Linköping's university/Linköping University | IDA/Department of Computer and information Science Bachelor thesis 16 hp | Educational Program: Högskoleingenjör i Datateknik Spring term 2020 | 9-IDA/LITH-EX-G—20/077--SE

Developing a web-application for collecting conversations in Lab Rooms

Saleh Salim Jaffer Ali Mousa

Tutor, Erik Berglund
Examiner, Sahand Sadjadee

LINKÖPINGS UNIVERSITET

Copyright

Abstract

During each lab session at Linköping University, it is common that teachers available in the lab rooms are asked questions about different topics. Each conversation contains the asked question and the answer given to the question. Unfortunately, it is only the participants in the conversation who benefit from it even though there can be others who are interested in the content of it.

This thesis will therefore explain how to create a SPA web application which can be used by a lab assistant in the lab rooms to record each conversation and store it as text in a database. The system that is being designed and developed is a web-based speech-text system. A database is going to be used to store the text that we have converted from speech. These texts can be downloaded to a mobile device via a web application. This thesis will also describe how to make the web app as effective as possible by observing the performance metrics of the app.

Acknowledgement

We want to thank our supervisor for all the help.
And we want to thank everybody that have had anything to do with this work.

Table of Content

1. Introduction

Web applications have increased in popularity and use over the past decade. This is largely because nowadays, it is so easy to enrich their web applications with pictures, games, videos, and more. Along with bigger and nicer web applications follow a negative part with, namely the size of the web application. The consequence of greater Web applications are the negative impact on performance and the more performance is affected the longer it takes for the web application to load the page for the user, especially on for example, a bad network. This is not something that is coveted by users who over time have has grown to expect good performance from web applications. We will focus on creating a great web application with great performance.

1.1 Background

Computerized systems that communicate over the Internet are today a common, and growing solution in modern companies to streamline their enterprise. A common structure of such a web system is where the company's data is stored centrally in databases and then made available to the company's employees through software installed on a web server. This type of system uses a so-called Service Oriented Architecture (SOA), also known as "cloud computing", where the installed software on the server is called a web service and contains functionality to retrieve data from the databases.

Web services can also be used directly by other software systems to form large complex web systems that have access to the same data. [22] Hendrick writes in an article about the relevance of the SOA system: “The reality is that IT is during a very significant transition from software to services. This transition is occurring because of architectural, technological, and business model changes that are designed to expand and enhance the role of IT within the enterprise. This transition from software to services is the next step in the evolution of SOA and provides confirmation that the principles behind SOA are sound and remain relevant in light of the changes occurring in IT. "
The degree project involves developing an IT artifact consisting of such a web system as described above. The system will, in successful implementation, be used by LIU in another project where the need for the development of such a system exists.

1.2 Problem

The system to be designed and developed is a web-based speech recognition system. The system will be able to record and then convert the recordings to text. This will be used by the lab assistant during labs in Linköping's university. A database should be used to save the text.

These texts should be downloadable to a mobile device via a web service. The user should be able to save and delete a text. There will also be search features in the app where u can search for conversations based on what course the recording was made.

For the implementation of this project, knowledge needs to be obtained in areas such as:

- Database
- Performance metrics
- Web Services

1.3 Purpose and Reaseach Issues

The purpose of the thesis is to describe the development process of the system. It shall describe the course of how the knowledge needs are met through research in the relevant areas of the research questions. It also describes how the research results are used to select appropriate techniques and tools to create a design that can be implemented in the creation of the IT artifact in the best effective way.   issue:

○ How can a web-based speech recognition system be developed to reach higher performance?

The following visions is expected by Linköping university:

- SPA Web application that is as effective as possible
- Search engine that allows the user to search for the following Search Phrase, Course Code, and date
- An effective database
- An effective server
- Should have a Reliability and availability in case of slow internet connection.
- Mobile-view compatible

1.4 Knowledge Stakeholders

The main knowledge stakeholder is Linköping's university, since the thesis work is carried out under the auspices of them. The author of the project is Sahand Sadjadee, who also acted as project manager and has compiled the requirements specification for the system. Interest lies in the techniques and tools used during the development of the IT artifact. Upon successful implementation of the project, the system as a whole or parts can be used in future projects at Linköping's University.

1.5 Disposition

The second chapter includes the method section, where the research methods are presented. The third chapter contains the theory that the research work deals with. Chapter four presents the empiricism based on the research from Chapter Three. Chapter five is the concluding part of the essay where conclusions and reflections are made on how far the work has succeeded or not.

2. Theory and Related work

To get a better understanding of what can prevent a web application from having a good one performance, a deeper understanding of web development optimization must be gained. This chapter presents the techniques and design choices to be used to make one optimized web application. It includes a list of concepts containing a variety of tools and techniques relevant to the development of this type of web system. After reading the theory chapter, you as a reader should know that the question is both well formulated and relevant. This chapter will contain theory that is useful for the study we are going to do. This applies to both technology and method. We will focus on the Optimization and on the web testing of the web application.

2.1 Front-End Optimization

When talking about front-end optimization, we talk about a process of refining the website to get to render the pages faster, to make it more user-friendly so that the user does not have to wait. This process means to reduce the size of the application files, remove the unnecessary code and minimize the number of files that needs to be downloaded for the page to be displayed for the user. The optimization can be done in several ways, which technique or techniques are used varies from page to page. Below are the most common techniques presented that you use when you want to optimize the front-end.

2.2 HTTP Optimization

Every time you visit a website for the first time, the visitors must download all the resources instead of load them through a browsers cache. Reducing the number of HTTP requests is common techniques to get an optimized page. Every video, image, script, and stylesheet that is available results in a HTTP request from the browser to the server. As more request are being made the more time it takes for the page to render.
The loading time only gets worse the more users that use the page. Another reason why the HTTP request may slow down the performance of the page is that the files of the page is too large. The larger the files, the longer HTTP requests must be made. How to solve poor performance due to HTTP request can be done in many ways the most common method is to use tools such as Wireshark, to get a better overview over how many HTTP request are made and how long it takes [1][2].

2.3 Cache Optimization

The Cache is used to store information such as HTTP request between client and server.
With the optimization of the cache, you can reduce the data exchange that takes place between the client and the server. This can improve the performance of the web application. The downloaded components are stored in the browser cache and the browser reads from the cache to reduce HTTP

requests. [4] Expires header is an approach where the webserver communicates with the client and says that if used, leading to a reduction in HTTP requests. [5] [6]

2.4 Code Optimization

To be able to make a website in the first place, an HTML document is needed which is a language for describing the structure of a web page. HTML can be said to be the base of a website. To make the presentation better CSS is used, which is a language that describes the presentation style. When you then want to make the website a little more interactive with the users, JavaScript comes into the picture. All these languages are necessary to make a working website. But if no structure is followed and a lot of duplicate code is written then this will affect the performance of the website. Writing code with structure that is divided into several files is not just about doing it more readable but also to help the performance of the web page.

2.4.1 HTML Optimization

HTML [7] stands for Hypertext Markup Language and is a markup language. HTML is a critical part of the web application and its performance. When a web page is updated, HTML documents will be reloaded. If the documents are very large, then it will slow down the whole process of retrieving the data [8]. This is because the HTTP request are very long or many if there are many documents that need to be reloaded, as described in section 2.2. It is also good to think about where in the code you implement you CSS stylesheet and JavaScript. Including your CSS stylesheet at the top of the HTML document, in the header tag, facilitates progressive rendering. With the help of progressive rendering, you give the browser a head starts in downloading all the necessary pieces of the webpage. Without progressive rendering users can experience the loading time as tough as you only get a blank image while the webpage is loading [9].

2.4.2 CSS Optimization

CSS [7] stands for Cascading Style Sheets and is as previously mentioned to help with the presentation of the website. The Easiest way to optimize CSS is to minimize, do more compact and split the CSS files. This means deleting duplicates of code, delete white characters and divide the files into different stylesheets. Use of CSS expressions, that is JavaScript implemented in CSS property can be very heavy for the performance of the webpage when these expressions are run each time the page is scrolled up or down. Even when something as simple as moving the mouse the CSS expressions is being run thousands of times.

2.4.3 JavaScript Optimization

Almost all web applications use JavaScript to make the pages more dynamic and user-friendly.
Writing good JavaScript code is important not only to optimize the web page but also because the things the user does on the web page to get a smoother experience. Such as waiting for a slider to arrive when you press a button. The first step that should be done is to implement the JavaScript at the end of the HTML document. If the JavaScript were to be implemented at the top of the HTML document the download process will be blocked for the HTML and CSS element. This Results in a blank

page while everything is loading, instead of a page being loaded progressively. If you have too many packages and other code that must be downloaded it will be a slower experience for the user as JavaScript can delay the interactive part with the interface. This only gets worse with one bad network connection. This can be greatly reduced by using code splitting (See section 2.5) and compressing the files. [10]

2.5 Code Splitting

Code Splitting is a method to split all JavaScript's files into several small packages so one large package does not need to be downloaded. Reason to do this is to avoid downloading JavaScript which is not used to the current page displayed to the user. For example, if the user is sitting with the login page then you will only need the JavaScript to integrate with the page. But many modern sites do that when you are on the login page, they download the whole webpage and not only the login page. This is something that is not necessary at all and something that just slows down the performance of the web page. It is even worse especially for mobiles that have a little worse performance and a bad network connection. Code splitting can be done as follows [11].

- **Vendor Splitting:** Separates vendor code from the code that is contained in the application. This hinders the negative performance that may occur in case of invalid cache for returning visitors when either vendor or the ap code changes. This is something that should be implemented by every application.
- **Entry Point Splitting:** Separates code by looking at entry points, that is where scripts for various tools such as webpack or parcel start. Works best for applications that do not use client-side routing.
- **Dynamic splitting:** Is the separation of code where dynamic "import ()" is used. This is best to implement for single page applications.

2.6 Render-blocking

Render blocking is a problem where the rendering of the web page is blocked. Some of the most common errors are to include JavaScript in the header tag of the HTML document. What happens is that the rendering of the webpage is stopped to execute the JavaScript. This is not necessary; the script is something that can be downloaded after the page itself has finished loading. Therefore, it is best to include the JavaScript at the end of the body tag or in the footer tag. Same applies with CSS / stylesheets, if the disabled attribute is active only when the page is to be rendered does not download browser stylesheets. This attribute can be set to false after the webpage is preloaded to avoid render blocking.

2.7 Minify

Minify JavaScript means reducing the size of the JavaScript files and thus, speed up the rendering of the web page. To achieve this, there are two common techniques that are usually followed. The first technology is called Minification which is to remove all white characters and code that is not necessary, this to make a smaller JavaScript file. One compression tool can be used to help with this. The Second Data Technology compression means compressing the code using tools that use compression algorithms, some tools that are good for this are Gzip [10].

2.8 Web Performance

We need to look at the performance factors to be able to implement the application in the right way. To be able to answer our research question.
Our primary purpose with the testing is to find failures in the application. Web Performance testing generally can be divided into three types: stress testing, load testing and strength test. Each test using the same script, testing tool, and environment, but different testing time intervals [1].

2.8.1 Stress Test

Stress testing is a technique used to determine whether a system under study meets its performance objectives. It involves emulating the request patterns of real users of a system within a controlled environment. The workloads used in such tests, called synthetic workloads, are intended to mimic a range of workload conditions including those observed at real systems. Performance measurements such as user response times and server utilizations are collected during the tests and used to support sizing, capacity planning, and service level management exercises [2].

2.8.2 Load Test

Web application load testing is an important part of web performance engineering. Load testing measures the response time, throughput, and availability of a target website from a client's perspective (usually a web browser). Loading test is made by degrees increasing the load, testing the changes in system performance [4].

2.8.3 Strength Test

Strength test is a longer interval load test or stress test. Unlike other tests is that the weight-bearing or tension testing interval of only tens of seconds to maintain the strength test should be delayed a few hours or even days. Strength testing often finds some inexplicable errors. For example, memory leaks, that is, memories, rollback segments exist in the database transaction were not submitted, or have a cumulative impact on system resources, errors and so on [5].

3. Method

This chapter describes how the group has gone about developing the product. Here, the structure of the development is described, which includes the creation of the server, Database and User interface.

3.1 Implementation

We made a sketch of the system's database through ER diagrams and table sketches. The system we had outlined did not require any complicated database to work, which meant that the table sketch was simple to create. We used SQLite because it is the most used database engine in the world. SQLite is a C-language library that implements a small, fast, self-contained, high-reliability, full featured, SQL database engine. SQLite is included all mobile phones and most computers and comes bundled inside countless other applications that people use every day.[25]
We then started to create a code skeleton of the system to get the page navigation to work as we wanted and to get the correct file structure and structure overall before we started to implement all

the functionality. This also enabled us to work in parallel with the system to streamline work and quickly get a grasp of the system's functionality. We had a picture of what the system would technically be able to do, so we focused on implementing as much of the functional requirements as we could before we started on the graphic piece.
The web application was implemented using previous experiences from the course TDDD97 Web programming at Linköping University. The application was based on the same structure. Further information needed during implementation was obtained from guides on the website www.W3Schools.com. The website has documentation and tutorials for both Html, CSS, JavaScript, and database storage.
One of the requirements placed on the system was that it should be efficient and achieve great performance as possible, which meant that we spent a lot of time meeting these criteria. We focused on a simple flow in the system and its functionality. The visual communication that the system requires to achieve this was a big part of what we worked with after we had been able to implement the functional requirements. We designed the graphical interface as simple as possible to achieve great performance. Throughout the implementation phase, our focus was on what could be streamlined and refined in the code as well as the implementation to achieve the best performance.

3.1.1 Documentation

When making a project of larger scale it is important to have a good structure on the files and documents you create and use. Logs, programming files and design documents are some of the documents we have worked on jointly. Since we have been two people in our degree project, smooth file sharing has been extra important, and we have used our help Google Drive cloud service. Google Drive is a free Google service where you can create and share folders, Google documents, files, and drawings with a Google account. A very useful service in the tool is that you can sit at one time and edit in a document from different computers. This service has been a great asset to us throughout the project, from start to finish.

3.1.2 Choice of test environment

To check the performance of a test environment, built-in tools in browsers can be used. The most popular tools for this are FireFox Developer Tools available in Firefox and DevTools such as available in Google Chrome. To be able to answer which tool works best for just this purpose, a deep dive was performed in the various functions found in the tools.

3.2 Client

The client page in a web application is the visual in the browser that the user integrates with. The interface is made with HTML which is the global standard. The programming was made the structure of HTML while visual choices such as colors and sizes object in a separate CSS file. To make a web application dynamic, so that something happens if you press more buttons for example, you need to use a front-end program language. For this purpose, we have selected JavaScript. To communicate with the server side, the JavaScript object is used XMLhttprequest, which allows us to retrieve and send data to the server without having to load about the page.

3.2.1 Implementation In HTML

The only HTML-document that have been created is “static/index.html”, that contains two main views. The first needs to identify the document type for the web browser to translate it. It can simply be done using html tag like following:

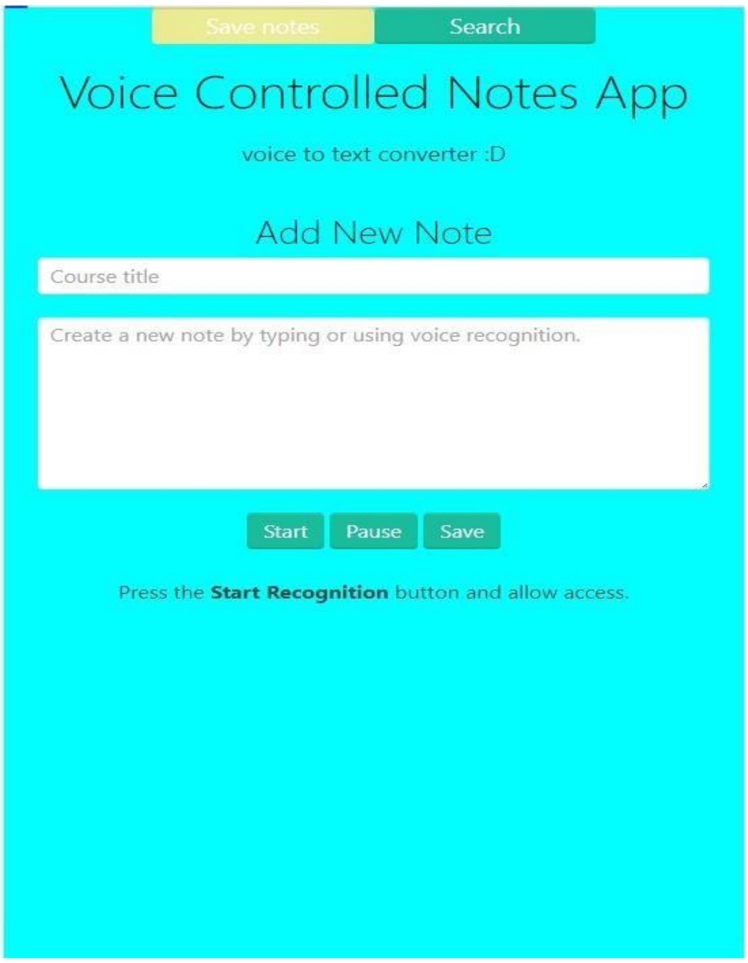
```
<!DOCTYPE html>
…
</html>
```

All html code lies between these two lines. The code starts directly after declaring the document type by using the head element. The head element is used to declare a title, document encoding and the location for the resources.

```
<head>
    •  Title
    •  Encoding
    •  Resources
</head>
```

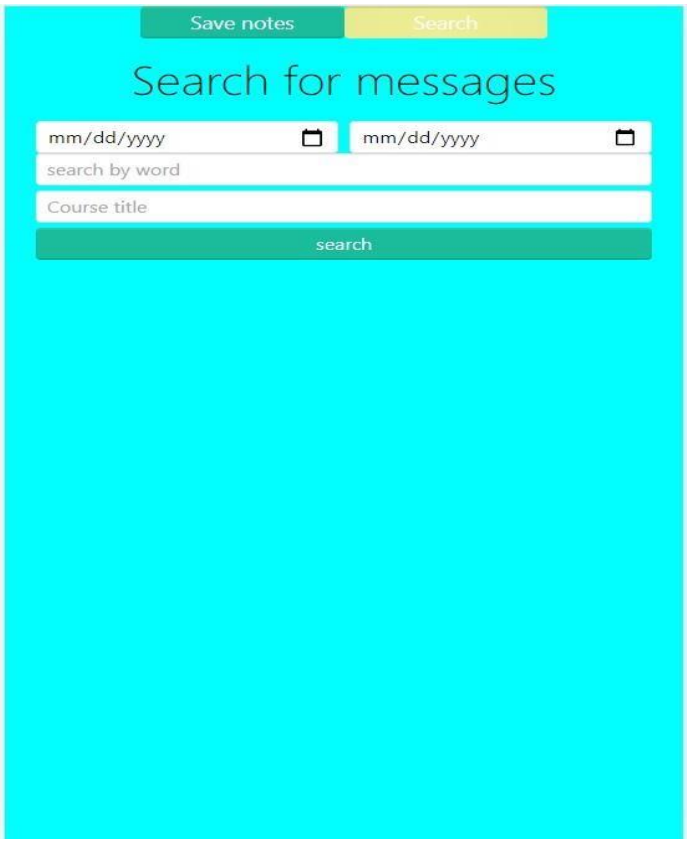
The head-element is then followed by the actual structure of the page (body-element). The body element declares the content of the page, the views, and tabs etc.
We have decided to make two different views that can be selected depending on what functions the user need. The views are simple div elements with specific content.

- Home-view: Has the necessary content that converts the speech to text and saves it.

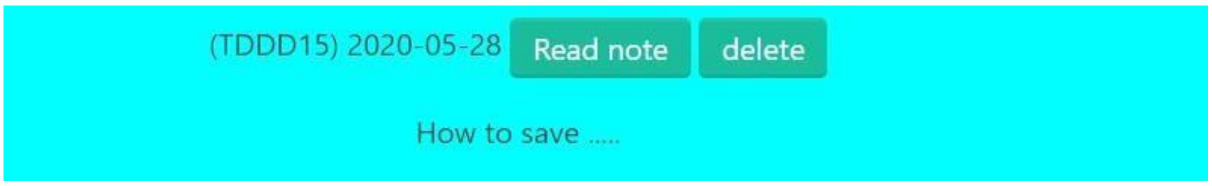

Save notes
Search
Voice Controlled Notes App
voice to text converter :D
Add New Note
Course title
Create a new note by typing or using voice recognition.
Start
Pause
Save
Press the **Start Recognition** button and allow access.

In this page (tab) the user can convert a speech to text or type in the text directly to store it. The user has also to type in course name for the text to be saved. The start button starts listening to the user and converting the speech, pause button pauses the listening process and the save button saves the conversation.

- Search-view: Has the necessary content to search the saved notes.



The second page enables the user to search for stored conversations. The user can search for notes by date, word and/or course name. The notes can be searched by combining the several search inputs or using only one.



When the user searches for notes, each note is shown like above. First the course name is shown followed by the date followed by two buttons, lastly the note is displayed in a new line. The buttons provide two functions, reading the note or delete it.

3.2.2 JavaScript File

The speech to text conversion occurs mainly on client side, and the code is created in “static/script.js”. The speech recognition API is limited to chrome and has a prefixed interface that is included in the code.

3.3 Server

When choosing programming languages on the server side, there were several good alternatives to choose from, Python, PHP, C++, and Java are some of the most popular back-end languages used today. After having compared the programming languages, Python was chosen. As a hallmark, Python has a very clear and readable code, the language is common so there are plenty of resources to learn and we both have previous experiences with it. To save data in our web application, a database manager was required. Given that the size of the company and the function of the product, we realized that the type of database manager would not have a major impact on the performance of the web application. We evaluated both the database managers SQLite and MySQL, of which we then decided on SQLite. SQLite is the most used database manager and was preferred over MySQL when making an application like ours.

3.3.1 Database

A little information about the database. The notes are being stored in a server after being converted to text. Active communication between the server and user interface is needed to be able to store the notes or delete them. The communication between the user interface and database occurs manly in the JavaScript file and is done by using XMLHttp-Requests.
A database contains different types of data that are in some way related to each other. The database consists of tables and the tables in turn consist of columns and rows. Each row is assigned one data type depending on what needs to be stored.
To work with the database, a connection must be established by a function.
Once the connection is established, the database can be used to store data and manipulate it by using some internal commands provided by the database itself like SELECT, INSERT, DELETE and other commands.

3.4 Measurements

The application has been measured on two devices, a Samsung galaxy S7 (android phone) and a custom-made (Windows 10 PC) with Ryzen 5 2600 (6 core 12 thread) CPU. Both devices have been provided with the same internet connection. The devices have different internet adapters, which in turn gives different speed despite having the same internet connection.
The application is run on the latest version of the chrome version 85.0.4183.102 for the PC and 85.0.4183.101 for the android phone. Google Chrome has built in tools that is used to measure the performance of the application.

We also used DevTools that is provided by google to complete the measurements. The measured values that we look for are the following:

- First Content Paint (FCP). This parameter checks how long it takes before the first the image or text is displayed on the website.

- Speed Index. This parameter measures how quickly the page content is displayed on the website below page load.

- Time to Interactive (TTI). This parameter checks how long it takes before the website is fully interactive.

- First Meaningful Paint (FMP). The parameter measures how fast the primary content appears on the page. Since this parameter is so like FCP, this parameter is not used.

- First CPU Idle (FCI). This parameter measures how long it takes a website to become minimally interactive. For example, when most of all UI elements are interactive.

- First Input Delay (FID). This parameter measures the time between the first interaction and the time it takes the browser to respond to this interaction.

The data also shows information about changes that can be implemented as potentially possible increase the performance of the website. Some of the changes that came after running tests on the web application was to eliminate render-blocking resources, Minify JavaScript, Pre connect to required origins. Below are methods presented to try to tackle these problems. Then a little deeper diving will take place where more general methods to improve the front-end performance.

3.4.1 Method for recommended performance techniques

To solve the problem of render-blocking resources, it will first be looked at if Inclusions with JavaScript and CSS templates and stylesheets are done correctly. It means that JavaScript is included at the end of the body tag or in the footer tag and that stylesheets have one disabled attribute that changes to true only when the web page has finished rendering [21]. What will be checked here is to check in the JavaScript and if there is a possibility for to make the code smaller by minification. [16]
To know if Pre connect to required origins is worth implementing, each must resource priority is carefully checked. This is to avoid problems such as wasted time with one connection or to download a resource twice double-fetching.

3.4.2 Code optimization

After checking out tips on improvements released by DevTools, we checked the web application on the more general techniques to increase the performance of web applications. This then means that the code must be examined for potential optimization, if cache optimization is used and if code splitting is implemented. What to look for is how the passcode uses the methods presented in section 2.4 "Optimizing codes".
When it comes to Cache optimization, it should be examined whether the expire header is used and if not used, how can this be implemented. Finally, an investigation into code splitting will be done. It will be checked if it is used and if so how.

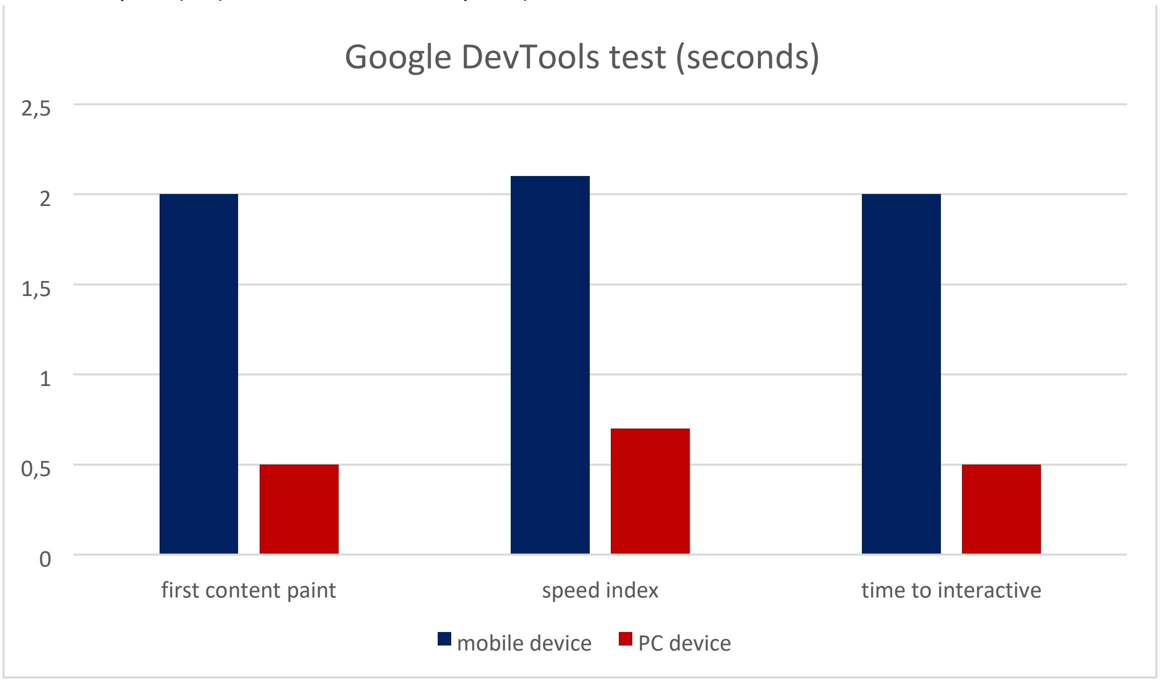
4. Result

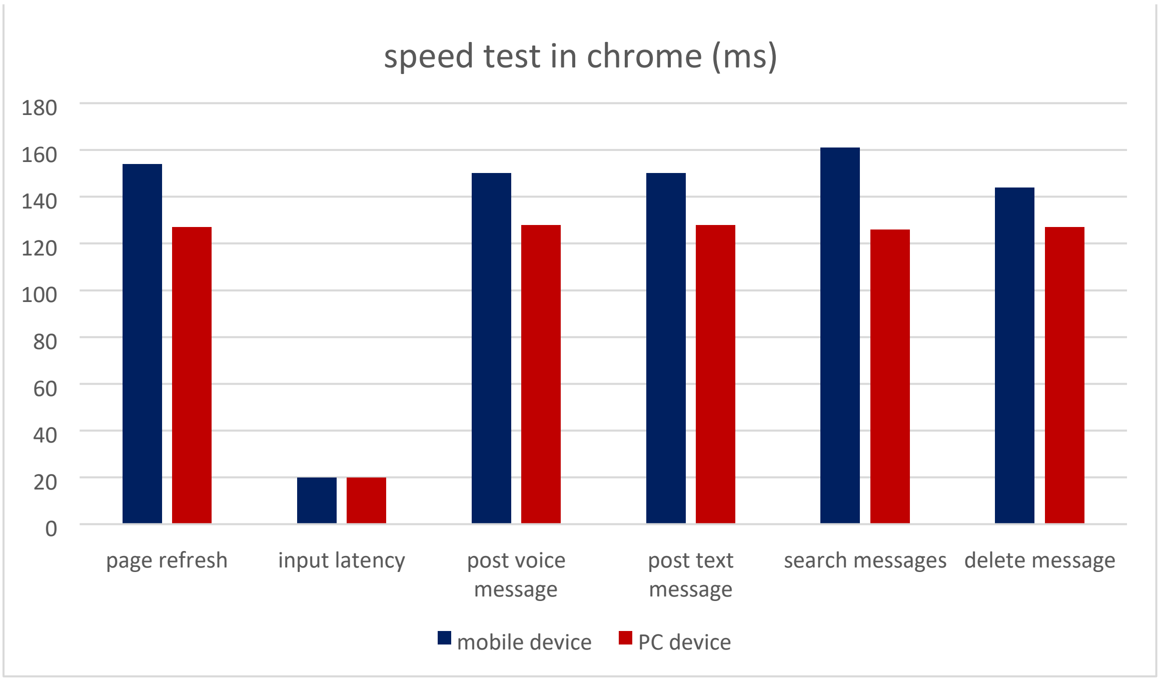
This chapter describes the results that came up by performing the methods presented in Chapter 3. The results will be presented in sections, one section for each technology that has examined.

4.1 Performance Metrics

We have produced a few data that shows the difference between the speed from the application while it performs some of the functions in the app. Both with a mobile device and a computer. We in this section measure load speed.[4]

○ Mobile-device (same internet connection) PC-device (235 MB/S Internet speed) (116 MB/S Internet speed)





4.2 Speech Conversation Test

xx

Here we want to try how our speech-text engine was working so we performed a test where we see how well the engine converts the words you say to text. We tried to test the accuracy of 10 different sentences in both English and in Swedish.

The following sentences were spoken in Swedish

- Jag behöver hjälp.
- Hur kan man klara den här uppgiften.
- Kan du förklara vad den här innebär.
- Jag undrar vad menas med detta.
- Vilka krav har den här uppgiften.
- Vi glömde våra väskor på busstationen.
- Jag tycker det är svårt.
- Hur ska ja tillväga.
- För länge sedan fanns en stad som heter Linköping.
- I förrgår såg jag min favorit och som börjar på TV.

The Application converted these Swedish sentences as following:

- Jag behöver hjälp.
- Hur kan man klara av den häruppgiften?
- Kan du förklara vad det här innebär?
- Jag undrar vad som menas med detta?
- Vilka krav har den här uppgiften?
- Vi glömde våra väskor på busstationen.
- Jag tycker det är svårt. o Hur ska ja gå tillväga?
- För länge sen fanns det en stad som heter Linköping.
- I förrgår såg jag min favorit popsångerska på tv.

-

The same sentences where translated to English to measure accuracy.

- I need help.
- How can I finish this task?
- Can you explain what this means?
- What conditions does this task have?
- We forgot our bags at the bus station.
- I think it is hard. o How should I approach it?
- A long time ago there was a city called Linköping.
- The day before yesterday I watched my favorite pop singer on TV.

The Application converted these English sentences as following:

- I need help o How can I Finish This task?
- Can you explain what this means?
- What conditions Does this has had.
- Reaper got our bags at the bus station.
- I think it is hard.
- How should I approach it? o

- a long time ago I was a City called Linköping.
- The Day Before Yesterday I watch my favorite pop singer on TV.

The following misunderstanding occurred in Swedish:

What was said	What the application converted to.
Kan du förklara vad det här innebär?	Kan du förklara vad den här innebär.
Hur ska ja gå tillväga?	Hur ska ja tillväga.
I förrgårsåg jag min favorit popsångerska på tv.	I förrgår såg jag min favorit och som börjar på TV.

And in English the following errors occurred:

What conditions does this task have?	What conditions Does this has had.
We forgot our bags at the bus station	Reaper got our bags at the bus station.

4.3 Sound Level Test

As we described we are going to test the decibel level of your voice for the microphone to be able to recognize the words that are coming out of your mouth.

conversation	
30 db whisper	Voice not recognized
35 db	Voice not recognized
40 db	Voice not recognized
45 db	Voice not recognized
50 db (conversation with low voice)	Voice recognized
55 db	Voice recognized
60 db (normal conversation)	Voice recognized
>60 db	Voice recognized

4.4 Background Sound

We started a recording of a conversation in a speaker in the background and tried to see how well the application was able to convert our conversation without it being affected by the background noise.

Background	
30 db whisper	No effect
35 db	No effect
40 db	Small disruption
45 db	Small disruption
50 db (conversation with low voice)	Noticeable disruption
55 db	High disruption
60 db (normal conversation)	Total disruption
>60 db	Total disruption

4.5 Sound Distance Test

In previous test we focused on the decibel level for how the background noise effects the application. But how far away can you stand from the mobile device or the computer for the application to be able to understand you. We used our mobile device to record and started to see how far away we were able to stand until the application started to misunderstand us.

Voice recognition at normal conversation (60 db)	
180 cm	Voice not recognized
175 cm	Small recognition
170 cm	Small recognition
165 cm	Mostly recognized
160 cm	Recognized
155 cm	Recognized
150 cm	Recognized
>145 cm	Recognized

5. Discussion

This project resulted in a fully functional web-application that converts speech to text and stores the conversation in a database. The application gives several options to search for saved notes. The application is running on Heroku which is an online platform that made it possible to run the application and operate in the cloud entirely.
First content paint is the first metric measured, the time for first content paint is the time it takes to render a defined object. The object could be anything defined such as text or an image. In this case there is significant difference between the desktop and mobile version of the application. The desktop version outperformed the mobile version by 1.1 seconds and that is mainly because of the hardware of the two devices. The desktop has more powerful CPU and GPU resulting shorter loading time for the content.
We have also measured the speed index of the two versions of the application. The speed index is the average time taken for visible objects to be displayed. The results are as expected being much lower in the desktop version (1.4 seconds faster than mobile version), due to same reasons (faster hardware).
The application showed slight time difference when calling the server requests. The results show that the desktop version is always faster than the mobile version although the desktop had much slower internet speed compared to the mobile (the phones internet adapter is better although the devices were connected to same network). The differences are negligible and will not make any noticeable difference for the users, and they manly exist because of the difference in hardware in the tested devices as mentioned previously.

5.1 Method

There are a lot of ways speech recognition can be executed, the method we chose only works in one web browser (Google Chrome). This is a huge drawback because some users may prefer to use other browsers but for this application, they will be forced to use Chrome. Additional measurements could also have been taken to compare how the application would perform on several web browsers, however this is not possible in our case.
The tests have been made on mainly two devices, a mobile phone, and a PC. We tried to make all variables beside the devices as equal as possible, but a few things could not be controlled to give equal advantage for both devices. The most obvious difference is in the internet speed the mobile phone had more than double the internet speed. The study would have been much more specific if the internet speed could have been then same.
As previously mentioned, the results are performed on two devices that represent two categories, however more test subjects would give a better and more scientific view on how the application would act on different devices. The cellphone category was represented by an android phone and the desktops were represented by a windows computer. If the study would have included IOS and mac devices, then we would have made a clearer point of view on the performance and we might be able to conclude what environment the application prefers.

5.2 Discussion

As we can see in the result the application starts to get a disturbance when you reach a little below normal conversation level. It starts to pick up the words from the background as well. This is not something good but if you want the application to pick up your conversation then this is a side effect.

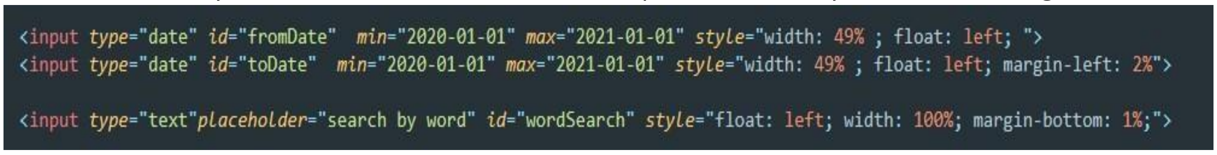
There should be a noise filter programmed but this is not something that was in the requirement and is hard to implement.

The application is accurate in converting simple sentences and it has been showing some misunderstanding of some words in both Swedish and English. This problem could be minimized by using a spelling correction tool to correct all words before converting them, however this method will slow down the conversion process especially when the application will be used for several languages because every language will need a specific spelling control tool and the words need to be inspected by them.
The application has been shown to recognize low voices and converting the speech around 50 dB sound level. However, it gets disturbed when the background noise is around 50 db.
The application can convert the sound from 145 cm and less distance from the phones microphone making it easy for users to speak to the application while holding the phone comfortably without having to take in consideration the distance to the phone.

The conversion of the sound does not have to be as we implemented in the frontend, it could also be done in the backend. To convert the sound in the backend we must send over the sound instead of the text to the server. However, the sound is always much larger in size compared to text, for example a 150 000-word text file is only 0.8 MB in size and 18 MB in size as a sound file [26]. This would make the application much slower especially with long speeches and therefore we decided it is better to convert the speech before sending it over the network.
It is obvious that the phone has a better internet adapter and has higher internet speed over the same network connection compared to the desktop in our case. Although the internet speed is an important factor in many applications, it is not as important as the previously mentioned factors due to the small sized packages sent over the network. We only send text over the network to be handled at the backend which does not need many bytes. However, the internet speed would be more important if we would send the sound instead of text to the backend.

To further improve the application, we could have:

• Separate the page content into separate .CSS, .js and .html files so that each is separate file can be saved by the browser in its cache. For example, we currently use the following code:

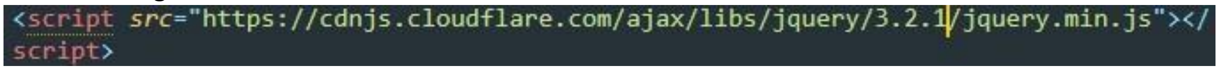


```
<input type="date" id="fromDate"  min="2020-01-01" max="2021-01-01" style="width: 49% ; float: left; ">
<input type="date" id="toDate"  min="2020-01-01" max="2021-01-01" style="width: 49% ; float: left; margin-left: 2%">

<input type="text"placeholder="search by word" id="wordSearch" style="float: left; width: 100%; margin-bottom: 1%;">
```

However, if we instead put all stylings in the CSS files, we could see more improvements.

• Minimize the total number of style templates and JavaScript library files as included in the page. This because that the browser only makes two file requests included in the page. Each file that the browser must wait for each request to be completed will create delays.
We use the following libraries in our application, they are 6 in total and the browser needs to make three calls to get all of them.



```
<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.2.1/jquery.min.js"></
script>
```

The performance would have been better if we could reduce the number of libraries, but we found all these libraries important for the application

- Consider using gzip compression on the data being sent. This would minimize the size of the traffic being sent over the network. However, the text being sent is already small sized and using the gzip would probably improve by an unnoticeable margin. It might even be more time consuming to compress the data instead of sending it directly, especially when it is relatively small.

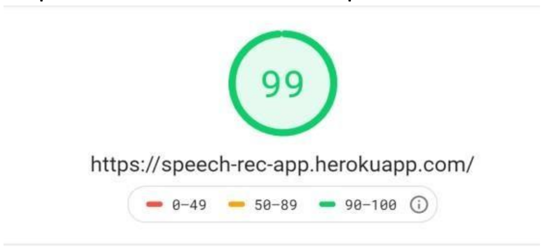
The application scored very high in benchmarks and it is being bottlenecked by the hardware of the device it runs on and further improvements in the code will be unnoticeable for users.

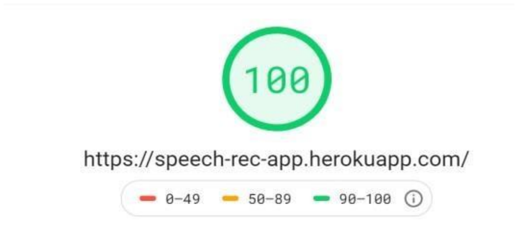
5.3 Source Criticism

In an area that is constantly evolving in the form of new libraries, technologies, languages, etc., it seems that the focus on performance is lagging. This was noticed when scientific articles were to be found. One of the bigger challenges was finding scientific articles where the web application is in focus. After searching a lot for good scientific articles on performance it was clear that this area was not something that people focus their research on. This feels a little strange then the importance of the performance of a web application is one of the most important factors for it to be useful. Only a few articles were found while the majority found were in other languages which made it useless for us in our study. This in turn led us turned to Google Developers a lot. Google Developers is Google's site for devtools, APIs and similar technical areas. There is also documentation on how these tools should be used as well as forums where the tools are discussed. With a little deeper diving would one probably finds even more scientific articles that could have been useful. But thank you whether the prevailing situation in the world as well as any other factors that were beyond our control were time limited.

6. Conclusion

In this article a web application has been implemented in flask and is being deployed to Heroku online server. Many performance tests have been performed and with the help of DevTools and chrome (provided by google). The results have been presented to help understand the application works and the possibilities for further improvements have been discussed.





The performance test tool assigned a very high score to the web application (99% for the mobile version and 10% for the desktop version). The results show that the application is very fast and further improvements can be made, however, the improvements will not be noticeable for the user.

7. Reference List

1. LSinha Neeta, Aror Poonam.” Effects of growing mobile usage at workplace and its impact on work productivity- A detailed analysis”, 2005

2. B. Cao, et al."The solution of web font-end performance optimization," 2017 10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), Shanghai, 2017, pp. 1-5, doi: 10.1109/CISP-BMEI.2017.8302083.

3. Cassone, G., et al. "Web Performance Testing and Measurement: a complete approach." *CMG ITALIA 2001 and CMG USA 2001 conference proceedings*. 2001.

4. Jeff Posnick, Ilya Grigorik, “Prevent unnecessary network requests with the HTTP Cache”, Apr. 17, 2020. web.dev/http-cache/.

5. GTmetrix, “Yslow: Add expires Header”, gtmetrix.com/add-expires-headers.html.

6. Iliev, et al. "Front end optimization methods and their effect." 2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). IEEE, 2014. doi: 10.1109/MIPRO.2014.6859613

7. Rolf Staflin. HTML och CSS boken. Pagina Förlags AB, 2008. ISBN: 978-91-636- 0939-82

8. J. Skansholm, Java Direkt. Lund, Sweden: Studentlitteratur, 2003.

9. B. Cao, et al."The solution of web font-end performance optimization," 2017 10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), Shanghai, 2017, pp. 1-5, doi: 10.1109/CISP-BMEI.2017.8302083.

10. Houssein Djirdeh, “Reduce JavaScript Payloads with Code Splitting”, Sep. 19, 2018, web.dev/reduce-javascript-payloads-with-code-splitting/

11. Ports, Dan R. K. et al. “Transactional Consistency and Automatic Management in an Application Data Cache.” OSDI (2010)

12. Rolf Staflin. HTML och CSS boken. Pagina Förlags AB, 2008. ISBN: 978-91-636- 0939-82.

13. Iliev, et al. "Front end optimization methods and their effect." 2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). IEEE, 2014. doi: 10.1109/MIPRO.2014.6859613

14. Remakanth, et al.” A Survey on Performance Testing Approaches of Web Application and Importance of WAN Simulation in Performance Testing”. International Journal on Computer Science and Engineering. 4. 2012,

15. P. Ling, "Based on web application front-end performance optimization, Proceedings of 2011 International Conference on Electronic & Mechanical Engineering and Information Technology, Harbin, 2011, pp. 234-237, doi: 10.1109/EMEIT.2011.6022862.

16. Jovanovski, et al. "Critical CSS Rules—Decreasing time to first render by inlining CSS rules for over-the-fold elements." Post proceedings of 2016 Seminar on Advanced Techniques and Tools for Software Evolution (SATToSE)”. 2016

17. Garrett, Jesse James. "Ajax: A new approach to web applications, 2005." *Adaptive Path Inc* (2010).