***Content***

1. ***Background***
2. ***Motivation***
3. ***Goals and Application Specification***
4. ***Choices of Development Tools and Programming Languages***
5. ***Hyper Text Mark-up Language (HTML) – Front-end***
6. ***Python and Flask – Back-end***
7. ***JavaScript and JQuery***
8. ***Google Maps API***
9. ***CSS***
10. ***CSS and Plugins***
11. ***MySQL and MySQL Workbench***
12. ***Android Studio and Java***
13. ***XCode and Swift***
14. ***Database Implementation***
15. ***Description***
16. ***Triggers***
17. ***Encryption of crucial data***
18. ***Geographic Database and Well-Known Text***
19. ***Improvement of MySQL relational***
20. ***Other comments about the database***
21. ***Code Structure***
22. ***Server-Client, Flask Framework and Jinja***
23. ***Server File structure and Back-end system***
24. ***JavaScript/JQuery – Front-end***
25. ***Mobile Application compatible to Android and iOS operating systems***
26. ***What is a web frame application?***
27. ***Advantages and disadvantages of web frame application***
28. ***Code Structure***
29. ***Responsive Design***
30. ***Security***
31. ***Unauthorized handling and Error notification***
32. ***Steps taken to secure the application***
33. ***Testing***
34. ***Feedbacks***
35. ***How feedbacks are gathered?***
36. ***Some interesting feedbacks we got***
37. ***Response to feedbacks***
38. ***Summary***
39. ***Background***

This project is a collaboration between the computer department and the city planning department at the university of Stavanger.

The idea behind this project is to develop a digital tablet application that enables interviewers to gather data about how people use and perceive the city’s facilities and other public and private areas. The data gathered is attended to be used by the city planning department for their academic researching to help improving the city by identifying certain problems about some facilities or places.

1. ***Motivation***

The city of Stavanger is experiencing a significant growth in the number of residents and visitors just like many cities around the world. According to the Statistical Central Agency of Norway SSB the number of residents is expected to grow up to 141 634 residents in 2030, that around 6,4 % increase of today’s population which lays at 133 139 residents.

Because of this expected growth in population, city planning and developing is now very important to help understand how our cities function and what can we do to improve the different facilities of the city to be prepared at the best way possible for the future.

1. ***Goals and Application Specification***

The application we will develop must meet the need of the city’s interviewers and analysts, this means that the application should be easy to use by the interviewer and also analysts need to get access to a flexible and reliable data gathered by the interviewers.

The interviewer at the field will have the ability to click on a digital bounded geographic map then draw a point, road/path or even an area. The form of paths and areas does not need to be at a certain recognized geometric form like a circle or a rectangle, instead the interviewer must have the ability to draw paths and areas at the map in a freely and not restricted to a certain type of geometric form.

The administrator(s) is the one who creates the map and assign it to one or more interviewers, after the interviewing is finished the one of the administrator can view the map to check if there is some mistakes in the data gathered, then he can download a CSV file containing the map and shapes details, this file can later be used in other programs just like ArcGIS.

The Interviewer will have the ability to view, delete and modify the shapes he/she has committed on the map, but they will not have access to what other interviewers has committed of shapes and off course they are not able to change any of those shapes.

The administrator(s) on the other hand will be able to view, delete and modify any shapes entered by anybody. In addition to this administrators can also add new shapes if needed.

Administrators are authorized to download the CSV file, something an interviewer cannot do.

The application should also offer a serious of functions that is attending to make analysing data under and after interviewing easier. Examples of such functions are as following:

1. filtering shapes by the creator of the shape: this function is only

allowed for administrators, since they are the only ones who have access to see all other shapes.

1. Adjusting the thickness of the paths on the map from 0 to 100%: this function make it easier to the interviewer to identify and distinguish the shapes entered earlier at the interview, and it also can be used by an administrator to better understand the shapes on the map
2. Adjusting the opacity of the areas: This function is meant to vary the opacity of the areas from 0%(not visible at all) to 100%(no transparency at all), this is for the same reason mentioned in part ii
3. Clustering: the word clustering here means grouping a set of shape objects in such a way that shapes close to each other are in the same group and will be displayed as one marker on the map, showing only the number of objects clustered in this group. Clustering is only implemented for point shapes this means paths and areas are not included or affected in this function. Additional clustering for paths and areas might be implemented later on

Clustering can be enabled and disabled in the moment locally at the user device after the need of the interviewer and the administrator.

1. Exporting the map: An administrator of the map will be able using these function to download a CSV file including all shapes entered by any interviewer or administrator and also containing the map details like the geographical bounds of this map
2. Additional functions might be added later
3. ***Choices of Development Tools and Programming Languages***

Our decision about what to use of tools and programming languages was built on what we are most experienced with and the needs of the department of City Planning at the University of Stavanger, they needed a web application that is easy to access from anywhere using the internet and can be compatible with touch tablet devices which is often used in interviewing, and also possible to integrate into a dedicated application for the Android and IOS operating systems.

We choose the following programming languages and tools:

1. ***Hyper Text Mark-up Language (HTML) – Front-end***

Html is the language used to format the content of websites containing hypertext and other type of information, the goal behind this language is to enable the net browser to view the appreciated content sent to the client from the server.

1. ***Python and Flask – Back-end:***

Python is a high-level programming language for general purpose programming; it offers better code readability and improved syntax that allows the programmer to express concepts in fewer lines of code. Python also has central features like automatic memory management, big comprehensive standard library that makes programming much easier.

Python offers two main types of frameworks when designing a back-end for a websites; the first one is Flask and the other one is Django. Flask is considered a light weight version of Django that was realised after Django in order to compress and reduce the source lines of code and improve the file system of the project which was earlier user in Django framework.

According to what we know about our application and what we were most experienced with, we decided to go with Python and use the Flask framework.

1. ***JavaScript and JQuery***

JavaScript is a high-level language which runs on the client device, JS is often used to make webpages interactive and more flexible, this is done because of availability to event-driven programming functions and triggers in this language.

But just like Java, JavaScript also force the programmer to write longer code, for this purpose an open source JavaScript Library was developed and realised in 2006 to improve the syntax of JavaScript and make navigation of the html document easier to manage. This is because jQuery library offers a lot of functions that gives the user many additional advantages like for example animation.

In this project we use Ajax, which enable sending a request to the server and getting the response without the need of refreshing the website. Ajax can be used by clean JavaScript Syntax or using the JQuery library. We prefer to use jQuery because the code will then be much easier to understand and more compressed.

1. ***Google Maps API***

Google Maps API is available for a lot of languages to implement, easy to use and will documented by the author of this API.

Google Maps offers a lot of information about almost anywhere in the world; it enables the user to see roads, houses, forests and many other geographic places in a recognizable way to the average user.

Alongside with these advantages, google maps API has a huge database that backs up the geocoding search for places on the map.

But google maps are not perfect, for the following reasons:

1. In order to view the houses and buildings on the map, the user needs to zoom to a certain level close enough, the user can avoid this problem by switching to satellite view om the map, but this comes with a cost of more relatively high use of internet bandwidth and can slow the website will loading the satellite image from the google maps database.
2. Although google maps is an open source API, google today charges 0.50USD for every 1000 requests that exceeds the free use limit of 25000 USD. The same pricing is also applied to the use of google web services like the GeoLocation and Geocoding API

We find it tempting to have the opportunity to 25000 requests a day for the map, since this limit is unlikely to be exceeded at the beginning of this project. Also since google maps API is the most used maps API in websites, this makes finding information about how to program using the interface easier and more reliable.

1. ***CSS:***

CSS stands for Cascading Style Sheets which is a programming language used to define the positioning, colours and other styling options of the elements in html- or xml-documents. CSS plays a role in this project both when it comes to styling the elements and when making the content of the html documents responsive so it adjust the elements to get customized according to the available window size on the user device.

1. ***CSS and JavaScript Plugins:***

A plugin is simply a software component that adds a specific feature to an existing program. Plugins is useful because they can save a lot of hard work when trying to build a widely used feature element that html does not include, an example of such plugin is the colour picker which gives the user the opportunity to pick a colour while defining a map area or path category. Plugins in web development often consists of a combination of JavaScript and CSS files which can be either downloaded or imported using the <script> tag.

At this project three types of plugins are used:

1. Bootstrap: is a huge library of HTML-, CSS- and JavaScript-based design templates developed to meet the need of modern websites. It offers a responsive and fine styling of HTML-elements saving the programmer the time and struggle to find the right design.

But bootstrap makes the website looks just like every other website using bootstrap in which there exist a lot of websites using this plugin. In addition to that bootstrap c lass names can sometimes collide with own defined class names which can result in an expected behaviour of the web design.

1. Icon Picker plugin:

This icon picker is most likely to change later on, so we save the explanation to the final version

1. Colour Picker plugin:

A HTML elements that helps the user to pick a certain colour is actually reliable to work on all browsers and devices especially when it comes to mobile devices, that why we decided to download a special costumed plugin design and use it for this purpose on our website.

The plugin is created by Mr. Brian Grinstead and it is downloaded from this website: <https://bgrins.github.io/spectrum/#why>

1. Other plugins might come later on
2. ***MySQL and MySQL Workbench***

At this project we use a MySQL relational database to save and retrieve data in a reliable and effective way.

MySQL runs on many operating systems, and offers high automatic optimizations for the queries sent to the database. MySQL also offers a user friendly graphic interface that enables managing the database both locally and remotely using the reliable TCP connection.

Since out application is expected to contain data and shapes relative to each map, we decided to go on and use a relational database in order to easily manage and manipulate data.

1. ***Android Studio and Java***

Android Studio is an integrated development tool used to develop applications on the Android operating system. It uses Java as a programming language and implements a good features to test apps both virtually using an emulator and realistically using an Android device connected to the workstation by a USB cable. Android Studio is available to use on both windows and IOS operating systems.

The tool is easy to manage and use, and none of the less it offers a good detailed error massages in case a crash should occur under developing and testing.

Android Studio and Java is used in this project to create a dedicated tablet and mobile application that implements our website as a web frame and runs on Android operating system.

1. ***XCode and Swift***

XCode is development tool used to produce applications compatible with Apples operating systems iOS and OS X. Using XCode requires knowledge about the SWIFT object oriented language; SWIFT has the languages C, Objective-C, Cocoa and Cocoa Touch framework.

XCode is free to download and use only on any device running the apples operating system iOS and OS X. but its also possible to emulate such operating system on devices that runs another operating system like Windows for example.

XCode and SWIFT is used in this project to create a dedicated tablet and mobile application that implements our website as a web frame and runs on iOS and OS X operating systems.

Although developing application using both Android Studio is free and does not require additional costs, publishing work is not. So the developer has to pay some amount of money in order to make the application accessible for the public, but this cost is currently and expected to be minimal and affordable.

1. ***Database Implementation:***
2. ***Description:***

* A person has a username, login password, first name, last name, e-post address and a telephone number. The person can be either an interviewer or an administrator in association with a certain map. But a person will be able to an interviewer in one map and an administrator in another.
* The administrator is able to create a map and also define some categories to follow with that map, along with the map the administrator can add other persons to be interviewer or to join him/her to in administrating the map.
* Editing the map which includes changes, adding, and removing categories, or changing descriptions or title of some map is restricted to the administrators.
* A map has a map id, map creator, title, description, issue date, expiry date, geographical boundary and zoom level. A map can contain some categories already defined by an administrator. A map can have as many users and administrators as desired.
* Every administrator have the opportunity to define new categories, a category entry have the map ID for the map its assigned to, an ID, a name, a type(Point, Area, Path)
* An administrator and interviewer can add shapes to the map, a shape entry will store the shape ID, category ID, shape creator, the center of the shape and a series of coordinates if the shape is an area or path. If the shape is an area or path the center will be the middle point in a path and the centroid in an area.



1. ***Triggers***

Coming later on

1. ***Encryption of crucial data***

Coming later on

1. ***Geographic Database and Well-Known Text***

MySQL offers the opportunity to store geographic data by using “Well-known Text” which is a mark-up language for representing vector geometry object on a map. The library of MySQL also offers a series of functions that makes is easier to combine and compare different geometry object with each other. A simple example of this is when we need to know if a certain point object is inside a polygon (area) object, then functions like “MBRContains()” helps in determining this issue.

1. ***Improvement of MySQL relational database***

Coming later on

1. ***Other comments about the database***

Coming later on

1. ***Code Structure***
2. ***Server-Client, Flask framework and Jinja***

A web application is based on a server-client model. Any user around the world can access the web application via a global internet interface, simply a web browser sends an HTTP or HTTPs request to the server with the address typed in the URL field in the browser, the browser then responds to the client with an HTTP or HTTPs response to the client browser.

A browser performs the task to process the html, css and javascript code received as a plain text in HTTP or in encrypted format HTTPs, the browser then displays the graphic interface to the user.

In order to build a web application we need to have a Back-end which represents the server and a Front-end structure that represents what the client gets as a response.

The Back-end language used to run the server is python, and the database is placed on the server and using MySQL syntax. A flask framework gives as the opportunity to use Jinja syntax to form an html page in a desired way for each client request, an example of this is viewing the privileged step to download a CSV file including the format of the map and shapes on it. This step is restricted to the administrators of the map and the export button on the html element should only be viewed if the user is registered as an administrator, to do so the following code is included in edit\_map.html file.

{% **if map**.**creater** == **map**.**logged\_in\_user** %}  
  
 <**div class="panel panel-info"** >  
  
 <**form action="/exportMap?mapid=**{{ **map**.**mapid** }}**'" method="post"**>  
 <**input id="submit" class="panel panel-info" type="submit" value="EKSPORT"**>  
 </**form**>  
 </**div**>  
  
 {% **endif** %}

1. ***Server File Structure and Back-end system***

A Flask framework restrict the user to have a file named “templates” that includes all the html pages and jinja syntax can be used directly between the html lines of these file like the example above. Another file is named “static” and this file can include customized directories and file structure to meet the need of the server administrator/programmer, the static file can have all files that do not need to change when it is being to the client and that is way Jinja syntax cannot be used here, an example of such files is JavaScript files (.js), CSS Stylesheets (.css), images, icons and plugins etc.…

Since our website design requires a head navigation bar on the top of each page, we decided to make a index.html page that includes the navigation bar and three main Jinja sections/blocks. The bone structure looks like this:

{% **block stylingblock** %}

{% **endblock** %}  
  
{% **block scriptblock** %}

{% **endblock** %}

<**header**>

<**nav id="nav\_bar" class="navbar navbar-inverse"**>

…

…

</**nav**>  
</**header**>  
  
{% **block content** %}

{% **endblock** %}

Any html page can extends or inherits from index.html using the {% extends ‘index.html’ %} and then adds what its needed of files and html content for this html page at this different blocks.

Our file structure on the server is based on the idea of having one dedicated stylesheet file and another one for JavaScript code if needed to each html page in the templates file.

The files can be imported at the header using the “stylingblock” and “scriptblock” after inheriting the bone structure from index.html. an example is taken from new\_map.html

{% **block stylingblock** %}

/\* importing stylesheet files \*/

*<!-- Link to static/stylesheets/new\_map\_stylesheet.css -->* <**link rel="stylesheet" type="text/css" href="**{{ **url\_for**(**'static'**,**filename**=**'Stylesheets/new\_map\_stylesheet.css'**) }}**"**>  
{% **endblock** %}  
  
{% **block scriptblock** %}  
 */\* importing javascripts file \*/*  
  
 *<!-- link to static/Scripts/new\_map\_JavaScript.js -->* <**script src="**{{ **url\_for**(**'static'**,**filename**=**'Scripts/new\_map\_JavaScript.js'**) }}**"**></**script**>  
  
{% **endblock** %}  
  
  
{% **block content** %}

<!-- html content -->

…

{% **endblock** %}

Running the CityShare.py on the server make the server able to listen to HTTP requests sent from clients through the internet.

CityShare.py includes different methods that handle the different URLs requested, following is a brief description on what each method and URL does return:

|  |  |
| --- | --- |
| Method | Purpose |
| **def** get\_db(): | Establishes a connection to the MySQL database |
| @app.route(**'/'**) **def** index(): | Returns the home page of the website |
| @app.route(**'/signup'**, methods=[**'POST'**,**'GET'**]) **def** signup(): | Handles the signing up of new users |
| @app.route(**'/signout'**) **def** signout() | Signs a user out by deleting the session of this user |
| @app.route(**'/login'**, methods=[**'POST'**,**'GET'**]) **def** login(): | Logs a user |
| @app.route(**"/searchUsers"**, methods=[**'POST'**]) **def** search\_for\_users(): | Search for a specific user by username or e-mail and return the results in json format |
| @app.route(**'/addnewmap'**, methods=[**'POST'**,**'GET'**]) **def** add\_new\_map(): | Adds a new map to database |
| @app.route(**"/showmaps"**, methods=[**'GET'**]) **def** show\_maps(): | Shows the maps the user have access to |
| @app.route(**"/EditMap"**, methods=[**'GET'**]) **def** edit\_map(): | Opens the editor of a map, allowing the interviewer/administrator to add/update/delete new shapes, and many other features |
| @app.route(**"/registerShape"**, methods=[**'POST'**]) **def** registerShape(): | Registers a new shape in database |
| @app.route(**"/updateShape"**, methods=[**'POST'**]) **def** updateShape(): | Updates a shape existing in database |
| @app.route(**"/deleteShape"**, methods=[**'POST'**]) **def** deleteShape(): | Deletes an existing shape from database |
| @app.route(**"/exportMap"**, methods=[**'POST'**, **'GET'**]) **def** exportMap(): | Downloads a CSV file containing the boundary of the map and the shapes entered on the map, the file is meant to be uploaded on ArcGis later on. |
| More methods might come later |  |

1. ***JavaScript/JQuery And Front-End***

* Coming later on

1. ***Mobile Application compatible to Android and iOS operating systems***
2. ***What is a web frame Application?***

A web frame application is an application that displays a dedicated website through the application window. A user of this application does not have to rely on a web browser to access the website and instead the user can use the application to directly access the website and does not have to type the address of the website any more.

1. ***Advantages and disadvantages of a web frame application.***

Web frame applications is easy to implement and test, and can be more secure than a normal designed application, this because accessing the database on the server side is done on the server, and the user communicates with the server through http or https requests and responses.

Accessing the database via a non-web frame application requires a connection establishment from the application using the crucial parameters for the database like a password and username. With reverse engineering of the application a programmer can get access to the source code of this application and then view these database parameters. This can lead to theft , manipulation of data stored in the database, or even a denial of service.

On the other hand, a web frame application leads to increase the number of http requests sent to the server. A server can handle a certain number of requests at the same time; a high number of requests to the server might put the service down due to high demand on the service.

A dedicated application that does not use a web frame can be able to communicate directly with database without the need of using http requests to the server.

A better solution is actually a combination of both solutions; a dedicated application with a specific design can be implemented to do the part that does need retrieved data from the database and for the need of these retrieved data from database an http request can be sent to the server to get a response with data in json format or any other desired format.

The last mentioned solution requires more work to build the application, and since our application is meant to be use by a small amount of users, we decided to go with the web frame solution to create to applications the one is build using XCode and Swift for iOS and the other is for Android devices using Android Studio and Java.

1. ***Code Structure.***

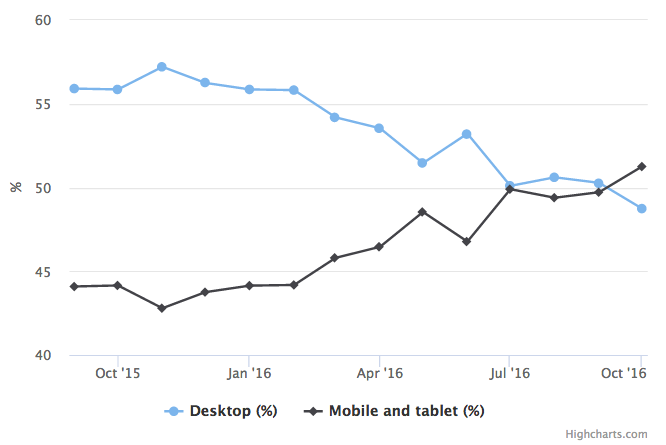
Coming later..

1. ***Responsive Design:***

Responsive Design of web application positions the content of a web page relative to the available window size on the user’s device. This step improves the way content is displayed and makes interacting with the graphic interface much easier.

The availability and spread use of smart mobile devices today, requires designing a responsive application that automatically adapts the content according to the available screen size on different devices with different viewports.

According to statistics the use of desktop devices is actually going down, and using tablets and mobile devices is on the raising and it is expected to keep rising.



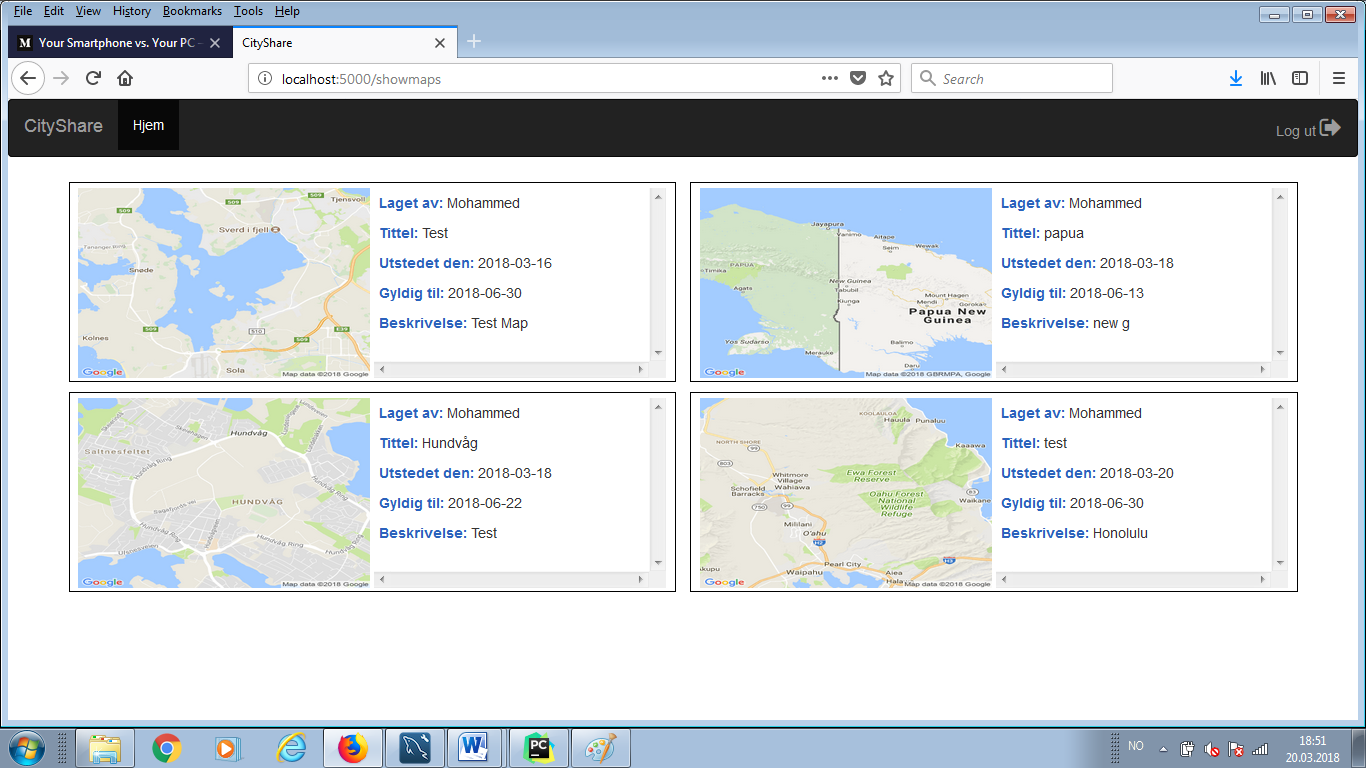
To create a responsive application a programmer can use CSS, JavaScript/JQuery or a combination of both. Using one of these alone does not satisfy our needs to create a readable and compressed code that effectively implements the responsive function. So in this application we used clean CSS on some HTML pages and a JQuery on others.

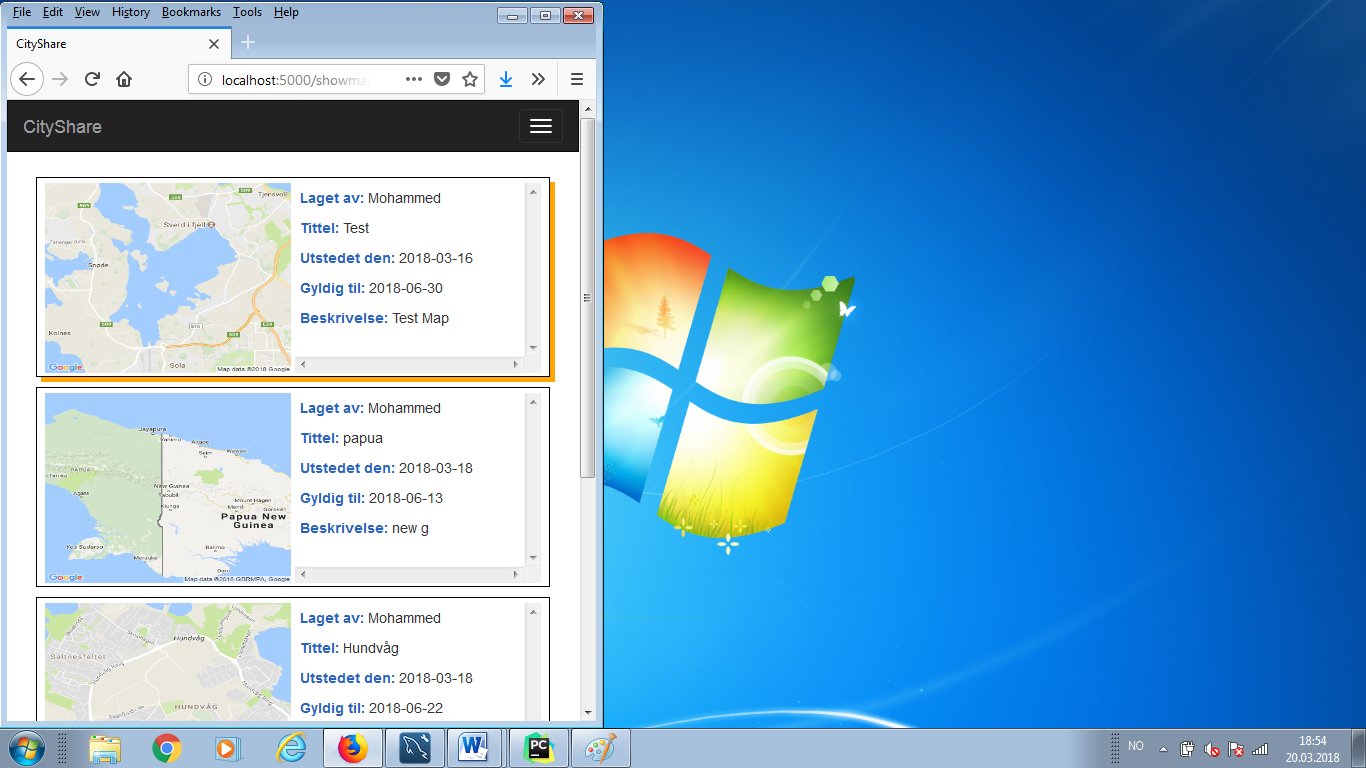
Pages like home, index.html, Signup.html and view\_maps.html is using clean CSS code to fix this issue, following is an example of how CSS is used to implements the responsive function on the page view\_maps.html

.**map\_box** {  
 **display**: **inline-block**;  
 **width**: 45%;  
 **margin**: 5**px**;  
 **padding**: 5**px**;  
 **height**: 200**px**;  
 **border**: 1**px solid black**;  
}

**@media only screen and** (**max-width**: 800**px**) {  
 .**map\_box** {  
 **width**: 90%;  
 }  
}

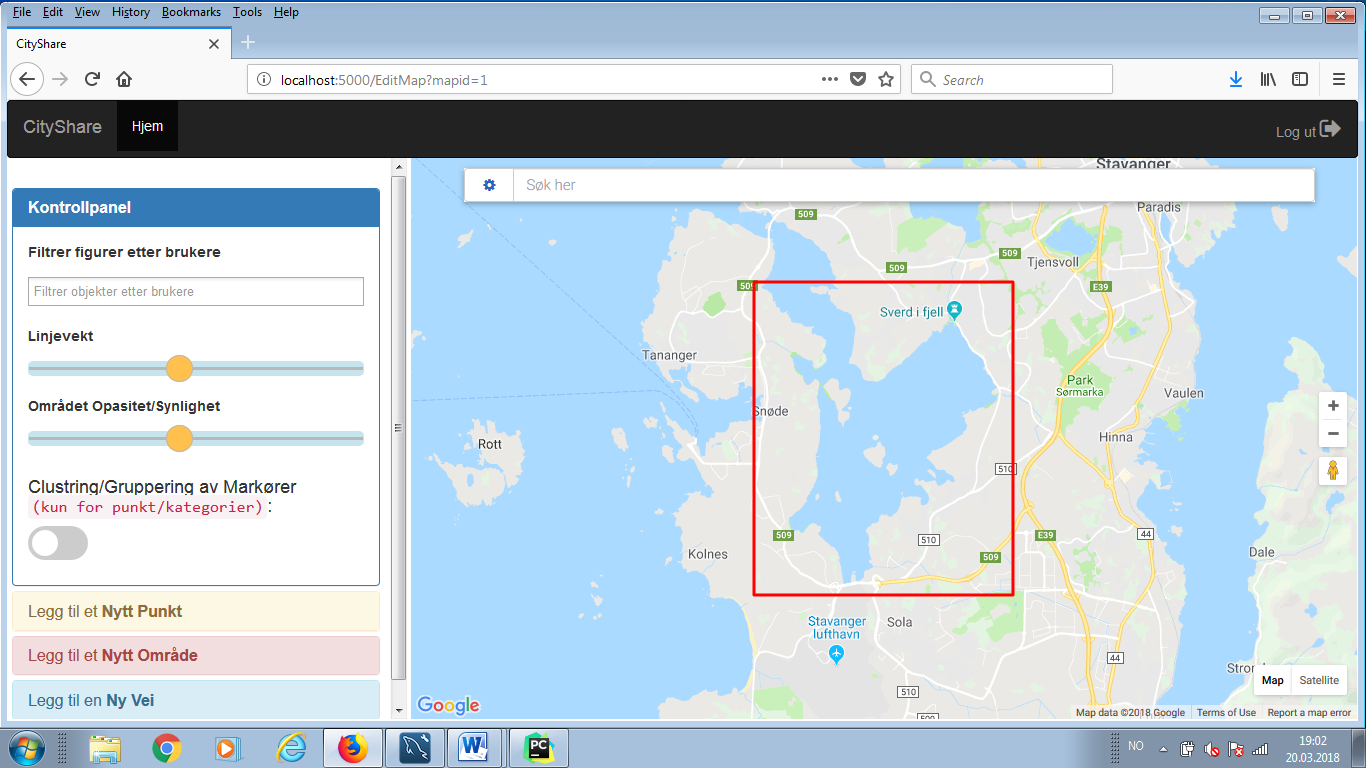
The past example shows how to change the width of the boxes inside the view\_maps.html from 45% to 90% on devices with window width size smaller than 801px. This makes the boxes show one at each row instead of showing two beside each other.



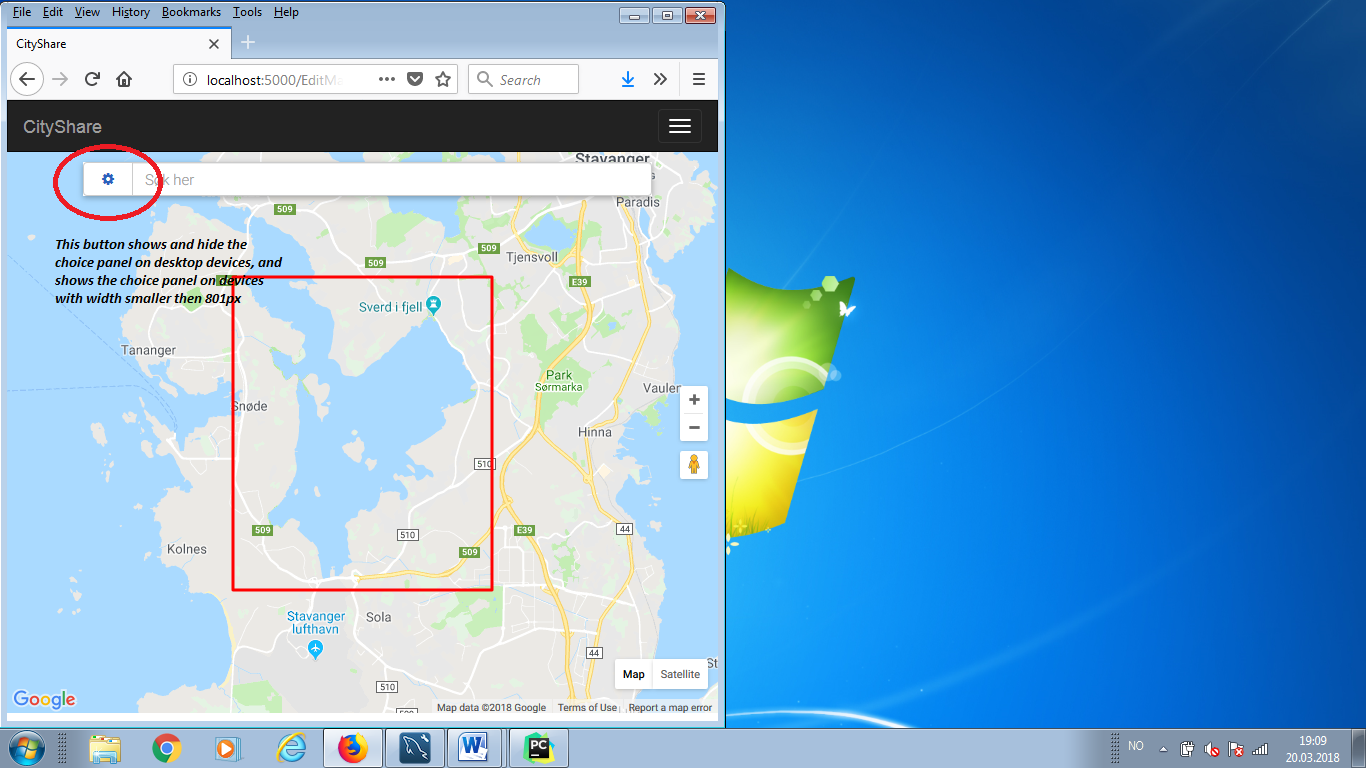


In addition to this functionality we also needed some JQuery functions that does two jobs, the first one is to modify the CSS of the elements to make it more responsive and the other is to show and hide buttons and panels so the user will always have a clear view on the map and by clicking on the setting button, the choice panel will then be showed.

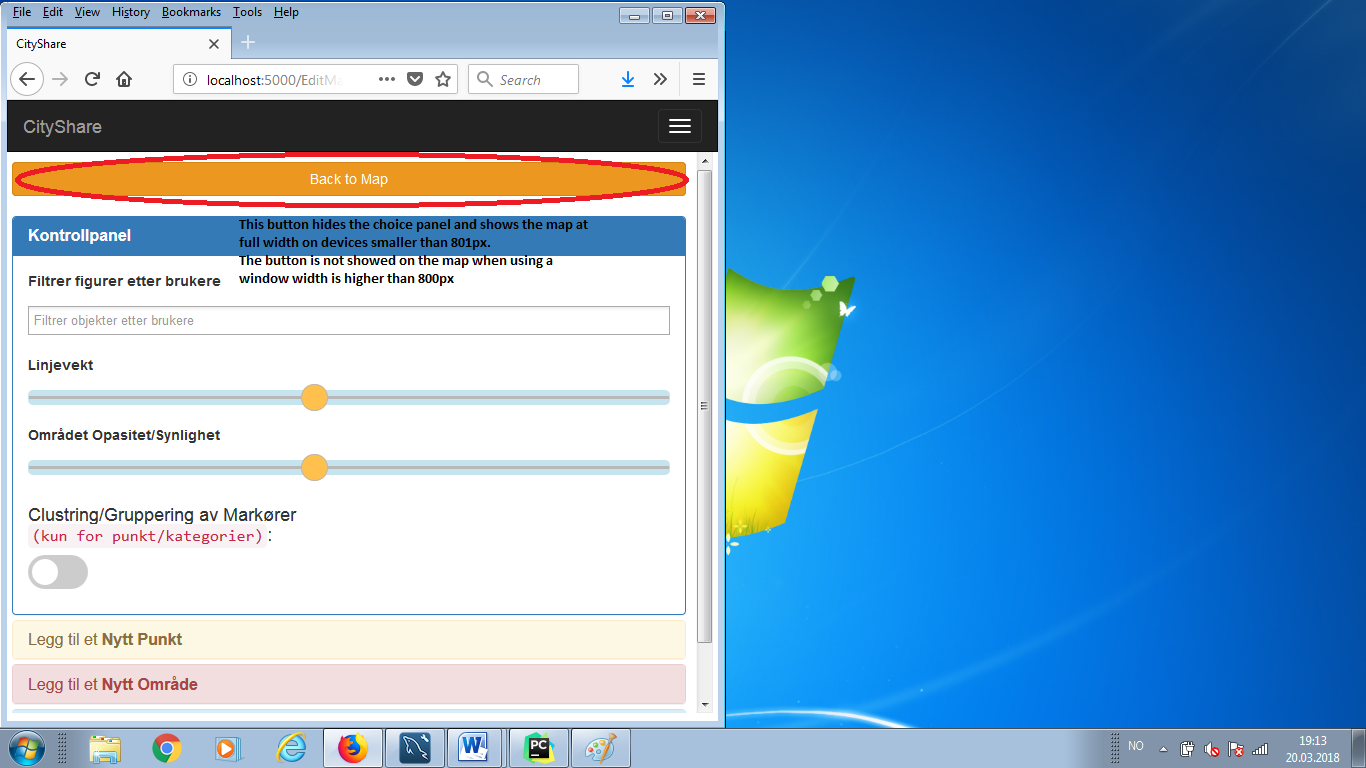
In the following figure we see the desktop view of the application.



The next figure shows that the choice panel gets hidden, when the window is resized to a smaller width



Next figure shows what happens when we click on the show choice panel beside the address search bar on the map. Then the map is shown with an orange button “Back to Map” that hide the choice panel after selection and shows the map viewed in the previous figure.



The code used to implement this functions is:

*// code for adjusting dimentions on big and small screen devices***var** choice\_panel\_toggled;  
**function** *auto\_adjust\_dimentions*() {  
 **window\_width** = $(***window***).width();  
 **if** (**window\_width** > 800) {  
 $(**"#Map"**).css({**"width"**:**"70%"**,**"display"**:**"inline-block"**});  
 $(**"#ChoicePanel"**).css({**"width"**:**"30%"**,**"display"**:**"inline-block"**});  
 $(**"#pac-card"**).css(**"width"**,**"90%"**);  
 $(**".btn-success, .btn-danger"**).css({**"width"**:**"49%"**,**"display"**:**"inline"**});  
 $(**"#show\_map\_btn"**).css(**"display"**,**"none"**);  
 choice\_panel\_toggled = **false**;  
 } **else** {  
 $(**"#Map"**).css({**"width"**:**"100%"**,**"display"**:**"inline-block"**});  
 $(**"#ChoicePanel"**).css(**"width"**,**"100%"**);  
 $(**"#ChoicePanel"**).*hide*();  
 $(**"#pac-card"**).css(**"width"**,**"80%"**);  
 $(**".btn-success, .btn-danger, #show\_map\_btn"**).css({**"width"**:**"32%"**,**"display"**:**"inline"**});  
 choice\_panel\_toggled = **true**;  
 }  
}  
*auto\_adjust\_dimentions*();  
$(***window***).resize(**function** () {  
 *auto\_adjust\_dimentions*();  
});  
  
$(**"#toggle\_panel"**).click(**function** () {  
 **window\_width** = $(***window***).width();  
 **if** (**window\_width** > 800) {  
 **if**(choice\_panel\_toggled) {  
 $(**"#Map"**).css({**"width"**:**"70%"**,**"display"**:**"inline-block"**});  
 $(**"#ChoicePanel"**).toggle();  
 } **else** {  
 $(**"#Map"**).css({**"width"**:**"100%"**,**"display"**:**"inline-block"**});  
 $(**"#ChoicePanel"**).toggle();  
 }  
 } **else** {  
 $(**"#ChoicePanel"**).toggle();  
 $(**"#Map"**).toggle();  
 }  
  
 **if**(choice\_panel\_toggled) {  
 choice\_panel\_toggled = **false**;  
 } **else** {  
 choice\_panel\_toggled = **true**;  
 }  
});  
  
$(**"#show\_map\_btn"**).click(**function** () {  
 *auto\_adjust\_dimentions*();  
});

This code is found in both edit\_map\_JavaScript.js and new\_map\_JavaScript.js. As we see the code is identical for both pages because both pages uses the same main graphic organization with the same ID’s for main common elements like **#ChoicePanel** and **#Map**.

1. ***Security***
2. ***Unauthorized handling and Error notification***

***Coming later …***

1. ***Steps taken to secure the application***

***Coming later …***

1. ***Testing***

***Coming later …***

1. ***Feedbacks***
2. ***How feedbacks are gathered?***

***Coming later …***

1. ***Some interesting feedbacks we got***

***Coming later …***

1. ***Response to feedbacks***
2. ***Coming later …***
3. ***Summary***

***Coming later …***