

Czech University of Life Science Prague

Faculty of Economics and Management

Department of Information Technologies



DIPLOMA THESIS

**Analysis and design of web application based on open
data**

Author: Damir Pirija

Supervisor: Ing. Miloš Ulman, Ph.D.

© 2015 Prague

CZECH UNIVERSITY OF LIFE SCIENCES PRAGUE

Department of Information Technologies

Faculty of Economics and Management

DIPLOMA THESIS ASSIGNMENT

Damir Pirija

Informatics

Thesis title

Analysis and design of web application based on open data

Objectives of thesis

The main aim of the thesis is to develop a web application based on open data.

Partial aims of the thesis are such as these:

- to explain the concept of open data, it's purpose ,
- to provide overview of web application development life cycle which includes analysis , design and implementation stages,
- to develop a web application with use of PHP framework which will make use of open data, dataset, and
- to present valuable information and conclusions from the dataset.

Methodology

First part will be focused on theory behind open data, it's use, and different models of open data. Technologies and methods will also be discussed. Development life cycle will be described and explained. Second part will consist of actual design and development of a web application which will use an open data dataset which is related to crime. Information about crimes, such as category, date, location, outcome will be loaded into the database from the dataset. Then, web application will present that data in a way that it can be easily understood and interpreted by a user, from which he or she can draw conclusions. The web application will be developed in modern approach with the use of modern web technologies, geolocation and Google Maps API. Based on the literature review and results of the practical part, conclusions will be formulated.

The proposed extent of the thesis

60-80 pages

Keywords

Open Data, PHP, PHP framework, Data set, Crime, Database, SQL, Web application, Analysis, Design, Open Data application

Recommended information sources

FRAIN, Ben. Responsive web design with HTML5 and CSS3: learn responsive design using HTML5 a CSS3 to adapt websites to any browser or screen size. Birmingham: Pack Publishing, c2012, vi, 305 s. ISBN 978-1-84969-318-9.

GURIN, Joel. Open data now: the secret to hot startups, smart investing, savvy marketing, and fast innovation. xvi, 330 pages. ISBN 00-718-2977-6.

HEEKS, Richard. Implementing and managing eGovernment: an international text. Thousand Oaks, Calif.: SAGE, 2006, ix, 293 p. ISBN 07-619-6792-3.

Janssen, M.F.W.H.A. . Charalabidis, Y. . Zuiderwijk, A.M.G. , Benefits, Adoption Barriers and Myths of Open Data and Open Government. Taylor & Francis, 2012. ISBN 1058-0530

SOMMERVILLE, Ian. Software engineering. 9th ed. Boston: Pearson, c2011, xv, 773 p. ISBN 978-013-7053-469.

TATROE, Kevin, Rasmus LERDORF a Peter MACINTYRE. Programming PHP. 3rd ed. Sebastopol, CA: O'Reilly Media, 2013, xxii, 514 p. ISBN 14-493-9277-6.

Expected date of thesis defence

2015/06 (June)

The Diploma Thesis Supervisor

Ing. Miloš Ulman, Ph.D.

Electronic approval: 10. 3. 2015

Ing. Jiří Vaněk, Ph.D.

Head of department

Electronic approval: 11. 3. 2015

Ing. Martin Pelikán, Ph.D.

Dean

Prague on 24. 03. 2015

DECLARATION

I declare that I have worked on my diploma thesis titled "Analysis and design of web application based on open data" by myself and I have used only the sources mentioned at the end of the thesis. As the author of the diploma thesis, I declare that the thesis does not break copyrights of any third person.

In Prague on 27.03.2015

Damir Pirija

ACKNOWLEDGEMENTS

First of all, I would like to thank my supervisor Ing. Miloš Ulman Ph.D. for advices and assistance during the work on this diploma thesis and all his support during whole practical work.

Thanks to all academic staff of Czech University of Life Sciences for their contribution to obtaining high-quality skills and knowledge. Most of all, I would like to thank my family for supporting me and my decisions through my studies in Czech Republic.

SOUHRN

Trendy ve vývoji webových aplikací se mění téměř každý rok, je proto potřeba je sledovat a řídit se správnými postupy v oblasti navrhování a vývoje webových aplikací. Jedině tak může být vyvinut produkt, jež bude široce využíván a bude schopen předkládat takové výsledky a závěry, které se od něj očekávají. Každý informační systém včetně webových aplikací by měl být vyvinut za použití následujícího životního cyklu informačního systému. Ten nás provádí celým procesem od chvíle, kdy se nápad zrodí v naší mysli, až po poslední fázi, tedy testování a údržbu. Navíc se práce bude věnovat pojmu „open data“. Tato data jsou vytvářena různými společnostmi a státními institucemi a jsou volně dostupná k použití. Tento typ dat osvědčil jako velice užitečný vzhledem k rozdílným atributům a situacím, ve kterých se data sbírají. Data mohou být poté použita k vytváření smysluplných výpočtů, předpovědí a závěrů, které mohou být ku prospěchu komunity. V průběhu čtení této práce se čtenář dozví více o životním cyklu informačního systému, technologiích využívaných při vývoji webových aplikací jako takových, a závěrech, přínosech i nevýhodách vývoje aplikací, jež využívají volně dostupných dat. Dále v praktické části bude vyvinuta webová aplikace za využití programovacích jazyků HTML5, CSS3, JavaScript a PHP na základě datasetu policie Spojeného Království, který obsahuje detaily o různých zločinech, jako je například typ, datum, místo a další atributy. Tato data budou vložena do SQL databáze, jež poté data nahraje do webové aplikace. Tato webová aplikace poté bude data prezentovat pomocí Google Maps API a dalších důležitých rozhraní, což umožní poskytnout informace a jakýsi náhled nejen místní komunitě, ale také turistům.

Klíčová slova: Otevřený dat, PHP, Datové sady, Zločin, Databáze, SQL, Webové aplikace, HTML, CSS, Analýza, Návrh, Open Data aplikací

SUMMARY

As trends in web application development change almost every year, it is important to follow them, and follow the best practice of web application design and development in order to produce a product which will be widely used and which will produce results and conclusions as are intended. Every information system, including web applications, should be created by following systems development life cycle. It guides us through the process of creating an application or a system from the moment the idea appears in our mind, until, the last, testing and maintenance period. Furthermore, the term “Open data” will be discussed. This type of data is generated by different companies and government institutions and is open for free use. This type of data has proven to be very useful due to the different attributes and situations from which it is collected. Then, this data can be used to produce meaningful calculations, predictions and conclusions which can benefit the community. During the reading of this work, reader will find out more about the systems development life cycle, technologies used to create the actual web application, and the results, benefits and drawbacks of development of application which makes use of open data. Furthermore, in the practical part, a web application will be created using HTML5, CSS3, JavaScript and PHP, based on dataset from United Kingdom’s police, which includes details about various crimes, such as type, date, location and other attributes. This data will be inserted into a SQL database, which will feed the data to the web application. Then, the web application will present this data using a Google Maps API and other meaningful sections, which will provide assistance and insight to the local community and tourists as well.

Keywords: Open Data, PHP, Data set, Crime, Database, SQL, Web application, HTML, CSS, Analysis, Design, Open Data application

Contents

1	INTRODUCTION.....	9
2	AIMS AND METHODOLOGIES	10
3	LITERATURE REVIEW.....	11
3.1	Need for Open Data.....	11
3.2	Concept of Open Data	13
3.3	Five star open data	16
3.3.1	One star	17
3.3.2	Two stars	18
3.3.3	Three stars	18
3.3.4	Four stars	19
3.3.5	Five stars	20
3.4	Benefits and Barriers of Open Data	21
3.4.1	Benefits	21
3.4.2	Barriers	23
3.5	Software engineering and the Web	26
3.5.1	Software process	27
3.5.2	System development life cycle processes	33
3.6	Technologies	38
3.6.1	HTML5	38
3.6.2	Cascading Style Sheets (CSS3)	41
3.6.3	PHP	43
4	PRACTICAL PART	44
4.1	Introduction.....	44
4.2	Open Data – City of London Police.....	44
4.3	Usecase Diagram	46

4.4	Class Diagram	47
4.5	Wireframes	49
4.6	Database - MySQL	51
4.7	Front End	54
4.7.1	About Page	55
4.7.2	Contact Page.....	57
4.7.3	Registration and Login Pages.....	58
4.7.4	Main Page – Index	60
4.8	Back End	66
4.8.1	Main Page	66
4.8.2	Registration and Login	75
5	RESULTS AND DISCUSSION	82
6	CONCLUSION	86
7	BIBLIOGRAPHY	87
8	TABLE OF FIGURES	89

1 INTRODUCTION

In order to describe analysis and design process of a web application which is based on open data, it is necessary to shortly describe what open data actually is. The idea behind open data is that certain data which can serve a greater purpose should be available to everyone to use. Main goals of the movement which is supporting the use of open data are that this data should be without any restrictions and should not be bounded by copyrights and patents. Open data should be open source, with open content and open access to everyone. The concept itself is not new, but when put in the use with World Wide Web (WWW) it gains different dimension of its meaning. Open data is often made of non-textual material, such as mathematical formulas, statistical data, various parameters, medical data, and other. It is often provided by government institutions.

Systems development life cycle (SDLC) is a cycle that guides us through different stages of systems development, such as software specification, design, implementation, validation and evolution. During the reading of this diploma thesis, reader will find details on processes that are happening inside these stages, depending on project requirements. Furthermore, a web application will be developed which will make use of open data in order to display the information in an understandable manner. Different sections of the web application will display the data by different means. These include mapping each row of the dataset to a particular location on a map, providing the dataset in tabular view, and computing certain statistics based on open data attributes and values.

Finally, possible benefits of a web application which uses open data will be described, and different ways of usage of the web application will be explained in order to provide maximum benefits to the end-user.

2 AIMS AND METHODOLOGIES

The diploma thesis investigates the advantages and disadvantages design and development of web application based on open data. Main goal of this diploma thesis is to develop a web application which will make use of open data in order to provide meaningful insight to final consumers based on the dataset provided. Partial goals are:

- To explain the concept and purpose of open data;
- To provide overview of web application development life cycle which includes analysis, design and implementation stages;
- To develop a web application which will make use of open data which will be imported to the database;
- To present valuable information and draw conclusions from the web application which uses open data dataset.

The several methodology parts are defined to accomplish the thesis. First part will be focused on the theory behind open data, its use, and different models of open data. Furthermore, technologies and methods which will be used for the development process will be discussed. Development life cycle will be described and explained. Second part consists of actual design and development of a web application which uses a dataset related to crime. Information about crimes will be loaded into the database. Then, the application will present that data in an understandable way to be interpreted by the user, from which he or she can draw conclusions. Based on the literature review and results of practical part, conclusions will be formulated.

3 LITERATURE REVIEW

3.1 Need for Open Data

Before it is discussed on what open data is, and what the benefits of using and publishing open data are, it is important to understand why there is a need for open data. It is a great resource which is still not used enough to show its true potential. Various individuals and companies are collecting various types of data for the purpose of performing their tasks, finishing business processes and achieving business goals. Government plays an important role in making data actually open data, because it collects more data about society than any other individual or company. Moreover, most of the data which is collected by the government is public data by law, thus it could be made open, and so others can freely use it (The Open Knowledge Foundation, 2012). The interest for using open data is that it can be of value in many areas, and there are already many examples of its use. Other than the government, many different companies, organisations, and groups of people can benefit from its availability.

Until now, a large number of areas has achieved benefits of using open government data. Some of these include:

- Transparency and democratic control
- Participation
- Self-empowerment
- Improved or new private products and services
- Innovation
- Improved efficiency of government services
- Improved effectiveness of government services
- Impact measurement of policies
- New knowledge from combined data sources and patterns in large data volumes

For most of these areas, we can already present some examples. For example, projects such as “Tax tree” from Finland, and “Where does my money go” from Great Britain show to the public how their tax money is being spent by the government. Furthermore, there is a case

of Canada saving \$3.2 billion in charity tax fraud because of their use of open data. Open data may help people make better decisions, based on the conclusions produced from observing and analysing open government data. A website called findtoilet.dk was built by a woman in Denmark. The purpose of this website was to show the locations of all Danish public toilets. Although at first sight, it may seem like an unnecessary website, but the woman, having in mind all the people with bladder problems in Denmark built it anyway. Now, these people can trust themselves, can plan their walk in the city, and can go out more often. Another example can be a public dataset which contains all the parks in a particular city, divided on parks to which you can bring your pets, and ones on which it is forbidden. In this case, pet owners can know where they can take their dogs for a walk, meet other pet owners, discuss with them and therefore improve their lifestyle, and the life of their pets. All of the mentioned examples have been done with the use of open government data. (The Open Knowledge Foundation, 2012)

As stated above, open data can provide benefits in many areas. It also produces benefits from the economical perspective. Some studies have shown that the economic value of using open data reaches tens of billions of Euros in the EU each year. Companies are using open data when developing new products and services. Google Translate, when developing their translation algorithms, has used large amount of EU documents which were written in all European languages, therefore improving the quality of their algorithms and thus, the quality of service. In the text above, it has been mentioned how open data has improved business processes and lifestyle of people, but open data also adds in value for the government itself. One example is where the Dutch Ministry of Education has published online all of their data which was related to education in Netherlands. From that point, the number of questions they have received from the public has decreased, therefore the work-load and costs have decreased too. Also, other questions which are not answered by the data released to the public are now easier to answer, because employees focus more on answering these particular questions and therefore provide better answers, which ultimately reduces costs and improves effectiveness. Although there are many examples in which open data is already producing value and benefits in social and economic areas, its potential for the future is still uncertain. The reason for this is that different new combinations of data will produce new knowledge and understanding, which can further produce new areas of its application. As it has already happened in the past, when the relationship between water pollution and cholera

was discovered in London in the 19th century, by combining information about the deaths caused by cholera with the location of water wells. This discovery has led to the building of sewage systems in London, which improved health of people living there. It is very possible that similar discoveries will be made, because of unexpected insights provided by different sets of open data. True potential can be discovered and achieved only if the data being provided by the government is truly open, without any restrictions such as legal, financial or technological, and if it is made available for everyone to re-use (The Open Knowledge Foundation, n.d.).

3.2 Concept of Open Data

“Open data is a growing industry built around the concept of collecting, organizing, and publishing data to be consumed by a range of audiences. Making data open is the act of taking any government collected or aggregated data and providing it to users for them to consume, manipulate, and create from. In other words, open data makes data transparent, helps people using the data to become more efficient, and sparks creativity from those employing the data for any number of uses.” (Socrata Insights, 2014)

Some of the characteristics of open data include:

- **Availability:** data needs to be available for download as a whole dataset, and the highest price should not exceed the reproduction cost. The data must be provided in a form that is easy to understand, use and modify.
- **Re-use and further distribution:** the data must be provided with a permit to re-use and further distribute the data, which includes the mixing of dataset with other datasets.
- **World Contribution:** every person must be able to use, re-use and distribute further – there should be no restrictions and discrimination against certain areas, individuals, and groups. For example, restrictions of non-commercial type, that prevent “commercial” use. (Socrata Insights, 2014)

Interoperability is the reason why it needs to be clear about what openness of data really means and why the definition above is used. Interoperability signifies the ability of different systems and organizations to work together. In this case, it means the ability to mix different datasets in order to seek for new insights and conclusions. It is important because different components are allowed to operate together. This is crucial for developing large and complex systems. We can apply the same logic to data. In order to gain insight into broad range of information, datasets need to have the possibility of being freely mixed with other. This creates the main practical benefit of open data, because combining different datasets together makes space for development of better products and services, which was discussed in the previous section.

When a clear definition of openness of data is provided, it ensures the possibility mix two open datasets obtained from different sources, therefore avoiding the situation where we have many datasets, but no ways to combine them into something greater, which deprives us from gaining real benefits (Veljković, et al., 2014).

Many kinds of open data exist, which can provide benefits from potential uses and applications:

- **Cultural:** Data which is collected regarding cultural works and artefacts, and usually by galleries, libraries, museums, and other cultural institutions.
- **Science:** Data which is created as part of scientific research .
- **Finance:** Data such as government accounts and information on financial markets (stocks, shares, etc).
- **Statistics:** Data produced by statistical offices such as the census and key socioeconomic indicators.
- **Weather:** The many types of information used to understand and predict the weather and climate.
- **Environment:** Information related to the natural environment such presence and level of pollutants, the quality and rivers and seas.
- **Transport:** Data such as timetables, routes, on-time statistics (The Open Knowledge Foundation, n.d.)

As it can be seen from the examples shown so far, various benefits can be created from different datasets, but it is important that none of the datasets contains personal data, that is, data which contains information about individual persons such as their personal information. In some cases of government data, some personal information can be disclosed at times when data about public persons such as politicians is being presented.

A new situation is created, with the publishing of data, in which the public can retrieve and publish information through collaborative networking (Chun, Shulman, Sandoval, & Hovy, 2010).

The public has become a part of the data processing system, and can review it, modify it, or combine it with other datasets. Also, the public can collect their own data (for example through measuring usage of their electronic devices). This shows a change in traditional boundaries between government, organizations and everyone in the world has access to certain datasets. The traditional boundaries are being erased, and the systems are becoming more and more opened. This situation draws attention to the difference between systems which are open, and systems which are closed to the environment. Closed systems are easier to maintain and manage, because they do not stand to be affected by external influences which are often unpredictable (Janssen, et al., 2012). There is less disruption from the external factors and environment, which allow central control and planning to be used. On the other hand, the flow of data, and operations with open data cannot be predefined. The process of opening a closed system has positive impact on the problem solving effect as opening of the data produces a wide range of possibilities for the future, because these possibilities are not anticipated and planned in advance.

Feedback is important term in open systems due to the results of activities where one type of data influences on conclusions made on another type of data. In systems theory, the notion of feedback in opening of their data, governments should not deliver one-way communication towards the public. Governments should include a two-way communication, which involves receiving feedback from the public and make good use of that feedback. Opening of systems provides room for creation of feedback loops in which, as public is learning from the government, the government can learn from the public too. (Veljković, et al., 2014) This implies that the existing connection between the government and the public

is subject to change, as previous relationship between governments' closed systems and the public can no longer operate in the same manner, through traditional planning and control instruments. Opening a system usually requires a change from mechanistic control to a new perspective which involves self organization. These feedback loops require new establishment of governance mechanisms, processes and capabilities. The result depends on the government's ability to respond to these organizational changes. However, these changes in organizations and governments may create instability, and stability is necessary for organizations to operate. Institutions might both allow and disallow the full adoption of open data (Janssen, et al., 2012). The outcomes of changing the organizational processes and technology within ICT systems are difficult to predict, because of many unanticipated and possibly undesirable effects which can appear. Institutional theory suggests that the introduction of IT does not often negatively affect organizations, but rather supports current workflow, work practices and organization structures. By opening data to the public, management of organizations discovers that they can reach benefits of open data at the expense of less control. Although open data, and actions which could be done with it cannot be controlled, institutional theory claims that the flow can be guided, and that sometimes steering instruments are required. (Peters & Pierre, 1998). Outside of government boundaries, command and control mechanisms cannot be put in use. However, although use of open data seems like a collective responsibility for government, organizations and individuals, it is most probable that if society experiences negative influences from the use of open data, it will expect some kind of intervention from the government and the government will be held responsible.

3.3 Five star open data

The man who has invented the World Wide Web and an initiator of Linked Data, Tim Berners-Lee, has suggested a deployment scheme for Open Data which was based on 5 stars. This section will show each level of his five star scheme. Example data which will be used in all descriptions is '*the temperature forecast for Galway, Ireland for the next 3 days*' (Lee, n.d.).

3.3.1 One star

This level represents the data which is made available under open license in any format of choice. The following figure represents an example of a chosen data format.

Figure 1 - Format of one star data

Temperature forecast for Galway, Ireland	
Day	Lowest Temperature (°C)
Saturday, 13 November 2010	2
Sunday, 14 November 2010	4
Monday, 15 November 2010	7

Source: (Lee, n.d.)

One star data represents the lowest level of 5 star deployment scheme for Open Data. Nonetheless, it provides some costs as well as benefits for consumers and publishers of open data.

Consumer can:

- Revise and analyse the data
- Print the data
- Download and store it locally on a hard drive, optical drive or a flash drive
- Enter the data into any other system
- Modify the data
- Share the data (Lee, n.d.)

Publisher can:

- Publish the data as it is rather simple to publish
- Publish the data without any restrictions and distributing licences, allowing the consumer to freely use it and distribute (Lee, n.d.)

3.3.2 Two stars

Second level of the 5 star deployment scheme is the data with two stars. It involves publishing data in a structured way. This means that instead of publishing data in a form of an image, or a table created in one of the text editors, the data is published in the form of an Excel spreadsheet, which can be downloaded and manipulated using Microsoft Excel software.

However, although the data is accessible in a structured way (machine-readable), the data is still locked in a document. This means that the only way to retrieve the data out of the document is to use proprietary software.

With two star data, consumer can do everything that he or she could have done with one star data, but now consumer can:

- Directly process the data with proprietary software to aggregate it, perform calculations, visualise it, etc.
- Export it into another (structured) format (Lee, n.d.).

Publisher still has the benefits of one star data. It is still simple to publish and make available for free, non-restricted use.

3.3.3 Three stars

Three star data represents a higher quality data than the previous two levels. Now, the data is not only available to view and download over the World Wide Web, but also it is easier to use. Instead of being provided as a picture, table, or an Excel spreadsheet as it was the case in the first two levels, the data is provided in a CSV format.

Consumer can do everything which was made possible in the first two levels, but now he can also:

- Manipulate the data in any way that seems appropriate to the consumer, without being confined by the capabilities of any particular software.

Publisher may:

- Need converters or plug-ins to export the data from the proprietary format.
- The data is still relatively simple to publish (Lee, n.d.).

3.3.4 Four stars

This level represents a significant change, in contrast to the previous three levels. Now, we use URIs to denote elements in our dataset, so that consumers can point to those elements. Instead of having data **on** the Web, the data is now **in** the Web. The data items have a URI and can be shared. A native way to represent the data is using RDF, however, other forms such as Atom can be converted and mapped if required.

Figure 2 - Format of four star data

Temperature forecast for Galway, Ireland

Day	Lowest Temperature (°C)
Saturday, 13 November 2010	2
Sunday, 14 November 2010	4
Monday, 15 November 2010	7

`Saturday, 13 November 2010`

Source: (Lee, n.d.)

Consumer can perform everything that could be performed with three star data and additionally can:

- Link to the data from any other place (over the web or locally).
- Bookmark it.
- Reuse parts of the data.
- Reuse existing tools and libraries, even if they only understand parts of the pattern the publisher used.
- Understanding the structure of an RDF "Graph" of data can be more effort than tabular (Excel/CSV) or tree (XML/JSON) data.
- Combine the data safely with other data. URIs are a global scheme so if two things have the same URI then it's intentional, and if so that's well on it's way to being 5 star data (Lee, n.d.).

Publisher can:

- Have fine-granular control over the data items and can optimise their access (load balancing, caching, etc.)
- Other data publishers can now link into your data, promoting it to 5 star
- Typically invest some time formatting and modifying data.
- Assign URIs to data items and think about how to represent the data.
- Either find existing patterns to reuse or create your own (Lee, n.d.).

Inspecting: <http://5stardata.info/gtd-4.html>

subject	predicate	object
<http://5stardata.info/gtd-4.html#forecast20101113>	meteo:predicted	"2010-11-13T00:00:00Z"^^<http://www.w3.org/2001/
<http://5stardata.info/gtd-4.html#forecast20101113>	meteo:temperature	<http://5stardata.info/gtd-4.html#temp20101
<http://5stardata.info/gtd-4.html#forecast20101114>	meteo:predicted	"2010-11-14T00:00:00Z"^^<http://www.w3.org/2001/
<http://5stardata.info/gtd-4.html#forecast20101114>	meteo:temperature	<http://5stardata.info/gtd-4.html#temp20101
<http://5stardata.info/gtd-4.html#forecast20101115>	meteo:predicted	"2010-11-15T00:00:00Z"^^<http://www.w3.org/2001/
<http://5stardata.info/gtd-4.html#forecast20101115>	meteo:temperature	<http://5stardata.info/gtd-4.html#temp20101
<http://5stardata.info/gtd-4.html#Galway>	rdf:type	meteo:Place
<http://5stardata.info/gtd-4.html#Galway>	meteo:forecast	<http://5stardata.info/gtd-4.html#forecast20
<http://5stardata.info/gtd-4.html#Galway>	meteo:forecast	<http://5stardata.info/gtd-4.html#forecast20
<http://5stardata.info/gtd-4.html#Galway>	meteo:forecast	<http://5stardata.info/gtd-4.html#forecast20
<http://5stardata.info/gtd-4.html#temp20101113>	meteo:celsius	"2"^^<http://www.w3.org/2001/XMLSchema
<http://5stardata.info/gtd-4.html#temp20101114>	meteo:celsius	"4"^^<http://www.w3.org/2001/XMLSchema
<http://5stardata.info/gtd-4.html#temp20101115>	meteo:celsius	"7"^^<http://www.w3.org/2001/XMLSchema
<http://5stardata.info/gtd-4.html>	dcterm:title	"Temperature forecast for Galway, Ireland"
<http://5stardata.info/gtd-4.html>	xhv:stylesheet	<http://5stardata.info/style.css>
<http://5stardata.info/gtd-4.html>	dcterm:title	"Temperature forecast for Galway, Ireland"

Figure 3 - Inspection of four star data format

3.3.5 Five stars

This level contains linking data to other open data dataset in order to provide a certain context. Both the consumer and the publisher benefit from the effect that has appeared due to the linking of data.

Consumer can do everything as in four star data and additionally:

- Can discover more (related) data while consuming the data.
- Can directly learn about the data schema.
- Consumer has to deal with broken data links, just like 404 errors in web pages.

- Presenting data from an arbitrary link as fact is as risky as letting people include content from any website in your pages. Caution, trust and common sense are all still necessary (Lee, n.d.).

Publisher can:

- Make data discoverable.
- Increase the value of data.
- Own organisation will gain the same benefits from the links as the consumers.
- Publisher will need to invest resources to link data to other data on the Web.
- Publisher may need to repair broken or incorrect links (Lee, n.d.).

3.4 Benefits and Barriers of Open Data

3.4.1 Benefits

As it has been mentioned in previous sections, usage and existence of open data brings many benefits to the society, organizations and individuals. Benefits can be separated into:

- Political and social
- Economic
- Operational
- Technical

Political and social benefits are considered as the most important category. The following table provides an overview of benefits of open data.

Table 1 - Benefits of open data

Area	Benefits
Political and social	<ul style="list-style-type: none"> • More transparency • Democratic accountability • More participation and self-empowerment of citizens (users) • Creation of trust in government • Public engagement • Scrutinization of data • Equal access to data • New governmental services for citizens • Improvement of citizen services • Improvement of citizen satisfaction • Improvement of policy-making processes • More visibility for the data provider • Stimulation of knowledge developments • Creation of new insights in the public sector • New (innovative) social services
Economic	<ul style="list-style-type: none"> • Economic growth and stimulation of competitiveness • Stimulation of innovation • Contribution toward the improvement of processes, products and/or services • Development of new products and services • Use of the wisdom of the crowds: tapping into the intelligence of the collective • Creation of a new sector adding value to the economy • Availability of information for investors and companies
Operational and technical	<ul style="list-style-type: none"> • The ability to reuse data / not having to collect the same data again • and counteracting unnecessary duplication and associated costs (also by other public institutions) • Optimization of administrative processes • Improvement of public policies • Access to external problem solving capacity • Fair decision-making by enabling comparison • Easier access to data and discovery of data • Creation of new data based on combining data • External quality checks of data (validation) • Sustainability of data (no data loss) • The ability to merge, integrate and mesh public and private data

Source: (Janssen, et al., 2012)

Benefits such as stimulated innovation and promotion of economic growth can be expected from opening data, however, there is not an existing solution to predict and calculate the actual return on.

It is not possible to predict how open data can be used, as combining data creates new possibilities for usage. Open data itself does not have any value, but, when its value is increased by the way it is used. For example, a certain dataset does not need to have a value of its own, but investors and organizations can use that dataset to determine the areas that seem good for investment. Furthermore, open data can strengthen accountability, build trust between government and society, and it can improve satisfaction of the population.

The main benefit of opening data is that organizations and the public can collaborate which will increase in innovation and creativity. Even though to determine real benefits of open data, it should be analysed through various case studies, the above table shows that potential benefits are broad.

3.4.2 Barriers

Although numerous benefits can be gained through usage of open data, there are also barriers that come with it. Most of the barriers appear because of the data publishers and data users, which results in publishers not wishing to publish data, or data users not using the data in an easy and right manner. Furthermore, barriers can be categorized into legislative, technical, and information quality barriers. The following table shows some of the barriers that have been identified.

Table 2 - Barriers of open data

Category	Barriers
Institutional	<ul style="list-style-type: none"> • Emphasis of barriers and neglect of opportunities • Unclear trade-off between public values (transparency vs. privacy values) • Risk-averse culture (no entrepreneurship) • No uniform policy for publicizing data

	<ul style="list-style-type: none"> • Making public only non-value-adding data • No resources with which to publicize data (especially small agencies) • Revenue system is based on creating income from data • Fostering local organizations' interests at the expense of citizen interests • No process for dealing with user input
Task complexity	<ul style="list-style-type: none"> • Lack of ability to discover the appropriate data • No access to the original data (only processed data) • No explanation of the meaning of data • No information about the quality of the open data (see category "Information Quality") • Apps hiding the complexity, but also potential other use of open data • Duplication of data, data available in various forms or before/after processing resulting in discussions about what the source is • Difficulty in searching and browsing due to no index or other means to ensure easy search for finding the right data • Even if data can be found, users might not be aware of its potential uses • Data formats and data sets are too complex to handle and use easily • No tooling support or help desk • Focus is on making use of single data sets, whereas the real value might come from combining various data sets • Contradicting outcomes based on the use of the same data • Invalid conclusions
Use and participation	<ul style="list-style-type: none"> • No incentives for the users • Public organizations do not react to user input • Frustration at there being too many data initiatives • No time to delve into the details or no time at all • Having to pay a fee for the data • Registration required before being able to download the data • Unexpected escalated costs • No time to make use of the open data • Lack of knowledge to make use of or to make sense of data • Lack of the necessary capability to use the information • No statistical knowledge or understanding of the potential and limitations of statistics • Thread of lawsuits or other violations

Legislation	<ul style="list-style-type: none"> • Privacy violation • Security • No license for using data • Limited conditions for using data • Dispute and litigations • Prior written permission required to gain access to and reproduce data • Reuse of contracts/agreements
Information quality	<ul style="list-style-type: none"> • Lack of information • Lack of accuracy of the information • Incomplete information, only part of the total picture shown or only a certain range • Obsolete and non-valid data • Unclear value: information may appear to be irrelevant or benign when viewed in isolation, but when linked and analysed collectively it can result in new insights • Too much information to process and not sure what to look at • (Essential) Information is missing • Similar data stored in different systems yields different results
Technical	<ul style="list-style-type: none"> • Data must be in a well-defined format that is easily accessible: while the format of data is arbitrary, the format of data definitions needs to be rigorously defined • Absence of standards • No central portal or architecture • No support for making data available • Lack of meta standards • No standard software for processing open data • Fragmentation of software and applications • Legacy systems that complicate the publicizing of data

Source: (Janssen, et al., 2012)

Usually, the existing structures are used with open data, and the users' need for finding, processing and using the dataset are often forgotten. Furthermore, many institutional elements contribute to the increase in complexity, which further complicates the use of open data. It is still a challenge to find the correct data, analyse it and use it in a correct manner. Many of the identified barriers can be overcome by having good structure and support mechanisms for the use of open data. Also, users may participate less in using the open data as they can feel that they don't have the support of government and institutions, so much

effort is needed to collect it and use it, and sophisticated knowledge is necessary to link and combined different open data datasets.

One of the other problems that may appear is that the data could be simply incorrect. Some essential attributes could be missing, such as the time period in which the data was recorded. This could create discrepancies and problems in data analysis.

3.5 Software engineering and the Web

When the World Wide Web (WWW) has appeared, it was primarily a source of universally accessible information, which has affected all of us profoundly, but it had little or no effect on existing software systems. These systems were running on local workstations and could only be accessible from inside an organization. Since then, the Web has evolved as more functionalities were added to browsers. This marked the beginning of time where web-based systems could be accessed from any location, any computer which had a web browser, instead of accessing over a specially developed software user interface. Since then, a great amount of new products and services have been created, which users could access over their web browser. (Sommerville, 2010)

The development of web products and services has increased and the existing number of them has exceeded the number of actual software. Now, when developers create an application, they usually deploy it on a web server to be used within an organization, instead of deploying it on each of the users' computers, and later upgrading it on each computer separately. This has involved reduced costs, as less time is needed to set up and maintain the software. Many companies have transferred their software to web-based solutions, which interact with the existing software solutions in the company. Web services deliver specific and useful functionalities which can be used over the Web. Web applications which are developed usually integrate some web services, which may be built within the company, or provided by other companies. Linking of services has been done dynamically, so web application may use different services each time it is running. One change in the way people use the software is also the way they pay for it. Usually, software should be bought, and it can be very expensive to obtain, but with the appearance of web-based software, users may

be given free access, and in return, they just get a few advertisements shown on their user interface. Before, business software was mostly running on single computers with local communications, within an organization, but now, software is highly distributed and applications involve reuse of small services and components.

These changes have led to changes in the engineering process of web-based systems and applications. Some of them include:

- Software reuse has become the dominant approach for constructing web-based systems. When building these systems, you think about how you can assemble them from pre-existing software components and systems.
- It is now generally recognized that it is impractical to specify all the requirements for such systems in advance. Web-based systems should be developed and delivered incrementally.
- User interfaces are constrained by the capabilities of web browsers. Although technologies such as AJAX (Holdener, 2008) mean that rich interfaces can be created within a web browser, these technologies are still difficult to use. Web forms with local scripting are more commonly used. Application interfaces on web-based systems are often poorer than the specially designed user interfaces on PC system products (Sommerville, 2010).

3.5.1 Software process

In order to provide a theoretical background into the design and development process of a web-based software, a web application, it is important to mention the software process and software process models in general.

“A software process is a set of related activities that leads to the production of a software product” (Sommerville, 2010).

Activities performed in the software process may include the development of a particular application from scratch, but it is not necessary for the development of business applications. New business applications are often developed by adding new functions to the existing

applications, or by modifying and configuring existing off-the shelf software or its components.

There are many different processes in a software life cycle, but there are four which must be included:

1. Software specification – The process where the functionality and constraints of an application are defined.
2. Software design and implementation – The process where the software is produced and which meets the specification defined in the previous activity.
3. Software validation – The process in which the software is validated to make sure it does what it is supposed to do
4. Software evolution – The process where an application is modified to meet the customer needs which change through time (Sommerville, 2010).

Although these processes are integral in software engineering, in practice, they are consisted from several complex processes and sub-processes such as requirements validation, testing, architectural design, documentation, configuration management and other.

When a software application is being specified and designed, activities such as database creation, user interface design, user experience and acceptance is discussed, but, process descriptions which can also include:

- Products – This term is not used for the actual software product, but a product of any activity. For example, the product of architectural design can be a model of software architecture.
- Roles – This term reflects responsibilities of people which are involved in the analysis, design, implementation, validation and evolution processes. For example, there can be roles such as developer, database administrator, project manager and other.
- Pre- and post-conditions – These are statements which specify the condition in which the system is prior to an activity, and after an activity has been done. For example, before an architectural design is done, we need to have all

requirements specified and approved. After the architectural design is done, we get architectural models such as UML models.

It is important to note that there is no universal set of software processes which can be used for development of any software application. They are complex processes which change according to the requirements and people included in the development. Processes should take advantage of the characteristics of the actual software system, and also the people which create it. If a software system for business is created, because the requirements are often changing in business surrounding, flexible process of software development would be more effective.

We can distinguish two software processes, which are plan-driven or agile processes. Plan-driven processes, as the name says, are operating according to a predefined plan, and every outcome of a finished activity is compared to that plan to make sure the requirement has been fulfilled. In agile processes, plan is not predefined, but is created in increments, as the actual development progresses. With agile development, it is easier to change certain components if the customer requirements change in the meantime. The best solution for software development is to find a balance between plan-driven and agile processes (Sommerville, 2010).

In this section, a number of general process models will be described and presented from an architectural perspective. These models can be considered as generic, and they are an abstraction of processes which can be used to describe different approaches to software development. These models include:

- The waterfall model – This model takes fundamental processes such as specification, development, validation, and evolution and separates them into separate phases such as requirements specification or analysis, software design, software implementation, software testing, and other.
- Incremental development model – The system that is created with this model is created as a series of versions, where each version goes through processes of specification, development and validation, and each version is an upgrade of the previous version.

- Reuse-oriented software engineering – This model is based on usage of a number of existing, reusable components. The development process which follows this approach focuses on connecting and integrating different components together into a system and not developing the system from nothing.

As it was the matter with software processes, these models are not mutually exclusive, and can be used and mixed together, especially in the case of large business systems. In that case, the best solution is to combine the best features of each of development models.

Waterfall model

The principal stages of the waterfall model directly reflect the fundamental development activities:

- Requirements analysis and definition - The system's services, constraints, and goals are established by consultation with system users. They are then defined in detail and serve as a system specification.
- System and software design - The systems design process allocates the requirements to either hardware or software systems by establishing an overall system architecture. Software design involves identifying and describing the fundamental software system abstractions and their relationships.
- Implementation and unit testing - During this stage, the software design is realized as a set of programs or program units. Unit testing involves verifying that each unit meets its specification.
- Integration and system testing - The individual program units or programs are integrated and tested as a complete system to ensure that the software requirements have been met. After testing, the software system is delivered to the customer.
- Operation and maintenance - Normally (although not necessarily), this is the longest life cycle phase. The system is installed and put into practical use. Maintenance involves correcting errors which were not discovered in earlier stages of the life cycle, improving the implementation of system units and enhancing the system's services as new requirements are discovered (Sommerville, 2010).

In principle, the result of each phase means that that phase is finished, and the next phase can begin. The following phase should not start until the results of the previous phase have been approved. In practice, it has been proved that these stages may overlap, and some stages require information from other stages. So, for example, during the design of a software solution, problems with requirements are identified. During the development, developers may find issues in design. This software process is not a linear activity, and it involves response from one phase to another. Documents which are created for each phase may have to be modified later to show the changes which were made later.

Because there are costs involved with production and approval of documents for each phase, these iteration can be costly and time consuming. Usually, if there is a problem in one phase, the work on that phase stops, and the work on a different phase starts. So, by continuing to work on a different phase, the result may be a solution of a problem which persisted in a different phase. This reduces the costs of writing and modifying documentation each time something is done in a particular phase. During the final phase of a software development life cycle, software is put into use for consumers. In this phase, final program errors and design issues are identified, so that they can be fixed, and new functionalities can be added in the evolution phase.

Incremental development

Incremental software development has proven to be better approach to software development for most business and e-commerce systems in comparison to the waterfall approach. It reflects the real way of solving problems, because software is rarely completed in one iteration. Usually, software development is done in a series of steps, where we revise each step in order to discover possible mistakes. With the incremental development, it costs less and requires less effort to implement the changes in the software. Each version of the systems include implementation of certain functionalities which are required by the customer. Usually, most important functionalities are implemented in the first increments of the system. It means that we can present the system early to the customer, and therefore, receive customer feedback in the early stages of the software development life cycle. Customer can evaluate the system and point to early mistakes and changes to be made.

Incremental development has three important benefits, compared to the waterfall model:

1. The cost of accommodating changing customer requirements is reduced. The amount of analysis and documentation that has to be redone is much less than is required with the waterfall model.
2. It is easier to get customer feedback on the development work that has been done. Customers can comment on demonstrations of the software and see how much has been implemented. Customers find it difficult to judge progress from software design documents.
3. More rapid delivery and deployment of useful software to the customer is possible, even if all of the functionality has not been included. Customers are able to use and gain value from the software earlier than is possible with a waterfall process.

Reuse-oriented software engineering

Today, there is software reuse in majority of software development projects. This happens when developers which are working on a certain project already have a block of code, a plugin, an API or other component, which could be implemented in their current project, instead of writing the same or very similar code from scratch. Developers usually modify these components to fit into the current project, and implement them into the system. Reuse-oriented approach relies on a large number of reusable components being connected in one system. Sometimes, these components can be in a form of a commercial off-the-shelf software, which provide a specific functionality.

Although the initial stages such as requirements specification can be similar to stages in different approaches, the middle stages are very different from other approaches. These stages are:

1. Component analysis - Given the requirements specification, a search is made for components to implement that specification. Usually, there is no exact match and the components that may be used only provide some of the functionality required.

2. Requirements modification - During this stage, the requirements are analysed using information about the components that have been discovered. They are then modified to reflect the available components. Where modifications are impossible, the component analysis activity may be re-entered to search for alternative solutions.

3. System design with reuse - During this phase, the framework of the system is designed or an existing framework is reused. The designers take into account the components that are reused and organize the framework to cater for this. Some new software may have to be designed if reusable components are not available.

4. Development and integration - Software that cannot be externally procured is developed, and the components and COTS systems are integrated to create the new system. System integration, in this model, may be part of the development process rather than a separate activity (Sommerville, 2010).

There are three types of software component that may be used in a reuse-oriented process:

- 1. Web services** that are developed according to service standards and which are available for remote invocation.
- 2. Collections of objects** that are developed as a package to be integrated with a component framework such as .NET or J2EE.
- 3. Stand-alone software systems** that are configured for use in a particular environment.

3.5.2 System development life cycle processes

As has been mentioned before, system development life cycle is a number of processes through which a software or web application goes through various stages of being specified, designed, implemented, tested, and updated. Many various software tools are being used during this life cycle. They are useful for editing documents during the project specification, or modelling detailed information into different UML diagrams.

Four basic processes, specification, development, validation and evolution are organized in a different manner, depending on the project and approach to the development. In the waterfall model, these activities are organized sequentially, as opposed to the incremental development where these processes are interleaved (Sommerville, 2010).

3.5.2.1 *Software specification*

This process, which is also called requirements engineering is where services which are required from software or web application are clearly defined, and constraints are identified. It is a critical stage of software development life cycle, because errors that are made in this stage are increased, and harder to fix and modify in latter stages. The aim of this process is to produce a document that contains system requirements which need to satisfy stakeholders' requirements. Usually, these are documented at two levels of detail. Part of the document needs to present the system and its processes to the customers, or end-users. Second part of the document is written in more detail and helps system developers in other SDLC stages.

There are four main activities in the requirements engineering process:

- 1. Feasibility study** - An estimate is made of whether the identified user needs may be satisfied using current software and hardware technologies. The study considers whether the proposed system will be cost-effective from a business point of view and if it can be developed within existing budgetary constraints. A feasibility study should be relatively cheap and quick. The result should inform the decision of whether or not to go ahead with a more detailed analysis.
- 2. Requirements elicitation and analysis** - This is the process of deriving the system requirements through observation of existing systems, discussions with potential users and procurers, task analysis, and so on. This may involve the development of one or more system models and prototypes. These help you understand the system to be specified.
- 3. Requirements specification** - Requirements specification is the activity of translating the information gathered during the analysis activity into a document that defines a set of requirements. Two types of requirements may be included in this document. User requirements are abstract statements of the system requirements for the customer and end-user of the system; system requirements are a more detailed description of the functionality to be provided.
- 4. Requirements validation** - This activity checks the requirements for realism, consistency, and completeness. During this process, errors in the requirements document are

inevitably discovered. It must then be modified to correct these problems (Sommerville, 2010).

3.5.2.2 Software design and implementation

In this stage, the system is actually being designed and developed according to the project specification and collected requirements. It involves processes of software design, where various UML diagrams are designed, and programming, where developers code the application in the programming language that is specified and required from the stakeholders. At this stage, if incremental approach to development is used, developers may sometimes need to modify and update the document made in software specification stage. Software design is actually a description of the structure of application which is to be developed, database design, models and structures to be used by the system. Furthermore, it includes design of user interfaces and interfaces between different components of the system. From this stage, it is almost impossible to successfully finish the design and development without modifying and revising previous work that has been done, in order to correct parts of the system where errors were made or have been overlooked.

Majority of software and web applications interface with other systems or application programming interfaces (APIs). These include the operating system, databases, middleware, and other applications. All of these are called the “software platform”, which is the environment in which the application operates. Information about applications in the platform is crucial to the design process, as designers have to decide how to integrate these smaller applications with the application that is built during the project. So, for example, in order to use one kind of database, we need to design or code some parts of the application differently. Specified requirements usually contain the description of the functionality of the software, and performance requirements. In order to achieve the required performance, the applications in the platform have to be carefully selected, as some operate slower in comparison to others.

The activities in the design and implementation process cannot be strictly defined, as they vary depending on the type of system. Some static web applications which are used only for

presentational purpose do not have the database, so the activity where database is designed is not necessary.

Following four activities can, but do not have to be a part of the design process:

1. **Architectural design**, where you identify the overall structure of the system, the principal components (sometimes called sub-systems or modules), their relationships, and how they are distributed.
2. **Interface design**, where you define the interfaces between system components. This interface specification must be unambiguous. With a precise interface, a component can be used without other components having to know how it is implemented. Once interface specifications are agreed, the components can be designed and developed concurrently.
3. **Component design**, where you take each system component and design how it will operate. This may be a simple statement of the expected functionality to be implemented, with the specific design left to the programmer. Alternatively, it may be a list of changes to be made to a reusable component or a detailed design model.
4. **Database design**, where you design the system data structures and how these are to be represented in a database. Again, the work here depends on whether an existing database is to be reused or a new database is to be created.

3.5.2.3 Software validation

The purpose of this stage of SDLC is to show that the system which was designed and developed conforms to the specified requirements and that customers' expectations are met. The main validation technique is program testing, where the system is put into operation with test data, which should produce specified outcomes during the simulation. Validation can include processes such as inspections and reviews, where requirements are checked at each stage of the software life cycle, from user requirements definition to program development. Due to time required for testing, and extensive testing processes, it is usually done during development and after an application is implemented. Larger systems should

not be tested as a single unit, but rather as a set of components, which each component, or functionality of the system is tested separately. When testing is done with the test data, the testing process can proceed with the use of real customer's data. In the best case, defects are discovered early during the testing process. However, as bugs are discovered, the program must be revised and modified, which may involve repetition of tests which were previously done. This is an iterative process, where information from the later stages of the process leads to the change in the earlier stages of validation.

The stages in the testing process are:

1. **Development testing** - The components making up the system are tested by the people developing the system. Each component is tested independently, without other system components. Components may be simple entities such as functions or object classes, or may be coherent groupings of these entities. Test automation tools, such as JUnit that can re-run component tests when new version of the component are created, are commonly used.
2. **System testing** - System components are integrated to create a complete system. This process is concerned with finding errors that result from unanticipated interactions between components and component interface problems. It is also concerned with showing that the system meets its functional and non-functional requirements, and testing the emergent system properties. For large systems, this may be a multi-stage process where components are integrated to form subsystems that are individually tested before these sub-systems are themselves integrated to form the final system.
3. **Acceptance testing** - This is the final stage in the testing process before the system is accepted for operational use. The system is tested with data supplied by the system customer rather than with simulated test data. Acceptance testing may reveal errors and omissions in the system requirements definition, because the real data exercise the system in different ways from the test data. Acceptance testing may also reveal requirements problems where the system's facilities do not really meet the user's needs or the system performance is unacceptable (Sommerville, 2010).

3.5.2.4 *Software evolution*

As opposed to hardware, software is very flexible and prone changes. At any time during or after the system development, changes can be implemented if the software does not meet the requirements or if the end-user has a desire to change a part of the system, or to implement a new component. Once the development stage is done, there is a clear gap between the development and evolution stages. Although this stage is not continuous, as maintenance and updating is done periodically, the costs can sometimes be greater than the costs of actual development of the system. However, the effort needed to maintain and update the system is smaller. Today, none of the systems is considered as a finished product after the development stage, but the development is an ongoing process which actually never ends, due to the maintenance, updating and evolution of the system as a whole.

At this stage, developers are open to suggestions from the end-users on how to improve the system, which new functionalities to include, or which old functionalities are no longer necessary. These suggestions are collected over time, and revised periodically. After that, approved changes to the system are being implemented, and the validation process is performed again to ensure that all new requirements are implemented and working as specified.

3.6 Technologies

3.6.1 HTML5

First developed in 2007, and spearheaded by Google and Apple, HTML 5 is the latest revision of Hypertext Markup Language (HTML), the industry standard for the structure and presentation of content on the World Wide Web. HTML 5 is powering an entirely new generation of Internet experiences, both for broadband Internet and mobile Internet. Today there are 1.4 billion Internet users. However, due to the proliferation and adoption of mobile devices, by 2015 there will be more people accessing the Internet via a mobile device than the traditional Internet (Mary Meeker, Morgan Stanley 2010).

What sets HTML 5 apart from the previous versions of HTML is that HTML 5 has the potential to bring together the last 20 years of the world's experience with the Internet, and unify it into a broadly supported standard.

HTML 5, once fully deployed, will help reduce device fragmentation issues in the market, enabling marketers to unleash their creativity to create new innovative, intuitive and compelling engagements via mobile:

- Cross-platform, cross-network, static and real-time rich media (e.g. video, audio)
- Streamlined social media integration and enablement
- Easier and more innovative customer care and relationship marketing
- Improved and enhanced user navigation, site usability and automatic content refreshing on mobile sites
- Ability to retain user selection, preference and history for a better user specific and context appropriate personalized mobile site experience
- New advertising and social media experiences utilizing local storage for context and history
- Native app experience within a browser, an experience also known as the “Hybrid App” or “Web App”
- Gaming, augmented reality and a wide range of other cloud computing based services which will evolve by leveraging HTML 5 standards and new features (Frain, 2012)

What makes these new engagements and delivery models possible is the introduction of a number of new web browser features in the HTML 5 standard that offer these capabilities:

- Embed video, audio, barcode scanning, rich media content and more within a browser without requiring third party plug-ins such as Flash, Silverlight and other media players for playback or functionality
- Lightweight size and streamlined coding of HTML 5 is ideal for limited memory devices like mobile phones since the page script will be more efficient

- Store/cache data and content locally on a device within the browser, which improves performance speed and supports offline access to the data or content experience
- Using the HTML 5 image canvas, full gaming environments and other interactive rich media can be created native within the HTML code
- Improved semantics providing contextual descriptions for content in the HTML code itself making it an easier to use and more elegant coding language
- Advanced Forms where the browser does more work for validation, making JavaScript more efficient and offering faster page rendering
- Create a wide range of new engagements that can be adjusted in real-time based on context including:
 - Interactive drag-and-drop image creation canvases
 - Content editing and creation (images, audio, and video) utilizing enhanced visual navigation and interaction controls
 - Media playback
 - Page rendering and presentation styles offering background media and style management, and over 14 new input form field types
 - Enhanced and easier authoring of touch and scrolling effects
 - Auto-complete and drop-down
- Other discussed features for HTML 5 include:
 - Standardization and unification of metrics and analytics cross platform
 - UI enhancements such as carrousel, pinch/spread/zoom, pop-over menus, selections spinners etc.
 - Cross browser consistency for in-page dialogs for confirm/cancel actions
- Access a device's hardware and software features, including:
 - Camera and microphone (Capture API)
 - Location/GPS chip using HTML 5 APIs
 - Contact services (personal contact software)
 - SMS (text message)

- Gyroscope, accelerometer, compass, file and media system using APIs
 - Notification systems (alarms, reminders)
 - 3D graphic engines (WebGL library)
 - NFC and RFID chips
- Integrated and streamlined use of background processes both on the device and through secure sockets, allowing for a richer engagement without interrupting the user's experience, and providing simpler cross-platform application content updates. The improved web service APIs enhances the performance of processor-intensive tasks and scripts in the background, independently of the overall experience.
 - Reduce the expense and time needed for the rigorous certification, testing and vetting requirements of application stores as they align to the new HTML 5 specification
 - Initiate content updates faster and more efficiently without complete reload of a web page, which is critical for small screen devices or low bandwidth environments
 - Interface with existing open standards, for example ORMMA (Open Rich Media for Mobile Advertising) (Moore, et al., 2011)
 - Segment an audience and deliver tailored content experiences based on a wide range of demographic, location and behavioural factors, or simply enable A/B testing of different marketing interactions utilizing local device segment data persistence
 - Leverage familiar and well-understood Internet development skills, tools and processes—HTML 5 is an extension of existing mobile site technologies, specifications and site development processes

3.6.2 Cascading Style Sheets (CSS3)

Cascading Style Sheets, most of the time abbreviated as CSS, is a stylesheet language used to describe the presentation of a document written in HTML or XML (including various XML languages like SVG or XHTML). CSS describes how the structured element must be rendered on screen, on paper, in speech, or on other media (Network, 2015).

CSS is one of the core languages of the open web and has a standardized W3C specification. Developed in levels, CSS1 is now obsolete, CSS2.1 a recommendation and CSS3, now split into smaller modules, is progressing on the standard track.

The basic goal of the Cascading Stylesheet language is to allow a browser engine to paint elements of the page with specific features, like colors, positioning, or decorations. The *CSS syntax* reflects this goal and its basic building blocks are:

- The **property** which is an identifier, that is a human-readable *name*, that defines which feature is considered.
- The **value** which describe how the feature must be handled by the engine. Each property has a set of valid values, defined by a formal grammar, as well as a semantic meaning, implemented by the browser engine.

Before CSS, web designers were limited to the layout and styling options of HTML. And if you surfed the Web in 1995, then you understand the emphasis on limited. HTML still forms the foundation of all pages on the World Wide Web, but it's simply not a design tool. Sure, HTML provides basic formatting options for text, images, tables, and other web page elements, and patient, meticulous webmasters can make pages look pretty good using only HTML. But the result is often sluggish web pages laden with clunky code (McFarland, 2013).

CSS, in contrast, offers the following advantages:

- Style sheets offer far more formatting choices than HTML. With CSS, you can format paragraphs as they appear in a magazine or newspaper (the first line indented and no space between each paragraph, for example) and control the leading (the space between lines of type in a paragraph).
- When you use CSS to add a background image to a page, you get to decide whether and how it tiles (repeats). HTML can't even begin to do that.
- Even better, CSS styles take up much less space than HTML's formatting options, such as the much-hated `` tag. You can usually trim a lot of kilobytes from text-heavy web pages by using CSS. As a result, your pages look great and load faster.
- Style sheets also make updating your site easier. You can collect all your styles into a single external style sheet that's linked to every page in your site. Then, when you

edit a style, that change immediately ripples through your site wherever that style appears. You can completely change the appearance of a site just by editing a single style sheet (McFarland, 2013).

3.6.3 PHP

PHP is a flexible, dynamic language that supports a variety of programming techniques. It has evolved dramatically over the years, notably adding a solid object-oriented model in PHP 5.0 (2004), anonymous functions and namespaces in PHP 5.3 (2009), and traits in PHP 5.4 (2012). PHP has a very complete set of object-oriented programming features including support for classes, abstract classes, interfaces, inheritance, constructors, cloning, exceptions, and more (Tatroe, et al., 2013).

What makes PHP different from other scripting languages such as JavaScript, is that the execution of the code happens on the server, which generates HTML which is further forwarded to the client. The client receives the final result that the script has produced, but still cannot know what the actual code was. It is possible to configure the web server to consider all HTML files as PHP, so the possibility of users knowing the code equals to zero.

The PHP software works with the web server, which is the software that delivers web pages to the world. When you type a URL into your web browser's address bar, you're sending a message to the web server at that URL, asking it to send you an HTML file. The web server responds by sending the requested file. Your browser reads the HTML file and displays the web page. You also request a file from the web server when you click a link in a web page. In addition, the web server processes a file when you click a web page button that submits a form. This process is essentially the same when PHP is installed. You request a file, the web server happens to be running PHP, and it sends HTML back to the browser, thanks to the programming in PHP (Lockhart, 2015).

4 PRACTICAL PART

4.1 Introduction

In order to present how open data can be used, accessed and viewed by general population, a web application was developed, which presents open data in a way that is understandable and interpretable. As it is previously mentioned, the web application which was developed, named “Crime Spotter”, uses open data to map crimes to a map, also providing detailed information about each crime in the database, and also certain statistics which are computed. The reason behind the development of a web application of this type lies in possible benefits which can be gained through its usage. By using this application, population can discover which crimes have been committed in the neighbourhood of their interest, which leads different conclusions. For example, a user which wants to buy a real estate in a certain part of city can use Crime Spotter and find which crime occurred in that neighbourhood. Based on that, he or she can decide whether to purchase or rent the real estate in question, or not. Furthermore, it is useful for families which want to relocate. They can make conclusions on whether a particular neighbourhood is a safe place for their children to grow up, and other different conclusions.

Web application was developed with use HTML, CSS, and JavaScript for the development of front-end, the design and structure of the application. Because the application needs to communicate with the server, a server-side scripting language, PHP, was used to establish connection to the database, and also to retrieve and insert the data into the database. Other functions which are written for the web application were also written in PHP. The database management system that is used for hosting the database is MySQL, which is integrated into the local server WAMP, intended for development on local machines.

4.2 Open Data – City of London Police

Data that is used by Crime Spotter is provided by the police of United Kingdom. Datasets can be downloaded from <http://data.police.uk>. Although this page contains open data from all police forces in the United Kingdom, for the purpose of development of this web

application, only data from London's police was used. The data contains information on crimes that occurred in December 2013, and December 2014.

The reason behind the selection of these two months is to make the comparison of crime numbers in the same month, and in two different years possible. With this decision, we can see an increase or decrease in crime that happened after one year. The data was downloaded in CSV format, which is classified as "Three Stars" category of a Five Star model which was previously described.

The columns in the CSV files are as follows:

Table 3 - Columns of CSV format of dataset

Field	Meaning
Reported by	The force that provided the data about the crime.
Falls within	The force which has jurisdiction of that area
Longitude and Latitude	The anonymised coordinates of the crime.
LSOA code and LSOA name	References to the Lower Layer Super Output Area that the anonymised point falls into.
Crime type	One of the crime types.
Last outcome category	A reference to whichever of the outcomes associated with the crime occurred most recently.
Context	A field provided for forces to provide additional human-readable data about individual crimes.

Source: (Police)

The following figure shows the contents of the CSV file which contains open data to be used in the creation of database, and the functioning of the web application being developed.

Figure 4 - Contents of dataset file

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Crime ID,	Month,	Reported by,	Falls within,	Longitude,	Latitude,	Location,	LSOA code,	LSOA name,	Crime type,	Last outcome category,	Context		
2	667192ce29db9089f9db6908544bffa9a61eface9cd43dddc113d846d2b8db6,	2014-12,	City of London Police,	City of London Police,	-0.111497,	51.51822								
3	ce5e6047bda6ef83794ead97e487066b39f37d73847f6d6464d38691db13d412,	2014-12,	City of London Police,	City of London Police,	-0.111497,	51.51822								
4	,	2014-12,	City of London Police,	City of London Police,	-0.097601,	51.520699,	On or near Carthusian Street,	E01000001,	City of London 001A,	Anti-social I				
5	f1f2f0f40aef77ec680875b1f9ee8ff1dea1fef657d612ac8516cc17c1168e6d,	2014-12,	City of London Police,	City of London Police,	-0.098642,	51.517146,	O							
6	a305e3406dea928b5826a8c49d8ff8723f9ae92b14617414cfea3d216cbf93ad,	2014-12,	City of London Police,	City of London Police,	-0.097601,	51.520699,								

Source: (Police, 2015)

Next to providing the data in a CSV format, the police of United Kingdom have created an API which can be used directly in the web application, without creation of database specifically to contain data about crimes.

The API is implemented as a standard JSON web service using HTTP GET and POST requests. Full request and response examples are provided in the documentation on their website.

4.3 Usecase Diagram

In order to show possible actions which users can perform on Crime Spotter, a unified modelling language diagram has been created, which explains which type of user can do which action. In this case, a usecase diagram was created.

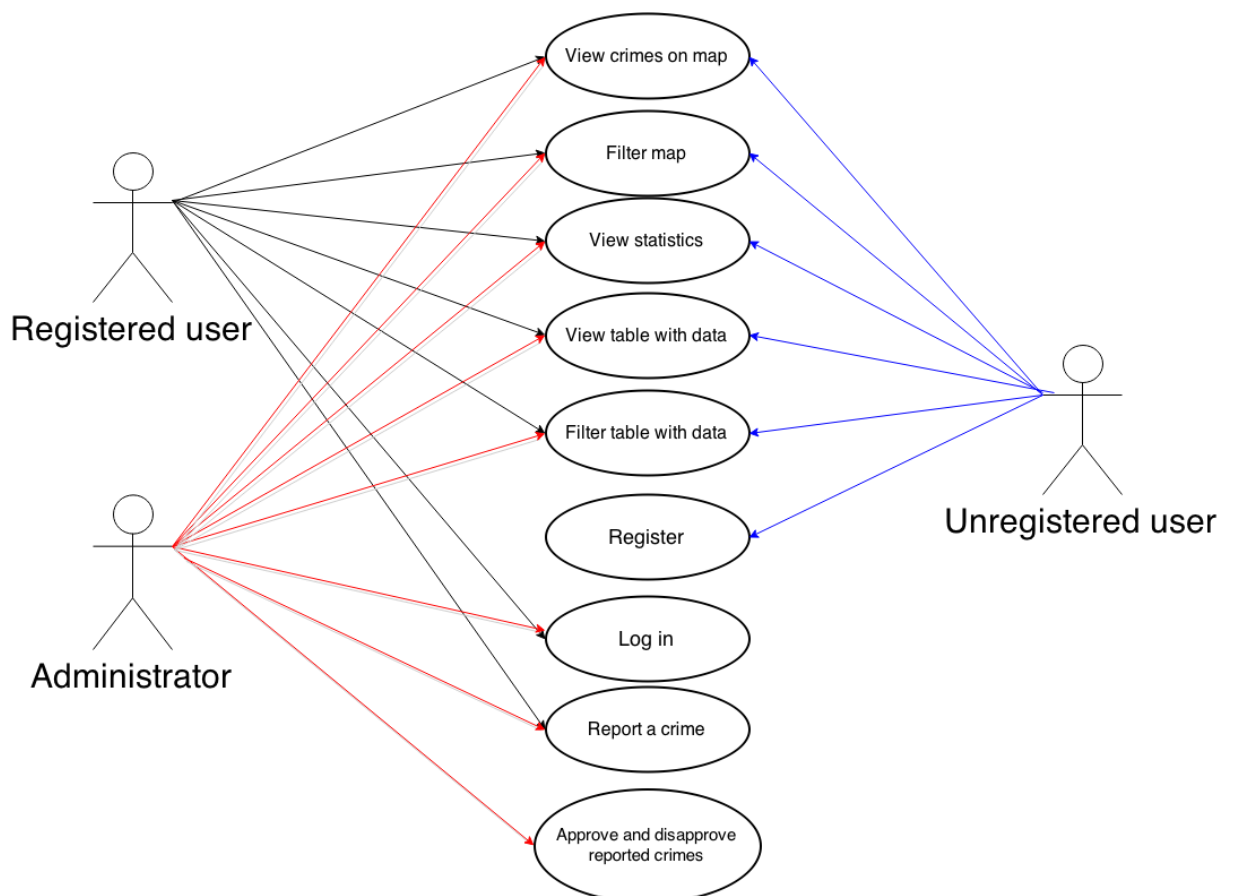


Figure 5 - Usecase diagram of Crime Spotter web application

As it can be seen, an unregistered user will have the possibility of:

- Viewing crimes on map
- Filtering a map based on type or date
- Viewing statistics such as pie and bar charts
- Viewing table that contains whole dataset
- Filtering table based user input
- Registering in order to gain more functionalities

A registered user will have a possibility of performing all actions that are available for unregistered user, but also he or she can:

- Report a crime by filling a form with appropriate data and valid source of information
- Log in in order to be able to report a crime

Administrator can perform all actions as other users, but he or she also has the possibility of:

- Reporting a crime
- Reviewing submitted (reported) crimes
- Approving or disapproving submitted crimes in case of valid or invalid source of information
- Logging in with administrator credentials in order to unlock actions available for administrator.

4.4 Class Diagram

In order to show theoretical structure of the database, and the functionality of the web application, a class diagram was created which shows how users interact with the data, and also the structure of classes and their relationships.

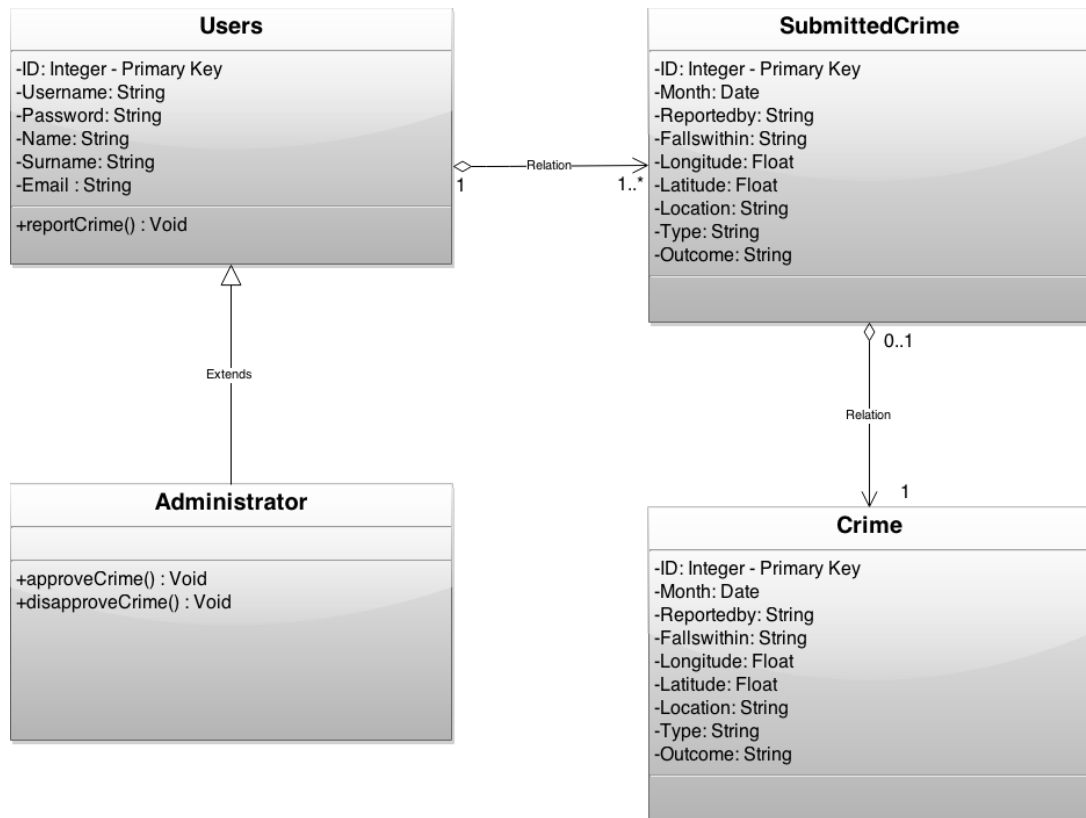


Figure 6 - Class diagram of Crime Spotter web application

As it can be seen in the figure shown above, there is a class “Users” which contains data about registered users, whether it is a regular user or administrator. Class “Administrator” extends the Users class, and contains the same set of attributes and operations as regular users have, but administrator also has two additional operations, to approve and disapprove a crime that is submitted. If the submitted crime is approved, it is inserted into a table that contains all crimes. If it is disapproved, it is deleted from the table of submitted crimes. Other two classes are “SubmittedCrime” and “Crime” with no difference in attributes and operations. The only difference is that the data which is shown live on Crime Spotter is retrieved from the Crime table, and SubmittedCrime table contains only the information on crimes which are about to be approved or disapproved by the administrator.

4.5 Wireframes

Before starting the actual development of Crime Spotter, it was important to create a wireframe which would approximately present the idea of the design of main page. Creation of wireframe serves as a guideline during the web application development as it shows designer's idea on the structure of a web application, and the positioning of HTML elements.

First wireframe that was created represents HTML elements that are viewed first, as soon as user opens a web application.

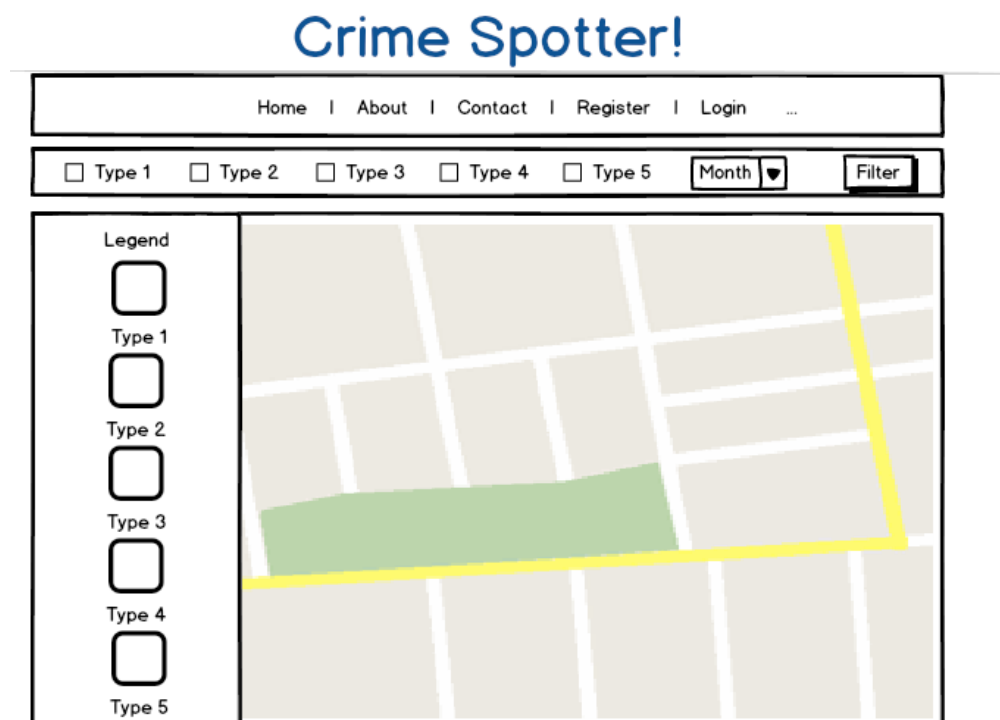


Figure 7 - Map section of main page of Crime Spotter

As it can be seen in the figure above, when a user opens Crime Spotter, he or she will be able to select other pages to view in the navigation bar. The navigation bar will be included in all pages of this web application. Next, there is a rectangle which contains certain check boxes and a combo box. These will serve as filters, which will send filter data to the application, and request filtered data from the database. Under that section, the main section of Crime Spotter is placed, which contains legend on the left side, in order to describe what

each icon means. On the right side, a map will be shown, which will contain all crimes from the database, mapped as markers on the map.

Next wireframe that was created also represents the main page of Crime Spotter, but shows different sections. The following picture shows charts which will present certain statistics about the data from the database.

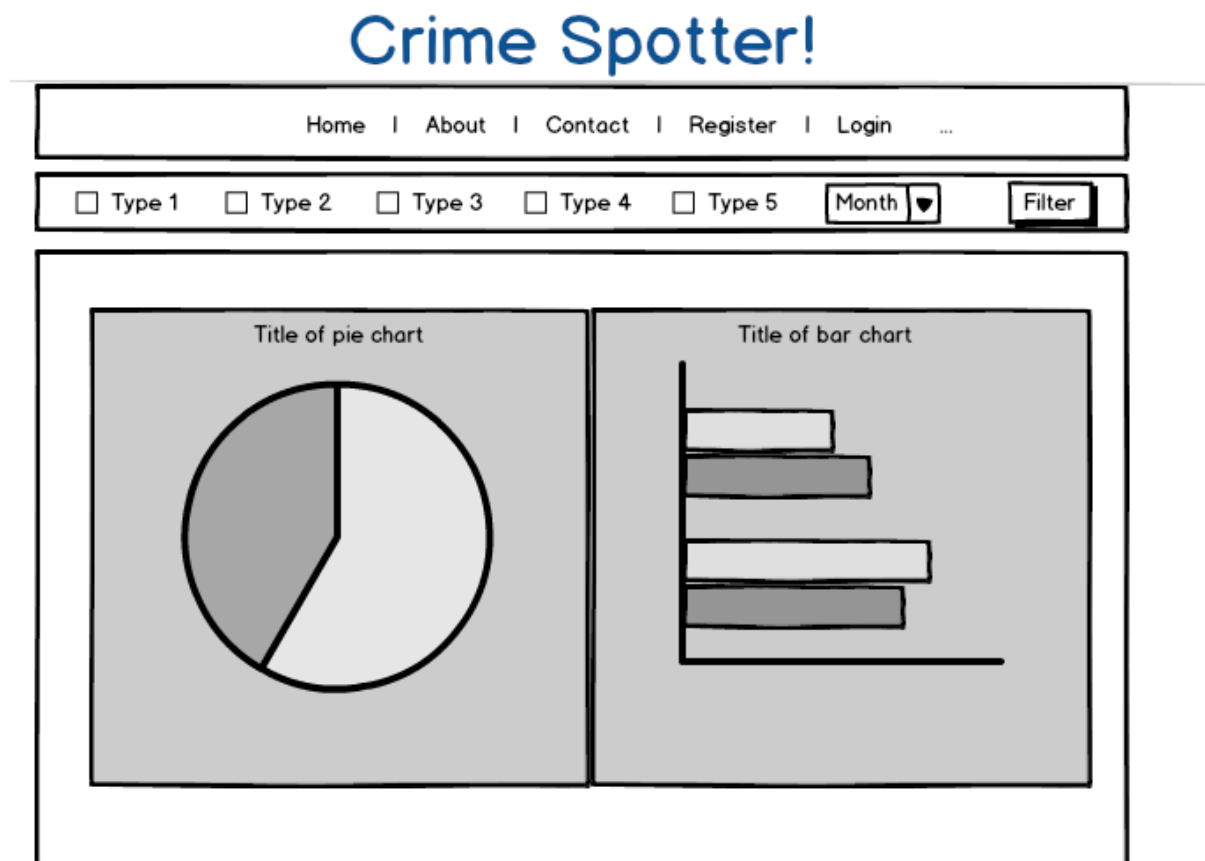


Figure 8 - Chart section of Crime Spotter

The pie chart shows percentage of crimes of certain type, compared to the total amount of all crimes for the particular month. Bar chart will show the number of crimes of certain type, compared to the number of crimes of same type, but in a different time period.

Last section of the main page will contain a table where all rows from the table that contains crimes will be shown. User will be able to filter the table according to the input he or she provides.

Crime Spotter!

The screenshot shows the main page of the 'Crime Spotter!' application. At the top, there is a navigation menu with links for Home, About, Contact, Register, and Login. Below the menu, there are five radio buttons labeled 'Type 1' through 'Type 5', a 'Month' dropdown menu, and a 'Filter' button. A second 'Filter' button is located above an input field labeled 'Input for filtering the table'. Below the input field is a table with six columns and five rows of data.

Column 1	Column 2	Column 3	Column 4	Column 5	Column 6
Item One	Item One	Item One	Item One	Item One	Item One
Item Two	Item Two	Item Two	Item Two	Item Two	Item Two
Item Three	Item Three	Item Three	Item Three	Item Three	Item Three
Item Four	Item Four	Item Four	Item Four	Item Four	Item Four

Figure 9 - Table section of main page of Crime Spotter

4.6 Database - MySQL

For the beginning of the development of Crime Spotter, database creation was necessary in order to store all the data regarding the registered users and the crimes that were committed. MySQL was the choice of database management system, as it come prepacked with WAMP local server, on which the application is developed and tested. As seen in constructed class diagram, it was necessary to create several tables that would store all the data.

The choice was to make the following tables:

- Users – which will store user's personal information as well as information needed for login process.
- SubmittedCrime – which will temporarily store the crimes that were submitted, until administrator decides on whether to approve their insertion

into the table that holds crimes, or to disapprove and delete the information about the crime from the database.

- Crime – which holds information about crimes and serves it to the web application

The structure of tables is as follows:

- Users:
 - ID – Integer ; Is not presented in web application, serves as identification for rows during queries to the database
 - Username – Varchar
 - Password – Char
 - Name – Varchar
 - Surname – Varchar
 - Email – Varchar
 - Salt – attribute used for security purposes; Value not shown in web application
- Crime:
 - ID – Integer; Identification for rows necessary for queries
 - Month – Varchar ; Written in YYYY-MM format
 - Reportedby – Varchar
 - Fallswithin - Varchar
 - Longitude – Float
 - Latitude – Float
 - Location – Varchar
 - Type – Varchar
 - Outcome – Varchar
- SubmittedCrime – has the same structure as Crime table

During the development of database structure, some attributes from the original data file were omitted since they were not so relevant for the purpose of Crime Spotter. In order to see the meaning of attributes in the Crime table, please refer to the previous section where the data in use is described.

The following figures show SQL queries that produce the database and tables with the above mentioned structure:

```
CREATE TABLE IF NOT EXISTS `crime` (
  `ID` int(11) NOT NULL AUTO_INCREMENT,
  `Month` varchar(7) DEFAULT NULL,
  `Reportedby` varchar(21) DEFAULT NULL,
  `Fallswithin` varchar(21) DEFAULT NULL,
  `Longitude` float DEFAULT NULL,
  `Latitude` float DEFAULT NULL,
  `Location` varchar(46) DEFAULT NULL,
  `Type` varchar(28) DEFAULT NULL,
  `Outcome` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`ID`),|
  KEY `ID` (`ID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO_INCREMENT=970 ;
```

Figure 10 - SQL query for creation of 'crime' table

```
CREATE TABLE IF NOT EXISTS `users` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `username` varchar(255) NOT NULL,
  `password` char(64) NOT NULL,
  `name` varchar(255) NOT NULL,
  `surname` varchar(255) NOT NULL,
  `email` varchar(255) NOT NULL,
  `salt` char(16) NOT NULL,
  PRIMARY KEY (`id`),
  KEY `username` (`username`,`email`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=4 ;
```

Figure 11 - SQL query for creation of 'users' table

```
CREATE TABLE IF NOT EXISTS `submittedcrime` (  
  `ID` int(11) NOT NULL AUTO_INCREMENT,  
  `Month` varchar(7) DEFAULT NULL,  
  `Reportedby` varchar(21) DEFAULT NULL,  
  `Fallswithin` varchar(21) DEFAULT NULL,  
  `Longitude` float DEFAULT NULL,  
  `Latitude` float DEFAULT NULL,  
  `Location` varchar(46) DEFAULT NULL,  
  `Type` varchar(28) DEFAULT NULL,  
  `Outcome` varchar(45) DEFAULT NULL,  
  PRIMARY KEY (`ID`),  
  KEY `ID` (`ID`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO_INCREMENT=61 ;
```

Figure 12 - SQL query for creation of 'submittedcrime' table

4.7 Front End

Before retrieving the data from the database, and displaying it to users of the web application, it was important to create a layout and structure for the application. This was done using HTML5, CSS, and JavaScript. Furthermore, a CSS framework called Bootstrap was used in development process to make it faster and simpler. Bootstrap comes with premade styles for layout and HTML elements, which is very useful for agile development. First sections which were created are the static sections of the web application.

These include:

- Header – navigation section
- About page
- Contact page
- Registration page

Header section was the first section to be developed. It contains the logo and the navigation bar.

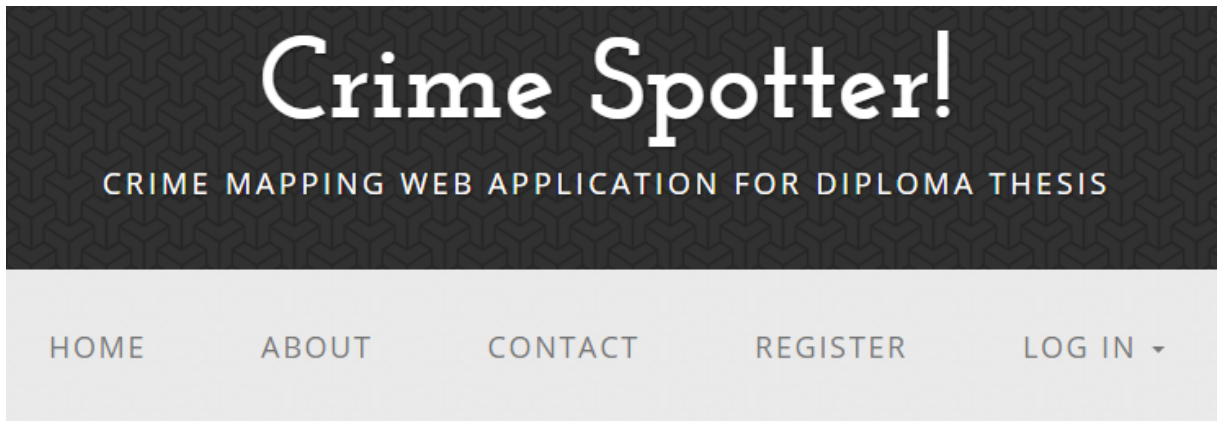


Figure 13 - Header section of Crime Spotter

Because the index page which is presented to the user first, contains the most important part of the web application, it will be described in later sections of this diploma thesis.

In the navigation section we have buttons to access other parts of the web application. These are:

- About page – Where Crime Spotter is briefly described, and some information on developer were given
- Contact page – Where contact form is present in case users have questions regarding Crime Spotter
- Register page- Where registration form is presented
- Log in dropdown button – which opens a form for logging in

4.7.1 About Page

In the next figure, it can be seen how Bootstrap's premade classes were used to quickly construct a section of this page.


```

<div class="row">
  <div class="box">
    <div class="col-lg-12">
      <hr>
      <h2 class="intro-text text-center">About
        <strong>Crime Spotter</strong>
      </h2>
      <hr>
    </div>
    <div class="col-md-6">
      
    </div>
    <div class="col-md-6">
      <p>Crime Spotter is a crime mapping web application developed for the purpose
      <p>It uses open data dataset from the police of London, UK. It maps all crimes
      <p>Users can view crimes on a map, view statistics about crimes, report a crim
    </div>
    <div class="clearfix"></div>
  </div>
</div>
<div class="row">

```

Figure 14 - HTML code of 'about' page

Although this code has created a good layout for our needs, additional CSS had to be written in order to make it appear properly and look visually satisfiable. Following figure shows the final appearance of this page.

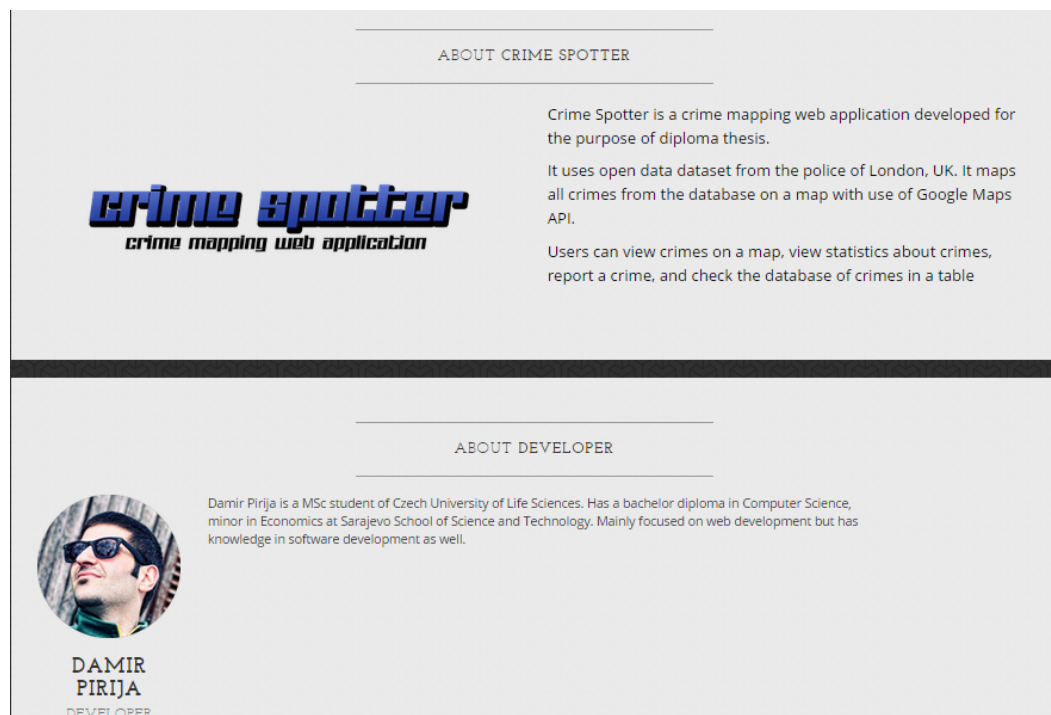


Figure 15 - Display of 'about' page

Code that was previously shown creates the “About Crime Spotter” section of the about page, but the other section “About Developer” is done in a similar manner.

4.7.2 Contact Page

This page was developed second, in order to provide simple contact form for the visitors of Crime Spotter. The following code presents a snippet of a contact form.

```
<form role="form">
  <div class="row">
    <div class="form-group col-lg-4">
      <label>Name</label>
      <input type="text" class="form-control">
    </div>
    <div class="form-group col-lg-4">
      <label>Email Address</label>
      <input type="email" class="form-control">
    </div>
    <div class="form-group col-lg-4">
      <label>Phone Number</label>
      <input type="tel" class="form-control">
    </div>
    <div class="clearfix"></div>
    <div class="form-group col-lg-12">
      <label>Message</label>
      <textarea class="form-control" rows="6"></textarea>
    </div>
    <div class="form-group col-lg-12">
      <input type="hidden" name="save" value="contact">
      <button type="submit" class="btn btn-default">Submit</button>
    </div>
  </div>
</form>
```

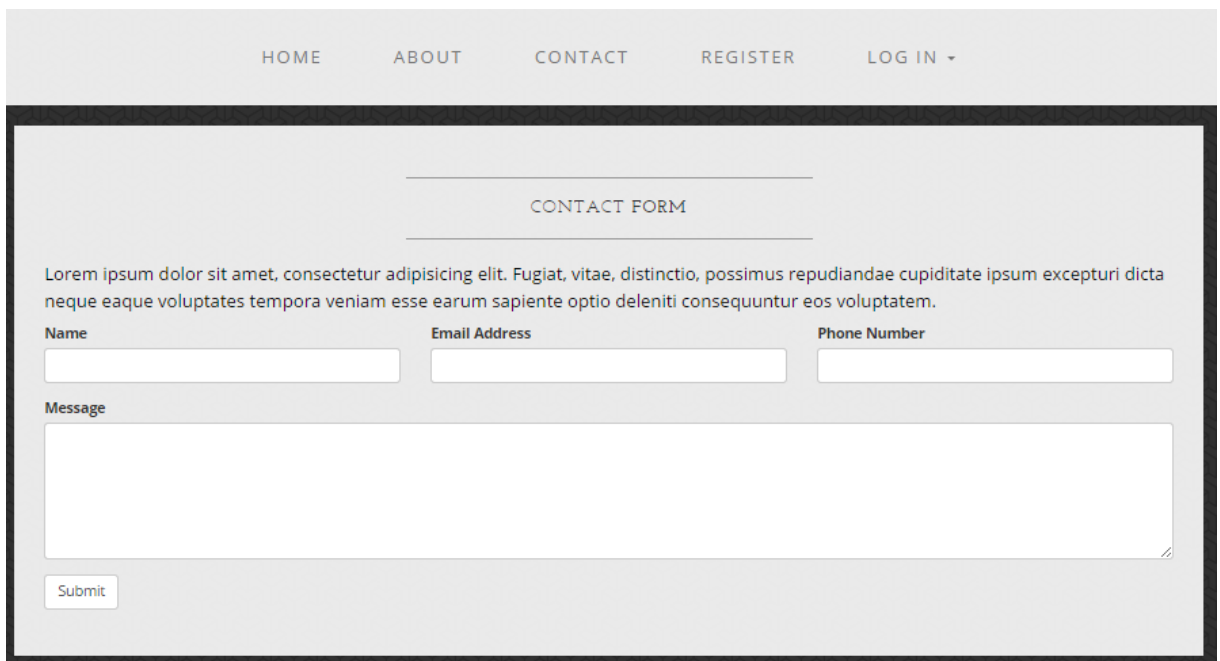
Figure 16 - HTML code of 'contact' page

As it can be seen, again Bootstrap classes were used, such as “form-group col-lg-4” which lay out HTML elements in a nice way and structure, depending on the used width of element. For the purpose of constructing the contact form, it has been decided to include input fields for:

- Name – in order to identify the sender of a message
- Email – in order to be able to reply to the sender
- Phone Number – Is not required, but could be used to reply depending on visitor’s wishes

- Message – Input field which contains the actual message from the visitor

Following figure shows how finished contact page looks like, after the CSS has been written as well.



HOME ABOUT CONTACT REGISTER LOG IN ▾

CONTACT FORM

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Fugiat, vitae, distinctio, possimus repudiandae cupiditate ipsum excepturi dicta neque eaque voluptates tempora veniam esse earum sapiente optio deleniti consequuntur eos voluptatem.

Name

Email Address

Phone Number

Message

Submit

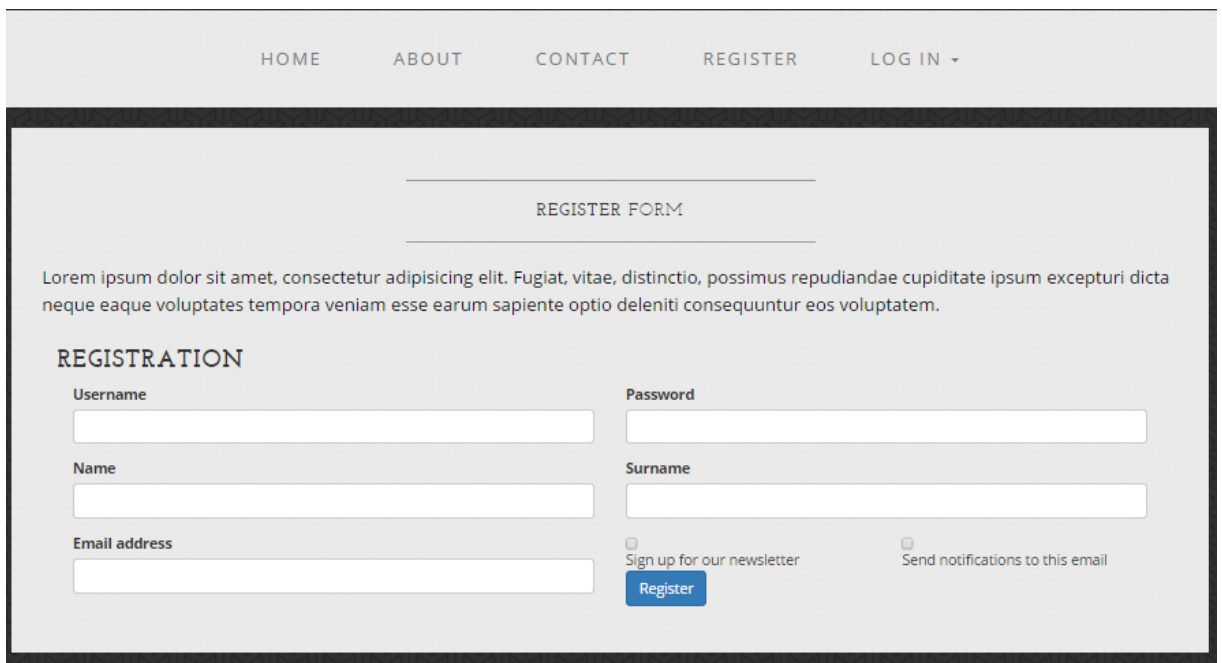
Figure 17 - Display of 'contact' page

After the user submits a contact form, the message and other variables will arrive on administrator's email account in form of an email message.

4.7.3 Registration and Login Pages

User has an option to register to Crime Spotter in order to unlock additional functionalities. In order to register, a registration page has been created. User is presented with a registration form, which, when filled properly, checks whether the user already exists. If not, the user is registered and can proceed to log in. The HTML structure of the registration page was done similarly as in case of contact form, but it performs additional checks with PHP, which will be described later, in the description of back end.

The following figure shows the final view of registration page.

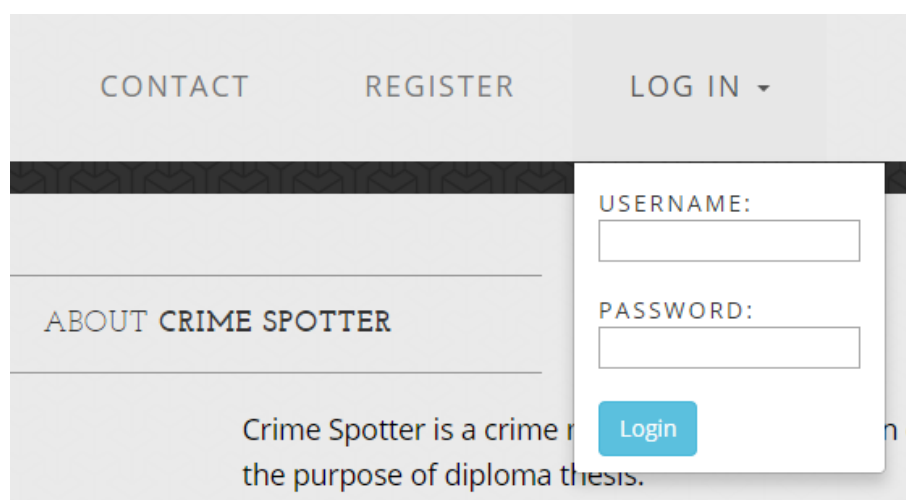


The screenshot shows a web page with a navigation bar at the top containing links for HOME, ABOUT, CONTACT, REGISTER, and LOG IN. The main content area is titled "REGISTER FORM" and contains a paragraph of placeholder text. Below the text is a "REGISTRATION" section with several input fields: Username, Password, Name, Surname, and Email address. There are also two checkboxes: "Sign up for our newsletter" and "Send notifications to this email". A blue "Register" button is positioned below the input fields.

Figure 18 - Display of registration form

As it can be seen, it looks very similar to contact form, with the addition of input fields for Username and Password. The visitor also has an option of checking two checkboxes, in case he or she wants to receive updates and notification to their email.

It was not required for the login form to be placed on a separate page, but it was done as a dropdown button to provide quick and easy login process for users.



The screenshot shows a web page with a navigation bar at the top containing links for CONTACT, REGISTER, and LOG IN. The main content area is titled "ABOUT CRIME SPOTTER" and contains a paragraph of placeholder text. A login window is displayed over the content, featuring input fields for USERNAME and PASSWORD, and a blue "Login" button.

Figure 19 - Display of login window

4.7.4 Main Page – Index

The most important section of Crime Spotter is the main page, where all the main functionality has been implemented. Because Crime Spotter is a mapping web application which uses a Google Maps API, a section for the map was created in HTML, but the actual implementation of the map was done using JavaScript.

```

<script type="text/javascript"
  src="https://maps.googleapis.com/maps/api/js?key=AIzaSyBC81mpNNMBzeTrxJYJqgnBhVwHAfr1o04">
</script>
<script src="https://code.jquery.com/jquery-1.11.1.min.js"></script>

<script type="text/javascript">

var map;

function initialize() {

// Giving the map some options
var mapOptions = {
  zoom: 15,
  center: new google.maps.LatLng(51.5155932, -0.0925699)
};

// Creating the map
map = new google.maps.Map(document.getElementById('map-canvas'), mapOptions);
var crime= <?php echo $crimearray; ?>

```

Figure 20 - JavaScript code for Google Maps initialization

JavaScript code from the figure above is generating a map and placing it inside an HTML element called “map-canvas”. Parameters defined in mapOptions describe default settings of the map.

We can see how it looks like in the next figure.

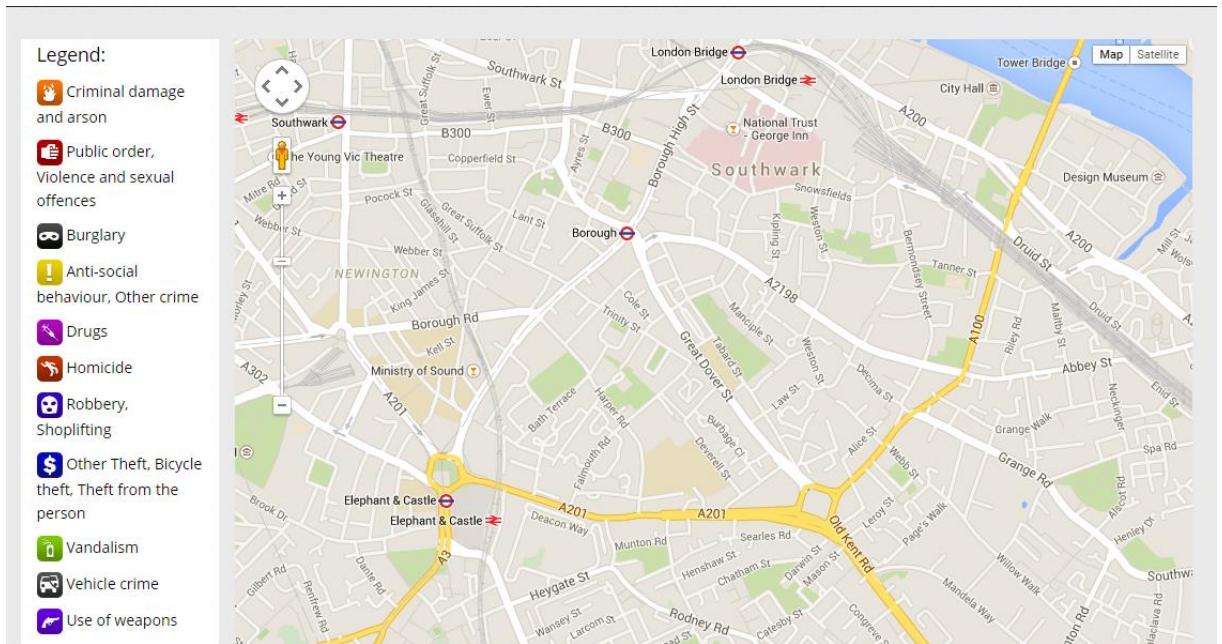


Figure 21 - Display of implemented Google Maps

As it can be seen, an HTML section was created next to the map, in order to provide meaning of all icons which will represent crimes as markers on the map. This will be done in backend, using PHP. For now, we can show how crime types are associated with different icons. This has been done using JavaScript as well.

```

var icon="";
switch (obj.Type) {
  case "Burglary":
    icon = "burglary";
    break;
  case "Criminal damage and arson":
    icon = "arson";
    break;
  case "Other Theft", "Bicycle theft", "Theft from the person":
    icon = "theft";
    break;
  case "Anti-social behaviour", "Other crime":
    icon = "disturbing";
    break;

  case "Public order", "Violence and sexual offences":
    icon = "assault";
    break;
  case "Shoplifting", "Robbery":
    icon = "robbery";
    break;
  case "Vehicle crime":
    icon = "vehicle";
    break;
  case "Drugs":
    icon = "drugs";
    break;
}

var image = "img/markers/" + icon + ".png";

```

Figure 22 - JavaScript code for linking icons to crime types

Next, it was necessary to create a function which will add new markers, based on their longitude and latitude attributes. Also, an icon attribute and title attributes were defined for each marker. Then, we have created an info window which shows detailed information about each crime in a small window, which is shown when a marker is clicked on.

The figure in the next page shows how this is done using JavaScript and Google Maps API.

```

// Adding a new marker for the object
var marker = new google.maps.Marker({
  position: new google.maps.LatLng(obj.Latitude,obj.Longitude),
  map: map,
  title: "Crime Spotted", // this works, giving the marker a title with the correct title
  icon: image
});
google.maps.event.addListener(marker, 'click', function() {
  infowindow.open(map,marker);
});

// Adding a new info window for the object
var clicker = addClicker(marker, contentString);

} // end loop

// Adding a new click event listener for the object
function addClicker(marker, content) {
  google.maps.event.addListener(marker, 'click', function() {

    if (infowindow) {infowindow.close();}
    infowindow = new google.maps.InfoWindow({content: content});
    infowindow.open(map, marker);

  });
}

```

Figure 23 - JavaScript code for adding markers to the map

Next section that was developed, contains pie and bar charts, which shows some statistics retrieved from the database. Charts were created using Google Visualisation API. The code that creates them is shown in the following figure.

```

<script type="text/javascript" src="https://www.google.com/jsapi"></scrip
<script type="text/javascript">

// Load the Visualization API and the piechart package.
google.load('visualization', '1.0', {'packages':['corechart']});

// Set a callback to run when the Google Visualization API is loaded.
google.setOnLoadCallback(drawChart);

// Callback that creates and populates a data table,
// instantiates the pie chart, passes in the data and
// draws it.
function drawChart() {

// Create the data table.
var data = new google.visualization.DataTable();
data.addColumn('string', 'Type of crime');
data.addColumn('number', 'Number of crimes for 2013-12');
data.addColumn('number', 'Number of crimes for 2014-12');
data.addRows([
['Burglary', <?php echo getTypeCount($db, 'Burglary', '2013-12');?>
['Other theft', <?php echo getTypeCount($db, 'Other theft', '2013-1
['Anti-social behaviour', <?php echo getTypeCount($db, 'Anti-social
['Bicycle theft', <?php echo getTypeCount($db, 'Bicycle theft', '20

```

Figure 24 - JavaScript code for initialization of charts

After the data table has been created, options for charts needed to be defined, and charts needed to be instantiated. This was done using the following code.

```

// Set chart options
var optionspie = {'title':'Crimes in December 2014',
                 'width':500,
                 'height':450};
var optionsbar = {'title':'Comparison of crime numbers in 2013-12 and 2014-12',
                 'width':500,
                 'height':450};

// Instantiate and draw our chart, passing in some options.
var piechart = new google.visualization.PieChart(document.getElementById('chart_div'));
piechart.draw(data2, optionspie);

var barchart = new google.visualization.BarChart(document.getElementById('barchart_div'));
barchart.draw(data, optionsbar);
}
</script>

```

Figure 25 - Options and instantiation of charts

As it can be seen from the figure above, pie chart and bar chart are mapped to HTML elements with chart_div and barchart_div ID values. The width and height of the charts was defined directly in JavaScript.

The final view of charts sections is shown in the following figure.

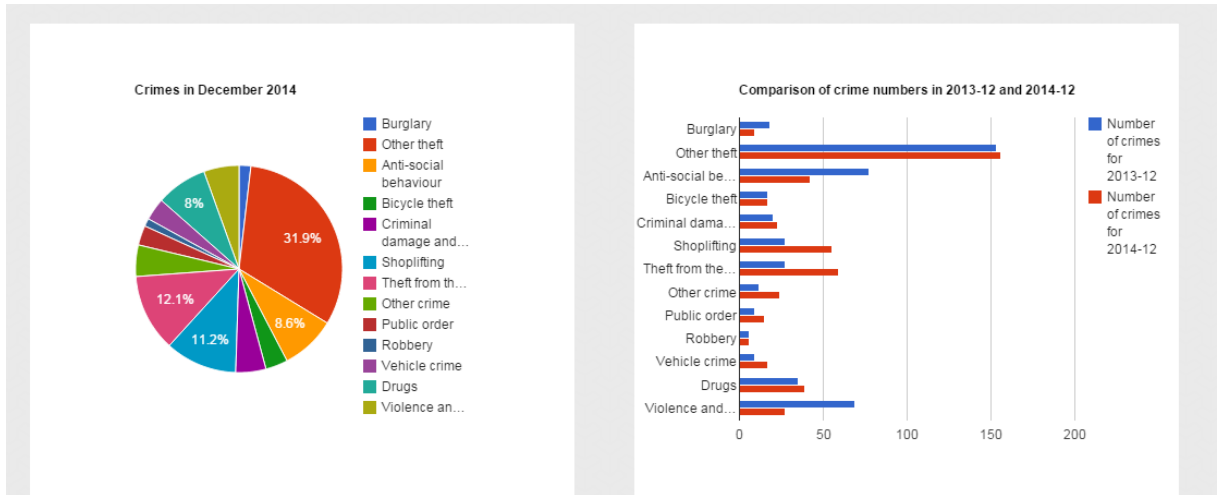


Figure 26 - Final display of created charts

In order to retrieve the actual number of crimes during a certain period, a PHP function was created to serve as communication channel between the database and the web application. This function will be explained later, in the back end section.

Third section of the main page was reserved for showing a table, which is populated with the information from the database. For now, only the final view of this section will be shown, as it will be further described in details in the back end section of this diploma thesis.

Month	Reported by	Falls within	Longitude	Latitude	Location	Type	Outcome
2014-12	City of London Police	City of London Police	-0.11149	51.5182	On or near Pedestrian Subway	Burglary	Under investigation
2014-12	City of London Police	City of London Police	-0.11149	51.5182	On or near Pedestrian Subway	Other theft	Under investigation
2014-12	City of London Police	City of London Police	-0.0976	51.5207	On or near Carthusian Street	Anti-social behaviour	
2014-12	City of London Police	City of London Police	-0.09864	51.5171	On or near Little Britain	Bicycle theft	Under investigation

Figure 27 - Display of table section

User has a possibility of filtering the table based on user input. It is shown in the next figure.

Filter	burg						
Month	Reported by	Falls within	Longitude	Latitude	Location	Type	Outcome
2014-12	City of London Police	City of London Police	-0.11149	51.5182	On or near Pedestrian Subway	Burglary	Under investigation
2014-12	City of London Police	City of London Police	-0.09446	51.5219	On or near Golden Lane	Burglary	Under investigation
2014-12	City of London Police	City of London Police	-0.0783	51.5181	On or near Widgegate Street	Burglary	Under investigation
2014-12	City of London Police	City of London Police	-0.09446	51.513	On or near Watling Court	Burglary	Awaiting court outcome

Figure 28 - Display of table filtering

By typing “burg” inside the input field, the table was automatically filtered to show rows where that string has been found. In our case, result is all fields which contain “Burglary” as type of crime.

This is done by a small JavaScript function which is shown in the following figure.

```

<script>
$(document).ready(function () {
  (function ($) {
    $('#filter').keyup(function () {
      var rex = new RegExp($(this).val(), 'i');
      $('.searchable tr').hide();
      $('.searchable tr').filter(function () {
        return rex.test($(this).text());
      }).show();
    });
  })(jQuery);
});
</script>

```

Figure 29 - JavaScript code for table filter

4.8 Back End

During the development of back end for Crime Spotter, several functionalities needed to be developed in order to provide meaning for the front end elements that were created. Server-side scripting language, PHP was used to create connection between the web application and the database that was developed. Furthermore, PHP was used to communicate with the database regarding the retrieval of data from the database, and also insertion of data such as data created by registration form, or form for reporting a crime. First, the back end that was done for needs of the main page (index page) will be described, and later on, other aspects of the web application, such as registration and login procedures will be explained in detail.

4.8.1 Main Page

As it was shown in previous section which described front end work that has been done, main page consists of three smaller sections.

- Map section
- Charts section
- Table section

All three of these section needed an underlying back end code in order to function properly. In order to start retrieving the data from the database, a connection with the database needed to be established. The connection was created with the use of PHP. The following figure represents the code that achieved this task.

```
// These variables define the connection information for your MySQL database
$username = "root";
$password = "";
$host = "localhost";
$dbname = "crimespotter";

$options = array(PDO::MYSQL_ATTR_INIT_COMMAND => 'SET NAMES utf8');
try { $db = new PDO("mysql:host={$host};dbname={$dbname};charset=utf8", $username, $password, $options); }
catch(PDOException $ex){ die("Failed to connect to the database: " . $ex->getMessage());}
$db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
$db->setAttribute(PDO::ATTR_DEFAULT_FETCH_MODE, PDO::FETCH_ASSOC);
header('Content-Type: text/html; charset=utf-8');
session_start();
```

Figure 30 - PHP connection to the database

First, authentication attributes were defined, such as username, password, host and dbname (name of the database). Then, using the PDO function of PHP, we have tried to establish a connection to the MySQL database using the mentioned attributes to authenticate. If the authentication failed, a web application shows an error message. If not, we can continue communication between Crime Spotter and the database.

Next section to be developed is the map section. Using the Google Maps API and JavaScript, we have already placed an empty map with default options inside the web application. The icons that are connected to each type of crime were already defined, but now, it was necessary to retrieve all these crimes from the database. Using PHP, and a SQL query, all rows from the crime table have been retrieved, and placed inside a JSON array. The reason why all rows were placed inside a JSON array is because Google Maps API can take JSON as input data, which will be shown on the map. This was done by the following code.

```
$statement=$db->prepare("SELECT * FROM crime");  
$statement->execute();  
$results=$statement->fetchAll(PDO::FETCH_ASSOC);  
  
$crimearray=json_encode($results);
```

Figure 31 - Query for retrieval of all crimes from the database

In this case, \$db attribute that is used was already defined in the code that handles connection to the database. Here, that attribute is just used again, instead of defining the connection again, for this, and all consecutive queries. Now, \$crimearray attribute holds our JSON array of query results.

```

var crime= <?php echo $crimearray; ?>
// Looping through all the entries from the JSON data
for(var i = 0; i < crime.length; i++) {

    // Current object
    var obj = crime[i];

    var contentString = '<div id="content">'+
        '<div id="siteNotice">'+
        '</div>'+
        '<h1 id="firstHeading" class="firstHeading">'+obj.Type+'</h1>'+
        '<div id="bodyContent">'+
        '<p><b>Month: </b>'+obj.Month+' </p>'+
        '<p><b>Reported by: </b>'+obj.Reportedby+' </p>'+
        '<p><b>Falls within: </b>'+obj.Fallswithin+' </p>'+
        '<p><b>Location: </b>'+obj.Location+' </p>'+
        '<p><b>Outcome: </b>'+obj.Outcome+' </p>'+

        '</div>'+
        '</div>';

    var infowindow = new google.maps.InfoWindow({
        content: contentString
    });
}

```

Figure 32 - Populating information window from the query results

The above figure shows the JavaScript code that was used. First, we copy the PHP attribute \$crimearray into the JavaScript variable. Then, a loop was established which loops through all rows of the query results, and maps each database column to a certain part of info window, which will be shown when marker on the map is clicked on. When this was done, our final map has been created, and is ready to be used by the visitor of Crime Spotter.

As it can be seen in the following figure, our map section contains a map filled with markers, and each marker is presented in a form of an icon which changes depending on the type of crime committed. In this case, we have clicked on a small red icon with fist, which presented an info window with more details about the crime. The window can be closed at any time, and the map is fully interactive.

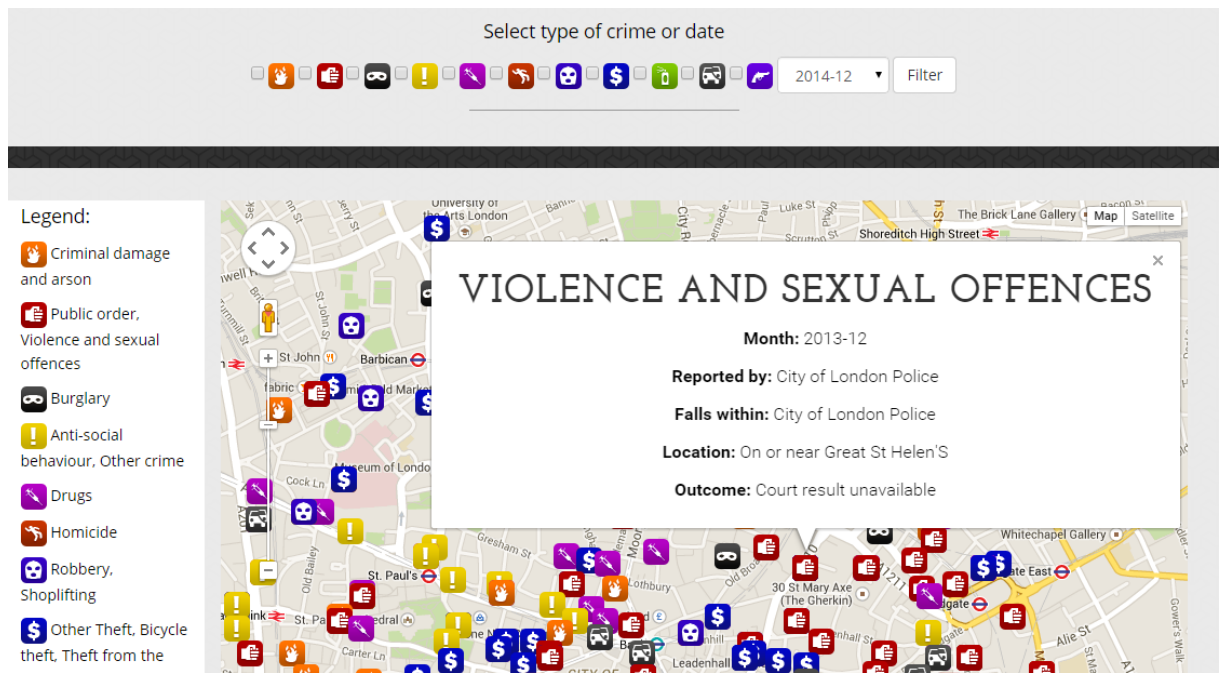


Figure 33 - Final view of map section filled with markers and information window

Furthermore, above the map is a filter section, in which it is possible to select a few checkboxes, and a date, to filter the map to show only the markers which interest us. For the purpose of showing how the filter actually works, we have decided to filter the map to show only crimes related to drugs, and thefts such as bicycle theft, theft from the person, or other types of thefts. Next figure shows how map has changed based on the filter.

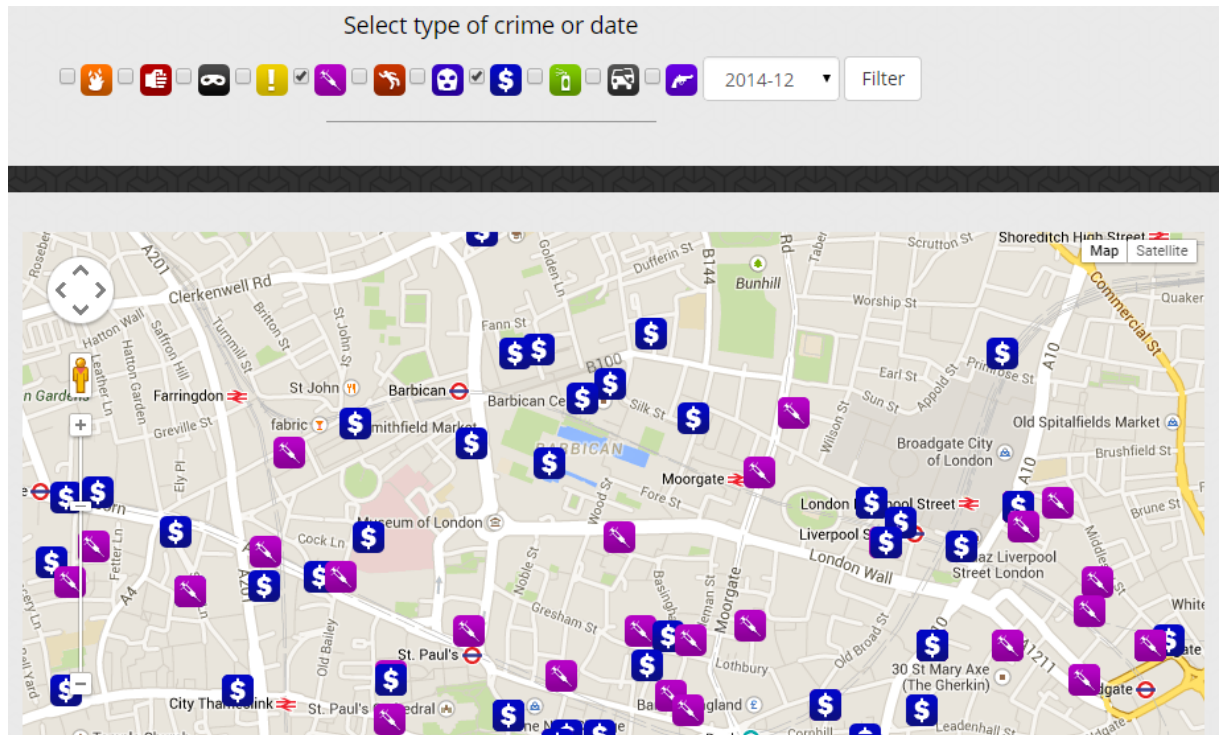


Figure 34 - Map filtering based on crime type and date

Once the complete functionality for map section was created, the development of next sections of main page could continue.

Now, we need to retrieve the data from the database, which charts will use to show correct results. As it was shown in front end description, pie and graph charts have been placed on the page. Pie chart shows in what percentage crimes of certain type have occurred, in comparison to the total number of crimes for that month. To achieve this, it was necessary to write a function, which will count the number of crimes for each type from the database, and return it to the web application.

This was done by creating a PHP function which will handle input and output.

```

function getTypeCount(PDO $pdo, $type, $month)
{
    global $statement;
    $statement = $pdo->prepare(
        "SELECT COUNT(Type) AS CNT FROM `crime` WHERE Type=:type AND Month=:month LIMIT 1");
    $statement->bindParam(':type', $type, PDO::PARAM_INT);
    $statement->bindParam(':month', $month, PDO::PARAM_INT);
    // get the result resource from the database
    $statement->execute();
    $obj = $statement->fetchObject();
    return $obj->CNT;
}

```

Figure 35 - PHP function which returns number of crimes of certain type

The function called `getTypeCount` takes input in form of database connection, type of crime, and month when the crime occurred. It counts crimes of certain type that were committed, and returns a result in form of integer.

How this is used in the JavaScript code written for the Google Visualisation API, is shown in the following figure.

```

data2.addRows([
    ['Burglary', <?php echo getTypeCount($db, 'Burglary', '2014-12');?>],
    ['Other theft', <?php echo getTypeCount($db, 'Other theft', '2014-12');?>],
    ['Anti-social behaviour', <?php echo getTypeCount($db, 'Anti-social behaviour', '2014-12');?>],
    ['Bicycle theft', <?php echo getTypeCount($db, 'Bicycle theft', '2014-12');?>],
    ['Criminal damage and arson', <?php echo getTypeCount($db, 'Criminal damage and arson', '2014-12');?>],
    ['Shoplifting', <?php echo getTypeCount($db, 'Shoplifting', '2014-12');?>],
    ['Theft from the person', <?php echo getTypeCount($db, 'Theft from the person', '2014-12');?>],
    ['Other crime', <?php echo getTypeCount($db, 'Other crime', '2014-12');?>],
    ['Public order', <?php echo getTypeCount($db, 'Public order', '2014-12');?>],
    ['Robbery', <?php echo getTypeCount($db, 'Robbery', '2014-12');?>],
    ['Vehicle crime', <?php echo getTypeCount($db, 'Vehicle crime', '2014-12');?>],
    ['Drugs', <?php echo getTypeCount($db, 'Drugs', '2014-12');?>],
    ['Violence and sexual offences', <?php echo getTypeCount($db, 'Violence and sexual offences', '2014-12');?>]
]);

```

Figure 36 - Adding data to charts

As it can be seen, we have listed all crime types which will be shown on the pie chart, and then we retrieve the total number of crimes for the appropriate type and month, using the previously shown function. This code has filled previously generated pie chart with the corresponding data. The final view of pie chart is shown in the next figure.

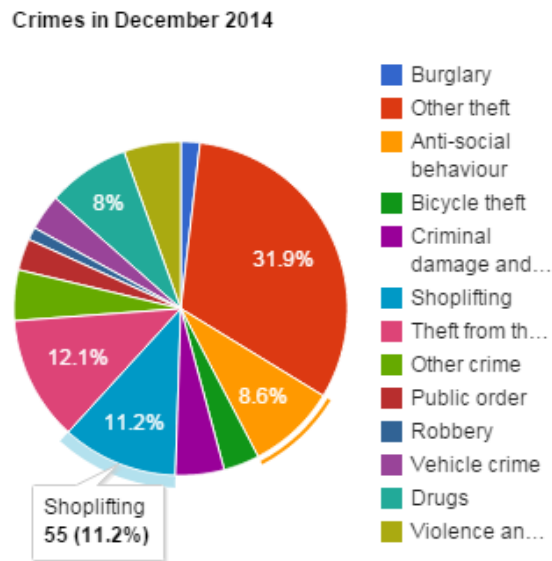


Figure 37 - Pie chart filled with data from the database

As it is shown, when highlighting a certain portion of the pie chart, we get information on the percentage of crimes that occurred for type “Shoplifting” in comparison to the total amount of crimes. Also, the PHP function that was written returns the number of 55, for shoplifting type and the date of December, 2014.

Next chart that was created was bar chart. It uses the same function as the pie chart, but now we define it two times, for two different months, in order to provide the comparison of crime count between two months. The bar chart is presented in the following figure.

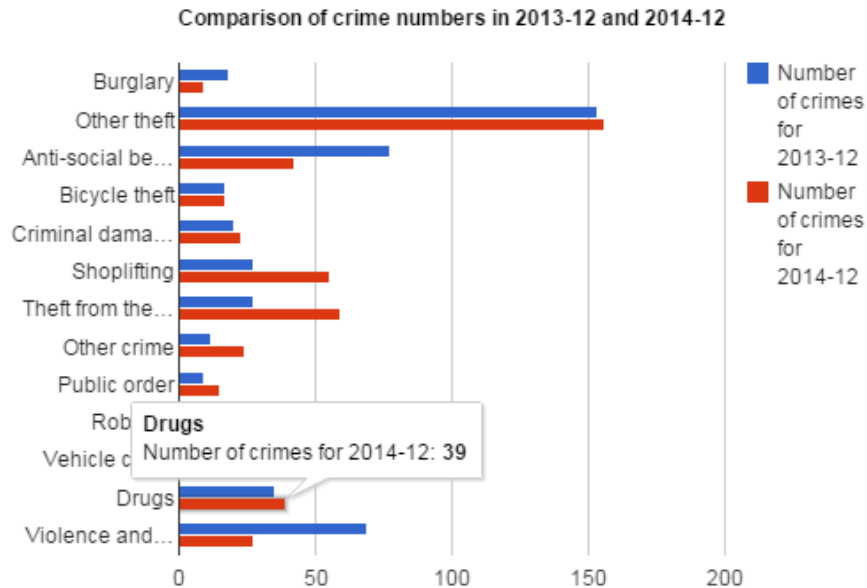


Figure 38 - Bar chart filled with data from the database

From the chart, we can deduce several conclusions regarding our data. For example, we can see that crimes of type “Shoplifting” were almost doubled in December, 2014, as opposed those in December, 2013. Furthermore, if we place the mouse cursor on any coloured bar, we can see the exact number of crimes that happened. In this case, it is visible that 39 crimes of type “Drugs” have occurred in December, 2014. If more data was introduced, including data for other months, the chart would be more detailed, and we could see the increase and decrease in crime across all months, which would make place for drawing many different conclusions from the data.

Third section of the main page is the table section, where all the data about crimes is retrieved and stored in a table, the same as it is retrieved for the purpose of mapping crimes in the first section. The final view of the table section was previously shown in the front end section of this document, and it can be seen on the following figure.

Month	Reported by	Falls within	Longitude	Latitude	Location	Type	Outcome
2014-12	City of London Police	City of London Police	-0.11149	51.5182	On or near Pedestrian Subway	Burglary	Under investigation
2014-12	City of London Police	City of London Police	-0.11149	51.5182	On or near Pedestrian Subway	Other theft	Under investigation
2014-12	City of London Police	City of London Police	-0.0976	51.5207	On or near Carthusian Street	Anti-social behaviour	
2014-12	City of London Police	City of London Police	-0.09864	51.5171	On or near Little Britain	Bicycle theft	Under investigation

Figure 39 - Table section of the main page

This was done by placing data from each column of the results into the appropriate column in the table. First, the query is performed in order to get all rows.

```
<?php
$sql= "SELECT * FROM crime";
$stmt = $db->prepare($sql);
$stmt->execute();
$result = $stmt->fetchAll();
?>
```

Figure 40 - Query for populating the table

After the results are fetched and stored in the \$result variable, each result column is mapped to the table.

```
<tr>
  <th data-field="month" data-align="center" data-sortable="true">Month</th>
  <th data-field="reportedby" data-align="center" data-sortable="true">Reported by</th>
  <th data-field="fallswithin" data-align="center" data-sortable="true">Falls within</th>
  <th data-field="longitude" data-align="center" data-sortable="true">Longitude</th>
  <th data-field="latitude" data-align="center" data-sortable="true">Latitude</th>
  <th data-field="location" data-align="center" data-sortable="true">Location</th>
  <th data-field="type" data-align="center" data-sortable="true">Type</th>
  <th data-field="outcome" data-align="center" data-sortable="true">Outcome</th>
</tr>
</thead>
<tbody class="searchable">
  <?php
  foreach($result as $row){
  echo "<tr>";
  echo "<td>{$row['Month']}</td>";
  echo "<td>{$row['Reportedby']}</td>";
  echo "<td>{$row['Fallswithin']}</td>";
  echo "<td>{$row['Longitude']}</td>";
  echo "<td>{$row['Latitude']}</td>";
  echo "<td>{$row['Location']}</td>";
  echo "<td>{$row['Type']}</td>";
  echo "<td>{$row['Outcome']}</td>";
```

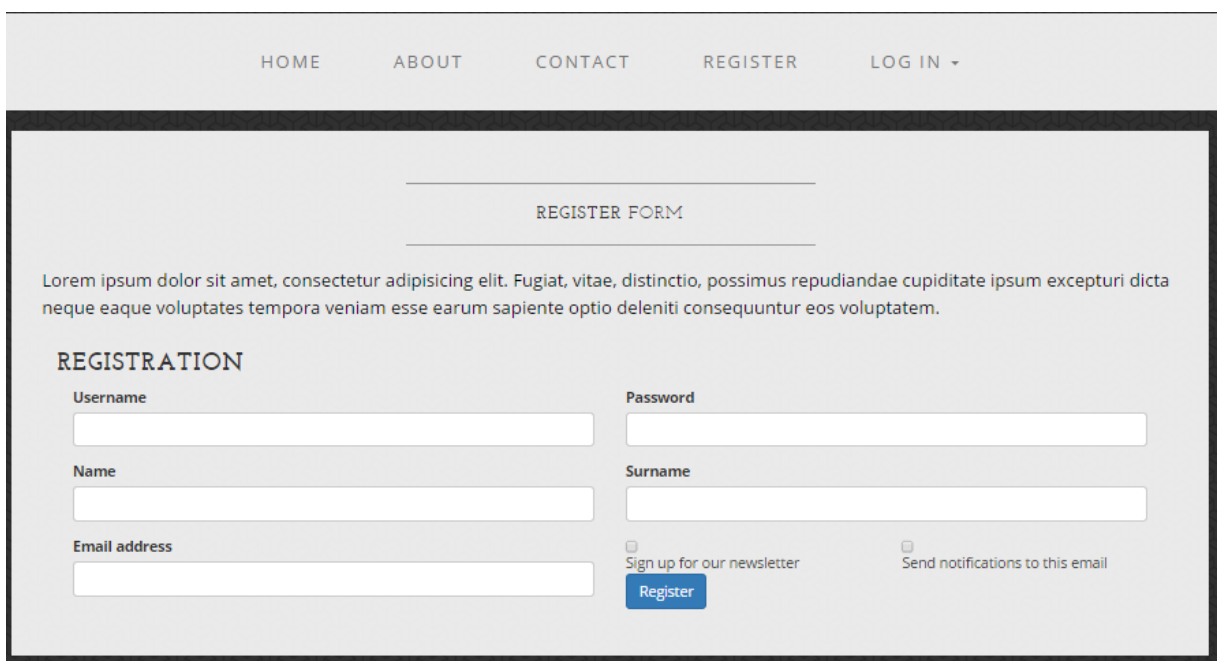
Figure 41 - HTML code for table creation and field population

Once this code was written, our table was populated with the appropriate data. Now, visitor of Crime Spotter can view the data, and filter it according to his or her preferences.

4.8.2 Registration and Login

The primary functionality of Crime Spotter is available to all its visitors, to view crimes, and statistics regarding the crimes that have been committed. However, users should have the ability to report a crime, soon after it happens, so that the database of crimes stays up-to-date. One possibility was to make this functionality open for all visitors, but there are a few possible problems that could rise out of making this option available. One of the problems is that anyone could anonymously report a crime at any location, even though it did not even occur. A solution for this problem is to make a registration and login procedures which would allow only registered users to report a crime, so it was decided to implement it in Crime Spotter.

First, a registration page was created that would take the input from the user, regarding the account details. The view that user sees is shown in the following picture.



The image shows a web browser window displaying a registration form. At the top, there is a navigation bar with links for HOME, ABOUT, CONTACT, REGISTER, and LOG IN. The main content area is titled "REGISTER FORM" and contains a placeholder text: "Lorem ipsum dolor sit amet, consectetur adipisicing elit. Fugiat, vitae, distinctio, possimus repudiandae cupiditate ipsum excepturi dicta neque eaque voluptates tempora veniam esse earum sapiente optio deleniti consequuntur eos voluptatem." Below this, the "REGISTRATION" section includes input fields for Username, Password, Name, Surname, and Email address. There are also two checkboxes: "Sign up for our newsletter" and "Send notifications to this email". A blue "Register" button is positioned at the bottom right of the form.

Figure 42 - Registration form

After the visitor clicks on the button to proceed with registration, several back end processes are done in order to check whether the user with the same account details already exists, and if not, to create a new account with entered account details. Following figure shows a part of HTML code that handles the registration form.

```
<form action="register.php" method="post">
<div class="form-group col-lg-6">
  <label>Username</label>
  <input type="text" name="username" class="form-control" id="" value="">
</div>

<div class="form-group col-lg-6">
  <label>Password</label>
  <input type="password" name="password" class="form-control" id="" value="">
</div>

<div class="form-group col-lg-6">
  <label>Name</label>
  <input type="text" name="name" class="form-control" id="" value="">
</div>
```

Figure 43 - HTML code for the registration form

As it can be seen, the form calls action from “register.php”. This file contains PHP code that handles form validation and insertion into the database.

```
if(!empty($_POST))
{
  // Ensure that the user fills out fields
  if(empty($_POST['username']))
  { die("Please enter a username."); }
  if(empty($_POST['password']))
  { die("Please enter a password."); }
  if(empty($_POST['name']))
  { die("Please enter a name."); }
  if(empty($_POST['surname']))
  { die("Please enter a surname."); }
  if(!filter_var($_POST['email'], FILTER_VALIDATE_EMAIL))
  { die("Invalid E-Mail Address"); }
  |
  // Check if the username is already taken
  $query = "
  SELECT
    1
  FROM users
  WHERE
    username = :username
  ";
  $query_params = array( ':username' => $_POST['username'] );
  try {
    $stmt = $db->prepare($query);
    $result = $stmt->execute($query_params);
  }
  catch(PDOException $ex){ die("Failed to run query: " . $ex->getMessage()); }
  $row = $stmt->fetch();
  if($row){ die("This username is already in use"); }
```

Figure 44 - PHP code that handles checking for existing users

The code written above shows the validation process of all input fields, and also checks whether the selected username is already in the database. If it is, the function returns an error message. The same procedure is done for the validation of email address. If a user wants to register an account with different username, but the same email address, the system would return an error message which says that the given email address is already in use. If all checks have not returned any error message, it means that the visitor can be registered, and account details can be inserted into the database. This is done by the following code.

```
// Add row to database
$query = "
    INSERT INTO users (
        username, password, name, surname, salt, email
    ) VALUES (
        :username, :password, :name, :surname, :salt, :email );";

// Security measures
$salt = dechex(mt_rand(0, 2147483647)) . dechex(mt_rand(0, 2147483647));
$password = hash('sha256', $_POST['password'] . $salt);
for($round = 0; $round < 65536; $round++){ $password = hash('sha256', $password . $salt); }
$query_params = array(
    'username' => $_POST['username'],
    'password' => $password,
    'name' => $_POST['name'],
    'surname' => $_POST['surname'],
    'salt' => $salt,
    'email' => $_POST['email']
);
try {
    $stmt = $db->prepare($query);
    $result = $stmt->execute($query_params);
}
catch(PDOException $ex){ die("Failed to run query: " . $ex->getMessage()); }
header("Location: index.php");
die("Redirecting to index.php");
```

Figure 45 - PHP code that handles creation of a new user

It is important to notice that security measures have been implemented for the registration process, in order to protect user from possible security threats. For example, password is hashed with sha256 encryption, which means that database administrator will not be able to view user's password, while inspecting the data in the user table. If the user is successfully registered, he is returned to the index page. Then, he or she can proceed to log in in order to activate additional functionality.

In the previous section, where front end that was created for Crime Spotter was described, a figure was placed which showed the login window. In order to log in, user clicks on “log in” button in the navigation bar, and a dropdown window appears, with input fields for username and password.

A visitor needs to enter correct username and password in order to proceed. Username and password validation is written in a separate file called “login.php”. The following figure shows the code that handles the login procedure.

```

if(!empty($_POST)){
    $query = "
        SELECT
            id,
            username,
            password,
            name,|
            surname,
            salt,
            email
        FROM users
        WHERE
            username = :username
    ";
    $query_params = array(
        ':username' => $_POST['username']
    );
    try{
        $stmt = $db->prepare($query);
        $result = $stmt->execute($query_params);
    }
    catch(PDOException $ex){ die("Failed to run query: " . $ex->getMessage()); }
    $login_ok = false;
    $row = $stmt->fetch();
    if($row){
        $check_password = hash('sha256', $_POST['password'] . $row['salt']);
        for($round = 0; $round < 65536; $round++){
            $check_password = hash('sha256', $check_password . $row['salt']);
        }
        if($check_password === $row['password']){
            $login_ok = true;
        }
    }
}

```

Figure 46 - PHP code that handles login procedure

As it can be seen, the code queries the database in order to find rows, where username and password match the data that was inputted by the visitor. However, in order to compare passwords, first it is necessary to decrypt the encrypted password in the database, and then compare it to the password from the input field. Once the check is performed, user is logged in, or displayed an error message.

```

if($Login_ok && $_POST['username']=='admin'){
    unset($row['salt']);
    unset($row['password']);
    $_SESSION['user'] = $row;
    header("Location: index.php");
    die("Redirecting to: index.php");
}
elseif($Login_ok){
    unset($row['salt']);
    unset($row['password']);
    $_SESSION['user'] = $row;
    header("Location: report.php");
    die("Redirecting to: report.php");
}
else{
    print("Login Failed.");
    $submitted_username = htmlentities($_POST['username'], ENT_QUOTES, 'UTF-8');
}

```

Figure 47 - PHP code that checks the type of user

The code above shows additional function of the login procedure. Because the application makes a distinction between two types of users, regular users and administrator, we have to check which account has the person that is logging in. For demonstration purposes, the only administrator account for the application has “admin” as username. If a visitor has admin account, he has additional function in the application. If it is a normal user, he is redirected to a page dedicated to reporting a crime, after the successful login. If a visitor enters incorrect login information, an error message is displayed.

We have mentioned that there are two types of registered users, regular users and administrator. Following images show different functions available to users, depending on user type.

HOME ABOUT CONTACT REGISTER REPORT LOG OUT

REPORT CRIME

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fugiat, vitae, distinctio, possimus repudiandae cupiditate ipsum excepturi dicta neque eaque voluptates tempora veniam esse earum sapiente optio deleniti consequuntur eos voluptatem.

Month (YYYY-MM)	Reported by	Falls within	Longitude
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
Latitude	Location (approximate)	Type	Outcome
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>

Figure 48 - Display of form for reporting a crime

Once a regular user logs in, he is redirected to a page with a form to report a crime. As it can be seen when compared to the regular navigation bar, there is a difference in given options. The registered user, once logged in, has an additional button “Report” which takes him to the form shown in the above image. It is important to note that from field “Reported by” has to contain valid source of information, in form of a link to the article, or official police website. Only when that field contains a proper source, will the crime be inserted into the database. If the source is missing, or is incorrect, an administrator can consider that report to be invalid, and delete it from the database.

Since these checks of valid report need to be performed, and someone need to review all the submitted reports, we will show what the duties of administrator are. The following figure shows how Crime Spotter’s interface looks like after the administrator has logged in.











Filter	Type here...								
Month	Reported by	Falls within	Longitude	Latitude	Location	Type	Outcome	Approve	Delete
2014-12	City of London Police	City of London Police	-0.11149	51.5182	On or near Pedestrian Subway	Burglary	Under investigation		
2014-12	City of London Police	City of London Police	-0.11149	51.5182	On or near Pedestrian Subway	Other theft	Under investigation		
2014-12	City of London Police	City of London Police	-0.09864	51.5171	On or near Little Britain	Bicycle theft	Under investigation		
2014-12	City of London Police	City of London Police	-0.0976	51.5207	On or near Carthusian Street	Bicycle theft	Under investigation		
2014-12	City of London Police	City of London Police	-0.09591	51.5203	On or near Beech Street	Criminal damage and arson	Under investigation		

Figure 49 - Display of administrator's table for reviewing submitted crimes

As it can be seen, the navigation bar has one additional option for administrators, which is the possibility of viewing submitted reports. They are displayed in the table under the navigation, as shown in the figure above. All of the crimes that are submitted are displayed in this table, and administrator has two buttons in last two columns, to approve or delete the row from the database.

In case of incorrect source of information, improperly filled input fields, or duplicate submission, administrator clicks on the red button and deletes the row. It is important to mention that all these reports are still in the database table of submitted reports, which do

not show live on Crime Spotter until they are approved. So, administrator has an option of approving or deleting reports in batches, or one by one as soon as they appear in the table.

5 RESULTS AND DISCUSSION

As soon as Crime Spotter, the web application has started gaining a final form of its structure and functionality, it was already possible to see the output of this kind of web application. In this case, raw open data, which was supplied by the police institution of United Kingdom, was processed in order to be highly interpretable by the end user.

The main page contains all the main functionalities of Crime Spotter. An interactive map, which was placed in the first section, was fed raw data that was imported into a MySQL database, processed, and stylized in order to improve user experience while using Crime Spotter. The map was created with the use of Google Maps API, and it contains crime that was mapped according to its location. End user was provided with the ability to check criminal activity that surrounds him and his loved ones. Based on these insights, user can choose what to do with this information, and how to use it to gain advantage in decision making process. These decisions come in various forms, such as choosing where to rent or buy a real estate, or through which street to pass on his way back home. Families with children can make a decision on whether a certain neighbourhood is safe for their children to grow up in, or go to school in.

After the map section was successfully developed, a new section was introduced, a section containing charts. As volume of criminal activity changes from month to month, a way to follow these changes was necessary to be implemented. By using Crime Spotter, a person can see how criminal activity has increased or decreased in their place of living. For the purpose of testing the application, a small dataset containing criminal activity data in the city of London has been used, but the database can be expanded with data from all parts of the world. Users can see whether they should increase their security measures regarding any type of property, based on the insights gained from provided charts. Third section that was created displays a table, with all information on criminal activity from the database. If the user is looking for information on specific crime, he can use a built in filter to search through the database of crimes for a particular activity of interest.

After the main page of Crime Spotter has been fully developed, there were more functions to implement. It is not rare that normal people are witnesses to criminal activity on the streets of their city. However, they are not provided with a safe, simple to use, way of reporting that

criminal activity. Yes, they can report it to the police, but often, the public never hears much information on the criminal activity that happened. If a visitor of this application registers, he is provided with a form, in which he can report the location, time, and type of crime that happened. Because the web application was developed with responsiveness in mind, it can be easily visited and used from all handheld devices.

The administrator in this case has a responsibility of reviewing information on reported crimes, and to decide which report has a source reliable enough for the actual crime to be listed on the map, and in the table of crimes. Crime Spotter is a scalable web application. Although it was written with a small dataset, it has been shown what it can do and what insights it can provide.

In order to compare features of Crime Spotter to those of similar existing web applications, two crime mapping platforms have been selected, **CrimeReports™** which is located at <https://www.crimereports.com/> and **CzechCrime** located at <http://www.czechcrime.org/>. It is important to note that these two mapping platforms are developed and maintained by teams of developers, and therefore have a higher set of functionality than **CrimeSpotter**. The following table introduces an overview of main features of each web application.

Feature	CrimeReports™	CzechCrime	CrimeSpotter
Data	Criminal activity for USA, Canada, UK	Criminal activity for Czech Republic	Criminal Activity for London, UK (test dataset)
Filter	Based on type of crime, exact date, time period, particular address, distance from current location, area, department	Based on type of crime, time period, address, crime rate	Based on type of crime, time period
Markers	Supported, show exact location of a particular crime	Not supported	Supported, show exact location of a particular crime

Areas	Not supported	Supported	Not supported
Statistics	Not supported	Supported; Number of crimes per area, number of crimes per type, graphical representation of crime rates, comparison between areas, charts per area	Supported; Number of crime per type, pie chart representing crime count per type in a certain period, bar chart providing crime counts per type in different periods
Other	Possibility of user registration, crime tip submission	No possibility of user registration, possibility of providing feedback.	Possibility of user registration, crime tip submission

Table 4 - Comparison of crime mapping web applications

The table above shows main differences between the three crime mapping web applications being compared. If we focus on **CrimeReports™**, we can see that the main, and only section is the map section, which has markers placed on exact locations where the crimes have happened. It is possible to filter the data by selecting crime type of interest, a certain date, or even a particular police agency. However, what is missing is an in-depth view of criminal activity. Each marker provides basic information on a particular crime, but there is no possibility of comparing overall criminal activity for a certain area or time. **CzechCrime** provides a very detailed user interface, along with the map of criminal activity. A user can find many comparisons, charts, statistics about crime overall, but also for a particular crime type. The map of Czech Republic is divided in smaller sections, and user can get information on number of crimes for a particular section. However, if a user wants to find a certain criminal activity in one of the streets, there is no possibility of that. The reason for this can be a constraint in data usage, in order to protect privacy. Overall, **CzechCrime** is a very good example of crime mapping web application with an in-depth overview of statistical information and trends in criminal activity in Czech Republic. **CrimeSpotter** is using a small dataset only for testing purposes. It provides basic statistical information in form of pie and bar charts. In order to expand a set of features, CrimeSpotter could implement some of the features of **CzechCrime**, similar to area filtering support. This can be useful if datasets from more countries would be included in the database, as displaying markers for every city

in every country increases loading time and requires more processing power. So, dividing the information into larger areas would decrease the time and power required for display of that information, which would lead to better user experience and usability of web application. However, it is important to note that a crime mapping web application, such as CrimeSpotter, is fully dependable on the open data it uses. If the data that is provided is inaccurate, or contains certain constraints, usage of such data will also be under constraint, which means that the end-user will not reap full potential benefit of using it. Furthermore, some countries still do not support publishing of data regarding criminal activity inside the country. This poses a barrier in actual development and usage of the application, because not all citizens from different countries will be able to properly use it. In order to overcome these barriers, it is important to further promote the usage and publishing of open data, as well as educating the society about the potential benefits which could be gained.

6 CONCLUSION

In order to conclude the writing of this diploma thesis, it is important to review what the expected objectives were, and in what measure were they fulfilled. The main aim of this work was to analyse, design, and develop a web application which uses open data. This was done with the development of Crime Spotter, which uses raw open data and displays it in several easy to understand views.

Before the actual development has started, concepts of open data, its benefits and barriers were described in detail. Furthermore, prior to the development of web application, the software and web development life cycles were presented, that would serve as a developers' guide to forming an approach to the development process. As a part of development, the open data that was to be used was analysed and described, in order to understand the dataset before usage. Furthermore, models were created that show the structure and intended functionality of the web application. In order to proceed with the development, several wireframes were designed which have shown the actual design idea of Crime Spotter.

Based on completed analysis and design, Crime Spotter was developed using client-side and server-side technologies. The web application provides a map of criminal activity for a particular location, which can be interpreted and used in many situations. Furthermore, charts were included which show trends in the numbers of criminal activity. Lastly, a table populated with detailed information on criminal activity was provided to be inspected, used and searched based on a particular crime in mind. As a result of production of this sort of web application, it was shown how end users can draw meaningful conclusions from the usage, and how the same application can be applied to provide benefits for small communities, as well as whole countries and continents.

7 BIBLIOGRAPHY

- Chun, S. A., Shulman, S., Sandoval, R. & Hovy, E.**, 2010. Government 2.0: Making. *Information Polity*, Issue 15, pp. 1-9.
- Frain, B.**, 2012. *Responsive web design with HTML5 and CSS3*. vi ed. Birmingham: Packt Publishing.
- Gurin, J., n.d.** *Open data now: the secret to hot startups, smart investing, savvy marketing, and fast innovation*. XVI ed. s.l.:s.n.
- Heeks, R.**, 2006. *Implementing and managing eGovernment: an international text*. IX ed. s.l.:Thousand Oaks.
- Holdener, A. T. I.**, 2008. *AJAX - The Definitive Guide*. 2nd ed. s.l.:O'Reilly Media, Incorporated.
- Janssen, M., Charalabidis, Y. & Zuiderwijk, A.**, 2012. *Benefits, Adoption Barriers and Myths of Open Data and Open Government*. s.l.:Taylor & Francis.
- Lee, T.-B.**, n.d. *Five Star Open Data*. [Online]
Available at: <http://5stardata.info/>
[Accessed 12 February 2015].
- Lockhart, J.**, 2015. *PHP The Right Way*. [Online]
Available at: <http://www.phptherightway.com/>
[Accessed 05 March 2015].
- McFarland, D. S.**, 2013. *CSS3: The Missing Manual*. Third ed. Sebastopol, CA 95472: O'Reilly Media Inc..
- Meeker, M.**, 2010. *Internet Trends 2010 by Morgan Stanley's Mary Meeker*. [Online]
Available at: <http://bbh-labs.com/internet-trends-2010-by-morgan-stanleys-mary-meeker/>
[Accessed 25 February 2015].
- Moore, R.** et al., 2011. *Understanding HTML5: Today's realities and its future market potential*. [Online]
Available at: <http://www.slideshare.net/iLoopMobile/understanding-the-state-of-html-5-and-its-potential>
[Accessed 02 March 2015].
- Network, M. D.**, 2015. *Mozilla Developer Network: Cascading Style Sheets*. [Online]
Available at: <https://developer.mozilla.org/en-US/docs/Web/CSS/Syntax>
[Accessed 25 February 2015].
- Peters, B. G. & Pierre, J.**, 1998. Governance without government? Rethinking public. *Journal of Public Administration and Theory*, Issue 8, pp. 223-243.

Police, U. K., 2015. *data.police.uk*. [Online]

Available at: <http://data.police.uk/about/#general>

[Accessed 10 February 2015].

Socrata Insights, 2014. *Open Data Benchmark Report*. [Online]

Available at: <http://www.socrata.com/2014-open-data-benchmark-report/>

[Accessed 15 January 2015].

Sommerville, I., 2010. *Software Engineering*. s.l.:Addison-Wesley.

Tatroe, K., Lerdorf, R. & Macintyre, P., 2013. *Programming PHP*. 3rd ed. Sebastopol, CA: O'Reilly Media.

The Open Knowledge Foundation, 2012. *The Open Data Handbook*. [Online]

Available at: <http://opendatahandbook.org/>

[Accessed 20 January 2015].

The Open Knowledge Foundation, n.d. *Open Knowledge*. [Online]

Available at: <https://okfn.org/opendata/>

[Accessed 19 January 2015].

Veljković, N., Bogdanović-Dinić, S. & Stoimenov, L., 2014. *Benchmarking open government: An open data perspective*. [Online]

Available at: www.elsevier.com/locate/govinf

[Accessed 08 February 2015].

8 TABLE OF FIGURES

FIGURE 1 - FORMAT OF ONE STAR DATA	17
FIGURE 2 - FORMAT OF FOUR STAR DATA	19
FIGURE 3 - INSPECTION OF FOUR STAR DATA FORMAT	20
FIGURE 4 - CONTENTS OF DATASET FILE	45
FIGURE 5 - USECASE DIAGRAM OF CRIME SPOTTER WEB APPLICATION	46
FIGURE 6 - CLASS DIAGRAM OF CRIME SPOTTER WEB APPLICATION	48
FIGURE 7 - MAP SECTION OF MAIN PAGE OF CRIME SPOTTER	49
FIGURE 8 - CHART SECTION OF CRIME SPOTTER	50
FIGURE 9 - TABLE SECTION OF MAIN PAGE OF CRIME SPOTTER	51
FIGURE 10 - SQL QUERY FOR CREATION OF 'CRIME' TABLE	53
FIGURE 11 - SQL QUERY FOR CREATION OF 'USERS' TABLE	53
FIGURE 12 - SQL QUERY FOR CREATION OF 'SUBMITTEDCRIME' TABLE	54
FIGURE 13 - HEADER SECTION OF CRIME SPOTTER	55
FIGURE 14 - HTML CODE OF 'ABOUT' PAGE	56
FIGURE 15 - DISPLAY OF 'ABOUT' PAGE	56
FIGURE 16 - HTML CODE OF 'CONTACT' PAGE	57
FIGURE 17 - DISPLAY OF 'CONTACT' PAGE	58
FIGURE 18 - DISPLAY OF REGISTRATION FORM	59
FIGURE 19 - DISPLAY OF LOGIN WINDOW	59
FIGURE 20 - JAVASCRIPT CODE FOR GOOGLE MAPS INITIALIZATION	60
FIGURE 21 - DISPLAY OF IMPLEMENTED GOOGLE MAPS	61
FIGURE 22 - JAVASCRIPT CODE FOR LINKING ICONS TO CRIME TYPES	61
FIGURE 23 - JAVASCRIPT CODE FOR ADDING MARKERS TO THE MAP	62
FIGURE 24 - JAVASCRIPT CODE FOR INITIALIZATION OF CHARTS	63
FIGURE 25 - OPTIONS AND INSTANTIATION OF CHARTS	63
FIGURE 26 - FINAL DISPLAY OF CREATED CHARTS	64
FIGURE 27 - DISPLAY OF TABLE SECTION	64
FIGURE 28 - DISPLAY OF TABLE FILTERING	65
FIGURE 29 - JAVASCRIPT CODE FOR TABLE FILTER	65
FIGURE 30 - PHP CONNECTION TO THE DATABASE	66
FIGURE 31 - QUERY FOR RETRIEVAL OF ALL CRIMES FROM THE DATABASE	67
FIGURE 32 - POPULATING INFORMATION WINDOW FROM THE QUERY RESULTS	68
FIGURE 33 - FINAL VIEW OF MAP SECTION FILLED WITH MARKERS AND INFORMATION WINDOW	69
FIGURE 34 - MAP FILTERING BASED ON CRIME TYPE AND DATE	70
	89

FIGURE 35 - PHP FUNCTION WHICH RETURNS NUMBER OF CRIMES OF CERTAIN TYPE	71
FIGURE 36 - ADDING DATA TO CHARTS	71
FIGURE 37 - PIE CHART FILLED WITH DATA FROM THE DATABASE	72
FIGURE 38 - BAR CHART FILLED WITH DATA FROM THE DATABASE	73
FIGURE 39 - TABLE SECTION OF THE MAIN PAGE	74
FIGURE 40 - QUERY FOR POPULATING THE TABLE	74
FIGURE 41 - HTML CODE FOR TABLE CREATION AND FIELD POPULATION	74
FIGURE 42 - REGISTRATION FORM	75
FIGURE 43 - HTML CODE FOR THE REGISTRATION FORM	76
FIGURE 44 - PHP CODE THAT HANDLES CHECKING FOR EXISTING USERS	76
FIGURE 45 - PHP CODE THAT HANDLES CREATION OF A NEW USER	77
FIGURE 46 - PHP CODE THAT HANDLES LOGIN PROCEDURE	78
FIGURE 47 - PHP CODE THAT CHECKS THE TYPE OF USER	79
FIGURE 48 - DISPLAY OF FORM FOR REPORTING A CRIME	79
FIGURE 49 - DISPLAY OF ADMINISTRATOR'S TABLE FOR REVIEWING SUBMITTED CRIMES	80
TABLE 1 - BENEFITS OF OPEN DATA	22
TABLE 2 - BARRIERS OF OPEN DATA	25
TABLE 3 - COLUMNS OF CSV FORMAT OF DATASET	45
TABLE 4 - COMPARISON OF CRIME MAPPING WEB APPLICATIONS	84