

# **Developing a Python based web scraper**

A study on the development of a web scraper for TimeEdit.

Pontus Andersson

Thesis – Examensarbete  
Main field of study: Computer Engineering  
Credits: 300  
Semester/Year: Summer 2021  
Supervisor: Magnus Eriksson  
Examiner: Patrik Österberg  
Course code/registration number: DT099G

## **Abstract**

The concept of scraping the web is not new, however, with modern programming languages it is possible to build web scrapers that can collect unstructured data and save this in a structured way. TimeEdit, a scheduling platform used by Mid Sweden University, has no feasible way to count how many hours has been scheduled at any given week to a specific course, student, or professor. The goal of this thesis is to build a python-based web scraper that collects data from TimeEdit and saves this in a structured manner. Users can then upload this text file to a dynamic website where it is extracted from the file and saved into a predetermined database and unique to that user. The user can then get this data presented in a fast, efficient, and user-friendly way. This platform is developed and evaluated with the resulting platform being a good and fast way to scan a TimeEdit schedule and evaluate the extracted data. With the platform built future work is recommended to make it a finishes product ready for live use by all types of users.

Keywords: Web Scrper, Dynamic Website, Unique Users, Structured data

## Sammanfattning

I en värld där alltmer information lagras på internet är det svårt för en vanlig användare att hänga med. Även när informationen finns tillgänglig på en och samma hemsida kan den hemsidan sakna funktioner eller vara svår att läsa av. Idén bakom att skrapa hemsidor, tidningar eller spel på information är inte ny och detta examensarbete fokuserar på att bygga en web scraper med tillhörande hemsida där användare kan ladda upp sitt schema skrapat från TimeEdit. Hemsidan ska sedan presentera denna skrapade data på ett visuellt tilltalande sett. När system är färdigutvecklade utvärderas dem för att se om examensarbetets mål har uppnåtts samt om systemen har förbättrat det befintliga sättet att hantera schemaläggning i TimeEdit hos lärare och studenter. I sammanfattningen finns sedan framtida forskning och arbeten presenterat.

Nyckelord: Web scraping, TimeEdit.

## Acknowledgements

I would like to acknowledge Magnus Eriksson on Mid Sweden University that provided me with the idea of the project. Magnus gave me important information regarding how the system works today and why it is an issue, and without this information it would have been hard to build a case. Outside this Magnus was also my mentor during the thesis.

# Table of Contents

<b>Abstract .....</b>	<b>1</b>
<b>Sammanfattning.....</b>	<b>2</b>
<b>Acknowledgements .....</b>	<b>3</b>
<b>1 Introduction .....</b>	<b>1</b>
1.1 Background and problem motivation.....	1
1.2 Overall aim.....	2
1.3 Limitations .....	2
1.4 Concrete and verifiable goals / Detailed problem statement .....	3
1.5 Scope .....	3
<b>2 Theory .....</b>	<b>4</b>
2.1 Programming languages.....	4
2.1.1 Web scraping.....	4
2.1.2 Python .....	5
2.1.3 Beautiful soup library .....	6
2.1.4 Alternatives to Beautiful Soup.....	6
2.1.5 PHP and SQL .....	6
2.1.6 PhpMyAdmin .....	7
2.1.7 HTML and CSS .....	7
2.2 One.com – hosting service .....	8
2.3 Visual Studio Code .....	8
2.4 User Stories .....	8
2.5 Requirement Specifications .....	8
2.6 Function analysis .....	9
2.7 Related works .....	9
2.7.1 An Overview on Web Scraping Techniques and Tools .....	9
2.7.2 A comparative study on web scraping.....	9
<b>3 Methodology / Model .....</b>	<b>11</b>
3.1 Time planning .....	11
3.2 Development process .....	11
3.2.1 Pre-study and data collection .....	11
3.2.2 Requirement specification and user stories .....	12
3.2.3 Functional analysis.....	12
3.2.4 Ethical Aspects, GDPR.....	12
3.2.5 Environmental Aspects.....	12
3.2.6 Traditional development.....	13
3.2.7 Python .....	13
3.2.8 Beautiful Soup.....	13

3.3	Evaluation .....	13
3.4	Presentation .....	14
<b>4</b>	<b>Design and Implementation .....</b>	<b>15</b>
4.1	Development .....	15
4.1.1	Pre study and Data Collection.....	15
4.1.2	Process of the platform .....	15
4.1.3	Information gathering.....	16
4.2	Requirement specification .....	17
4.3	Function analysis .....	19
4.4	Ethical Aspects, GDPR .....	20
4.5	Environmental Aspects .....	20
4.6	Development .....	20
4.6.1	Analyzing TimeEdit .....	20
4.6.2	Building a python-based web scraper .....	23
4.6.3	Building a dynamic website.....	25
4.7	Testing the efficiency .....	28
4.7.1	Testing the developed system .....	28
4.7.2	Testing the old system .....	29
<b>5</b>	<b>Results .....</b>	<b>30</b>
5.1	Web scraper .....	30
5.2	Dynamic website.....	31
5.3	Analysis of results and developed program.....	33
5.3.1	Analysis of the web scraper .....	33
5.3.2	Analysis of the website .....	34
5.4	Results from evaluation of efficiency .....	35
5.4.1	Results from developed system.....	35
5.4.2	Results from the old system.....	36
<b>6</b>	<b>Conclusions / Discussion .....</b>	<b>37</b>
6.1	Methods chosen.....	37
6.2	Working with Python.....	37
6.3	Concrete and verifiable goals.....	38
6.3.1	Summary of goals.....	39
6.4	Ethical and societal aspects.....	39
6.5	Future work .....	40
6.6	Problems during development .....	41
	<b>References .....</b>	<b>42</b>

# Terminology

## Acronyms/Abbreviations

HTML	Hypertext Markup Language
CSS	Cascading Style Sheet
PHP	PHP: Hypertext Preprocessor+
SQL	Structured Query Language

# 1 Introduction

TimeEdit is a scheduling platform built by the company Evolvera. When the platform was originally built in 1992 it would mainly be used by the department of education at the university of Gothenburg for scheduling [1]. Since then, this platform has expanded and is now used by hundreds of different universities, some of them being Mid Sweden University and Gothenburg University.

The platform is used by universities around the world to schedule classes, exams, and meetings.

TimeEdit is mainly used by Mid Sweden University's schedulers to help manage and schedule students and teachers, exams, and laboratory sessions that both students and teachers attend. The system is intended to work good for presenting schedules for both students and teachers and is connected to both general and individually connected schedules that can be reached by the students or teacher's unique logins.

The system is not designed to present information regarding number of hours scheduled between students and teachers, how many hours each course has scheduled for any given month or if a teacher has too many hours scheduled, something that can become a problem on big institutions with a lot of people connected

## 1.1 Background and problem motivation

With the functionality that is available on TimeEdit today it is difficult for students, teachers, and study directors to follow up on whether the allocated working time is in relation to the number of lessons that each teacher gives. Since there is no automated system for this, students and teacher's need to hand-count the number of lessons scheduled for each period or term, a very time-consuming task.

In a university system where this needs to be done every period and term, every year, it can be difficult for teachers to detect errors. A common problem is that teachers have too many hours scheduled for certain weeks or periods so the total amount of hours for the course is correct, but a large portion of those hours are concentrated to specific days or weeks. This can lead to stress and overload.



This does not only affect teachers, but also students. Some periods have three courses running simultaneously that contest for time. As a student, especially first term student, it can be difficult to get an overview of how much time is scheduled each week during the entire period. This can lead to less planning and more stress.

A system that has high usability and efficiency, that is cheap to use and that is scalable is therefore needed to help students and teachers alike with planning and structure. The system needs to use existing methods and systems to work.

## **1.2 Overall aim**

The overall purpose and goal of this degree is to develop a system for TimeEdit's scheduling platform that will meet the need for teachers and students to be able to review the number of hours that each teacher and student has been scheduled for any period. This information then needs to be presented on a website with high usability, users-friendliness, and efficiency. The system needs to be scalable so hundreds of students and teachers can use it.

A python-based web scraper will be developed to scrape all the available information from TimeEdit. A dynamic and scalable website will be developed where the user can upload the web scraped information. This information will then be extracted, cleansed, and uploaded to a database. The user will be able to review the uploaded data that is being presented in a user-friendly way.

The thesis will conclude if the new system is more efficient than the old one and draw conclusions from that. There will be less focus on added functionality and further development.

## **1.3 Limitations**

This degree project at Mid Sweden University has a time limit in the form of 15 point, corresponding to 400 working hours. Because of this time constraint the project will be limited to development of both a Python based web scraper and the associated website and evaluation on if the new system is more efficient than the old one. There will be close to no focus on further developments of the website or web-scraper as this is beyond the 400-hour limit.

## 1.4 Concrete and verifiable goals / Detailed problem statement

With this project the following verifiable goals have been set:

1. Build a Python based web scraper that can scrape information from TimeEdit's scheduling platform.
2. Build a dynamic and scalable website where the data can be uploaded by the user to a database.
3. Build functions to sort and present the data in a user-friendly way for every unique user.
4. Implement all three parts of the system so they work together.
5. Evaluate if the new system is more efficient than the old one.

*5b: Evaluate the number of clicks needed for the user to display data.*

## 1.5 Scope

This project will only focus on building the platform needed to reach the goals. No hardware will be developed and only TimeEdit will be scraped. The website will only work with the scraped data from this web scraper and it will only do what it needs to reach the goals.

Handling scraped data outside of the goals of this project will not be a part of the programming, and the web scraper will only be able to handle TimeEdit. The website will only be able to handle said data from that specific webscraper.

## 2 Theory

In this chapter all theory related subjects will be presented together with articles that are related to this work.

### 2.1 Programming languages

During this project multiple different programming languages will be used to build the entire pipeline from scraping parsed data from TimeEdit to displaying this in a readable way for the user on a website. During the chapter 2.1.x the languages used, and more importantly, why they are used will be outlined. For details about how the languages are used to build the functions, see chapter three, method, and chapter four, implementation.

#### 2.1.1 Web scraping

The concept of gathering data from the internet is not new. With approximately 1.2 billion websites in the world as of May 2021 and more than 4.5 billion active internet users the amount of information that exists can be hard to find and structure in a meaningful way. [2][3]

The term web scraping comes from gathering data by “*scraping*” the information from a website. It is, in theory, possible to scrape more things than websites and things such as documents and games could be possible candidates as well, however, most scraping is usually done towards websites.

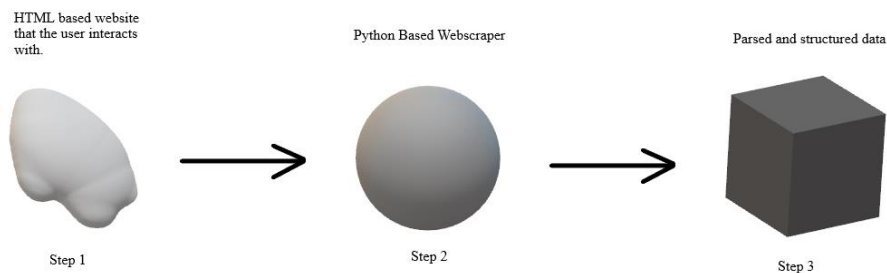
Building a web scraper is not the same as using a pre-defined API from websites. Examples of this could be Twitter and Wikipedia. The Twitter API lets users analyze already organized, cleansed and well-formatted data, something that is much more effective than building your own web scraper for the same task. The API often gives users the tools to choose and change what type of data they want to gather. [4]

Most websites, like TimeEdit that this project focuses on, do not have well documented and well-developed API tools that the users can utilize to gather the data, and thus, a “*web scraper*” needs to be built.

A web scraper, in short, is a program built to collect data directly from the source code of the website, both HTML, CSS or other information on

the website, and then parse the scraped data so the developer can choose what parts of data, information or text that should be extracted.

A benefit and downside of web scraping is that there is no predetermined solution for all websites. There is currently no feasible way to build a web scraper that works for all types of websites. For every website that the developer wants to scrape adjustments to the code needs to be done. This is because the scraper uses classes and names from each specific website to collect and parse data. With a well-developed scraper where good programming practices has been used it is often possible to adjust the code to work with other websites without having to build a new web scraper from scratch.



*Figure 1: Basic pipeline of a web scraper*

### 2.1.2 Python

Python, at the time of writing this article, has quickly become one of the most popular programming languages and was as of November 2020 ranked as the second most popular programming language after C. [5] Python was first released in 1991 but has since been re-worked and re-done, the biggest update being in 2008 when Python 3.0 was launched.

Python is a strongly typed language as the compiler keeps track of what types of variables you use and what type of data you are working with, and typing errors are prevented by the compiler during runtime. Despite this, Python as a language do allow variables to change names and is not as strict as other strongly typed programming languages such as Perl. [6]

It is an object-oriented programming language that is very well known for using significant indentation. This concept forces developers to write

very clean, easy to read and logical code and, because of this, it works well for big projects. Python has an extensive standard library with many different tools, as BeautifulSoup. [7]

### **2.1.3 Beautiful soup library**

Beautiful Soup is the most well used library to scrape and parse data from websites. According to the developers of the library it parses anything that it is given. BeautifulSoup does this using simple methods and pythonic idioms to build a parse tree that can be navigated and searched. The upside of using BeautifulSoup is that it converts parsed data to UTF-8, something that is well used on the internet. [8]

The web scraper in this project was built using the tools available in the BeautifulSoup library, and all the documentation regarding the library can be found under source 8.

### **2.1.4 Alternatives to BeautifulSoup**

There are several different alternatives to the BeautifulSoup library, with the biggest one being lxml. The libraries work in a similar way with similar functions. Historically lxml was a faster parser than BeautifulSoup, hence, it was used when speed was a vital factor and BeautifulSoup was used with bigger documents that lacked structure. [9]

Today, BeautifulSoup uses the lxml parser and therefore both libraries work similarly.

Besides lxml there are other alternatives to BeautifulSoup as well, such as Selenium and Scrapy where Selenium has some extra tools to work with websites whilst Scrapy works more as a framework rather than a library. [9]

### **2.1.5 PHP and SQL**

PHP is a modern and well used general-purpose scripting language that was developed to build more advanced websites. PHP stands for PHP: Hypertext Preprocessor and is developed as an open-source language. [10]

What makes PHP strong is that it executes server-side code on the website that is then displayed as normal HTML for the user. This makes it possible to build much more competent platforms where it is possible to have unique users, save information, and start sessions for a more

dynamic experience that changes based on what the developer wants the user to see.

PHP was chosen over JavaScript or other scripting languages for this project as the goal is to build a scalable and dynamic website that changes based on what user is online, something that is possible when building with PHP in combination with SQL.

SQL is the backbone of the website and stands for Structured Query Language. SQL is the tool used to communicate between the website that is built using PHP, HTML and CSS with the database where all the information about the user is being stored. SQL is the most used database management system, and it works using commands such as insert, update, delete, create, and drop. Commands used to remove and insert information from the website into the database. [11]

#### **2.1.6 PhpMyAdmin**

phpMyAdmin is the tool used to create, handle, and store the database. As a tool it helps developers build a database using a visual interface. From phpMyAdmin developers can create, drop, rename, or alter the databases that are being used. It is possible to import data directly into the database and that data can then be shown to users through the website using PHP and MYSQL. [12]

phpMyAdmin was used for this project because of the user-friendly interface and ease of use when working with big amounts of user stored data.

#### **2.1.7 HTML and CSS**

HTML, short for HyperText Markup Language, is the standard language used to build websites that are displayed for the user. HTML is a way to build structured websites that is rendered by the browser.

CSS, short for Cascading Style Sheets, is a style sheet language that is used to design the website. The CSS is used to design the elements and classes that developers use on the website. It is here that the developer implements everything from the design of a box to the color of text.

## **2.2 One.com – hosting service**

This project was done during the covid-19 pandemic which meant that no user testing could be done in person. To work around this problem the entire website was built towards the hosting service one.com.

One.com offers tools to upload and download files and edit the source code directly in the control panel. When working with a hosting service the website is always live and iterative work can be done whilst getting user feedback along the way.

During a non-pandemic development localhost can, and should, be used for programming.

## **2.3 Visual Studio Code**

To structure and write good code all developers need a good code editor. Visual Studio Code was chosen for this task because it is a lightweight code editor with a deep extensions marketplace.

Visual Studio Code handles HTML, CSS, SQL, PHP and Python, all languages used in this project. The built-in terminal and debugger are easy to use and makes iterative changes easy.

## **2.4 User Stories**

User stories is a way for the developer to get a good understanding of what the user needs from a given product. User stories, or the content generated from user stories, is formatted in a way as if it was written from the user's perspective. User stories are a part of the agile framework and is often used to get a quick understanding of what the value is for the customer. [13]

## **2.5 Requirement Specifications**

Requirement specifications is a way of summarizing the requirements that the customer and developer has on the product or program. It is often done in different types of lists and stretches from design to functions and backend. When all the demands in the requirement specification is met the product is done. [14]

## **2.6 Function analysis**

Function analysis can be similar to requirement specification, but is instead divided into different categories. These categories can often be, but are not limited to, main function and sub functions to that function. For the main function to be done the sub functions also need to be done. These different functions can be divided into classes, such as class 1, class 2 and class 3. [15]

## **2.7 Related works**

Related work was researched for this project to get a good baseline understanding of how web scraping work. During this research two interesting articles were found.

### **2.7.1 An Overview on Web Scraping Techniques and Tools**

In this article written by Anand V. Saurakar, Kadar G. Pathare and Shweta A. Gode the authors explain in detail what web scraping is, how it works and what it can be used for.

For this project the most important part of this article is where the authors explain what web scraping is and how it works. In the article the authors say that web scraping is a way to pull unstructured information from a website in a structured manner so it can be easily accessible for the users or developers. [16]

This can be done to all types of different data, but a few examples mentioned are stock price, item pricing, product details or reports. [16]

The article goes on to mention that a website can be seen in three different ways. The HTML website that the user sees, the HTML code and the document object model and they visualize this with the following graphic. [16]

It is through these three different viewpoints that the web scraper uses HTML tags to scrape the data from the website.

### **2.7.2 A comparative study on web scraping**

The second article is written by SCM de S Sirisuriya and here the author explains that web scraping is a way for developers and user to extract data from big websites with sometimes multiple pages of data in an



automatic or vastly faster way. This data can then be visualized in a better way. [17]

SCM de S Sirisuriya's article is interesting for this project as it aligns with the goals that has been set in this project. In the article the author mentions several different ways to handle web scraping but is focused on already pre-built solutions. These solutions often have visual interfaces where the user can decide what should be scraped from a website. These solutions are beyond what will be done in this project but showcase different solutions for web scrapers.

## **3 Methodology / Model**

This report and project can be broken into a few different steps with the over-arching goal of building a platform where users can web scrape their TimeEdit scheduling pages, upload this parsed, structured, and cleansed data a website where the data gets stored in a database. This data can then be presented in a visually pleasing way to the user.

In chapter 3, Methodology, the different processes chosen for the project are presented together with the main steps done to ensure that the project is followed through.

The methodology used in this project is a combination of previously acquired knowledge about methods and projects done at Mid Sweden University.

### **3.1 Time planning**

Due to the scope of the project the structure and planning are key elements to achieve the set goals. Three different objects need to be built with a python-based web scraper, a dynamic website that can handle active users and a database to store both log-in information regarding the user, but also the scraped data that the user uploads.

To achieve this a Gant-based schedule was chosen to plan how the time was divided between the three different objects and writing the report. [18]

### **3.2 Development process**

A development process was created early in the project to go from a concept and idea to developed product. The development process for this project can be divided into 7 different steps.

#### **3.2.1 Pre-study and data collection**

The first step in the pre study is to analyze what needs to be done. How much research is needed to learn how to build a python-based web scraper. During this step questions like if code from previous projects can be re-used or not is answered. It's also the step where the baseline of the platform is outlined.

### **3.2.2 Requirement specification and user stories**

The second step is to do a requirement specification. What must be developed to accomplish the project based on the goals and boundaries that has been previously mentioned. Creating a solid requirement specification is important so the development has an end goal. The requirement is usually what the project needs to accomplish when done.

### **3.2.3 Functional analysis**

The third step is to do a functional analysis. Step two, requirement specification, makes it clear what the projects requirements are, what it needs to achieve, while functional specification is a continuation of this.

The functional analysis is done to evaluate what functions the platform needs to have. These functions are then presented in a table that clearly outlines what the developed product needs to be able to do.

### **3.2.4 Ethical Aspects, GDPR.**

Step four is to evaluate what ethical aspects that needs to be considered during the project. As of the 25<sup>th</sup> of May 2018, the GDPR, short for General Data Protection Regulation was passed. This is a data privacy law that all states in the European union need to follow. As this project will handle users and data from those users, the GDPR laws must be considered. [19]

### **3.2.5 Environmental Aspects**

Step five is environmental aspects of the project. As of 4<sup>th</sup> of November 2016, the Paris agreement was implemented over the world. The object of this agreement is the keep the average temperature below 2 degrees, but preferably 1.5 degrees, compared to pre-industrial levels.

This project does not have a direct impact on the environment, but, living in a modern world where society at large is trying to reach these goals it is important to evaluate what impact the project can have, both good and bad. An example of this could be less travel time to the university because the developed product leads to better planning, or vice versa. [20]

### **3.2.6 Traditional development**

During the project iterative and agile development was used. A lot of knowledge was gained, and a lot of research was read during the project, which mean that the most important part of development was having an iterative pipeline where code and functions could be changed and iterative as the development continued.

The result of the development is the finished project.

### **3.2.7 Python**

As discussed in chapter 2.1.2, Python is a well-developed strong language with a vast library, with the most important one for this project being beautifulsoup for web scraping.

Different languages were considered for this project the choice of Python was motivated through previous experience with the language and the existence of the web scraping library beautiful soup.

### **3.2.8 Beautiful Soup**

Beautiful Soup is the library used for this project because of all the well documented functions. There are other libraries that are as easy to use, such as lxml, but Beautiful Soup has a big and dedicated user base that publishes solutions to a lot of problems. This makes it easy to troubleshoot issues on the internet and reach out for help. For more advanced projects or other types of web scraping better libraries exist, however, the learning curve of these can be steep and to big for the scope of this project.

## **3.3 Evaluation**

Once everything has been developed an evaluation is done to see if the new system is better than the old system. This is a way of quality control to make sure that the goals of the project were reached.

The evaluation will be done measuring two different variables. The number of clicks needed for both the new system and the old system, and the amount of time needed for the old system and the new system. This will then be presented under chapter 5.4.

The number of clicks is measured through counting the number of clicks from start to result. The amount of time is measured by the amount of time from start to presented result. The tests are done by the author of this report, and done back to back to ensure the fairest amount of measurements.

### **3.4 Presentation**

When the development process is done, and the coding and development of the project is finished a technical report is written. This report includes the theory behind the project, the tools used and how the project was constructed.

During the entire project an iterative design and development process was used to make sure that time was allocated as best it could be and that the project reached completion in time for the deadline.

## 4 Design and Implementation

Chapter four showcase how the system will be designed, developed, and implemented. It is divided up into sub chapters for each step taken in the development and is split between the web scraper and the website.

### 4.1 Development

The development process was divided into several different steps listed below.

#### 4.1.1 Pre study and Data Collection

The first step of the project is to analyze how the system would work and what needed to be done to get the information needed for the project from the website. Before a general design was created literature was read on the internet to get a grasp of how it works. The previously mentioned articles were read to get an understanding of already working web scrapers.

#### 4.1.2 Process of the platform

A sketch of how the finished platform would work was done early to make sure that everything needed was developed.

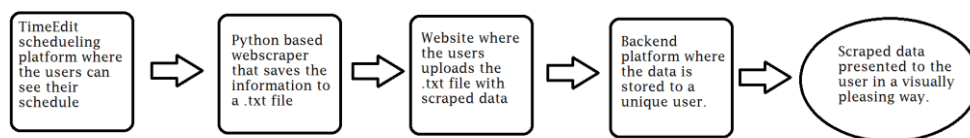


Figure 2: first draft of how the developed platform will handle data.

Whilst analyzing what needed to be built to reach the projects goals a process was developed. It consisted of five major steps that needed to be created to reach the set goals of the project.

1. The user needs to find their schedule on the TimeEdit platform. Either the personal schedule or the schedule for a specific class or teacher.
2. A python-based web-scraper scrapes the data from the schedule and saves this data to a text file.

- a. The correct information needs to be collected. A modern website has thousands of lines of code, and the web scraper needs to scrape information about classes.
3. A website where the user can log in needs to be developed. It is important that the user can log in so that the scraped data can be saved to the correct profile in a secure and private manner. When the user uploads the .txt file containing the data to the website the website needs to parse this data again and add this to the database.
  - a. By parsing the data at two different steps, the web scraper can be built to scrape more data than is needed for this project. More functions can then be added to the website down the line without further development to the web scraper.
4. A backend platform where the user's scraped data is stored and can be collected for use. The .txt file should also be saved for future use.
5. A profile page where the user can get the data presented in a visual manner.

By developing these main objectives, the entire platform will be able to do what the project has set out to achieve.

#### **4.1.3 Information gathering**

To develop a python-based web scraper a lot of research was done. The library for using BeautifulSoup is very well documented on the website for the library. This, combined with knowledge from previous courses at Mid Sweden University together with the help of google made it possible to build a web scraper using python. [21]

Building the dynamic website became a bigger task than the actual web scraper as the development for this had more moving parts. Knowledge about the how to build a login system with unique users had to be acquired. That unique user had to be able to upload a file, and the website extract the text strings from that file and add them into the database.

Once the user has done these steps the website has to be able to display and show this to the user. This requires different get methods in SQL in combination with HTML and CSS to display the information correctly.

A big part of the project was spent gathering information regarding how to develop and build these modules of the project. A lot of knowledge was able to be reused from previous courses at Mid Sweden University which saved time during development.

## 4.2 Requirement specification

A shorter requirement specification was done during this project. As this is a smaller project the requirement specification focusses broadly on what must be developed, and the functional analysis, 4.3, focus on what functions need to develop to reach those goals.

System	Demandtype	Demand
Webscraper	Must	The webscraper needs to be able to scrape information regarding classes from TimeEdit.
	Must	The webscraper needs to be able to save this information to a text file.
	Must	The information saved needs to be structured with HTML tag
	Should	The webscraper needs to be able to save this information to a text file.
	Must	The website needs to be able to handle unique users
Dynamic WS	Must	The user need to be able to upload a .txt file
	Must	The website needs to be able to extract information from .txt file and display it for the user
	Should	The website needs to should the extracted information from the website to a database
	Should	The user should to be able to find this information when logging back into the website again
	Should	The information should be displayed in a visually pleasing way.
	Should	

Table 1: Requirement Specification



If everything in the must column gets developed the project will have reached most of the goals set out. If all the demands in the must and should categories have been reached all the goals in the project will have been reached.

After the requirement specification was done user stories were performed on two students to acquire if something important has been missed. Here, the students followed the process of the platform after being pitched the idea. The students wrote down what requirements they would have from a website like this, and demands were created from this.

The demands and user stories are not sorted in any order.

<b>Demand</b>	<b>User Stories</b>
<b>Easy to use</b>	<i>As a user it needs to be easy to use the web-scraper.</i>
<b>Sort function</b>	<i>As a user I should be able to sort what I want to collect data from. For example if I only want to see how much math I have.</i>
<b>Work on multiple TimeEdit pages</b>	<i>As a user I want to be able to scrape my teachers schedule</i>
<b>Login function.</b>	<i>As a user I want to be able to log into my own account on the website.</i>
<b>Easy to use website</b>	<i>As a user I want it to be easy to log in to the website.</i>
<b>Possible to save uploaded data</b>	<i>As a user I want to be able to go back and see my previously scraped data</i>
<b>The process needs to be iterative.</b>	<i>As a user I want to be able to upload data more than once.</i>
<b>Secure and unique login.</b>	<i>As a user I want to know that not everyone can access my data and exams.</i>
<b>Navigation bar</b>	<i>As a user I want to be able to navigate the website using a menu</i>
<b>Delete the account</b>	<i>As a user I want to be able to remove my account.</i>
<b>Visually pleasing results.</b>	<i>As a user I want to get good feedback on how many hours I am scheduled this period on in what course.</i>
<b>High usability</b>	<i>As a user I want to be able to navigate the website with high efficiency.</i>

Table 2: User Stories.

With user stories and requirement specifications done it is possible to do a function analysis based on these requirements.

### 4.3 Function analysis

The functions are based on the user stories and requirement specifications. These functions are needed to meet the requirement specifications of the project and they are sorted into three different classes. Class 1 is functions that are demanded, class 2 is functions that are needed, and class 3 is functions that are wanted.

	Function	Class
Web Scraper	Scrape class time	2
	Scrape class code	1
	Scrape class name	2
	Scrape professor name	2
	Print results into a .txt file	1
	Print results sorted	1
	Print results without HTML tags	1
	Sort courses based on inputs	3
	Save everything in one file	2
Dynamic website	Create a user	1
	Save user to database	1
	Login to website	1
	Database to store user	1
	Unique profile page	2
	File upload	1
	Text extraction from files	1
	Sorting extracted text	1
	Upload sorted extracted text to database	1
	Present visually pleasing information to the user	2
	User friendly UI	3
	Let the user come back and see results again.	1
	Allow user to upload new data	3

Table 3: Function Analysis

#### **4.4 Ethical Aspects, GDPR**

During the project several different ethical aspects were considered, the biggest one being GDPR. When working with user data, and more specifically, personal user data, great care has been taken about how that data is stored.

Great emphasis was put on having hashed and protected passwords, where only the user that was logged in only had access to the data that was connected to that user. No other user should or could be able to see the data that they had uploaded.

Personal information was not needed during user stories or testing of the platform. As the test group was two students the information could be anonymized and saved in excel-sheets to develop the requirement specification and functional analysis.

#### **4.5 Environmental Aspects**

This project was created on a home computer and is being hosted on a hosting service. The research and development during the project have no environmental impact. The long-term effects of the developed product can have positive effects such as smarter scheduling of students and professors, however, this is not something that is being considered during the project.

#### **4.6 Development**

Chapter 4.6 is divided into subchapter to better present the construction of the platform.

##### **4.6.1 Analyzing TimeEdit**

The first step of building a web scraper is deciding what on the website should be collected by the web scraper. TimeEdit is designed in a way that creates both challenges but also makes it easy to divide the scraped data.

# Developing a Python based web scraper – A study on the development of a web scraper for TimeEdit

Pontus Andersson

2021-09-13

Time	Kurs		Lokal	Moment	Grupp	Föreläsare	Info, Examination	Länk
Fri 2021-08-20 w33								
10:15 - 11:00	MA009X, K2039	Matematik BE, Preparandkurs för civilingenjörstudier		Frågestund		Tomas Nilson	Frågestund 2 <a href="https://miun-se.zoom.us/j/6858822323">https://miun-se.zoom.us/j/6858822323</a>	
Mon 2021-08-23 w34								
11:15 - 12:00	MA009X, K2039	Matematik BE, Preparandkurs för civilingenjörstudier		Frågestund		Tomas Nilson	Frågestund 3 <a href="https://miun-se.zoom.us/j/6858822323">https://miun-se.zoom.us/j/6858822323</a>	
Wed 2021-08-25								
15:15 - 16:00	MA009X, K2039	Matematik BE, Preparandkurs för civilingenjörstudier		Frågestund		Tomas Nilson	Frågestund 4 <a href="https://miun-se.zoom.us/j/6858822323">https://miun-se.zoom.us/j/6858822323</a>	

Figure 3: TimeEdit design

As seen on figure 6 every scheduled moment gets a box where the information regarding time, course code, course name, moment, professor, and information is being saved. These boxes are built as dynamic cubes on the website generated in the backend of TimeEdit.

The first step is to recognize where on the website this data is being presented. This can be done through the inspection tool on modern web scrapers.

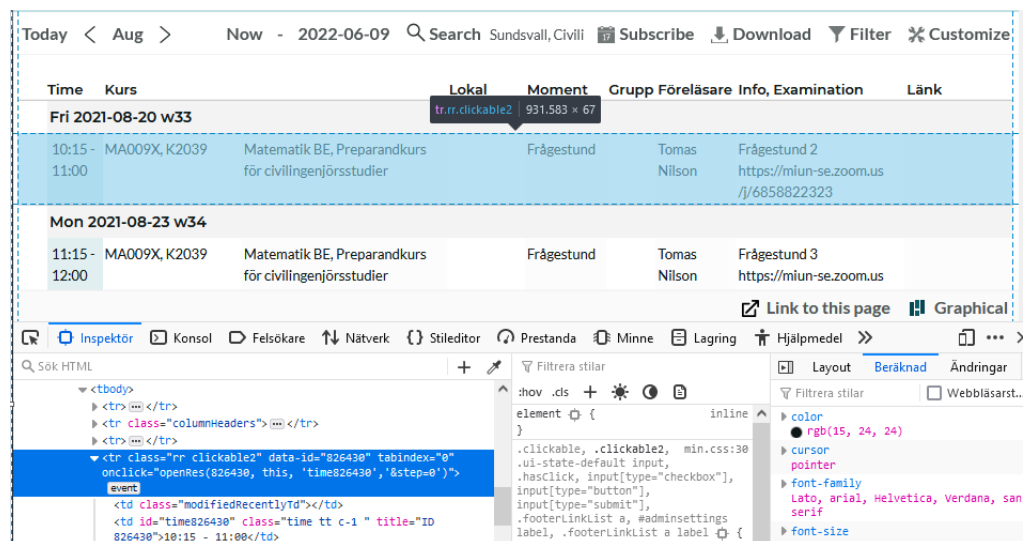


Figure 4: source code of TimeEdit

Knowing what all the classes of the website are named is needed to scrape the data. Figure 7 shows that all the information needed for this project is stored in the class "rr\_clickable2". Scraping "rr\_clickable2" would not be sufficient, as the result would be all the containing classes inside this class.

By going further into the class, the containers containing the information needed can be found.

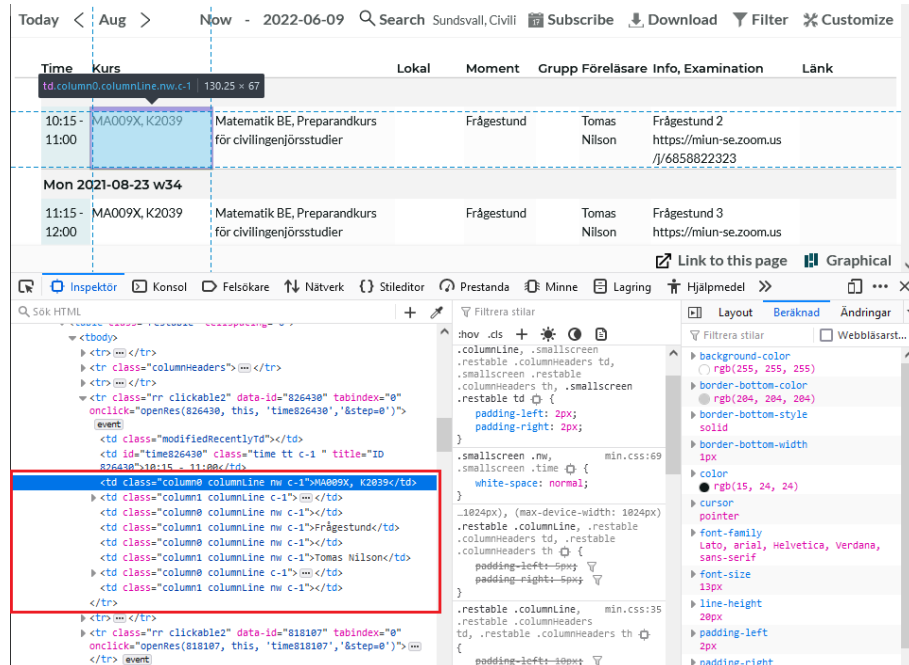


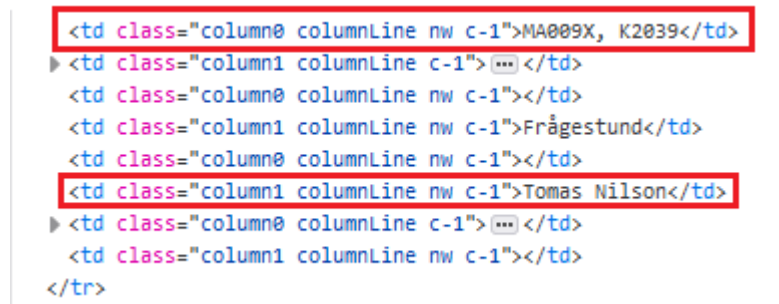
Figure 5: Classes containing the data for the project where the red box highlights the classes.

Figure 5 show the classes needed to scrape the data. By scraping the specific classes containing the data needed the saved data is more structured and easier to handle and less work needs to be done on the website down the line.

By inspecting the classes in the source code of TimeEdit one major issue arise. Some of the classes have identical names to each other and when working with web scarpers this is a critical problem.

As the concept goes out on scraping by identifying classes one of the classes will either not be scraped or will be scraped at the same time leading to corrupt data that can't be used.

In figure 5 the red box around the code visualizes what classes that display the information on the TimeEdit website and in figure 6 the red boxes showcase that two of those classes have identical names but different content.



```
<td class="column0 columnLine nw c-1">MA009X, K2039</td>
<td class="column1 columnLine c-1">...</td>
<td class="column0 columnLine nw c-1"></td>
<td class="column1 columnLine nw c-1">Frågestund</td>
<td class="column0 columnLine nw c-1"></td>
<td class="column1 columnLine nw c-1">Tomas Nilson</td>
<td class="column0 columnLine c-1">...</td>
<td class="column1 columnLine nw c-1"></td>
</tr>
```

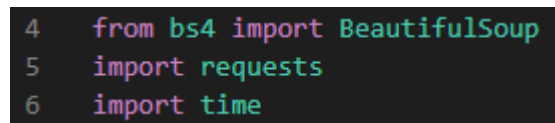
Figure 6: Identical class-names where the red box highlights the identical ones.

This problem was never solved and is discussed further in chapter 6, discussion.

#### 4.6.2 Building a python-based web scraper

When the knowledge about what should be scraped from TimeEdit is known it is possible to start building the actual web scraper.

All needed libraries are installed on the project computer and imported into the code.



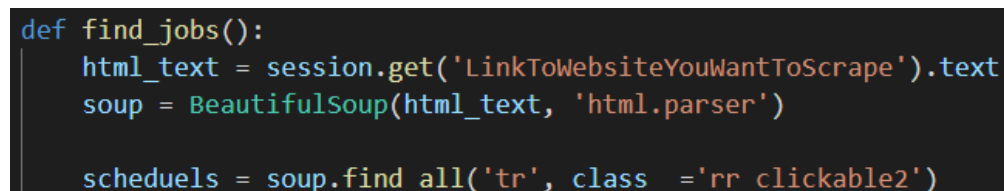
```
4 from bs4 import BeautifulSoup
5 import requests
6 import time
```

Figure 7: libraries used.

The scraper needs to know what website it scrapes. There are commands ready for this in Python. The first step was making sure that it was possible to scrape any data from TimeEdit.

By telling the web scraper to scrape all boxes it was possible to evaluate if data could be scraped or not.

In figure 11 the text "LinkToWebsiteYouWantToSCRAPE" is replaced with the link for the TimeEdit that the user wants to scrape.



```
def find_jobs():
    html_text = session.get('LinkToWebsiteYouWantToScrape').text
    soup = BeautifulSoup(html_text, 'html.parser')

    scheduels = soup.find_all('tr', class_='rr clickable2')
```

Figure 8: scraping all boxes.

During the development of the web scraper a major flaw was found with scraping TimeEdit. For a student to see their personal schedule the student needs to be online on the website. The website uses a third-party login system to connect with different universities all around the world. When logging in to the website the user gets forwarded to Mid Sweden Universities log in system.

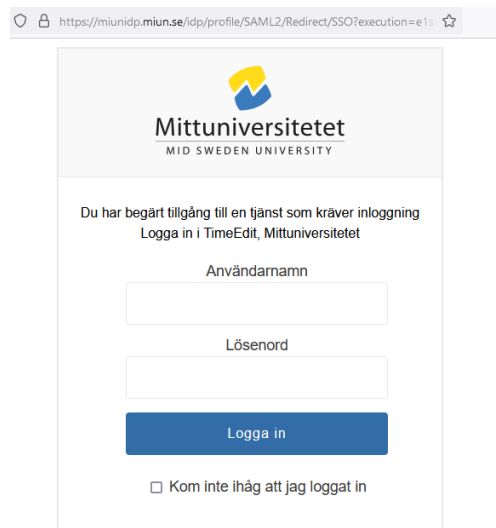


Figure 9: MIUN log in system

The issue with this is that it starts a session for that user, and it requires the user several clicks to reach the log in page. It is possible to scrape websites locked behind log in requirement using functions within beautiful soup, see figure 10, however, as the Mid Sweden University link is a session-based log-in, no way was found during this project to make the scraper automatically log in to the website as no work around to all the forwarding was found.

```
#Skapar en payload
payload = {'j_username' : 'ANVÄNDARNAMN', 'j_password' : '#LÖSENORD!', '_eventId_proceed' : 'Logga in'}
#hemsidan som jag vill logga in på.
s = session.post("https://miunidp.miun.se/idp/profile/SAML2/Redirect/SSO?execution=e1s2", data=payload)
```

Figure 10: logging into a website.

The issue of this is that a major part of the goals of the project is that the user should be able to scrape the individual data of that user. Thus, not being able to log in to the personalized schedule is a major drawback.

That being said, a workaround to this problem comes from how TimeEdit works. On TimeEdit it is possible for the user see the schedule

for their year and course. By using these tools, the user can select what year and program they study and get the schedule for each period. A lecturer can then sort courses based on the name and see all courses the chosen lecturer is hosting.

This problem is further discussed in chapter 6, discussion.

At this point the source and class variables being scraped are known and it is possible to start scraping the information from the TimeEdit. As it is no longer possible to scrape the users personalized schedule a function to sort out specific courses by typing them into the program was developed.

```
#time = scheduel.find('td', class_ = 'time tt c-1').text
course_code = scheduel.find('td', class_ = 'column0 columnLine nw c-1').text#.replace(' ', '')
course_name = scheduel.find('td', class_ = 'column1 columnLine c-1').text
type = scheduel.find('td', class_ = 'column1 columnLine nw c-1').text

if unwanted_courses not in course_code:
    with open(f'files/test.txt','a') as f:

        #f.write(f"{time},")
        f.write(f"{course_code}")
        f.write(f";{course_name};")
        f.write(f"{type}")
        f.write(f"\n")
```

Figure 11: variables being scraped.

As TimeEdit uses the same class name for the lecturer and course code both cannot be scraped using these methods. The choice to only scrape the course code was done.

With the web scraper it is possible to scrape the time of the lecture as well. TimeEdit changes the format of how this is saved. As this format changed the choice to remove time from the scraped data was taken as it made the data in the file hard to predict and handle.

All the data scraped from the website is then saved to the file test.txt.

The actual sorting of the scraped data will be done on the website where the user uploads the file.

#### 4.6.3 Building a dynamic website

Once the scraped data exists and can be collected the work on building the website where the functions to display the data back to the user exists.



As previously discussed in this report a main goal of the project was to create a platform with unique users that has hashed and secure passwords.

A basic form is created where the user enters the username chosen and the password connected to that user. This is stored in a database together with a unique id that gets created together with the user. The website has built in features that makes sure that no two users with the same name gets created but even if this happens, they would be separated with the unique ID.



	idUsers	uidUsers	emailUsers	pwdUsers
<input type="checkbox"/> Redigera <input type="plus"/> Kopiera <input type="minus"/> Radera 1	PontusAdmin		pontus@surfar.se	\$2y\$10\$XnSZU.HWkmDEeCRtHnn3iOUqnUG8DGy3Bb162KHh
<input type="checkbox"/> Redigera <input type="plus"/> Kopiera <input type="minus"/> Radera 2				\$2y\$10\$Td8UTVfSGGJbDJgdGaDD7uLsQtN99YlaVusjRUxjzwo
<input type="checkbox"/> Redigera <input type="plus"/> Kopiera <input type="minus"/> Radera 3				\$2y\$10\$084ZIFBnlkmR0nNXrkxB.dhy.hS65Bv.C6eky6hT5p...
<input type="checkbox"/> Redigera <input type="plus"/> Kopiera <input type="minus"/> Radera 4				\$2y\$10\$LLwEc8E5HLTKR9Jk2pSICet6DNeob3nM3soGqXSAXx
<input type="checkbox"/> Redigera <input type="plus"/> Kopiera <input type="minus"/> Radera 5				\$2y\$10\$.frL0.b726WvYr.XkDdqjer2svbKJbAvavGon.Isann...
<input type="checkbox"/> Redigera <input type="plus"/> Kopiera <input type="minus"/> Radera 6				\$2y\$10\$4R/.2eUVJV0JIYvk9YbKJOycZN6uGukWSU2omyWIP7:

Figure 12: stored users in the database.

This unique ID is what enables the entire website to be dynamic towards the user that is currently logged in.

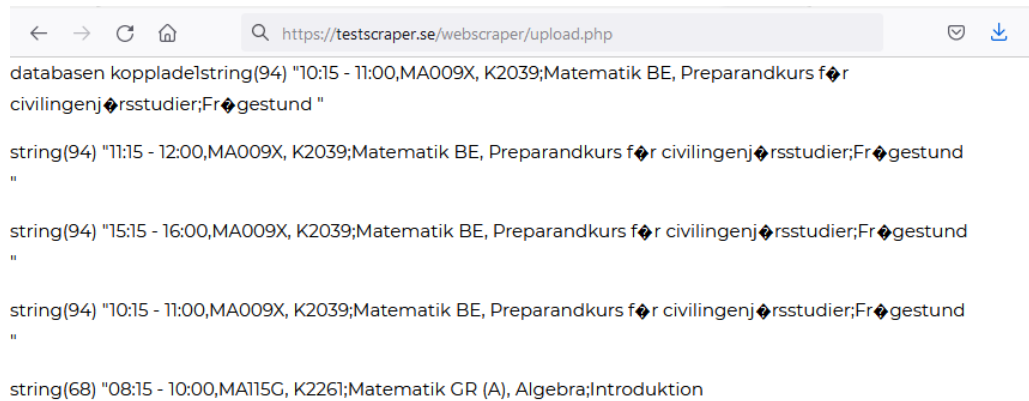
This system also includes features such as log in through email and forgotten passwords in case a user cannot access the account anymore. The website is designed in such a way that every time a user navigates to a new page a session is started. This session includes the previously mentioned user data and can change according to that.

With a working login system, a function to handle file uploads is built. To build a system where it is possible to upload a file to the database is done by using simple web programming methods, however, due to the design of this project the website needs to be able to handle text files with different names from different parts of a computer, and this is a more complex problem as files can clash with the same name in the database.

To work around this the information in the file was extracted before the file was uploaded to the database.

Php can handle contents of a through already existing functions. By using the function explode in php the website changes the name of the .txt file but keeps the extension, making sure that it gets saved to the database with a random name.

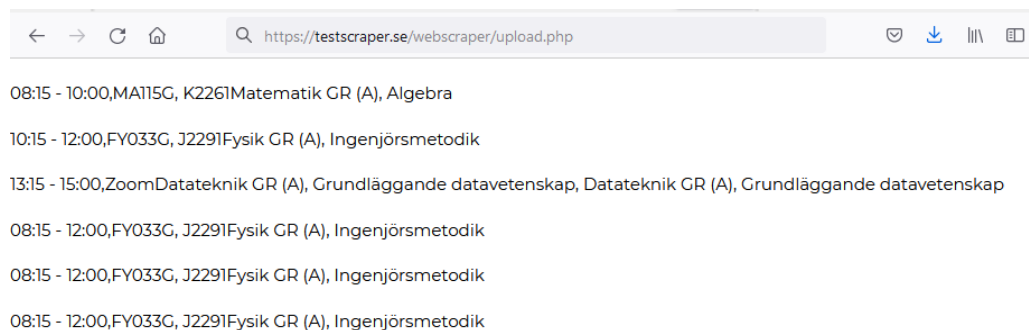
When the users press the button upload the website creates a temporary array. Inside this array the contents of the file get read line by line and are saved as strings, see figure 16.



```
databasen kopplade string(94) "10:15 - 11:00, MA009X, K2039; Matematik BE, Preparandkurs för civilingenjörsstudier; Frögestund "
string(94) "11:15 - 12:00, MA009X, K2039; Matematik BE, Preparandkurs för civilingenjörsstudier; Frögestund "
string(94) "15:15 - 16:00, MA009X, K2039; Matematik BE, Preparandkurs för civilingenjörsstudier; Frögestund "
string(94) "10:15 - 11:00, MA009X, K2039; Matematik BE, Preparandkurs för civilingenjörsstudier; Frögestund "
string(68) "08:15 - 10:00, MA115G, K2261; Matematik GR (A), Algebra; Introduktion "
```

*figure 13: raw data*

As seen in figure 16 this raw data is too unorganized to use and needs to be structured. At this stage it is possible to see why it was important to scrape the data in a structured manner when building the web scraper. By adding characters such as ";" and "," during the scraping process it is now possible to use php functions to explode and structure this data, see figure 17.



```
08:15 - 10:00, MA115G, K2261; Matematik GR (A), Algebra
10:15 - 12:00, FY033G, J2291; Fysik GR (A), Ingenjörsmetodik
13:15 - 15:00, Zoom; Datateknik GR (A), Grundläggande datavetenskap, Datateknik GR (A), Grundläggande datavetenskap
08:15 - 12:00, FY033G, J2291; Fysik GR (A), Ingenjörsmetodik
08:15 - 12:00, FY033G, J2291; Fysik GR (A), Ingenjörsmetodik
08:15 - 12:00, FY033G, J2291; Fysik GR (A), Ingenjörsmetodik
```

*figure 14: structured data.*

The data is now presented in a manner that is possible to add to the database connected to the website. Because of the unique user id this data is saved to the correct user inside the database.

kurskod	kursnamn	idUsers
MA009X, K2039	Matematik BE, Preparandkurs för civilingenjörstud...	1
MA009X, K2039	Matematik BE, Preparandkurs för civilingenjörstud...	1
MA009X, K2039	Matematik BE, Preparandkurs för civilingenjörstud...	1
MA009X, K2039	Matematik BE, Preparandkurs för civilingenjörstud...	1
MA009X, K2039	Matematik BE, Preparandkurs för civilingenjörstud...	1
MA115G, K2261	Matematik GR (A), AlgebraIntroduktion	1
DT027G, F2011	Datateknik GR (A), Grundläggande datavetenskaplnt...	1
MA115G, K2261	Matematik GR (A), AlgebraFöreläsning	1
FY033G, J2291	Fysik GR (A), IngenjörsmetodikFöreläsning	1

figure 15: database with structured data.

With the data saved to a database in a structured way it is now possible to present this in any way, shape, or form to the user and a results page was built to display it. This is shown in chapter 5, results.

To present this data the SQL (\*) count function was used.

## 4.7 Testing the efficiency

To test the efficiency of the system two different variables were used. The number of clicks the user needs to do from start to finish, divided between three different steps on the new developed system and the old way of doing it, by counting for hand. The time needed to do these clicks were recorded to evaluate how much faster the new system is.

The aim is to measure the same variables between both systems, even if they are done in different ways. TimeEdit was set to display a period of a week. This is done to quantify if the new system is more efficient than the old system.

### 4.7.1 Testing the developed system

The test was divided into three steps, as seen below.

- Reach their schedule in the web browser.
- Scrape the schedule with the web scraper.
- Upload the file from the web scraper to the website and get the result presented.

#### **4.7.2 Testing the old system**

The old system was counting the number of lectures by hand and then looking at how many hours that is per week. The same variables were measured for the old system as well, but with a different approach.

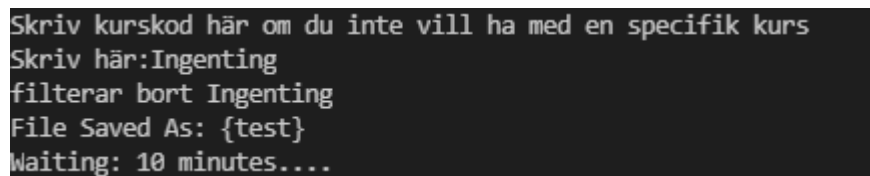
- Reach their schedule in the web browser.
- Count the number of lectures in the schedule by hand.
- Calculate the amount of time those lectures amount to.

## 5 Results

In chapter 5, results, figures, and illustrations show the developed platform and systems in full. The developed systems are being evaluated against the goals set out at the start of the project. The program is also evaluated to see if the developed solution is faster than the old method used for the same task.

### 5.1 Web scraper

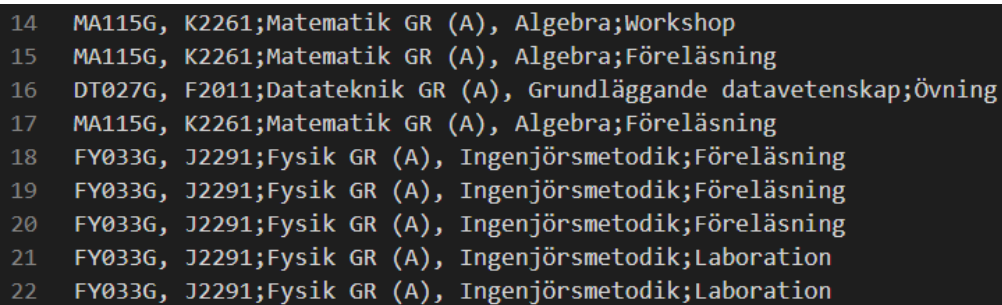
The developed web scraper has no interface but can be saved as an executable file that the user can run. This file will start up the terminal and ask the user to remove what courses that should be sorted out.



```
Skriv kurskod här om du inte vill ha med en specifik kurs
Skriv här:Ingenting
filterar bort Ingenting
File Saved As: {test}
Waiting: 10 minutes....
```

Figure 16: text-based interface.

The interface is text based and tells the user what has been done. The result in the file test is sorted, structured and cleansed.



```
14 MA115G, K2261;Matematik GR (A), Algebra;Workshop
15 MA115G, K2261;Matematik GR (A), Algebra;Föreläsning
16 DT027G, F2011;Datateknik GR (A), Grundläggande datavetenskap;Övning
17 MA115G, K2261;Matematik GR (A), Algebra;Föreläsning
18 FY033G, J2291;Fysik GR (A), Ingenjörsmetodik;Föreläsning
19 FY033G, J2291;Fysik GR (A), Ingenjörsmetodik;Föreläsning
20 FY033G, J2291;Fysik GR (A), Ingenjörsmetodik;Föreläsning
21 FY033G, J2291;Fysik GR (A), Ingenjörsmetodik;Laboration
22 FY033G, J2291;Fysik GR (A), Ingenjörsmetodik;Laboration
```

Figure 17: scraped data.

The developed web scraper can scrape the course code, course name and extra information. It cannot scrape the name of the lecturer. As seen in figure 20 characters such as “;” and “,” is used to divide the data inside file. This becomes important when uploading it to the website.

Figure 20 shows a small sample of everything scraped from the website.

The web scraper can handle the encoding iso-8859-1 that is Swedish letters such as “ä”, “å” and “ö”. Without this encoding errors could occur when scraping websites with letters outside the English alphabet.

Once the program is run again the file is emptied and the new data is added. If the data inside the file is not uploaded the database before it is run again the data inside the file is lost.

Because of this the data uploaded to the website is always new and never old.

## 5.2 Dynamic website

The website uses a fully functioning system where users can create accounts and log in using both email and passwords and the website follows a simple design principle where all the pages have the same layout but with different content.



Figure 18: sign in page.

Once the user has logged on the website directs to the start page. Here the user can choose to upload the scraped data or see the previous results. The previous results only works if the user has already uploaded information to the web page.

The website also reacts to the user being online.

# Developing a Python based web scraper – A study on the development of a web scraper for TimeEdit

Pontus Andersson

2021-09-13

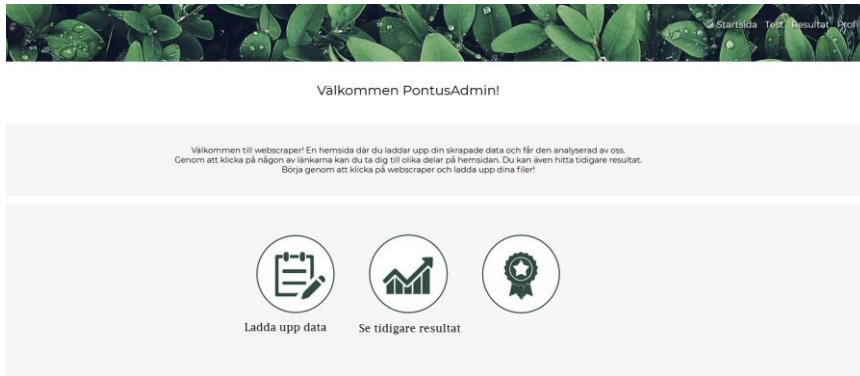


Figure 19: start page.

By following the website, the user gets to the page where the data should be uploaded.

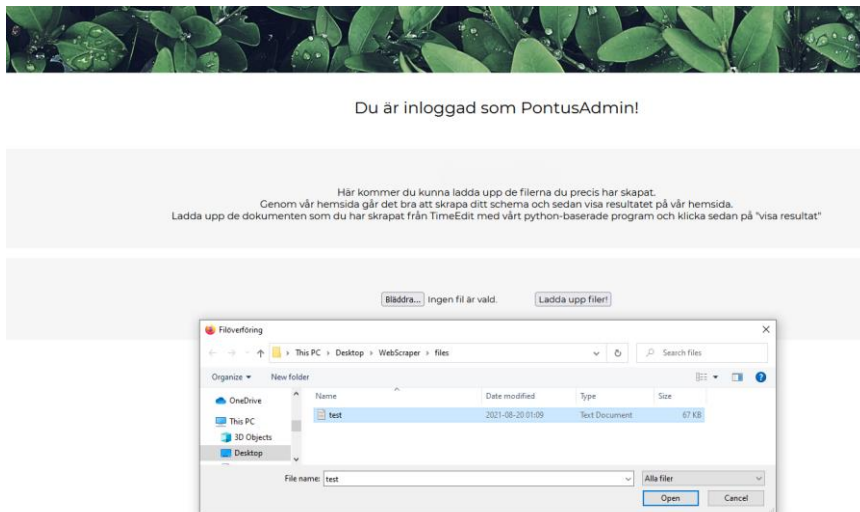


Figure 20: user uploads data

When the user presses upload the functions built into the website uploads it into the database in a structured, clean, and functional way. This data is bound to the user that uploaded the file to the website.

+ Alternativ		
kurskod	kursnamn	idUsers
MA009X, K2039	Matematik BE, Preparandkurs för civilingenjörstud...	1
MA115G, K2261	Matematik GR (A), Algebra	1
DT027G, F2011	Datateknik GR (A), Grundläggande datavetenskap	1
MA115G, K2261	Matematik GR (A), Algebra	1
FY033G, J2291	Fysik GR (A), Ingenjörsmetodik	1
Zoom	Datateknik GR (A), Grundläggande datavetenskap, Da...	1
FY033G, J2291	Fysik GR (A), Ingenjörsmetodik	1
FY033G, J2291	Fysik GR (A), Ingenjörsmetodik	1

Figure 21: data in the database

The website can now present this to the user. The data is sorted by the course code and then presented to the user on the results page. By knowing how many times a course exists in the database it is possible to calculate how many hours the student or professor needs to attend to that course over the period or week, depending on how long a time frame the data has been scraped from.

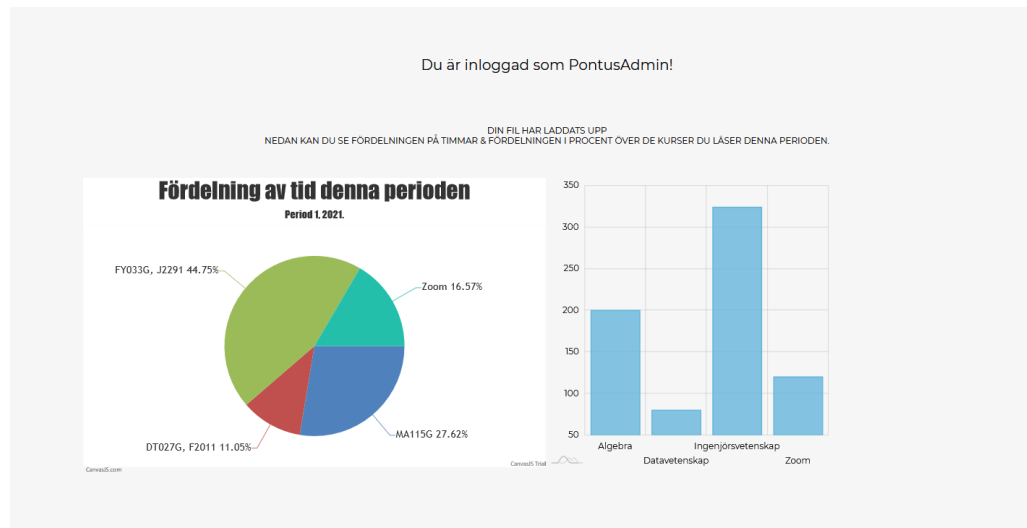


Figure 22: the data presented to the user.

## 5.3 Analysis of results and developed program

This project set out to build a platform where students and professors could evaluate the number of hours that had been scheduled and see if this stands in relation to how many hours should be worked every week. The result of this could in the long term give the opportunity to plan better, work ahead and help schedulers divide the time in a better way.

The developed platform can do this – but it does have some drawbacks listed below.

### 5.3.1 Analysis of the web scraper

A web scraper is very practical in theory but has some major drawbacks in practice. The TimeEdit website is dynamic, and a lot of information is being done server-side where it is not possible for the user to inspect the code. Therefore, different classes can have the same name when inspecting the website, as the code run in the background fetches information to put into that class from the back end.



Because of this the web scraper designed to scrape TimeEdit is not as powerful as it could have been.

The web scraper also has a hard time adapting to changes on the website. If TimeEdit changes or if the classes change names the python code has to be updated to match the new updated website which in practice makes the web scraper obsolete very quickly and in need of constant updates.

TimeEdit also uses a third-party log in system bound to Mid Sweden University. Because of how this system is designed the web scraper cannot scrape personalized schedules. The web scraper works around this by scraping the entire course schedule for that program, but this means that a student with a personalized schedule will not be able to use the program.

The user must put the link that needs to be scraped directly into the source code, a task that can be seen as very easy by a developer. Tests of the system show that this is a major drawback as someone unfamiliar to code have a hard time understanding how to get started.

The reason for the web link having to be hard coded is because the program got stuck in a loop when the user input the link. Just as with the time variable it did on occasion work, however, the scraper returned empty data most of the time. The program did not know how to handle this and crashed/could not find a website to scrape.

The working theory on this is that the TimeEdit link sometimes change, or that the program could not handle all the signs that are contained inside a website link. This problem was never resolved, and the link was hard coded to the project could continue.

The web scraper can scrape the information that is needed from the website for this project despite the drawbacks of the functions included. A total of three clicks are needed to start the web scraper for a user. Additionally clicks are needed to add the link into the code.

### **5.3.2 Analysis of the website**

The web site developed to analyze the scraped data works above expectations set in this project. A user can log in, upload the file scraped from the web scraper and see the results presented in a visually pleasing manner.

The website handles the .txt file without issues as the file is converted into a random name. The contents of the file are automatically extracted and uploaded into the database that is behind the website. The extracted content is then bound to the user by the unique ID generated in the sign-up process.

The website is user friendly and easy to navigate and short user tests during development shows that a user can quickly learn how to find results and upload the scraped data to the website.

The data is protected as the password for the users is hashed and securely stored inside the database.

A total of five clicks are needed to upload the web scraped file to the website. The result is a user-friendly website where five clicks are needed to upload the scraped file leading to a very efficient platform.

## **5.4 Results from evaluation of efficiency**

As the test only included one shown week on TimeEdit it was inherently favored towards the old system. The web scraper does not take longer or shorter to show the results based on a given amount of time scrapes, and gives the same over two weeks, four weeks, a period, a term, or a year.

This is not the case with hand counting. Counting a period of a week is faster than counting a period of a month.

### **5.4.1 Results from developed system**

Table 4 shows the result of the test done on the developed system. The test shows that it took a total of 2 minutes and 20 seconds to complete the entire process and get a result of how many hours the user was scheduled the scraped week.

Table 4 and table 5 show that the new system is 59% faster in terms of reaching the goal of knowing how many hours the user is scheduled that week, but that the user needs to do more clicks to get the same work done. This result, however, shows that the new system is measurably more efficient than the old one.

Developed solution		
Activity	Clicks	Time
Reach schedule in web browser	8	1 minutes 12 seconds
Scrape schedule with web scraper	5	0 minutes 32 seconds
Upload the file to website and get result	8	0 minutes 36 seconds
<b>Total</b>	<b>21</b>	<b>2 minutes och 20 seconds</b>

Table 4: Clicks and time from developed solution.

#### 5.4.2 Results from the old system

Figure 28 shows the results of the test done on the old system, counting by hand. The test shows that it took 5 minutes and 44 seconds do the entire process.

Counting for hand		
Activity	Clicks	Time
Reach schedule in web browser	8	1 minutes 9 seconds
Count number of lectures by hand	2	3 minutes 36 seconds
Count number of total hours from lectures	0	0 minutes 52 seconds
<b>Total</b>	<b>10</b>	<b>5 minutes and 44 seconds</b>

Table 5: Clicks and time from old system, counting by hand.

## 6 Conclusions / Discussion

In chapter 6, conclusion, the result, and conclusion of the work done in this project will be discussed if the goals set out in the start of the project were reached.

### 6.1 Methods chosen

The biggest methods chosen for this project was an agile workflow which includes methods like doing a solid pre study, creating requirement and function specifications with the help of user stories and being able to adapt development to the responses.

Having an agile workflow was paramount for the development of this project and made it possible to change how the platform worked based on problems that were found along the way.

By doing a pre study where a lot of information was gathered it was easier to invest time in the areas where this was needed. The pre study showed that build the web scraper in pure code was feasible, but a lot of time had to be spent on analyzing how TimeEdit works and how to scrape the correct information.

The pre study showed what type of data was scraped, how to scrape it and what the general philosophy behind web scrapers are. This led to a faster development period where the developing could be more focused on the pain points.

By constantly iterating based on the problems encountered it was possible to build the entire platform despite having to accept some drawbacks. If further development is done in this project continuing development in an agile environment is highly recommended.

### 6.2 Working with Python

Python is a highly competent programming language, but it is not a language heavily used during the Master of Science, computer science, program at Mid Sweden University. Despite having almost no previous knowledge about the language it was easy to understand and start working with.

The biggest benefit with using python during this project is the competent libraries that exist. Build a web scraper using the beautiful soup library allows for much faster development as the actual functions already exist and only needs to be implemented.

The conclusion from this project is that Python is a strong language to use for this type of web scraper due to its ease of use and if future development of the web scraper is done Python is recommended because of the libraries connected to it.

### 6.3 Concrete and verifiable goals

At the start of this project five different goals were set out to be fulfilled. During the project the focus has been on achieving all these goals.

#### **1: Build a python-based web scraper that can scrape information from TimeEdit's scheduling platform.**

The first goal has been achieved as the web scraper is more than capable to scrape information from TimeEdit. The web scraper has some drawbacks to it, some specific to this web scraper and some specific to web scraping in general.

The research value from this goal is that web scrapers are good tools for collecting data, but that they need to be constantly updated and developed if websites change. They have some drawbacks, such as dynamic websites where it's not always clear what classes can be scraped, however, with the right development web scrapers are competent tools to gather and structure data.

The goal could be improved upon by building a web scraper where the user can scrape data from a personal schedule, or where the user can input the link directly to the schedule into the program.

#### **2: Build a dynamic and scalable website where the scraped data can be uploaded by the user into a database.**

The second goal has also been achieved. This was done using a lot of knowledge gained from previous courses at Mid Sweden University and working with tokens, php and SQL was not new.

The website does what the goal set out for it to do, and the user can upload data from the scraped .txt file directly into the database.

**3: Build functions to sort and present data in a user-friendly way for every unique user**

Goal three was achieved. The website can display the data that the user uploads in a user-friendly way with the help of easy-to-understand graphs. The graphs display the raw data in numbers and percentages so the user in very little time can see how big percentage a course is for a given period, see figure 22.

**4: Implement all three parts of the system so they work together.**

Goal four was achieved. During the project a web scraper that can scrape data from TimeEdit was developed together with a website where the user can upload the data to a database and then get it presented in a visually pleasing way. All three parts of the system were implemented so that they work together and the goal was achieved.

**5: Evaluate if the new system is more efficient than the old one.**

**5b: Evaluate the number of clicks needed for the user to display data.**

During the project goal 5 and 5b were done. The new system is faster than the old one, see chapter 5.4. It did require more clicks but the timer per click is a lot lower. The new system also scales better than the old one, where it takes the same amount of time to scrape one week as one year with the new system, it takes a linear amount of more time to count 1 week and 1 year by hand.

**6.3.1 Summary of goals**

All the goals set out in this project has been fulfilled or partly fulfilled. The project has opened interesting questions regarding future research, read more about this in chapter 6.4, and the developed project is usable today.

The new system is more efficient than the old one.

**6.4 Ethical and societal aspects**

The platform handles user data, and that user data is saved to a database hosted in Europe. The biggest ethical aspect this project handles is the

use of personal data, especially with the new European law GDPR. Another ethical aspect to be considered is how the data is stored. Even without GDPR it is important that the data is stored in a way such that it does not leak. Different users should not be able to see each other's data. It is also important that data generated during user test is anonymous such that no GDPR laws apply.

When working with a tool that is meant to make it less stressful for students it is also important to try to deliver on this. If the system does not work correctly this can lead or displays the wrong information for the user, this can lead to unnecessary stress.

## **6.5 Future work**

A lot of future work can be done in this area. Developers are just now discovering the potential behind web scrapers and the one developed for this project is very simple.

Future work can be done on building an automated web scraper where you user can press what they want to scrape from the website. A user interface can also be developed for the web scraper so that the user can easily understand how the web scraper work.

The scraper can also be developed further internally, by adding more things to scrape and making sure that it handles the different types of data that exists on TimeEdit.

Future research can be done on how data mining methods can be applied to the dataset. If more data was scraped, or data over an entire year, different data mining methods can be applied to analyze the data set.

Research in this area could show where students succeed or fail and how different arrangement of courses affect the students. With the help of web scraping and data mining good statistics can be produces and these statistics can be used to further develop the program.

Perhaps, developing a website where the user can add a link which then gets web scraped by a Python script in the background is a good start for future work. This project shows that the potential upsides of web scraping is massive but that more research can be done on how to implement it.

## 6.6 Problems during development

During the development of this project a lot of different problems occurred. Quite a few of these problems came down to how web scraping works. Web scraping in nature is a bit tricky as it relies on the website being designed and built in a way such that it is possible to scrape data from it.

TimeEdit uses a third-party log in system that redirects the user several times before the actual schedule is presented. You can reach this log in system through different routes, (*University website, TimeEdit or the link directly*), making it hard to design a program where the scraper understands how to scrape the website.

TimeEdit also uses the same class name for different types of information that is being displayed, and this information is being generated in the back end. As web scraping centers around looking for class names in the HTML code this becomes tricky.

The same types of problem apply when hard coding the link. When using a user input the program understands it some of the time, but not all times. The problem seems intermittent in a way. This comes down to the program not always understanding the URL that the user posted to the input.

Work arounds to all the above problems could have been developed with more time. The scope of this project in its infancy did not sound or feel too big, however, due to all the moving parts and the nature of how web scraping works the time had to be split between functions and goals. Because of this, compromises had to be taken and not all functions worked as first set out, or with the user friendliness desired. Even if the argument can be made that the goals have been fulfilled, they could have been so in a better and cleaner way.



## References

- [1] Wikipedia [www]. St. Petersburg (FL): Wikimedia Foundation, Inc; 2001. *TimeEdit*; [revised 2017-10-29; cited 2021-07-25]. Available from: <https://sv.wikipedia.org/wiki/TimeEdit>
- [2] Siteefy, "How many websites are there" [www] <https://siteefy.com/how-many-websites-are-there/>  
Updated 2021-08-19. Read 2021-07-25
- [3] Statista, "Digital Population Worldwide" [www] <https://www.statista.com/statistics/617136/digital-population-worldwide/>  
Published 2021-01. Read 2021-07-26
- [4] Twitter, "Twitter-API" [www] <https://developer.twitter.com/en/docs/twitter-api>,  
Read 2021-07-26
- [5] Zdnet, "Programming language python's popularity ahead of java for the first time but still trailing c" [www] <https://www.zdnet.com/article/programming-language-pythons-popularity-ahead-of-java-for-first-time-but-still-trailing-c/>  
Published: 2020-11-04. Read: 2021-07-26
- [6] Python, "Why python is a dynamic language and also a strongly typed language" [www] <https://wiki.python.org/moin/Why%20is%20Python%20a%20dynamic%20language%20and%20also%20a%20strongly%20typed%20language>  
Read: 2021-07-26
- [7] Python, "The Python Wiki", [www] <https://wiki.python.org/moin/FrontPage>  
Read: 2021-07-27
- [8] Crummy, "Making the soup" [www] <https://www.crummy.com/software/BeautifulSoup/bs4/doc/#making-the-soup>  
Read: 2021-07-26
- [9] Elite Data Science, "5 Tasty Python Web Scrping Libraries" [www] <https://elitedatascience.com/python-web-scraping-libraries>  
Read: 2021-09-12

- [10] Php, "What is php?" [www]  
<https://www.php.net/manual/en/intro-what.php>  
Read 2021-07-29
- [11] Sqlcourse, "What is SQL?" [www]  
<http://www.sqlcourse.com/intro.html>  
Read 2021-07-29
- [12] Phpmyadmin, "Bringing MySql to the web" [www]  
<https://www.phpmyadmin.net/>  
Read 2021-07-30
- [13] Atlassian, "User Stories with Examples and Template" [www]  
<https://www.atlassian.com/agile/project-management/user-stories>  
Read: 2021-09-12
- [14] Projekttleding, "Kravspecifikation", [www]  
<https://projekttledning.se/kravspecifikation>  
Read: 2021-09-12, Published: 2018-05-10
- [15] LinkFang, "Funktionsanalys (Teknik)", [www]  
[https://sv.linkfang.org/wiki/Funktionsanalys\\_%28teknik%29](https://sv.linkfang.org/wiki/Funktionsanalys_%28teknik%29)  
Read: 2021-09-13
- [16] Saurkar, A. S., Pathare, K. P. & Gode, S. G. (2018). [article]  
An Overview On Web Scraping Techniques And Tools.  
*International Journal On Future Revolution In Computer Science & Communication Engineer*, 4(4), 363 – 367.  
<http://www.ijfrsce.org/index.php/ijfrsce/article/view/1529/1529>
- [17] SCM de S Sirisuriya, (2015). [article]  
A Comparative Study on Web Scraping.  
*General Sir John Kotelawala Defence University, Proceedings of 8th International Research Conference, 2015, 6 pages.*  
<http://ir.kdu.ac.lk/bitstream/handle/345/1051/com-059.pdf?sequence=1&isAllowed=y>
- [18] Project Manager "The ultimate guide to a gant chart" [www]  
<https://www.projectmanager.com/gantt-chart>,  
Read: 2021-08-05
- [19] Intersoft consulting, "General Data Protection Regulation", [www]  
<https://gdpr-info.eu/>,  
Read 2021-08-06

- [20] United Nations, "*The Paris Agreement*" [www]  
<https://unfccc.int/process-and-meetings/the-paris-agreement/the-paris-agreement> Read 2021-08-06  
Read 2021-08-06
  
- [21] Crummy, "*find\_parents() and find\_parent()*" [www]  
<https://www.crummy.com/software/BeautifulSoup/bs4/doc/#find-parents-and-find-parent>,  
Read: 2021-08-26