, 1940. The prototype was able to represent the storylines' involved parties, e.g., Blücher and Oscarsborg Fortress, in a map as markers, and key date-times in a timeline. The prototype offers dynamic maps and timelines that evolve as the reader progresses through the text, only highlighting locations and key date-times that is essential to the reader's position in the text. For example, Blücher's travel path was expanded in the map as its position advanced from Swinemunde to Drøbak.

A user test was conducted by having five participants use the prototype. They provided valuable feedback that contributed to changes in the prototype and gave recommendations for future functionalities. Multiple test subjects stated that the prototype communicated locations in a precise way, and provided a joyful and informative reading experience. This confirms that the service can indeed offer an enriched reading experience

Sammendrag

Litteratur omhandler ofte en rekke geografiske steder. Disse er ofte dårlig beskrevet, noe som kan føre til at leseren blir usikker på nøyaktig hvor bokens hendelsesforløp foregikk. Lesere må ofte gjøre oppslag i andre tjenester for å få en mer presis geografisk informasjon. Dette kan være tidkrevende, gi unøyaktige resultater, samt føre til at leseren faller ut av bokens hendelsesforløp.

Målet for denne oppgaven har vært å undersøke den tekniske gjennomførbarheten og ønskelig funksjonalitet for en tjeneste som kombinerer eksisterende litteratur med kart og bilder. Tjenesten er ment å kunne tilby en informativ og berikende leseopplevelse, som presist kommuniserer geografiske steder ved hjelp av kart, med mer. En slik tjeneste er ennå ikke utviklet.

Arbeidsmetoden har vært å lage en prototype, samt å utføre et casestudie og en brukertesting av prototypen. Casestudiet bestod av å legge inn en eksisterende tekst. Dette ble brukt som valideringsgrunnlag og som grunnlag i brukertestingen. Den innhentede teksten omhandlet rollen den tunge tyske krysseren Blücher hadde under det tyske angrepet på Oslo, Norge i 2. verdenskrig, April 1940. Prototypen var i stand til å representere de involverte partene i historien i kart-format. Blücher og Oscarsborg festning, ble for eksempel plassert i kartet som markører, og viktige datoer og tidspunkter i en tidslinje. Prototypen viste dynamiske kart og tidslinjer som oppdaterte seg etterhvert som leseren forflyttet seg i teksten, slik at kartet og tidslinjen kun fremhevet steder og nøkkeldatoer som var aktuelle for leserens posisjon i teksten. For eksempel ble Blücher sin reiserute utvidet etterhvert som posisjonen forflyttet seg mellom Swinemünde og Drøbak.

En brukertest ble gjennomført ved å la fem deltakere bruke prototypen. De ga verdifulle tilbakemeldinger, noe som bidro til endringer i prototypen, og førte til anbefalinger for fremtidig funksjonalitet. Flere testpersoner uttalte at prototypen kommuniserte geografisk informasjon på en presis måte, og ga en gledelig og informativ leseopplevelse. Dette bekrefter at tjenesten faktisk kan tilby en beriket leseopplevelse.

Contents

Acronyms

AJAX Asynchronous JavaScript and XML. ix, 17

CSS Cascading Style Sheets. ix, 27, 70

DB database. ix, 36–40, 43–46, 48, 54, 67

DOM Document Object Model. ix, 15, 20

EF Entity Framework. ix, 24, 40, 42

ER Entity Relationship. ix, 40, 41

ESRI Environmental Systems Research Institute. ix, 5–7, 14

HBO Home Box Office. ix, 10–12

HTML HyperText Markup Language. ix, 20–23, 27, 30–33, 35, 67, 70

HTTP HyperText Transfer Protocol. ix, 16, 34, 35, 39, 44, 70

IDI Department of Computer and Information Science. i, ix

ITEM Department of Telematics. i, ix

JS Javascript. ix, 16, 20, 23, 27, 28, 30–32, 34, 35, 39, 70

JSON JavaScript Object Notation. ix, 16, 25, 35, 39, 46

LOTR Lord of the Rings. ix, 8

NTNU Norwegian University of Science and Technology. i, ix

ORM Object-Relational Mapping. ix, 24, 40

OS Operating System. ix, 15

SQL Structured Query Language. ix, 24, 40

UI User Interface. ix, 2, 8, 9, 14–20, 22–24, 27–31, 33–35, 39, 43, 54, 57–62, 66–71, 74

URL Uniform Resource Locator. ix, 70

WWW World Wide Web. ix, 16

Chapter 1

Introduction

1.1 Background

Literary plots often evolve around a series of geographical locations. Books often provide interleaved maps and photos with notes and highlighted locations. However, books often fail to provide precise details on locations, which may leave the reader uncertain of where the events exactly unfolded. It may also, for the reader, be challenging to keep track of continuously changing positions as the storyline progresses. Readers often have to rely on third party services in order to get more precise geographic information for a given book. This could, e.g., be the world atlas or a web search. Third party look ups can be time consuming, yield inaccurate results, create frustrations and might lead to loss of the storyline.

The need for third party look ups can be avoided if books were presented in a digital service with complementary maps and photos. Such a service could be beneficial in several ways. Primarily, the reader only needs to consult a single service, which may prevent loss of storyline. Additional information provided in map context may be less open for misinterpretation. A digital service may also be able to offer dynamic maps that evolves as the reader progresses through the literary plot, only highlighting locations that is essential to the reader's position in the text. Such a service should be able to present any existing literature with additional photos and map information. The reading experience may be richly enhanced and accurately perceived by this way of reading.

There are currently few services that combines text, maps and photos. Some of them are tailored to specific novels. Others have a general approach that allows arbitrary text to be combined with maps and photos. However, they all prioritize displaying maps and photos, leaving little room for textual content.

A service that presents existing literature, i.e., books, in its full form with the addition of supplementary maps and photos is yet to be developed. Such a service should not be limited to only presenting summaries or appendices. The main focal point should be textual content, analogous to electronic book services such as Kindle, which is a tablet specifically designed for reading text. Maps and photos should be supplementary elements to complement and enrich the text, not the other way around.

1.2 Objectives

The thesis' objective is to investigate the technical feasibility and desirable functionality of a service that combines existing literature with supplementary maps and photos. The service is intended to provide readers with an enriching reading experience, that precisely communicates locations in map context. The completed service must consider two parties, the reader and the author. The reader should be presented with a User Interface (UI) consisting of a novel and supplementary elements, such as maps and photos. The author should be presented with an editor for creating and editing content, both text and maps.

1.3 Limitations

A user friendly way to create and update storylines, i.e., the part of the service concerning authors, should be provided in a finalized version of the service. This is, however, not important in demonstrating whether or not a service for combined text and maps gives an enriched reading experience. This part is therefore less prioritized in favor of displaying storylines and maps, i.e., the part of the service concerning readers. The thesis' scope is limited to finding a pleasing way to display supplementary data with literature, and investigate the technical feasibility of realizing such a service.

1.4 Approach

The thesis' objective is approached using prototyping in combination with case study and user testing. Case study, consisting of creating a case for the prototype based on an existing storyline, is used as the basis for validation of the prototype and foundation for user testing. This approach may, to a certain degree, reveal strengths, weaknesses, limitations and overall capability of such a service.

1.5 Structure of the Report

The rest of the report is organized as follows.

- Chapter 2 presents an evaluation of related work.
- Chapter 3 presents planning of the thesis prototype in terms of underlying technology, structure, communication and which third party solutions are utilized.
- Chapter 4 presents the methods and technology used for implementing the prototype.
- Chapter 5 presents a case implemented in the prototype.
- Chapter 6 presents feedback from a user test of the prototype, and conclusion of the feedback.
- Chapter 7 presents recommendations for future work, divided into Short Term and Long Term plans.
- Chapter 8 presents a summary, conclusion and discussion.

Chapter 2

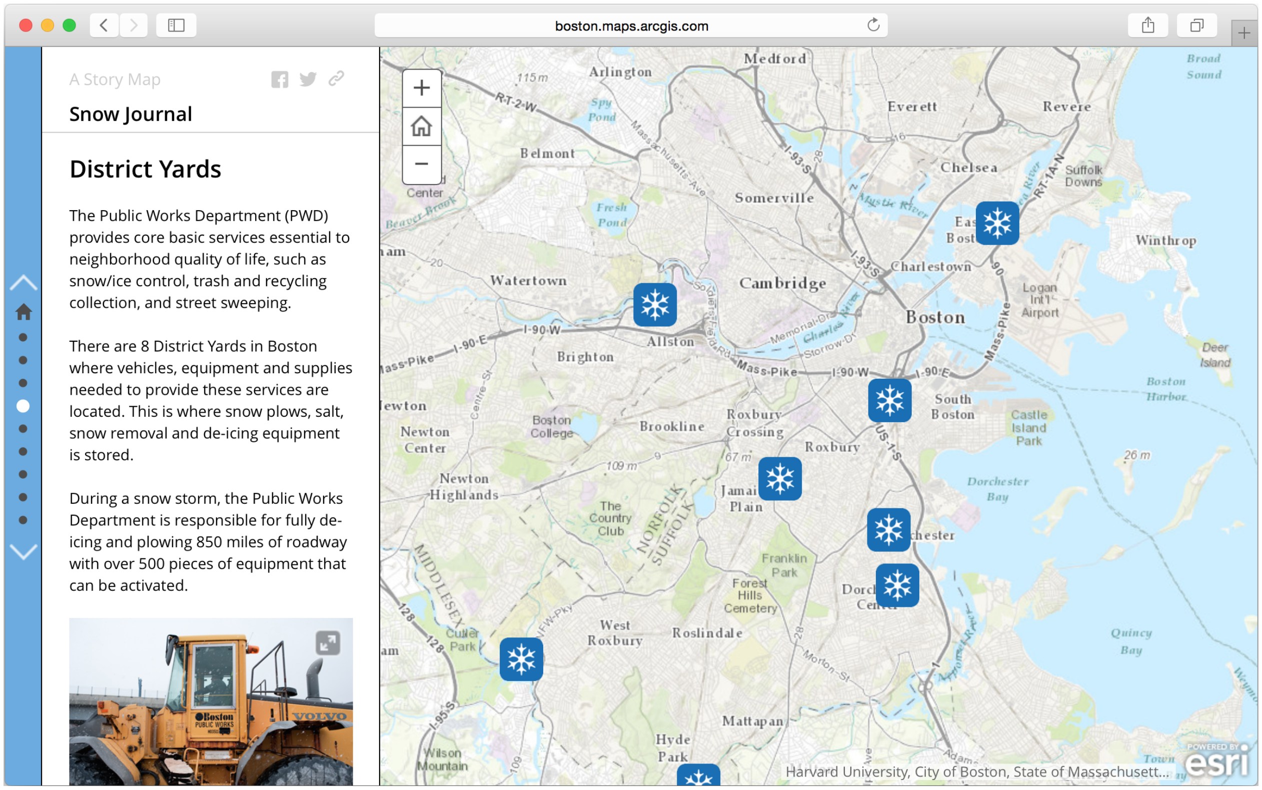
Related Work

This chapter presents an evaluation of related work.

2.1 ESRI - Storytelling with Maps

Description

Environmental Systems Research Institute (ESRI) [1] offers a web service that allows users to create stories with accompanying maps and photos. The service also presents a collection of stories created by users. The user created stories feature topics such as natural disasters, population growth, city development and historical events, e.g., the assassination of Abraham Lincoln. Each user created story has its own sub-domain in the ESRI web page. The available stories have been given individual customized styling, however, they all follow the same structure. The structure is selected from a collection of templates. The templates include a large view where maps, photos and graphs are displayed. The view can either be a scrollable list of maps and graphs, or a single map or graph. The map or graph occupies about 80 % of the screen. They also have a smaller view for text and photos. However, the text view is not optimal for reading large quantities of text because of its narrow width, which is roughly one quarter of a normal book page in width. An example of a story created in the ESRI service is shown in Figure 2.1.



boston.maps.arcgis.com

A Story Map

Snow Journal

District Yards

The Public Works Department (PWD) provides core basic services essential to neighborhood quality of life, such as snow/ice control, trash and recycling collection, and street sweeping.

There are 8 District Yards in Boston where vehicles, equipment and supplies needed to provide these services are located. This is where snow plows, salt, snow removal and de-icing equipment is stored.

During a snow storm, the Public Works Department is responsible for fully de-icing and plowing 850 miles of roadway with over 500 pieces of equipment that can be activated.

Figure 2.1: Example storyline created in ESRI.

The text in the text view is separated into multiple subsections that differ in color. The text section centered in the text view is highlighted by a darker color. The highlighting is updated when the reader scrolls to the next or previous text section. Each text section is also represented as a dot in a vertical list. This list is placed inside the text view. The dots are highlighted the same way as its corresponding text section. The dot is an alternative to text scrolling. When a dot is clicked, the text will automatically be scrolled up or down to make the corresponding text section visible.

The map is populated with markers of interest to the storyline. Each marker has an information window next to it. Information windows are populated with general information about the location or events that took place at this location. The markers can also be connected with edges to visualize paths. The map used by ESRI does not offer user interaction beyond adjusting zoom level and clicking on markers. It does not offer satellite photos or street views, in contrast to other major map providers such as Google Maps [4]. The map information is simple and low on detail and does not provide elevation data. Supplementary map data, such as paths and information windows, tend to clutter the map view.

A subset of markers in the map corresponds to a section of the text in the text view. The map is updated to highlight a new subset of markers when the reader progresses from one text section to the next. Markers are highlighted by showing their information window. The map will translate its center location and update its zoom level to outline the highlighted subset of markers. The highlighted markers are also assigned with an identifier. The identifier is an integer that starts at one and increments in accordance with the text sections. The identifier is visible in the information windows.

Pro

- The current text section is clearly highlighted with a darker color.
- The reading experience is richly enhanced by visualized information.
- Storylines can be created with different templates and customized appearance.

Con

- The map tends to get cluttered with markers and information windows.
- The text view is very small compared to other features, such as maps and graphs. This makes it demanding to read large quantums of text.
- The maps are rather rough and does not, e.g., offer elevation data.

Conclusion

ESRI offers a rich reading experience by adding maps and photos to storylines. The maps and photos help the reader understand where story events took place. However, litte room has been left for the text. The text is perceived as complementary description to the maps and photos. The stories available in ESRI are brief and suited to be told via maps and stories. This does not coincide with the intentions of this thesis. This thesis is about creating a service that provides readers with existing storylines, i.e., books, and maps and photos as additional information. Not the other way around.

However, the text highlighting is a nice feature that indicates the relationship between map data and text.

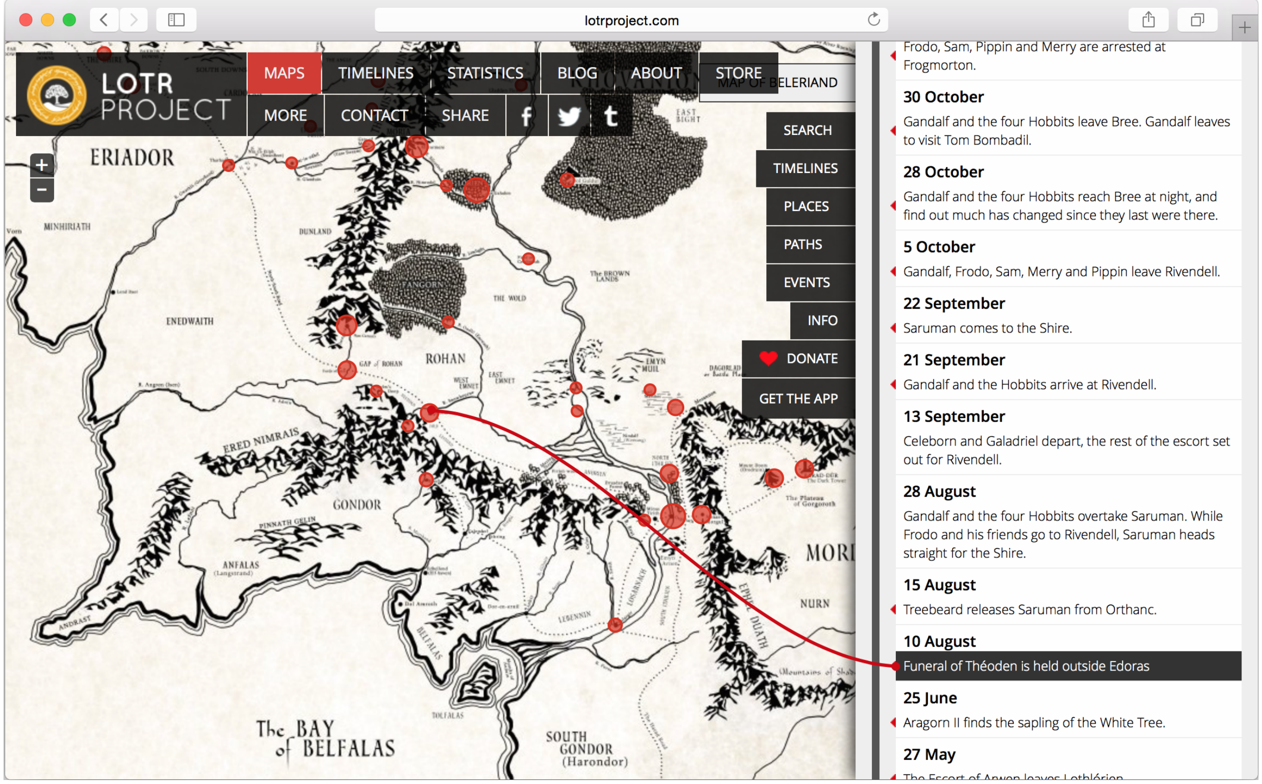
2.2 The LOTR Project

Description

The LOTR Project [12] is a map representation of the popular fantasy novels *Lord of the Rings (LOTR)* [14] by J.R.R. Tolkien. It is made available as a website where the reader can see where and when the main events of the novel took place. The website is offered as a complement to the book series. It does, therefore, not provide the reader with the full novels. The website offers two main information representations. These representations are minimalistic and easy to use.

The map representations show a full screen map with markers of interest. The map is rough and low in details, only displaying main roads, rivers, mountains, and so on. It does not display elevation data or minor details, such as trails or streams. A large information window is displayed when the reader clicks on a location marker. The information window presents a list of events that took place ar the location. In order to see the events that unfold in other locations, the reader must click on the respective location markers. This can result in many clicks if the reader wants to view the information window for all locations. The events in the information windows are sorted by date. The reader can also view paths of all main characters of the novel. A list of main characters are presented in a drop down menu. Paths display the transportation routes for a specific character. Transportation routes goes through a subset of markers of interest. Paths for different characters are given individual colors. The reader can select which subset of paths, i.e., characters, to include in the map via the drop down menu.

The timeline representation shows a map and a timeline. This is shown in shown in Figure 2.2. The map is similar to the one provided in the full screen representation, displayed on the rightmost side of the UI. The map is populated with the same markers of interest, with the exception of the information windows. The timeline is a vertical list of summarized events, displayed on the rightmost side of the UI. It contains the same event information used in the map representation, however, collected into one list, as opposed to dispersed in multiple information windows. All of the location markers and paths are displayed simultaneously. The map does not evolve as the user progresses through the timeline. The map highlights the markers of interest when the reader clicks on a timeline event.



Frodo, Sam, Pippin and Merry are arrested at Frogmorton.

30 October
Gandalf and the four Hobbits leave Bree. Gandalf leaves to visit Tom Bombadil.

28 October
Gandalf and the four Hobbits reach Bree at night, and find out much has changed since they last were there.

5 October
Gandalf, Frodo, Sam, Merry and Pippin leave Rivendell.

22 September
Saruman comes to the Shire.

21 September
Gandalf and the Hobbits arrive at Rivendell.

13 September
Celeborn and Galadriel depart, the rest of the escort set out for Rivendell.

28 August
Gandalf and the four Hobbits overtake Saruman. While Frodo and his friends go to Rivendell, Saruman heads straight for the Shire.

15 August
Treebeard releases Saruman from Orthanc.

10 August
Funeral of Théoden is held outside Edoras

25 June
Aragorn II finds the sapling of the White Tree.

27 May
The Escort of Arwen leaves Lothlórien

Figure 2.2: The LOTR Project

Pro

- The user can decide which character specific information is displayed in the map.
- The UI is minimalistic and easy to use.
- Map markers of interest are highlighted when the user click on an event in the timeline.

Con

- The reader must perform many clicks in order to view all the events in the map.

- The service is tailored for one specific book series.

- The maps are rather rough and does not, e.g., offer elevation data.

Conclusion

The LOTR Project offers little detailed information about the book series. The service is not meant as an alternative way of reading the books. It is meant as an appendix to the books. Readers of the book series may use the LOTR Project subsequently to get a summary of locations, transportation paths and timelines, as this information is widely dispersed in the book.

The LOTR Project differs from the goal for this thesis. The objective is to provide readers with an alternative way of reading an existing storyline. However, the timeline is a nice data representation for summarizing the main events of the storyline.

2.3 Game of Thrones, HBO Viewers Guide

Description

The Home Box Office (HBO) Viewers Guide [2] to the popular TV-show “The Game of Thrones” is a web site offering summarizations of the individual TV-episodes. The website is separated into two main pages. The first page offers textual summarizations. The second page offers a fictional map where the events of "Game of Thrones" unfold. The map representation consists of a full screen map, shown in Figure 2.3. The user can alter the map outline by zooming and translating the center position. A smaller map is displayed in the bottom left corner that illustrates where the map outline is defined.

The user must first select a specific episode in order to view the map. The plot of a single episode might only concern a single location. In that case, the map will only be populated with a single location marker. This ensures that the map is not cluttered with a large set of locations irrelevant to the episode. The content provided in the map is displayed with few visual distractions, such as trails, multiple text windows, and so on.

Markers will, when clicked on, display a photo of the respective locations. The photo is presented above the marker. They are, however, large in size, and cover large areas of the map when visible. The reader can get a thorough textual description of the location by opening a full screen overlay. This is done by clicking a button presented on top of the photos. The information provided for each location is collected from the appendices of the "Game of Thrones" book series. The TV-show is based on a small subset of the book-series. The book series provide a lengthy history on each location.

The user must click on every markers in order to read about the different locations. There is no timeline or textual summary other than what is presented in the full screen overlays.

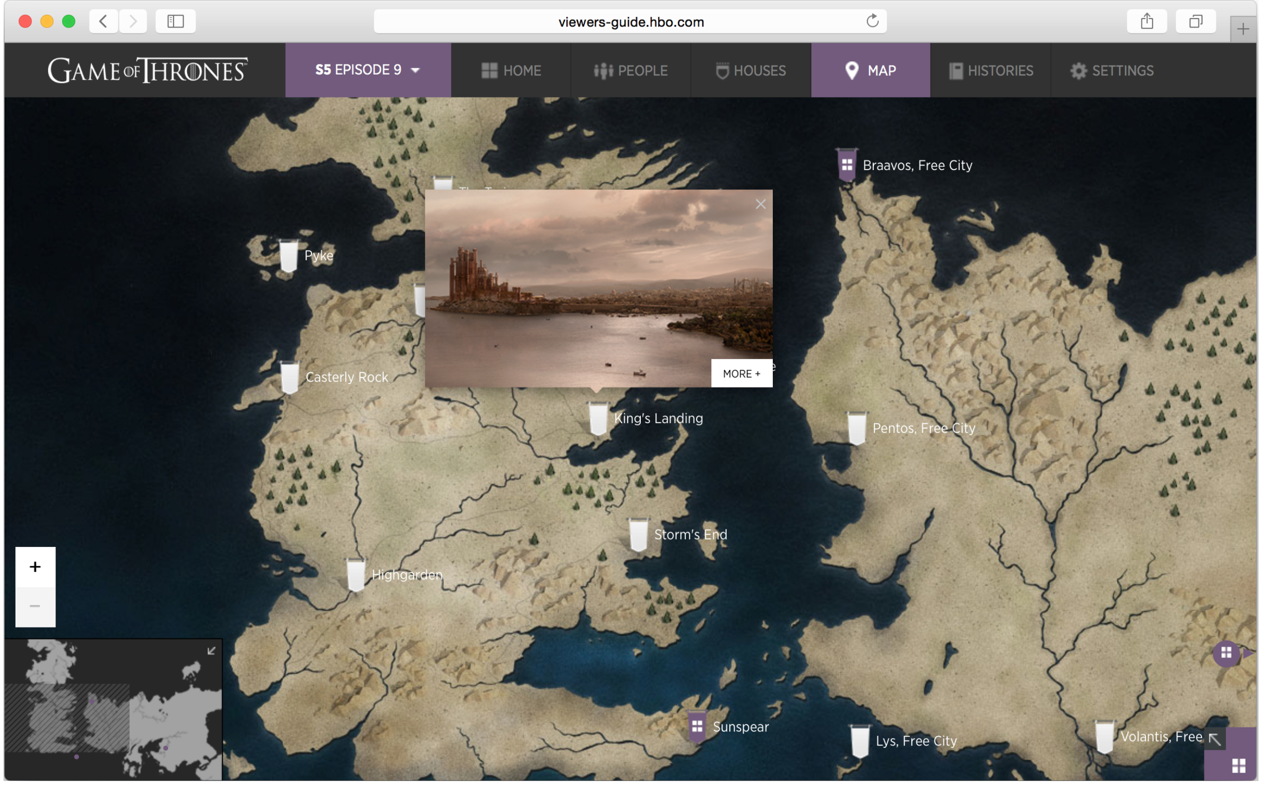


Figure 2.3: Game of Thrones, HBO Viewers Guide

Pro

- Locations are described thoroughly with text and photos.
- The content provided in the map is displayed with few visual distractions.
- The smaller map shows where the map outline is defined for the fullscreen map.

Con

- Full screen windows must be opened to read about the locations. This hinders the user from exploring the map and reading text at the same time.
- The service is tailored for one specific TV-show.
- The location photos cover a large section of the map when visible.

Conclusion

The service is intended as an information center where the reader can get a summary of the TV-episodes or learn more about the locations briefly named in the TV-series. Readers will most likely use this service in order to gain deeper knowledge of the series. The information gives the reader additional history and facts about the fantasy novel not provided by the TV-series. However, the user must have a good knowledge of the TV- or book-series prior to using this service, in order to make use of its content. This approach does not coincide with the thesis, because it focuses on providing literary, in its full form, alongside supplementary information, such as maps.

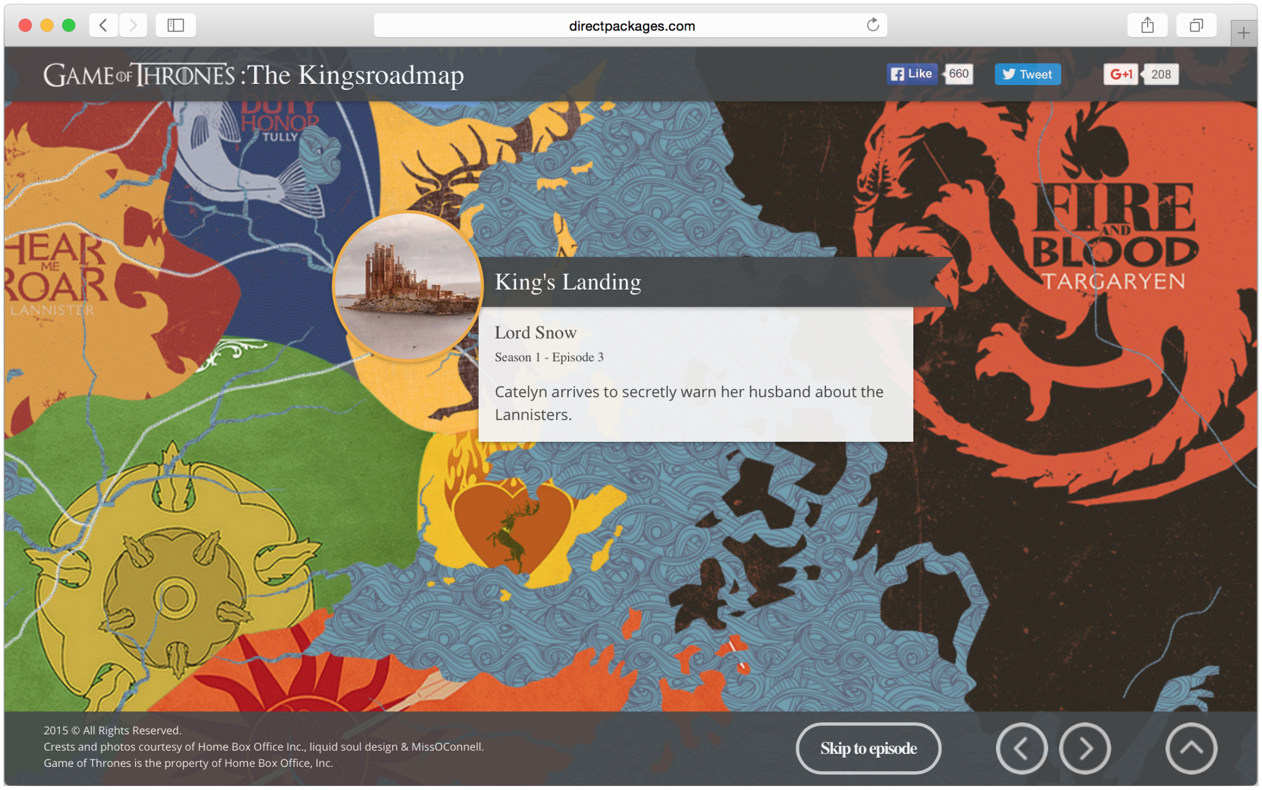
2.4 Game of Thrones, The Kingsroadmap

Description

The Kingsroadmap [5] is a web site that offers map representation of the popular HBO TV-show “Game of Thrones”. The service is intended as a guide that offers brief episode summary in form of text and map information. The web site shows a full screen view of the fictional map where the events of "Game of Trones" unfold. The map does not allow interaction in the form of translation or zooming. The user can view summary of one episode at a time. He can select a specific episode from a submenu or easily skip between episodes using navigation buttons. The map will, for a given episode, show a single geographical marker with an associated information window. The information window holds a brief episode summary. The marker will appear at the location where the course of event takes place. The text window and marker is relocated when the user skips between episodes. He must therefore shift focus in order to read the next, or previous, summary.

The map will also highlight areas of lands to indicate who it belongs to. This is illustrated by coloring land areas with the respective family banner. A family banner is similar to a nation flag, shown in Figure 2.4. Areas and families are not further described in the web site.

A lengthy animation is initiated when the user skips between episodes. Animations consist of updating the map center, relocating the marker and repopulating the information window. The animations clearly indicate where the new and previous map outlines are defined. However, the text is hidden during animation, which hinders the user from continuous reading.



King's Landing

Lord Snow

Season 1 - Episode 3

Catelyn arrives to secretly warn her husband about the Lannisters.

2015 © All Rights Reserved.
Crests and photos courtesy of Home Box Office Inc., liquid soul design & MissOConnell.
Game of Thrones is the property of Home Box Office, Inc.

Figure 2.4: Game of Thrones - The Kingsroadmap

Pro

- Areas, in addition to locations, are highlighted.
- The episode navigation buttons are easy to use.

- Map animations clearly indicate where the new and previous map outlines, i.e., center position, are defined.

Con

- Text is displayed at varying locations for different episodes. The user is therefore forced to shift focus whenever the text gets updated.
- Text is hidden during animations. This hinders continuos reading.
- The service is tailored for one specific book series.

Conclusion

The Kingsroadmap is intended as a brief episode guide for the TV show. It gives the user an alternative geographical representation via the map. The storyline is not included. Therefore, users must have knowledge of the TV-series or books in order make full use of the map. This thesis' goal is to provide users with storylines and supplementary information in form of maps. Therefore, the Kingsroadmap does not coincide with the thesis' goal.

2.5 Conclusion

The evaluated services does not provide alternative ways of reading literature, i.e., books. They either provide supplementary information to literature in form of maps, photos and brief summaries, or textual descriptions to a series of images or locations. Therefore, the content of the services evaluated does not coincide with this thesis' goal.

However, they provided valuable input for UI structure and reading flow. The prototype includes a form of text highlighting, inspired by in ESRI, and a timeline, inspired by the LOTR Project.

Chapter 3

Technology Decisions and Planning

This chapter presents planning of the thesis prototype in terms of underlying technology, structure, communication and which third party solutions are utilized.

3.1 Platform

The thesis prototype is developed as a web service. This decision is made primarily because web services do not force the end user to install any programs. He can, independently of Operating System (OS) and device, access the service via a web browser, given he has a valid Internet connection. Web services are also available to multiple users at the same time. An alternative would be to create device specific versions of the software for the most common OSs and platforms. The prototype would in this case have been developed for one specific platform and OS due to the limited timespan. This is a time consuming approach, which takes away valuable time from the overall thesis goal. Updates to web services are instantly available to all users because they reference the same data source. Device specific applications, however, would be kept in local memory and therefore needs to be re-downloaded when new updates becomes available.

The web service should give the end user a seamless reading experience, without being faced with loading screens prior to displaying new information. Delays occur in web browsers when data is requested from the database and when the Document Object Model (DOM) is redrawn, i.e., when the UI is redrawn to display new information. To achieve a seamless reading experience, communication with the database must be asynchronously, and the UI must be rendered continuously.

The web service has been divided into a Client side and a Server side. The Client side represents the web browser, and is kept in local memory once it is loaded. This is where the user is presented with a UI. The server side represents the web server, and will supply the end users with information. This is where the database and business logic is located. The business logic will, upon request from the client side, fetch data from the database and return it to the client side. This is illustrated in Figure 3.1.

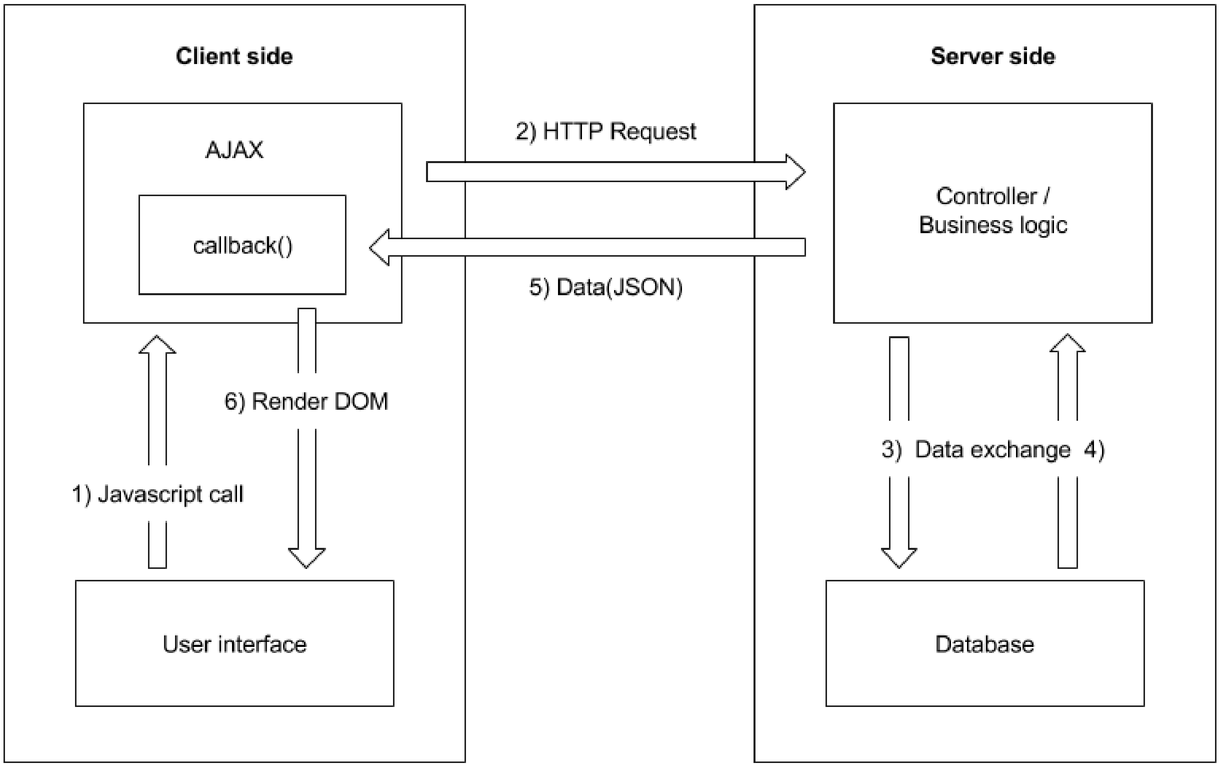


Figure 3.1: Client-Server communication.

Whenever new information is needed in the client's web browser, a Javascript (JS) event will fire. This in turn creates a HyperText Transfer Protocol (HTTP) request, which is sent from the client side to the server side. HTTP is the common protocol for exchanging and transferring hypertext in the World Wide Web (WWW). When the server side receives the HTTP request, the business logic, i.e the controller, will interpret the request, fetch the correct information from the database, process the data into JavaScript Object Notation (JSON) and return it to the client side. JSON is the standard format for transmitting data objects consisting of attribute–value pairs, and is primary used for asynchronous client and server communication. The client-server communication utilizes Asynchronous JavaScript and XML (AJAX), which is a combination of web development techniques used to create asynchronous web applications. AJAX allows data to be sent and retrieved in the background, without interfering with the behavior of the existing page. AJAX does this by providing a callback function to every request. Instead of waiting for the web server to respond to a request, it tells the web server to deliver the response on this callback when the response is ready. The callback, when it receives the response, will process the response before updating the UI with the new information. This approach will only render the UI where it is needed. The user will not suffer from the page being fully redrawn. The UI will stay responsive throughout the communication process, which allows the user to keep reading and interacting with the map.

3.2 Client Side

This section presents decisions made on behalf of the client side of the prototype. This includes markups of the UI, how the UI is monitored, how data is displayed and how data is inserted into the UI.

3.2.1 User Interface

As frequently experienced while evaluating existing services in Chapter 2, the focus lies mainly on providing readers with a large Map-view, and a small set of textual information to accompany these maps. The example storylines portrayed by the existing softwares are in most cases highly suited to be displayed within the context of a map or by photos, connected to a timeline. In these cases, the textual information does not require much attention, as the map is the main focal point. However, this thesis' goal is to provide the reader with the full storyline, i.e., a novel, and display key events in the context of a map and a timeline in addition. Therefore, the layout features a large Text-view on the leftmost side, to contain the novel. This is as opposed to having one smaller Text-view or multiple smaller Text-views displayed on top of the map. The Text-view is similar to a book page in shape and size. The layout is illustrated in Figure 3.2.

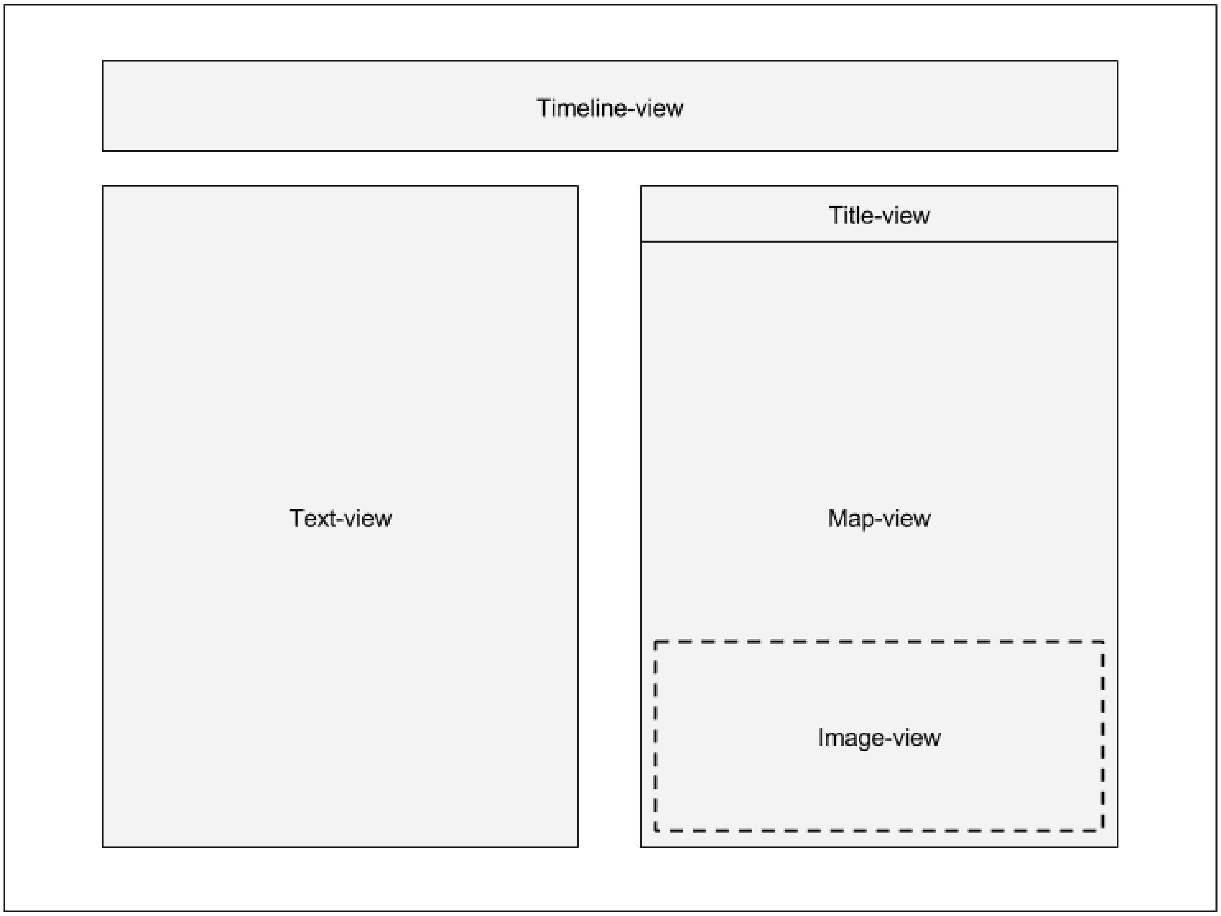


Figure 3.2: UI. Dotted square is pop-up window. Lined quares are permanent windows.

The second view element is the Map-view, displayed on the rightmost side in the layout. This view will display a single map. The map will be continuously updated with content, and dynamically change its appearance during the reading session. A single map is decided on as opposed to multiple maps in order to offer a simple service with a static layout. An alternative would be to separate the Map-view data into multiple Map-views. Since maps are often not able to display its content properly if being small, one single Map-view is utlilized. An Image-view is also included inside the Map-view. This is illustrated by the stapled frame inside the Map-view in Figure 3.2. The Image-view will display images associated with individual markers of interest in the map. When the reader clicks on a marker that has an associated photo, the Image-view becomes visible. Otherwise, the Image-view is hidden. A Title-view will be included on top of the Map-view to give a short description to what the map is currently displaying. Lastly, a Timeline-view is included on top of the UI. This view displays a timeline with dots, i.e., date-time stamps, to display when events took place.

3.2.2 Map Framework

The initial goal is to create a service that supports both fictive and non-fictive maps, i.e., fantasy maps and the World Atlas or parts thereof. Unfortunately, there are no map frameworks that supports both these cases. The initial goal would have required a new map framework developed specifically for the thesis. Developing a map framework would introduce a long list of time consuming tasks, including zooming, translation and defining a coordinate system to handle positioning of markers. Creating a new map framework falls outside the scope of this thesis, since it is to develop a functioning prototype. Therefore, the service is integrated with an existing map framework, which limits the service to the World Atlas.

There are several well built map frameworks that can be integrated into web services. The selected map framework should have the following features.

- The user must be able to translate the map, i.e., shift the point of view.
- The user must be able to adjust the zoom level.
- The map must be able to display multiple markers.
- Map markers must have a range of available colors.
- The map must be able to display paths.
- Paths must have a range of available colors.
- The map should display elevation data.
- The map should display location names.

All the map frameworks provided by Google [4], Bing [8], MapQuest [6] and OpenLayers [7] meet these requirements. From this map framework set, Google Maps is by far the most popular, as concluded in the article *"Top 10 Mapping APIs"* [16] by Janet Wagner. The article states that Google Maps offers the best documentation for integration and usage. The prototype consists of many map related features, such as drawing, editing and removing paths and markers, therefore, the Google Maps framework is applied. Google Maps is not optimal for showing elevation data and minor details such as ,e.g., trails, but it is sufficient for demonstrating purposes, i.e., investigating desired functionalities.

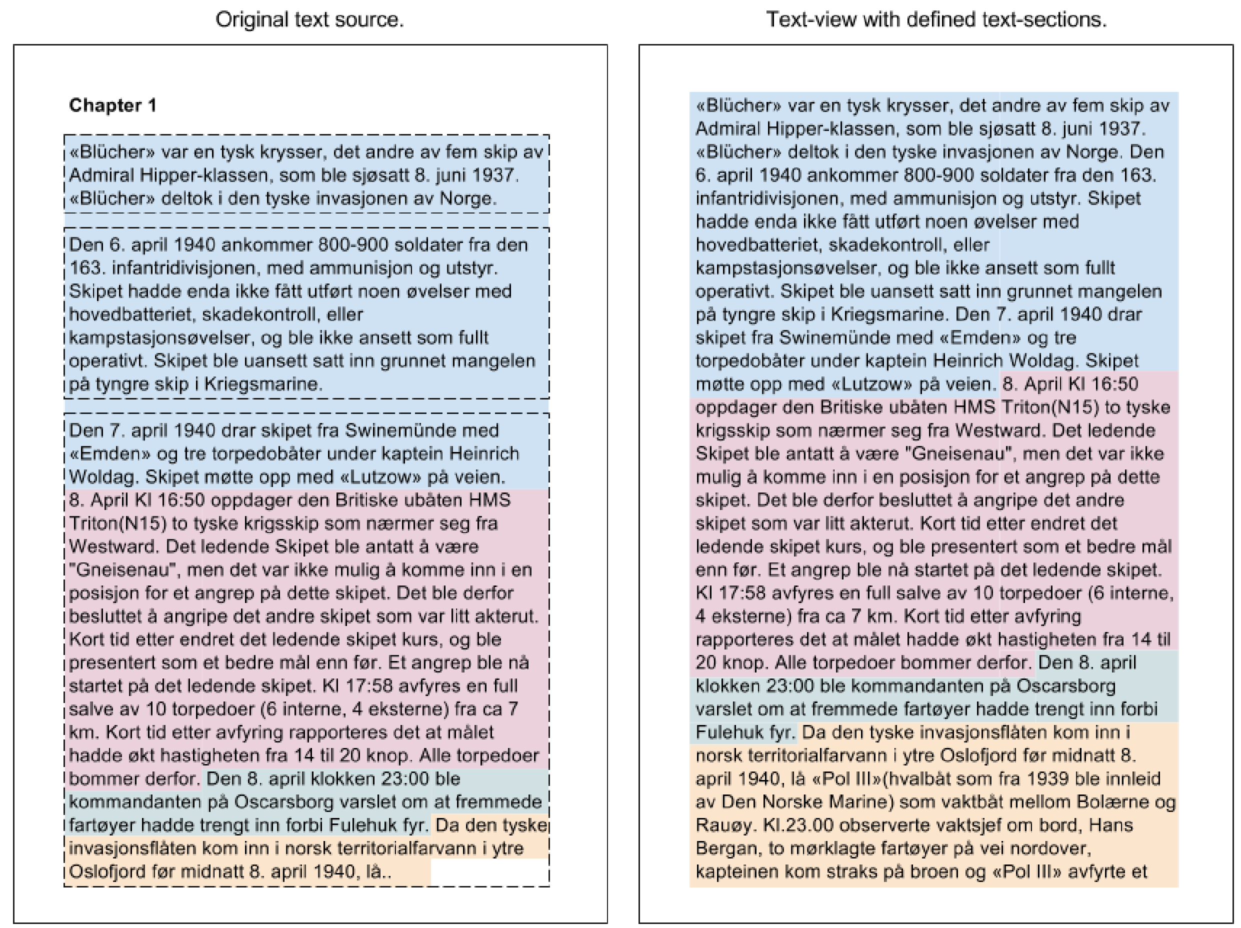
3.2.3 Text Representation

The Text-view must serve two purposes. Display the text and tell the service which supplementary information to display based on what part the user is currently reading. The info shown in the Map-, Title- and Timeline-views must always be coherent with the text being read. It is therefore important that the service has a way of monitoring the text, i.e., the readers text interaction. Textual content, such as a novel, is typically obtainable as a PDF-file. PDF-files can be included directly in the Text-view of the layout. It will, most likely, have the correct font sizes, paragraph offset, line-spacing, etc., so no additional textual styling is needed. However, it might be challenging using PDF-files because they are non-editable, and there is no way to monitor with it.

An alternative is to transform the textual content of the PDF-file into HyperText Markup Language (HTML), which will be accessible as nodes in the DOM, i.e., which is visualized in the UI. This approach is more time consuming than including a PDF-file, because text has to be translated into a new document format. In return, however, it allows the service to restructure the text, which in turn allows JS to communicate with the text. This is a crucial feature for the service. Therefore, the service utilizes transformed text.

The original text source will be divided into multiple text sections, i.e., restructured. A text section will be defined to enclose one specific event in the text that is considered reasonable to have represented in a map and a timeline. The text sections will not be defined by paragraphs or chapters from the original text source because they will most likely describe multiple events that should each be represented in the map and timeline at separate times. E.g. a book featuring the German attack on Oslo, Norway, in World War 2 would typically have a chapter about the Battle of Drøbak, where Oscarsborg Fortress opened fire on Blücher. However, the chapter may also contain events before and after the battle, such as its encounter with the British submarine HMS Triton (N15) outside of Denmark. Therefore, the exact part of the chapter that describes the Battle of Drøbak will be enclosed as a text section. The surrounding text will be enclosed in other text sections. The separation of text allows for map and timeline data to be created for the individual text sections, i.e., events. In this case, when the reader is currently reading about the battle, the map would ,e.g., display a marker on the exact coordinates of the Oscarsborg fortress. The timeline would highlight a dot with the exact battle date and time.

Transformation from an original text source into multiple text-sections, i.e., the HTML Text-view, is illustrated in Figure 3.3.



Original text source.

Chapter 1

«Blücher» var en tysk krysser, det andre av fem skip av Admiral Hipper-klassen, som ble sjøsatt 8. juni 1937. «Blücher» deltok i den tyske invasjonen av Norge.

Den 6. april 1940 ankommer 800-900 soldater fra den 163. infantridivisjonen, med ammunisjon og utstyr. Skipet hadde enda ikke fått utført noen øvelser med hovedbatteriet, skadekontroll, eller kampstasjonsøvelser, og ble ikke ansett som fullt operativt. Skipet ble uansett satt inn grunnet mangelen på tyngre skip i Kriegsmarine.

Den 7. april 1940 drar skipet fra Swinemünde med «Emden» og tre torpedobåter under kaptein Heinrich Woldag. Skipet møtte opp med «Lutzow» på veien. 8. April Kl 16:50 oppdager den Britiske ubåten HMS Triton(N15) to tyske krigsskip som nærmer seg fra Westward. Det ledende Skipet ble antatt å være "Gneisenau", men det var ikke mulig å komme inn i en posisjon for et angrep på dette skipet. Det ble derfor besluttet å angripe det andre skipet som var litt akterut. Kort tid etter endret det ledende skipet kurs, og ble presentert som et bedre mål enn før. Et angrep ble nå startet på det ledende skipet. Kl 17:58 avfyres en full salve av 10 torpedoer (6 interne, 4 eksterne) fra ca 7 km. Kort tid etter avfyring rapporteres det at målet hadde økt hastigheten fra 14 til 20 knop. Alle torpedoer bommer derfor. Den 8. april klokken 23:00 ble kommandanten på Oscarsborg varslet om at fremmede fartøyer hadde trengt inn forbi Fulehuk fyr. Da den tyske invasjonsflåten kom inn i norsk territorialfarvann i ytre Oslofjord før midnatt 8. april 1940, lå..

Text-view with defined text-sections.

«Blücher» var en tysk krysser, det andre av fem skip av Admiral Hipper-klassen, som ble sjøsatt 8. juni 1937. «Blücher» deltok i den tyske invasjonen av Norge. Den 6. april 1940 ankommer 800-900 soldater fra den 163. infantridivisjonen, med ammunisjon og utstyr. Skipet hadde enda ikke fått utført noen øvelser med hovedbatteriet, skadekontroll, eller kampstasjonsøvelser, og ble ikke ansett som fullt operativt. Skipet ble uansett satt inn grunnet mangelen på tyngre skip i Kriegsmarine. Den 7. april 1940 drar skipet fra Swinemünde med «Emden» og tre torpedobåter under kaptein Heinrich Woldag. Skipet møtte opp med «Lutzow» på veien. 8. April Kl 16:50 oppdager den Britiske ubåten HMS Triton(N15) to tyske krigsskip som nærmer seg fra Westward. Det ledende Skipet ble antatt å være "Gneisenau", men det var ikke mulig å komme inn i en posisjon for et angrep på dette skipet. Det ble derfor besluttet å angripe det andre skipet som var litt akterut. Kort tid etter endret det ledende skipet kurs, og ble presentert som et bedre mål enn før. Et angrep ble nå startet på det ledende skipet. Kl 17:58 avfyres en full salve av 10 torpedoer (6 interne, 4 eksterne) fra ca 7 km. Kort tid etter avfyring rapporteres det at målet hadde økt hastigheten fra 14 til 20 knop. Alle torpedoer bommer derfor. Den 8. april klokken 23:00 ble kommandanten på Oscarsborg varslet om at fremmede fartøyer hadde trengt inn forbi Fulehuk fyr. Da den tyske invasjonsflåten kom inn i norsk territorialfarvann i ytre Oslofjord før midnatt 8. april 1940, lå «Pol III»(hvalbåt som fra 1939 ble innleid av Den Norske Marine) som vaktbåt mellom Bolærne og Rauøy. Kl.23.00 observerte vaktsjef om bord, Hans Bergan, to mørklagte fartøyer på vei nordover, kapteinen kom straks på broen og «Pol III» avfyrte et

Figure 3.3: Transformation from PDF to HTML. Original text source to the left. Stapled square shows paragraphs. Text-view to the right. Colored squares enclose text-sections.

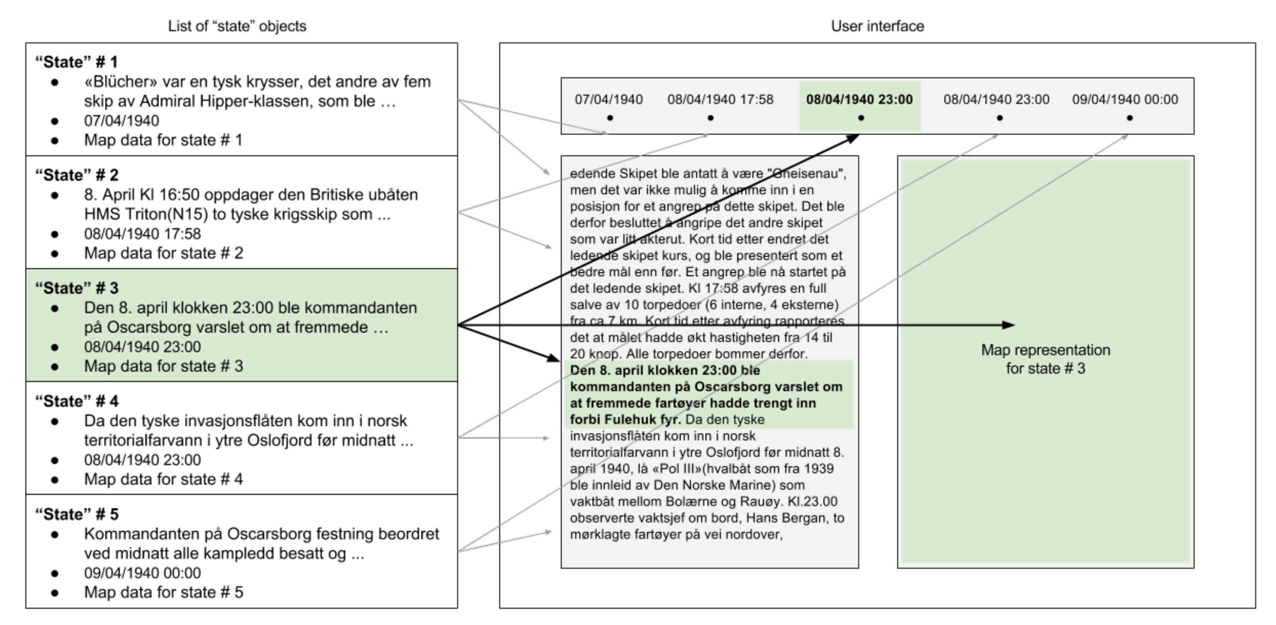
The left side of Figure 3.3 illustrates an existing text source, where dotted frames enclose paragraphs. The original text source has three paragraphs. The first two paragraphs and the beginning of the third paragraph describe one event that is considered reasonable to have represented in the map and timeline. This is illustrated with a blue square. The third paragraph describe three new events that are considered reasonable to have represented in the map and timeline. This is illustrated with pink, green and yellow squares. It would therefore have been illogical to create map and timeline data for each paragraph in the existing text source.

The right side of Figure 3.3 illustrates the HTML Text-view with text translated from the original text source. Colored squares illustrates text sections, i.e., separate events.

3.2.4 Triggering Updates

The storyline will be represented by multiple text sections that together constitutes the full storyline. Data groupings are created to contain text sections and coherent map and timeline data. Each data grouping consists of a single text section, the data needed to reflect the text in the map and timeline and an identifier. In other words, a single data grouping consists of data meant to be displayed and highlighted at the same time. These data groupings are defined as "state" objects in the client side, and is described in detail in Section 4.2.1. The "state" identifier, named "state value", will be incremented between the first and last "state" object.

Figure 3.4 illustrates how the UI is populated with data from multiple "state" objects.



“State” # 1
- «Blücher» var en tysk krysser, det andre av fem skip av Admiral Hipper-klassen, som ble …
- 07/04/1940
- Map data for state # 1

“State” # 2
- 8. April Kl 16:50 oppdager den Britiske ubåten HMS Triton(N15) to tyske krigsskip som ...
- 08/04/1940 17:58
- Map data for state # 2

“State” # 3
- Den 8. april klokken 23:00 ble kommandanten på Oscarsborg varslet om at fremmede …
- 08/04/1940 23:00
- Map data for state # 3

“State” # 4
- Da den tyske invasjonsflåten kom inn i norsk territorialfarvann i ytre Oslofjord før midnatt ...
- 08/04/1940 23:00
- Map data for state # 4

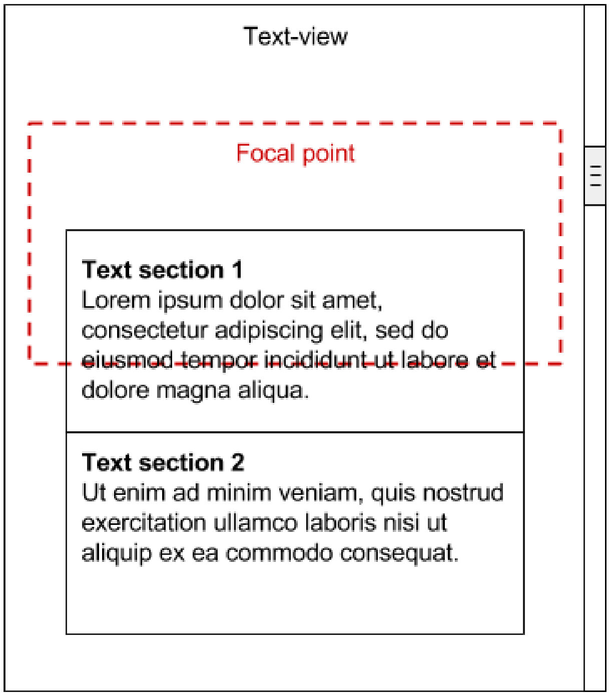
“State” # 5
- Kommandanten på Oscarsborg festning beordret ved midnatt alle kampledd besatt og ...
- 09/04/1940 00:00
- Map data for state # 5

edende Skipet ble antatt å være "Gneisenau", men det var ikke mulig å komme inn i en posisjon for et angrep på dette skipet. Det ble derfor besluttet å angripe det andre skipet som var litt akterut. Kort tid etter endret det ledende skipet kurs, og ble presentert som et bedre mål enn før. Et angrep ble nå startet på det ledende skipet. Kl 17:58 avfyres en full salve av 10 torpedoer (6 interne, 4 eksterne) fra ca 7 km. Kort tid etter avfyring rapporteres det at målet hadde økt hastigheten fra 14 til 20 knop. Alle torpedoer bommer derfor. **Den 8. april klokken 23:00 ble kommandanten på Oscarsborg varslet om at fremmede fartøyer hadde trengt inn forbi Fulehuk fyr.** Da den tyske invasjonsflåten kom inn i norsk territorialfarvann i ytre Oslofjord før midnatt 8. april 1940, lå «Pol III»(hvalbåt som fra 1939 ble innleid av Den Norske Marine) som vaktbåt mellom Bolærne og Rauøy. Kl.23.00 observerte vaktsjef om bord, Hans Bergan, to mørklagte fartøyer på vei nordover,

Figure 3.4: Populating the UI with data. The current "state" is highlighted with green background. Arrows show where "state"-data is displayed. Multiple "states" populate the Timeline-view and Text-view, but only the highlighted populate the Map-view.

A collection of "state" objects are stored in a list accessible to Client Side JS. This is presented in Section 4.1.3. In Figure 3.4, the UI is populated with data from five "state" objects. The text section from each "state" is inserted into the Text-view, and the timeline data, i.e., the date-time value, for each "state" is inserted into the Timeline-view. The data will be inserted into the UI as HTML. This way, text section and timeline data from all "states" will be visible in the UI at the same time. However, only data from one "state" is displayed in the Map-view. The map will display data coherent with "state" number three if the user is reading the text section in "state" number three.

In order to trigger updates to the map, the service must be able to recognize which text section the reader is currently reading. In order to achieve this, each HTML text section will be given an embedded anchor. The purpose of anchors is to tell the service when its text section is inside a defined Text-view focal point via a function. The Text-view will have a defined focal point. Each anchor will compare its position in the Text-view against the focal point. If the anchor recognizes that its text section is within the focal point, it will trigger its function. This is illustrated in Figure 3.5.



Text section 1
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Text section 2
Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Figure 3.5: Focal point of Text-view.

Each anchor will hold the identifier from the respective "state" object. This identifier will be passed as a function parameter. When a function is triggered, the "state" corresponding with the parameter from the function will be retrieved from storage. The map can then be updated with map data stored in the retrieved "state" object. Implementation of the anchors are described in detail in Section 4.1.2.

In Figure 3.5, "Text section 1" is inside the defined focal point range, and will tell the map to display map content related to "Text section 1". As the reader progresses, i.e., scrolls the content of "Text section 2" into the focal point, the map will be updated once again to display content related to "Text section 2". This approach is used because it allows the reader to pay full attention to the text, map and timeline, and thereby provide a seamless reading experience.

3.3 Server Side

This section presents the underlying technologies and code languages that has been used, how data is stored and how the implementation is structured.

3.3.1 Programming Language

The server side development is based on Microsoft web service technology. Microsoft uses the programming language C#, which is object oriented. Microsoft technology is used mainly because it offers an extensive library, .NET, for developing web based services. The library has functionality for building client side UIs, as well as communication between client and server side.

3.3.2 Database

A Structured Query Language (SQL) database is used as the underlying data source provider. The SQL database is also supported by Entity Framework (EF), which is an Object-Relational Mapping (ORM) framework included in the .NET library. EF works as a layer between the business layer, i.e., the logic, and the SQL database. Its purpose is to mask the syntax used to form and interact with the SQL database. It does this by mapping object oriented classes into database entities. EF is used because it gives the possibility to deal with data entities in one specific way throughout their lifecycle on the server side. These object oriented classes are also more suited to be transferred into JSON format, which is the format used to send information to the client side of the service.

Chapter 4

Implementation

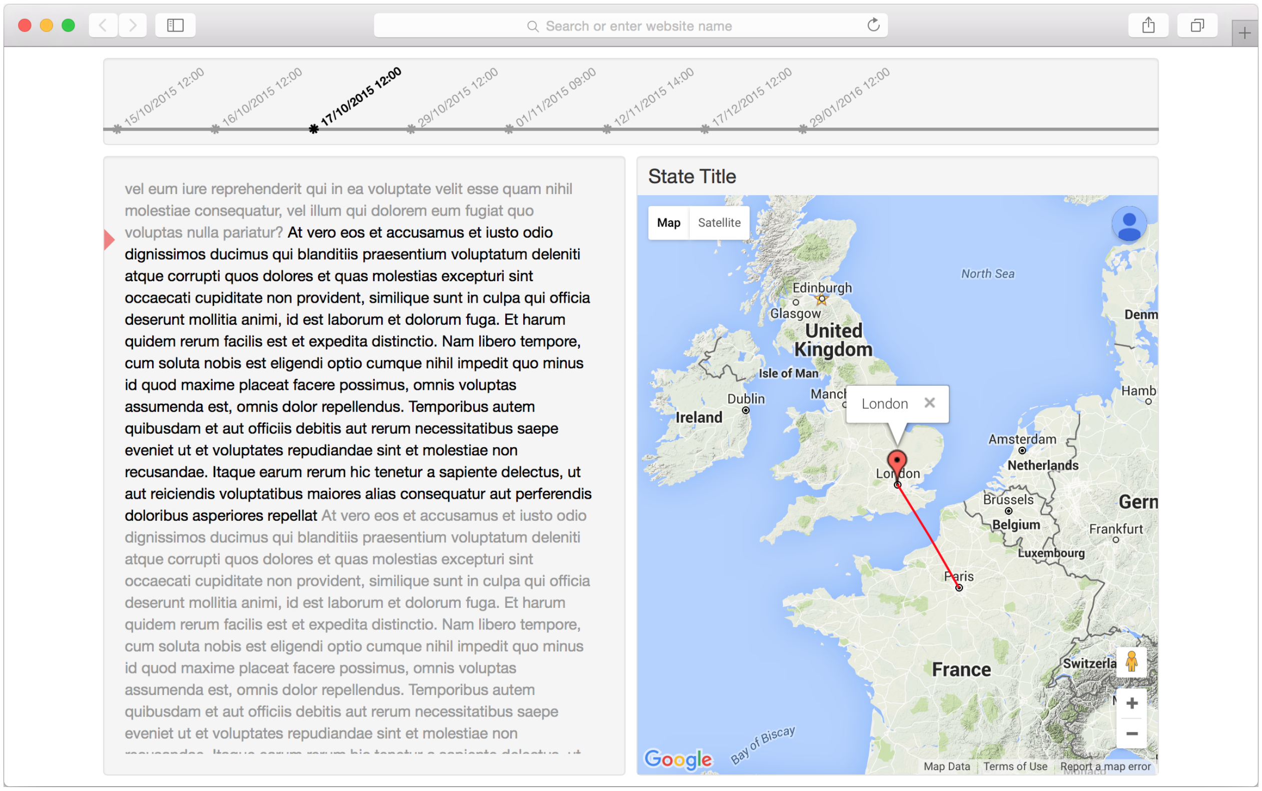
This chapter presents the methods and technology used for implementing the prototype.

4.1 Client Side

The client side consists of a UI available in any web browser. The web browser utilizes JS to manage updates to the UI and client server communication.

4.1.1 User Interface

The UI is implemented based on Figure 3.2. The final result is shown in Figure 4.1. It is implemented using HTML for structure and Cascading Style Sheets (CSS) for styling. JS is used to handle updates to the map, timeline and text, the readers activities, data group storage and client server communication.



State Title

vel eum iure reprehenderit qui in ea voluptate velit esse quam nihil molestiae consequatur, vel illum qui dolorem eum fugiat quo voluptas nulla pariatur? At vero eos et accusamus et iusto odio dignissimos ducimus qui blanditiis praesentium voluptatum deleniti atque corrupti quos dolores et quas molestias excepturi sint occaecati cupiditate non provident, similique sunt in culpa qui officia deserunt mollitia animi, id est laborum et dolorum fuga. Et harum quidem rerum facilis est et expedita distinctio. Nam libero tempore, cum soluta nobis est eligendi optio cumque nihil impedit quo minus id quod maxime placeat facere possimus, omnis voluptas assumenda est, omnis dolor repellendus. Temporibus autem quibusdam et aut officiis debitis aut rerum necessitatibus saepe eveniet ut et voluptates repudiandae sint et molestiae non recusandae. Itaque earum rerum hic tenetur a sapiente delectus, ut aut reiciendis voluptatibus maiores alias consequatur aut perferendis doloribus asperiores repellat At vero eos et accusamus et iusto odio dignissimos ducimus qui blanditiis praesentium voluptatum deleniti atque corrupti quos dolores et quas molestias excepturi sint occaecati cupiditate non provident, similique sunt in culpa qui officia deserunt mollitia animi, id est laborum et dolorum fuga. Et harum quidem rerum facilis est et expedita distinctio. Nam libero tempore, cum soluta nobis est eligendi optio cumque nihil impedit quo minus id quod maxime placeat facere possimus, omnis voluptas assumenda est, omnis dolor repellendus. Temporibus autem quibusdam et aut officiis debitis aut rerum necessitatibus saepe eveniet ut et voluptates repudiandae sint et molestiae non recusandae. Itaque earum rerum hic tenetur a sapiente delectus, ut

Figure 4.1: UI.

The Map-view on the rightmost side holds a single Google Map. The single map is populated with data from one data grouping, i.e., "state", at the time via JS. Map data is displayed as location markers and paths. The reader is able to explore the maps outline by translating its center and adjusting the zoom level. Translating is done by dragging the mouse cursor inside the map. Zoom adjustment is done with buttons provided by the map framework. Map elevation data is enabled for better recognition of mountains and valleys. The reader is also able to view streets and view photos via the map container. This is done by dropping a provided pin on an arbitrary location, given that such data exists for the location in question. This is shown in Figure 4.2. The Map-view contains an Image-view, which is toggled between a visible and hidden state. In case the reader clicks on a location marker in the map and the location marker has been provided with a complementary photo from the "state" object, the image will appear in the bottom of the Map-view. The Image-view is also complemented with a photo caption overlay. The photo caption consists of the locations title and description and is found in the "state" object. An example of this is shown in Figure 4.3. The Map-view also contains a Title-view in the top. The Title-view shows a short description corresponding to the information being displayed in the map.

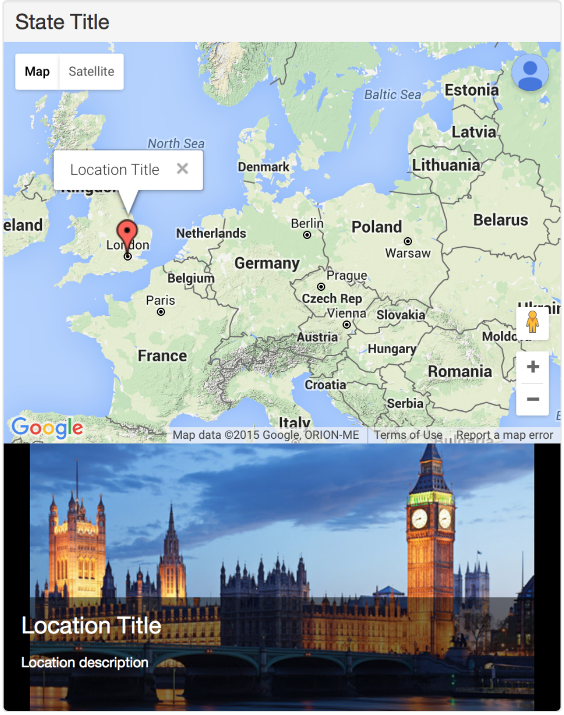


Figure 4.2: Image-view associated with map marker.

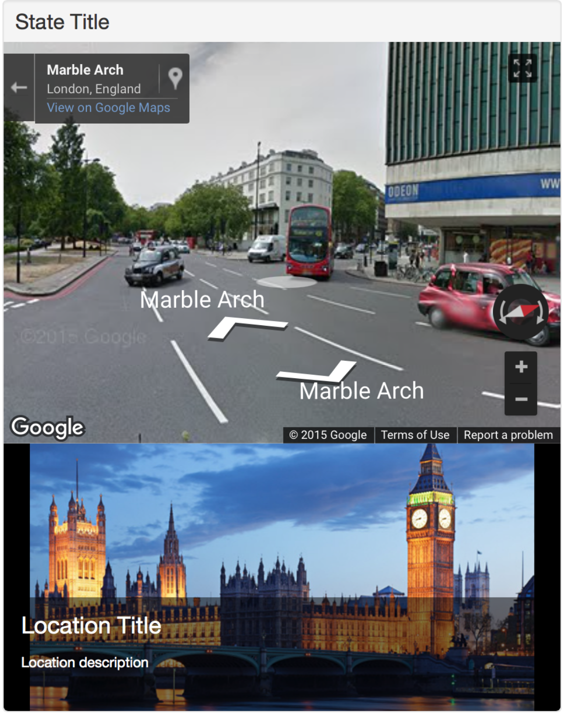


Figure 4.3: Map in Google Street view.

The Timeline-container is displayed on top of the UI, and shows a horizontal list of timeline information from "state" objects. Timeline information is represented by date-time values, i.e., dots. The single date-time value that corresponds to the text section currently being read in the Text-view will be highlighted with a darker color in order to stand out. The timeline is shown in Figure 4.4.

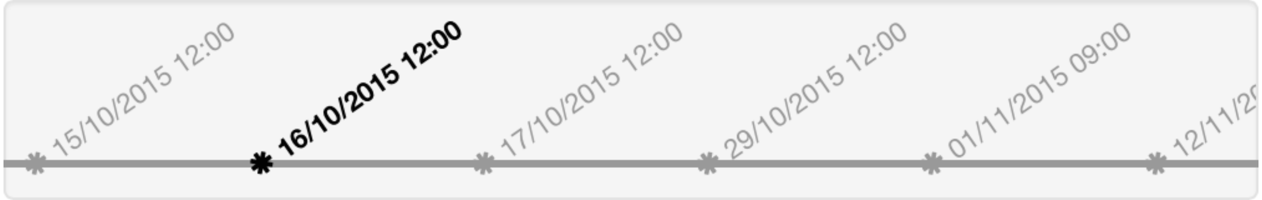
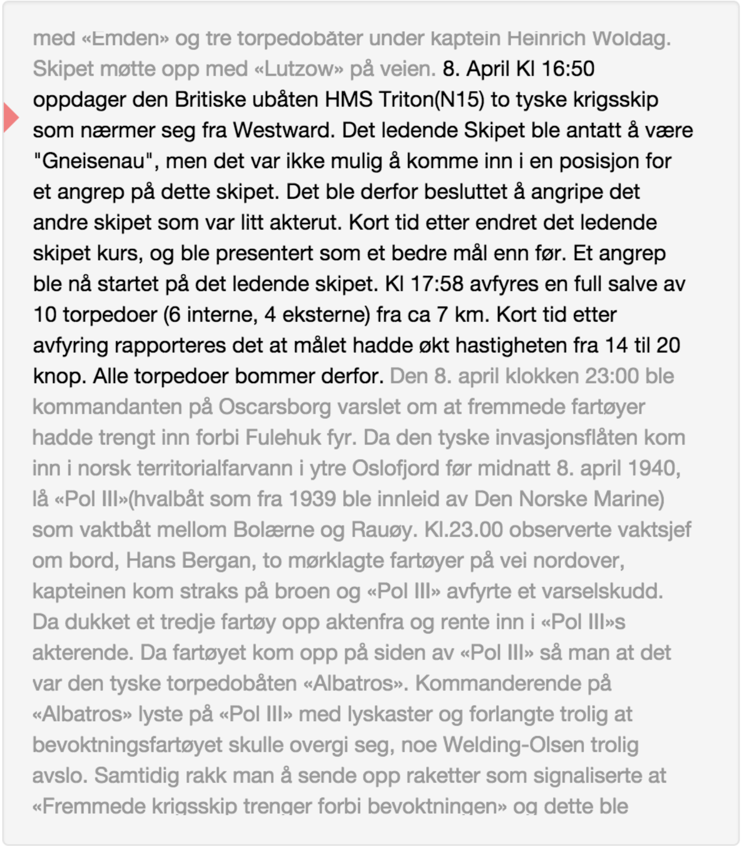


Figure 4.4: Timeline in UI.

The dots in the timeline are listed ascending from left to right. The dots have a fixed distance between each other. The timeline can therefore only display a subset of dots at the time if the number exceeds the length of the timeline. The dots overflowing will initially appear out of bounds on the rightmost side, and are therefore hidden. The dots are shifted one step to the left when the highlighting reaches one step beyond the rightmost dot, i.e., when the highlighting goes out of bounce. From this point and out, the highlighting will stick to the rightmost visible dot.

The Text-view is located at the leftmost side of the UI. It is implemented as a scroll-view. Scroll-views are one of the most common forms for displaying large amounts of text. The reader is likely to be accustomed to this. The Text-view is populated with text sections from multiple "state" objects at the same time. The text sections are injected into the Text-view as HTML, utilizing JS. The individual text sections overlap seamlessly, and will not introduce gaps at unwanted places. The text, therefore, appears as one larger section even though it consists of multiple sections. The text section that is currently inside the focus point of the Text-view will be highlighted with a darker color, much similar to highlighting in the Timeline-view. This is illustrated in Figure 4.5. The highlighted text section will correspond with the highlighted dot in the Timeline-view and the content being displayed in the map.



med «Emden» og tre torpedobåter under kaptein Heinrich Woldag. Skipet møtte opp med «Lutzow» på veien. 8. April Kl 16:50 oppdager den Britiske ubåten HMS Triton(N15) to tyske krigsskip som nærmer seg fra Westward. Det ledende Skipet ble antatt å være "Gneisenau", men det var ikke mulig å komme inn i en posisjon for et angrep på dette skipet. Det ble derfor besluttet å angripe det andre skipet som var litt akterut. Kort tid etter endret det ledende skipet kurs, og ble presentert som et bedre mål enn før. Et angrep ble nå startet på det ledende skipet. Kl 17:58 avfyres en full salve av 10 torpedoer (6 interne, 4 eksterne) fra ca 7 km. Kort tid etter avfyring rapporteres det at målet hadde økt hastigheten fra 14 til 20 knop. Alle torpedoer bommer derfor. Den 8. april klokken 23:00 ble kommandanten på Oscarsborg varslet om at fremmede fartøyer hadde trengt inn forbi Fulehuk fyr. Da den tyske invasjonsflåten kom inn i norsk territorialfarvann i ytre Oslofjord før midnatt 8. april 1940, lå «Pol III»(hvalbåt som fra 1939 ble innleid av Den Norske Marine) som vaktbåt mellom Bolærne og Rauøy. Kl.23.00 observerte vaktsjef om bord, Hans Bergan, to mørklagte fartøyer på vei nordover, kapteinen kom straks på broen og «Pol III» avfyrte et varselskudd. Da dukket et tredje fartøy opp aktenfra og rente inn i «Pol III»s akterende. Da fartøyet kom opp på siden av «Pol III» så man at det var den tyske torpedobåten «Albatros». Kommanderende på «Albatros» lyste på «Pol III» med lyskaster og forlangte trolig at bevoktningsfartøyet skulle overgi seg, noe Welding-Olsen trolig avslo. Samtidig rakk man å sende opp raketter som signaliserte at «Fremmede krigsskip trenger forbi bevoktningen» og dette ble

Figure 4.5: Text-view inside UI.

4.1.2 Triggering Updates

Updates to the UI, i.e., map, timeline, title and text, will take effect when the reader skips from one text section to the next, etc. The data that is displayed in the map and the title and highlighted in the timeline is determined by the text section that is currently being read, i.e., the highlighted text within the focal point of the Text-view.

The text section in each "state" object is given a surrounding span structure when being inserted into the Text-view as HTML. This structure will, in turn, allow JS to detect skips between text sections. A span is a HTML structure that can surround other HTML objets and hold on to values at the same time. In this case, a span will surround a text section and hold on to its respective “state value". The spans are classified as "anchors". All of the span structures, i.e., text sections, are collected inside a HTML structure that represents the Text-view. This structure is given the identifier “text-view".

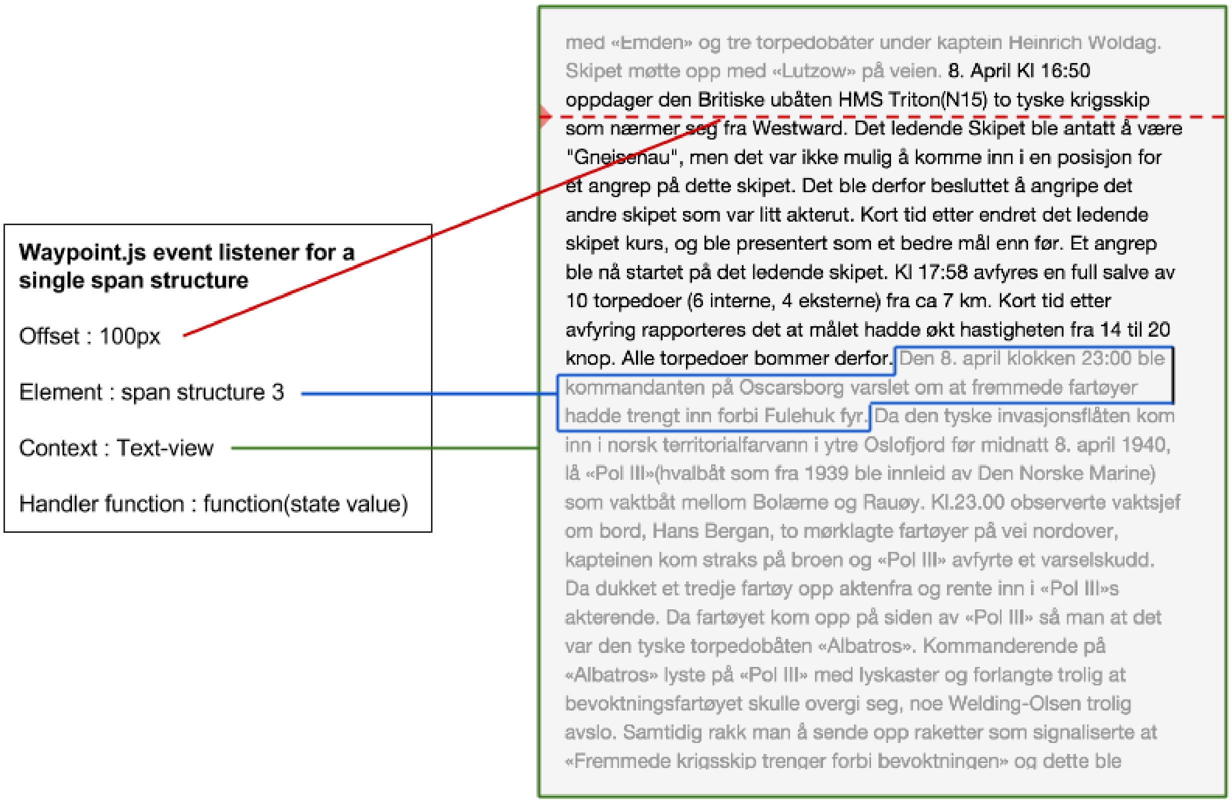
The complete HTML structure for the Text-view is shown in Listing 1.

```
<div id="text-view">
    <span id="state-value-1" class="anchor highlighted">
        Text section 1
    </span>
    <span id="state-value-2" class="anchor">
        Text section 2
    </span>
    <span id="state-value-3" class="anchor">
        Text section 3
    </span>
</div>
```

Listing 1: Insertion of waypoints in HTML.

The service must register event listeners to each span structure, classified as "anchor", in the Text-view. This is done in order to communicate with the "anchors". A third party JS framework, named Waypoints.js, is used to register the event listeners. The purpose of event listeners is to alert the software when the reader progresses from one text section to the next, etc.

The client side JS iterates through each span structure, i.e., text section, in the HTML Text-view. Each span-element is given an event listener, provided by Waypoint.js, consisting of four values, an "element", a "handler function", a "context" and an "offset value". A single event listener is illustrated in Figure 4.6.


Waypoint.js event listener for a single span structure
Offset : 100px
Element : span structure 3
Context : Text-view
Handler function : function(state value)

med «Emden» og tre torpedobåter under kaptein Heinrich Woldag. Skipet møtte opp med «Lutzow» på veien. 8. April Kl 16:50 oppdager den Britiske ubåten HMS Triton(N15) to tyske krigsskip som nærmer seg fra Westward. Det ledende Skipet ble antatt å være "Gneisenau", men det var ikke mulig å komme inn i en posisjon for et angrep på dette skipet. Det ble derfor besluttet å angripe det andre skipet som var litt akterut. Kort tid etter endret det ledende skipet kurs, og ble presentert som et bedre mål enn før. Et angrep ble nå startet på det ledende skipet. Kl 17:58 avfyres en full salve av 10 torpedoer (6 interne, 4 eksterne) fra ca 7 km. Kort tid etter avfyring rapporteres det at målet hadde økt hastigheten fra 14 til 20 knop. Alle torpedoer bommer derfor. Den 8. april klokken 23:00 ble kommandanten på Oscarsborg varslet om at fremmede fartøyer hadde trengt inn forbi Fulehuk fyr. Da den tyske invasjonsflåten kom inn i norsk territorialfarvann i ytre Oslofjord før midnatt 8. april 1940, lå «Pol III»(hvalbåt som fra 1939 ble innleid av Den Norske Marine) som vaktbåt mellom Bolærne og Rauøy. Kl.23.00 observerte vaktsjef om bord, Hans Bergan, to mørklagte fartøyer på vei nordover, kapteinen kom straks på broen og «Pol III» avfyrte et varselskudd. Da dukket et tredje fartøy opp aktenfra og rente inn i «Pol III»s akterende. Da fartøyet kom opp på siden av «Pol III» så man at det var den tyske torpedobåten «Albatros». Kommanderende på «Albatros» lyste på «Pol III» med lyskaster og forlangte trolig at bevoktningsfartøyet skulle overgi seg, noe Welding-Olsen trolig avslo. Samtidig rakk man å sende opp raketter som signaliserte at «Fremmede krigsskip trenger forbi bevoktningen» og dette ble

Figure 4.6: Illustration of a single event listener. The blue square is a span structure. The green square is the Text-view. The red dotted line shows the offset value from the top of the Text-view. The handler function will trigger when the blue square is above the red dotted line.

The span structure itself is specified as the "element" value in the event listener. This tells Waypoint.js to apply the event listener to this specific span structure. The "context" is the HTML Text-view, which tells Waypoint.js to operate within this part of the UI. The "offset value" is a minimum distance between the top of the Text-view and the "element", i.e., span structure, in question. If this distance is less than the specified offset, the "handler function" will be triggered. The "handler function" will, when triggered, extract the “state value” from the span structure in question, and send it as a parameter to another part of script which is in charge of updating the map and the timeline. This script is described in detail in Section 4.1.3. The "handler function" will also give the text section a darker color, to indicate that it is now corresponding to the new data soon to be displayed in the map and highlighted in the timeline. The core logic of the script is shown in Listing 2.

```
//Function used to add waypoint for each text fragment object
function addWaypoints(){
  var waypoints = document.getElementsByClassName(’anchor’)
  //For each waypoint in the text-view
  for (var i = 0; i < waypoints.length; i++) {
  //Add waypoint
  new Waypoint({
      element: waypoints[i],
      handler: function() {
        var id = this.element.id;
        $("#text-content > .anchor").removeClass("highlighted");
        $("#" + id).addClass("highlighted");
        updateModel(id);
      },
      context: document.getElementById(’text-view’),
      offset: (_textOffset*100).toString() + ’\’
    });
  }
}
```

Listing 2: Registering span structures in the Text-view with Waypoint.js event listeners

4.1.3 Scripts

The client side JS must keep references to the different aspects of the UI, manage local storage of storyline related data, be able to communicate with the web server, monitor the readers actions, and manipulate the UI. The script runs in the web browser when the user opens the UI.

The script will set up a single instance of Google Maps when it is initially run. This instance will be kept alive throughout the readers session. There is a noticeable delay when the map is initially drawn. By updating the same instance with new content, rather than redrawing the map at every update, the reader only witnesses this delay once in the start-up phase.

Next, the script will request the first set of "states" from the web server via a HTTP request. Communication with the web server introduces a short delay to the service. This means that the reader will be kept on hold when new data is collected. The reader will suffer a high frequency of short delays if communication with the web server is frequently needed, i.e., if only a small set of "states" is returned at a time. The reader will suffer one long delay if communication with the web server is limited to one request, i.e., the entire set of "states" is returned. Therefore, the script will request a set of "states" when the user progresses from one chapter to the next, i.e., prefetch one chapter at a time. Each "state" object will have a value that specifies which chapter it belongs to. These chapters are defined based on the original text source.

"States" are returned from the web server as a list in JSON format. Data must be received in JSON format in order to be processable by the JS. The "state" object is described in detail in Section 4.2.1. The HTTP request and the web server response is sent asynchronously. This ensures that the script does not pause, and can still perform all of its other tasks while communicating with the web server.

The response, i.e., the list of "state" objects, is further processed by the script into a hash map. A hash map is a key-value structure for storing data, where the key is used as a lookup reference to access values. A hash map has quicker lookup than a list because it is not necessary to iterate through the entire collection in order to find a specific object. In this case, it will extract the "state value" from each "state" in the response list and build the hash map with "state values" as the key and "Story states" as values.

Further, the timeline and Text-view is populated with data. The script iterates through the "state" objects in the hash map and collect the text sections and date-time values. The data is injected into the Text-view and Timeline-view respectively as HTML. All of the timeline data and text section data is inserted into the UI when a new list of "state" objects is received from the web server. Text secions are given "anchors" when injected, as described in Section 4.1.2.

When the reader starts reading, i.e., the first text section is position within the focal point of the Text-view, an anchor update function is triggered with the "state value" from the first text section as parameter. The function will use this "state value" as key in the hash map in order to retrieve the first "state" object. The text section in the Text-view and dot in the Timeline-view that corresponds to the retrieved "state" object will be highlighted, and the map will be populated with data from the retrieved "state" object.

The map is updated in multiple stages. First, all existing data in the map is cleared out to make room for new data. The new map data is located in the retrieved "state" object. The center position and the zoom level for the map is then specified. All positions in the "state" object are stored as longitude-latitude pairs, which is the format used by Google Maps. The script then iterates through each location, i.e., location marker, in the "state" object. For each location, the script extracts the title, description, position, path, marker color, path color, path thickness and complementary image. These values are mapped into a function provided by Google Maps in order to create a map marker.

A pop-up is also created for each map marker. These pop-ups are small text container placed directly above the markers to display the title of the location. Only one pop-up can be visible at the time. When the reader clicks on a marker, its pop-up will be displayed and the other visible pop-ups will be hidden. This is done to not clutter the map with too much textual information. Clicking the marker will, in addition to displaying its pop-up, display its complementary image in the Image-view, given that an image exists for the location in question. A pop-up is shown in Figure 4.7. Lastly, the title of the "state" is injected into the Title-view.

The script will perform a new lookup in the hash map every time the reader shift from one text section to the next or previous, i.e., when the "state value" is incremented or decremented. The map will be updated right away if the corresponding "state" is found locally in the hash map. Otherwise, the script will request the next set of "states" from the web server, i.e., the next chapter. The Text-view and Timeline-view is redrawn, and the hash map is updated with the new key-value pairs when a new set of "states" is received.

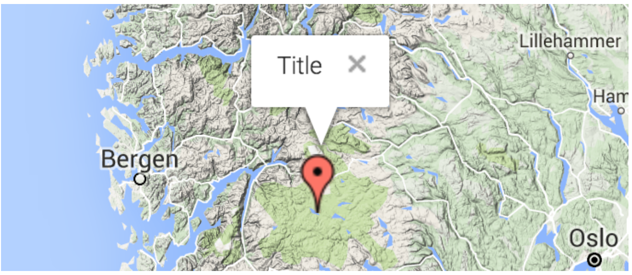


Figure 4.7: Map marker with pop-up

4.2 Server Side

The server side is responsible for storing data in a database (DB) and sending data to the client side.

4.2.1 Response Model

The response model, shown in Listing 3, is an object oriented representation of a "state" object. It holds a single text section together with map and timeline data coherent with the text section. It is named "state" object on the client side, and response model on the server side.

The response model, i.e., the "state" object, is populated with data from the DB based on a "state value", i.e., the identifier used to identify "states". The query used to collect data from the DB for response models is described in Section 4.2.3. The purpose of the response model is to hold the response of this DB query.

The response model will be used as an intermediary data structure to send data from the DB to the client side. It will not be used as the underlying structure for the DB itself. This is beneficial for two reasons. First, the response model contains the exact data needed by the client side for each text section, and nothing more. A lot of unnecessary data would be sent to the client side if the underlying data structure were transferred instead of the response model. This would generate larger transfer files which in turn results in greater transfer delays.

Second, if the response models were used as the underlying data structure, a lot of duplicate data would be stored. E.g, each "state" can contain the same "location". This "location" should only be stored once in the DB. This problem is avoided by having a separate underlying DB structure designed for optimal storing without duplicates and an intermediary response model.

The underlying data structure model is described in Section 4.2.2.

```
public class Position{
    public int Lat { get; set; }
    public int Lng { get; set; }
}
public class Location{
    public string Title { get; set; }
    public string Description { get; set; }
    public Position Position { get; set; }
    public bool IsHighlighted { get; set; }
    public string ImageUrl { get; set; }
    public Path Path { get; set; }
    public string MarkerColor { get; set; }
    public string PathColor { get; set; }
    public string PathThickness { get; set; }
    public Position Direction { get; set; }
}
public class ResponseModel{
    public int StateValue { get; set; }
    public string TextSection { get; set; }
    public string Title { get; set; }
    public string DateTime { get; set; }
    public Position CenterPosition { get; set; }
    public int Zoom { get; set; }
    public List<Location> Locations { get; set; }
}
```

Listing 3: Response model as object oriented classes.

The response model includes a text-section, that will be displayed in the Text-view, a "date-time" that will be displayed in the timeline" and a "title" that will be displayed in the Title-view. It also contains a "state value" identifier and values needed by Google Maps to a create map representation of the "state". The "state value" identifier has multiple purposes. The client side script uses it as a key to access "state" objects from local storage, Waypoint.js uses the identifier to alert which text section is inside the focal point of the Text-view and the server side queries uses it to collect data from the DB.

The map data consist of a "center-position" to set the outline in Google Maps, a "zoom" value to set the initial zoom value and a list of "locations" that will be inserted into the map as location markers. A location consist of a "position" represented by a latitude-longitude pair, a "path" represented by multiple "positions", a "direction", colors for the marker and the path line and a value for path thickness. "Directions" are drawn in the map as a vector between two points, represented by a dotted line. The dotted line starts in the location marker's center and ends in the "position" provided by the "direction" value. A "position", i.e., a latitude-longitude pair, is therefore stored to represent the "direction" value. A "location" also contains a "title", which will be used in the markers pop-up, a "description" and "image" which in combination with the "title" will populate the Image-view and its caption. Lastly, it contains a value to determine if the markers should have its pop-up and Image-view visible when the "state" is initially presented in the UI. Only one marker can have this enabled at the time.

The response model is created as an object oriented class on the web server. This is the same format used by EF to create the underlying data structure for the DB, which allows query results to be directly placed inside the response model without any additional formatting. However, the response model must be formatted into JSON before it is transmitted to the client side. This is because JS does not recognize the object oriented classes used on the web server. The conversion from an object oriented class into JSON is performed utilizing a Microsoft framework named Newtonsoft.JSON, which is found in the .NET library. JSON uses attribute-value pairs to store data, whereas object oriented classes uses class properties with values. The object oriented class property values are mapped into HTTP values with their property name as the HTTP attribute. The resulting format from converting the response model into HTTP is shown in Listing 4.

```
{
    "state-value" : 0,
    "text-section" : "string",
    "title" : "string",
    "date-time" : "string",
    "center-position" : { "lat" : 0, "lng" : 0 },
    "zoom" : 0,
    "locations" : [
        {
            "title" : "string",
            "description" : "string",
            "position" : { "lat" : 0, "lng" : 0 },
            "isHighlighted" : true,
            "imageUrl" : "string",
            "path" : [
                {"lat": 0, "lng": 0}
            ],
            "marker-color" : "string",
            "path-color" : "string",
            "path-thickness" : "string",
            "direction" : { "lat" : 0, "lng" : 0 }
        }
    ]
}
```

Listing 4: Response model in Javascript Object Notation

4.2.2 Database Model

The purpose of the DB Model is to store information for multiple storylines, i.e, multiple books. A SQL DB is used as the underlying data source provider, and Entity Framework (EF) as an Object-Relational Mapping (ORM). EF will map object oriented classes into DB entities. The object oriented classes used to define the DB are listed in Listing 5. The resulting DB created by EF is illustrated as an Entity Relationship (ER) Diagram shown in Figure 4.8.

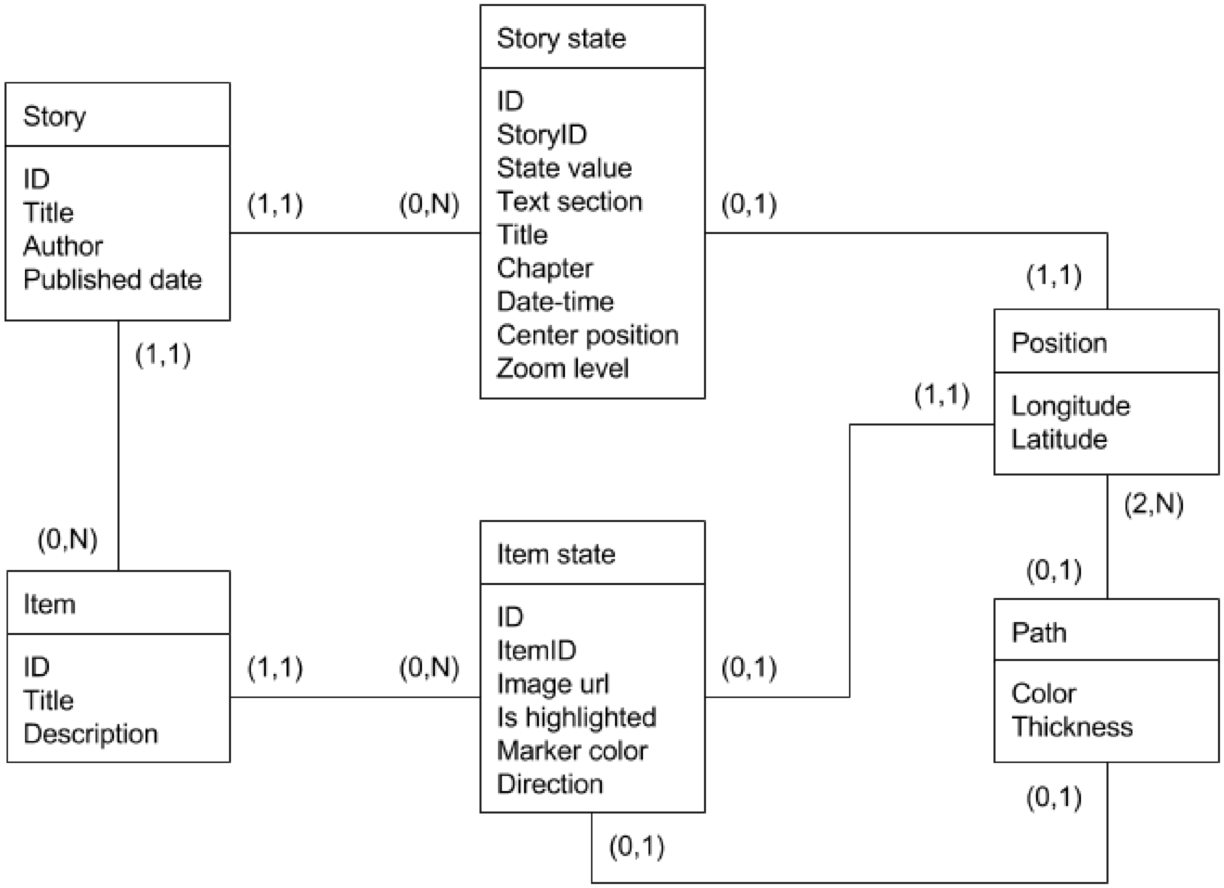


Figure 4.8: ER Diagram for Database Model.

```
public class Position{
    public double Lat { get; set; }
    public double Lng { get; set; }
}
public class Path{
    public List<Position> Positions { get; set; }
    public string Color { get; set; }
    public string Thickness { get; set; }
}
public class Item{
    public int ID { get; set; }
    public string Title { get; set; }
    public string Description { get; set; }
    public Dictionary<int, ItemState> ItemStates { get; set; }
}
public class ItemState{
    public int ID { get; set; }
    public int ItemID { get; set; }
    public Position Position { get; set; }
    public Path Path { get; set; }
    public string ImageUrl { get; set; }
    public bool IsHighligted { get; set; }
    public string MarkerColor { get; set; }
    public Position Direction { get; set; }
}
public class StoryState{
    public int ID { get; set; }
    public int StoryID { get; set; }
    public int StateValue { get; set; }
    public string TextSection { get; set; }
    public string Title { get; set; }
    public int Chapter { get; set; }
    public DateTime DateTime { get; set; }
    public Position CenterPosition { get; set; }
    public int Zoom { get; set; }
}
public class Story{
    public int ID { get; set; }
    public string Title { get; set; }
    public string Author { get; set; }
    public DateTime PublishedDate { get; set; }
    public List<StoryState> StoryStates { get; set; }
    public List<Item> Items { get; set; }
}
```

Listing 5: Object oriented classes used by EF to build the Database Model.

The DB holds the data for each "state" object in a literary storyline. It is, however, dispersed into multiple DB tables. A DB table is created for each entity in Figure 4.8. This is done to prevent duplicate storing of data.

The databae queries would have been simple if the "states", i.e., response models were defined as databse tables, since "states" could be retrieved based on their identifier. However, the DB would contain a lot of dublicates. Each state in the storyline may contain the same location, e.g., "london". A single location should only be stored once in the DB.

The following entities define the DB.

Story

The “Story” entity is the root entity of the data model, and contains the complete list of items that is present in the storyline. These items are ,e.g., ships, infantry, vehicles, etc. The “Story” entity also contains a list of “Story state” entities. "Story state" entities holds, amongst other data, text sections that constitutes the storyline. Lastly, the "Story" entity contains general information about the storyline, including a title, when it was published and the name of the author.

Story state

The “Story state” entity is the root entity for text-, map- and timeline information for a single “state” object. The “Story state” entity contains a single text section from the storyline and the name of the chapter it belongs to. The chapter property will be used to group data when transmitting information to the client side. It also hold a title that will be displayed in the Title-view of the UI and a date-time value that will be displayed in the Timeline-view. The title will hold a short description of what the map is displaying. The “Story state” entity has a center position and a zoom property used to adjust the map outline to the location markers visible for this "state". Lastly is the “state value” identifier itself, used as a reference.

Item

The “Item” entity represent an item that is included in the storyline. This could ,e.g, be a boat or a city. The "Item" entity stores a title and description. Items has a tendency to appear on multiple geographical locations throughout the storyline, therefore, data relative to different "states" are kept separated from the items general data. Map and timeline data for different "states" are kept in "Item state" entities, stored in a Dictionary where the key is the “state value” identifiers stored in the "Story state" entity.

ItemState

The “Item state” entity holds the information needed to represent an item in a specific "state". The item will not be represented in the "state" if the "state value" is missing from the "Item" entity’s dictionary, i.e., if the "Item state" entity is not provided for a given "state value". The "Item state" entity contains position, path, image and marker color for a single item. Position and path is represented by the "Position" and "Path" entities respectively. The "Item" entity also defines whether or not the item is highlighted in the given "state" in the map. Highlighted items will have its title visible next to the location marker in the map when the map is initially drawn.

Position

The “Position” entity holds a pair of longitude-latitude values, i.e., coordinates. It is used by the “Story state” entity to define a center position for the map outline and by the “Item state” entity to define a position for an item in a given "state". The latter is represented as a marker in the map.

Path

The “Path” entity holds a list of "Position" entities and the paths thickness and color. A path is represented by multiple latitude-longitude pairs which is connected by straight lines.

4.2.3 Database Query

DB queries will be executed when the client side requests data from the server side. A request is sent with a story and chapter-identifier as parameters, requesting every "state" object defined within the given chapter of the story. The request will be sent to a web server function via HTTP. This function executes the query. A DB query is then needed to collect the correct data from the DB. The result of the query will be placed inside a list of response models, i.e., "states", that is readable for the client side and returned. The Client Server communication is illustrated if Figure 4.9.

DB

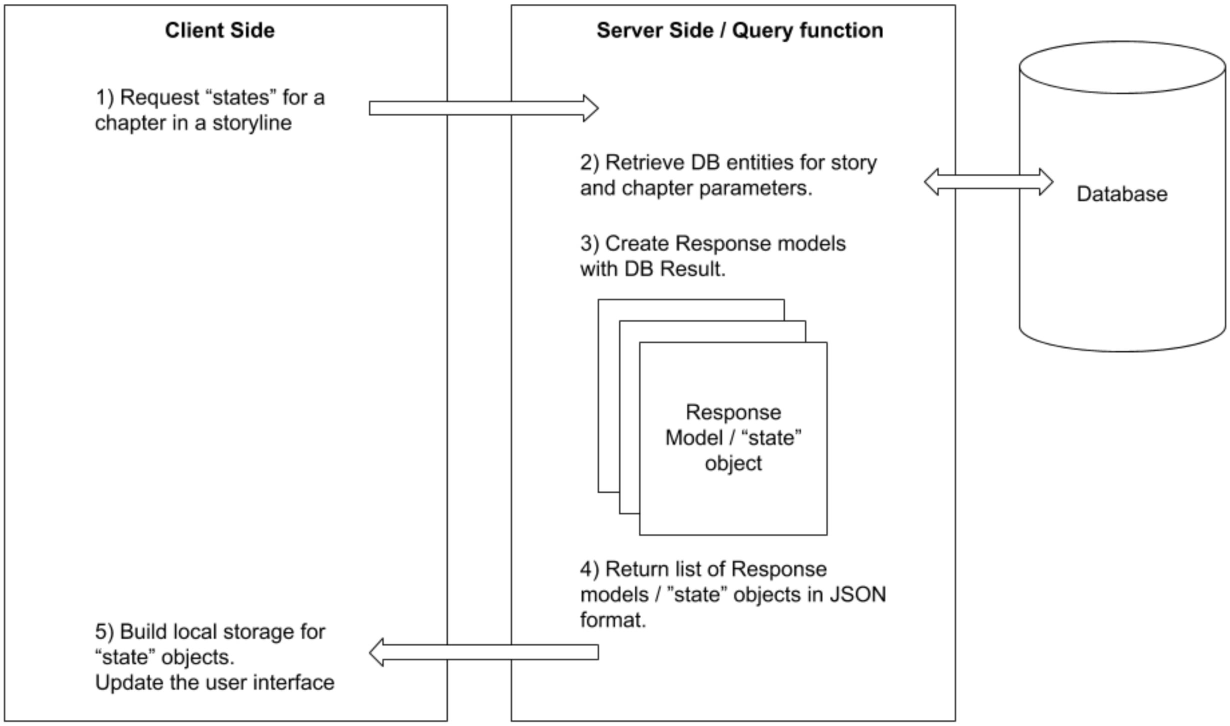

Client Side
1) Request “states” for a chapter in a storyline
5) Build local storage for “state” objects.
Update the user interface
Server Side / Query function
2) Retrieve DB entities for story and chapter parameters.
3) Create Response models with DB Result.
Response Model / “state” object
4) Return list of Response models / ”state” objects in JSON format.
Database

Figure 4.9: Database Model

A DB query for a given chapter collects data in the following way. The "Story" root entity will be retrieved from the DB based on the story identifier provided by the request. A list of "Story state" entities defined within the requested chapter is created by selecting the "Story state" entities from the "Story" entity where their chapter value matches the chapter parameter in the request. Blank response models, i.e., "state" objects, are created for each "Story state" in the list. A list of "location" object" created for each respective "Story state". Items are selected from the "Story" root entity. Items are included if they have an "Item state" entity present in the dictionary for the respective "state value". A "location" object is created for each "Item" in the list. The location object is populated with general data from the "Item" entity and geographical data from the "Item state" entity. The list of "location" objects are placed inside the blank response model for its respective "Story state". The response model is further populated with data, such as text-section and date-time, from the "Story state" entity. The list of response models are converted into JSON format and returned to the client side.

4.2.4 Database Population

The web service should have an administration subpage that facilitates creating and editing storylines, i.e., a tool that helps authors create data for a given storyline that eventually will populate the DB. However, due to the limitations of this thesis, this feature is deferred in favor of having a functional prototype for the readers perspective. The DB will, at this stage, be populated by directly seeding objects into it. This is done via a seed method provided by EF. The seed method allows developers to write object oriented data models based on the DB entities provided in Section 4.2.2. Data supplied to the seed method will be inserted in the DB when the web server runs. This is not an acceptable solution for a commercialized service, however, it will suffice for demonstration purposes.

Chapter 5

Case Study

This chapter presents a case implemented in the prototype.

5.1 Background

The case used in this Chapter is a summarized presentation of the German attack on Oslo, Norway in World War 2. It features the participation of the heavy German cruiser Blücher. Blücher was leading a convoy of one heavy cruiser Lützow, one light cruiser Emden and several smaller escorts into the Oslofjord on the night of 8. April 1940 to seize Oslo, the capital of Norway. This story has been used because it includes multiple parties and a variety of events taking place in different geographical locations. The textual storyline has been included in Appendix A.1. The case has been used as the foundation for user feedback in Chapter 6.

The case should ideally have been a full novel as opposed to a summary of events. However, a summary should be sufficient for demonstration purposes. It also makes testing more feasible considering duration and effort, as the test subjects does not have to read a full novel before giving feedback. However, a longer text might have provided a better basis for an evaluation and might have introduced new and unexpected aspects.

The service should be as general as possible to represent any literature, i.e., books, in the context of a map and a timeline. Strengths, weaknesses and limitations may be revealed as more use cases are implemented and tested. The proof of concept may be optimal for some cases, and less optimal for other. Therefore, the requirements are expected to change as different cases are implemented and tested.

5.2 Results

The textual content for the case has been collected from multiple sources, however, mainly from the short summary *«Blücher»* [17] by Wikipedia. The text presents the main events of Blücher's whereabouts and actions in low detail. For example, it states that the Norwegian coast guard vessel Pol III was able to alert the Norwegian Army about Blücher's presence in the Oslofjord. It is hard to create specific map markers, path lines and date-times for the map and timeline based on this statement. In cases like this, additional, and more detailed, content has been obtained from other sources [18] [19] [20] [21] [3] [10] and pasted into the main summary.

The text has been separated into multiple text sections, where each text section encloses a single event that is considered reasonable to represent in the Map-view and Timeline-view. For example, the paragraphs that presents Blücher's encounter with the British sumbarine HMS Triton (N15) has been defined as one text section.

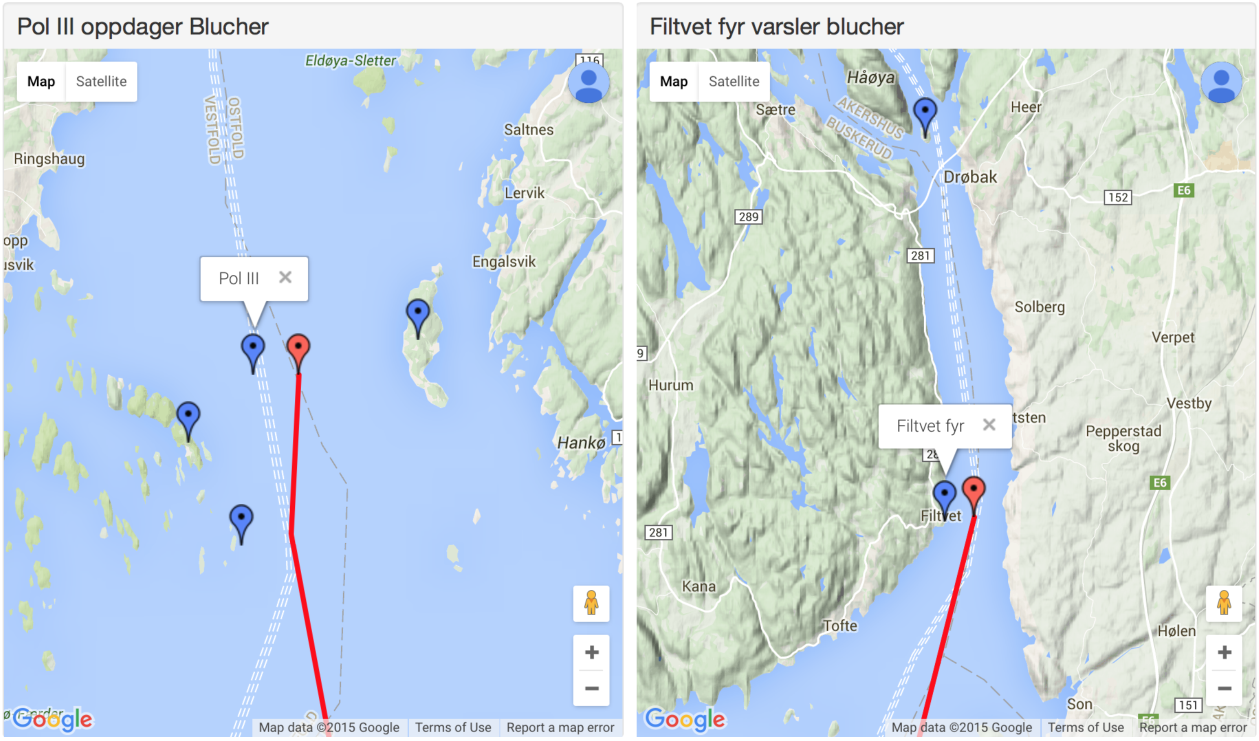
"Story state"-, "Item"-, "Item state"-, "Position"- and "Path"- entities are created for each text section. These entities are added to the DB, and constitute all the map-, timeline- and textual data in the prototype. For example, Blücher is represented as a single "Item", with one "Item state" for each position along its journey.

The map and timeline data for each "state" needs to be created by the author of the case, as this is not obtainable in the same way as the text. Map and timeline data has been created based on the textual information. The summary lists where and when some of the events took place. This was in some cases very detailed, and in other cases open for interpretation. Additional sources are used to find more accurate data for geographical locations and time of events. Presumptions has been made where the required information is missing or low in details. The text does, e.g., not describe Blücher's travel path between Swinemünde and Skagen, i.e., if Blücher traveled east or west of Copenhagen in order to arrive at Skagen. A presumption has, therefore, been made in this case.

The storyline features German, Norwegian and British military units, including submarines, ships, fortresses and airplanes. These units, i.e., items, are represented as markers in the Map-view. Each party has been assigned with their own marker color. Blue represents Norwegian entities, green represents British entities and red represents German entities. Paths has been colored the same way as their respective markers. Paths are expanded as the storyline progresses, e.g., Blücher will have a path starting in its first location, "Swinemünde", connected by its intermediary locations as it gets closer to Oslo. Stapled black lines are used to illustrate lines of fire, e.g., when the Norwegian fortress, Oscarsborg, opens fire on Blücher. This data were placed inside the "direction" value in the "state" objects.

A different zoom and center position is provided for each "state" object, i.e., for each map representation. The combination of these values are set considering two aspects. First, the values should not differ too much from its previous values, etc. The reader should be able to recognize where the previous events took place based on the new map outline. The reader is likely to be confused as for where the new map outline has been defined if it differs too much from the previous. An example of an unclear shift in map outline is shown in Figure 5.1. Another example is shown in Figure 5.2. It clearly displays where Blücher has travelled since the last "state". Second, the outline should be detailed enough to separate markers that appears close to each other. An example outline with a low zoom level is shown in Figure 5.3. In this case, the markers almost appears to be in the same location. It is hard to see where they are positioned relative to each other. Another example is shown in Figure 5.4. It features the same center position as in Figure 5.3 with the exception of a higher zoom level. This map outline makes it easier to separate the involved markers.

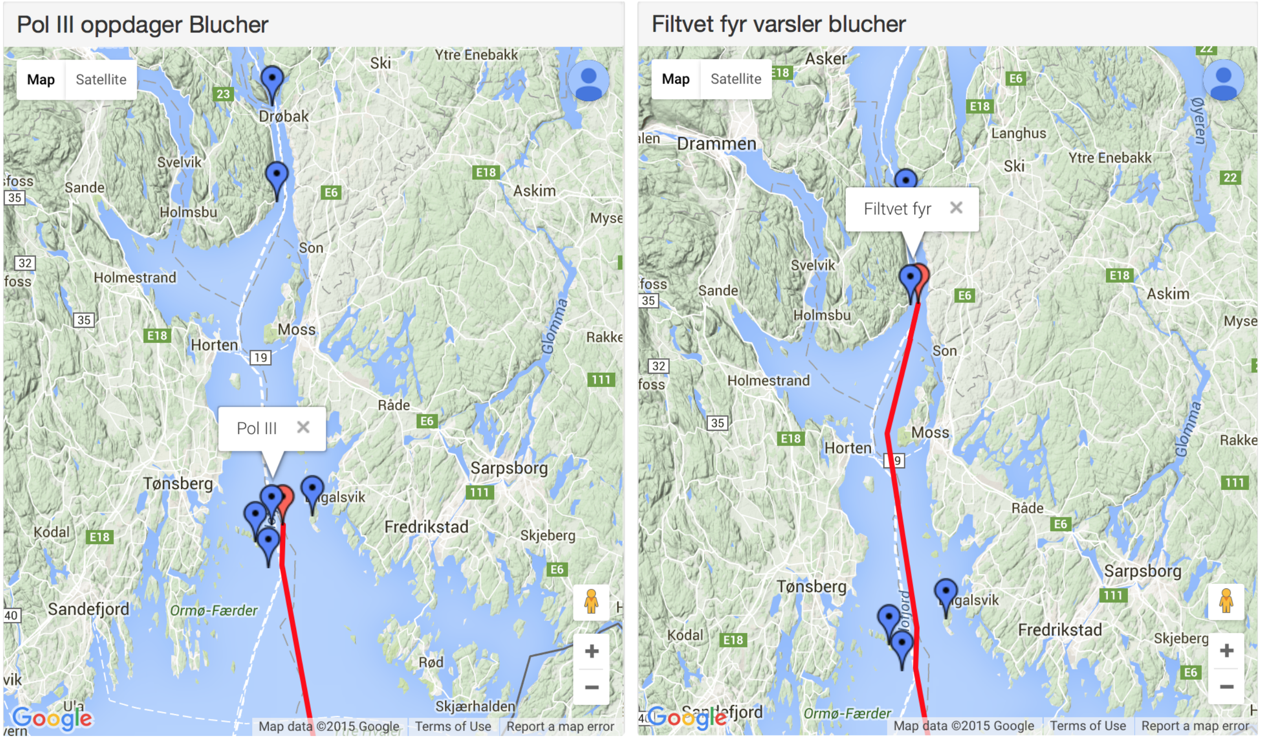
Parts of the resulting case implementation is show in Figures 5.5, 5.6, 5.7 and 5.8.



Pol III oppdager Blucher

Filtvet fyr varsler blucher

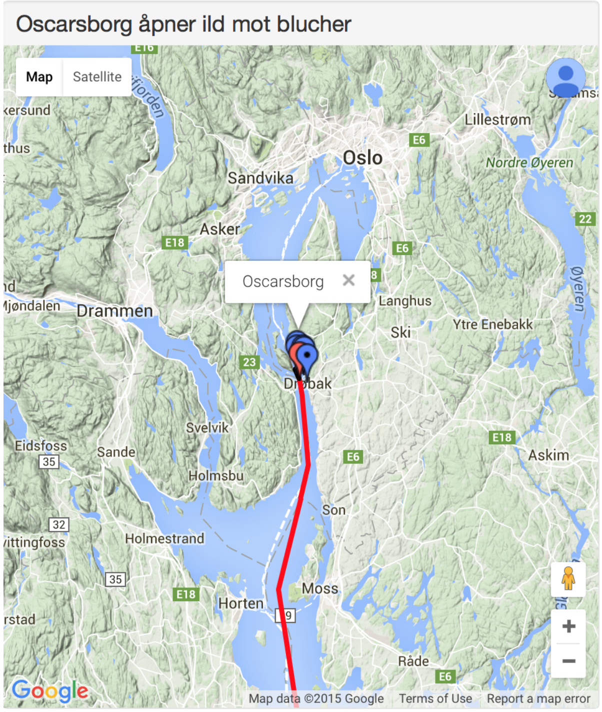
Figure 5.1: Possible unclear transition between two map oulines.



Pol III oppdager Blucher

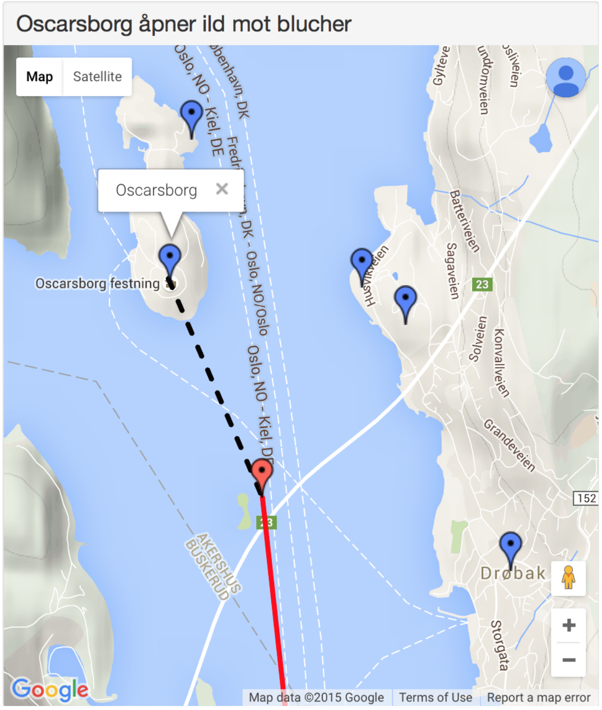
Filtvet fyr varsler blucher

Figure 5.2: Possible clear transition between two map oulines.



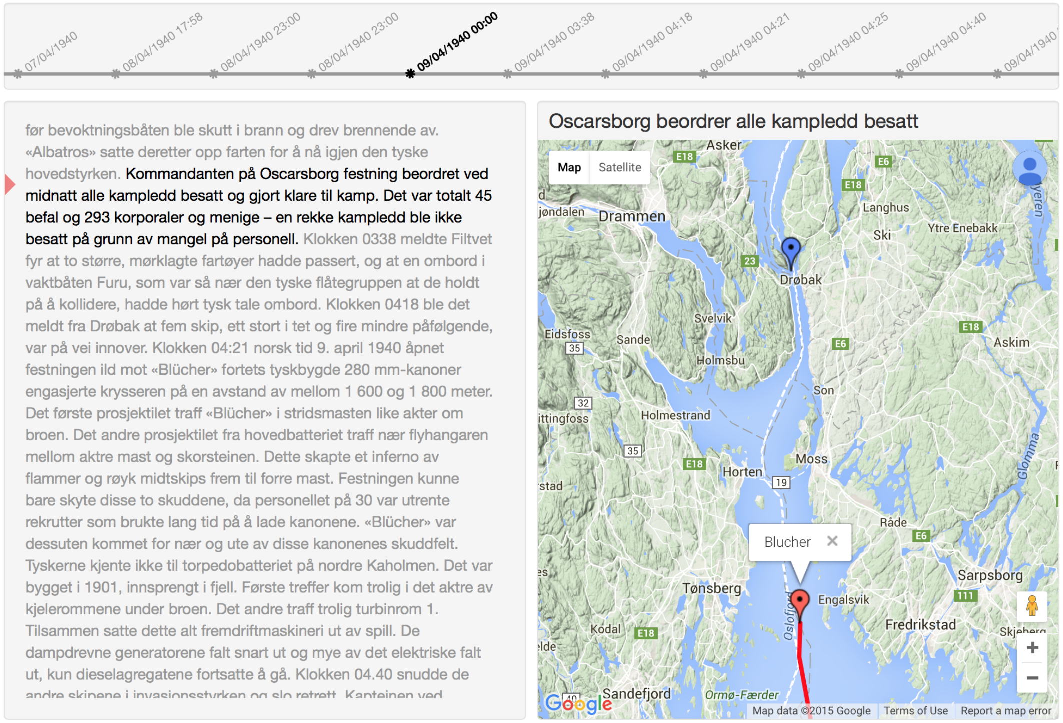
Oscarsborg åpner ild mot blucher

Figure 5.3: Map outline with low zoom level.



Oscarsborg åpner ild mot blucher

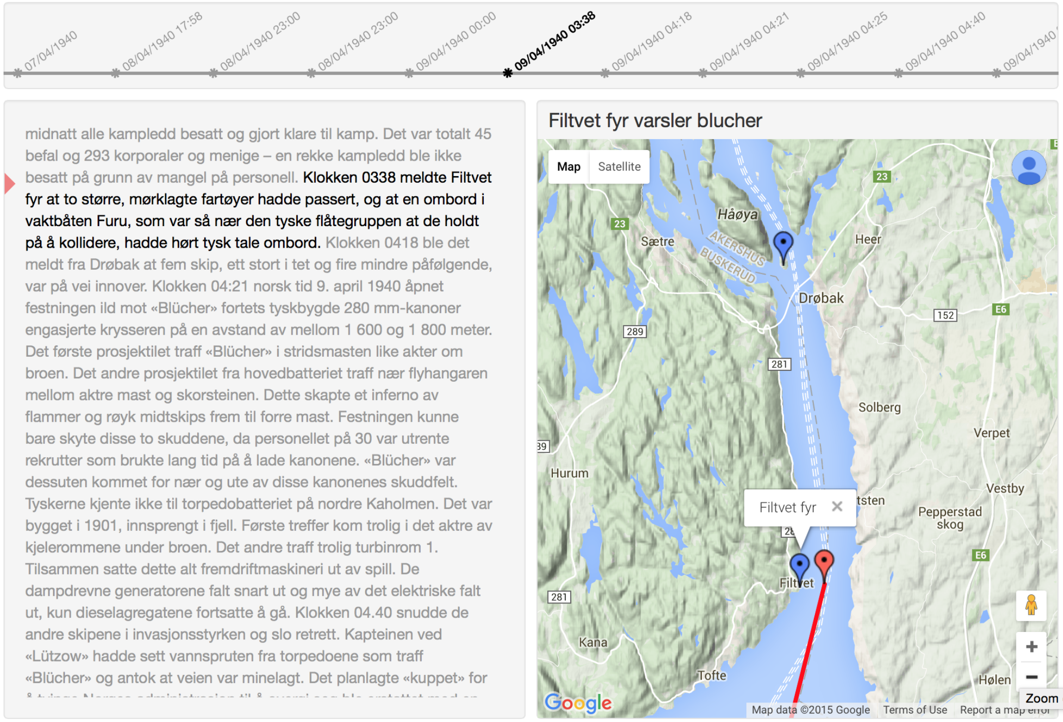
Figure 5.4: Map outline with high zoom level.



Oscarsborg beordrer alle kampledd besatt

før bevoktningsbåten ble skutt i brann og drev brennende av. «Albatros» satte deretter opp farten for å nå igjen den tyske hovedstyrken. Kommandanten på Oscarsborg festning beordret ved midnatt alle kampledd besatt og gjort klare til kamp. Det var totalt 45 befal og 293 korporaler og menige – en rekke kampledd ble ikke besatt på grunn av mangel på personell. Klokken 0338 meldte Filtvet fyr at to større, mørklagte fartøyer hadde passert, og at en ombord i vaktbåten Furu, som var så nær den tyske flåtegruppen at de holdt på å kollidere, hadde hørt tysk tale ombord. Klokken 0418 ble det meldt fra Drøbak at fem skip, ett stort i tet og fire mindre påfølgende, var på vei innover. Klokken 04:21 norsk tid 9. april 1940 åpnet festningen ild mot «Blücher» fortets tyskbygde 280 mm-kanoner engasjerte krysseren på en avstand av mellom 1 600 og 1 800 meter. Det første prosjektilet traff «Blücher» i stridsmasten like akter om broen. Det andre prosjektilet fra hovedbatteriet traff nær flyhangaren mellom aktre mast og skorsteinen. Dette skapte et inferno av flammer og røyk midtskips frem til forre mast. Festningen kunne bare skyte disse to skuddene, da personellet på 30 var utrente rekrutter som brukte lang tid på å lade kanonene. «Blücher» var dessuten kommet for nær og ute av disse kanonenes skuddfelt. Tyskerne kjente ikke til torpedobatteriet på nordre Kaholmen. Det var bygget i 1901, innsprengt i fjell. Første treffer kom trolig i det aktre av kjelerommene under broen. Det andre traff trolig turbinrom 1. Tilsammen satte dette alt fremdriftmaskineri ut av spill. De dampdrevne generatorene falt snart ut og mye av det elektriske falt ut, kun dieselagregatene fortsatte å gå. Klokken 04.40 snudde de andre skipene i invasjonsstyrken og slo retrett. Kapteinen ved

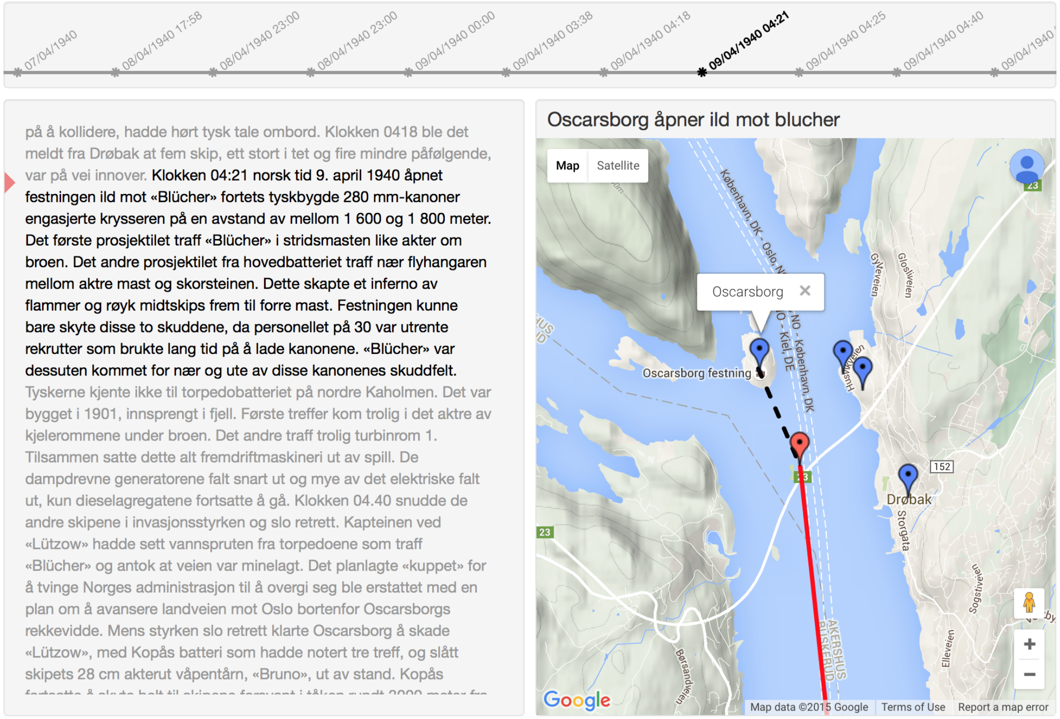
Figure 5.5: Screenshot of resulting case implementation, #1.



Filtvet fyr varsler blucher

midnatt alle kampledd besatt og gjort klare til kamp. Det var totalt 45 befal og 293 korporaler og menige – en rekke kampledd ble ikke besatt på grunn av mangel på personell. Klokken 0338 meldte Filtvet fyr at to større, mørklagte fartøyer hadde passert, og at en ombord i vaktbåten Furu, som var så nær den tyske flåtegruppen at de holdt på å kollidere, hadde hørt tysk tale ombord. Klokken 0418 ble det meldt fra Drøbak at fem skip, ett stort i tet og fire mindre påfølgende, var på vei innover. Klokken 04:21 norsk tid 9. april 1940 åpnet festningen ild mot «Blücher» fortets tyskbygde 280 mm-kanoner engasjerte krysseren på en avstand av mellom 1 600 og 1 800 meter. Det første prosjektilet traff «Blücher» i stridsmasten like akter om broen. Det andre prosjektilet fra hovedbatteriet traff nær flyhangaren mellom aktre mast og skorsteinen. Dette skapte et inferno av flammer og røyk midtskips frem til forre mast. Festningen kunne bare skyte disse to skuddene, da personellet på 30 var utrente rekrutter som brukte lang tid på å lade kanonene. «Blücher» var dessuten kommet for nær og ute av disse kanonenes skuddfelt. Tyskerne kjente ikke til torpedobatteriet på nordre Kaholmen. Det var bygget i 1901, innsprengt i fjell. Første treffer kom trolig i det aktre av kjelerommene under broen. Det andre traff trolig turbinrom 1. Tilsammen satte dette alt fremdriftmaskineri ut av spill. De dampdrevne generatorene falt snart ut og mye av det elektriske falt ut, kun dieselagregatene fortsatte å gå. Klokken 04.40 snudde de andre skipene i invasjonsstyrken og slo retrett. Kapteinen ved «Lützow» hadde sett vannspruten fra torpedoene som traff «Blücher» og antok at veien var minelagt. Det planlagte «kuppet» for å tvinge Norges administrasjon til å overgi seg ble erstattet med en

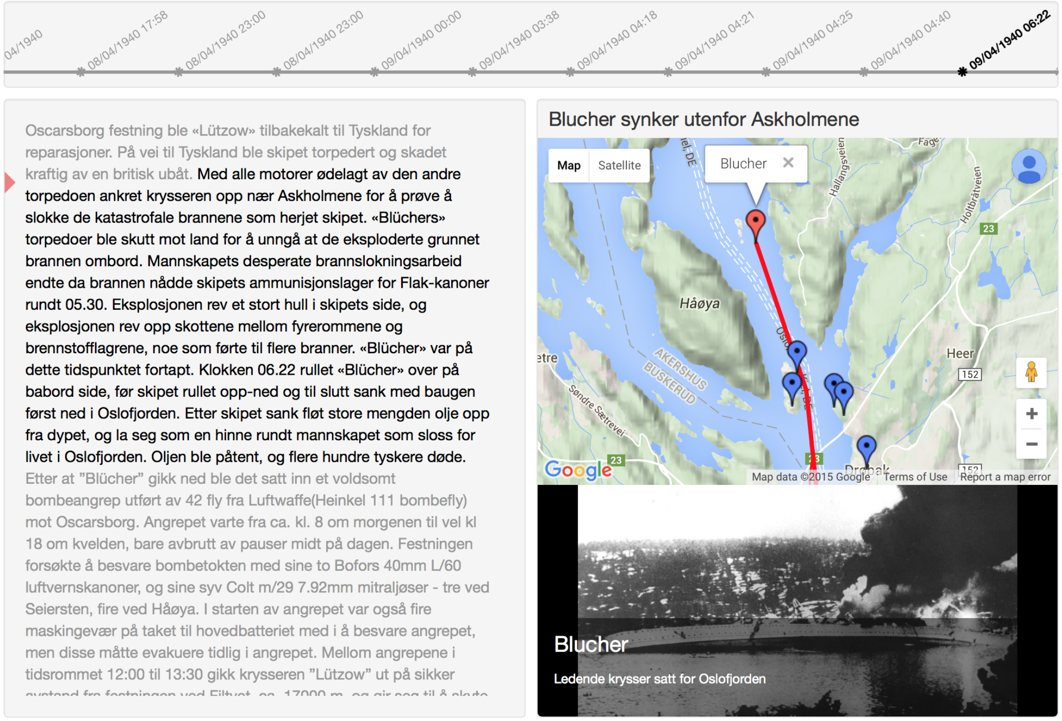
Figure 5.6: Screenshot of resulting case implementation, #2.



på å kollidere, hadde hørt tysk tale ombord. Klokken 0418 ble det meldt fra Drøbak at fem skip, ett stort i tet og fire mindre påfølgende, var på vei innover. Klokken 04:21 norsk tid 9. april 1940 åpnet festningen ild mot «Blücher» fortets tyskbygde 280 mm-kanoner engasjerte krysseren på en avstand av mellom 1 600 og 1 800 meter. Det første prosjektilet traff «Blücher» i stridsmasten like akter om broen. Det andre prosjektilet fra hovedbatteriet traff nær flyhangaren mellom aktre mast og skorsteinen. Dette skapte et inferno av flammer og røyk midtskips frem til forre mast. Festningen kunne bare skyte disse to skuddene, da personellet på 30 var utrente rekrutter som brukte lang tid på å lade kanonene. «Blücher» var dessuten kommet for nær og ute av disse kanonenes skuddfelt. Tyskerne kjente ikke til torpedobatteriet på nordre Kaholmen. Det var bygget i 1901, innsprengt i fjell. Første treffer kom trolig i det aktre av kjelerommene under broen. Det andre traff trolig turbinrom 1. Tilsammen satte dette alt fremdriftmaskineri ut av spill. De dampdrevne generatorene falt snart ut og mye av det elektriske falt ut, kun dieselagregatene fortsatte å gå. Klokken 04.40 snudde de andre skipene i invasjonsstyrken og slo retrett. Kapteinen ved «Lützow» hadde sett vannspruten fra torpedoene som traff «Blücher» og antok at veien var minelagt. Det planlagte «kuppet» for å tvinge Norges administrasjon til å overgi seg ble erstattet med en plan om å avansere landveien mot Oslo bortenfor Oscarsborgs rekkevidde. Mens styrken slo retrett klarte Oscarsborg å skade «Lützow», med Kopås batteri som hadde notert tre treff, og slått skipets 28 cm akterut våpentårn, «Bruno», ut av stand. Kopås fortsatte å skyte helt til skipene forsvant i tåken rundt 2000 meter fra

Oscarsborg åpner ild mot blucher

Figure 5.7: Screenshot of resulting case implementation, #3.



Oscarsborg festning ble «Lützow» tilbakekalt til Tyskland for reparasjoner. På vei til Tyskland ble skipet torpedert og skadet kraftig av en britisk ubåt. Med alle motorer ødelagt av den andre torpedoen ankret krysseren opp nær Askholmene for å prøve å slokke de katastrofale brannene som herjet skipet. «Blüchers» torpedoer ble skutt mot land for å unngå at de eksploderte grunnet brannen ombord. Mannskapets desperate brannslokningsarbeid endte da brannen nådde skipets ammunisjonslager for Flak-kanoner rundt 05.30. Eksplosjonen rev et stort hull i skipets side, og eksplosjonen rev opp skottene mellom fyrerommene og brennstofflagrene, noe som førte til flere branner. «Blücher» var på dette tidspunktet fortapt. Klokken 06.22 rullet «Blücher» over på babord side, før skipet rullet opp-ned og til slutt sank med baugen først ned i Oslofjorden. Etter skipet sank fløt store mengden olje opp fra dypet, og la seg som en hinne rundt mannskapet som sloss for livet i Oslofjorden. Oljen ble påtent, og flere hundre tyskere døde. Etter at ”Blücher” gikk ned ble det satt inn et voldsomt bombeangrep utført av 42 fly fra Luftwaffe(Heinkel 111 bombefly) mot Oscarsborg. Angrepet varte fra ca. kl. 8 om morgenen til vel kl 18 om kvelden, bare avbrutt av pauser midt på dagen. Festningen forsøkte å besvare bombetokten med sine to Bofors 40mm L/60 luftvernskanoner, og sine syv Colt m/29 7.92mm mitraljøser - tre ved Seiersten, fire ved Håøya. I starten av angrepet var også fire maskingevær på taket til hovedbatteriet med i å besvare angrepet, men disse måtte evakuere tidlig i angrepet. Mellom angrepene i tidsrommet 12:00 til 13:30 gikk krysseren ”Lützow” ut på sikker avstand fra festningen ved Filtvet, ca. 17000 m, og gir seg til å skyte

Blucher synker utenfor Askholmene

Blucher

Ledende krysser satt for Oslofjorden

Figure 5.8: Screenshot of resulting case implementation, #4.

5.3 Discussion

Collecting data needed to have each text section represented in the map proved to be a comprehensive task, as the storyline discusses geographical locations in low detail. It is easy to assign markers to cities, but, e.g., more challenging when events unfold in the ocean. Some archives offered detailed information of events with specific coordinates. Finding the specific information was, however, a time consuming task.

Adding paths is easy when the involved markers are already defined, e.g., the sequence of previous locations for Blücher is included in its path as it travels.

It proved to be challenging to define map outlines for each "state", i.e., each map representation. The different map outlines were corrected multiple times during the implementation.

At first, the entire convoy of warships lead by Blücher was represented as individual markers. These were in most cases altered to one single marker because they tended to appear on top of each other. It is hard to single out and click on markers when they appear on roughly the same location.

It would have be beneficial to, e.g., represent Blücher as a photo of a ship instead of a red marker. This alternative was explored. The map framework used in the prototype, Google Maps, allows for custom markers in the form of photos, but it does not support scaling and rotating of the photos. A new photo is therefore needed for every "state" in order to represent Blücher in correct proportion and direction. This feature were therefore ignored for now.

Extracting dates and times from the storyline were in this case easy compared to extracting map relevant data, because they are precisely listed.

Creating object oriented classes for each "Story state", "Item", "Item state", "Position" and "Path" as well as linking them together was time consuming and challenging. A UI to handle creating and editing "state" object will presumably save time and make the task feasible for authors without background in software engineering. The current solution for populating the DB will not be acceptable in a commercialized version of the prototype.

Implementing the case was overall successful. The prototype is able to represent the different entities involved in the storyline with, as intended, varying level of detail. However, the prototype can not be finalized based on this case alone. Other cases might require new features in order to be properly represented. A single case is not considered as sufficient in order to reveal all the limitations and challenges with the current prototype.

Chapter 6

Testing

This chapter presents feedback from a user test of the prototype, and conclusion of the feedback.

6.1 Background

User testing has been used as an approach to validate the prototype. A user test should help reveal what potential users like and dislike about the prototype, which element of the UI they focus on, and their thoughts of the general usability. The reader's general impression of the prototype is valuable in terms of evaluation and further development. People read in different ways and have different opinions for desired functionality. A part of the prototype might be rewarding for some readers, while others find it hard to use.

The following questions concern the thesis' objectives, and has been the foundation for user testing. Will the prototype enrich the reading experience? In other words, will the addition of supplementary data give the reader a more accurate perception of literary plots? Which supplementary elements do the readers prefer?

It it considered reasonable to conduct a series of short and informal user tests during prototyping, because the prototype is in constant change. A systematic and thorough test is more viable when the prototype is close to being finished. This is concluded in the article *Why You Only Need to Test with 5 Users* [11] by Jakob Nielsen. The article also states that five test participants is enough to reveal any main problems or concerns of a prototype. Figure 6.1 indicates that five test participants are able to uncover approximately 85% of the problems. This percentage is acceptable given that the prototype is at an early stage of development.

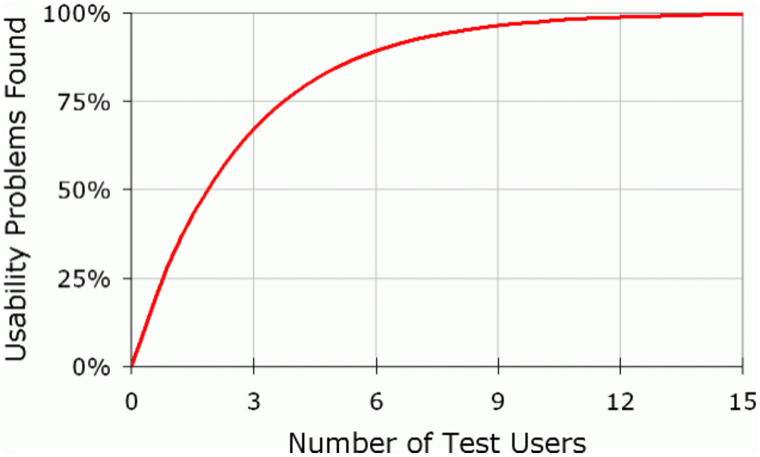


Figure 6.1: Percentage of problems found compared to number of test users, according to the article "Why You Only Need to Test with 5 Users" *[11] by Jakob Nielsen.*

The prototype is intended to be used by readers of all ages. The test participants were therefore selected with varying age and level of computer experience, as they are likely to have different likings or concerns. The test participants should not have been involved in developing the prototype, as developers are likely to overlook problems and concerns when testing their own softwares.

The test scenario consisted of presenting each test participant with the case presented in Chapter 5. They were given a short introduction to both the case storyline and the prototype's functionalities. The test participants were told to read and interact with the prototype before answering a few informal questions about their user experience, which part of the UI they focused on and if the prototype provided precise geographical information. An informal approach has been used because the prototype is at an early stage of development, and the test is intended to reveal main issues, likings and concerns. Minor issues and adjustment are not important at this stage of development.

6.2 Results

This section presents the feedback received from five test participants.

Participant 1

Test participant number one is a 48 year old female accountant. She had good knowledge of the storyline prior to using the prototype. However, she claimed to have gotten more precise knowledge about the involved parties' whereabouts after using the prototype. For instance, she did not know where in the North Sea the British submarine HMS Triton (N15) fired torpedoes towards Blücher until she saw it in the map.

It took her just a couple of scrolls to understand how the highlighted text collaborated with the map and timeline. The title above the map was not noticed in the very beginning, but she thought it was a nice asset to summarize what the map was presenting. Lastly, the test participant was annoyed by the copyright line at the bottom of the map because it partially overlapped with a location marker in one of the "states". She did, however, understand why it is present.

She thought testing the prototype was a rewarding experience. She is definitely interested in reading literature this way if something similar is commercialized.

Participant 2

Test participant number two is a 36 year old male sales representative. His knowledge of the storyline was limited to a brief summary prior to using the prototype. He found the complementary map to be supportive in terms of reading experience. He stated that he got a more precise understanding of where and when the storyline plot unfolded after using the prototype. He also learned some new things about the historical storyline, such as the involvements of "Pol III", a coast guard vessel in the Norwegian navy that patrolled between Bolærne and Rauøy as Blücher and the rest of the convoy made its way up the Oslofjord.

The test participant really liked the simple and minimalistic UI. He mentioned that the text did not suffer from the supplementary elements, i.e., the map and timeline. The test participant found every aspect of the prototype to be self-explanatory, but he did, however, not pay attention to the timeline and title. He only focused on the text and map.

He recommended that the focal point in the Text-view should be personalized to match individual reading patterns, as people read and interact with text in different ways. Some people like to keep the text section they are reading in top of the page, others like to keep it at the bottom, and so on. The test participant's reading pattern did not completely match the specified focal point. This resulted in early change of text highlighting and map update before he had read the current text section. However, he quickly altered his way of reading to eliminate this problem.

He was generally positive about using the prototype, but concerned about reading on computers. He would ideally like to have the prototype optimized for tablets, as he is used to reading e-books this way. He presumed that it would be hard to present all the UI elements provided by the prototype on a narrow tablet screen. He suggested that the timeline and map could be toggled between a visible and hidden state, allowing the text, timeline and map to fill the entire screen interchangeably.

Participant 3

Test participant number three is a 26 year old male electrical engineer. He had no knowledge of the storyline prior to using the prototype. He commented right away that the text is missing paragraph offsets, as this is not included for the text used in the case. He, however, felt that text highlighting made up for the lack of paragraph offsets.

He thought the concept of combining literature with maps, timelines and photos was a clever idea. He did not question how to interact with the UI as he felt that the different elements were self-explanatory. He did, however, have some trouble understanding where the map outline was defined for the different "states", especially when the zoom level was high. In these cases, he adjusted to a lower zoom level as well as toggling back and forth between the current and previous "state" trying to understand where the current map outline was defined. He suggested that map outlines should be defined with a low zoom level for the case.

Lastly, the test participant concluded that the prototype provided an interesting and informative way of reading text, due to the timeline, title, map and photos. The supplementary data definitely helped him get a good understanding of the storyline, even though he occasionally struggled with the map outlines.

Participant 4

Test participant number four is a 20 year old female psychology student. She had only heard about Blücher prior to using the prototype, but did not know anything about the storyline. She expected the map to be static in the very beginning of using the prototype, leaving her confused when she was presented with a map that outlines Swinemunde, the Polish harbor where Blücher started its journey. However, she quickly realised that the map displayed different data as she progressed through the text. She was very pleased about how the map and timeline always complemented highlighted text once she understood how the system works.

The test participant was annoyed because the text highlighting shifted long after she had finished reading text sections. She prefers reading from the very bottom of the page. This caused her to stay ahead of the "state" being displayed in the map, title and timeline. She eventually started reading from the top of the page, i.e., from the highlighted text, in order to keep text, title, map and timeline coherent. The new reading pattern did not bother her too much toward the end of her reading session. The test participant tried to use the arrow keys on the computer keyboard in order to skip to the next text section, etc. She was annoyed that this feature was missing.

She found the reading experience to be joyful, even though her reading pattern contrasted the specified focal point in the text. She stated that the prototype had provided her with an easily understandable representation of the storyline, and that she often remembered visual information better than text alone. She would like to read other literary storylines via the prototype, given that the text highlighting can occur at the bottom of the page, complementing her reading pattern.

Participant 5

Test participant number five is a 24 year old male computer engineer. He was well familiarized with the storyline of Blücher prior to using the prototype, and stated that he had pretty accurate knowledge of where and when different events unfolded. He did not need any guidance in order to use the prototype, as he thought the UI was self-explanatory. He, however, did not pay much attention to the title and timeline. He felt that the timeline, especially, was unnecessary. He suggested a couple of features that he felt would improve the timeline. Primarily, he suggested that clicks on the timeline dots, i.e., the date-time values, should make the prototype skip to the corresponding "state". This includes updating the map and highlighting the text to correspond to the clicked dot.

Secondly, he suggested that the highlighted dot in the timeline should always be kept centered. This would allow the reader to see dots listed before and after the currently highlighted dot. The dots in the timeline are listed ascending from left to right. The dots have a fixed distance between each other. The timeline can therefore only display a subset of dots at the time if the number exceeds the length of the timeline. The overflowing dots will initially appear out of bounce on the rightmost side, and are therefore hidden. The dots are shifted one step to the left when the highlighting reaches one step beyond the rightmost dot, i.e., when the highlighting goes out of bounce. From this point and out, the highlighting will stick to the rightmost visible dot. This does not allow the reader to view dots appearing after the highlighted dot.

6.3 Discussion

The user test indicated that the prototype was easy to use and self-explanatory. The test participants were able to utilize every aspect of the UI without guidance. They understood how the map, timeline and title collaborated with the literary text, and how the reading pattern worked. The test participants stated that the supplementary elements, i.e., the Map-view and Timeline-view gave a rewarding reading experience, which lead to precise knowledge of where and when the storyline plot unfolded. Some test participants paused reading after updates, i.e., skips between "states", in order to explore the map outline, map markers and timeline. Others did not pay too much attention to the supplementary elements, e.g., the Title-view and Timeline-view.

The user test did, however, reveal some issues with the prototype. Multiple test participants stated that the position of highlighted text in the Text-view did not complement their individual reading pattern, which lead to an early update in supplementary data. This indicated that the focal point in the Text-view should be adjusted for individual readers. One test participants also struggled to pinpoint locations when the map was presenting a high detailed outline of a small area, i.e., when the zoom level was high. Some of the revealed issues are listed as recommendations for further work in Chapter 7.

However, the test result indicates that the prototype provides readers with an enriching reading experience, that precisely communicates locations in map context. This coincides with the thesis' goal.

Chapter 7

Future Work

This chapter presents recommendations for future work, divided into Short Term and Long Term plans.

7.1 Short Term

Case studies

The case study conducted in Chapter 5 in combination with user testing conducted in Chapter 6 has given valuable input in terms of design and features, as well as help proving the technical feasibility of the service. The prototype was, to a certain degree, able to reflect the case. This might, however, not hold true for other literary storylines. The prototype service is likely to benefit from more case studies, preferably by using literature of other genres. A war history may require a different set of map elements than e.g. a romantic novel. The prototype should be able to handle all types of literary storylines. It is therefore important to validate the prototype using storylines with varying themes and genres.

Map animation

The map updates its center position and zoom value, i.e., its outline, when readers skip from one text section to the next. The map may, e.g., display location markers in Scotland in "State 1" and Italy in "State 2". The map should then be focused around Scotland in "State 1" and Italy in "State 2". The map outline is updated instantaneously to reflect the change from "State 1" to "State 2". It might be hard for readers to know where the new map outline has been defined after an update, especially if the zoom level is high, i.e., the map displays high detailed information for a small land area. This issue was pointed out during user testing in Chapter 6.

Animations might solve this issue by animating the map transition from "State 1" to "State 2". A map animation could consist of the following sequence.

1. Zoom out from "location 1" into a low level zoom.

2. Translate the maps' center position from "location 1" to "location 2"

3. Zoom in on "location 2"

This animation sequence should help the reader pinpoint where the new map outline is defined, by briefly showing a broad map outline. An example for a map animation sequence is shown in Figure 7.1.

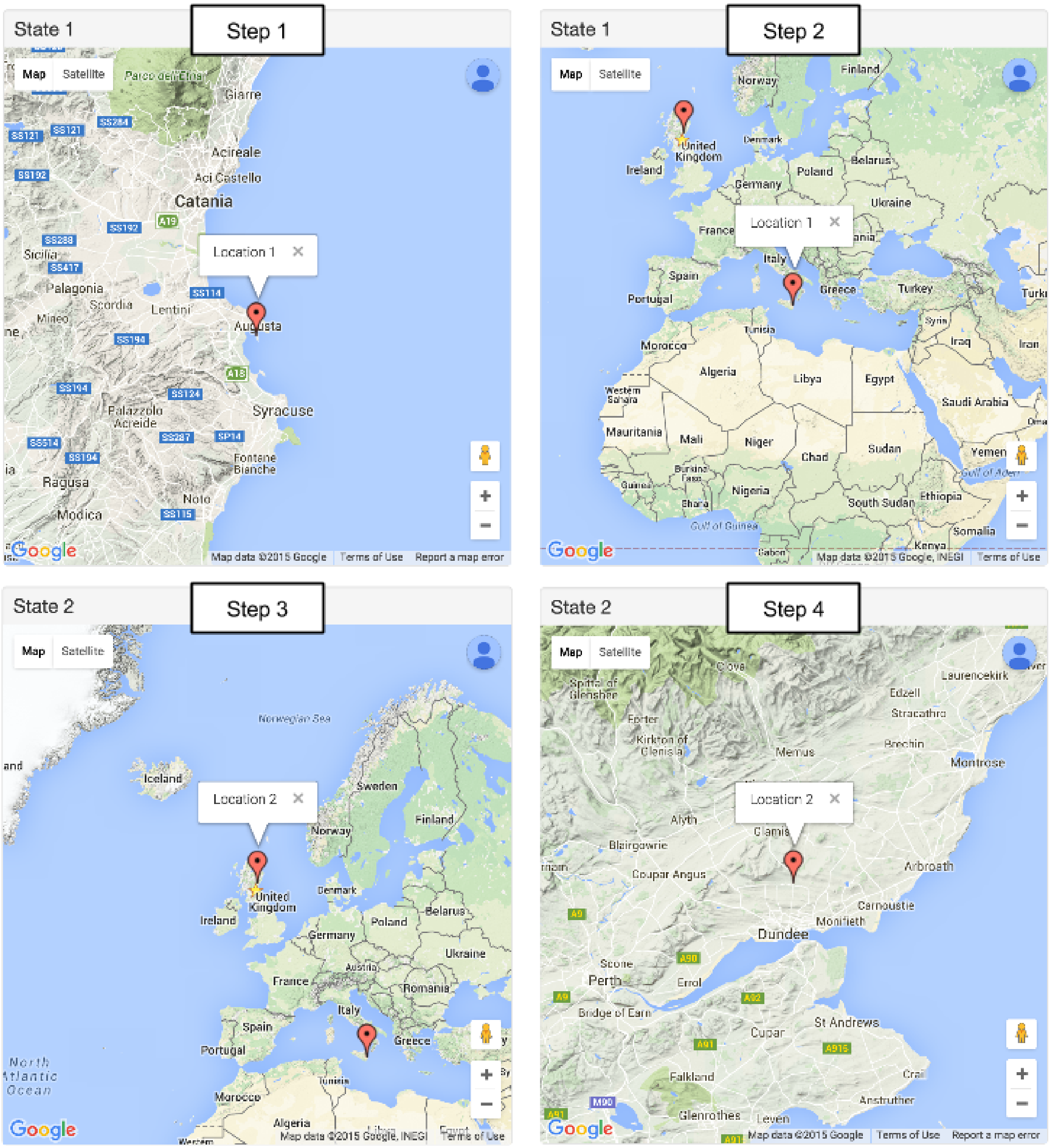


Figure 7.1: Map animation sequence. Step 1 shows the map outline for "State 1". Step 2 shows the update to a low level zoom. Step 3 shows the transition from "State 1" to "State 2". Step 4 shows the updated map outline for "State 2".

However, animation sequences require a small execution timespan, disabling the reader to interact with the map. The text scrolling should also be disabled during animations to prevent the user from triggering multiple animations at the time, i.e., setting of a chain reaction of map animations.

Disabling the UI for short periods of time may be considered annoying by some readers, as they lose interaction with the service. Map animations should therefore be made optional for individual readers.

Map information

The map is able to display markers and paths with different colors, direction indicators and information windows. The map elements may be given different meanings based on color. The map markers used in the case study in Chapter 5 were color coded based on nation. German items, e.g., Blücher, were colored blue, Norwegian items red and British items green.

Readers might find this hard to understand without further information. An Information-view should, therefore, be created to explain the different map elements. The Information-view should contain a figure for each map element represented in the storyline along with a textual description. This is illustrated in Figure 7.2. The Information-view should be implemented as a pop-up in order to save space in the UI.

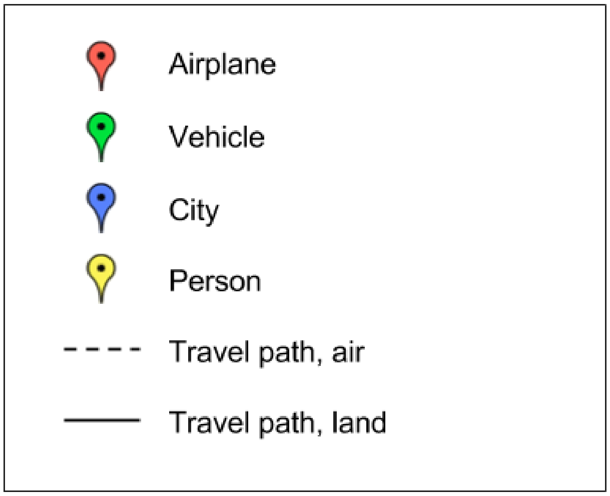


Figure 7.2: Information-view for map element descriptions.

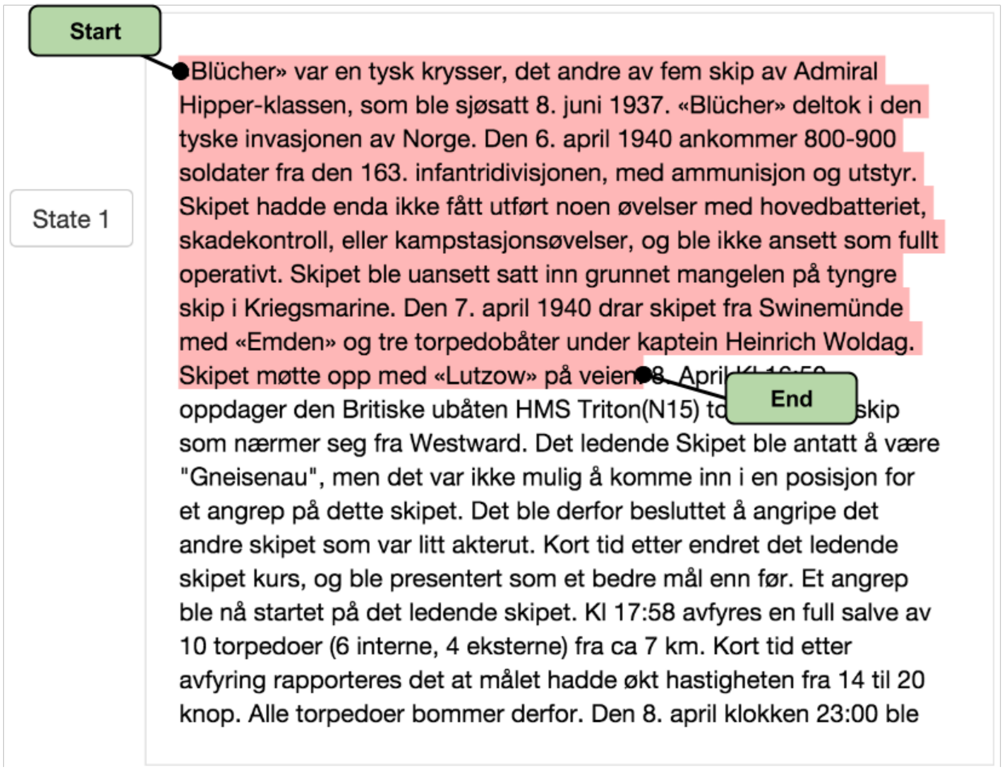
7.2 Long Term

User interface for creating and editing storylines

The prototype is populated with data, i.e., storylines, by adding object oriented classes directly into the DB. This approach is time consuming and limited to users with DB knowledge. The DB should eventually be populated by authors without DB knowledge. It is also beneficial if the process of creating a storyline with additional map, timeline and photos is as easy and quick as possible.

A simple UI for creating and editing storylines should therefore be implemented. The UI must contain a number of features, including import of literature text, separation of text into text sections and managing timeline, photos, map markers, map outlines and titles.

The author should be able to select an existing literary source in the form of a PDF file, or similar. The prototype should then import the file and automatically convert the text to HTML text. The author should be able to separate the text into text sections. This can be done by clicking on a start and end position in the text, i.e., marking a text section with the mouse cursor. This is illustrated in Figure 7.3. The marked text sections should be given an identifier, i.e., a "state value". The author could navigate between the different identifier in order to group map- and timeline related data to the respective text sections.



Blücher» var en tysk krysser, det andre av fem skip av Admiral Hipper-klassen, som ble sjøsatt 8. juni 1937. «Blücher» deltok i den tyske invasjonen av Norge. Den 6. april 1940 ankommer 800-900 soldater fra den 163. infantridivisjonen, med ammunisjon og utstyr. Skipet hadde enda ikke fått utført noen øvelser med hovedbatteriet, skadekontroll, eller kampstasjonsøvelser, og ble ikke ansett som fullt operativt. Skipet ble uansett satt inn grunnet mangelen på tyngre skip i Kriegsmarine. Den 7. april 1940 drar skipet fra Swinemünde med «Emden» og tre torpedobåter under kaptein Heinrich Woldag. Skipet møtte opp med «Lutzow» på veien 8. April Kl 16:50 oppdager den Britiske ubåten HMS Triton(N15) to skip som nærmer seg fra Westward. Det ledende Skipet ble antatt å være "Gneisenau", men det var ikke mulig å komme inn i en posisjon for et angrep på dette skipet. Det ble derfor besluttet å angripe det andre skipet som var litt akterut. Kort tid etter endret det ledende skipet kurs, og ble presentert som et bedre mål enn før. Et angrep ble nå startet på det ledende skipet. Kl 17:58 avfyres en full salve av 10 torpedoer (6 interne, 4 eksterne) fra ca 7 km. Kort tid etter avfyring rapporteres det at målet hadde økt hastigheten fra 14 til 20 knop. Alle torpedoer bommer derfor. Den 8. april klokken 23:00 ble

Figure 7.3: Selecting text sections from text.

A map outline and map markers should be created for each text sections, i.e., each "state". This can be done by presenting the author with a blank map, where he adjusts the center position and zoom level, as well as inserting location markers inside the map. The author should be able to add a date-time value for each state, as well as a map title. He should also be able to attach a title, a description and a photo to each location, i.e., each map marker.

Tablet optimization

The prototype is available as a website and therefore accessible on all devices with a display and an internet connection, e.g., mobiles, laptops, desktops and tablet. However, the UI has been optimized for wide and large screens, provided by laptops and desktops. A wide screen allows for a large side-by-side presentation of the Map-view and Text-view. A device with a narrow screen, e.g., a tablet, is not able to present the Text-view and Map-view side-by-side without compressing the size.

Some readers prefer to read on tablets, as pointed out during user testing in Chapter 6. The

UI should, therefore, also be improved for these devices. A possible solution would be to only display the Text-view and Timeline-view, and have the Map-view toggled between visible and hidden. This is illustrated in Figure 7.4. The green button in the bottom right corner is used to show and hide the map. The prototype could also notify the reader when the map is updated. The client side script could automatically measure the screen resolution and ratio before drawing the UI, and provide either the tablet- or desktop optimized layout accordingly.

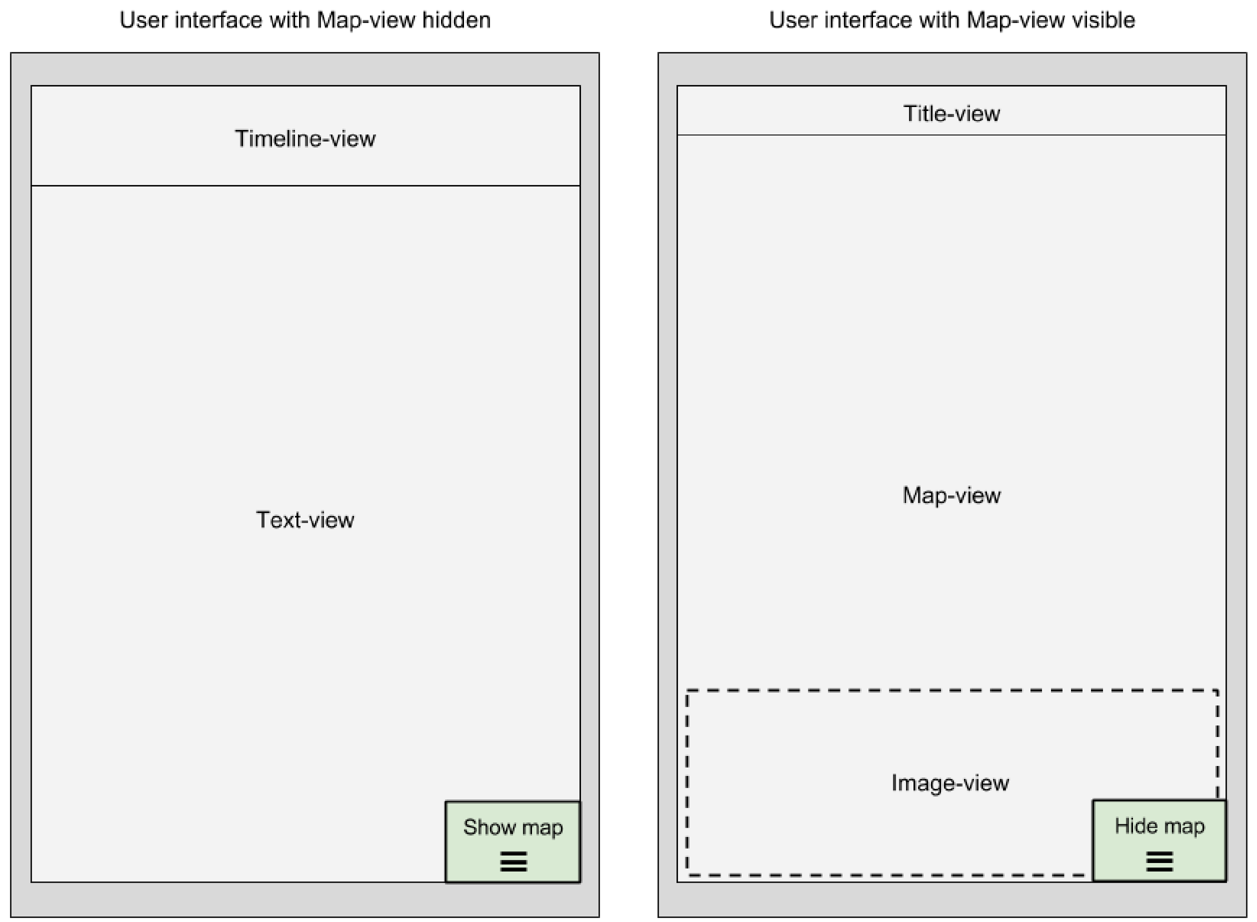

User interface with Map-view hidden
Timeline-view
Text-view
Show map
☰
User interface with Map-view visible
Title-view
Map-view
Image-view
Hide map
☰

Figure 7.4: UI optimized for tablet screens. The left side illustrates the default view, consisting of a Text-view and a Timeline-view. The right side illustrate the Map-view, which can be toggled between hidden and visible by the reader.

Multiple screens

The UI could be extended by introducing an additional screen for supplementary data. The main screen could present the Text-view while the additional screen presents the Map-, Timeline-

and Title-view. Utilization of two screens could be an alternative, or an addition, to having text and supplementary data visible at different times, as suggested in Section 7.2. Splitting up the UI might be beneficial to devices with a narrow screen, meaning the Map- and Text-view could be displayed in full size on two devices, e.g., two tablets or two computer monitors. The web based prototype must provide readers with two separate pages, i.e., website Uniform Resource Locators (URLs), in order to achieve this.

It should be a priority to update both screens, i.e., update text highlighting, timeline highlighting and map content, when the reader interacts with the Text-view in the main screen. This feature requires the web service to keep references and push updates to two screens. This can, for example, be done utilizing SignalR [9]. SignalR is a Microsoft framework with the purpose of pushing information to a selected set of web browser windows. SignalR works by having the web server push data to the client side, rather than having the client side request data from the server side. In this case, SignalR would push map related data to the supplementary screen when the reader request new text sections for the main screen, as well as pushing map updates to the supplementary screen when text highlighting is updated in the main screen.

Offline usage

The UI is populated with data from the web server via a series of HTTP requests. The prototype is, therefore, dependent of an internet connection in order to work. This structure limits the usage, as internet is only obtainable in certain geographical locations.

The service should alternately be able to function without internet. A possible solution is website caching. Website caching allows readers to locally store websites when connected to the internet. This is done by downloading a website's underlying UI structure, i.e., its HTML, CSS and JS. The stored websites can then be accessed at later times without internet. Some internet browsers, e.g., Google Chrome and Mozilla Firefox, offer web site caching as a default service.

"State" objects are originally transferred sequentially when the reader progresses from one chapter to the next. This structure requires the reader to have internet connection throughout the reading session. In this case, the reader would have to store the prototype website, i.e., the UI, as well as the full list of "state" objects.

7.3 Discussion

Further development of the prototype should primarily be focused on investigating desired functionality, and providing the reader with simple and informative supplementary data for an existing literary storyline. The prototype should be easy to use, self-explanatory and capable of presenting any existing literary storyline. More case studies and user tests can be good means for evaluating the prototype.

Short term recommendations, such as adding description to the map elements, should be included in later prototype iterations because they have potential to make the prototype's content more understandable. Long term recommendations, such as developing a UI for creating and editing storylines, should be implemented in the final stage of prototyping. These recommendations concerns the process of creating a commercialized product. However, they hold little value in demonstrating whether or not a service for combined text and maps gives an enriched reading experience.

Chapter 8

Summary

8.1 Summary and Conclusions

A prototype for combining existing literature with supplementary maps and photos has been developed using web based technology. Planning and early prototyping has been based on an evaluation of similar services. The evaluation is described in Chapter 2.

A case study, described in Chapter 5, has been used to evaluate the technical feasibility of the prototype. The case consisted of presenting an existing literary storyline in the prototype. The storyline featured the involvement of the heavy German cruiser Blücher during the German attack on Oslo, Norway in World War 2, 1940. The prototype was able to represent the storylines' involved parties, e.g., Blücher and Oscarsborg Fortress, in a map as markers, and key date-times in a timeline. The map and timeline was continuously updated to reflect different stages of the plot. For example, Blücher's travel path was expanded in the map as its position advanced from Swinemunde to Drøbak. Updates occurred automatically as the reader progressed in the storyline text.

A user test, described in Chapter 6, was conducted by five participants using the prototype. They provided valuable feedback that contributed to changes in the prototype and gave recommendations for future functionalities, described in Chapter 7. Multiple test participants stated that the prototype communicated locations in a precise way, and provided a joyful and informative reading experience. This confirms that the prototype can indeed offer an enriched reading experience, which coincides with the thesis' objective.

8.2 Discussion

A number of advantages are obtained by using web based technology as the underlying prototype platform. The prototype can be accessed though any internet browser. It is not limited to a specific device or platform, e.g., iOS tablets or Windows desktop computers. All web based services can also be accessed by multiple users at the same time. However, the prototype requires constant internet access in order to send and receive necessary data. This imposes limitations to where the prototype can be used, as internet is unavailable at some locations. This is often the case in, e.g., airplanes and uninhabited areas. Potential users might find this inconvenient or event unacceptable. The prototype should, therefore, be considered moved to a platform that functions without internet, or adjusted to overcome the issue.

The prototype UI has been optimized for desktop computers and laptops, i.e., large and wide screens. Devices with narrower screens need to compresses the UI. This might not be preferable to potential users, as tablets has become a common device for reading.

The literary storyline, used for case study in Chapter 5, featured a great amount of events with geographic significance. The case was, therefore, well suited to be presented in timeline- and map context. Other literary storylines might not be as suitable for the prototype. For example, the storyline of the book *Twelve Angry Men* [13] by Reginald Rose unfold in one single location, a court room, and in a relatively short timespan. It would have been demanding to offer enriching map and timeline data in this case. The prototype may, therefore, be limited to literary with a certain type of theme and genre.

The user test, conducted in Chapter 6, was concluded overall successful. Test participants were chosen with varying age and computer experience. This approach was expected to result in wide feedback, i.e., reveal more issues and likings. However, the user test included a relatively low amount of test participants. The conclusion may have differed by using more- or a different set of test participants or by using another case study. It is considered necessary to conduct more user tests and case studies before establishing a fixed set of UI elements, i.e., desired functionalities.

There is also alot of work left turning the prototype into a commercialized product. This mainly concerns creating a UI for authors, i.e., editing and updating storylines, as described in

Section 7.2. However, this falls outside of the thesis’ scope.

Appendix A

Additional Information

A.1 Case Study Literature

The literature used as foundation for the Cast Study conducted in Chapter 5 is included below. The literature features the involvement of heavy German cruiser Blücher during the German attack on Oslo, Norway in World War 2, 1940.

Blücher's involvement in the German attack on Oslo, Norway in World War 2, 1940.

«Blücher» var en tysk krysser, det andre av fem skip av Admiral Hipper-klassen, som ble sjøsatt 8. juni 1937. «Blücher» deltok i den tyske invasjonen av Norge.

Den 6. april 1940 ankommer 800-900 soldater fra den 163. infantridivisjonen, med ammunisjon og utstyr. Skipet hadde enda ikke fått utført noen øvelser med hovedbatteriet, skadekontroll, eller kampstasjonsøvelser, og ble ikke ansett som fullt operativt. Skipet ble uansett satt inn grunnet mangelen på tyngre skip i Kriegsmarine.

Den 7. april 1940 drar skipet fra Swinemünde med «Emden» og tre torpedobåter under kaptein Heinrich Woldag. Skipet møtte opp med «Lutzow» på veien. 8. April Kl 16:50 oppdager den Britiske ubåten HMS Triton(N15) to tyske krigsskip som nærmer seg fra Westward. Det ledende Skipet ble antatt å være "Gneisenau", men det var ikke mulig å komme inn i en posisjon for et *angrep på dette skipet. Det ble derfor besluttet å angripe det andre skipet som var litt akterut. Kort tid etter endret det ledende skipet kurs, og ble presentert som et bedre mål enn før. Et angrep ble nå startet på det ledende skipet. Kl 17:58 avfyres en full salve av 10 torpedoer (6 interne, 4 eksterne) fra ca 7 km. Kort tid etter avfyring rapporteres det at målet hadde økt hastigheten fra 14 til 20 knop. Alle torpedoer bommer derfor.*

Den 8. april klokken 23:00 ble kommandanten på Oscarsborg varslet om at fremmede fartøyer hadde trengt inn forbi Fulehuk fyr. Da den tyske invasjonsflåten kom inn i norsk territorialfarvann i ytre Oslofjord før midnatt 8. april 1940, lå «Pol III»(hvalbåt som fra 1939 ble innleid av Den Norske Marine) som vaktbåt mellom Bolærne og Rauøy. Kl.23.00 observerte vaktsjef om bord, Hans Bergan, to mørklagte fartøyer på vei nordover, kapteinen kom straks på broen og «Pol III» avfyrte et varselskudd. Da dukket et tredje fartøy opp aktenfra og rente inn i «Pol III»s akterende. Da fartøyet kom opp på siden av «Pol III» så man at det var den tyske torpedobåten «Albatros». Kommanderende på «Albatros» lyste på «Pol III» med lyskaster og forlangte trolig at bevoktningsfartøyet skulle overgi seg, noe Welding-Olsen trolig avslo. Samtidig rakk man å sende opp raketter som signaliserte at «Fremmede krigsskip trenger forbi bevoktningen» og dette ble oppfattet på Rauøy fort. Dermed var det norske forsvaret varslet om den tyske invasjonen. Welding-Olsen beordret en matros frem på bakken for å legge kanonen i vannrett stilling etter varselskuddet, om bord på «Albatros» trodde man «Pol III» ville åpne ild og åpnet selv ild med maskinkanoner og mitraljøser. Kulene slo inn i «Pol III»s overbygg og bro med stor kraft og Welding-Olsen ble drept. Han ble den første nordmann som falt for en inntrenger på norsk territorium siden 1814. «Albatros» tente så lyskasteren igjen og plukket opp de øvrige besetningsmedlemmene fra «Pol III» og de to fra prammen, før bevoktningsbåten ble skutt i brann og drev brennende av. «Albatros» satte deretter opp farten for å nå igjen den tyske hovedstyrken.

Kommandanten på Oscarsborg festning beordret ved midnatt alle kampledd besatt og gjort klare til kamp. Det var totalt 45 befal og 293 korporaler og menige – en rekke kampledd ble ikke besatt på grunn av mangel på personell. Klokken 0338 meldte Filtvet fyr at to større, mørklagte fartøyer hadde passert, og at en ombord i vaktbåten Furu, som var så nær den tyske flåtegruppen at de holdt på å kollidere, hadde hørt tysk tale ombord. Klokken 0418 ble det meldt fra Drøbak at fem skip, ett stort i tet og fire mindre påfølgende, var på vei innover. Klokken 04:21 norsk tid 9. april

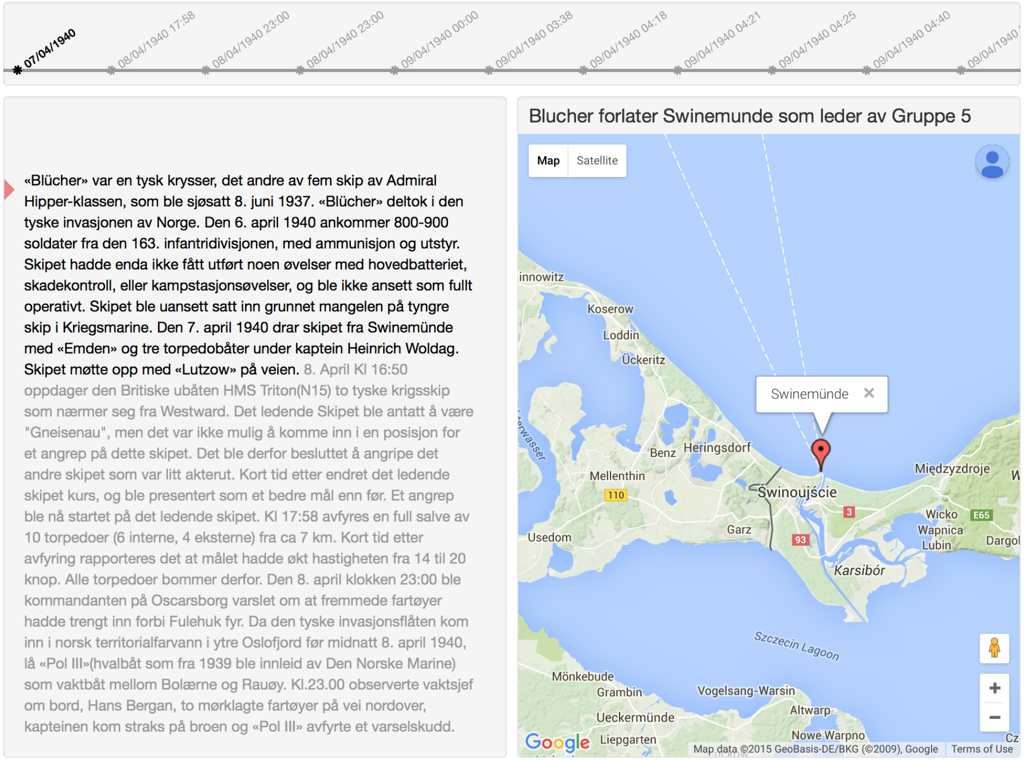
1940 åpnet festningen ild mot «Blücher» fortets tyskbygde 280 mm-kanoner engasjerte krysseren på en avstand av mellom 1 600 og 1 800 meter. Det første prosjektilet traff «Blücher» i stridsmasten like akter om broen. Det andre prosjektilet fra hovedbatteriet traff nær flyhangaren mellom aktre mast og skorsteinen. Dette skapte et inferno av flammer og røyk midtskips frem til forre mast. Festningen kunne bare skyte disse to skuddene, da personellet på 30 var utrente rekrutter som brukte lang tid på å lade kanonene. «Blücher» var dessuten kommet for nær og ute av disse kanonenes skuddfelt. Tyskerne kjente ikke til torpedobatteriet på nordre Kaholmen. Det var bygget i 1901, innsprengt i fjell. Første treffer kom trolig i det aktre av kjelerommene under broen. Det andre traff trolig turbinrom 1. Tilsammen satte dette alt fremdriftmaskineri ut av spill. De dampdrevne generatorene falt snart ut og mye av det elektriske falt ut, kun dieselagregatene fortsatte å gå. Klokken 04.40 snudde de andre skipene i invasjonsstyrken og slo retrett. Kapteinen ved «Lützow» hadde sett vannspruten fra torpedoene som traff «Blücher» og antok at veien var minelagt. Det planlagte «kuppet» for å tvinge Norges administrasjon til å overgi seg ble erstattet med en plan om å avansere landveien mot Oslo bortenfor Oscarsborgs rekkevidde. Mens styrken slo retrett klarte Oscarsborg å skade «Lützow», med Kopås batteri som hadde notert tre treff, og slått skipets 28 cm akterut våpentårn, «Bruno», ut av stand. Kopås fortsatte å skyte helt til skipene forsvant i tåken rundt 3000 meter fra batteriet. Totalt ble seks drept og 25 såret under slaget ved Oscarsborg festning.

Etter å ha blitt skadet under slaget ved Oscarsborg festning ble «Lützow» tilbakekalt til Tyskland for reparasjoner. På vei til Tyskland ble skipet torpedert og skadet kraftig av en britisk ubåt. Med alle motorer ødelagt av den andre torpedoen ankret krysseren opp nær Askholmene for å prøve å slokke de katastrofale brannene som herjet skipet. «Blüchers» torpedoer ble skutt mot land for å unngå at de eksploderte grunnet brannen ombord. Mannskapets desperate brannslokningsarbeid endte da brannen nådde skipets ammunisjonslager for Flak-kanoner rundt 05.30. Eksplosjonen rev et stort hull i skipets side, og eksplosjonen rev opp skottene mellom fyrerommene og brennstofflagrene, noe som førte til flere branner. «Blücher» var på dette tidspunktet fortapt. Klokken 06.22 rullet «Blücher» over på babord side, før skipet rullet opp-ned og til slutt sank med baugen først ned i Oslofjorden. Etter skipet sank fløt store mengden olje opp fra dypet, og la seg som en hinne rundt mannskapet som sloss for livet i Oslofjorden. Oljen ble påtent, og flere hundre tyskere døde.

Etter at "Blücher" gikk ned ble det satt inn et voldsomt bombeangrep utført av 42 fly fra Luftwaffe(Heinkel 111 bombefly) mot Oscarsborg. Angrepet varte fra ca. kl. 8 om morgenen til vel kl 18 om kvelden, bare avbrutt av pauser midt på dagen. Festningen forsøkte å besvare bombetokten med sine to Bofors 40mm L/60 luftvernskanoner, og sine syv Colt m/29 7.92mm mitraljøser - tre ved Seiersten, fire ved Håøya. I starten av angrepet var også fire maskingevær på taket til hovedbatteriet med i å besvare angrepet, men disse måtte evakuere tidlig i angrepet. Mellom angrepene i tidsrommet 12:00 til 13:30 gikk krysseren "Lützow" ut på sikker avstand fra festningen ved Filtvet, ca. 17000 m, og gir seg til å skyte mot Oscarsborg. Til sammen falt det mellom fem og seks hundre bomber over Oscarsborg. Etter at tyskerne hadde besatt Kopåsbattteriet ca kl. 1730, inngikk kommandanten oberst Birger Eriksen forhandlinger med tyskerne om overgivelse av Festningen. Den ble overgitt til tyskerne kl. 0900 den 10. april. Av angrepsstyrken på totalt 42 fly ble ett skutt ned og flere skade

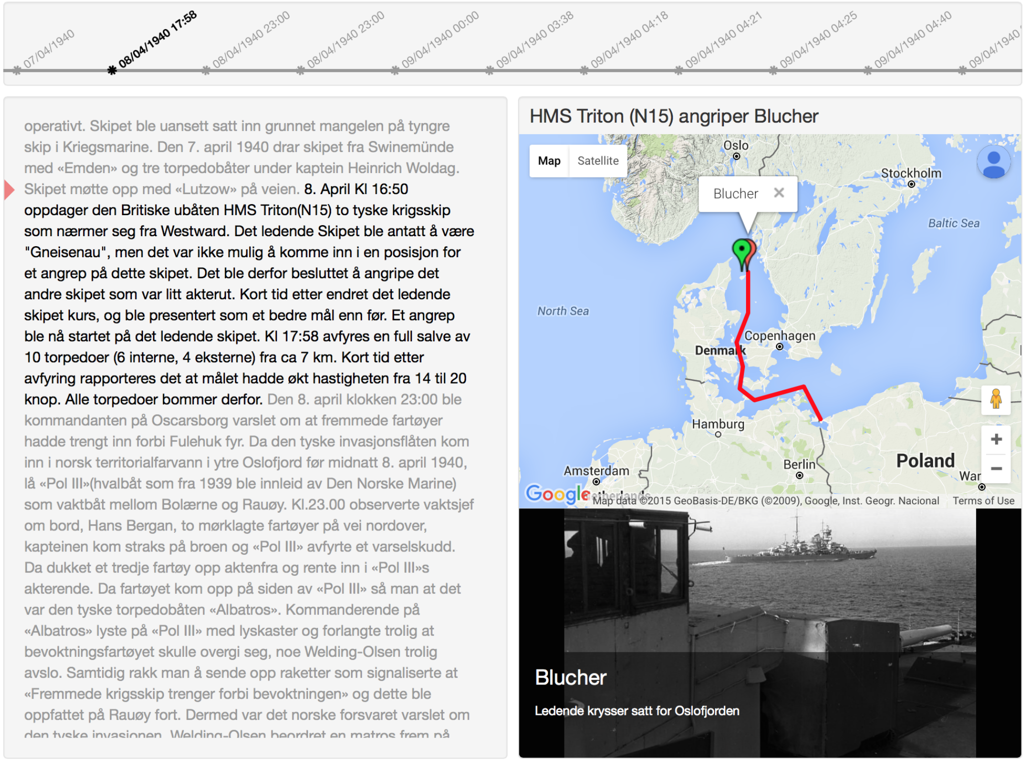
A.2 Case Study Results

Resulting screenshots of the Case Study, conducted in Chapter 5, are included below. The prototype has also been made available at www.folk.ntnu.no [15].



«Blücher» var en tysk krysser, det andre av fem skip av Admiral Hipper-klassen, som ble sjøsatt 8. juni 1937. «Blücher» deltok i den tyske invasjonen av Norge. Den 6. april 1940 ankommer 800-900 soldater fra den 163. infantridivisjonen, med ammunisjon og utstyr. Skipet hadde enda ikke fått utført noen øvelser med hovedbatteriet, skadekontroll, eller kampstasjonsøvelser, og ble ikke ansett som fullt operativt. Skipet ble uansett satt inn grunnet mangelen på tyngre skip i Kriegsmarine. Den 7. april 1940 drar skipet fra Swinemünde med «Emden» og tre torpedobåter under kaptein Heinrich Woldag. Skipet møtte opp med «Lutzow» på veien. 8. April Kl 16:50 oppdager den Britiske ubåten HMS Triton(N15) to tyske krigsskip som nærmer seg fra Westward. Det ledende Skipet ble antatt å være "Gneisenau", men det var ikke mulig å komme inn i en posisjon for et angrep på dette skipet. Det ble derfor besluttet å angripe det andre skipet som var litt akterut. Kort tid etter endret det ledende skipet kurs, og ble presentert som et bedre mål enn før. Et angrep ble nå startet på det ledende skipet. Kl 17:58 avfyres en full salve av 10 torpedoer (6 interne, 4 eksterne) fra ca 7 km. Kort tid etter avfyring rapporteres det at målet hadde økt hastigheten fra 14 til 20 knop. Alle torpedoer bommer derfor. Den 8. april klokken 23:00 ble kommandanten på Oscarsborg varslet om at fremmede fartøyer hadde trengt inn forbi Fulehuk fyr. Da den tyske invasjonsflåten kom inn i norsk territorialfarvann i ytre Oslofjord før midnatt 8. april 1940, lå «Pol III»(hvalbåt som fra 1939 ble innleid av Den Norske Marine) som vaktbåt mellom Bolærne og Rauøy. Kl.23.00 observerte vaktsjef om bord, Hans Bergan, to mørklagte fartøyer på vei nordover, kapteinen kom straks på broen og «Pol III» avfyrte et varselskudd.

Figure A.1: Prototype result, screenshot #1



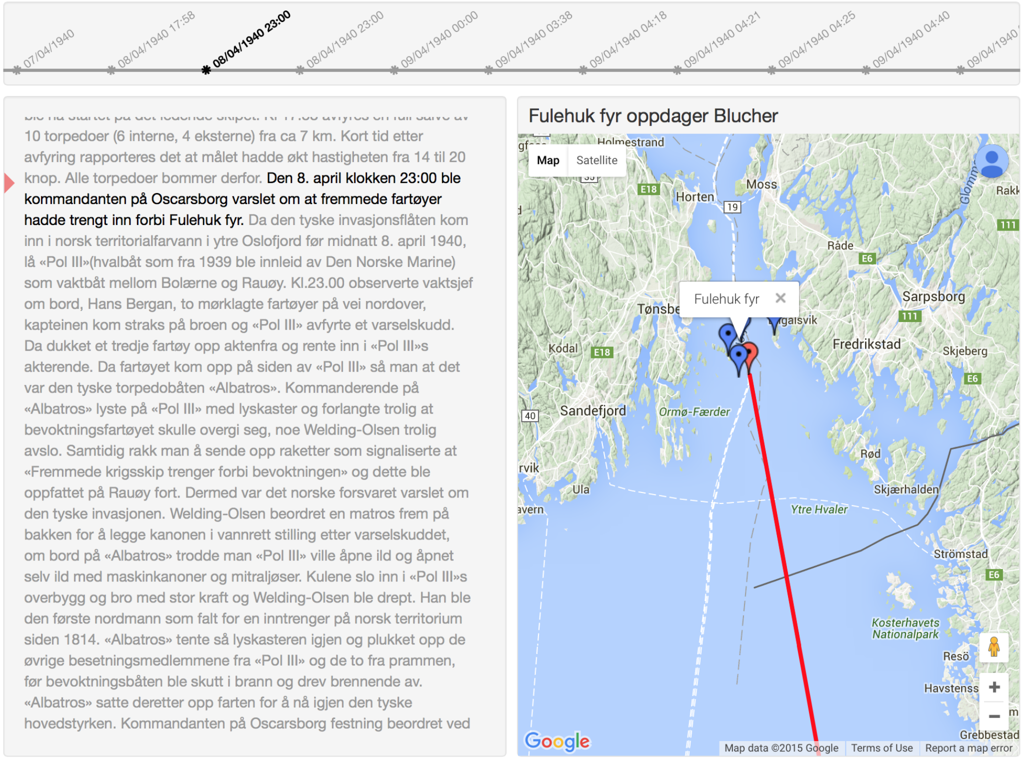
operativt. Skipet ble uansett satt inn grunnet mangelen på tyngre skip i Kriegsmarine. Den 7. april 1940 drar skipet fra Swinemünde med «Emden» og tre torpedobåter under kaptein Heinrich Woldag. Skipet møtte opp med «Lutzow» på veien. 8. April Kl 16:50 oppdager den Britiske ubåten HMS Triton(N15) to tyske krigsskip som nærmer seg fra Westward. Det ledende Skipet ble antatt å være "Gneisenau", men det var ikke mulig å komme inn i en posisjon for et angrep på dette skipet. Det ble derfor besluttet å angripe det andre skipet som var litt akterut. Kort tid etter endret det ledende skipet kurs, og ble presentert som et bedre mål enn før. Et angrep ble nå startet på det ledende skipet. Kl 17:58 avfyres en full salve av 10 torpedoer (6 interne, 4 eksterne) fra ca 7 km. Kort tid etter avfyring rapporteres det at målet hadde økt hastigheten fra 14 til 20 knop. Alle torpedoer bommer derfor. Den 8. april klokken 23:00 ble kommandanten på Oscarsborg varslet om at fremmede fartøyer hadde trengt inn forbi Fulehuk fyr. Da den tyske invasjonsflåten kom inn i norsk territorialfarvann i ytre Oslofjord før midnatt 8. april 1940, lå «Pol III»(hvalbåt som fra 1939 ble innleid av Den Norske Marine) som vaktbåt mellom Bolærne og Rauøy. Kl.23.00 observerte vaktsjef om bord, Hans Bergan, to mørklagte fartøyer på vei nordover, kapteinen kom straks på broen og «Pol III» avfyrte et varselskudd. Da dukket et tredje fartøy opp aktenfra og rente inn i «Pol III»s akterende. Da fartøyet kom opp på siden av «Pol III» så man at det var den tyske torpedobåten «Albatros». Kommanderende på «Albatros» lyste på «Pol III» med lyskaster og forlangte trolig at bevoktningsfartøyet skulle overgi seg, noe Welding-Olsen trolig avslo. Samtidig rakk man å sende opp raketter som signaliserte at «Fremmede krigsskip trenger forbi bevoktningen» og dette ble oppfattet på Rauøy fort. Dermed var det norske forsvaret varslet om den tyske invasjonen. Welding-Olsen beordret en matros frem på

HMS Triton (N15) angriper Blucher

Blucher

Ledende krysser satt for Oslofjorden

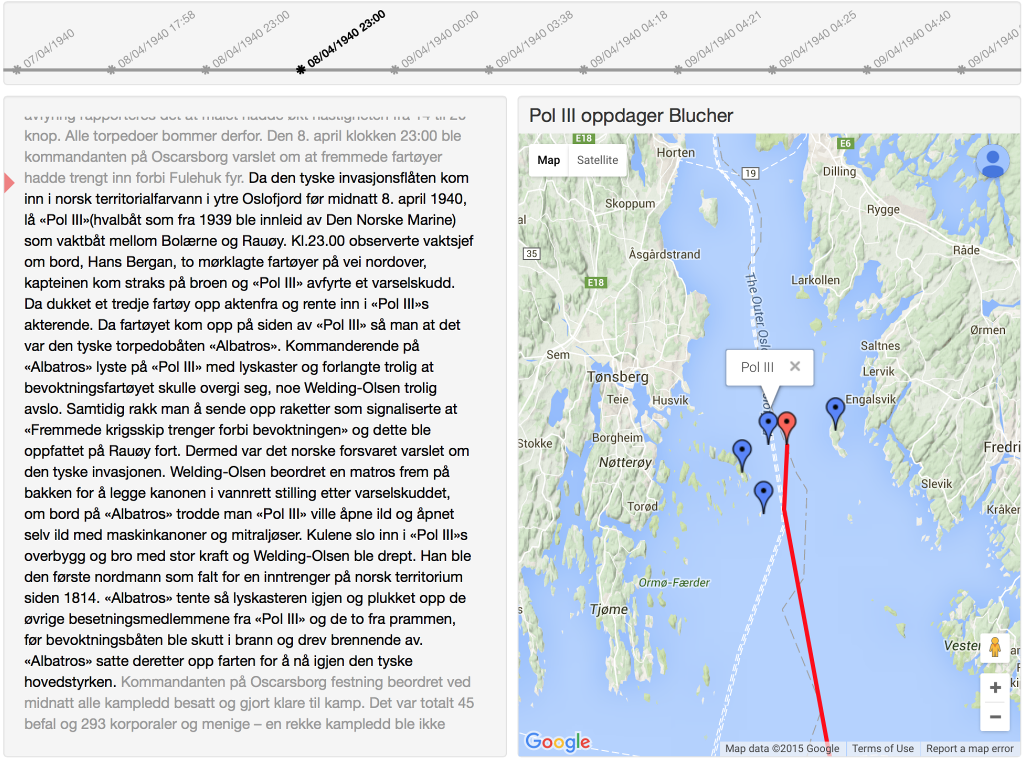
Figure A.2: Prototype result, screenshot #2



ble nå startet på det ledende skipet. Kl 17:58 avfyres en full salve av 10 torpedoer (6 interne, 4 eksterne) fra ca 7 km. Kort tid etter avfyring rapporteres det at målet hadde økt hastigheten fra 14 til 20 knop. Alle torpedoer bommer derfor. Den 8. april klokken 23:00 ble kommandanten på Oscarsborg varslet om at fremmede fartøyer hadde trengt inn forbi Fulehuk fyr. Da den tyske invasjonsflåten kom inn i norsk territorialfarvann i ytre Oslofjord før midnatt 8. april 1940, lå «Pol III»(hvalbåt som fra 1939 ble innleid av Den Norske Marine) som vaktbåt mellom Bolærne og Rauøy. Kl.23.00 observerte vaktsjef om bord, Hans Bergan, to mørklagte fartøyer på vei nordover, kapteinen kom straks på broen og «Pol III» avfyrte et varselskudd. Da dukket et tredje fartøy opp aktenfra og rente inn i «Pol III»s akterende. Da fartøyet kom opp på siden av «Pol III» så man at det var den tyske torpedobåten «Albatros». Kommanderende på «Albatros» lyste på «Pol III» med lyskaster og forlangte trolig at bevoktningsfartøyet skulle overgi seg, noe Welding-Olsen trolig avslo. Samtidig rakk man å sende opp raketter som signaliserte at «Fremmede krigsskip trenger forbi bevoktningen» og dette ble oppfattet på Rauøy fort. Dermed var det norske forsvaret varslet om den tyske invasjonen. Welding-Olsen beordret en matros frem på bakken for å legge kanonen i vannrett stilling etter varselskuddet, om bord på «Albatros» trodde man «Pol III» ville åpne ild og åpnet selv ild med maskinkanoner og mitraljøser. Kulene slo inn i «Pol III»s overbygg og bro med stor kraft og Welding-Olsen ble drept. Han ble den første nordmann som falt for en inntrenger på norsk territorium siden 1814. «Albatros» tente så lyskasteren igjen og plukket opp de øvrige besetningsmedlemmene fra «Pol III» og de to fra prammen, før bevoktningsbåten ble skutt i brann og drev brennende av. «Albatros» satte deretter opp farten for å nå igjen den tyske hovedstyrken. Kommandanten på Oscarsborg festning beordret ved

Fulehuk fyr oppdager Blucher

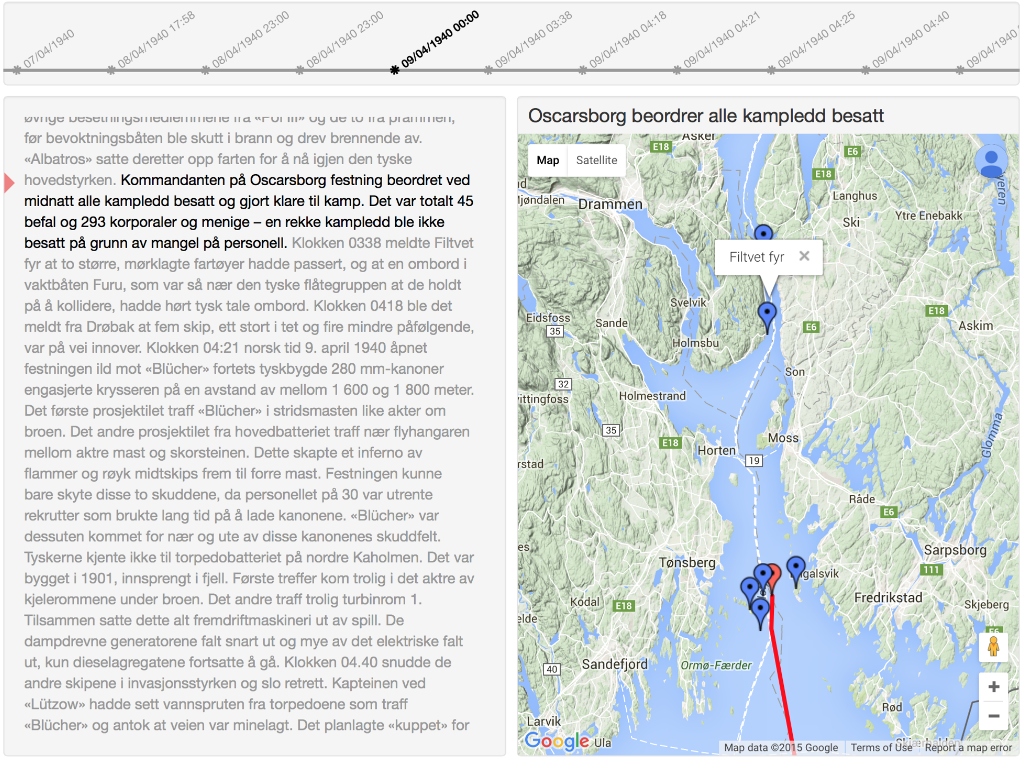
Figure A.3: Prototype result, screenshot #3



avfyring rapporteres det at målet hadde økt hastigheten fra 14 til 20 knop. Alle torpedoer bommer derfor. Den 8. april klokken 23:00 ble kommandanten på Oscarsborg varslet om at fremmede fartøyer hadde trengt inn forbi Fulehuk fyr. Da den tyske invasjonsflåten kom inn i norsk territorialfarvann i ytre Oslofjord før midnatt 8. april 1940, lå «Pol III»(hvalbåt som fra 1939 ble innleid av Den Norske Marine) som vaktbåt mellom Bolærne og Rauøy. Kl.23.00 observerte vaktsjef om bord, Hans Bergan, to mørklagte fartøyer på vei nordover, kapteinen kom straks på broen og «Pol III» avfyrte et varselskudd. Da dukket et tredje fartøy opp aktenfra og rente inn i «Pol III»s akterende. Da fartøyet kom opp på siden av «Pol III» så man at det var den tyske torpedobåten «Albatros». Kommanderende på «Albatros» lyste på «Pol III» med lyskaster og forlangte trolig at bevoktningsfartøyet skulle overgi seg, noe Welding-Olsen trolig avslo. Samtidig rakk man å sende opp raketter som signaliserte at «Fremmede krigsskip trenger forbi bevoktningen» og dette ble oppfattet på Rauøy fort. Dermed var det norske forsvaret varslet om den tyske invasjonen. Welding-Olsen beordret en matros frem på bakken for å legge kanonen i vannrett stilling etter varselskuddet, om bord på «Albatros» trodde man «Pol III» ville åpne ild og åpnet selv ild med maskinkanoner og mitraljøser. Kulene slo inn i «Pol III»s overbygg og bro med stor kraft og Welding-Olsen ble drept. Han ble den første nordmann som falt for en inntrenger på norsk territorium siden 1814. «Albatros» tente så lyskasteren igjen og plukket opp de øvrige besetningsmedlemmene fra «Pol III» og de to fra prammen, før bevoktningsbåten ble skutt i brann og drev brennende av. «Albatros» satte deretter opp farten for å nå igjen den tyske hovedstyrken. Kommandanten på Oscarsborg festning beordret ved midnatt alle kampledd besatt og gjort klare til kamp. Det var totalt 45 befal og 293 korporaler og menige – en rekke kampledd ble ikke

Pol III oppdager Blucher

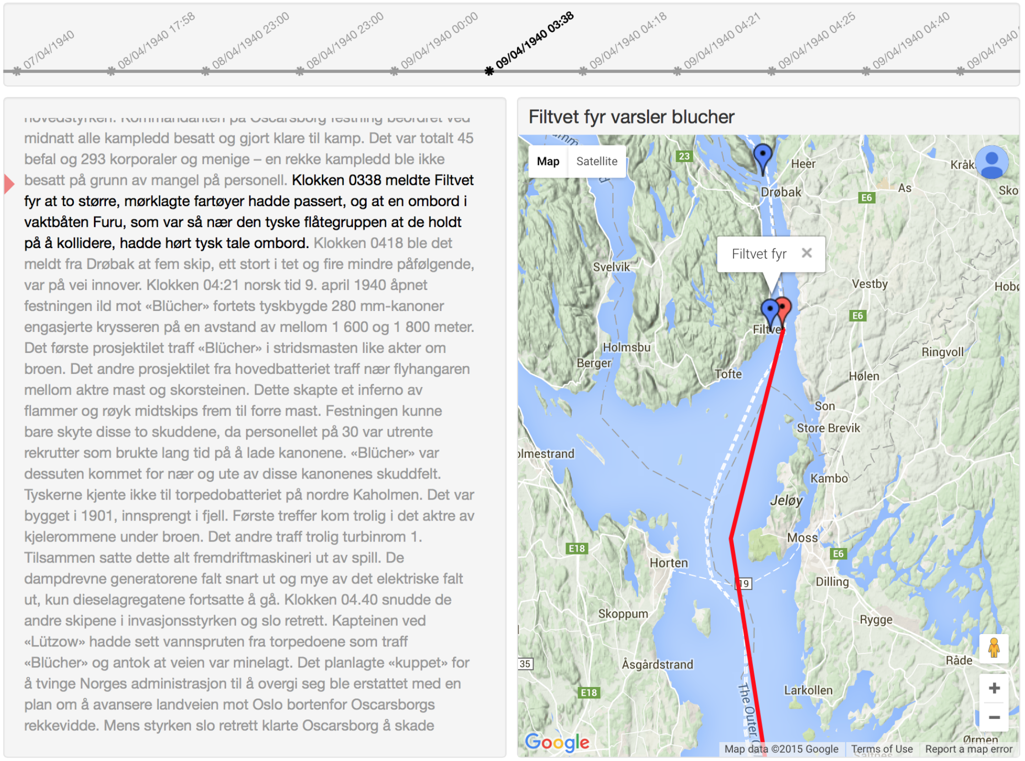
Figure A.4: Prototype result, screenshot #4



øvrige besetningsmedlemmene fra «Pol III» og de to fra prammen, før bevoktningsbåten ble skutt i brann og drev brennende av. «Albatros» satte deretter opp farten for å nå igjen den tyske hovedstyrken. Kommandanten på Oscarsborg festning beordret ved midnatt alle kampledd besatt og gjort klare til kamp. Det var totalt 45 befal og 293 korporaler og menige – en rekke kampledd ble ikke besatt på grunn av mangel på personell. Klokken 0338 meldte Filtvet fyr at to større, mørklagte fartøyer hadde passert, og at en ombord i vaktbåten Furu, som var så nær den tyske flåtegruppen at de holdt på å kollidere, hadde hørt tysk tale ombord. Klokken 0418 ble det meldt fra Drøbak at fem skip, ett stort i tet og fire mindre påfølgende, var på vei innover. Klokken 04:21 norsk tid 9. april 1940 åpnet festningen ild mot «Blücher» fortets tyskbygde 280 mm-kanoner engasjerte krysseren på en avstand av mellom 1 600 og 1 800 meter. Det første prosjektilet traff «Blücher» i stridsmasten like akter om broen. Det andre prosjektilet fra hovedbatteriet traff nær flyhangaren mellom aktre mast og skorsteinen. Dette skapte et inferno av flammer og røyk midtskips frem til forre mast. Festningen kunne bare skyte disse to skuddene, da personellet på 30 var utrente rekrutter som brukte lang tid på å lade kanonene. «Blücher» var dessuten kommet for nær og ute av disse kanonenes skuddfelt. Tyskerne kjente ikke til torpedobatteriet på nordre Kaholmen. Det var bygget i 1901, innsprengt i fjell. Første treffer kom trolig i det aktre av kjelerommene under broen. Det andre traff trolig turbinrom 1. Tilsammen satte dette alt fremdriftmaskineri ut av spill. De dampdrevne generatorene falt snart ut og mye av det elektriske falt ut, kun dieselagregatene fortsatte å gå. Klokken 04.40 snudde de andre skipene i invasjonsstyrken og slo retrett. Kapteinen ved «Lützow» hadde sett vannspruten fra torpedoene som traff «Blücher» og antok at veien var minelagt. Det planlagte «kuppet» for

Oscarsborg beordrer alle kampledd besatt

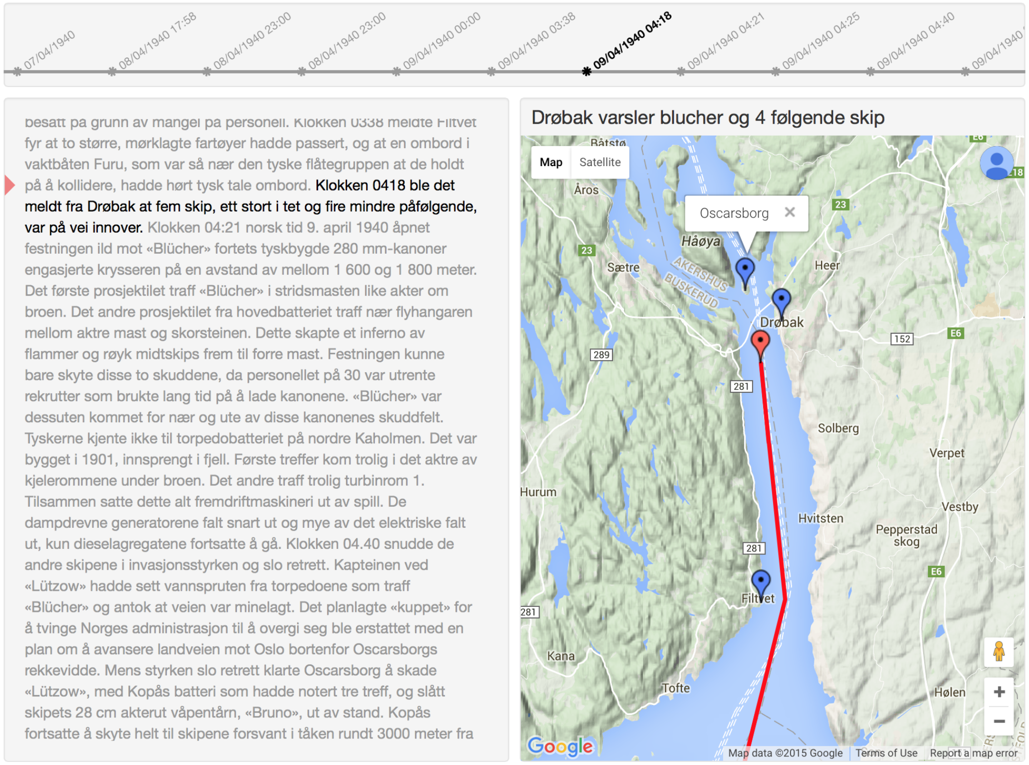
Figure A.5: Prototype result, screenshot #5



hovedstyrken. Kommandanten på Oscarsborg festning beordret ved midnatt alle kampledd besatt og gjort klare til kamp. Det var totalt 45 befal og 293 korporaler og menige – en rekke kampledd ble ikke besatt på grunn av mangel på personell. Klokken 0338 meldte Filtvet fyr at to større, mørklagte fartøyer hadde passert, og at en ombord i vaktbåten Furu, som var så nær den tyske flåtegruppen at de holdt på å kollidere, hadde hørt tysk tale ombord. Klokken 0418 ble det meldt fra Drøbak at fem skip, ett stort i tet og fire mindre påfølgende, var på vei innover. Klokken 04:21 norsk tid 9. april 1940 åpnet festningen ild mot «Blücher» fortets tyskbygde 280 mm-kanoner engasjerte krysseren på en avstand av mellom 1 600 og 1 800 meter. Det første prosjektilet traff «Blücher» i stridsmasten like akter om broen. Det andre prosjektilet fra hovedbatteriet traff nær flyhangaren mellom aktre mast og skorsteinen. Dette skapte et inferno av flammer og røyk midtskips frem til forre mast. Festningen kunne bare skyte disse to skuddene, da personellet på 30 var utrente rekrutter som brukte lang tid på å lade kanonene. «Blücher» var dessuten kommet for nær og ute av disse kanonenes skuddfelt. Tyskerne kjente ikke til torpedobatteriet på nordre Kaholmen. Det var bygget i 1901, innsprengt i fjell. Første treffer kom trolig i det aktre av kjelerommene under broen. Det andre traff trolig turbinrom 1. Tilsammen satte dette alt fremdriftmaskineri ut av spill. De dampdrevne generatorene falt snart ut og mye av det elektriske falt ut, kun dieselagregatene fortsatte å gå. Klokken 04.40 snudde de andre skipene i invasjonsstyrken og slo retrett. Kapteinen ved «Lützow» hadde sett vannspruten fra torpedoene som traff «Blücher» og antok at veien var minelagt. Det planlagte «kuppet» for å tvinge Norges administrasjon til å overgi seg ble erstattet med en plan om å avansere landveien mot Oslo bortenfor Oscarsborgs rekkevidde. Mens styrken slo retrett klarte Oscarsborg å skade

Filtvet fyr varsler blucher

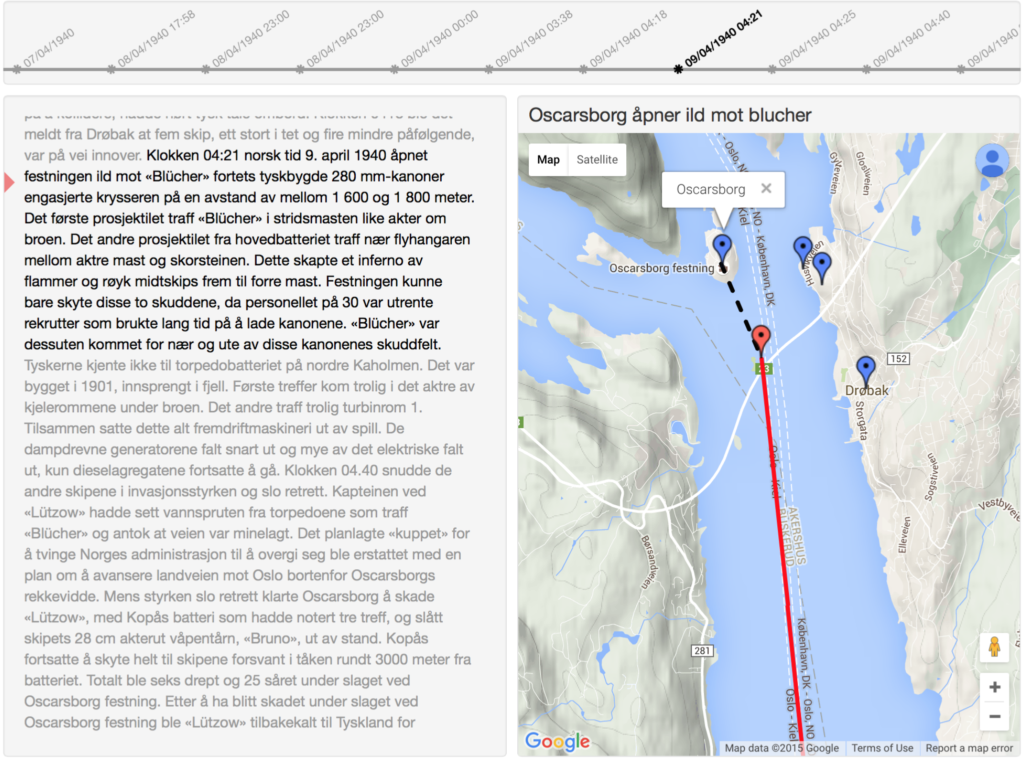
Figure A.6: Prototype result, screenshot #6



besatt på grunn av mangel på personell. Klokken 0338 meldte Filtvet fyr at to større, mørklagte fartøyer hadde passert, og at en ombord i vaktbåten Furu, som var så nær den tyske flåtegruppen at de holdt på å kollidere, hadde hørt tysk tale ombord. Klokken 0418 ble det meldt fra Drøbak at fem skip, ett stort i tet og fire mindre påfølgende, var på vei innover. Klokken 04:21 norsk tid 9. april 1940 åpnet festningen ild mot «Blücher» fortets tyskbygde 280 mm-kanoner engasjerte krysseren på en avstand av mellom 1 600 og 1 800 meter. Det første prosjektilet traff «Blücher» i stridsmasten like akter om broen. Det andre prosjektilet fra hovedbatteriet traff nær flyhangaren mellom aktre mast og skorsteinen. Dette skapte et inferno av flammer og røyk midtskips frem til forre mast. Festningen kunne bare skyte disse to skuddene, da personellet på 30 var utrente rekrutter som brukte lang tid på å lade kanonene. «Blücher» var dessuten kommet for nær og ute av disse kanonenes skuddfelt. Tyskerne kjente ikke til torpedobatteriet på nordre Kaholmen. Det var bygget i 1901, innsprengt i fjell. Første treffer kom trolig i det aktre av kjelerommene under broen. Det andre traff trolig turbinrom 1. Tilsammen satte dette alt fremdriftmaskineri ut av spill. De dampdrevne generatorene falt snart ut og mye av det elektriske falt ut, kun dieselagregatene fortsatte å gå. Klokken 04.40 snudde de andre skipene i invasjonsstyrken og slo retrett. Kapteinen ved «Lützow» hadde sett vannspruten fra torpedoene som traff «Blücher» og antok at veien var minelagt. Det planlagte «kuppet» for å tvinge Norges administrasjon til å overgi seg ble erstattet med en plan om å avansere landveien mot Oslo bortenfor Oscarsborgs rekkevidde. Mens styrken slo retrett klarte Oscarsborg å skade «Lützow», med Kopås batteri som hadde notert tre treff, og slått skipets 28 cm akterut våpentårn, «Bruno», ut av stand. Kopås fortsatte å skyte helt til skipene forsvant i tåken rundt 3000 meter fra

Drøbak varsler blucher og 4 følgende skip

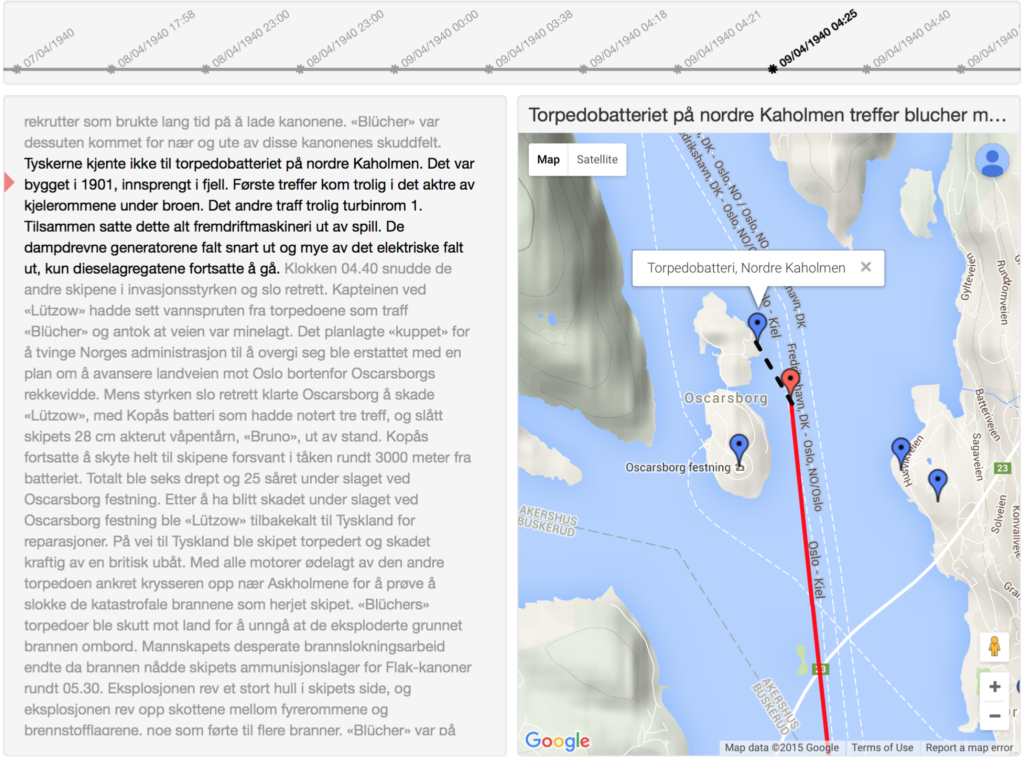
Figure A.7: Prototype result, screenshot #7



meldt fra Drøbak at fem skip, ett stort i tet og fire mindre påfølgende, var på vei innover. Klokken 04:21 norsk tid 9. april 1940 åpnet festningen ild mot «Blücher» fortets tyskbygde 280 mm-kanoner engasjerte krysseren på en avstand av mellom 1 600 og 1 800 meter. Det første prosjektilet traff «Blücher» i stridsmasten like akter om broen. Det andre prosjektilet fra hovedbatteriet traff nær flyhangaren mellom aktre mast og skorsteinen. Dette skapte et inferno av flammer og røyk midtskips frem til forre mast. Festningen kunne bare skyte disse to skuddene, da personellet på 30 var utrente rekrutter som brukte lang tid på å lade kanonene. «Blücher» var dessuten kommet for nær og ute av disse kanonenes skuddfelt. Tyskerne kjente ikke til torpedobatteriet på nordre Kaholmen. Det var bygget i 1901, innsprengt i fjell. Første treffer kom trolig i det aktre av kjelerommene under broen. Det andre traff trolig turbinrom 1. Tilsammen satte dette alt fremdriftmaskineri ut av spill. De dampdrevne generatorene falt snart ut og mye av det elektriske falt ut, kun dieselagregatene fortsatte å gå. Klokken 04.40 snudde de andre skipene i invasjonsstyrken og slo retrett. Kapteinen ved «Lützow» hadde sett vannspruten fra torpedoene som traff «Blücher» og antok at veien var minelagt. Det planlagte «kuppet» for å tvinge Norges administrasjon til å overgi seg ble erstattet med en plan om å avansere landveien mot Oslo bortenfor Oscarsborgs rekkevidde. Mens styrken slo retrett klarte Oscarsborg å skade «Lützow», med Kopås batteri som hadde notert tre treff, og slått skipets 28 cm akterut våpentårn, «Bruno», ut av stand. Kopås fortsatte å skyte helt til skipene forsvant i tåken rundt 3000 meter fra batteriet. Totalt ble seks drept og 25 såret under slaget ved Oscarsborg festning. Etter å ha blitt skadet under slaget ved Oscarsborg festning ble «Lützow» tilbakekalt til Tyskland for

Oscarsborg åpner ild mot blucher

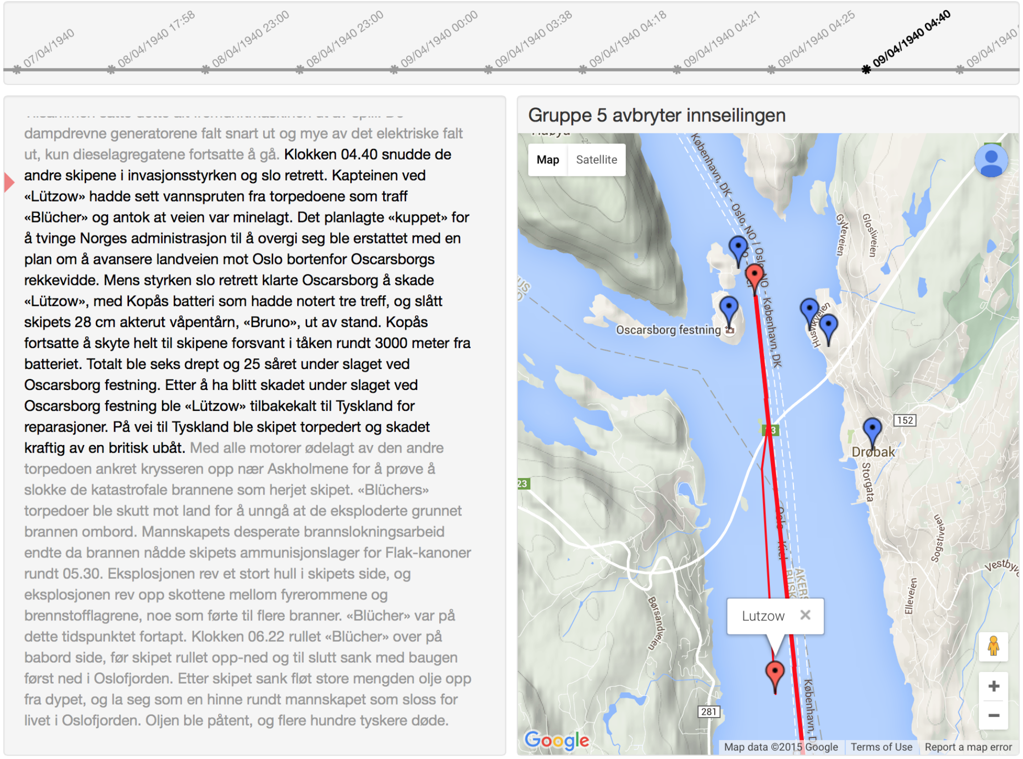
Figure A.8: Prototype result, screenshot #8



rekrutter som brukte lang tid på å lade kanonene. «Blücher» var dessuten kommet for nær og ute av disse kanonenes skuddfelt. Tyskerne kjente ikke til torpedobatteriet på nordre Kaholmen. Det var bygget i 1901, innsprengt i fjell. Første treffer kom trolig i det aktre av kjelerommene under broen. Det andre traff trolig turbinrom 1. Tilsammen satte dette alt fremdriftmaskineri ut av spill. De dampdrevne generatorene falt snart ut og mye av det elektriske falt ut, kun dieselagregatene fortsatte å gå. Klokken 04.40 snudde de andre skipene i invasjonsstyrken og slo retrett. Kapteinen ved «Lützow» hadde sett vannspruten fra torpedoene som traff «Blücher» og antok at veien var minelagt. Det planlagte «kuppet» for å tvinge Norges administrasjon til å overgi seg ble erstattet med en plan om å avansere landveien mot Oslo bortenfor Oscarsborgs rekkevidde. Mens styrken slo retrett klarte Oscarsborg å skade «Lützow», med Kopås batteri som hadde notert tre treff, og slått skipets 28 cm akterut våpentårn, «Bruno», ut av stand. Kopås fortsatte å skyte helt til skipene forsvant i tåken rundt 3000 meter fra batteriet. Totalt ble seks drept og 25 såret under slaget ved Oscarsborg festning. Etter å ha blitt skadet under slaget ved Oscarsborg festning ble «Lützow» tilbakekalt til Tyskland for reparasjoner. På vei til Tyskland ble skipet torpedert og skadet kraftig av en britisk ubåt. Med alle motorer ødelagt av den andre torpedoen ankret krysseren opp nær Askholmene for å prøve å slokke de katastrofale brannene som herjet skipet. «Blüchers» torpedoer ble skutt mot land for å unngå at de eksploderte grunnet brannen ombord. Mannskapets desperate brannslokningsarbeid endte da brannen nådde skipets ammunisjonslager for Flak-kanoner rundt 05.30. Eksplosjonen rev et stort hull i skipets side, og eksplosjonen rev opp skottene mellom fyrerommene og brennstofflagrene, noe som førte til flere branner. «Blücher» var på

Torpedobatteriet på nordre Kaholmen treffer blucher m…

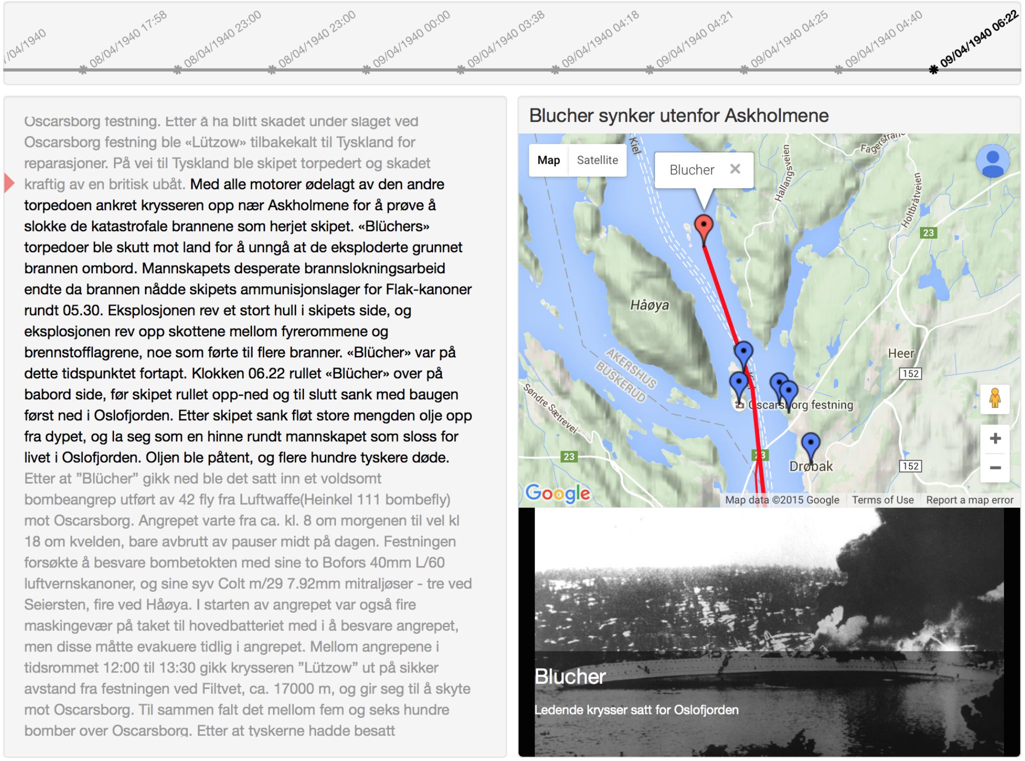
Figure A.9: Prototype result, screenshot #9



dampdrevne generatorene falt snart ut og mye av det elektriske falt ut, kun dieselagregatene fortsatte å gå. Klokken 04.40 snudde de andre skipene i invasjonsstyrken og slo retrett. Kapteinen ved «Lützow» hadde sett vannspruten fra torpedoene som traff «Blücher» og antok at veien var minelagt. Det planlagte «kuppet» for å tvinge Norges administrasjon til å overgi seg ble erstattet med en plan om å avansere landveien mot Oslo bortenfor Oscarsborgs rekkevidde. Mens styrken slo retrett klarte Oscarsborg å skade «Lützow», med Kopås batteri som hadde notert tre treff, og slått skipets 28 cm akterut våpentårn, «Bruno», ut av stand. Kopås fortsatte å skyte helt til skipene forsvant i tåken rundt 3000 meter fra batteriet. Totalt ble seks drept og 25 såret under slaget ved Oscarsborg festning. Etter å ha blitt skadet under slaget ved Oscarsborg festning ble «Lützow» tilbakekalt til Tyskland for reparasjoner. På vei til Tyskland ble skipet torpedert og skadet kraftig av en britisk ubåt. Med alle motorer ødelagt av den andre torpedoen ankret krysseren opp nær Askholmene for å prøve å slokke de katastrofale brannene som herjet skipet. «Blüchers» torpedoer ble skutt mot land for å unngå at de eksploderte grunnet brannen ombord. Mannskapets desperate brannslokningsarbeid endte da brannen nådde skipets ammunisjonslager for Flak-kanoner rundt 05.30. Eksplosjonen rev et stort hull i skipets side, og eksplosjonen rev opp skottene mellom fyrrommene og brennstofflagrene, noe som førte til flere branner. «Blücher» var på dette tidspunktet fortapt. Klokken 06.22 rullet «Blücher» over på babord side, før skipet rullet opp-ned og til slutt sank med baugen først ned i Oslofjorden. Etter skipet sank fløt store mengden olje opp fra dypet, og la seg som en hinne rundt mannskapet som sloss for livet i Oslofjorden. Oljen ble påtent, og flere hundre tyskere døde.

Gruppe 5 avbryter innseilingen

Figure A.10: Prototype result, screenshot #10



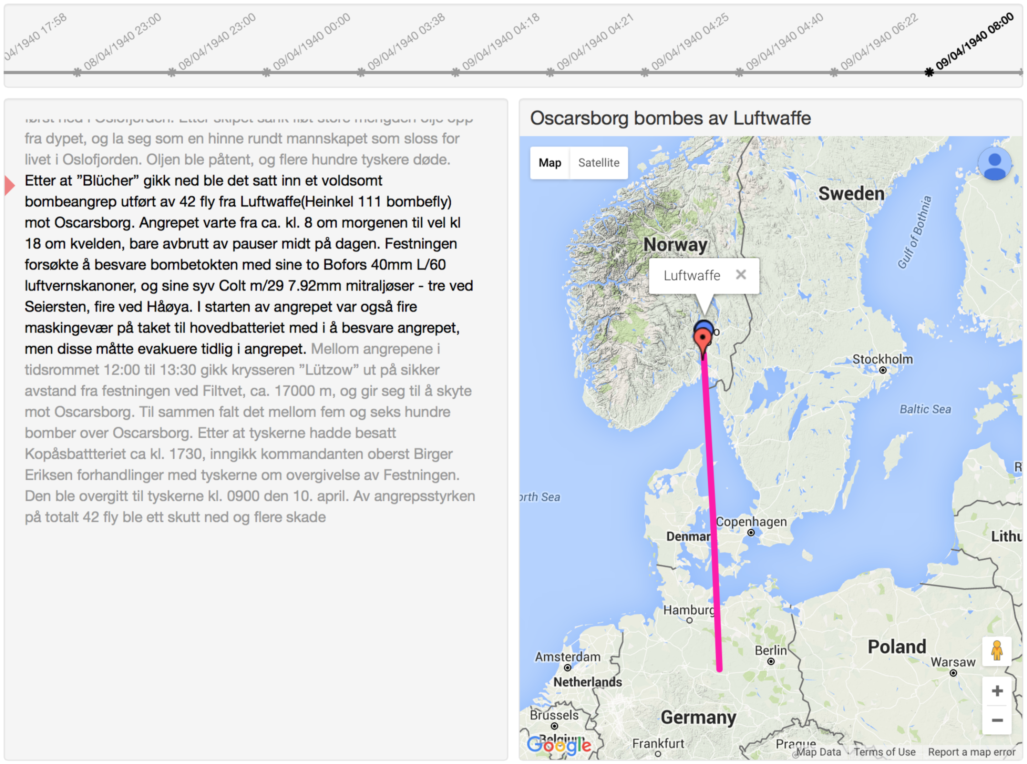
Oscarsborg festning. Etter å ha blitt skadet under slaget ved Oscarsborg festning ble «Lützow» tilbakekalt til Tyskland for reparasjoner. På vei til Tyskland ble skipet torpedert og skadet kraftig av en britisk ubåt. Med alle motorer ødelagt av den andre torpedoen ankret krysseren opp nær Askholmene for å prøve å slokke de katastrofale brannene som herjet skipet. «Blüchers» torpedoer ble skutt mot land for å unngå at de eksploderte grunnet brannen ombord. Mannskapets desperate brannslokningsarbeid endte da brannen nådde skipets ammunisjonslager for Flak-kanoner rundt 05.30. Eksplosjonen rev et stort hull i skipets side, og eksplosjonen rev opp skottene mellom fyrrommene og brennstofflagrene, noe som førte til flere branner. «Blücher» var på dette tidspunktet fortapt. Klokken 06.22 rullet «Blücher» over på babord side, før skipet rullet opp-ned og til slutt sank med baugen først ned i Oslofjorden. Etter skipet sank fløt store mengden olje opp fra dypet, og la seg som en hinne rundt mannskapet som sloss for livet i Oslofjorden. Oljen ble påtent, og flere hundre tyskere døde. Etter at "Blücher" gikk ned ble det satt inn et voldsomt bombeangrep utført av 42 fly fra Luftwaffe(Heinkel 111 bombefly) mot Oscarsborg. Angrepet varte fra ca. kl. 8 om morgenen til vel kl 18 om kvelden, bare avbrutt av pauser midt på dagen. Festningen forsøkte å besvare bombetokten med sine to Bofors 40mm L/60 luftvernskanoner, og sine syv Colt m/29 7.92mm mitraljøser - tre ved Seiersten, fire ved Håøya. I starten av angrepet var også fire maskingevær på taket til hovedbatteriet med i å besvare angrepet, men disse måtte evakuere tidlig i angrepet. Mellom angrepene i tidsrommet 12:00 til 13:30 gikk kryssseren "Lützow" ut på sikker avstand fra festningen ved Filtvet, ca. 17000 m, og gir seg til å skyte mot Oscarsborg. Til sammen falt det mellom fem og seks hundre bomber over Oscarsborg. Etter at tyskerne hadde besatt

Blucher synker utenfor Askholmene

Blucher

Ledende krysser satt for Oslofjorden

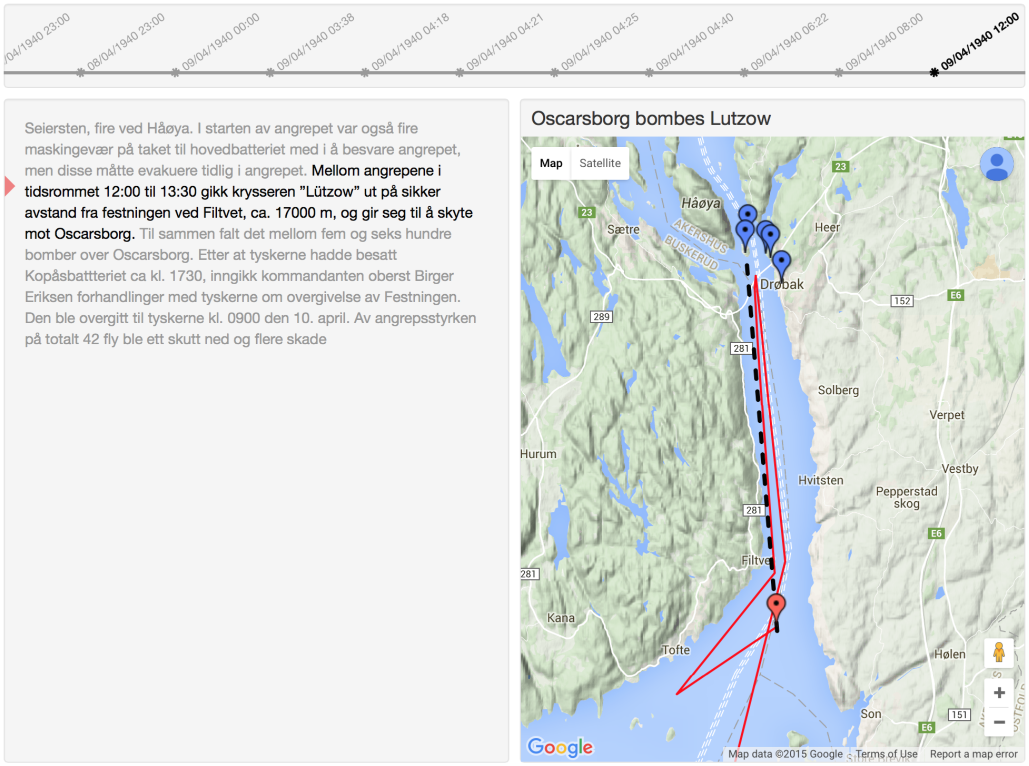
Figure A.11: Prototype result, screenshot #11



først ned i Oslofjorden. Etter skipet sank fløt store mengden olje opp fra dypet, og la seg som en hinne rundt mannskapet som sloss for livet i Oslofjorden. Oljen ble påtent, og flere hundre tyskere døde. Etter at ”Blücher” gikk ned ble det satt inn et voldsomt bombeangrep utført av 42 fly fra Luftwaffe(Heinkel 111 bombefly) mot Oscarsborg. Angrepet varte fra ca. kl. 8 om morgenen til vel kl 18 om kvelden, bare avbrutt av pauser midt på dagen. Festningen forsøkte å besvare bombetokten med sine to Bofors 40mm L/60 luftvernskanoner, og sine syv Colt m/29 7.92mm mitraljøser - tre ved Seiersten, fire ved Håøya. I starten av angrepet var også fire maskingevær på taket til hovedbatteriet med i å besvare angrepet, men disse måtte evakuere tidlig i angrepet. Mellom angrepene i tidsrommet 12:00 til 13:30 gikk krysseren ”Lützow” ut på sikker avstand fra festningen ved Filtvet, ca. 17000 m, og gir seg til å skyte mot Oscarsborg. Til sammen falt det mellom fem og seks hundre bomber over Oscarsborg. Etter at tyskerne hadde besatt Kopåsbattteriet ca kl. 1730, inngikk kommandanten oberst Birger Eriksen forhandlinger med tyskerne om overgivelse av Festningen. Den ble overgitt til tyskerne kl. 0900 den 10. april. Av angrepsstyrken på totalt 42 fly ble ett skutt ned og flere skade

Oscarsborg bombes av Luftwaffe

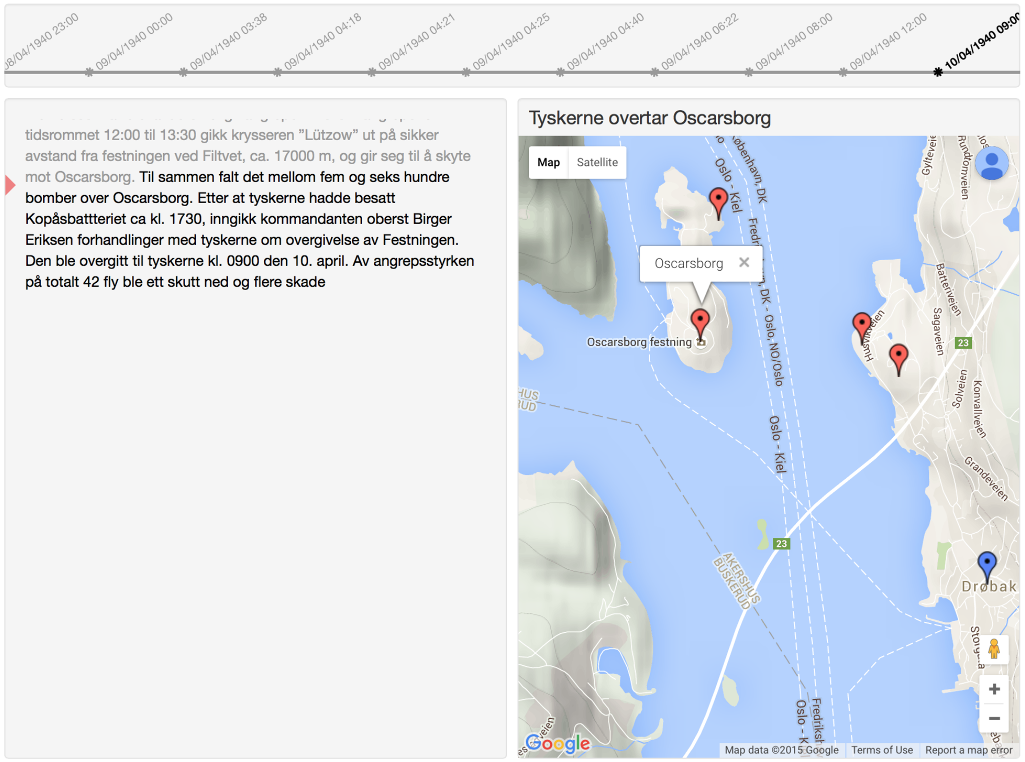
Figure A.12: Prototype result, screenshot #12



Seiersten, fire ved Håøya. I starten av angrepet var også fire maskingevær på taket til hovedbatteriet med i å besvare angrepet, men disse måtte evakuere tidlig i angrepet. Mellom angrepene i tidsrommet 12:00 til 13:30 gikk krysseren ”Lützow” ut på sikker avstand fra festningen ved Filtvet, ca. 17000 m, og gir seg til å skyte mot Oscarsborg. Til sammen falt det mellom fem og seks hundre bomber over Oscarsborg. Etter at tyskerne hadde besatt Kopåsbattteriet ca kl. 1730, inngikk kommandanten oberst Birger Eriksen forhandlinger med tyskerne om overgivelse av Festningen. Den ble overgitt til tyskerne kl. 0900 den 10. april. Av angrepsstyrken på totalt 42 fly ble ett skutt ned og flere skade

Oscarsborg bombes Lutzow

Figure A.13: Prototype result, screenshot #13



tidsrommet 12:00 til 13:30 gikk krysseren "Lützow" ut på sikker avstand fra festningen ved Filtvet, ca. 17000 m, og gir seg til å skyte mot Oscarsborg. Til sammen falt det mellom fem og seks hundre bomber over Oscarsborg. Etter at tyskerne hadde besatt Kopåsbattteriet ca kl. 1730, inngikk kommandanten oberst Birger Eriksen forhandlinger med tyskerne om overgivelse av Festningen. Den ble overgitt til tyskerne kl. 0900 den 10. april. Av angrepsstyrken på totalt 42 fly ble ett skutt ned og flere skade

Tyskerne overtar Oscarsborg

Figure A.14: Prototype result, screenshot #14

Bibliography

[1] ESRI. Mapping without limits, 2013. URL http://www.esri.com/. [Accessed: 2015-08-22].

[2] HBO. Viewers guide, 2011. URL http://viewers-guide.hbo.com/game-of-thrones/season-1/episode-1/map/. [Accessed: 2015-08-22].

[3] Gudmundur Helgason. Uboat.net - hms triton (n 15), 2015. URL http://uboat.net/allies/warships/ship/3484.html. [Accessed: 2015-09-15].

[4] Google Inc. Google maps, 2007. URL https://www.google.no/maps. [Accessed: 2015-08-26].

[5] The Kingsroadmap. An interactive journey through game of thrones season 1,2,3,4 and 5, 2013. URL http://www.directpackages.com/thekingsroadmap/. [Accessed: 2015-08-22].

[6] Inc. MapQuest. Mapquest, 2012. URL http://www.mapquest.com/. [Accessed: 2015-08-26].

[7] MetaCarta. Openlayers, 2005. URL http://openlayers.org/. [Accessed: 2015-08-26].

[8] Microsoft. Bing maps, 2010. URL https://www.bing.com/maps/. [Accessed: 2015-08-26].

[9] Microsoft. Learn about asp.net signalr, 2015. URL http://www.asp.net/signalr. [Accessed: 2015-11-15].

[10] Forsvarets Museer. 9.april 1940, 2015. URL http://forsvaretsmuseer.no/Oscarsborg/Festningens-historie/9.april-1940. [Accessed: 2015-09-15].

[11] Jakob Nielsen. Why you only need to test with 5 users, 2000. URL http://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/. [Accessed: 2015-11-18].

[12] LOTR Project. Visualizing tolkien's works on the web, 2012. URL http://lotrproject.com/map/. [Accessed: 2015-08-22].

[13] Reginald Rose. *Twelve Angry Men*. Penguin Classics, 2006. ISBN 0143104403.

[14] J.R.R Tolkien. *The Lord of the Rings*. Mariner Books, 2012. ISBN 0544003411.

[15] Steffen Eriksen Vegard Lundberg Holter. Map application, 2015. URL http://folk.ntnu.no/steffe/master/angrepNorge/. [Accessed: 2015-11-19].

[16] Janet Wagner. Top 10 mapping apis: Google maps, microsoft bing maps and mapquest, 2015. URL http://www.programmableweb.com/news/top-10-mapping-apis-google-maps-microsoft-bing-maps-and-mapquest/analysis/2015/02/23. [Accessed: 2015-09-10].

[17] Wikipedia. Blücher, 2015. URL https://no.wikipedia.org/wiki/%C2%ABBl%C3%BCcher%C2%BB. [Accessed: 2015-09-14].

[18] Wikipedia. Hvb pol iii, 2015. URL https://no.wikipedia.org/wiki/HVB_%C2%ABPol_III%C2%BB. [Accessed: 2015-09-14].

[19] Wikipedia. Oscarsborg festning, 2015. URL https://no.wikipedia.org/wiki/Oscarsborg_festning. [Accessed: 2015-09-14].

[20] Wikipedia. Slaget ved oscarsborg festning, 2015. URL https://no.wikipedia.org/wiki/Slaget_ved_Oscarsborg_festning. [Accessed: 2015-09-14].

[21] Wikipedia. Angrepet på norge i 1940, 2015. URL https://no.wikipedia.org/wiki/Angrepet_p%C3%A5_Norge_i_1940. [Accessed: 2015-09-14].