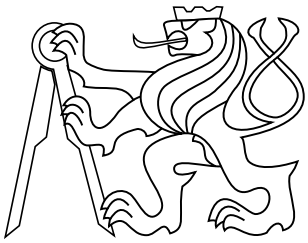
Bachelor's thesis

Web Front-End for Student Data Analysis Application

Jonas Vaclavek



May 2015

Czech Technical University in Prague
Faculty of Electrical Engineering, Department of Cybernetics

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: Jonáš V á c l a v e k

Studijní program: Otevřená informatika (bakalářský)

Obor: Informatika a počítačové vědy

Název tématu: Webový front-end aplikace pro analýzu studentských dat

Pokyny pro vypracování:

1. Seznamte s aktuální verzí aplikace pro analýzu studentských dat (informace o projektu a veškeré materiály jsou k dispozici na [1]). Analyzujte slabiny systému z hlediska webové prezentace a analyzujte požadavky na rozšíření stávající funkčnosti front-endu aplikace.
2. Na základě provedené analýzy navrhněte úpravy a rozšíření aplikace.
3. Vámi navržené řešení implementujte.
4. Ověřte funkčnost navrženého front-endu.
5. Zhodnoťte dosažené výsledky a další možné pokračování tohoto projektu.

Seznam odborné literatury:

[1] OU Analyse project, analyse.kmi.open.ac.uk
[2] Fielding Jonathan - Beginning Responsive Web Design with HTML5 andC SS3 - Apress 2014
[3] Few Stephen - Information Dashboard Design: The Effective Visual Communication of Data - O'Reilly Media 2006
[4] Souders Steve - High Performance Web Sites: Essential Knowledge for Front-End Engineers - O'Reilly Media 2007

Vedoucí bakalářské práce: Ing. Jakub Kužílek, Ph.D.

Platnost zadání: do konce letního semestru 2015/2016

L.S.

doc. Dr. Ing. Jan Kybic
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 2. 12. 2014

Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Cybernetics

BACHELOR PROJECT ASSIGNMENT

Student: Jonáš V á c l a v e k

Study programme: Open Informatics

Specialisation: Computer and Information Science

Title of Bachelor Project: Web Front-End for Student Data Analysis Application

Guidelines:

1. Familiarize with current version of student data analysis application (all required information available at [1]). Analyse system weaknesses with respect to web presentation and analyse requirements for application front-end development.
2. Suggest changes and improvements of application according to analysis.
3. Implement these suggestions.
4. Test functionality of newly created front-end.
5. Evaluate achieved results and outline future development of project.

Bibliography/Sources:

[1] OU Analyse project, analyse.kmi.open.ac.uk
[2] Fielding Jonathan - Beginning Responsive Web Design with HTML5 andC SS3 - Apress 2014
[3] Few Stephen - Information Dashboard Design: The Effective Visual Communication of Data - O'Reilly Media 2006
[4] Souders Steve - High Performance Web Sites: Essential Knowledge for Front-End Engineers - O'Reilly Media 2007

Bachelor Project Supervisor: Ing. Jakub Kužílek, Ph.D.

Valid until: the end of the summer semester of academic year 2015/2016

L.S.

doc. Dr. Ing. Jan Kybic
Head of Department

prof. Ing. Pavel Ripka, CSc.
Dean

Prague, December 2, 2014

Poděkování

Děkuji PhD. Jakubu Kužílkovi za odborné konzultace, připomínky a cenné rady, které mi předal při vypracovávání bakalářské práce. Děkuji své rodině, která mi poskytla potřebnou podporu po celou dobu mého studia.

Prohlášení autora práce

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V praze dne 21.5.2015 .....

Podpis autora práce

Abstract

OU Analyse je webová aplikace vyvíjena na Open University ve Velké Británii. Zabývá se predikováním úspěchu studentů v kurzech. V této bakalářské práci je analyzován její dashboard z hlediska výkonu, podpory webových prohlížečů a mobilních zařízení. Aplikace je nejdříve zkoumána vzhledem k požadavkům projektu. Poté je porovnána s běžně používanými postupy při vývoji webových aplikací. Na základě provedené analýzy jsou navržena řešení problémů a vylepšení nedostatků, která vedou ke zvyšené rychlosti aplikace, vyšší multiplatformní kompatibilitě a zvýšené přehlednosi webových stránek při prohlížení z odlišných zařízení. V závislosti na navržených změnách, je vytvořena nová verze aplikace, která je porovnána s původní verzí.

Klíčová slova

OU Analyse; Webová aplikace; Front-end; Responsivní design; Multiplatformní kompatibilita

Abstract

OU Analyse is a web application developed under The Open University in Great Britain, which aims at prediction of success of students in a course. In this thesis, dashboard of the application is analysed from the performance, cross platform compatibility and mobile device support point of view. The application is analysed with respect to the requirements of the OUA project and then with respect to the general practices used while developing web applications. Based on the analysis, solutions of the problems and improvements of the weaknesses are suggested, which leads to increased speed on the web site, higher cross-platform compatibility and layout improvements, when accessing the application from different devices. These suggestions are implemented and comparison between the old and the new version is made.

Keywords

OU Analyse; Web application; Front-end; Responsive design; Cross-platform compatibility

Contents

1 Introduction

In last years, web applications (WA) have become one of the very common way people interact with computers thanks to the increasing internet speed. WA are capable of more complex operations and provide various functionalities. In many instances they tend to replace classic desktop applications for many reasons, e.g. cross-platform compatibility support or ability to be easily updated and maintained. However, while providing many advantages, there are certain weaknesses of which one has to be aware when implementing such application. The main disadvantage stems from the fact that developer does not know what the hardware, software and network the application will run on. For example, a general drawback of WA is their slower response to user actions when compared to a reaction speed of desktop applications. It is often caused by inappropriate front-end development [1].

In my thesis, I focus on front-end development with the main goal to increase WA performance, make WA more user friendly and create WA compatible with a multiple browsers. Also, I add new functionalities to enhance user experience while using WA. I have taken a real world example of a WA developed at the Open University (OU) and focused on improving this application's front-end. Both OU and the WA are presented in the rest of this chapter. Chapter 2 provides a technical introduction into the problem (what is a web application, what technologies are used, etc). Chapter 3 shows the OU Analyse application in details and problems in the current iteration are pointed out with a main focus on front-end development. Chapter 4 is dedicated to solving of the previously found problems and implementing new features. Finally, testing of the newly created version is taken and results are compared to the original version in last part.

1.1 Open University

The Open University is one of the largest distance learning universities in the United Kingdom with more than two hundred thousand students. It offers hundreds of distance learning courses, which can be studied both as part of the university degree or as standalone module. Students use Virtual Learning Environment (VLE) accessible via Internet to access study materials, for submitting their assignments, accessing forum, etc. Students of one module are typically divided into several study groups of no more than twenty persons. Every study group is assigned an associate lecturer (also called tutor), who guides students throughout the module.

The OU has implemented several interventions in order to increase student retention. Since number of students in each module can be more than few thousand, the interventions have to be carefully planned. Right intervention is important mainly for two reasons. Firstly, students who are at risk of failing a module can be offered appropriate help. Because one module at the Open University costs up to 1200 pounds, such help can also lead to significant financial savings for these students. Secondly, more precise intervention management can help to increase student retention therefore to increase income of the university and also decrease money spent on interventions made. [2]

1.2 OU Analyse

OU Analyse (OUA) project is being developed at the OU to help module teams and tutors to manage the interventions. It aims at predicting students at risk of failing. This prediction is based on demographic data about students, legacy data about the module and data collected from the VLE. All collected data are evaluated by four predictive models using machine learning methods to create list of students, who are likely to fail the module. The module team receives weekly updated lists of students predicted as at-risk. VLE is also used for modelling of all possible study trails which students can follow throughout the module and which lead to success. Based on that student can be recommended an activity in order to improve his performance in the course. Simple diagram of the process can be seen in Fig. 1.

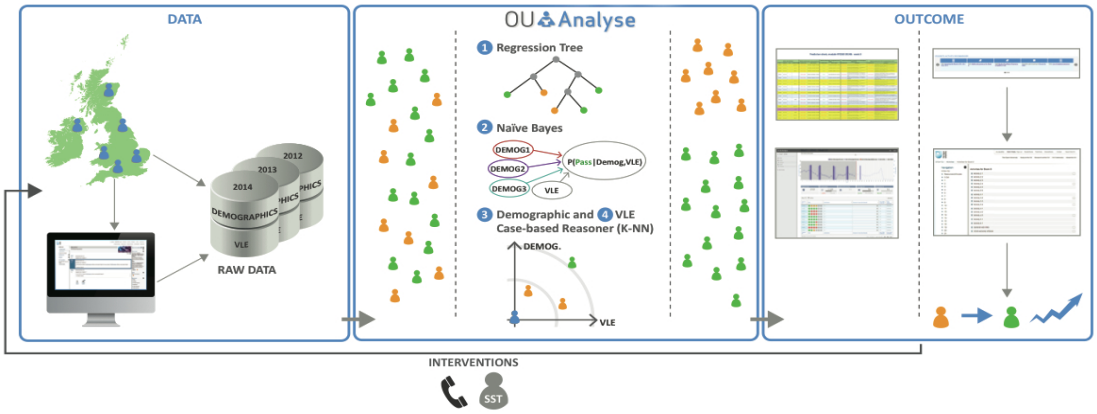


Figure 1 OU Analyse data analysis process [3]

Dashboard for OUA has been implemented to simplify access to predictions of students' results and other student data. It offers course overview showing what is the average VLE activity, details about previous presentations of same course, etc. It also shows a list of students attending the course and their predictions for the next assignment along with their final result.

The dashboard is mainly used by module teams and tutors to access data about students. It is important that tutors do not struggle to use the dashboard and so dashboard has to be as user friendly and well-organized as possible. Because every dashboard is different from case to case, it can not be simply copied from an already existing system. Therefore, it takes some time to find a proper way to structure, style and implement interactions with a dashboard. The design process is usually iterative and comes with problems of many types [4].

2 Theoretical part

This section briefly summarizes the concepts on which OUA and my work are based on. It shortly explains what is web application, which parts it consists of and also underlines the most important technologies used for development of WA.

2.1 Web application

WA can be described using simple client-server model (Fig. 2). WA is an application using a web browser as a client. Client communicates over the Internet with a server using Hypertext Transfer Protocol (HTTP) in order to deliver content of WA to a user, who interacts with a browser (client). More details about client, server and HTTP can be found in sections 2.2, 2.3 and 2.4 respectively.

Using browser as a client has many benefits. One of the most important is fact that updates of WA need to be made only on server side. When users access an application after an update, new version of WA is automatically used. This removes the need of installing new version of the application on potentially thousands of client computers. It also makes cross-platform compatibility support easy to accomplish. Finally, users do not need to own high performance computers, since only browser is needed for using WA. Other benefits usually depend on the type of WA.

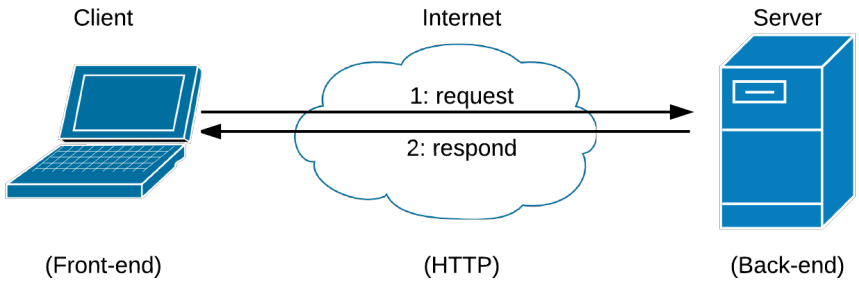


Figure 2 Simple web application architecture using client-server model.

2.2 Front-end

Development of client side of WA is called front-end development and front-end (FE) itself is then understood as every content user can see and can interact with in a browser [5]. It is usually a mixture of Hypertext Markup Language (HTML), Cascading Style Sheet (CSS) and JavaScript (JS). All these languages are interpreted and controlled by a web browser resulting in a web page - interface user can interact with. Set of all available web pages is called web site [6].

2.2.1 Hypertext Markup Language

HTML is a mark-up language used to create web pages [7]. It belongs to a family of declarative programming languages [8], which means that HTML specifies content of a web page and its structure, but does not specify how this content is styled or behaves. Parts of HTML document are labelled by using HTML tags, each tag specifies type of content. It consists of a tag name wrapped together with tag attributes in angle brackets, i.e. "<name attributes>". Then, HTML element is a combination of opening and closing tags enclosing the content. Structure of a web page is created by nesting HTML elements inside of one another. Fig. 3 shows simple HTML document with mandatory tags marked.

```
1  <html>                          //mandatory
2   <head>                         //mandatory
3    <title>Example page</title>   //mandatory
4   </head>
5   <body>                         //mandatory
6    <ul class="list">
7     <li>Item 1</li>
8     <li>Item 2</li>
9    </ul>
10   <button id="example-btn" type="button">
11     I am button
12   </button>
13   <a href="http://analyse.kmi.open.ac.uk">
14     OU Analyse web site
15   </a>
16  </body>
17 </html>
```

Figure 3 Example HTML code

Before browser renders a page, HTML file is parsed and Document Object Model tree (DOM) is created for inner representation of the HTML document inside a browser [9]. DOM is an interface of HTML elements to other languages like JS. This interface enables access to HTML elements but also enables dynamic changes in HTML code. Fig. 4 shows DOM tree created from HTML code from Fig. 3.

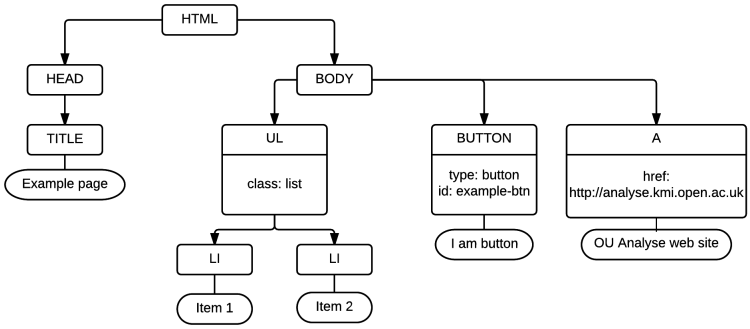


Figure 4 Example of DOM tree

Once the DOM is created a page is rendered in a browser Fig. 5.

Web sites and their pages are connected using Hypertext method, which allows user to navigate from page to another page. Navigation to other page is done by clicking on a special element which has a link to the page specified as a part of HTML tag, an example of such element is *<a>* tag from previous figure.

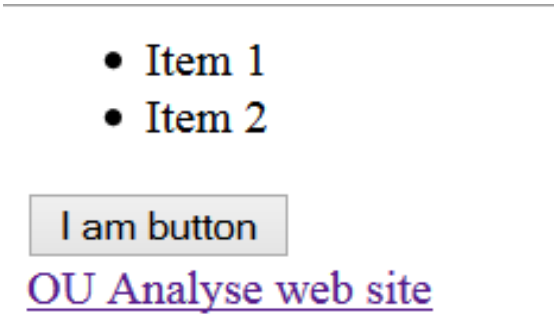


Figure 5 Rendering of example HTML code Fig. 3

2.2.2 Cascading Style Sheet

It is a good practice to separate page structure from its design. To do so, CSS is used as a styling language. It defines layout, color, size and other attributes of page elements. CSS document is a text file containing a set of CSS rules, where rule consist of selector and declaration block. Each rule affects page content specified in selector pointing either to:

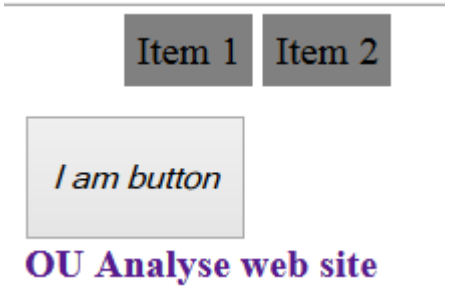
- All elements of one type in a DOM, e.g. all *<li>* tags.
- Elements of a same class or id (both are tag attributes). In CSS, these rules start with dot (.) and hash (#) for classes and ids, respectively, e.g. ".list" and "#example-btn".
- Elements in a special state, e.g. mouse over the element [10].
- Elements based on relative position in a DOM tree. These are specified as a combination of previous types of selectors, e.g. ".list li" matches only list items *<li>* of ".list" class elements.

Declaration block is a semicolon separated list of individual style declarations in a form: "property name" : "value" , e.g. "color : red".

CSS can be included to HTML file by using one of the following methods:

- **External style sheet**, CSS is written into external file and included using *<link>* tag in *<head>* of HTML file.
- **Internal style sheet**, CSS rules are written as a part of *<style>* tag directly in HTML code.
- **Inline style**, CSS rules are applied to single element using "style" attribute of the relevant tag.

Fig. 6a demonstrates result of CSS styling Fig. 6b applied on code Fig. 3.



a) Result of CSS applied on HTML snippet (Fig. 5)

```
1  .list li{
2    list-style-type: none;
3      padding: 5px;
4      background: grey;
5      display: inline;
6  }
7  #example-btn{
8      height: 50px;
9      font-style: italic;
10 }
11 a{
12       text-decoration: none;
13       font-weight: bold;
14 }
```

b) Example CSS code

Figure 6 Demonstration of using CSS to style HTML document

2.2.3 JavaScript

JS is interpreted programming language generally used at client side to control user input, control browser actions, load additional content to a browser and make changes to the DOM. In short, it gives functionality to a page. JS code can be added to HTML document in two ways:

- **Inline JavaScript**, JS code is written as content of HTML document between opening and closing *<script>* tag.
- **External JavaScript**, JS code is written in external file and reference to the file is defined in *<script>* tag attribute. Any content enclosed by the *<script>* tag is not executed.

When HTML *<script>* element is reached, during parsing of HTML document, the JS code defined by the tag is immediately executed in a browser using built-in JS interpreter.

2.2.4 Implementation in browsers

Even though HTML, CSS and JS are standardized [11], [12]. It can be shown (e.g. charts at portal "Can I use" [13]) that their implementation across available browsers is not the same and so it is challenging to create WA which behaves identically (at least very similarly) in all possible browsers. Tab. 1 lists the most popular web browsers and their market share.

Name	Chrome	Internet Exlorer	Firefox	Safari	Opera	Others
Usage	46.5%	20.4%	17.5%	10.5%	1.4%	3.7%

Table 1 The most popular web browsers on the market [14].

2.3 Back-end

Back-end (BE) is server side of WA. Even if understanding of BE in today's WA is fairly more complex than shown in Fig. 2, BE is not in the main concern of this thesis and thus I will describe parts relevant to FE development and at a level necessary to understand how it influences FE performance.

BE usually consists of three parts:

- **Web server**, a computer running all necessary software to convey communication between user and web application possible.
- **Database**, stores data and reacts to a Create, Read, Update, Delete (CRUD) operations of application.
- **Application scripts and services**, decide what actions should be carried out and how should they be done with respect to a current state of application.

Model of WA with extended back-end can be seen in Fig. 7.

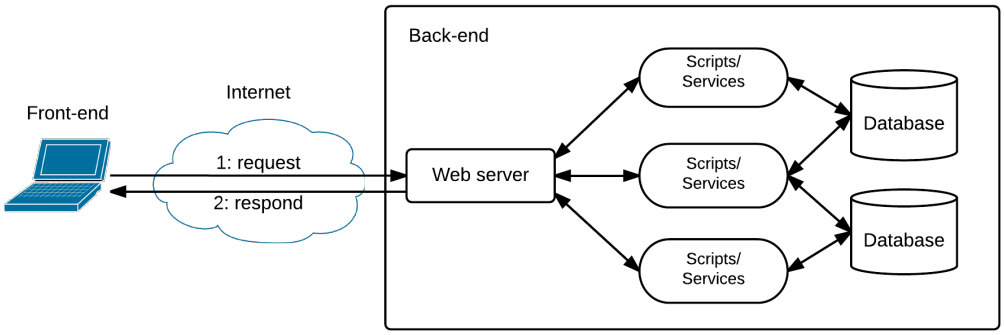


Figure 7 Web application model with extended server-side

2.4 Hypertext Transfer Protocol

HTTP is a protocol describing how browsers and servers communicate with each other over the Internet in order to exchange data. The process works as follows [15]:

1. Establishment of a **connection** between client and WA server.
2. If the connection is successful **request** is sent by the client. It is a message requesting data from a WA server. The data can be of various types - web pages, images, client-side scripts, confirmation of user authentication and more.
3. The server then sends **response** which contains requested data together with a status code. There are several types of status codes depending on success or failure of the requested operations.
4. Last step **closes** the connection by either both parties.

Both, request and response, are text based messages, each message has three parts [15]:

- an **initial line**, defines mainly source of information,
- **header**, zero or more lines specifying additional parameters of the request (see 2.4.2),
- an optional message **body**, contains data sent by client/server.

2.4.1 Types of HTTP request

When user access the WA for the first time, browser sends an *unconditional* HTTP requests to a server. Server sends back the requested data and the browser may cache the data for later use, if the response's header allows it.

If a subsequent request is made, *expires* and *max-age* parameters of the copied response are checked in order to determine whether the resource is fresh or not. If the cached copy is not expired, it is used and no HTTP request is made.

On the other hand, if the resource is expired, the browser sends a *conditional* request to a server to check whether the resource has been modified since last request or not. The request contains If-modified-since header field indicating last version of the resource in a cache. The server returns either response containing Not Modified header, signalling that the cached copy is up to date and therefore can be used, or response containing new data in a body of the response.

2.4.2 HTTP header fields

Following is a list of HTTP header fields whose setting relates to a performance of FE:

- **Expires** field containing date/time information when the response becomes stale.
- **Cache-control** specify the maximum time in seconds to cache response in a client's memory. If browser supports expires and cache control and both are present, according to HTTP specification [15], cache control parameter is preferred.
- **Last-modified** information specifying date/time of last modification of requested file.
- **If-Modified-Since** used when conditional request is made (see 2.4.1).
- **Accept-Encoding** is list of compression algorithms supported in a browser.

Definition of all available header fields can be found on [16].

2.5 JavaScript Object Notation

As mentioned in section 2.4, HTTP responses/requests can serve for sending data. JavaScript Object Notation [17], mostly called JSON, is a human-readable lightweight data-interchange format using two structures to transmit data from server to client, those are:

- **Object** - a collection of name/value pairs, where individual pairs are separated by a comma. An object is wrapped into curly brackets "{" and "}" (see Fig. 8).
- **Array** - an ordered list of comma separated values. An Array is wrapped into square brackets "[" and "]" (see Fig. 9).

```
{
"name":"value",
"name":"value",
...
}
```

Figure 8 JSON object example

```
[
value,
value,
...
]
```

Figure 9 JSON array example

Values, either in object or array structure, can be of following data types:

- String
- Number
- Object
- Array
- boolean values
- null

More complex object are created by nesting values. Example JSON consisting of objects, arrays and nested values is shown in Fig. 10.

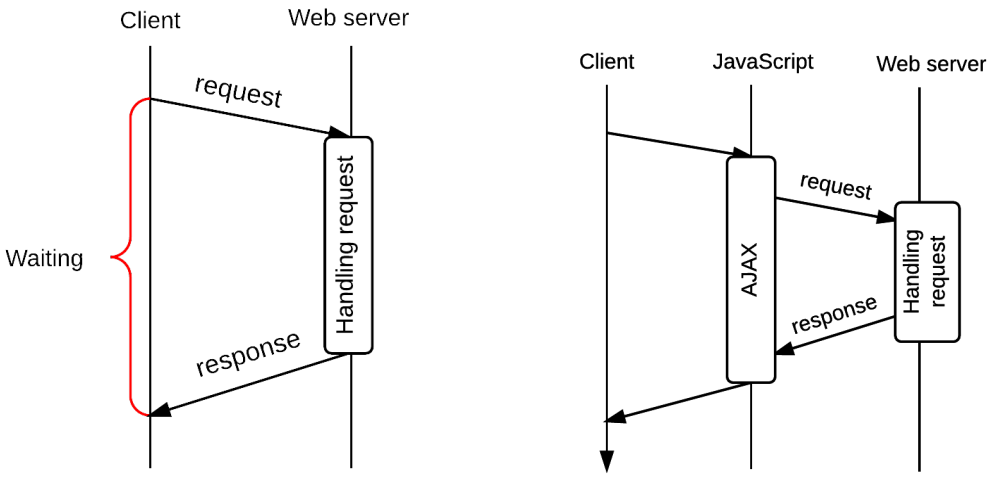
```
{
"JSON":{
  "name" : "Example JSON",
  "type" : "object",
  "array" : [
   {"name" : "First", "type" : "nested"},
   {"name" : "Second", "type" : "nested"},
   {"name" : "Third", "type" : "nested"}
  ],
  "itemsInArray" : 3
  "isJSON" : true
  }
}
```

Figure 10 JSON object example

This way of storing information permits data interchange among all programming languages, therefore JSON can be used to send data between FE and BE regardless of the technology used [18].

2.6 AJAX

AJAX [19], is an acronym for Asynchronous JavaScript and XML ([20]), is a group of technologies working together in order to create web pages more interactive. Using AJAX allows browser to make asynchronous HTTP requests to a server, obtained response is processed by provided AJAX APIs into a JS code and later the data can be used to update web page content (e.g. using DOM operations) without reloading the entire page. Fig. 11a demonstrates flow of user activity while normal HTTP call is used. When request is made, browser is waiting for response from a server. During this period it seems to the user that browser is not working. On the other hand, AJAX makes HTTP calls asynchronously therefore user activity can be continuous as shown on Fig. 11b. When event requesting HTTP call occurs, AJAX functions are called so that request can be made, but browser does not become irresponsible. AJAX makes HTTP call behind the scene and as soon as response from server is obtained client carry out appropriate actions.



a) User activity flow while using synchronous HTTP calls.

b) User activity flow while using AJAX to make HTTP calls.

Figure 11 Schema of typical HTTP call and HTTP call using AJAX

2.7 JavaScript libraries and frameworks

In this section JS libraries and frameworks are introduced with respect to FE development and OU Analyse dashboard.

2.7.1 Libraries

JS library is a library of pre-written JS. It is very useful to use libraries while developing WA since they make development of common features less time-consuming [21].

jQuery is the most popular JS library [22] for JS scripting emphasizing support of all main browsers[1]. It provides plenty of functions for DOM operations, handling of events, AJAX calls, JSON operations and more. jQuery library is also extendible for providing functions not yet implemented or improving already implemented functions. It can be done by using so called "plug-ins", which are basically another JS library dependent on jQuery functions. jQuery functions can be called either using jQuery or dollar $ sign. There are two types of jQuery functions:

- **Object functions** - when these functions are called, an jQuery object is returned. This object can be used in another jQuery function, which makes chaining of functions possible, because all object functions return jQuery object.
- **Utility functions** - these are useful for accomplishing routine programming tasks [23]. Even though browsers natively implement some of these functions their behaviour differ from one to another and using jQuery removes these behaviour inconsistencies. An example of this problem is function *trim* which removes whitespace characters from beginning and end of a string. This function is not implemented by Internet Explorer 8, but using jQuery utility function trimming can be achieved (Fig. 12).

[1] jQuery home page - http://jquery.com (visited on 19/05/2015)

```
1 $.trim("    To be trimmed    ");   Resulting in "To be trimmed"
```

Figure 12 Example of using utility function using trim function

Fig. 13 demonstrates a typical using of jQuery library. Other examples are used throughout the rest of the work.

```
1  $(document)                          Use jQuery and select HTML document
2  .ready(function(){                   When document ready execute following
3
4  var count = 0;                       Variable count initialization
5
6  $.each(                              Iterate over elements defined in
7    $("ul").find("li"),                first argument and apply function
8    function(){count++;}               specified in second argument on
9  );                                   each of them
10
11 $('body').prepend(                   Prepend to BODY element
12   $('<div>').text(                   DIV element
13 "List with " + count + " items")      with following text
14 );
15
16 $("#example-btn")                    To elements with specified ID
17   .css("height", "50px")             Set following CSS styling
18   .css("font-style","italic");
19 });
```

Figure 13 jQuery example code (left) and explanation (right)

Fig. 14 shows how example code (Fig. 13) changed the previously shown example of HTML (Fig. 5).

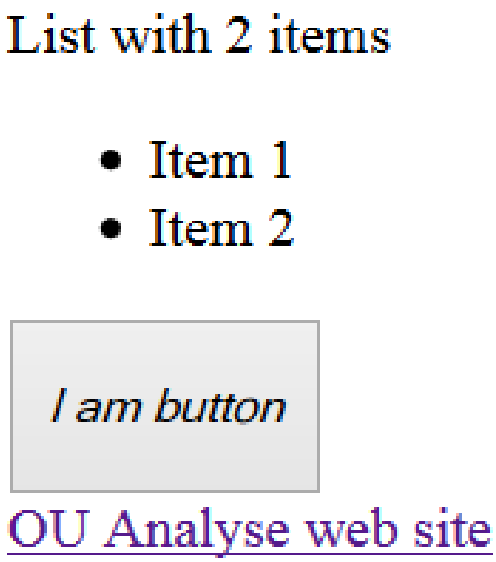


Figure 14 Demonstration of jQuery functions

2.7.2 Framework

JS libraries are often referred as frameworks and even though framework is also set of pre-written JS codes there is a difference between them. If application calls library function, its result is carried out by the application. On the other hand, framework deals with all these steps and developer just specifies particular parameters of framework's behaviour.

2.8 Best practices

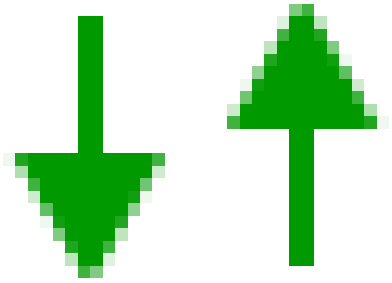
Following best practices are, in my experience, important while developing an application. This section introduces best practices concerning the development of WA.

One of the measures of good quality WA is its response time [1]. According to [1] only 20% of total response time is spent for retrieving the HTML document and the rest (80%) is spent on parsing it, requesting additional page components (JS scripts, CSS style sheets, images etc.) and putting all together.

Since loading every component means to establish one HTTP connection, reducing number of connections leads to improved response time. First few best practices from following list demonstrates how to reduce the number of connections. Second set of best practices explains how to make the HTTP communication, itself, faster. Last few best practices help to write HTML document in a way that downloading of additional components (JS code, CSS styles, etc.) does not slow down web page in its rendering.

2.8.1 Minimize number of images

CSS sprites is method for reducing the number of images by combining them into less files. This can reduce the number of HTTP requests needed to load all images to only one request. Also, size of sprite file is often smaller than sum of file sizes of individual images [1]. CSS sprite image[2] combining two arrow images and CSS code needed to using the sprite is shown on Fig. 15.



```
.arrow-parent {
      background:url(arrows.png);
}
.green-down{
  background-position: 0px 0px;
}
.green-up{
  background-position: -18px 0px;
}
```

Figure 15 CSS sprite image of two arrows.

2.8.2 Minimize number of JS and CSS files

Reducing number of JS and CSS files saves another HTTP requests. It can be done by using simple concatenation of files without the loss of any information. However it is important to be aware of file dependencies.

[2]CSS sprite was created using online tool - http://csssprites.com (visited on 20/04/2015). File size of the CSS sprite is 188 bytes, sum of file sizes of individual images is 482 bytes.

2.8.3 Use browser cache

As mentioned earlier, components with an indication of expiration are cached and reused in subsequent requests from a browser's cache with no unnecessary HTTP requests. In case the expiration is not set, conditional request is sent and even if the response contains a few header lines (in case the cached copy had not been changed), those conditional requests adds up. Proper setting of expires and max-age header field for all types of resources can avoid conditional requests.

2.8.4 Write external JS and CSS

Saving JS and CSS code into external files rather than writing them as inline code [1] is another good practice. Modern browsers support downloading about six files in parallel [24] and even if it means more HTTP requests parallel download is generally faster than sequential. Also, JS and CSS code is static and therefore suitable for caching.

2.8.5 Compress HTTP response

Vast amount of response time can be saved using compression. When HTTP request is made, browser adds *Accept-Encoding* header field with list of supported file formats. If server accepts any of these formats, file which is requested is compressed and sent in response along with the type of compression method used in header. Gzip is the most popular file format and application used for compression and decompression of files on the web [25]. Using Gzip can save about 70% of file size resulting in reducing response time about 50% [1].

2.8.6 Minify JS code

Second method helping to reduce size of files is minification. It is a process during which unnecessary characters, mainly comments and white-space characters, are removed from file [1]. The size of file is reduced and as a consequence the response time is improved. Basically any component can be minified but [1] shows that only JS files are worth to do so. Combination of minification and Gzip compression reduces size of original file about 75%.

2.8.7 Use a content delivery network

Content delivery network (CDN) is network of interconnected computers across many locations worldwide [26]. The goal of CDN is to response to a request as quickly as possible by choosing server which is the most suitable at the moment of request. The choice is based on location of client and server, number of redirection between them, usage of a server and many more. Many libraries and frameworks can be found on Content Delivery Network (CDN) servers and can be included to the project. It decreases, in general, the response time of a page [1].

2.8.8 Put your style sheets in the document HEAD

In order to prevent unnecessary redraws if page style changes, some browsers do not render a page until all CSS files are loaded [1]. During this time just white screen is shown to a user, creating illusion nothing is happening. This problem can be resolved by moving *<link>* tags into *<head>* of HTML document, which makes browser to render page progressively [1].

2.8.9 Move JS scripts to the bottom of the page

When JS files are downloaded, browser immediately executes them. While file is being executed, browser stops rendering other content until execution of the script is finished. It may result in the same problem as with CSS file - white screen. If script is in the head of HTML document, parsing of HTML will be stopped and so nothing is shown to a user. If all JS scripts are moved to the bottom of a page, browser will render entire page and then JS files will start to be executed without influencing rendering of a page.

2.9 Responsive design

WA is accessed not only from computers but also from mobile devices [27]. These devices have diverse screen sizes thus web pages might be rendered inappropriately.

Responsive design is a method of styling web pages depending on a screen size. It can be implemented using CSS media queries [28], which are CSS rules for adjusting content of web page optimally based on a screen parameters, e.g. some content can be hidden on small screens while extra content is shown on big ones.

3 OU Analyse web application

This chapter introduces OUA web site. First part describes all web pages one by one, second part presents technologies the web pages are based on and finally performance analysis of the WA is made in last part.

3.1 Contents of OUA dashboard

Following is list of web pages the OUA WA consist of, content of each web page is described component by component together with snapshots:

- **Project info page** - OUA introductory page consisting of project description, buttons for accessing the dashboard application, introduction of members of the OUA team, list of publications related to OUA and contacts to OUA team. Fig. 16 shows small part of the project info page.

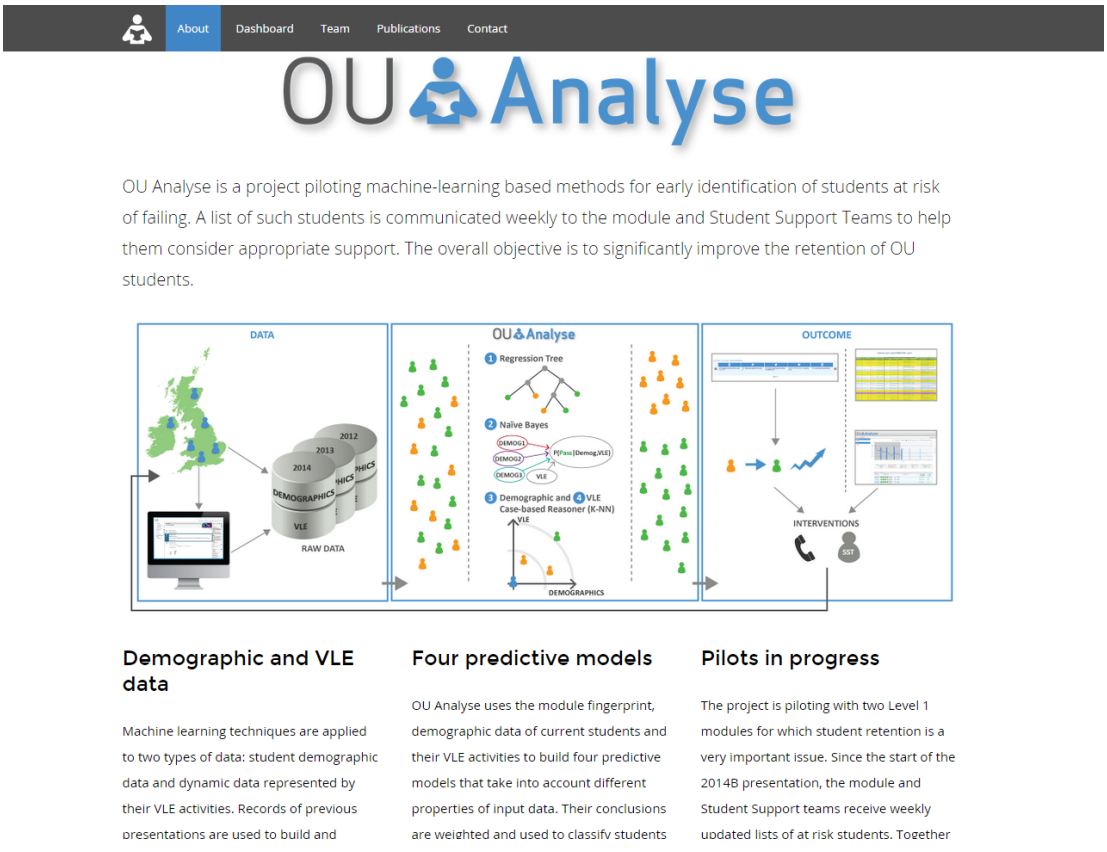


Figure 16 Part of project info page (original version)

- **Module and student overview pages** - pages showing important information about courses and students. Both consist of following parts:
  - **Top menu** - contains project logo and menu used to navigate between modules and their presentations (Fig. 17).

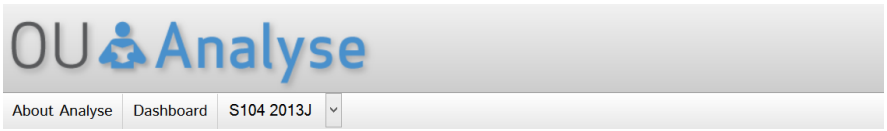


Figure 17 Top menu of module and student overview pages

– **Left menu** - contains other navigation options and servers as a space for adding links to new features (Fig. 18).

Figure 18 Left menu of module and student overview pages

– **Central part** - contains main content of the page. While top and left parts are the same for both types of pages, the main content is different, see sections 3.1.1 and 3.1.2 for more details about module and student overview, respectively.

- **Service pages** - set of pages shown, when authentication is needed or in error occurs during loading. OUA implements three service pages:
  – Authentication page (Fig. 19)

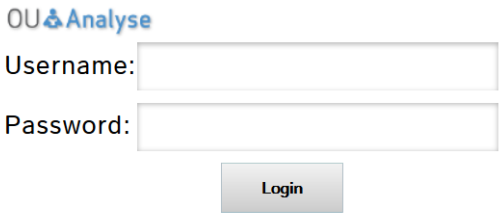


Figure 19 Login form in original implementation

  – Permission denied page (Fig. 20)

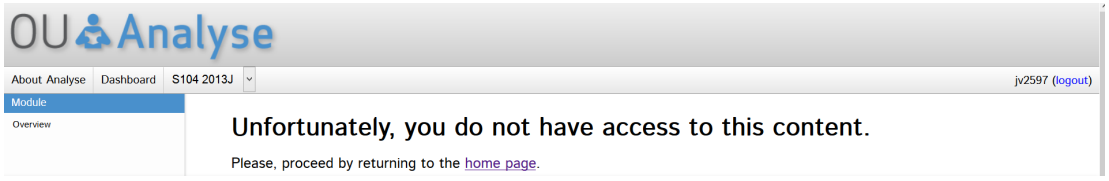


Figure 20 Error page in original implementation

  – General error page (Fig. 21)

Figure 21 Error page in original implementation

3.1.1 Module overview

Module overview page contains three components:

- **VLE graph** - displays VLE activities of students in the current and previous presentation. Graph also shows results of Tutor Marked Assignments (TMA) for current and previous presentation (Fig. 22).

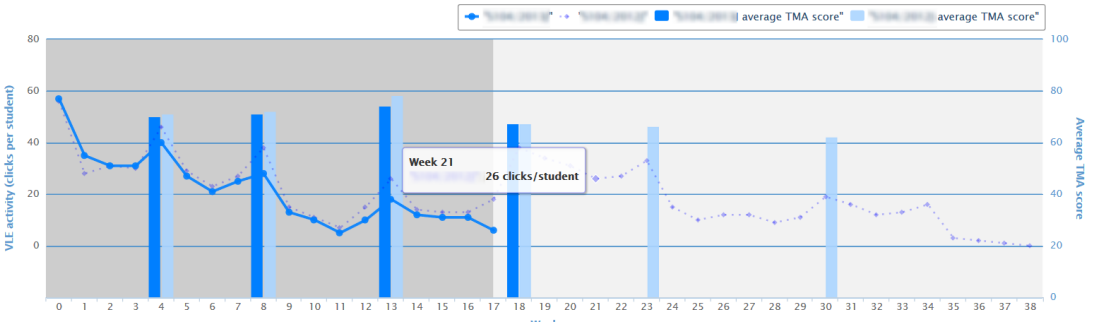


Figure 22 Graph of VLE activities and TMA results on module overview page.

- **Trends** - this component shows at a glance current state of a course both graphically and as text (Fig. 23), e.g. number of registered students, number of student at risk of failing in next TMA etc.

Figure 23 Component emphasizing important trends in a course.

- **Students table** - lists all students attending course displaying their scores from previous TMAs, prediction for the next TMA and prediction of the final result. Students in the table can be filtered using various criteria - gender, occupation, highest education and more (Fig. 24).

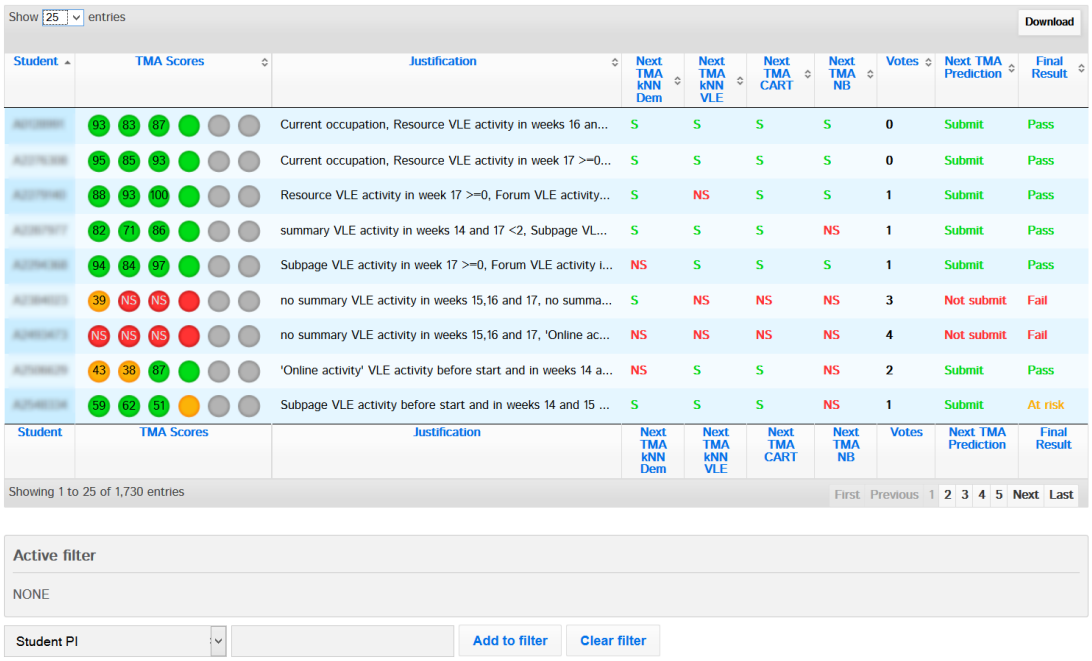


Figure 24 Table displaying all students attending a course. Information about results of TMAs, prediction for next TMA and prediction of final result are shown.

3.1.2 Student overview

By choosing particular student from the students table, user is redirected to student overview page consisting of following components:

- **VLE graph** - student VLE activity and TMA results compared to whole cohort of students in a course (Fig. 25).

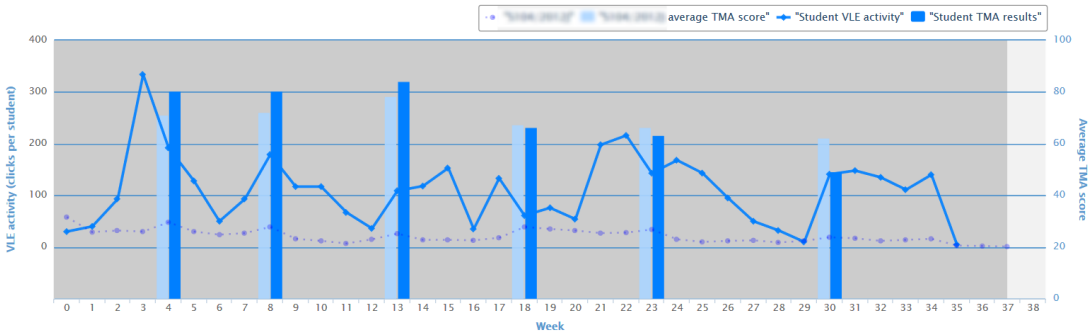
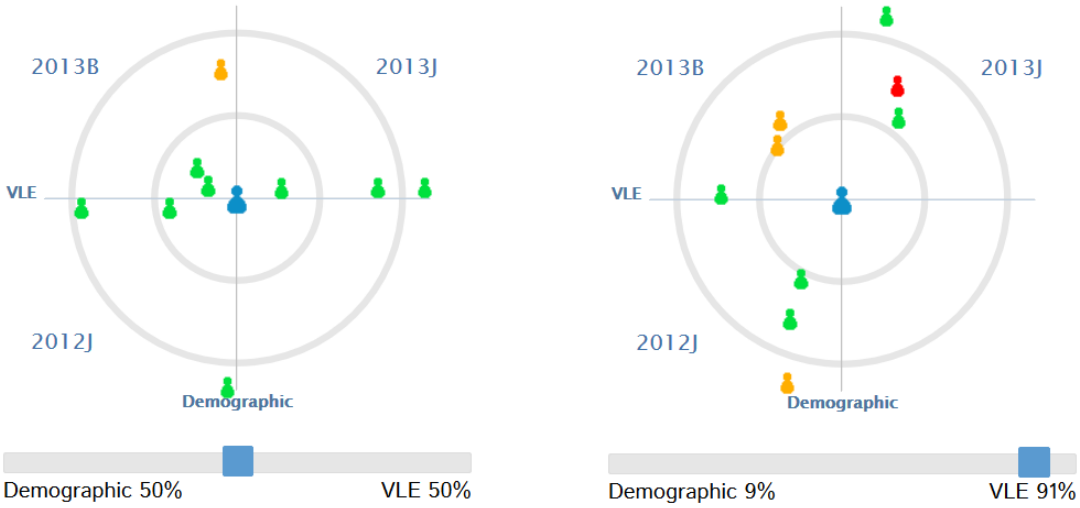


Figure 25 Graph of VLE activities and TMA results of a student.

- **Nearest neighbours** - graph (Fig. 26a) of students who are the nearest ones to the chosen student based on demographic data and VLE activities in a course [29]. Using slider at the bottom, user can give more weight to either demographic data or VLE activities and consequently the component is updated (Fig. 26b).



a) Nearest neighbours component. **b)** Weight given to the VLE.

Figure 26 Nearest neighbour component

- **Score overview** - table showing real and predicted results of TMAs together with the justification of the prediction (Fig. 27).

Assignment	Prediction	Real	Justification
TMA 1	Submit	44	Resource VLE activity in week 4 >=0 Homepage VLE activity in week 4 >=0 summary VLE activity in week 4 >=0
TMA 2	Submit	58	URL VLE activity in week 8 >=0 URL VLE activity in weeks 7 and 8 >=0 URL VLE activity in weeks 6 and 8 >=0
TMA 3	Submit	49	no summary VLE activity in weeks 9,10 and 13 no Homepage VLE activity in weeks 9,10 and 13 no summary VLE activity in weeks 9,12 and 13
TMA 4	Submit	NA	Homepage VLE activity in week 15 >=0 summary VLE activity in week 15 >=0 URL VLE activity before start and in week 17 >=22

Figure 27 Table showing real and predicted results of TMAs for particular student

- **Recommender** - set of activities student should do in order to succeed in the course (Fig. 28). Control elements on both sides and at the bottom of the recommender component serve to slide to more recommendations.

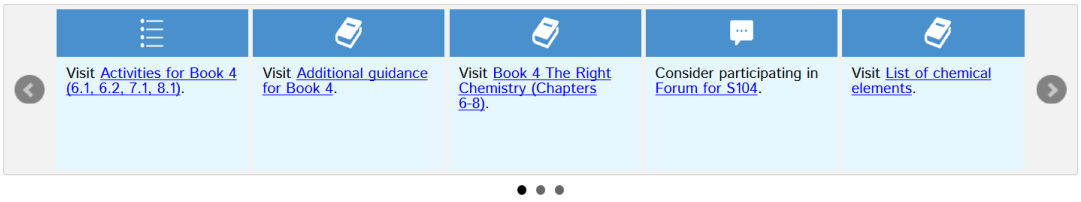


Figure 28 Component recommending activities activities aimed at helping students with passing the course.

3.2 Technologies behind the components

Tab. 2 divides components introduced in this chapter into several groups based on their type.

Type	Component
HTML	Project info page
	Service pages
	Navigation menu
Graph	Module overview VLE
	Student overview VLE
	Nearest neighbours
Table	Students table
	Score overview
Carousel	Trends
	Recommender

Table 2 Components of OUA and their implementation method

Each category is implemented using different technology:

- **HTML** - Project info page design is based on HTML and CSS template available at [30]. The other parts, namely trends and navigation menu, are coded by OUA team.
- **Graphs** - are built on Highcharts JS library[1].
- **Table** - jQuery plugin Datatable[2] is used.
- **Carousel** - Another plugin bxslider[3] is responsible for function of activity recommender component. The trends component is coded by OUA team.

3.3 Analysis of OU Analyse

This section investigates OUA implementation, its responsiveness and compatibility across browsers.

3.3.1 Best practices

YSlow [31] is a tool evaluating the rate of implementation of best practices introduced in section 2.8. It grades each best practice by mark from A to F, where A is the best. Tab. 3 shows results of YSlow tool applied on all OUA web pages[4]. It can be seen that many of best practices are poorly graded therefore should be improved in order to increase performance of the web site.

[1] Highcharts home page - http://www.highcharts.com (visited on 21/05/2015)
[2] Datatable plugin home page - http://datatables.net (visited on 21/05/2015)
[3] bxSlider home page - http://bxslider.com (visited on 21/05/2015)
[4] Grade in a table is calculated as average of partial grades on each page.

Best practice	Grade
CSS Sprite	F
Minimize number of JS and CSS files	C
Expires & control-cache	F
External JS and CSS	A
Using Gzip	F
Minified JS	E
Using CDN servers	F
CSS in head	A
JS in bottom	D
Cached AJAX	A

Table 3 Table demonstrating scope of implementation of best practices based on YSlow tool grades.

3.3.2 Responsiveness of OUA

The only responsive page on the OUA web site is project info page. HTML and CSS code of the page is based on a template [32], which is a small framework providing responsive design functionality. Fig. 29 displays different layout of the page viewed from small screen devices (compare to Fig. 16). Without the layout reorganization components would look distorted, e.g. it would be hard to read the text, because each column of text would be too narrow.

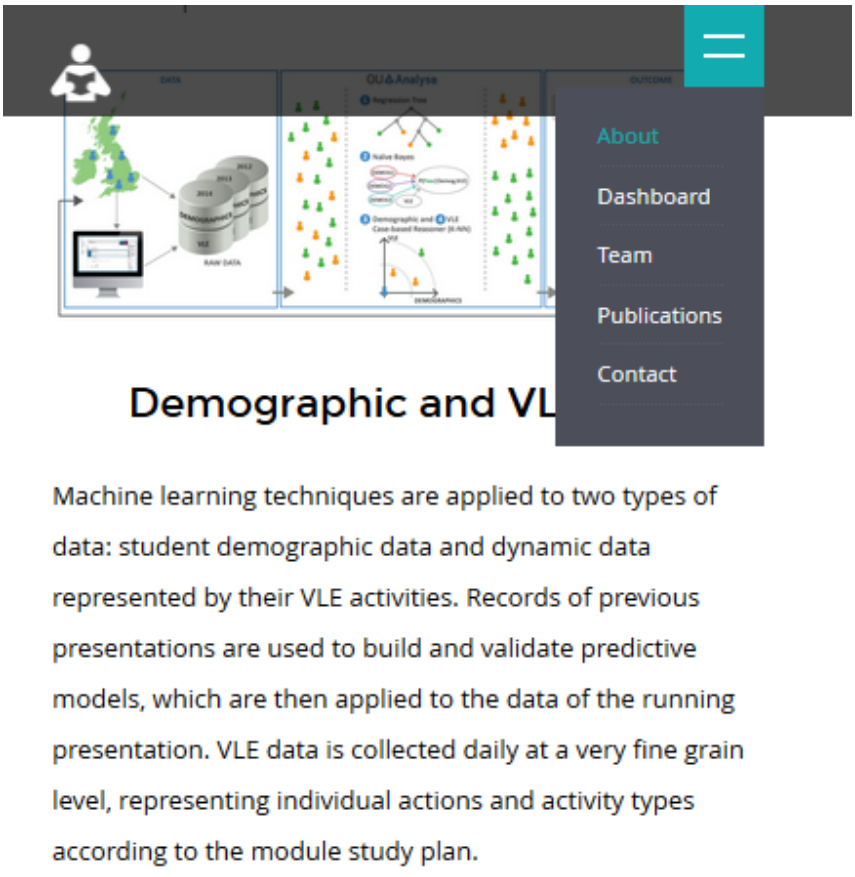


Figure 29 Project info page view from small screen device.

Other pages does not support responsive design. This leads to following issues, when pages are viewed from small screen devices:

- **Left menu is overlapping the main content** - the table becomes too wide to be completely shown on a screen, therefore user has to scroll to see hidden content. When this happens, the left navigation menu is also overlapping and partially covering other parts of the page (Fig. 30).

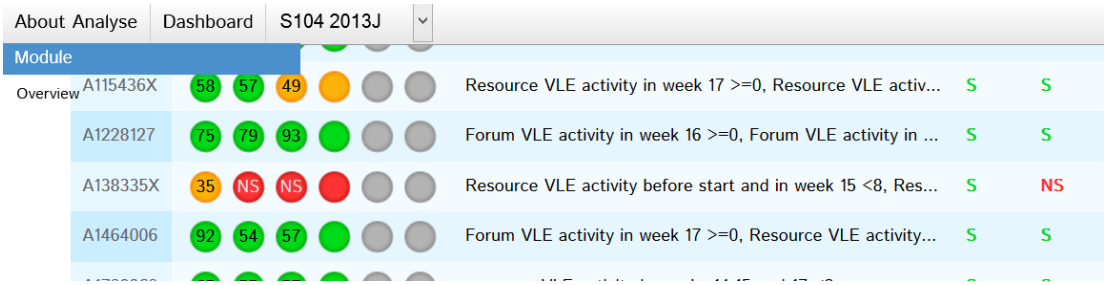


Figure 30 Module overview page viewed from small screen device.

- **Components become too narrow** - this issue concerns trends and recommender components. One trend from the trends component is missing and the others are too narrow. Also, trends overflow the specified area. (Fig. 31).

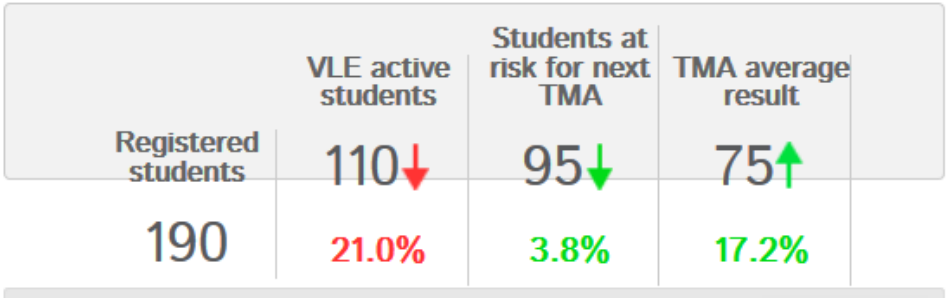


Figure 31 Trends component displayed on small screen device.

Even if recommender is implemented using Bxslider library, which claims to be responsive, the individual recommendations become impossible to read (Fig. 32).

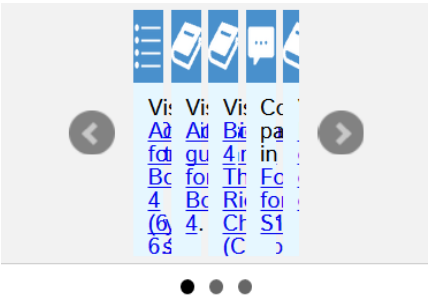


Figure 32 Recommender component displayed on small screen device.

3.3.3 Compatibility across multiple browsers

In majority of web browsers, the original version of OUA is displayed correctly. However, Internet Explorer browsers prior to version 9 do not support several CSS attributes

and JS functions are missing or behave differently than in other browsers. However, it is vital that the OUA dashboard supports also Internet Explorer 8 (IE8), because majority of OU staff, who are the main users of the dashboard, are required to use IE8.

Project info page in IE8 is almost without errors, the only problem is with navigation menu, which is positioned vertically instead of horizontally. Also, layout of some components is incorrect (Fig. 33).

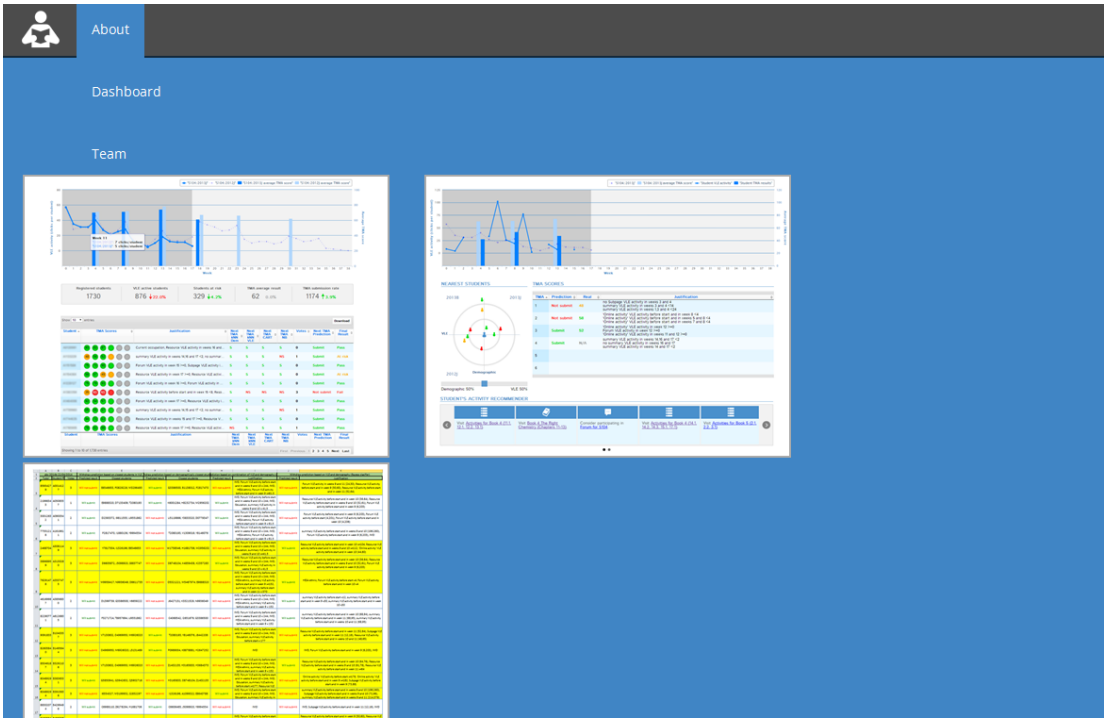


Figure 33 Project info page displayed using IE8.

Module and student overview pages do not work as expected, because of using unsupported JS functions. When such function is reached, IE8 stops executing any other JS code, which results in a blank space and other issues Fig. 34.

3.4 Summary

One of the main tasks of this work is to look into the weaknesses of the OUA project. Following is a summary of the problems in the original implementation of OUA, which are shown in this chapter:

- **Design** - The figures in section 3.1 shows, that web pages have different design implementation, which may lead in confusion of the user in terms of using the web site.
- **Responsiveness** - It has been shown that the web pages does not render on devices with different screen sizes properly. Solution of *design* and *responsiveness* problems is described in section 4.1
- **IE8 compatibility** - Since most of the users access the WA from IE8, its support is a crucial requirement for the new version. Section 4.2 deals with the problem.
- **Performance** - YSlow analyser graded many of the web site properties poorly, which leads to slow response time of the pages. These problems are tackled in section 4.3.

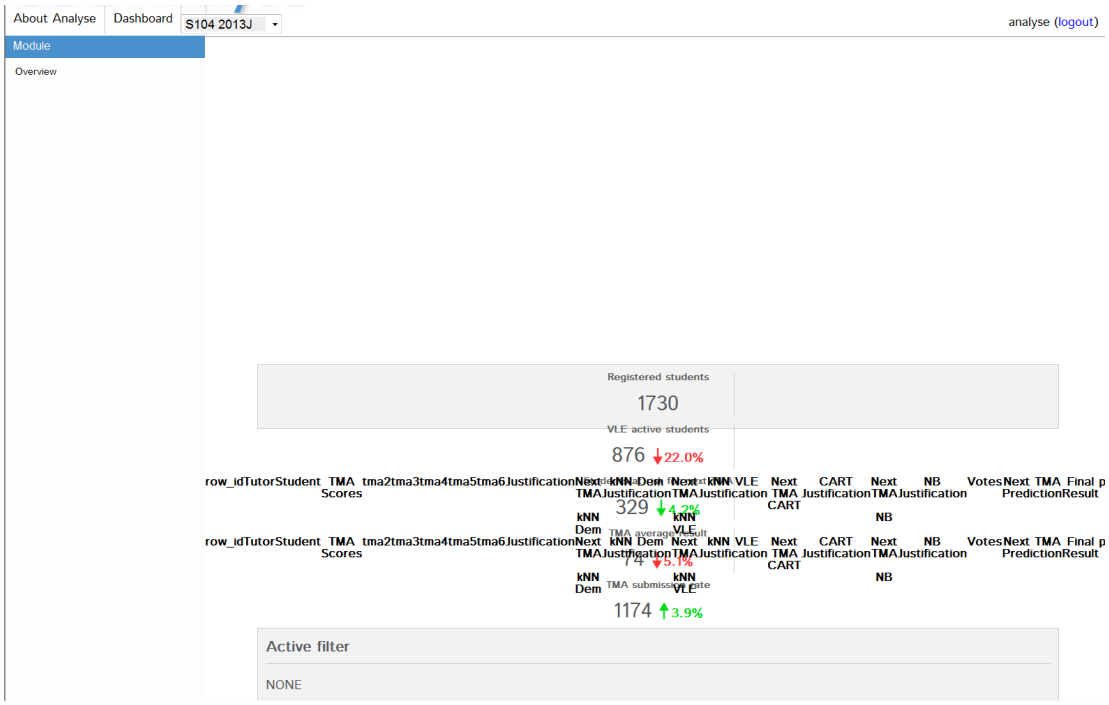


Figure 34 Project info page displayed using IE8. Problems are with navigation menu, which is positioned vertically, as well as with the layout of some components.

4 Practical part

In this chapter new version of OUA application is implemented reflecting the weak points demonstrated in chapter 3.

4.1 CSS framework

It can be seen from figures in previous chapter, that the design of OUA web pages is not uniform, e.g. pallets of colors on project info page and dashboard pages are different. One of my first tasks during the work on the OUA project was to choose the most suitable CSS framework and convert the original version of OUA using new framework. Using it significantly simplifies creating unified design across web site [27]. Also, unified design across all pages of a web site can positively increase user experience [1].

4.1.1 Choosing a framework

Important criteria while choosing a new framework:

- Broad browser support - Framework should support major browsers (Tab. 1) including IE8.
- Responsive design - framework is responsive across variety of available devices.
- Size - The smaller framework file size is, the faster a web page is downloaded, therefore user can sooner interact with the page.
- Support - Large user community is useful when an issue occurred and help is needed.

Of all frameworks supporting IE 8 [33], Bootstrap [34] is the one with the largest community of users. It also provides full supports for responsive design creation. The disadvantage of Bootstrap is slightly bigger size compared to other frameworks providing similar number of functions. Despite that, I have chosen Bootstrap framework for its IE8 support and broad support user community.

4.1.2 New project info

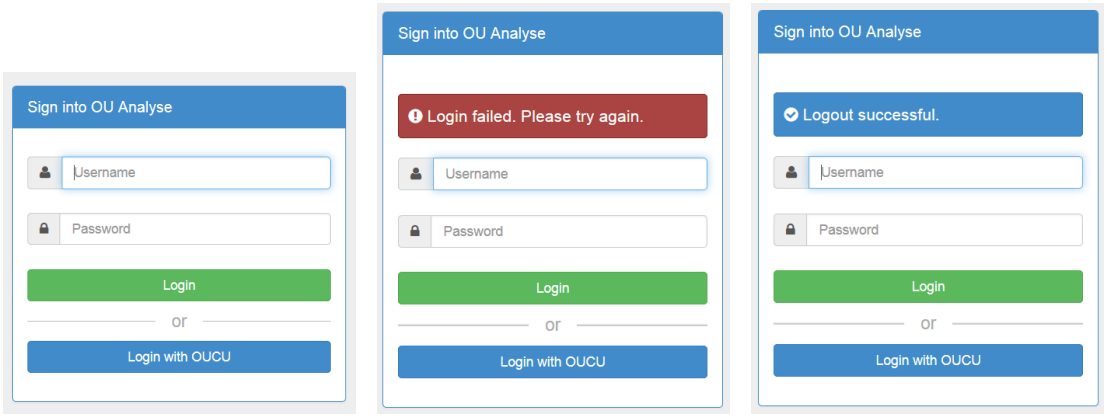
Syntax of the CSS framework[1] used to built the project info page is similar to syntax used in Bootstrap therefore conversion of project info HTML code was straightforward. The conversion was done by swapping of corresponding CSS classes and HTML elements. Since one of the requirements of using a CSS framework was to keep the same design of the page, new version looks the same.

4.1.3 New login page

In new version of OUA, users must login to WA using either OUA account or using university account. Authentication with the university account can be done only through university web site, therefore an additional button had to be added to the original form.

[1] Free Awesome Website Templates - http://www.styleshout.com (visited on 19/04/2015)

Design of the form including the new button is shown in Fig. 35a. Furthermore, an indication of a failed login (Fig. 35b) and a successful logout (Fig. 35c) has been added.



a) Normal **b)** Failed login **c)** Successful logout

Figure 35 Login forms implemented using Bootstrap framework.

4.1.4 New permission denied and general error pages

The other service pages remain almost same. Small design changes were made and responsiveness was achieved by word wrapping text to screen size Fig. 36.

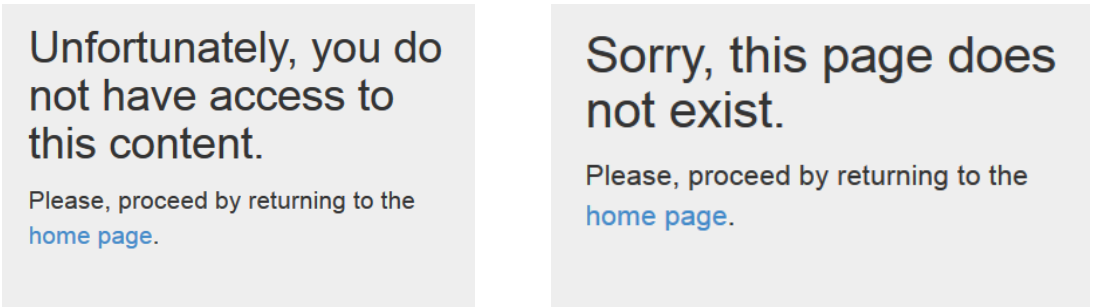
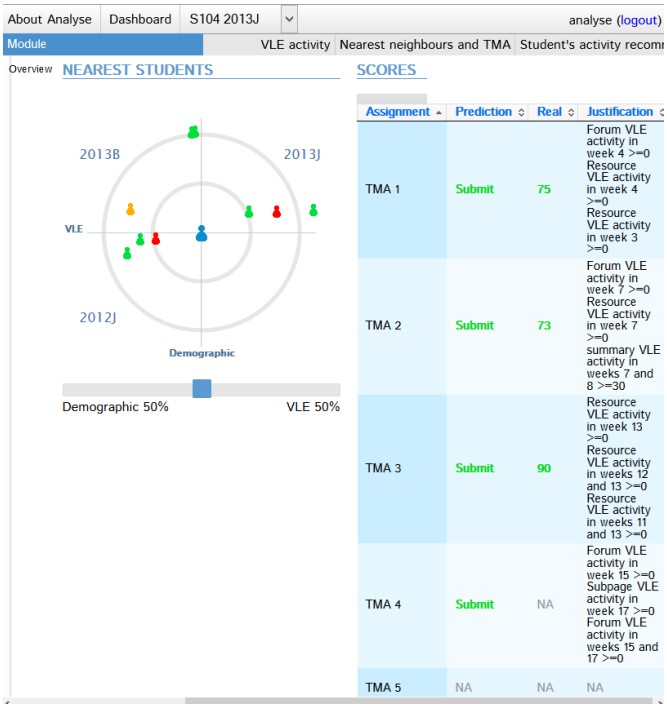


Figure 36 Responsive versions of permission denied page and general error page.

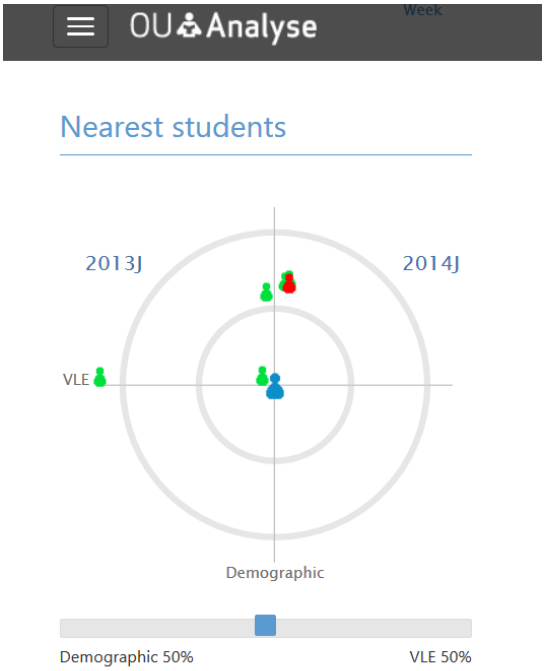
4.1.5 New module and student overview pages

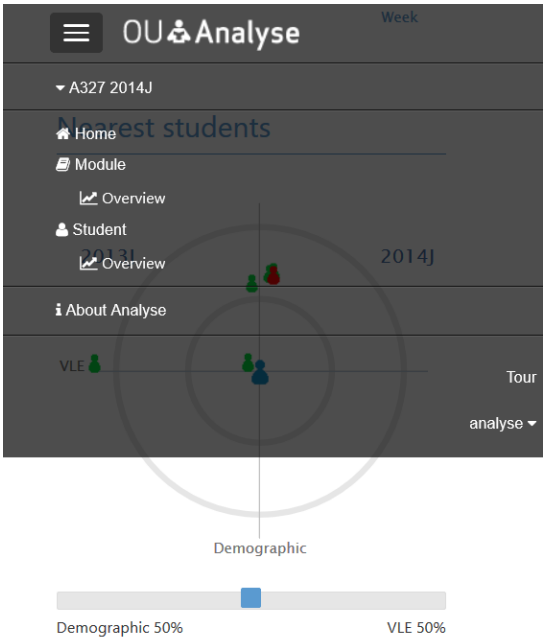
Major changes were done to both overview pages for the sake of responsive and consolidated design of the web site:

- **Making pages responsive** - when page is viewed from small screen device, left menu transforms into small button in top menu and thus whole screen can be used to show main content. Any content from left menu becomes available when the button is clicked (Fig. 37).



a) Original design





b) New design with closed navigation menu. **c)** New design with opened navigation menu.

Figure 37 New design of the navigation menu viewed from small screen device.

Position of nearest neighbours and score components in the original version are strictly fixed next to each other on a line. New version is able to reflect the screen size and move the score table under neighbours graph if they do not fit on a screen next to each other (Fig. 38).

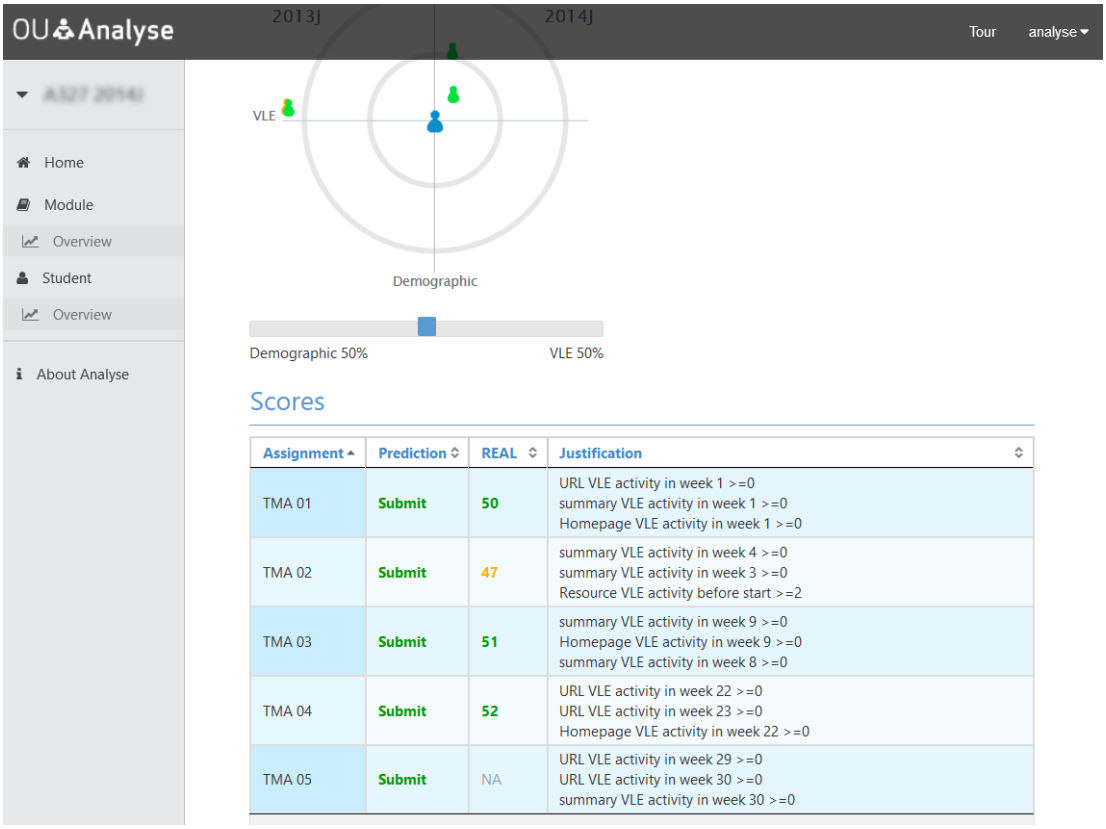


Figure 38 Neighbour graph and score table position on small screen

- **Design unification** - To unify the design, the top navigation menu from the project info page is used and the button elements from original top menu are moved to the left menu, because they do not fit to the top menu. Colour pallet of left menu and other components is chosen according to top menu styling (Fig. 39).

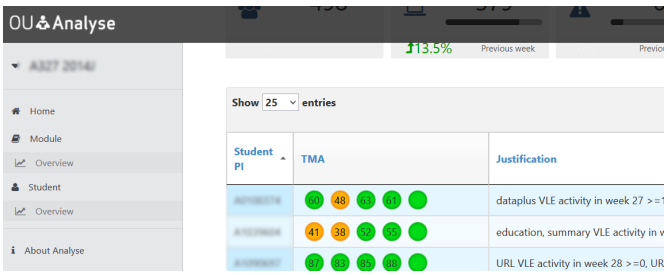


Figure 39 New layout of the top navigation and the left navigation menu.

- **Other adjustments** - Include mainly adjusting colour pallet to mirror project page and adding variable text size in order to increase responsiveness of the pages.

4.2 Solving IE8 compatibility issues

Section 3.3.3 outlines problems of the original version in IE8. Issues on project info page are fully removed by migrating to Bootstrap.

The overview pages do not work at all because they use JS functions which are not available in IE8:

- Trim - see Fig. 12
- Stringify - converts JS object into JSON string
- Parse - converts JSON string into JS object
- indexOf - searches for position in array of specified element

These had to be substituted, in order to make the page work. jQuery provides an alternative to all of the listed functions. Therefore using of the jQuery functions instead of the native ones solves the problem.

4.3 Performance improvements

Properties graded D and lower by YSlow tool (Tab. 3) has been improved in new version of the WA to increase performance of the web site. This section describes how these changes are achieved.

4.3.1 Expires header

OUA original version does not set neither *expires* nor *max-age* parameters in the HTTP response. There are different ways how to set the response headers. Because OUA runs in Apache Tomcat [35], change on the server needs to be done. Set of rules is defined, where each rule consist of two parts:

- **Content-type** - defines file type rule applies to, e.g. images or JavaScript.
- **Expiration** - defines how long a response is valid from the moment it is sent.

These rules are set on the server and every time a request is made, response with both parameters is sent based on the rules specified.

4.3.2 Gzip

Section 2.8.5 explains how WA can benefit from Gzip compression. The original version of OUA does not support HTTP response compression. Gzip needs to be enabled on the server in order to compress the HTTP response [36]. The server has defined a set of file types, which should be gzipped in case a client supports this type of compilation.

4.3.3 Minification

YUIcompressor tool[2] has been included into new version of OUA to accomplish minification of selected files. These files are minified during build process of the application. Furthermore, it can be set that minification procedure happens only for files which are used in production version. This is very useful since minified code is hard to read and it is more convenient to use not minified version during developing and debugging new features.

4.3.4 Others

Other properties require just a small change in the original implementation:

- **CSS Sprite** - created using the process shown in section 2.8.1.
- **Minimize number of JS and CSS files** - Besides minification of files, YUIcompressor tool provides another functionality - concatenation of files. In new version of OUA JS and CSS files can be merged into one JS and one CSS files using YUIcompressor.

[2]YUIcompressor home page - http://yui.github.io/yuicompressor (visited on 15/05/2015)

- **Using CDN servers** - Unfortunately, CDN servers are quite expensive and so whole OUA application cannot be moved entirely on CDN server. However, all the JS frameworks and libraries used in new version can be found on free CDN servers. Hence, new version download majority of JS and CSS files from one, namely [37].
- **JS in bottom** - 2.8.9 refer about importance including JS files at the bottom of *<body>* tag. Overview pages of the original version include JS files in the *<head>*. New version has all *<script>* tags at bottom of the HTML document as section 2.8.9 suggests.

4.4 New features

Previous sections of this chapter show implementation changes which deal with the problems in original version of OUA. This section introduces extra features of the new version, which help to increase user experience but also to simplify development of another features.

While choosing a new framework, the same criteria applied while choosing the CSS framework were considered, i.e. broad browser support, size of the framework files, support community and ability to reflect the screen size. In addition, compatibility with the already chosen frameworks, e.g. Bootstrap, is taken into account.

4.4.1 Mustache.js

OUA creates parts of HTML code dynamically both on server side and client side using JavaServer Pages[3] technology and JS, respectively. New version of OUA tries to move creation of dynamic content to a client. Mustache.js[4] is HTML template library used to create dynamic HTML. Process of using the library follow:

- **Mustache template** - Mustache template is HTML code, where dynamically changing parts of HTML are replaced with mustache tags. Tags are enclosed into double curly brackets "{{ key }}", where key is the tag's identifier.
- **Data specification** - JSON data to be put on a place of mustache tag identifier.
- **Rendering** - Conversion of Mustache template into fully functional HTML code is done using Mustache render function. The function matches key identifiers in JSON and Mustache template and assigns corresponding values.
- **Add to DOM** - HTML returned from render function is added to DOM, e.g. using jQuery.

Simple example code demonstrating using Mustache can be seen on Fig. 40a. Returned HTML code Fig. 40b is added to a DOM three resulting in Fig. 40c.

[3] JavaServer Pages is technology used on server side to create dynamic HTML content. See [38] for more information.

[4] Mustache.js home page - https://github.com/janl/mustache.js (visited on 20/05/2015)

```
<html>
<body>
<ul class="list"></ul>
<script id="example-template" type="x-tmpl-mustache">
<li>Item {{ position }}</li>
</script>
</body>
</html>

$(document).ready(function(){
  var template = $("#example-template").html();
  var result = [],position = 0;
  for(position=0; position < 3; position++){
    var JSON = {position:position};
      result.push(Mustache.render(template, JSON));
    };
  $("ul.list").html(result);
});
```

a) Code

```
<li>Item 0</li>
<li>Item 1</li>
<li>Item 2</li>
```

b) Resulting HTML code of list element

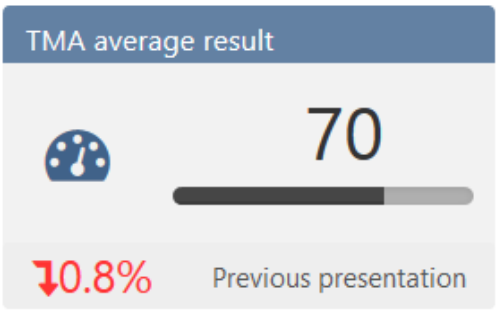
- Item 0
- Item 1
- Item 2

c) Code from Fig. 40 rendered in browser

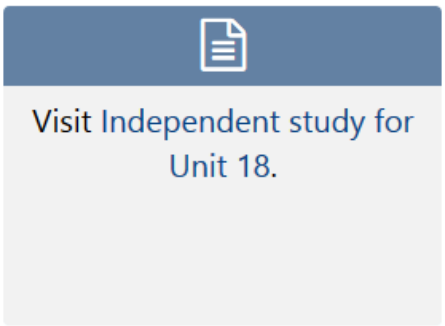
Figure 40 Example of using mustache

4.4.2 Owl carousel

Implementation of trends and recommender components turned out to be problematic in terms of responsive design. Since both consist of variable number of elements coupled into one component, they are reimplemented using Owl carousel[5] framework (OC). An element of trends (Fig. 41a) and recommender (Fig. 41b) has been created and implemented using mustache templates.



a) Trend tile



b) Recommendation tile

Figure 41 New design of individual elements in trends and recommender component

Rendering of trends and recommender component using OC is described on following lines:

[5] Owl Carousel framework home page - http://owlgraphic.com/owlcarousel/index.html (visited on 21/05/2015)

- **Data** - Data to be shown are obtained using AJAX in JSON form.
- **HTML** - Received JSON is adapted to fit mustache template, HTML code is generated and appended to DOM.
- **Carousel** - OC carousel is created and based of screen size, carousel displays just few elements so that they fit to the screen. The other are hidden and may be shown using control buttons either on bottom or both sides of the carousel. Number of elements and width of individual element depends on screen size. Fig. 42 and Fig. 43 are examples of recommender and trends components displayed on big screen, while Fig. 44 and Fig. 45 are rendered on a small screen.

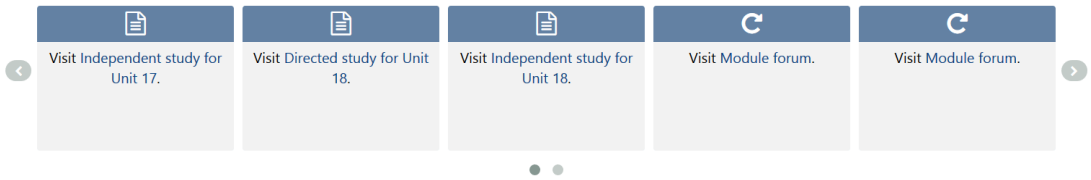


Figure 42 New design of recommender component view from big screen

Figure 43 New design of trends component view from big screen

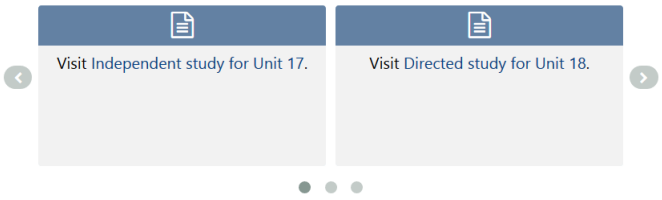


Figure 44 New design of recommender component viewed from small screen

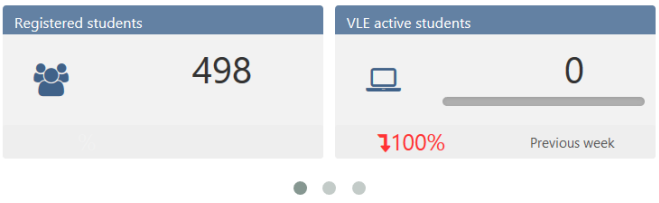
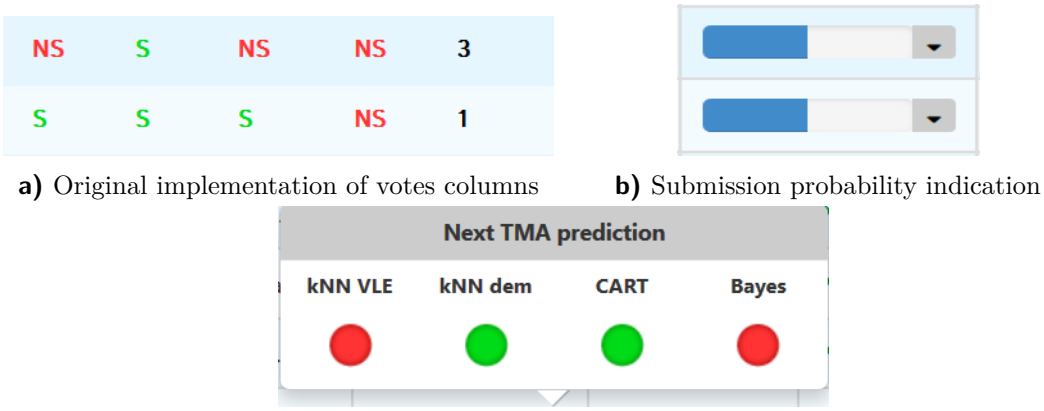


Figure 45 New design of trends component viewed from small screen

4.4.3 Votes indicator

OUA uses four predictive models for predicting student success. While making predictions, each predictive model votes whether the student submit next TMA or not. Then, voting is weighed and final decision is made. Individual decisions are shown in the students prediction table, where each vote is shown in separate column and an

additional column is added to indicate the sum of the votes (Fig. 46a). In order to retain both information, simplify the table and at the same time provide a graphical interpretation of the textual information, the original textual columns were substituted by a bar indicator (Fig. 46b). If the bar is clicked, extra window appears and displays the individual votes (Fig. 46c). Both information are presented in a table and table is clear. Furthermore, if new predictive model is used, it will be simply added to the extra window and the main table will remain without change.



a) Original implementation of votes columns

b) Submission probability indication

c) Extra table showing individual votes

Figure 46 New design of the way predictions are displayed in the students table.

4.4.4 Time machine

The results of predictions are sent to the module teams on weekly basis. However, the dashboard of the original version shows data just at current state of a course. New version of OUA is able to display data from the previous weeks. Analysing the former data may lead to increased precision while planning the interventions. This functionality is available on both the module and the student overview pages. A slider was added to the top of the page which allows the selection of the week which should be displayed (Fig. 47).

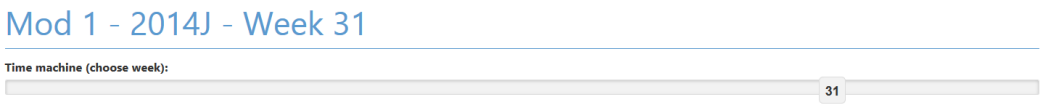


Figure 47 Slider component used to switch weeks.

When week is changed, reloading entire is unnecessary, because only data in individual component are different. Publish/subscribe pattern [39] is used to notify the components about newly selected week and each component carry out appropriate actions. VLE graphs does not require additional data because they display cumulated data for every week from the beginning of a course to the current state. Hence data for previous weeks, can be extracted from already available data. It means no HTTP request and update is almost instant. On the other hand data in other components are not that related across weeks and cannot be grouped as it is in case of VLE graphs. They have to be requested for every week individually. While new data is requested, loading indicator is shown (Fig. 48).

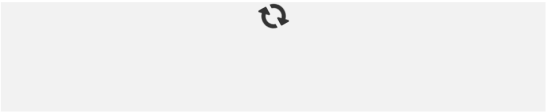


Figure 48 Trend component when content is loading from server. Rotating arrows are shown as a indication of loading.

4.4.5 Tour

OUA has quite a big number of components and features. In some moments user can be uncertain how to interact with them. In order to improve user experience, new version of OUA introduces a *tour*, which guides the user through the dashboard showing its features. It is built on Bootstrap Tour, plug-in to Bootstrap [6]. The tour consists of predefined steps, where each step explains one component or particular function of a component. Fig. 49 shows one step of a tour, which describes a button for navigation to the top of a page. Similarly, functionality of other components is explained. To navigate over the steps of the tour, user can use either control buttons (bottom part of Fig. 49) or left/right arrows on the keyboard.

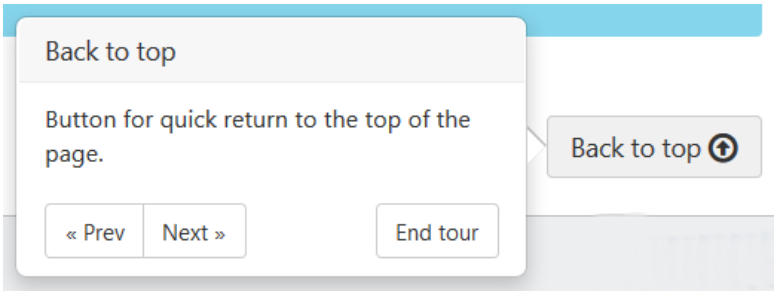


Figure 49 One step of tour

[6] Bootstrap Tour home page - http://bootstraptour.com/ (visited on 21/05/2015)

5 Testing

One of the main goals during work on OU Analyse, was to increase the performance of the original version of the OU Analyse web site. Previous chapters analyse weaknesses of the original version of OUA, suggest changes which should be made and how the changes have been implemented into the new version of the OUA WA. In this chapter, testing of the new version is made and comparison between the original and the new version is carried out.

Performance measurements of the web site were done using Network performance meter implemented as a part of Mozilla Firefox browser [1]. It measures response time from the moment when user sends an initial request for a web page to the moment when all content is loaded. Two results are obtained in one test:

- **Empty cache response time** - simulates the first access of a user and therefore all resources are loaded using unconditional requests,
- **Primed cache response time** - simulates subsequent access of a user and so cached files do not have to be downloaded again.

Response times of all web pages were measured in different versions of OUA WA, the versions are as follows:

- **Original version** - version of WA introduced in chapter 3.
- **New version** - version of WA where all changes introduced in chapter 4 have been applied except of the adding expiration header, gzip of HTTP response and minification.
- **Header version** - *new version* with expires and max-age HTTP response header set. Static files, e.g. JS files, expire after ten years from the moment request is made. Dynamic content, e.g. data for VLE graphs, are set to expire one week from the moment request is made.
- **Gzip version** - *new version* with gzipped HTTP response.
- **Final version** - *new version* with all previous changes applied.

Response time of each version was tested ten times on all three pages (results of the testing are available Appendix A). Based on these data, average response time is computed. Because response time depends on the speed of Internet connection. Tab. 4 shows only a comparison with the *new version*[2] expressed as a percentage change. The results are as follows:

- **Original version** - Neither *original* nor *new* version sets the expiration information in HTTP response therefore caching is not used and measuring time with primed cache is therefore irrelevant. Response time of project info page in new version is 15% faster, because there are more HTTP requests in original version. On the other hand overview pages are loaded faster, because the new version imple-

[1] Mozilla Firefox Network Monitor - https://developer.mozilla.org/en-US/docs/Tools/Network_Monitor (visited on 21/05/2015)

[2] New version is chosen, because it has the same settings of the server as the original version. Besides that new, header, gzip and final versions consist of the same files and therefore contribution of individual improvement can be measured more precisely.

ments new features and uses new additions frameworks, which results in increased page size and consequently longer response time.

- **Header version** - Loading pages with empty cache shows increase in response time by units of percent. This is caused by expiration information in the HTTP response, which makes the response size bigger hence more data is needed to be transferred. On the other hand loading with primed cache improves the response time by about 75% in case of overview pages and 95% in case of project info page. Data needed for the *main content* components (VLE graph, trends, students table) are not cached, because they change frequently and therefore it is necessary to load them every time the page is loaded. This results in the 20% difference between overview pages and project info page.
- **Gzip version** - *Gzip version* has also no difference in response times with empty or primed cache so measuring primed cache speed is irrelevant. However the saving compared to *new version* is about 60%. This is a result which confirms information from 2.8.5.
- **Final version** - Results obtained for the *final version* are a combination of the previously mentioned results for individual versions.

	Original	Header	Gzip	Final
Project info	+10% / -	+3% / -96%	-60% / -	-64% / -98%
Module overview	-7% / -	+1% / -73%	-62% / -	-64% / -78%
Student overview	-12% / -	+1% / -76%	-65% / -	-69% / -77%

Table 4 Average improvement of response time compared to *new version* of WA. Result with empty cache on the left, results of primed cache on the right after slash.

6 Conclusion and future plans

Many desktop applications have been lately implemented as web applications (WA). On one hand, they provide a number of benefits, like cross-platform compatibility or easy way to update the existing application. On the other hand, there are also many drawbacks, one of the biggest ones is slower response to user actions.

The WA introduced in this thesis is a part of OU Analyse (OUA) project that aims at predicting students' results in a course. The WA serves module teams and tutors to display these predictions. Because users access the application from a broad scale of device and browsers, the application should be compatible with a majority of them. However, it has been shown that the original vision does not work properly in IE8, even though using the browser is required by some users. The improvements introduced in section 4.2 have been implemented in the new version, therefore the new version supports all of the major browsers including the IE8. Another upgrade involves the rendering problems of the original application while accessing from small screen devices. The problem has been solved by applying the Bootstrap framework and Owl Carousel. The Bootstrap has also been used to create a uniform design across the web site.

Furthermore, the performance weaknesses has been detected by YSlow tool. Section 4.3 provides a detailed exposition of changes implemented in the new version. The YSlow tool graded B all pages of the web site of the new version, whereas the original version was graded B on the project info page and D on the overview pages.

Finally, it turns out that all pages of the new version are loaded faster. When cache is empty, it is by about 65%. When subsequent load is made, the project info page loads by about 98% faster and both overview pages approximately by about 75%.

To sum up, many improvements have been achieved in order to provide a more reliable and convenient application. However, there is still much that can be done.

Thanks to the growing computational possibilities of the mobile devices, a version of OUA application created specially for them could provide additional features compared to the original version, e.g. support for direct communication with students. Since the WA is not accessible for students, they could use the mobile version to obtain information about their results. Another improvement can involve the recommender component that displays various types of activities. If a filter of activity types was added, users would be able to search only the types they are interested in. The same feature can be added to the trend component. Moreover, users might appreciate saving filter settings to access them quickly in the future. Also, feedback from users could make the prediction more accurate. An extra component could be added for evaluation of the prediction by users. The OUA project has a potential to prevent students from failing the course and therefore helps the OU to increase student retention. A simple and well organized WA plays an important role in achieving the goal.

Appendix A

Project info page

Original	Basic	Header	Gzip	Final
2.55	2.25	2.28	1.02	0.78
2.48	1.92	1.96	1.05	0.87
2.78	2.28	2.31	1.12	0.89
2.01	2.43	2.59	0.75	0.83
2.88	2.40	2.36	0.91	0.85
2.92	2.51	2.46	0.77	0.82
2.03	2.36	2.47	1.11	0.88
2.56	2.13	2.34	0.81	0.75
3.00	2.30	2.74	0.79	0.78
2.02	2.47	2.22	0.81	0.73

Module overview page

Original	Basic	Header	Gzip	Final
2.96	2.38	4.00	1.17	1.59
4.30	3.73	4.73	1.30	1.34
3.52	3.76	3.65	1.15	1.34
3.30	4.06	3.42	1.30	1.31
3.51	3.56	3.64	1.25	1.08
3.48	3.90	3.93	2.61	1.05
3.52	4.01	4.13	2.34	1.34
3.49	3.99	3.82	0.97	1.41
3.54	4.26	3.75	1.19	1.26
3.46	4.19	2.91	1.12	1.75

Student overview page

Original	Basic	Header	Gzip	Final
4.77	4.56	3.55	1.03	1.36
3.64	3.90	3.54	1.33	1.21
3.01	4.80	3.52	1.20	1.11
2.90	3.55	4.62	2.16	0.98
2.88	3.99	3.59	1.49	1.11
3.11	3.95	3.87	1.33	1.14
2.83	3.01	3.39	1.05	1.23
3.06	2.44	3.76	1.26	0.97
3.08	3.53	3.61	1.01	1.16
3.19	3.32	3.63	1.15	1.13

Table 5 Response time results of various versions of OUA application with empty cache.

Project info page

Original	Basic	Header	Gzip	Final
2.52	2.22	0.09	0.99	0.04
2.42	1.86	0.08	1.02	0.05
2.75	2.23	0.08	1.09	0.05
1.98	2.39	0.08	0.72	0.05
2.85	2.13	0.09	0.88	0.04
2.90	2.32	0.14	0.74	0.06
1.98	2.33	0.08	1.08	0.04
2.53	2.00	0.08	0.78	0.04
2.97	2.27	0.08	0.76	0.05
1.99	2.42	0.08	0.78	0.06

Module overview page

Original	Basic	Header	Gzip	Final
2.96	2.34	1.12	1.13	1.00
4.30	3.69	1.01	1.26	0.82
3.52	3.72	0.99	1.11	0.94
3.30	4.02	1.00	1.26	0.89
3.51	3.52	0.96	1.21	0.74
3.48	3.86	1.00	2.57	0.89
3.52	3.97	1.01	2.30	0.69
3.49	3.95	0.94	0.93	0.65
3.54	4.22	1.20	1.15	0.78
3.46	4.15	0.89	1.08	0.65

Student overview page

Original	Basic	Header	Gzip	Final
4.77	4.52	0.88	0.99	0.76
3.64	3.86	0.87	1.29	0.94
3.01	4.76	0.78	1.16	0.79
2.90	3.51	1.23	2.12	0.89
2.88	3.95	0.88	1.45	0.87
3.11	3.91	0.79	1.29	0.86
2.83	2.97	0.88	1.01	0.85
3.06	2.40	0.84	1.22	0.86
3.08	3.49	0.85	0.97	0.80
3.19	3.28	0.81	1.11	0.83

Table 6 Response time results of various versions of OUA application with primed cache.

Bibliography

[1] Steve Souders. *High performance web sites: essential knowledge for frontend engineers*. Edition 1. O'Reilly, 2007. ISBN: 978-0-596-52930-7.

[2] J. Kuzilek. "OU Analyse: Analysing At-Risk Students at The Open University". In: *Learning Analytics Review* LAK15-1 (2015).

[3] *OU Analyse website*. URL: `www.analyse.kmi.open.ac.uk` (visited on 21/05/2015).

[4] Few Stephen. *Information dashboard design: the effective visual communication of data*. O'Reilly, 2006. ISBN: 0-596-10016-7.

[5] Josh Long. *I Don't Speak Your Language: Frontend vs. Backend*. URL: `http://blog.teamtreehouse.com/i-dont-speak-your-language-frontend-vs-backend` (visited on 08/04/2015).

[6] *Whatis.com - online dictionary of tech definitions*. URL: `http://searchsoa.techtarget.com/definition/Web-site` (visited on 08/04/2015).

[7] Ross Shannon. *What is HTML*. 2007. URL: `http://www.yourhtmlsource.com/starthere/whatishtml.html` (visited on 05/04/2015).

[8] *Free online dictionary of computing*. URL: `http://foldoc.org/declarative+language` (visited on 25/04/2015).

[9] *Document Object Model*. URL: `http://www.w3.org/DOM/` (visited on 08/04/2015).

[10] W3Schools. *CSS Pseudo-classes*. URL: `http://www.w3schools.com/css/css_pseudo_classes.asp` (visited on 05/05/2015).

[11] *The World Wide Web Consortium*. URL: `http://www.w3.org` (visited on 05/04/2015).

[12] *European Computer Manufacturers Association*. URL: `http://www.w3.org` (visited on 05/04/2015).

[13] *Can I use - List of technologies supported in main modern browsers*. URL: `http://caniuse.com` (visited on 05/04/2015).

[14] *StatCounter Global Stats- online visitor stats tool*. URL: `http://gs.statcounter.com` (visited on 05/04/2015).

[15] Network Working Group. *Hypertext Transfer Protocol – HTTP/1.1*. 1999. URL: `http://www.w3.org/Protocols/rfc2616/rfc2616.html` (visited on 05/04/2015).

[16] W3 Consorcium. *Header Field Definitions*. URL: `http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html` (visited on 19/05/2015).

[17] *JavaScript Object Notation*. URL: `http://json.org/` (visited on 08/04/2015).

[18] *The JSON Data Interchange Format*. URL: `http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf` (visited on 08/04/2015).

[19] *JavaScript Web APIs*. URL: `http://www.w3.org/standards/webdesign/script` (visited on 10/04/2015).

[20] *Extensible Markup Language*. URL: `http://www.w3.org/TR/xml` (visited on 10/04/2015).

[21] W3Schools. *JavaScript Libraries.* URL: http://www.w3schools.com/js/js_libraries.asp (visited on 05/05/2015).

[22] W3 Techs. *Usage of JavaScript libraries for websites.* URL: http://w3techs.com/technologies/overview/javascript_library/all (visited on 19/05/2015).

[23] jQuery. *jQuery list of utility functions.* URL: http://api.jquery.com/category/utilities/ (visited on 19/05/2015).

[24] *Browserscope - project for profiling web browsers.* URL: http://www.browserscope.org/?category=network (visited on 19/04/2015).

[25] *GNU Gzip.* URL: https://www.gnu.org/software/gzip/ (visited on 17/04/2015).

[26] Erik Nygren, Ramesh K. Sitaraman, and Jennifer Sun. "The Akamai network". In: *ACM SIGOPS Operating Systems Review* vol. 44.issue 3 (2010-08-17), pp. 2–. ISSN: 01635980. URL: http://portal.acm.org/citation.cfm?doid=1842733.1842736.

[27] *Beginning responsive web design with html5 and css3.* Berkeley: Apress, 2014. ISBN: 978-143-0266-945.

[28] W3 Schools. *CSS media rule.* URL: http://www.w3schools.com/cssref/css3_pr_mediaquery.asp (visited on 19/05/2015).

[29] T. Cover and P. Hart. "Nearest neighbor pattern classification". In: *IEEE Transactions on Information Theory* vol. 13.issue 1 (1967), pp. 21–27. ISSN: 00189448. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1053964.

[30] Styleshout. *Free website templates.* URL: http://www.styleshout.com/ (visited on 19/05/2015).

[31] YSlow. *YSlow the web page performance analyser.* URL: http://yslow.org/ (visited on 19/05/2015).

[32] *Free Awesome Website Templates.* URL: http://www.styleshout.com (visited on 19/04/2015).

[33] usabli.ca. *A Collection of front end frameworks for web development.* URL: http://usablica.github.io/front-end-frameworks/compare.html (visited on 19/05/2015).

[34] Otto M. and Thornton J. *HTML, CSS and JS framework for developing responsive, mobile first projects on the web.* URL: http://getbootstrap.com/ (visited on 19/05/2015).

[35] Apache Tomcat. *Apache Tomcat^{TM} - an open source software implementation of the Java Servlet and JavaServer Pages technologies.* URL: https://tomcat.apache.org/index.html (visited on 19/05/2015).

[36] Viral Patel. *Enable GZIP compression in Apache Tomcat.* URL: http://viralpatel.net/blogs/enable-gzip-compression-in-tomcat/ (visited on 19/05/2015).

[37] Kirkman R. and Davis T. *Free CDN servers.* URL: https://cdnjs.com/ (visited on 19/05/2015).

[38] Oracle. *JavaServer Faces Technology.* URL: http://www.oracle.com/technetwork/java/javaee/jsp/index.html (visited on 19/05/2015).

[39] Patrick Th. Eugster et al. "The many faces of publish/subscribe". In: *ACM Computing Surveys* vol. 35.issue 2 (2003-06-01), pp. 114–131. ISSN: 03600300. URL: http://portal.acm.org/citation.cfm?doid=857076.857078.