Harto Kuukka

DEVELOPMENT OF PROJECT MONITORING SOFTWARE

Bachelor's thesis

Degree Programme in Information Technology

2020

XAMK

South-Eastern Finland
University of Applied Sciences

South-Eastern Finland
University of Applied Sciences

Tutkintonimike	Insinööri (AMK)
Tekijä/Tekijät	Harto Kuukka
Työn nimi	Projektinseurantasovelluksen kehitys
Toimeksiantaja	Xamk Informaatioteknologian koulutusyksikkö
Aika	marraskuu 2020
Sivut	51 sivua
Työn ohjaaja	Niina Mässeli

TIIVISTELMÄ

Opinnäytetyön tavoitteena oli suunnitella ja kehittää sovellus, jolla voi seurata ja hallita peliprojekteja. Työ toteutettiin Kaakkois-Suomen ammattikorkeakoulun informaatioteknologian koulutusyksikön toimeksiantona peliohjelmoinnin opiskelijoille sekä henkilökunnalle.

Teoria osuudessa käsitellään kehittämiseen käytetyt alustat ja ohjelmointikielet, täsmentäen päätoimintoja, mitkä mahdollistivat lopputuotteen. Suunnitteluosuudessa korostetaan responssiivisen webbisovelluksen tärkeydestä nykyaikaisia käyttöliittymiä huomioiden.

Sovellusta kehitettiin ketterällä kehitysprosessilla. Ensin esiteltiin alustavaa konseptia sovelluksen päätoiminnoista, tuntien kirjaamisesta, asiakkaalle. Sovellusta lähdettiin rakentamaan toiminto kerrallaan, lisäämällä ominaisuuksia aina niiden tullessa esille, vahvistamalla projektin suunnan ajoittain kokouksissa. Sovelluksen siirtyessä tuotantoon, käyttäjien palautteen avulla sovellusta voitiin optimoida vielä paremmaksi.

Opinnäytetyöllä tuotettiin webbisovellus. Hyödyntäen olemassa olevia käyttötunnuksia, joita opiskelijat ja henkilökunta ovat käyttäneet, sovelluksella pystyy kirjaamaan nopeasti projektitunteja ja seurata niiden etenemistä datan visualisointityökaluilla, tukien projektiluontoista opiskelua. Sovellusta voi myös potentiaalisesti laajentaa muihin koulutusohjelmiin.

Avainsanat: HTML, JavaScript, ohjelmistosuunnittelu, PHP, projektinhallinta, tietokantaohjelmat

XAMK
South-Eastern Finland
University of Applied Sciences

Degree	Bachelor of Engineering
Author (authors)	Harto Kuukka
Thesis title	Development of Project Monitoring Software
Commissioned by	Xamk Department of Information Technology
Time	November 2020
Pages	51 pages
Supervisor	Niina Mässeli

ABSTRACT

The objective of this thesis was to design and develop a software for managing as well as monitoring game projects. The work was carried out for the students and faculty of the Game Programming degree.

The theory section of this thesis covered the platform and tools used, the key features that made the final product possible. The design aspect was also discussed, the web application required to be responsive, clear and useable on modern platforms.

The development process was conducted in an agile manner. First, a proof of concept of the application’s main loop was demonstrated to the client. From there, the app was built piece by piece, adding features as they came up and having regular meetings to confirm fresh implementations and reaffirm the course of the project. Once the software went into production, live feedback from users was leveraged to optimize the product.

The result of this project was a web application accessible with the same login credentials the students and staff already used, offering ease of access to submit their projects and work hours, as well as supporting their project-oriented studies with live data visualization tools and potential for expansion to other degrees programmes.

Keywords: database management, HTML, JavaScript, PHP, project management, software development

CONTENTS

1 INTRODUCTION 7

2 OVERVIEW 8

2.1 Xamk 8

2.2 Xamk Game Programming – Game Projects 8

2.3 Research Problem 9

2.4 Similar Thesis 10

2.5 Similar Software 10

3 PLATFORM AND TOOLS 10

3.1 Web Applications 10

3.2 JavaScript 12

3.2.1 Objects 13

3.2.2 Prototyping 13

3.2.3 jQuery 14

3.2.4 Asynchronous JavaScript 15

3.3 PHP & MySQL 17

4 DESIGN 19

4.1 Responsiveness 19

4.2 App structure 20

4.3 Dark mode 24

4.4 App communication 26

5 IMPLEMENTATION 28

5.1 Hour logging procedure 29

5.2 Flexible code 32

5.3 Data visualization 33

5.4 AJAX in UX 40

5.5 Applied MySQL methods 41

5.5.1 Date & Time conversions 41

5.5.2 Partitioning 42

5.5.3 Case statements 44

6 RESULTS 44

7 CONCLUSIONS 47

REFERENCES 48

LIST OF FIGURES 50

ABBREVIATIONS

AJAX	Asynchronous JavaScript And XML
CSS	Cascading Style Sheets
CSV	Comma-separated values
DML	Data Manipulation Language
DOM	Document Object Model
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
JS	JavaScript
PHP	PHP: Hypertext Preprocessor
SQL	Structure Query Language
UX	User experience
XML	Extensive Markup Language

1 INTRODUCTION

Wells (2015, 2) defines a project as ”a onetime undertaking that will result in a new product, event, or way of doing things. It will have a defined start and finish— though a project’s duration could be anywhere from a few hours to many years.”

Managing and monitoring projects without a dedicated system can be incredibly convoluted. If workers are allowed to document their work hours into spreadsheets in a free-form manner, the staff receiving, reading and managing these project reports will end up with mountains of tedious (digital) paperwork. Going through different Excel sheets or text files and checking for errors takes an unnecessary amount of time from one’s already busy schedule. To make managing teams and projects easier, businesses can, and often will, resort to various services dedicated to handling these tasks. Applications such as Zapier or monday.com offer tools to automate workflow and manage projects in quick and easy steps. However, while many of these services provide a great deal of customization, sometimes the best solution is software handcrafted specifically to meet the organization’s requirements.

This thesis will cover a software for monitoring projects in an organization. The software was developed for Xamk’s Game Programming degree. The thesis is divided into the following core sections: defining development platform and tools, followed by the design procedure including the theory behind it. After these have been established, the actual developed software is reviewed and finally the results of the produced application, how it has performed, what has been gained from it and potential future development.

2 OVERVIEW

This chapter will go over the foundation of this thesis. The background of Xamk, game projects at Xamk's Game Programming degree and the research problem that arose from them.

2.1 Xamk

South-Eastern Finland University of Applied Sciences – Xamk – is an institute of higher education. Xamk stands as the fifth largest university of applied sciences in Finland, with four campuses located in Kotka, Kouvola, Mikkeli and Savonlinna. Xamk offers education in eight fields of study with over 70 degree programmes. One of the degree programmes at the information technology department of education (ITY) in Xamk is Game Programming in Kotka. (Xamk 2020.)
According to a survey conducted in 2019, graduated students rated Xamk to have the best education in the field of Information Technology (Student feedback for UAS at the graduation phase (AVOP), UAS qualification 2019).

2.2 Xamk Game Programming – Game Projects

The Bachelor's Degree in Game Programming in Kotka offers an expansive array of courses, covering multiple aspects of game/software development from designing, testing, modelling to programming. GameLab learning environment, located at the Kotka campus, offers industry-standard tools for students to enable working on games and software at a professional level. One of the core implementations in this degree programme is titled "Game Projects". Throughout the four years / 240 credits of the degree, students work on projects, developing games and software either for themselves or in a lot of cases to a proper business or a client. The goal is to build your portfolio with tangible work and accomplishments during studies. On the education side, this is measured in credits. A single credit equates to 27 hours of work, and the total required amount of credits reserved for the game projects is 30, which in turn equates to 810 hours. Traditionally, students have to report their progress in increments of 75 hours of work or every four months. This is how the faculty is able to keep track of everyone’s progress and yield credits in increments of 5, in other words for every 135 hours of work. The 30 credits are divided into six Game Project ”courses”, which mainly serve as checkpoints to let the students know how things are proceeding.

2.3 Research Problem

This thesis aims to explain the process of developing a web application for project monitoring, define the tools and programming languages used and how they have been utilized. The subject of database management will also be discussed as well as the topic of data visualization. The end product will be software that supports project-oriented studying and improves project monitoring.

Before, the game projects were handled utilizing the university platforms and Excel spreadsheets or other methods of logging work hours. The issue with these was that it was difficult for both teachers and students to know whether a student was actively working on a project, what projects were going on, did anyone need additional members for their team, and so on, excluding the handful that worked on their projects at the GameLab. However, a great number of students preferred to work remotely at home. The global epidemic of COVID-19 that pushed everyone to work remotely in March of 2020 meant that following everyone’s progress became even more sparse. To solve this problem, the development of a custom project monitoring software was initiated. The requirements for this application were to provide a simple and clean interface for users to manage their projects and work hours with minimal clicks and hassle. It should be compatible on desktop browsers as well as mobile devices (Android and iOS respectively). The user (be it a student or a teacher) should not have to question what the different buttons or input fields of the system mean and any action the user performs should be followed by a response from the application that communicates to the user whether their action has succeeded.

2.4 Similar Thesis

Aadamsoo (2010) wrote a thesis on web-based project management systems, in which she found that being a programmer and a developer of a web application for project management, in addition to writing code, it is important to understand the client’s needs and analyse the requirements to design and develop a high-quality product that meets their criteria. What she ended up with was a software built on top of a pre-existing, open-source application called TRAC. Their solution utilized Python programming language with PHP and HTML.

2.5 Similar Software

Researching into existing applications that would have achieved similar results as the project monitoring software, the most prominent option was monday.com. An application that offers a multitude of tools, widgets and integrations to manage workflows and the ability to develop your own with Node.js. In the end, developing an entirely new application was decided, removing any additional costs that would incur from utilizing third-party software and licenses, as well as retaining the ownership over any data and code in-house.

3 PLATFORM AND TOOLS

While this thesis will not delve into the intricacies of programming and how web applications function, the following paragraphs will, however, cover some of the base concepts and utilities of web services and JavaScript.
This project was largely done in Brackets, a lightweight, open-source text editor. Originally established by Adobe as a community guided project for web development (Brackets 2020).

3.1 Web Applications

The structure of a web application can be broken down into three layers: client-side, server-side and database (w3schools). In this sample case (Figure 1), the client makes an HTTP request to the server, a server returns an HTML file and the browser renders the file on the screen.

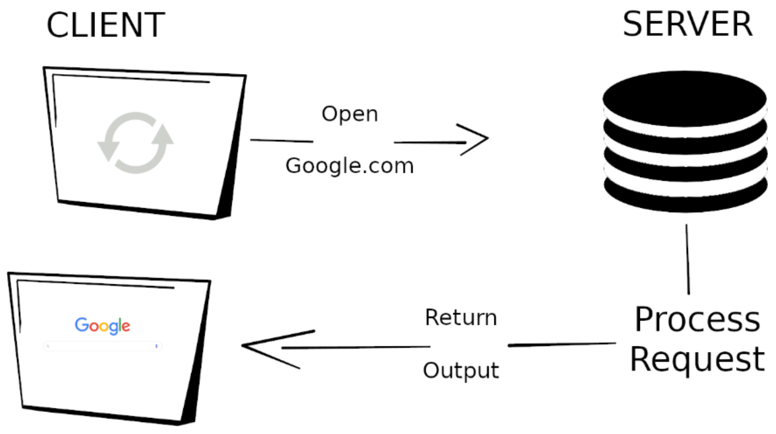
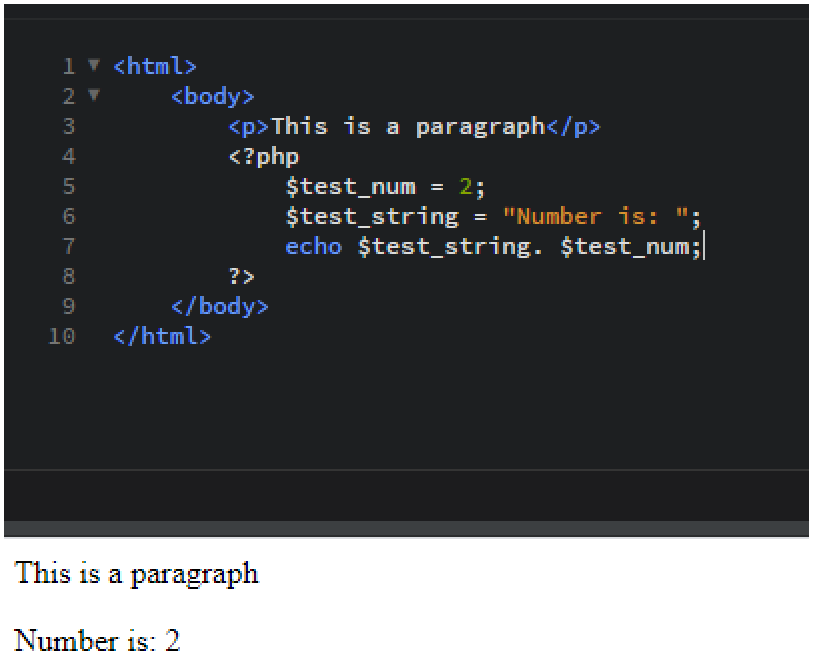


Figure 1 HTTP Request

Of course, that is an overly simplified version of the process. Modern web development is more flexible and dynamic. Instead of loading entire web pages with every HTTP request, techniques such as AJAX are used to provide a faster and better user experience by handling requests in the background without requiring a full-page load (Joshi 2010, 67). For the software covered in this thesis, the counterpart on the server-side for these AJAX requests is PHP scripting language. The strengths of PHP lie in that it can output HTML while keeping all the executed code on the server side, hidden from the client-side (Figure 2).



```
<html>
    <body>
        <p>This is a paragraph</p>
        <?php
            $test_num = 2;
            $test_string = "Number is: ";
            echo $test_string. $test_num;
        ?>
    </body>
</html>
```

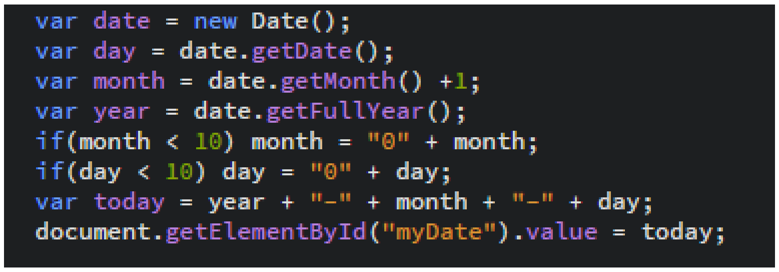
```
This is a paragraph

Number is: 2
```

Figure 2 PHP output to browser

If you were to view the source of the page on browser, you would be able to see the HTML elements (<html>, <body>, <p>), however, nothing encased in <?php?> will be visible on the client side excluding the contents that the PHP code prints to the client with the echo statement.

3.2 JavaScript

JavaScript (later JS) is a lightweight, single-threaded scripting language widely used for web pages, but also non-browser software as well. It contains features such as object orientation, prototyping and much more (MDN web docs 2020). This chapter will cover specifically the aforementioned aspects of JS as they will become relevant later. Single-threaded means that JS executes each command one at a time, in order. Computers in essence are machines waiting for the user's commands, execute the command and then wait for the next one. Programs are a collection of commands, with a potential for the user to choose which command to issue. Using the example seen in Figure 3. A user has opened a web page, and on this web page a script is executed. First, it creates a *Date* object (objects covered next), then extracts the day, month and year from it to separate variables. Next, it checks if month and day -values are below 10, in which case it adds a “0” in front of them (2 => 02). After this, the code creates a variable *today* and forms a string (sequence of characters) as its value out of the previous variables. And finally, it finds an HTML element on the page with the Id -attribute “myDate” and grants it the value of “2020-09-11” (if the date in question was September 11th 2020). To trail back to the concept of single-threaded programming, while the web page is executing the code sequence above, it would not be running any other scripts or functions on the page at the same time.



```
var date = new Date();
var day = date.getDate();
var month = date.getMonth() +1;
var year = date.getFullYear();
if(month < 10) month = "0" + month;
if(day < 10) day = "0" + day;
var today = year + "-" + month + "-" + day;
document.getElementById("myDate").value = today;
```

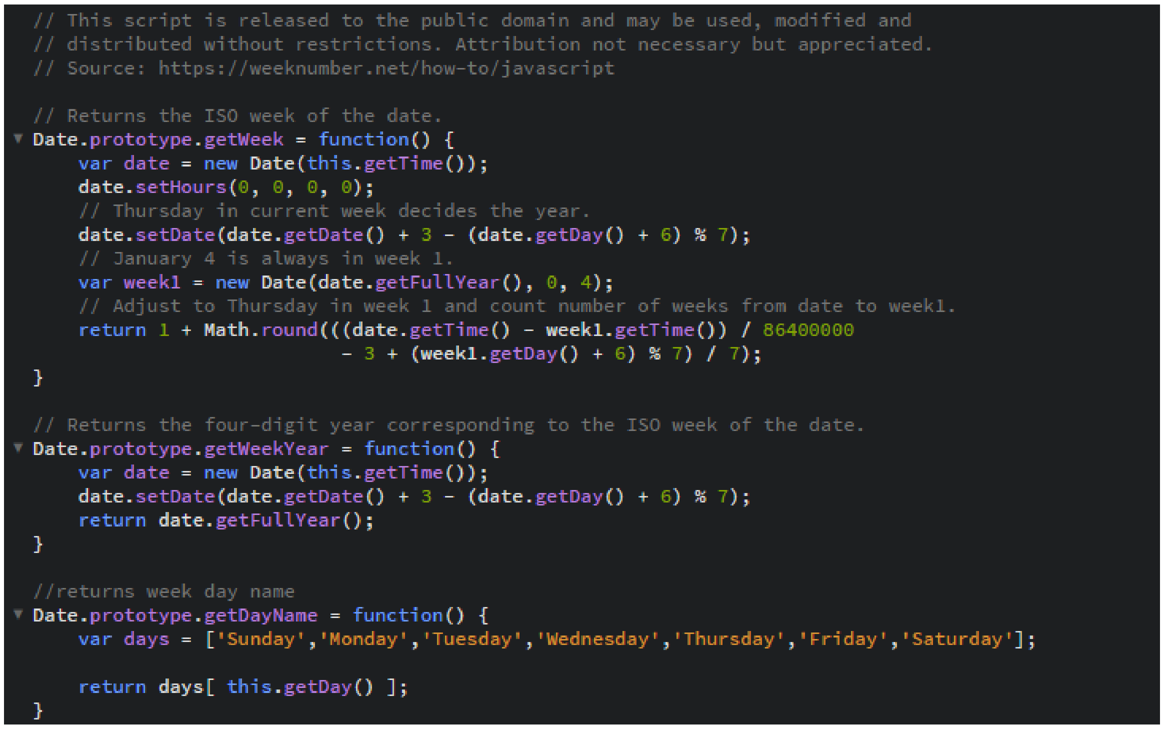
Figure 3 JS sequence of date object

3.2.1 Objects

Objects in programming are essentially capsules of predefined *types* of information or functions. When the program creates an object, it generates an instance of that object. (MDN web docs.) Using the example in Figure 3, *Date()* is an object in JS that contains a number that represents the amount of time elapsed since midnight January 1st 1970 UTC (Coordinated Universal Time) in milliseconds. The Date object also has several methods that return the date or specific elements of the date in a more readable format. In the example, the various get- methods return the day (1-31), month (0-11) and year of that Date instance. The *getMonth()* method returns a value between 0 and 11, 0 being January and 11 being December. Therefore, to avoid confusing the user, the month value is incremented by one in the above example. There are other ways to display the date in a more readable format, which will be covered later.

3.2.2 Prototyping

All objects in JS inherit their properties and methods from a model of that object referred to as a prototype. In a more traditional class-based language such as C++, the properties and functions are declared in that class script, no new functions or properties can be added *to* the class itself afterwards. JS is a more dynamic language, through prototyping, it is possible to add new properties to object constructors (MDN web docs). For instance, in Figure 4, new methods are added to the Date prototype to return the week number, day name and year based on the week number.

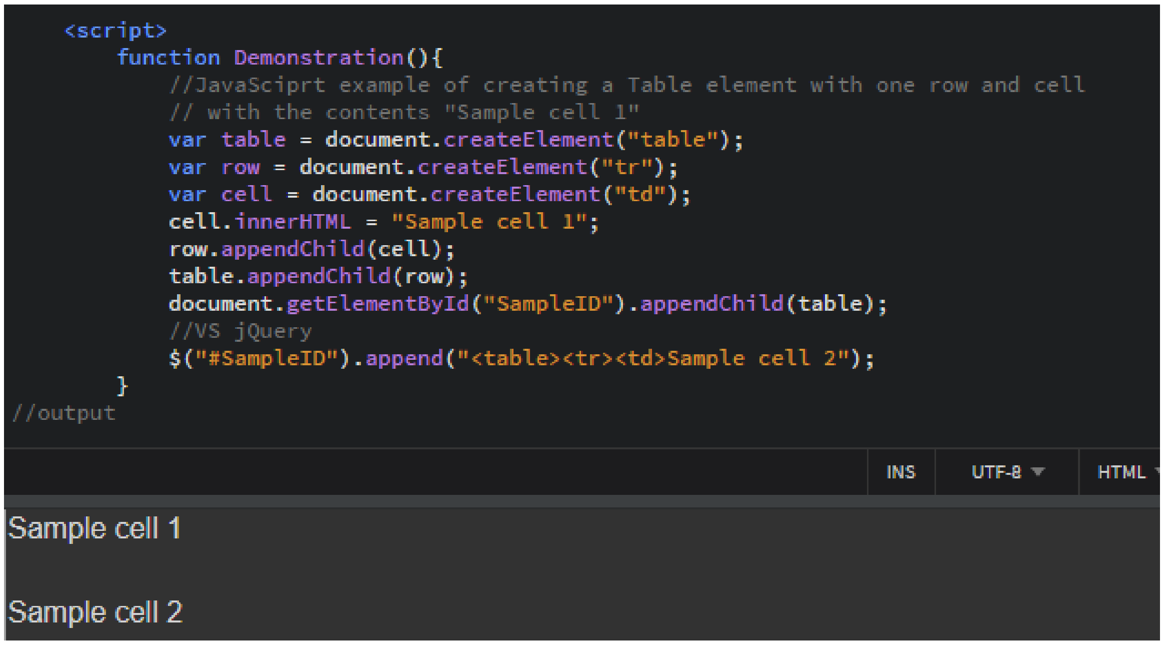


```
// This script is released to the public domain and may be used, modified and
// distributed without restrictions. Attribution not necessary but appreciated.
// Source: https://weeknumber.net/how-to/javascript

// Returns the ISO week of the date.
Date.prototype.getWeek = function() {
    var date = new Date(this.getTime());
    date.setHours(0, 0, 0, 0);
    // Thursday in current week decides the year.
    date.setDate(date.getDate() + 3 - (date.getDay() + 6) % 7);
    // January 4 is always in week 1.
    var week1 = new Date(date.getFullYear(), 0, 4);
    // Adjust to Thursday in week 1 and count number of weeks from date to week1.
    return 1 + Math.round(((date.getTime() - week1.getTime()) / 86400000
                          - 3 + (week1.getDay() + 6) % 7) / 7);
}

// Returns the four-digit year corresponding to the ISO week of the date.
Date.prototype.getWeekYear = function() {
    var date = new Date(this.getTime());
    date.setDate(date.getDate() + 3 - (date.getDay() + 6) % 7);
    return date.getFullYear();
}

//returns week day name
Date.prototype.getDayName = function() {
    var days = ['Sunday','Monday','Tuesday','Wednesday','Thursday','Friday','Saturday'];

    return days[ this.getDay() ];
}
```

Figure 4 Prototype date

3.2.3 jQuery

jQuery is not a programming language itself, but rather an extensive JavaScript library designed to make writing JavaScript more efficient. jQuery is a part of the OpenJS Foundation, a community consisting of 32 open source JavaScript projects (openjsf.org). Figure 5 demonstrates the effectiveness of jQuery compared to standard JS. In that code snippet, the program creates an HTML table element with one row and column, and places the contents within an element with the Id “SampleID”. The difference in the amount of lines in the code is evident.



```
    <script>
        function Demonstration(){
            //JavaSciprt example of creating a Table element with one row and cell
            // with the contents "Sample cell 1"
            var table = document.createElement("table");
            var row = document.createElement("tr");
            var cell = document.createElement("td");
            cell.innerHTML = "Sample cell 1";
            row.appendChild(cell);
            table.appendChild(row);
            document.getElementById("SampleID").appendChild(table);
            //VS jQuery
            $("#SampleID").append("<table><tr><td>Sample cell 2");
        }
//output
```

```
Sample cell 1

Sample cell 2
```

Figure 5 jQuery example comparison

jQuery is easy to tell apart from standard JS. jQuery generally begins the selector prompt $(), with the selected element or class defined inside the brackets as a string (such as ”input[type=’text’]”). While a great deal of code in this particular project has been compressed to use jQuery, there still remains some traditional JS.

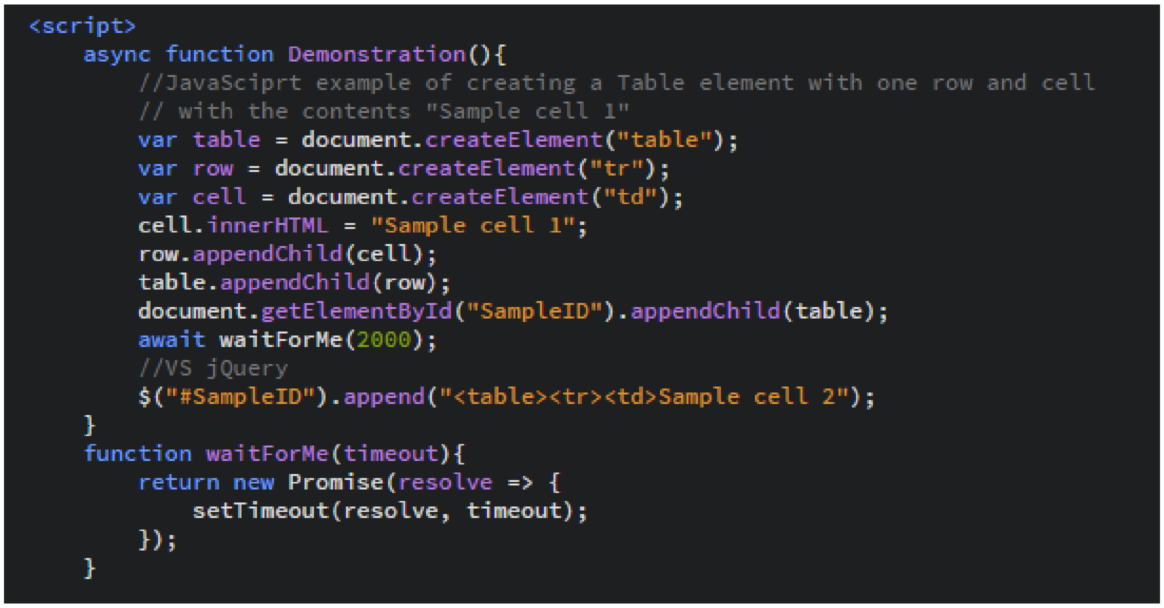
3.2.4 Asynchronous JavaScript

As established, JS is single-threaded. Each task in the program has to finish before the next one can run. Generally, this works fine, computers are fast enough to calculate some numbers and create HTML elements on a web page without hindering the user experience. The problem arises when the program tries to load an image or connect to a server, tasks that may take a bit longer than a couple of milliseconds. JS being single-threaded means that the web page is blocked from doing anything else until the task of loading the image is finished. To combat this problem, browsers are able to run certain operations asynchronously. An example of this is the Promise object. A Promise essentially represents the *eventual* completion or failure of an operation. It allows the user to continue with other activities on the page while the browser waits for the Promise to return its result in the background. (MDN web docs.)

To enable asynchronous operations on a function, the prefix “async” must accompany the function declaration.

function Test() { => async function Test() {

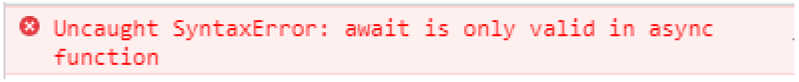
As an example, we can look at Figure 6. The previous Demonstration function has been altered to be asynchronous. In between the printing of ”Sample cell 1” and ”Sample cell 2”, there’s an arbitrary delay in the form of a *waitForMe()* function. The *waitForMe* returns a Promise in which the program waits 2 seconds using the *setTimeout()* method. In a browser, ”Sample cell 2” appears 2 seconds after “Sample cell 1”, during which the user can engage with the other features of the page. The await operator is used here to achieve the delay properly. The await pauses the asynchronous function until the Promise is resolved.



```
<script>
    async function Demonstration(){
        //JavaSciprt example of creating a Table element with one row and cell
        // with the contents "Sample cell 1"
        var table = document.createElement("table");
        var row = document.createElement("tr");
        var cell = document.createElement("td");
        cell.innerHTML = "Sample cell 1";
        row.appendChild(cell);
        table.appendChild(row);
        document.getElementById("SampleID").appendChild(table);
        await waitForMe(2000);
        //VS jQuery
        $("#SampleID").append("<table><tr><td>Sample cell 2");
    }
    function waitForMe(timeout){
        return new Promise(resolve => {
            setTimeout(resolve, timeout);
        });
    }
```

Figure 6 async wait example

Again, the Demonstration function requires the ”async” prefix, otherwise trying to run the program yields an error (see Figure 7 below).

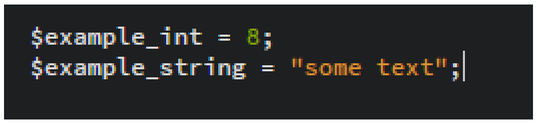


```
Uncaught SyntaxError: await is only valid in async
function
```

Figure 7 async syntax error

3.3 PHP & MySQL

PHP is an open source, server-side scripting language, programmed in C. To quote Andrew Caya (2018, 82): “PHP 7 is in itself a major optimization” from his book on Mastering the Faster Web with PHP… Where it is said that version 7 was significantly rewritten to run faster than previous iterations. The accessibility of PHP makes it one of the top languages for web developers (Joshi 2010, 21). PHP, similar to JS, is a fairly loose language, meaning that variables can be declared without specifying their type. In C++, you would declare an integer with the “int” identifier or a string variable with “string”. While on PHP you simply declare a variable and the type is more based on what is on the right side of “=” (see Figure 8).



```
$example_int = 8;
$example_string = "some text";
```

Figure 8 PHP variables

As a result, PHP acts as an efficient intermediary between the client and the server, quickly processing the data client submits and responds back. PHP, designed as a server-side language, contains effective methods to scrap unwanted content from the submitted data and prepare query statements for connecting to the database.

Databases are collections of data, recorded facts of implicit meaning. These days databases are practically essential, as most of our daily activities rely on having access to a persistent storage of data. Be it for logging in to email or bank account or to get your saved data on a mobile game. Database systems are computerized repositories of data files. Users are able to do requests to the system to perform various actions such as add new data, update existing data and so forth. These are done through Data Manipulation Language (DML), commonly known as queries. Programs can communicate with databases using a multitude of query languages, depending on the system used. One such language is SQL or Structured Query Language. A factor that may play into SQL’s and its many derivatives popularity is the ease of use. Many commands

resemble spoken English, making it very human friendly. (Vidhya 2016.) One of the derivatives of SQL is an open-source relational database system called MySQL, which is what the backend of this thesis utilizes.
Here are some examples of MySQL's syntax:
Let us say we have a DB table called "Sample_table" with the following structure seen in Figure 9.

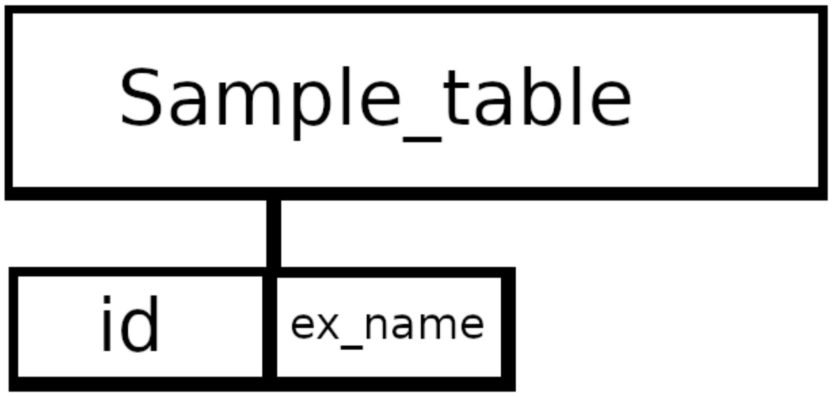
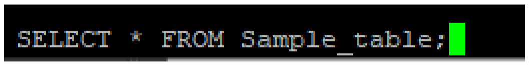


Figure 9 Sample database table

In relational databases, data is stored in the form of tables. A table comprises columns and rows, the columns are defined during the creation of the table though they can be altered after the fact. Each column requires a name and a data type. For example, in the sample table above, the id column could be an INT or integer type, while the ex_name column could be a *VARCHAR(255)* type. *VARCHAR* stands for a variable-length character string where the length of the string is defined in brackets. In the case of *VARCHAR(255)*, the row in the ex_name column can have a maximum of 255 characters. (Vidhya 2016).
To retrieve data from this table, a *SELECT* query must be performed. Figure 10 demonstrates the syntax for the *SELECT* statement.



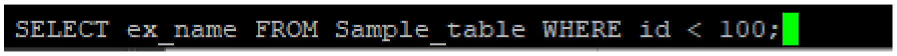
```
SELECT * FROM Sample_table;
```

Figure 10 SELECT query

The * means that you are selecting all columns from the table, however, it is possible to select specific columns as well (e.g. "SELECT ex_name FROM Sample_table;"). *FROM* is required to state which table the data is retrieved from.

The semicolon indicates an end of the statement. Additionally, it is possible to provide extra clauses to the statement to perform more specific queries. This is referred to as the *WHERE* clause. To expand the Sample_table query, in Figure 11 we request the ex_name column of all rows from Sample_table where the id column value is below 100. In later chapters, there will be more advanced MySQL statements, but the basic structure of the query remains the same:

- DML command (SELECT, UPDATE, INSERT, etc.) clause
- FROM clause
- WHERE clause



```
SELECT ex_name FROM Sample_table WHERE id < 100;
```

Figure 11 Sample table query with where clause

The server-side code snippets used in this thesis for PHP and MySQL are not direct examples of the developed project for security reasons.

4 DESIGN

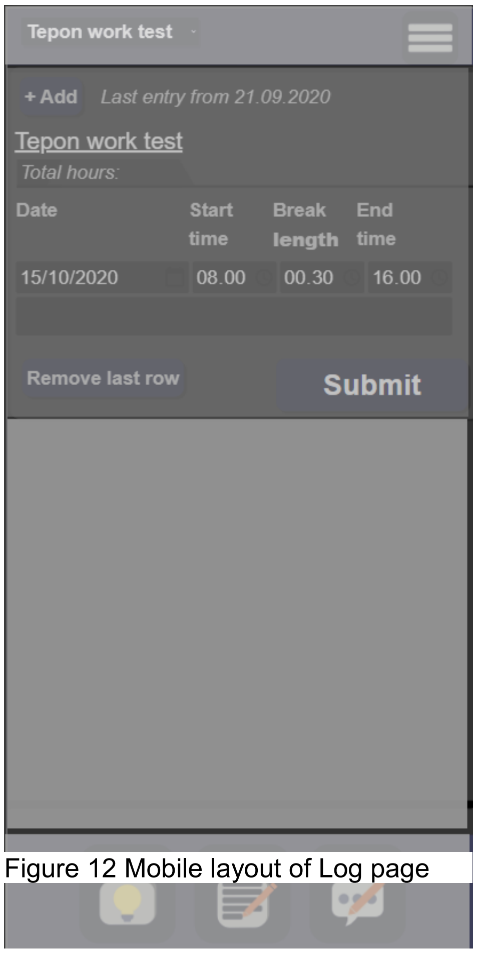
This chapter will cover the design aspect of web development from picking the colour choices to the layout of the various elements.

4.1 Responsiveness

When designing a web application, the word *responsive* is typically one of the key requirements. Websites are accessed through a multitude of different devices, of a variety of different sizes. There is no one standard screen size. Neither is there one standard method of interacting with the site. A web app should accommodate both mobile and desktop users, and have an interface that suits the input methods available to those devices. Keyboard and mouse for the computer, and touch controls for mobile devices respectively. The recommended design philosophy, referred to as *mobile-first*, often yields more appealing results than the alternative. That is due to the single restraint it imposes: keeping it simple. With the more limited screen space available on a mobile, the designers are forced to remove all unnecessary clutter and focus only on the essentials. (Carlos & Carols 2013.)

4.2 App structure

While a mobile application was on the docket for the requirements of this software, it was clear from the beginning that it would be a more trimmed version. This is a software where the main operations are submitting data and then read tables of the submitted data. It was important to establish which features are the most essential to be functional on a mobile screen, and those would be at the centre of the design process.



Last entry from 21.09.2020

Tepon work test

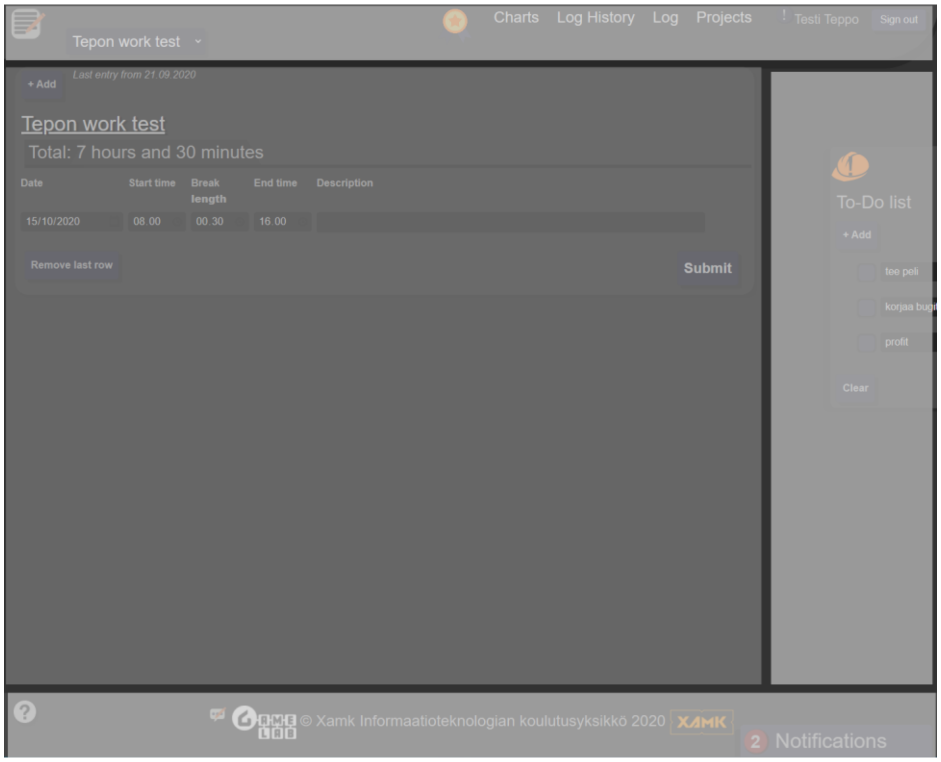
Total hours:

Date	Start time	Break length	End time
15/10/2020	08.00	00.30	16.00

Figure 12 Mobile layout of Log page

Figure 12 shows the position of different elements on the Log hours page, where the darker container is the prioritized content. The top and bottom rectangles are the header and footer sections used for navigation on the site and to access other universal features, while the lighter grey box in the middle is free space the main content can leverage.

To compare it to the desktop view in Figure 13, the main content takes full advantage of the extra screen space, some extra features are revealed on the right-side bar and the navigation tools are taken out of the footer and menu icon and laid out on the header. The structure of the page reflects this design mentality, at the root of the body on every page are containers for header, content and footer. The *content* has its width property set to 100%, meaning it takes the entire width of the window, this allows all the *child* content inside it to use the parent width as a reference point.



Date	Start time	Break length	End time	Description
15/10/2020	08.00	00.30	16.00	

Figure 13 Desktop log page

Figure 14 demonstrates the parent-child method of handling content layout. The admin page has many functions, each separated into small boxes which themselves are placed into containers that have a width of 30% relative to the parent content, with a minimum width defined in pixels. This results in a grid-like layout with a bit of space between each column. The minimum width is there to prevent the columns from grouping too closely together. This uses the flex display property, with the flex-direction being ”row”. The flex results in each direct child element within the flex parent to be positioned next to the previous without having to fiddle with exact measurements. On mobile, the rows take 100% of the screen width, instead of this multi-column structure on a single screen the interface takes the form of a swipe left and right to navigate between them.

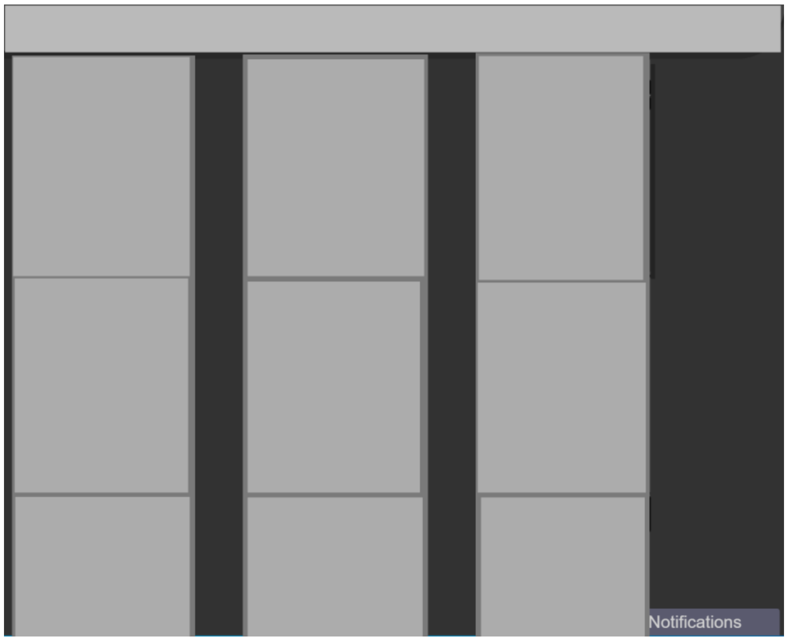


Figure 14 Admin page layout

The projects page, where users can view their own projects, create new projects as well as view all active projects that others have created and apply for them, the overall layout is largely the same on any device, shown in Figure 15.

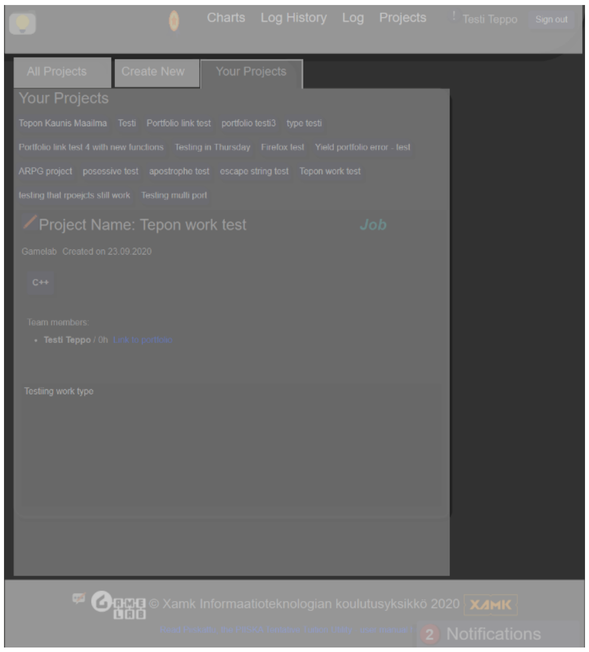


Figure 15 Projects page layout (general)

There are three tabs: All projects -tab, Create New -tab and Your Projects -tab. Clicking these tabs toggles which content is shown below them. On a wider screen, however, Your Projects -section is detached from the main content and aligned to the right of it (Figure 16). As that particular section contains information about your projects, having it on the side, while you are creating new projects or applying for someone else's, is a convenient addition, allowing the user to view two tabs at once.

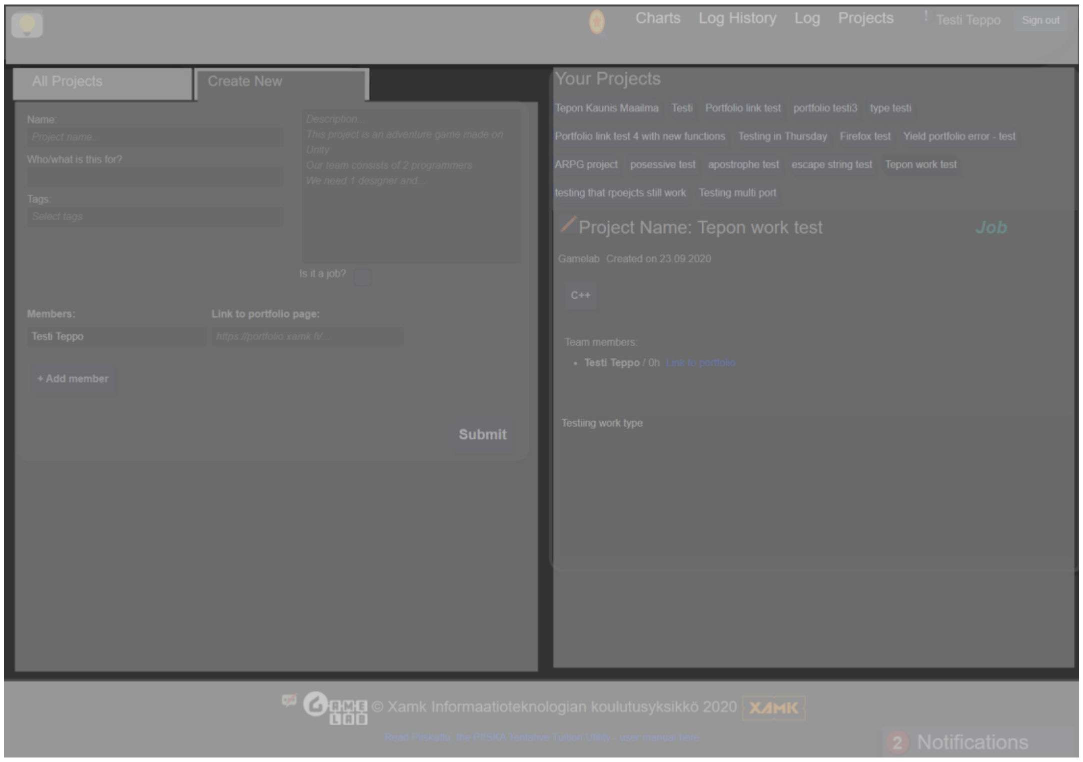


Figure 16 Projects page layout (wide screen)

The project approval page and log history page both share a similar function: viewing data in a table format. As a result, the layout on these pages is largely the same, just slightly different form of content.

To reiterate, the goal was to define universal CSS classes that could be used on any page and achieve the same result. For ease of development for any current or future developer of this software, a stylesheet reference was created. This reference document essentially lists all the classes and styles that can be used anywhere, describing which stylesheet to find them on and what each class achieves, organized by their stylesheet and context.

4.3 Dark mode

In addition to mobile-first, another thing to consider on the design is who the primary user group is going to be. The Project Monitoring Software is geared towards game programming students, of which a large portion are gamers. One common factor in both of these groups is the preference for darker backgrounds, “dark mode” as it is colloquially known. Dark mode is a theme used in various applications. Its primary indicators are bright coloured text on a dark background, the stark contrast to this is known as “light mode”: white background, black text. Dark mode has become increasingly more popular in the past decade. Most applications allow the user to toggle between these two modes, but some of the more popular apps have dark mode on by default, examples of these being the movie and TV show streaming service Netflix, music app Spotify, PC’s largest video game digital distribution service Steam and the ever-growing instant messaging app Discord. The appeal of dark mode is in that it looks sophisticated and stylish. Eyes are more drawn to highlighted, vibrant content against the dark background than they would be against the white background. A study conducted by Aleman et al. (2018) on reading and myopia suggests that reading black text on white background for prolonged periods of time may risk myopia, while reading in dark mode might inhibit myopia. Myopia, also known as near-sightedness, is a condition where the eyes struggle to focus when looking at a long distance. Bright, white screens may also be quite straining for the eyes. A Harvard Health Letter (updated in July 2020), though largely focused on the effects of blue light, states that exposure to bright lights in the evening may hinder sleep.

A small query was conducted on the game programming students of Xamk on their preferred theme of choice regarding Light mode and Dark mode for the common applications used in this field of study. These applications being Microsoft’s Visual Studio and Teams, Brackets, Discord as well as YouTube was added for good measure. The results indicate that upwards of 90% of the students and faculty in Xamk Gamelab use Dark mode for Visual Studio and Discord. Discord has this on by default and Visual Studio, upon launching it for the first time, displays a theme selector, which likely factor in why those two, in particular, had high dark mode usage. Roughly 70% use Dark mode on YouTube where the toggle for the two themes is displayed under the user's profile picture (Figure 17). Teams and Brackets had around 50-50 results. What could be surmised from this query is that the more effort the user has to put in to change the theme of an application, the less likely they are to do it. Additionally, Visual Studio is used significantly more than Brackets (a code editor designed for web development), which also played a part in users adjusting the visual presentation of the software to their liking.

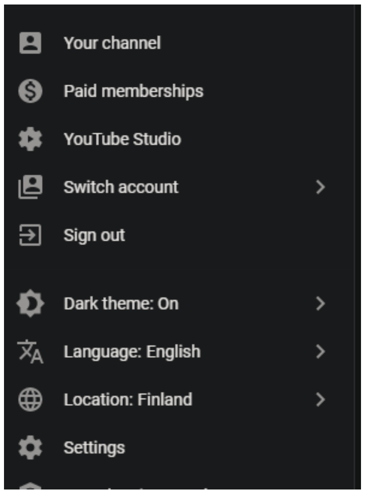


Figure 17 YouTube dark theme

To adhere to these modern design concept, the project monitoring software was designed with a darker colour scheme from the beginning (see Figure 18). However, it is not as pitch black as some other popular platforms to be suitable for usage in daytime, but still to be easy for the eyes for those who log in their work hours late at night. The background colour on the bottom layers is generally darker than the lighter grey elements containing information for the user to interact with.

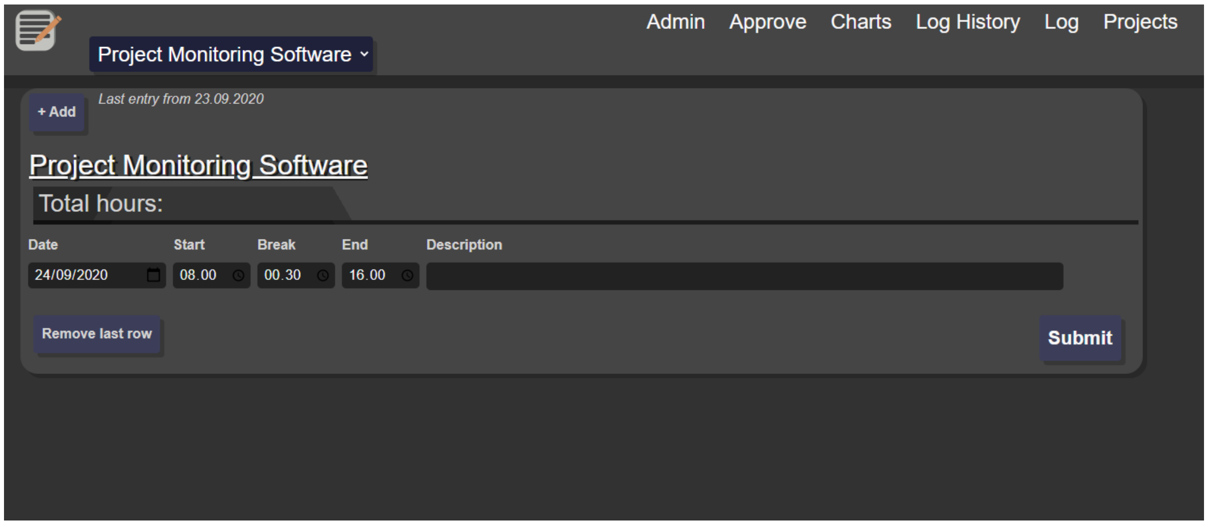


Figure 18 Project Monitoring Software design - background colour scheme

4.4 App communication

Applying the concept of drawing a person’s attention to brighter containers on dark layouts. The lighter grey containers generally emit a shadow over the darker background, *lifting* them off the layout to further pop them out. Input fields then revert to the darker tone to highlight the white text in them. On the project monitoring interface, buttons and dropdown selectors portray a bluish colour. All button instances that submit a form, such as a new project or a work hour submission, are located on the bottom right corner of their respective containers for consistency. The reason for this decision is that typically, the text is read from left to right, from top to bottom. The submit button being located at the bottom right corner creates a natural flow to filling up a form. The common colour associations of green = positive, red = negative and yellow = danger. When a project is active, it has a green “Active” as a status, indicating everything is ok. The initial pending status for projects has yellow, meaning it is waiting for approval. Red is reserved for “Rejected” status, something about the project has not met the correct criteria

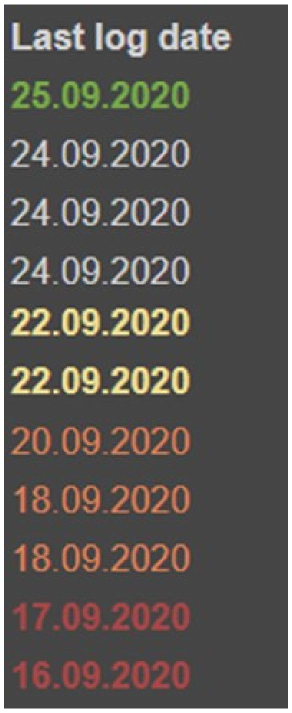

Last log date
25.09.2020
24.09.2020
24.09.2020
24.09.2020
22.09.2020
22.09.2020
20.09.2020
18.09.2020
18.09.2020
17.09.2020
16.09.2020

Figure 19 Last log date

and thus needs to be fixed. Similarly, an admin can observe when the users last logged in their hours (Figure 19, names left out for privacy reason),

and on this list, the colour of the text is highlighted in different colours based on how many days it has been since the last log:

- Green – logged hours today.
- White – logged within the last 3 days.
- Yellow – logged within the last 3-5 days.
- Orange – logged within the last 6-7 days.
- Red – it has been over a week since the user last logged in work hours.

With this, the admin (teacher) can get an overall picture of students' activity with a quick glance at the list, and if needed, notify those in red. The web application colour scheme was handled by Oiva Kytömäki, another student at Xamk who also handled the mobile design under the author's supervision. The colours were picked utilizing an online tool called Coolors. Coolors can generate colour palettes, with tones complementing one another (Bianchi 2020). Figure 20 displays the colour palette of the project monitoring software's interface, excluding the grey colours used for the containers and background. The six-figure codes at the bottom of each colour represent the hexadecimal value for the colour which can be implemented into CSS and HTML.

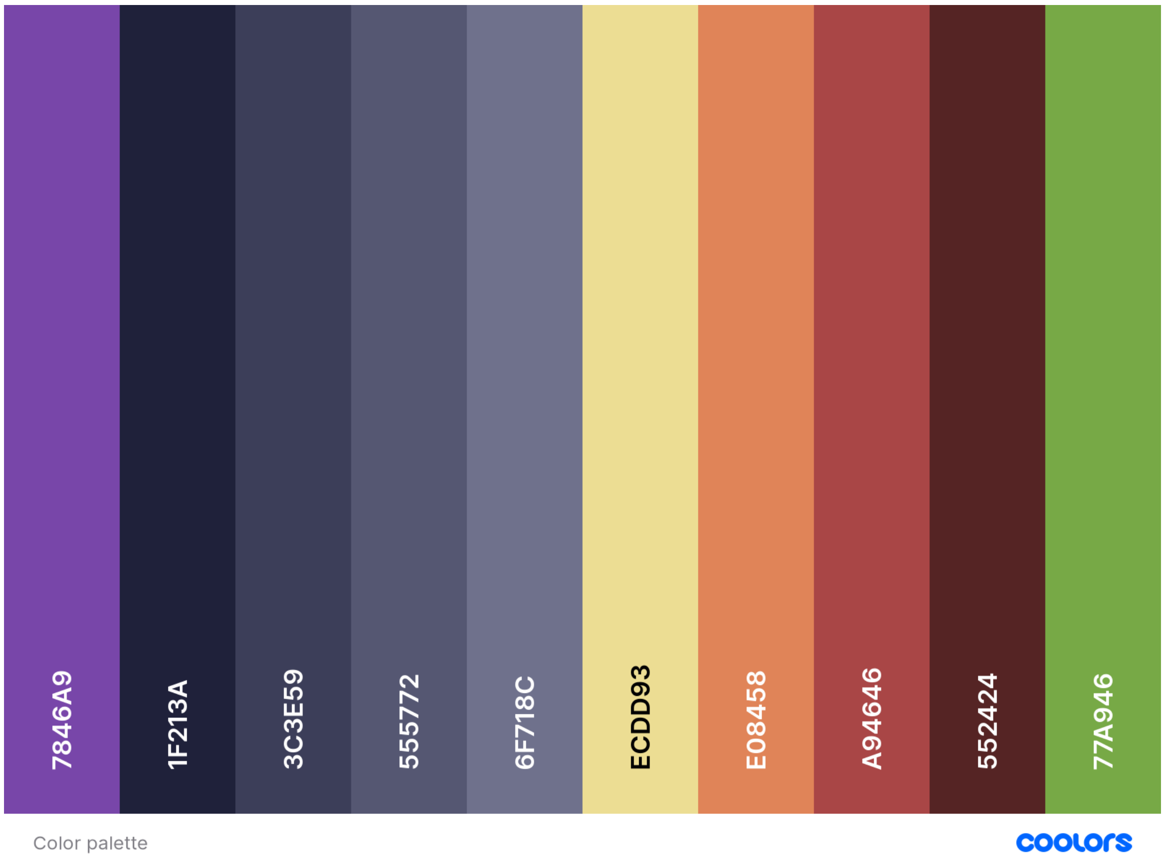


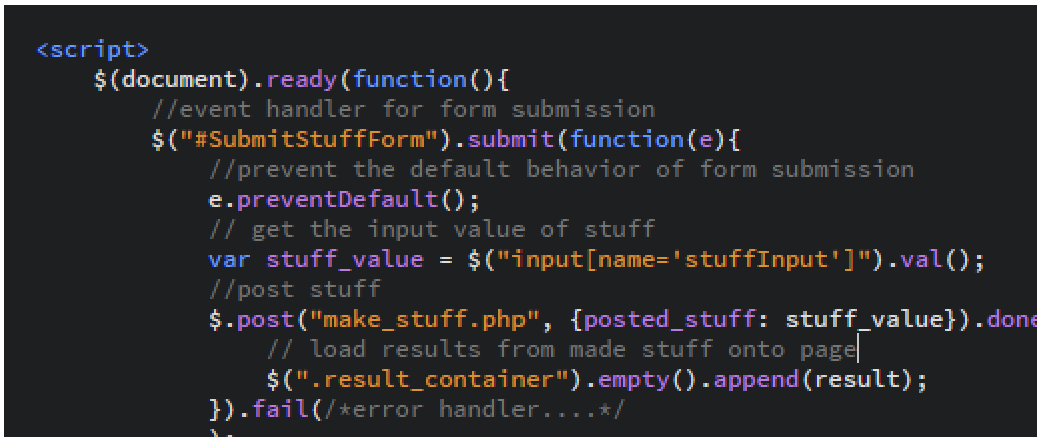
Figure 20 Coolors palette

Layout-wise, the design mentality of the software is that on desktop, the users have access to the full features of each page, with certain elements adapting to the extra screen space. To use the earlier example of Figure 18 of the log work hours template, the *Description* column stretches out the longer the window is

5 IMPLEMENTATION

Next is the practical application on the programming side of development. Concepts mentioned in chapter 3 will be applied in practice.

The combination of AJAX and PHP allows the usage of different PHP files as methods or objects in a way. Figure 21 contains a sample code of this usage. To break it down line by line, firstly an event listener is created when the page finishes loading, this event triggers when the Submit Stuff form is submitted. By default, when a form is submitted on a web page, the page redirects to a new page or to itself if no specific page is declared. This is referred to as a form action. The *.preventDefault()* method stops this action, allowing the custom form handling to proceed. In this example, a value from an input field is inserted into a variable *stuff_value*. The program performs a POST request, sending the data to the server, loading a php file ”make_stuff”. This file would get the submitted stuff value, perform some validation methods to make sure the value is safe, follow-up with a sort of query based on the value and return the results in a readable, HTML format such as a table. This sequence can be made into a function and thus would be usable on every page without repeating the same lines of code. The *make_stuff.php* file would also need to verify that when it is requested, there is a valid form submission from a logged-in user, otherwise, it should ignore users trying to access it.



```
<script>
    $(document).ready(function(){
        //event handler for form submission
        $("#SubmitStuffForm").submit(function(e){
            //prevent the default behavior of form submission
            e.preventDefault();
            // get the input value of stuff
            var stuff_value = $("input[name='stuffInput']").val();
            //post stuff
            $.post("make_stuff.php", {posted_stuff: stuff_value}).done
                // load results from made stuff onto page
                $(".result_container").empty().append(result);
            }).fail(/*error handler....*/
```

Figure 21 Example of post stuff

5.1 Hour logging procedure

The default location a student is directed to when signing in is the log hours page, as it is the most common action students would perform. The page was given a prerequisite however: having at least one active project. If the user who is trying to access this page doesn’t have any active projects to their name, they are directed to the Projects page instead where they may create new projects. Figure 22 expresses the flow of this sign-in process.

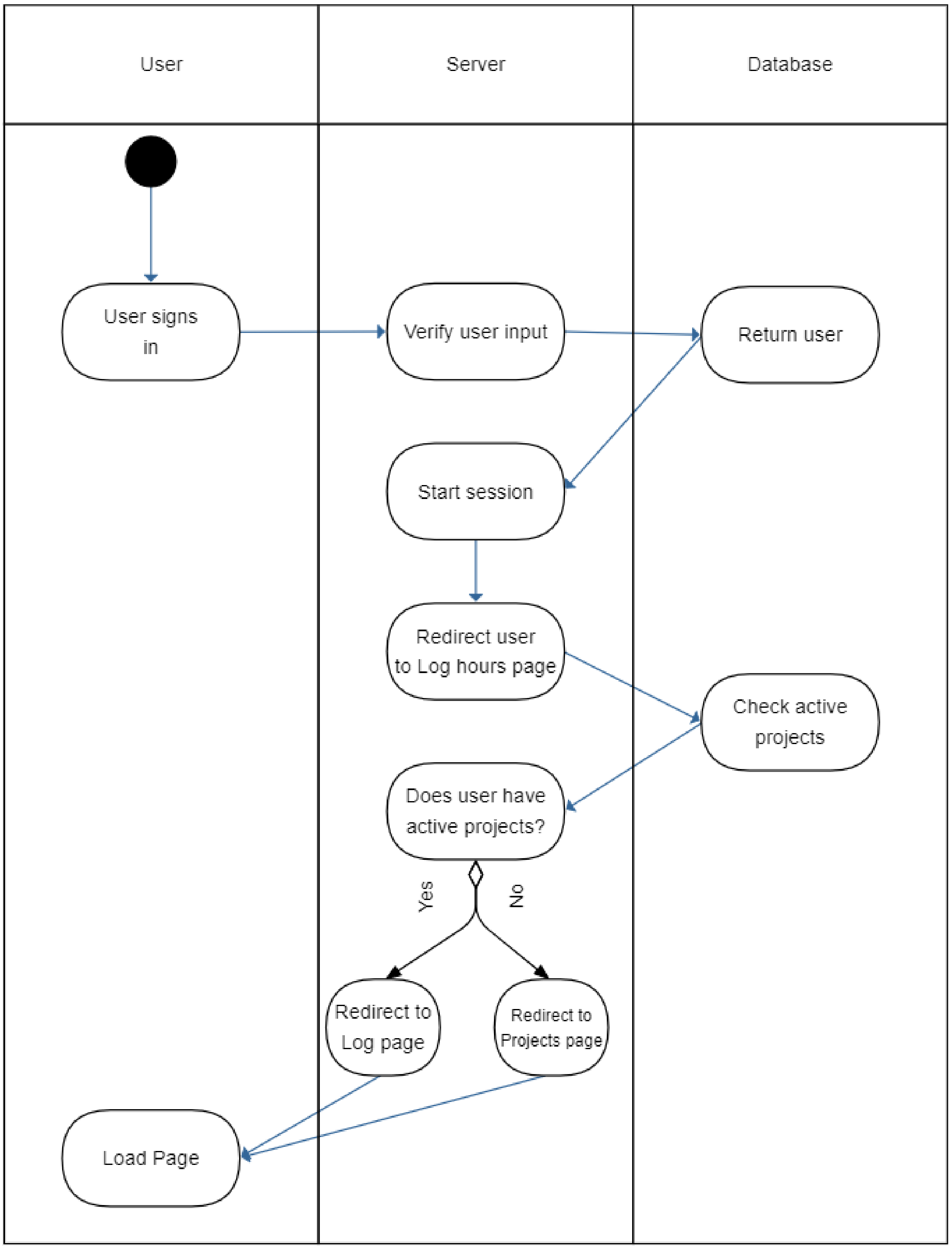


Figure 22 Sign redirection

Once the user has created a project and the project is approved, the log page can be entered. The app will request a list of the user's projects, setting the one with the most recent log entry as the default selection. Additionally, the date and the description from that last entry will be displayed on a hoverable field for the user to quickly check their recent activity. The table where the user can log hours has a row with most of the columns prefilled with default values: current date, a starting time of 8:00, a thirty-minute break and an end time of 16:00. The break time can alternatively be the most common break time of the user, if such a value can be calculated from the user’s existing entries. The date field is limited to only allow dates between current day and 7 days back from it. This was implemented to encourage students to log their work on at the very least a weekly basis. Functionally this was implemented both on the HTML input field’s min and max values as well as in JS. When the user unfocuses the date field, a function is called to check the date, if it is not within the parameters, the date is set to current date. Focusing out of the time fields on the other hand updates the total sum of hours from the table, which is displayed above it.

Adding new rows on the table can be done either by clicking a button assigned to that function, or via a keyboard shortcut Alt + A. The new row will either increment the date field by one day or if it were to surpass current date, remain the same. Rows can be deleted in three ways: removing the last row either by a button at the end of the table or by Alt + Z shortcut, the third method removes a row via an X-button at the end of said row. These particular key combinations were chosen due to the standard Ctrl + A/Z already have common functions across most software. However, Alt key did not have any shortcuts with the aforementioned keys in browsers. “A” was used for the add command by first letter association. In similar vein to Ctrl + Z undoing your last action, Alt + Z undoes the last row.

All fields must be filled to be able to submit the hour logs. The submitted logs are validated on the PHP-side and the total hours are calculated there as well. The server will insert these into the database, logging the submission date as well. Figure 23 visualizes this procedure.

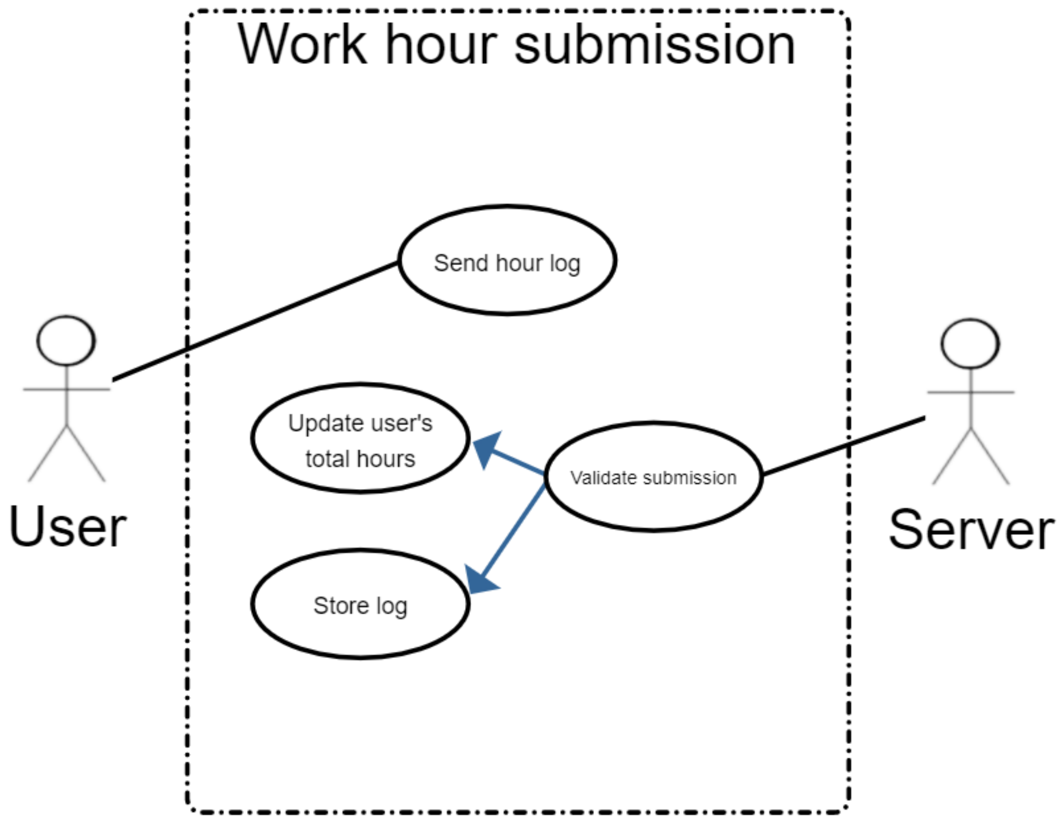


Figure 23 Log work hours - use case

5.2 Flexible code

When writing efficient code, one should consider if it can be used for one than one purpose. The project monitoring software has many instances of this. To examine a few cases, a simple starting point is the pop-up window which is used for user profiles and project details. The same event handler mediates which content to load on the pop-up window depending on context, these being clicking a username or a project name. The draggable windows all start from the same function, having a window that the user can drag by clicking and holding the top of the window like most application on computers, and an X-button in the corner to close the draggable window. Then the specific content is loaded into this draggable container.

In another case, users can define tags for their projects (such as an application, a game or C++), who or what is the client as well as roles that the user has had in projects (a programmer, a designer and etc.). These three categories are predefined by the admins and functionally it is done using the same method depicted in Figure 24. If a page utilizes a list of names multiple times, it would be unnecessary to load said list of names from the server multiple times for different purposes.

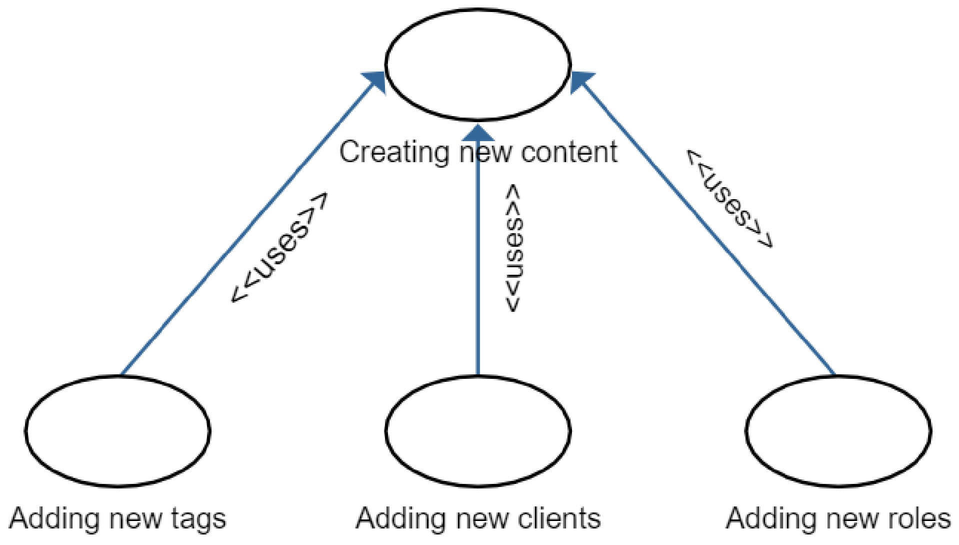


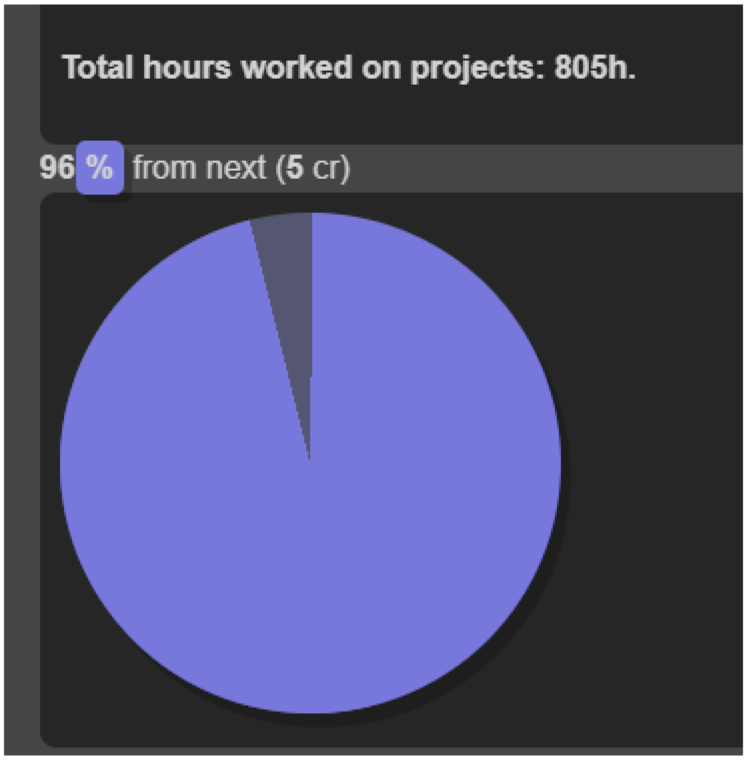
Figure 24 Tags, clients and roles - use case

A few instances allow the user to edit their content. Once again, the context of what is edited differs, but the same framework is used to handle these functions. A global edit mode is toggled when the user clicks the appropriate pencil icon, what they do afterwards depends on the page.

5.3 Data visualization

One of the major benefits of the project monitoring software is the utilities of the collected data, for both the students and faculty. It can be used to gauge how actively students work on projects and when as well as to view overall progress. The data helps teachers know if students are falling behind on their projects. However, all these utilities and more can be difficult to read in raw form. A table with raw names and numbers requires more focus and time to comb through, and that is where the aspect of data visualization comes in. As the main purpose of visualizing data is to make it simple, quick and intuitive to understand (Thomas 2015).

For instance, a user can view from their profile their total hours worked on projects and how far they are from the next five credit threshold, shown in Figure 25. As students are aware that five credits equate to 135 hours of work, they could calculate from 805 hours where the next threshold is, they could also figure it out from the percentage number given. However, it is more likely, before they've read either number, that they'll take a second to look at the pie chart and immediately know that they will reach the next threshold soon.


Total hours worked on projects: 805h.

96 % from next (5 cr)

Figure 25 User profile pie chart

The prior example is in aid of students. Figure 26 displays the average work amount per weekday over the course of a month for three groups. There are many ways to adjust the chart data, from total hours per group or by average, projects by specific tags such as Game Jam projects, grouping the chunks of data by day, week or month. And in time of writing, available chart formats include line and bar charts, but to compensate there is the option to download the chart data in CSV format to open on other applications such as Microsoft Excel. A user can use the chart feature to view their progress over a course of time, compare it to their group's average or to the most active user also referred to as the ”MVP” of the group or Most Valuable Player in gaming terms. A student can include themselves in the chart legend but others are displayed anonymously in groups.

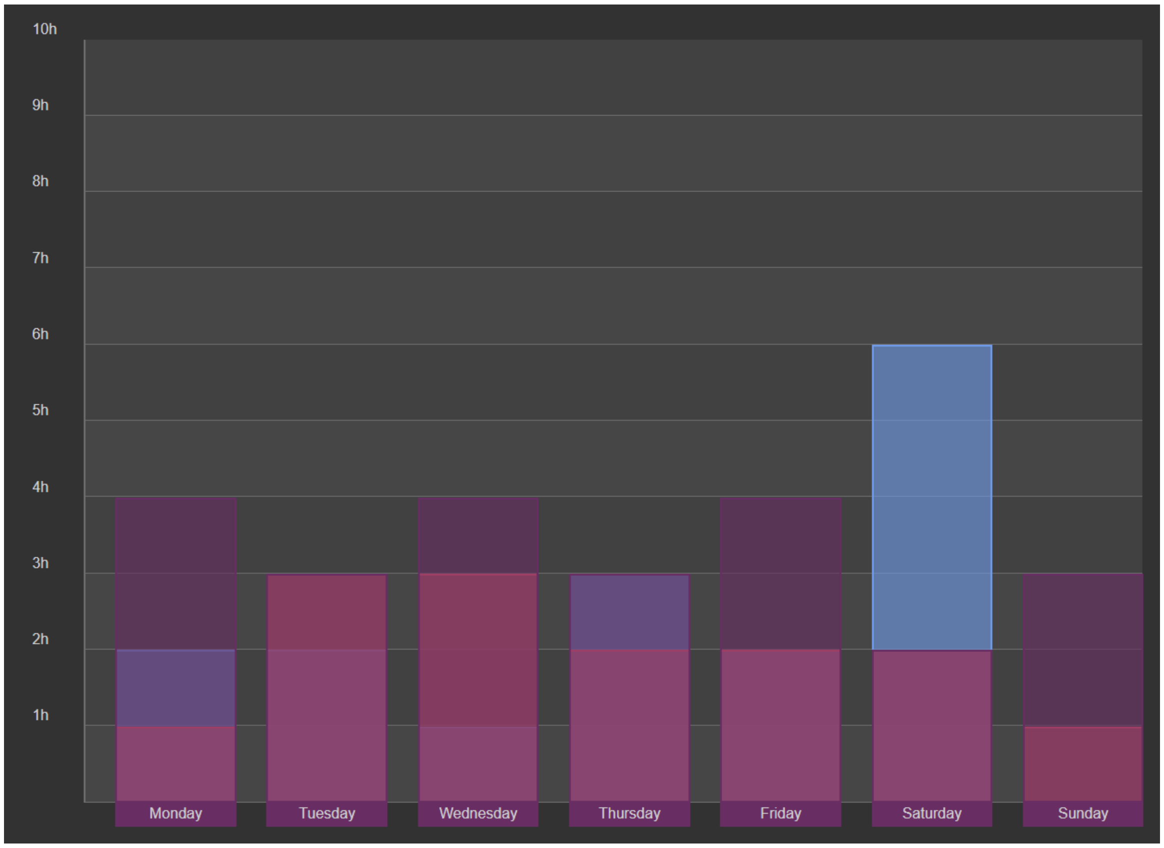
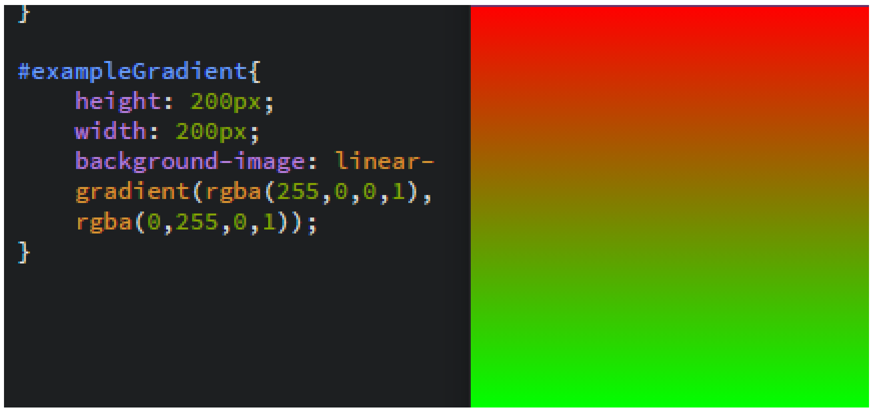


Figure 26 Weekday bar chart

These charts are made with plain CSS and JavaScript. The pie chart consists of three DIV-elements. The parent is an empty circle, done by giving the element 100% border radius. The fill portion of the pie is achieved with linear-gradient background-image -property. The *linear-gradient()* function creates a background of two or more colours (Figure 27 shows a gradient between red and green). It is also possible to define the direction of the gradient as well as the stop positions for the colours, which is the solution to the pie chart.



```
}

#exampleGradient{
    height: 200px;
    width: 200px;
    background-image: linear-
    gradient(rgba(255,0,0,1),
    rgba(0,255,0,1));
}
```

Figure 27 Example Linear-Gradient

By stacking multiple linear-gradients of which one has 50% grey (the empty area of the chart) and the other 50% transparent, and the second gradient is 50% fill colour and 50% grey, modifying the direction of the second gradient in degrees results in a slice of the pie capable of filling up to half of the chart. Adding a third gradient of the fill colour makes it possible to fill the rest of the pie. How much of the pie is filled needs to be calculated. We know the maximum hours and the angle of the circle (135 hours and 360°) as well as the current number of hours the user has by getting the remainder (% -operator in programming) of the current hour value out of 135. Using the example in Figure 25, the calculated result from 130 current hours (remainder of 805h) would be about 346°. In the CSS style, another 90° is added to it to set the starting point of the fill to the top (at 12 o’clock). The code checks if the value is over 180°, and if so it creates a second slice to fill the second half of the chart by taking the remainder of the [result divided by 180]. For the final angle of this second result, the program *subtracts* 90° to set the starting point to the opposite side of the first half, resulting in a two-colour pie chart created with plain CSS, JavaScript and math.

The line and bar charts utilize more complex functions, consisting of far more data and requiring to be significantly more flexible. Line chart essentially forms of a two-dimensional grid, the y-axis portraying a more fluctuating value while the x-axis has a more constant unit such as time. The four key values needed for the line chart were the x and y coordinates, the width of the line from those coordinates to the next and the rotation angle between the two points.

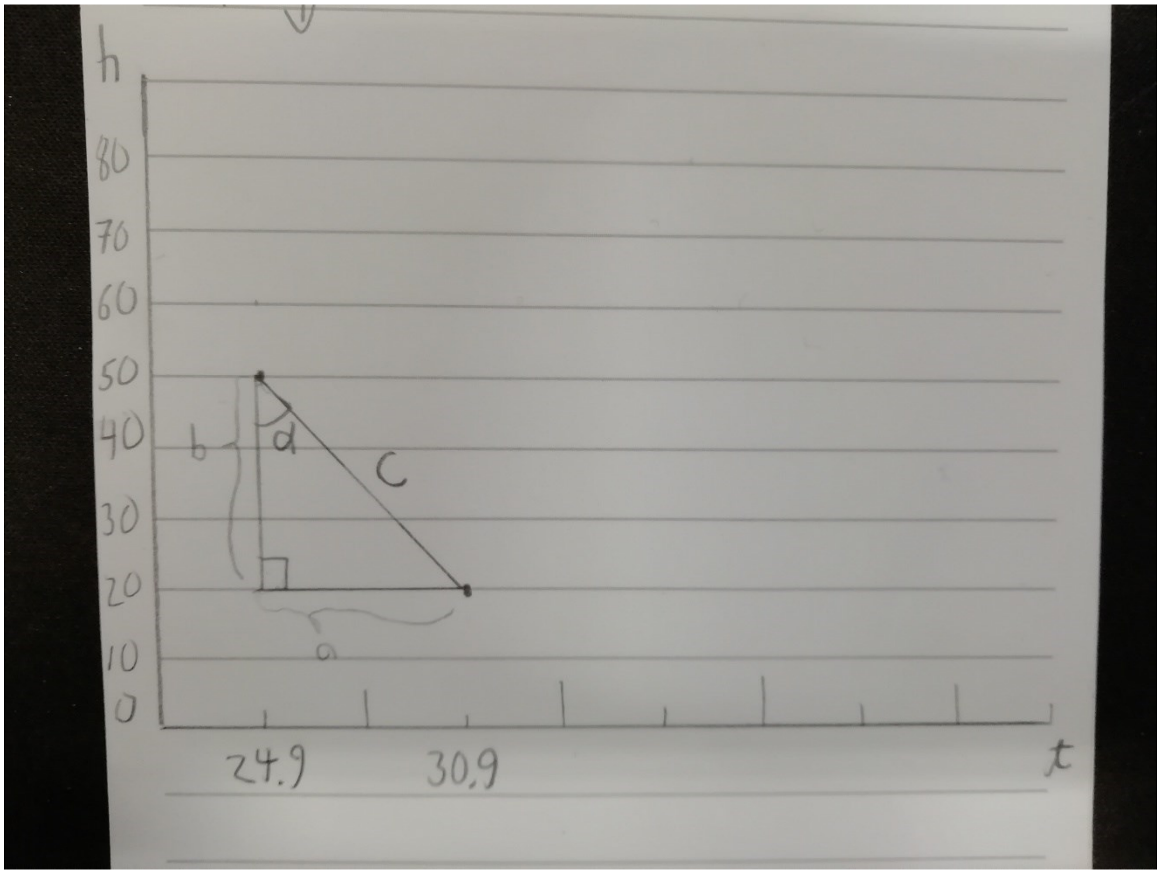


Figure 28 Line chart with trigonometry

The designing for the chart system began by sketching a coordinate axes and calculating the angle between two points using trigonometry (shown in Figure 28). Before any calculations could be done, the zero points needed be defined. Thought the chart displays hours on the y-axis and days on the x-axis, the actual measurements in the code are in pixels. When the program acquires the data for the chart, it first iterates the contents to figure out what the minimum and maximum dates as well as work hours are for the current chart, providing reference points for the draw function. The query provides the data in order by date, so in the code we can simply get the first and last entry as the minimum and maximum points on the x-axis. The coordinates for each point are calculated with taking the offset width/height (depending on axis) and in the case of the y-axis subtracting the result of the current hour amount times the offset height divided by the maximum hour amount, applying the system of equations theory. The sequence is better visualized in Figure 29.

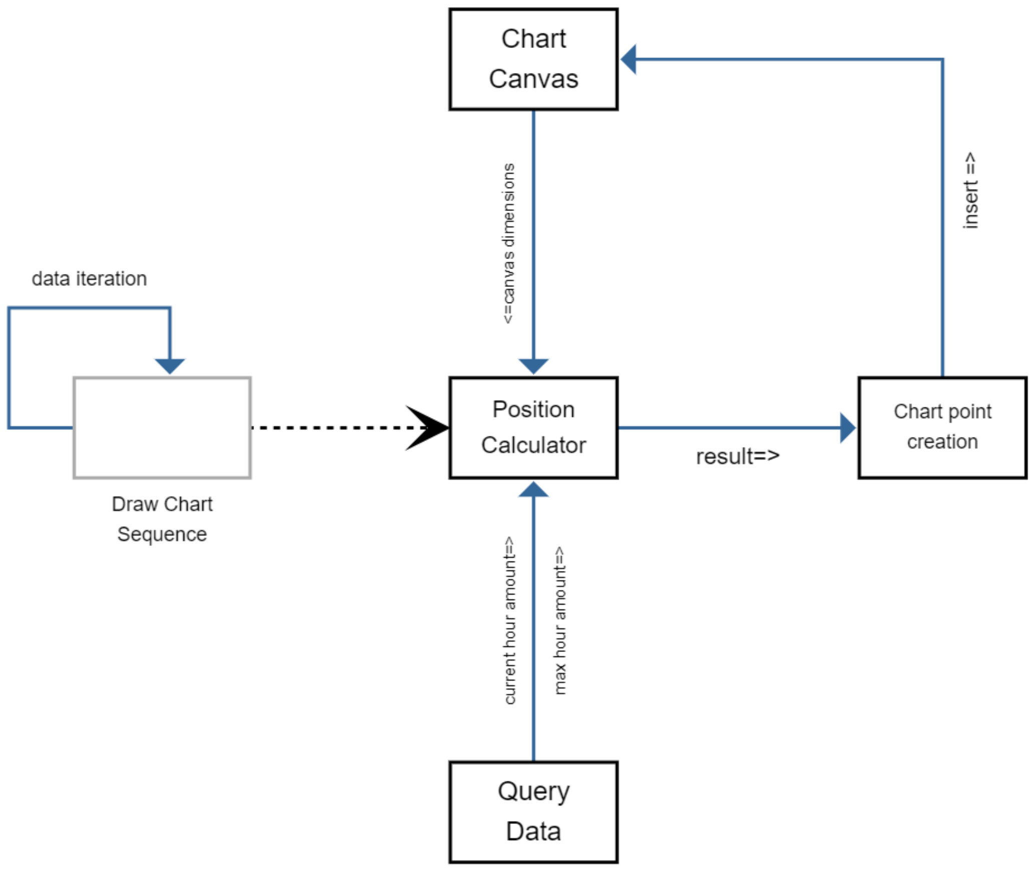


Figure 29 Point position calculation in chart

The x-axis calculates the position of a point with some additional variables. For all the calculations, the date values are converted into single numeric values using the *.getTime()* method of the Date() object. The y-axis always has 0 as the minimum value, however, dates cannot follow the same formula. Instead, to calculate the max point as all as the current date value in relation to 0 as minimum, the code subtracts the minimum date from them and divides the result by a single day. Having calculated the x and y points means the code can perform the operations for rotation and width. For the rotation angle, the inverse trigonometric function arctan (or atan in JavaScript) is used by taking the difference of the next y point and current y, divided by the difference of the next x point and current x. The *Math.atan()* method yields the result in radians, therefore to get the angle in degrees, it is multiplied by 180 and divided by Pi. The width of the line equates to the C side of the triangle in Figure 28, in other words the hypotenuse of a triangle, calculated with the following equation:

$$width = \sqrt{(x_{n+1} - x_n)^2 + (y_{n+1} - y_n)^2}$$

With the calculations done, the program can insert these four values (x, y, rotation and width) into the style attributes of the HTML created to visualize the point on the chart. The y is applied to the *top* property in pixels (px), x goes to the *left* property in px. A child element is created for the point which is given the width in px. The last value, rotation is assigned in degrees (deg) to the *transform* property using the *rotate()* CSS function. The point is visually represented by a small circle, therefore a separate element is created underneath it to act as the line. The end result is demonstrated in Figure 30.

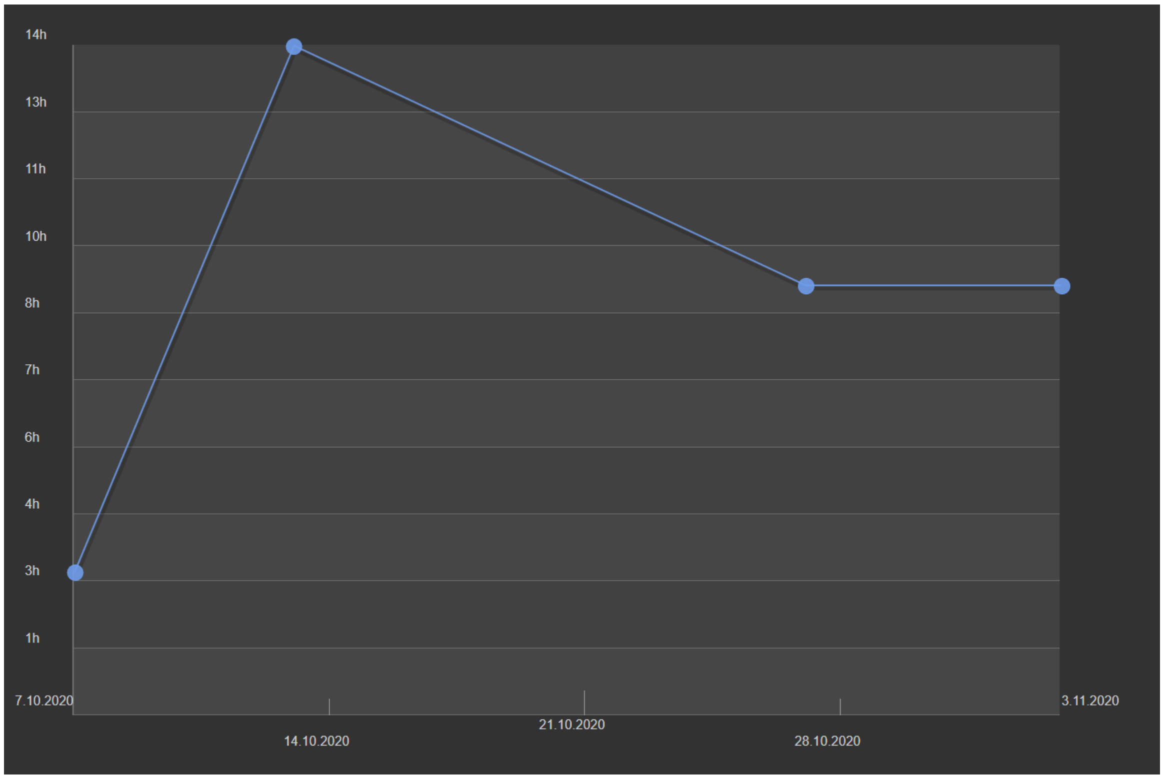


Figure 30 Line chart example

Each chart query is temporarily stored on the page in an object array, allowing the user to review them without having to request the data again from the server. The past queries are represented as buttons below the chart form section.

Toggling the edit mode on allows the user to rename these for clarity, as show in Figure 31. The yellow background on the edit icon indicating that edit mode is active, during which hovering over one of the stored charts buttons enables editing the text on them.

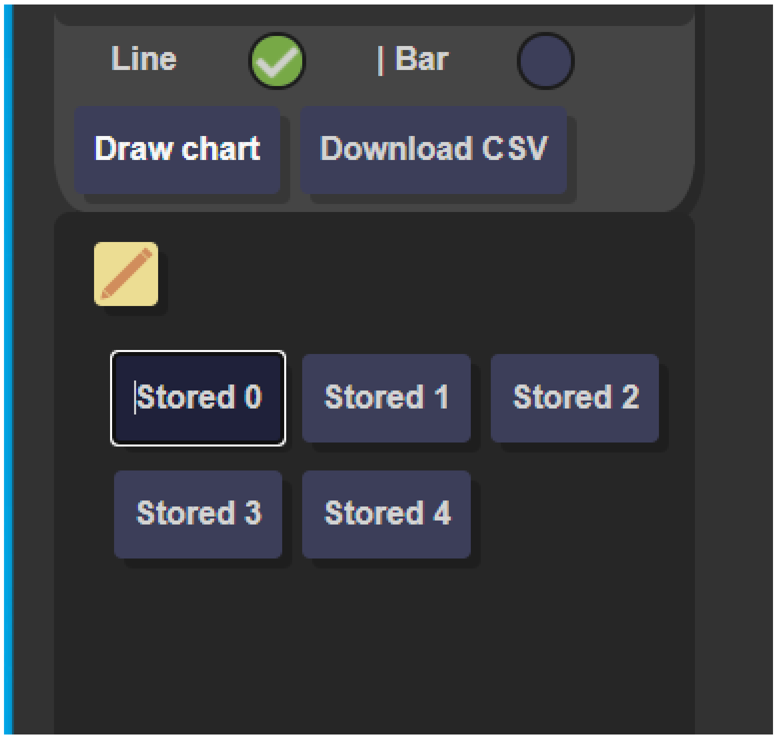


Figure 31 Stored charts

These stored chart queries are also used for the slideshow mode available on the page. Activating this mode hides all the UI from the screen, displaying only the chart itself and its legend. The program will then start drawing the charts found in the stored charts array, changing the chart in 10 second intervals. Pressing any key will end the slideshow and make the interface visible. The purpose behind this small feature was to provide an automated way to display data.

5.4 AJAX in UX

As it were, the function described above drew the entire chart instantly. To make the process of drawing a chart more enticing and “cinematic”, the draw function was given the *async* declaration. At the end of a loop when iterating the chart data, an await method is called, waiting for the current line to draw before continuing the loop. The line is a DIV element with an animation defined in its CSS class. The animation transitions from 0% width to 100% width, creating the sense of being drawn from left to right. What is described here is related to the user experience (UX) of the software. Hodent (2014) aptly defined UX as the feeling the targeted user gets when they interact with the software. As computers execute code in very rigid and near-instant manner, the use of animation and other visual cues is implemented to give the user feedback on their actions. Ideally these responses would be snappy, for example if a transition from one menu to another is slow the user may get frustrated and quit. In the project monitoring software, when HTML tables load they have a quick fade-up animation to appear as if they loaded up. When the user updates their entries or project details, the edit window was given a progress bar at the bottom which fills when the user clicks “Update”, the bar itself serves no actual functionality, it is merely there to tell the user that their changes are being made, using an async method it closes the pop-up window once it finishes. The charts page, however, was more of a tertiary feature, not essential to the core loop of the application, so it can be afforded a longer animation sequence. Creating a feeling of a competition of sorts as different colours simultaneously race to the finish line and perhaps improving the user experience.

5.5 Applied MySQL methods

The project monitoring software’s functions largely weigh around the information the database can provide. The data is handled on the client side, server side and database side in different ways.

5.5.1 Date & Time conversions

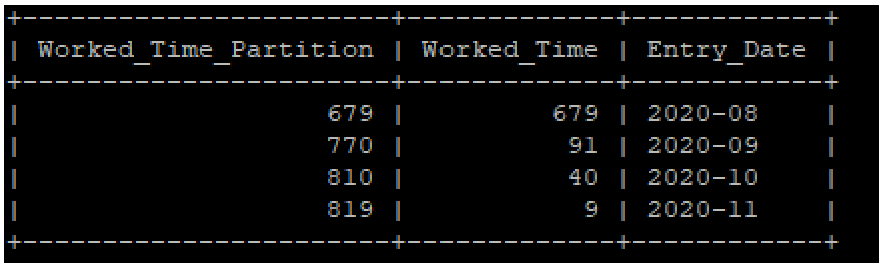
Being an application for logging work hours, a good bulk of the data is in date and time units. MySQL offers a large variety of methods to format datetime queries. The value in a time field in a data table is in the form of “HH:MM:SS”, but in queries it is possible to format the result in the desired way. For example, to get the sum of a collection of work hours, the *SUM()* function can be used. However, the raw sum of a collection of time columns doesn’t produce accurate results. To get the accurate sum of time columns, let us use a sample table TimeExample with a *TIME* type column Worked_Time. If there was a total amount of worked hours equating to 819 hours when including minutes. A query requesting *SUM(Worked_Time)* would yield a seven-figure number of unreliable amounts. A query *SUM(HOUR(Worked_Time))* requesting the sum of hours specifically would yield closes to the true value, in the example table printing “805”. For the result to include the minutes, one solution was to use a combination of *SEC_TO_TIME* and *TIME_TO_SEC* functions. Using the example case, in a query *HOUR(SEC_TO_TIME(SUM(TIME_TO_SEC(Worked_Time))))* the request first converted the Time values it found into seconds (*TIME_TO_SEC*), followed by a calculation of the sum of these values (*SUM*). This sum of seconds would then be converted back into a *TIME* type (*SEC_TO_TIME*) and the final output would be the hour part only (*HOUR*), which in this case would be 819.

Displaying these *TIME* entries in specific chunks, for example in hours per month, at the end of the *SELECT* statement a *GROUP BY* command can be included, grouping the rows of the result by the specified parameters. Say this example table had a column for the date of the entry, let us call it “Entry_Date”, the grouping by months would be “*GROUP BY DATE_FORMAT(Entry_Date, ‘%Y-%m’)*”. *DATE_FORMAT* function takes a DATE type value and changes the output of it to the format given in the parenthesis, in which “*%Y*” refers to year and “*%m*” refers to month. The result of the query would separate all entry sums as rows by the year and month.

5.5.2 Partitioning

In some cases, the data might be requested in increments over time, each entry adding up to one another. One solution to this is the use of partitioning. The *PARTITION BY* clause, similar to *GROUP BY*, partitions the rows into groups, the functional difference is that it is utilized within the *SELECT* statement via the use of another clause called *OVER*, which is used to define the method of partition. The *OVER* clause is needed here as otherwise the entire table would be considered as single partition. (GeeksforGeeks, 2019). To build on the example table, if the table would also contain a column called “User_Class” with rows such as “2B” or “9S”, the partitioning could be stated as *SUM([the select sum* *statement]) OVER (PARTITION BY User_Class ORDER BY Entry_Date) FROM TimeExample [conditions].*

The previously made *SELECT* statement needs to encased in another *SUM()* function to properly include the previously partitioned rows. The results of the example would be like in Figure 32.



```
+-----------------------+-------------+------------+
| Worked_Time_Partition | Worked_Time | Entry_Date |
+-----------------------+-------------+------------+
|                   679 |         679 | 2020-08    |
|                   770 |          91 | 2020-09    |
|                   810 |          40 | 2020-10    |
|                   819 |           9 | 2020-11    |
+-----------------------+-------------+------------+
```

Figure 32 Partition Query result

To query the progress of work from a specific timeframe while still accounting for hours submitted prior to that timeframe, the partitions were added to a sum of hours that predate the timeframe. This however required a null check, otherwise if there were nothing to add up to, the entire result would be null. Fortunately, MySQL has a function for this; the *IFNULL()*. This function takes two parameters, being the expression to check and the alternative value to return if the expression yields null. In the example this value would be 0. The completed query can fetch data and with it, for example, draw a graph to represent the progression of a student's work as seen in Figure 33.

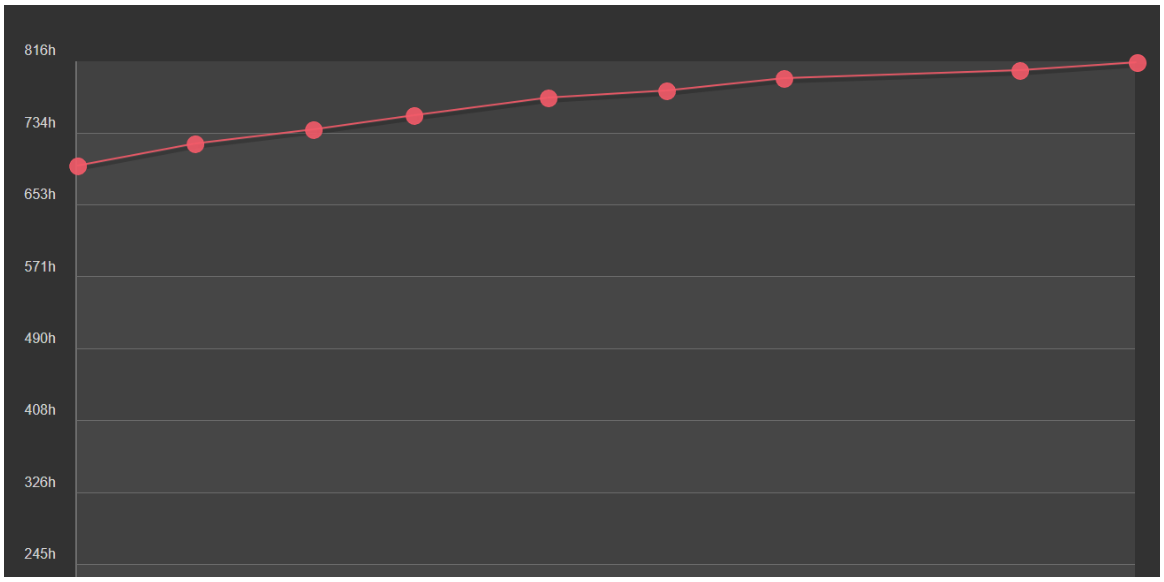


Figure 33 Progress chart

5.5.3 Case statements

CASE is a statement similar to an if-else in other programming languages, where specific conditions are addressed and if these are met, results are produced. Developing the table structure from the example above even further, suppose these worked hours all correlate to projects that they are linked to. The projects would be in their own table, *Projects_Example*. Multiple students could be linked to a same project, needing their own instance of the project, sharing some of the same details, while having other distinct ones. If a project member wanted to update some parts of the project unique to them while also updating some shared data such as the project's name, a query to update utilizing a *CASE* statement could be used in the following manner:

```
UPDATE Projects_Example SET [the unique column name] = CASE
WHEN [project member column name] = [this member's name or id]
THEN [new or changed information]
ELSE [unique column name]
END, [shared column name] = [updated or changed content]
WHERE [conditions to apply only to this group's project entries]
```

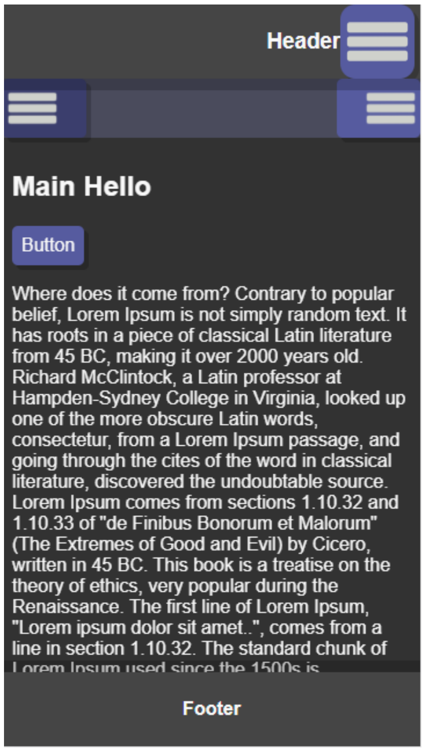
To break down this query, the entries to be updated in the Projects_Example table are defined in the *WHERE* part. Of these entries, the unique column is updated only to the user who is making the query, the *ELSE* portion makes sure that the unique column for the other project members is left as is. An alternative would be to update the shared column and unique column in separate queries, but with *CASE* it can be done in one go.

6 RESULTS

The project monitoring software was put into production in September once all game programming students had resumed their studies and began to work on their projects again. With this particular userbase, trained to find and report bugs, helpful feedback quickly accumulated and were handled to with fast response times, whether they were bugs or feature requests. The application and its data have made it possible for the teachers to keep track of students' activity regularly, especially during the remote working periods of 2020.

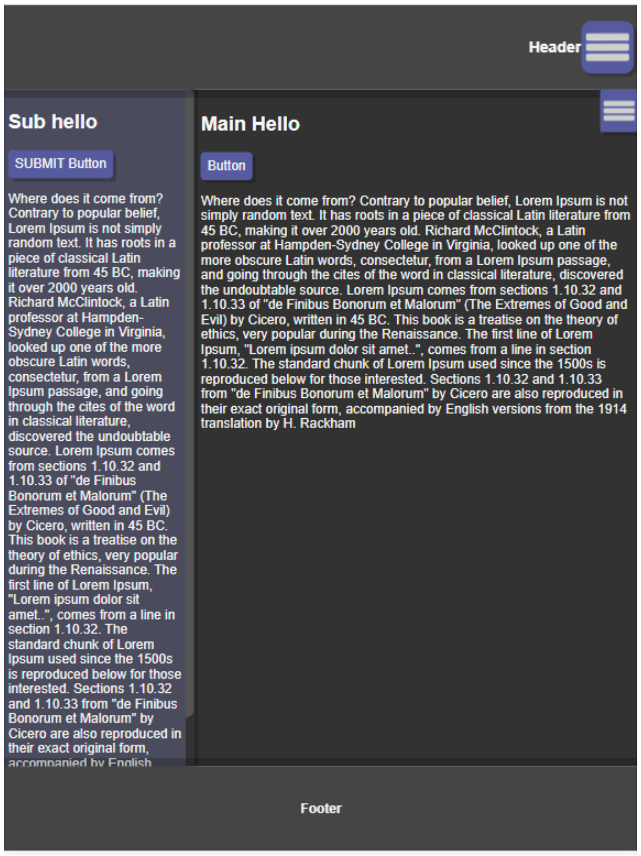
The results of the user experience survey suggest that most of the userbase feel the project monitoring software to be an improvement on the previous system, finding the new system easier and more coherent, with everything related to projects being in one place as well as offering live data and automation on some functions. More importantly the survey allowed for freeform feedback, as unfiltered criticism and requests lets the developer know what the software could improve upon to achieve a better user experience.

The software has received small updates throughout the Fall period, adjusting to the feedback. The designs for a larger update have already begun as well as training the future developers for this instance of the application, with plans to make it even simpler and more efficient. Figure 34, Figure 35 and Figure 36 show how the new interface would transition between mobile, tablet and desktop layouts.



Where does it come from? Contrary to popular belief, Lorem Ipsum is not simply random text. It has roots in a piece of classical Latin literature from 45 BC, making it over 2000 years old. Richard McClintock, a Latin professor at Hampden-Sydney College in Virginia, looked up one of the more obscure Latin words, consectetur, from a Lorem Ipsum passage, and going through the cites of the word in classical literature, discovered the undoubtable source. Lorem Ipsum comes from sections 1.10.32 and 1.10.33 of "de Finibus Bonorum et Malorum" (The Extremes of Good and Evil) by Cicero, written in 45 BC. This book is a treatise on the theory of ethics, very popular during the Renaissance. The first line of Lorem Ipsum, "Lorem ipsum dolor sit amet..", comes from a line in section 1.10.32. The standard chunk of Lorem Ipsum used since the 1500s is

Figure 34 New Mobile layout



Sub hello

SUBMIT Button

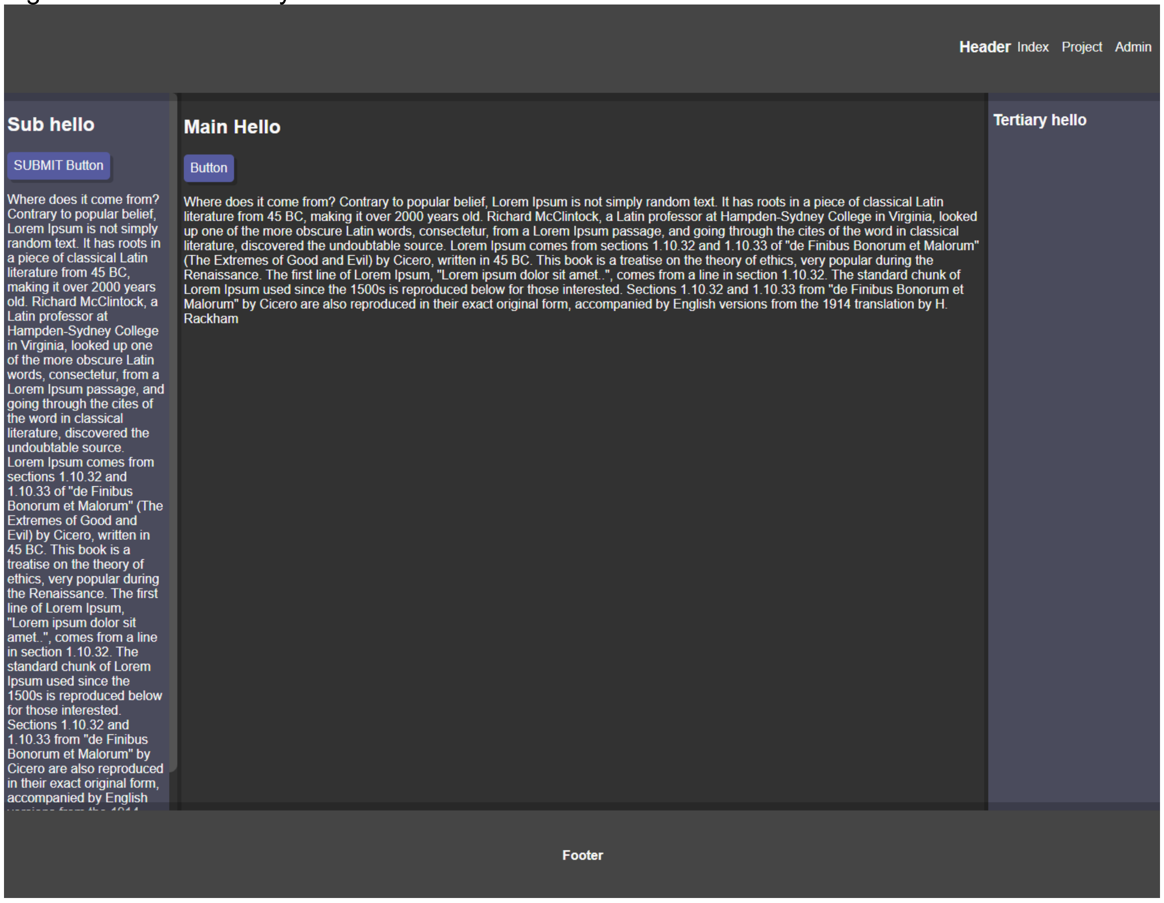
Where does it come from? Contrary to popular belief, Lorem Ipsum is not simply random text. It has roots in a piece of classical Latin literature from 45 BC, making it over 2000 years old. Richard McClintock, a Latin professor at Hampden-Sydney College in Virginia, looked up one of the more obscure Latin words, consectetur, from a Lorem Ipsum passage, and going through the cites of the word in classical literature, discovered the undoubtable source. Lorem Ipsum comes from sections 1.10.32 and 1.10.33 of "de Finibus Bonorum et Malorum" (The Extremes of Good and Evil) by Cicero, written in 45 BC. This book is a treatise on the theory of ethics, very popular during the Renaissance. The first line of Lorem Ipsum, "Lorem ipsum dolor sit amet..", comes from a line in section 1.10.32. The standard chunk of Lorem Ipsum used since the 1500s is reproduced below for those interested. Sections 1.10.32 and 1.10.33 from "de Finibus Bonorum et Malorum" by Cicero are also reproduced in their exact original form, accompanied by English

Main Hello

Button

Where does it come from? Contrary to popular belief, Lorem Ipsum is not simply random text. It has roots in a piece of classical Latin literature from 45 BC, making it over 2000 years old. Richard McClintock, a Latin professor at Hampden-Sydney College in Virginia, looked up one of the more obscure Latin words, consectetur, from a Lorem Ipsum passage, and going through the cites of the word in classical literature, discovered the undoubtable source. Lorem Ipsum comes from sections 1.10.32 and 1.10.33 of "de Finibus Bonorum et Malorum" (The Extremes of Good and Evil) by Cicero, written in 45 BC. This book is a treatise on the theory of ethics, very popular during the Renaissance. The first line of Lorem Ipsum, "Lorem ipsum dolor sit amet..", comes from a line in section 1.10.32. The standard chunk of Lorem Ipsum used since the 1500s is reproduced below for those interested. Sections 1.10.32 and 1.10.33 from "de Finibus Bonorum et Malorum" by Cicero are also reproduced in their exact original form, accompanied by English versions from the 1914 translation by H. Rackham

Figure 35 New tablet layout



Sub hello

SUBMIT Button

Where does it come from? Contrary to popular belief, Lorem Ipsum is not simply random text. It has roots in a piece of classical Latin literature from 45 BC, making it over 2000 years old. Richard McClintock, a Latin professor at Hampden-Sydney College in Virginia, looked up one of the more obscure Latin words, consectetur, from a Lorem Ipsum passage, and going through the cites of the word in classical literature, discovered the undoubtable source. Lorem Ipsum comes from sections 1.10.32 and 1.10.33 of "de Finibus Bonorum et Malorum" (The Extremes of Good and Evil) by Cicero, written in 45 BC. This book is a treatise on the theory of ethics, very popular during the Renaissance. The first line of Lorem Ipsum, "Lorem ipsum dolor sit amet..", comes from a line in section 1.10.32. The standard chunk of Lorem Ipsum used since the 1500s is reproduced below for those interested. Sections 1.10.32 and 1.10.33 from "de Finibus Bonorum et Malorum" by Cicero are also reproduced in their exact original form, accompanied by English

Main Hello

Button

Where does it come from? Contrary to popular belief, Lorem Ipsum is not simply random text. It has roots in a piece of classical Latin literature from 45 BC, making it over 2000 years old. Richard McClintock, a Latin professor at Hampden-Sydney College in Virginia, looked up one of the more obscure Latin words, consectetur, from a Lorem Ipsum passage, and going through the cites of the word in classical literature, discovered the undoubtable source. Lorem Ipsum comes from sections 1.10.32 and 1.10.33 of "de Finibus Bonorum et Malorum" (The Extremes of Good and Evil) by Cicero, written in 45 BC. This book is a treatise on the theory of ethics, very popular during the Renaissance. The first line of Lorem Ipsum, "Lorem ipsum dolor sit amet..", comes from a line in section 1.10.32. The standard chunk of Lorem Ipsum used since the 1500s is reproduced below for those interested. Sections 1.10.32 and 1.10.33 from "de Finibus Bonorum et Malorum" by Cicero are also reproduced in their exact original form, accompanied by English versions from the 1914 translation by H. Rackham

Tertiary hello

Figure 36 New Desktop layout

7 CONCLUSIONS

This project offered a great opportunity to develop a software at a full-stack level from start to finish, delving into each aspect of software development. Initially most of the frontend code was written with basic JavaScript methods, but gradually, as jQuery became more and more familiar, the code evolved into more efficient and cleaner solutions. Some of the earlier implementations were updated or changed later on as new techniques and features came up. When the task of creating the charts feature was on the table, initially a pre-existing JS library, Chart.js, was experimented with. This library was quite expansive and neat visually, however in the end I found it more convenient and beneficial on an educational level for myself to write my own chart drawing code.

From the development perspective, being largely a one-man undertaking, this was developed in a very agile way. Creating incremental minimum viable product pieces, continuously evolving the application as feedback and new ideas came in, not holding onto a strict design document. This meant that the documentation was rarely up-to-date. An alternative that was used was a weekly update channel on a Microsoft Teams group for the project, where new features and major changes were explained on a weekly basis. Additionally, having thorough commit messages for the GitHub project daily created a well-documented timeline for the development.

The main goal of this project was to create something that would improve the project-oriented studies for both the students and the teachers alike, accessible from anywhere. Based on the feedback received from the users, I would say this goal was achieved, however it doesn’t mean the work is finished. In due time, whether it be by my hand or any future developers, this software might go through a complete overhaul, as it can and will continue to be improved until it is the best that it can be. As the execution methods for these project studies evolve, so should this software.

REFERENCES

Aadamsoo, A.M. 2010. Web Based Project Management System. Thesis. Available at: https://www.theseus.fi/handle/10024/16996 [Accessed 7 October 2020].

Brackets. 2020. Brackets – A modern, open source code editor that understands web design. WWW document. Available at: http://brackets.io/ [Accessed 16 September 2020].

Coolors. 2020. Fabrizio Bianchi. Available at: https://coolors.co/ [Accessed 28 September 2020].

Carlos, G., Carols, G. 2013. Responsive Web Design with jQuery. Ebook. Packt Publishing, Limited.

Education Statistics Finland. 2019. Student feedback for UAS at the graduation phase (AVOP), UAS qualification. Available at: https://vipunen.fi/en-gb/_layouts/15/xlviewer.aspx?id=/en-gb/Reports/Ammattikorkeakoulutus%20-%20opiskelijapalaute%20-%20AMK%20-%20ohjauksen%20ala_EN.xlsb [Accessed 14 September 2020].

MySQL PARTITION BY Clause. 2019. GeeksforGeeks. WWW-document. Available at: https://www.geeksforgeeks.org/mysql-partition-by-clause/ [Accessed 10 November 2020].

Harvard Health Letter. 2020. Blue light has a dark side. WWW-document. Available at: https://www.health.harvard.edu/staying-healthy/blue-light-has-a-dark-side [Accessed 16 September 2020].

Hodent, C. 2014. Developing UX Practices at Epic Games. GDC. PDF-document. Available at: https://www.gdcvault.com/play/1020934/Developing-UX-Practices-at-Epic [Accessed 6 November 2020].

Joshi, Vijay. 2010. PHP jQuery Cookbook. Ebook. Packt Publishing, Limited.

MDN web docs. Available at: https://developer.mozilla.org/en-US/ [Accessed 14 September 2020].

OpenJS Foundation. Available at: https://openjsf.org/ [Accessed 14 September 2020].

Thomas, S. 2015. Datavisualization with JavaScript. Ebook. No Starch Press, Incorporated.

Vidhya, V., Jeyaram, G., Ishwarya, K.R. 2016. Database Management Systems. Ebook. Alpha Science International.

Wells, K. & Kloppenborg, T. 2015. Project Management Essentials. Ebook. Business Expert Press.

W3schools.com. Available at: https://www.w3schools.com/whatis/whatis_fullstack.asp [Accessed 14 September 2020].

Xamk https://www.xamk.fi/koulutukset/insinoori-amk-peliohjelmointi/ [Accessed 14 September 2020].

LIST OF FIGURES

Figure 1 HTTP Request
Figure 2 PHP output to browser
Figure 3 JS sequence of date object
Figure 4 Prototype date
Figure 5 jQuery example comparison
Figure 6 async wait example
Figure 7 async syntax error
Figure 8 PHP variables
Figure 9 Sample database table
Figure 10 SELECT query
Figure 11 Sample table query with where clause
Figure 12 Mobile layout of Log page
Figure 13 Desktop log page
Figure 14 Admin page layout
Figure 15 Projects page layout (general)
Figure 16 Projects page layout (wide screen)
Figure 17 YouTube dark theme
Figure 18 Project Monitoring Software design - background colour scheme
Figure 19 Last log date
Figure 20 Coolors palette
Figure 21 Example of post stuff
Figure 22 Sign redirection
Figure 23 Log work hours - use case
Figure 24 Tags, clients and roles - use case
Figure 25 User profile pie chart
Figure 26 Weekday bar chart
Figure 27 Example Linear-Gradient
Figure 28 Line chart with trigonometry
Figure 29 Point position calculation in chart
Figure 30 Line chart example
Figure 31 Stored charts
Figure 32 Partition Query result

Figure 33 Progress chart

Figure 34 New Mobile layout

Figure 35 New tablet layout

Figure 36 New Desktop layout