OULUN AMMATTIKORKEAKOULU

Dalun Chen

PERSONAL HEALTH RECORD PROTOTYPE

PERSONAL HEALTH RECORD PROTOTYPE

Dalun Chen
Bachelor's Thesis
Spring 2016
Degree Programme in Business
Information Technology
Oulu University of Applied Sciences

ABSTRACT

Oulu University of Applied Sciences
Degree Programme, Business Information Technology

Author: Dalun Chen
Title of Bachelor´s thesis: Personal health record prototype
Supervisor: Pekka Ojala
Term and year of completion: Spring 2016 Number of pages: 74

Cardiovascular disease (CVD), chronic obstructive pulmonary disease (COPD) and diabetes are the most common chronic diseases in the world, in addition to cancer. CVD is a general term for heart and circulatory conditions, e.g. coronary heart disease (CHD) and stroke, whereas COPD is a chronic condition of the lungs including chronic bronchitis and emphysema. CHD, stroke and COPD are globally the top three causes of mortality. Diabetes, on the other hand, affects one in twelve adults in the world and can lead to CVD and other complications. Diabetes is a chronic metabolic condition. 90% of diabetes is type 2 diabetes. The risks of these chronic diseases are often associated with unhealthy diet, sedentary lifestyle, drinking alcohol and smoking.

To combat these diseases, living a healthy lifestyle is as important as adhering to treatment. A healthy lifestyle includes eating a balanced diet, exercising sufficiently, drinking less alcohol and smoking cessation. As a result, it helps to keep the body healthy and thus reduce the risk of chronic diseases and related complications.

eHealth tailored to achieve specific goals has the potential to empower people to live a healthier life. ProWellness Health Solutions, a Finnish IT company which has dedicated itself to chronic disease management system, is endeavoring to build an advanced personal health record application. The application will help people with CVD, COPD and diabetes and others who are at high risk of these diseases to manage their health more effectively and efficiently.

This thesis project was commissioned by the company to create a new prototype for a web-based personal health record application. The prototype should be modern and tailored to a set of specific requirements.

Much effort have been made to deliver the prototype. Prior to implementation, a literature review on the target diseases was conducted. In addition, self-study on JavaScript and jQuery was carried out. Furthermore, proactive and regular communication with the commissioner and the instructing teacher had been kept throughout the thesis work to ensure that the project advances.

Finally, a new prototype was created according to the requirements. It is responsive and interactive; it presents demo data with configurable charts and sortable tables and can leverage the company's technology. The prototype serves as a medium for obtaining further requirements from the company's stakeholders. A multidisciplinary approach will be integral to future development. As for the future bachelor thesis, a usability study on well-received personal health applications can provide valuable insights.

Keywords: chronic disease, healthy lifestyle, personal health record, prototype, visualization

CONTENTS

1 INTRODUCTION

For Pat, Sirpa and Jim, cardiovascular disease, chronic obstructive pulmonary disease (COPD) and diabetes have caused inconveniences in life and placed obstacles to self-fulfillment. Pat had a heart attack in 2012. Before that, it had never come to her mind that she would one day be plagued with a heart attack, a type of cardiovascular disease. In order to cope with the illness, she quit smoking and began to take medication, pay attention to her diet and exercise regularly. (Titaania suonessa 18.7.2013, cited 19.5.2015.) Sirpa, instead of having a heart attack, has been living with COPD for 13 years. COPD makes her cough and breathe with difficulty. She has quit smoking and changed her lifestyle – included giving up dancing, once her hobby. (Yle 5.6.2012, cited 19.5.2015.) Jim, on the other hand, has type 2 diabetes. A heel injury almost caused him to lose his leg. Luckily, after spending seven weeks in the hospital, he did not lose his leg. However, the same heel was infected again, and the infection triggered a heart attack. As a result, he was admitted to the hospital again. (Diabetes UK 2016, cited 4.2.2016.) These chronic conditions, in addition to cancer, have been recognized as the most common chronic diseases in the world. They have prohibited tens of millions of people from living their life to the fullest and claim 38 million lives every year. The risks of these diseases are often associated with living an unhealthy lifestyle: eating unhealthy diet, living a sedentary lifestyle, drinking alcohol and smoking. (WHO 2015a, cited 19.5.2015.)

To combat these chronic diseases, living a healthy lifestyle is as important as adhering to treatment. Living a healthy lifestyle reduces patients' risk of developing complications. For people without the conditions, chronic diseases can be postponed if not avoided by living a healthy life. According to WHO (2014a, cited 20.5.2015), eating healthy diet, exercising sufficiently, drinking less alcohol and smoking cessation can "decrease premature deaths by half to two-thirds".

Adopting a healthy lifestyle is, however, easier said than done. Inadequate health literacy, poor health conditions, adverse life circumstances and lack of support can keep people from taking care of themselves (Kousoulis, Patelarou, Shea, Foss, Knutsen, Todorova, Roukova, Portillo, Pumar-Méndez, Mujika, Rogers, Vassilev, Serrano-Gil & Lionis 2014, 11; Korhonen, Seppälä, Järvenpää & Kautiainen 2014, 73). Therefore, healthcare systems are expected to provide patients with more guidance on self-management and to reach out to others at high risk of chronic diseases. However, the resources of healthcare systems seem unlikely to be increased

due to adverse economic conditions. (Kousoulis et al. 2014, 13.) Such a challenge has initiated the studies of eHealth in the context of chronic diseases, and the results are promising: eHealth which is tailored to achieve specific goals and leverages expertise in different fields can empower people to take better care of themselves while alleviating the financial burden on healthcare systems (Wicks, Stamford, Grootenhuis, Haverman & Ahmed 2014, 196).

Hence, in an attempt to help people like Pat, Sirpa and Jim; and others at high risk of chronic diseases manage their health more effectively and efficiently, the commissioner of this thesis, ProWellness Health Solutions is endeavoring to build an advanced eHealth solution. Prior to this endeavor, the company had developed an award-winning chronic disease management system (CDMS). Nowadays, the system is used by healthcare organizations in Finland and in the UK. (ProWellness Health Solutions 2013, cited 4.2.2016; ProWellness Health Solutions 2014, cited 4.2.2016.) By leveraging the expertise in CDMS and exploring different technologies, the company is devising an advanced personal health record application (PHR) for self-management.

The company commissioned this thesis project to create a new PHR prototype. Prior to the thesis project, prototypes for a web-based PHR were created. However, they do not satisfy the new set of requirements obtained by the commissioner. After consideration, the commissioner decided that there is a need to create a new prototype tailored to the new requirements. Therefore, it became the goal of the project.

Many efforts have been made to create the new PHR prototype. A literature review on cardiovascular disease, COPD and diabetes was conducted prior to development phase, in order to appreciate the context of the target diseases and to fully understand the commissioner's requirements. In addition, self-study on JavaScript and jQuery had been carried out before and during the development phase, since implementing interactive webpages was the focus of prototyping. Furthermore, I had maintained proactive and regular communication with the commissioner and the instructing teacher throughout the thesis project to ensure that the prototype was delivered by the end of the development.

The main results are documented in this report. The context of the target diseases is introduced in the second chapter. The main technologies and tools chosen for prototyping are justified and presented in the third chapter. An account of major implemented features and means of testing is given in the fourth chapter. The knowledge gained from the thesis project and the major

characteristics of the delivered prototype are summarized in the fifth chapter. Finally, personal reflection and suggestions are kept in the last chapter.

2 TARGET CHRONIC DISEASES

People with cardiovascular disease, chronic obstructive pulmonary disease and diabetes, as well as people at high risk of those chronic conditions are the target users for the personal health record application which the commissioner is developing. During earlier communication with the commissioner, I had identified the need to equip myself with adequate knowledge of those diseases, in order to better understand the commissioner's requirements and thus deliver a prototype accordingly.

Before the development phase commenced, I had conducted a literature review on the target diseases. The main sources included WHO, national healthcare institutions in the UK and Finland, Finnish official statistics, international and local chronic disease associations, professional articles and online medical libraries. The literature review helped me understand how the target chronic diseases develop in the body and what people can do to control and prevent these diseases. In addition, I had visited the websites of multinational technology companies and online application stores, in order to perceive the current development of personal health applications and the general trend of eHealth.

The results of the literature review are documented in this chapter. It begins with an introduction to each target chronic diseases. The introduction explains how each target chronic disease develops in layman's terms and outlines the prevalence of each disease in Finland, the UK and the world. Since these target diseases share common methods for control and prevention, the shared methods are recorded together after the introduction. Finally, the potential of eHealth for empowering people to take better care of themselves is presented in the end of this chapter.

2.1 Cardiovascular disease

The heart and the vascular system are responsible for supplying oxygen and nutrients to the body through circulation. The heart pumps blood, which carries oxygen and nutrients through arteries to the organs and muscles in the body. Oxygen and nutrients are then exchanged for carbon dioxide and other waste matter from the organs and muscles in smaller blood vessels called capillaries. After the exchange, the blood carries carbon dioxide and other waste matter and

returns to the heart through veins to be pumped again. The continual supply of oxygen and nutrients and removal of carbon dioxide and waste matter are called circulation. The network of blood vessels, where circulation takes place, is termed the vascular system. (NHS choices 2014a, cited 18.7.2015; Johns Hopkins Medicine 2015a, cited 18.7.2015.)

Atherosclerosis and embolism impair the heart and the vascular system thus causing cardiovascular disease. When a build-up of fatty matter like cholesterol thickens and hardens the walls of arteries, the condition is called atherosclerosis. On the other hand, when a blood clot, an air bubble or a foreign substance blocks an artery, it is called an embolism. Both atherosclerosis and embolism will reduce or prohibit the flow of blood and thus damage the organ which cannot receive sufficient oxygen and nutrients. (NHS choices 2014b, cited 18.7.2015; NHS choices 2015a, cited 18.7.2015.) Therefore, if coronary arteries, the arteries supplying blood to the heart are becoming clogged, it will lead to coronary heart disease (CHD). If the coronary arteries are completely blocked, it will cause a heart attack. If carotid arteries, the arteries transporting blood to the brain fail to provide the brain with needed oxygen and nutrients, it will trigger a stroke. If the blood flow to the arms and legs is restricted because of atherosclerosis and embolism, it will result in peripheral arterial disease (PAD). Finally, atherosclerosis could also constitute a risk of developing aortic disease, which weakens the largest artery, the aorta. As a result, it can cause part of the aorta to swell or bulge and possibly lead to a fatal burst. These aforementioned diseases are common medical conditions related to the heart or the vascular system; therefore, they have been categorized under a general term: cardiovascular disease. (NHS choices 2014c, cited 20.7.2015; NHS choices 2014d, cited 20.7.2015; NHS choices 2014e, cited 20.7.2015; Johns Hopkins Medicine 2015b, cited 20.7.2015.)

Cardiovascular disease has been associated with a wide spectrum of risk factors. High blood pressure (hypertension), high cholesterol (hyperlipidemia) and high blood sugar (hyperglycaemia) can cause atherosclerosis and embolism, thus increasing the risk of developing cardiovascular disease. Together with overweight and obesity, they are considered as intermediate risk factors. While these intermediate risk factors can lead directly to the development of cardiovascular disease, they are influenced by a much broader collection of interrelated factors which ranges from individual to societal level. (NHS choices 2014f, cited 29.5.2015; NHS choices 2014g, cited 29.7.2015; NHS choices 2015b, cited 29.7.2015; NHS choices 2015c, cited 29.7.2015; WHO 2015b, cited 21.7.2015.) At the individual level, growing old, being male, having one parent or a sibling affected by CVD, having South Asian or African Caribbean ethnic background,

experiencing stress constantly, eating unhealthily, drinking alcohol excessively, doing little or no exercise and smoking can all add to a person's risk of developing cardiovascular disease (NHS choices 2014f, cited 29.5.2015). At the societal level, rapid unplanned urbanization has been blamed for providing poor healthcare service and for exposing urban residents to health threats; on the other hand, globalization has been blamed for spreading unhealthy lifestyles. In addition to population aging and poverty, they can influence the prevalence of cardiovascular disease in a society. (World Heart Federation 2015, cited 29.7.2015; WHO 2015a, cited 19.5.2015; WHO 2015b, cited 21.7.2015.)

In Finland, cardiovascular disease is still the most frequent cause of mortality, although the number of people whom have died from it has decreased over decades. From 1971 to 1987, the disease used to account for over 50% of national mortality. But the mortality rate caused by CVD has decreased since then. (Statistics Finland 2014b, cited 24.7.2015.) In 2013, CVD constituted 38% of the total mortality in 2013 (N=51,478). Coronary heart disease was responsible for 54% of the deaths from CVD (N=19,548), whereas stroke and other brain-related vascular diseases accounted for 23%. (Statistics Finland 2014a, cited 23.7.2015.) According to the national institute for health and welfare, there were more than 180,000 people taking medication for coronary heart disease CHD in 2012. In the same year, there were 21,769 episodes of heart attack or other CHD symptoms such as chest pain; nearly one out of five who had a heart attack or chest pain was between age 15-64. 11,591 people died from CHD; one out of ten who died from CHD was between age 15-64. Every year, coronary heart disease alone claims on average 12,000 lives; no other disease kills as many as CHD does. Despite the number of death from cardiovascular disease has decreased, the national institute for health and welfare is concerned that the number of the population affected by coronary heart disease would increase in the future, as the country is ageing. (Statistics Finland 2013, cited 22.7.2015; Statistics Finland 2014a, cited 23.7.2015; Statistics Finland 2014b, cited 24.7.2015; Terveyden ja Hyvinvoinnin Laitos 2014, cited 19.5.2015.)

In the UK, the leading cause of mortality is coronary heart disease. The British Heart Foundation announced that the deaths from CVD accounted for 28% of the national mortality in 2012; it was one percent less than the mortality of cancer. It was the first time after five decades that the leading cause of mortality in the country is not cardiovascular disease. Nevertheless, according to the UK's mortality statistics, CHD alone killed 73,680 people in 2012, which accounted for 13% of the national mortality and made CHD the most notorious killer in the country. Together, coronary

heart disease and strokes were responsible for 71% of the CVD deaths and 20% of the national mortality in 2012. Furthermore, the latest report published by the British Heart Foundation revealed that nearly 2.3 million people in the UK were affected by coronary heart disease; 1.17 million people had suffered a stroke; about one million people were troubled with the condition of irregular and rapid heart rate, called atrial fibrillation; 480,000 people struggled with heart failure, which means the heart does not have enough strength to maintain the circulation. The report also indicated that 22% of the premature deaths, i.e. die earlier than age 75, in 2012 were due to cardiovascular disease. In terms of monetary costs, an estimation of over eight billion pounds in total was spent by the whole National Health Service in 2012 fiscal year on treating cardiovascular disease. In addition to the NHS expenditure on CVD, the report suggested that non-healthcare costs such as informal care for CVD patients and productivity losses due to CVD could be 1.5 times bigger than the NHS spending. (NHS choices 2014h, cited 27.7.2015; Townsend, Williams, Bhatnagar, Wickramasinghe & Rayner 2014, 6, 13-15, 19-21, 59, 63, 86, 92-94; NHS choices 2015d, cited 27.7.2015.)

On the global scale, cardiovascular disease is the leading cause of mortality. WHO estimated that CVD killed 17.5 million people in 2012, which accounted for 31% of the global deaths in that year; more than 75% of the global CVD deaths occurred in low- and middle-income countries. The disease was also responsible for 37% of the globally premature deaths, i.e. die before age 70. (2015b, cited 21.7.2015.)

2.2 Chronic obstructive pulmonary disease

When a person inhales, air enters the person's nose or mouth and travels through the person's throat, where the air flows into the trachea, the airway to the lungs. At the end of the trachea, air enters the lungs through the bronchi, the large airways branched from the trachea, and then flows into the bronchioles, the small airways branched from the bronchi. Finally, the air fills the alveoli, the tiny air bags at the end of the bronchioles. Through the thin blood vessels (capillaries) on the walls of the alveoli, oxygen from the inhaled air is exchanged for carbon dioxide from the capillaries. The oxygen-rich blood then returns to the heart to be pumped into the body, whereas the air which contains carbon dioxide after the exchange is pushed upward by the diaphragm, the muscle between the chest and the abdomen. Through the same path, the air is released from the nose or mouth, when the person exhales. The nose, throat, trachea and lungs form the

respiratory system, which exchanges the carbon dioxide produced by the body for oxygen. (The Nemours Foundation 2012, 1-2, cited 31.7.2015.)

Chronic obstructive pulmonary disease (COPD) is the general term for persistent infection of the lower part of the respiratory system, i.e. the bronchi, bronchioles and alveoli. The disease causes airways to gradually clog up and includes chronic bronchitis and emphysema. (Koskela 2005, cited 3.8.2015; WHO 2015c, cited 31.7.2015.) Chronic bronchitis is a prolonging infection of the bronchi, which causes persistent irritation and inflammation in the airways; as a result, less air can enter the lungs, because the airways become narrow and produce more mucus than normal. (Johns Hopkins Medicine 2015c, cited 31.7.2015.) On the other hand, emphysema is a condition where the amount of the healthy alveoli decreases due to permanent damage, thus reducing the amount of oxygen entering the capillaries when inhaling (ibid. 2015d, cited 31.7.2015). Therefore, a person with COPD would cough more often and cough up mucus or even blood; the person would easily feel short of breath and could have difficulty in breathing even when performing daily activities like doing grocery shopping and taking out garbage. The person also becomes more susceptible to other lung infection like pneumonia. If the person left COPD untreated, her/his lungs will continue deteriorating and eventually stop functioning. (NHS choices 2014i, cited 31.7.2015; NHS choices 2014j, cited 31.7.2015; NHS choices 2014k, cited 31.7.2015; NHS choices 2015e, cited 31.7.2015.)

Smoking, exposing regularly to dusty air or chemical gases, using biomass fuels such as manure for heating and cooking indoors, as well as having lung infections frequently in one's childhood are commonly known to increase a person's risk of developing COPD. In high- and middle-income countries, smoking is the biggest risk factor of COPD, whereas using biomass fuels indoors is the most frequent cause of COPD in low-income countries. (WHO 2015d, cited 3.8.2015.)

In Finland, COPD kills about 1,000 people every year. In addition, it has been estimated that 200,000 people are affected by the disease; whereas 200,000 more have chronic bronchitis, which can deteriorate and lead to full-blown COPD, if the condition is left untreated. (Koskela 2005, cited 3.8.2015.) The main cause of COPD in the country is smoking – nine out of ten COPD patients were smokers before they were diagnosed (Mustajoki 2014, cited 3.8.2015). Other reasons include exposing to dust or chemical gases at work and lack of alpha-1 antitrypsin in one's genes. It has been anticipated that the prevalence of COPD might increase not only

because the country is ageing, but also because more women and young people are smoking nowadays. (Koskela 2005, cited 3.8.2015.)

In the UK, 25,000 - 30,000 people died from COPD every year. The British Lung Foundation estimated that three million people in the country are having COPD, but only one third of them are receiving treatment. It is because people are less aware of the disease and its severity; if they are smokers, they usually think the symptoms of COPD as merely “smoker’s cough” and do not seek treatment for it. (British Lung Foundation 2014, cited 4.8.2015; NHS choices 2014i, cited 31.7.2015.) However, the cost for neglecting “smoker’s cough” could be high: one in four smokers can be affected by COPD later, while 80% of the COPD in the UK are caused by smoking (British Lung Foundation 2014; NHS choices 2014l, cited 4.8.2015). On the other hand, the Health and Safety Executive (2014, 2) indicated that 15% of COPD cases could be caused by exposing to dust and chemical gases at work. While smoking and exposing to dust and chemical gases at work are the major causes of COPD, a few cases are caused by genetic deficiency and exposure to air pollution which is not work-related (NHS choices 2014l, cited 4.8.2015).

In the world, more than three million people died from COPD in 2012, which made the disease the third biggest killer of the year after coronary heart disease and stroke (WHO 2014b, cited 4.8.2015.) According to WHO, the mortality of COPD in low- and middle-income countries constitutes over 90% of global COPD deaths. The reason has been attributed to poor prevention and control. In terms of gender, COPD affects as much women as it affects men nowadays, since the number of women in high-income countries who are smoking has increased and many women in low-income countries continue exposing themselves to indoor air pollution from using biomass fuels and coal. (2015e, cited 4.8.2015.) Overall, it has been estimated that 64 million people worldwide are suffering from COPD; smoking is the main reason. (WHO 2015c, cited 31.7.2015; WHO 2015d, 3.8.2015.)

2.3 Diabetes

Glucose, a basic type of carbohydrate is one of the primary energy sources for the human body as it can be effectively converted to energy by the body cells. Furthermore, it is the preferred energy source to the brain (Gebel 2009, cited 6.1.2016; Gebel 2011, cited 6.1.2016).

For the body cells to consume glucose, insulin is required. After digestion, the carbohydrates in the food and drink consumed by a person disintegrate into glucose molecules, which are then released into the vascular system. As a result, blood glucose level, the amount of glucose in the blood increases. Normally, the increase signals the pancreas, a long-shape gland behind the stomach to supply more insulin. Insulin is a necessary hormone for glucose molecules entering the body cells, thus reducing the blood glucose level. Once the molecules enter the body cells, they are converted to energy or stored as glycogen in the liver or muscles for later use. (Gebel 2009, cited 6.1.2016; GroupHealth 2014a, cited 6.1.2016; GroupHealth 2014b, cited 6.1.2016.)

Therefore, when insulin is absent or insufficient, or when the body cells resist insulin, blood glucose level will rise and remain at a high level which can eventually damage the body. The persisting condition is known as diabetes. Specifically, a person will be diagnosed as having diabetes, if her/his fasting blood glucose level after waking up continues to be 7 mmol/l or higher, if the level is higher than 11 mmol/l after an oral glucose tolerance test, or if the amount of glucose in the hemoglobin in red blood cells is 48 mmol/mol (6.5%) or more. (Mustajoki 2015, cited 6.1.2016; Diabetes.co.uk 2016, cited 7.1.2016.) When the body cells resist insulin, glucose cannot be consumed for energy; thus, a person with diabetes gets tired more easily. The person also urinates more frequently as the body is trying to remove extra glucose through urine. Consequently, she/he feels thirsty more often. These are some of the common symptoms. (NHS choices 2014o, cited 7.1.2016.) If diabetes is left uncontrolled, the condition will deteriorate and can cause complications by damaging the vascular system, the peripheral nerve system, eyesight, the kidneys, muscles and skins (NHS choices 2014p, cited 7.1.2016).

Diabetes is usually categorized into type 1, type 2 and gestational diabetes (NHS choices 2014m, cited 7.1.2016). In type 1 diabetes, the pancreas cannot secrete insulin, because the cells responsible for producing the hormone have been destroyed by the immune system. Therefore, a person with type 1 diabetes depends on regular injection for insulin supply. Type 1 diabetes usually develops in a patient's childhood; the exact cause of the autoimmune condition remains unknown. (Mustajoki 2015, cited 6.1.2016.) On the other hand, in type 2 diabetes, either the pancreas fails to supply sufficient insulin or the body cells resist the hormone. A person's risk of developing type 2 diabetes is believed to increase as the person gets old, if she/he has a large waist, if her/his parent or sibling also has type 2 diabetes, or even by being of certain ethnic origin. (NHS choices 2014n, cited 7.1.2016.) Especially, for overweight or obese middle-age people, the risk rises 10 to 20-fold (Mustajoki 2015, cited 6.1.2016). Gestational diabetes, on the

other hand, is a temporary condition which might occur for some women during their pregnancy. If a woman has gestational diabetes, her risk of type 2 diabetes in the future increases to 30%. Overall, the most prevalent category is type 2 diabetes. (NHS choices 2014m, cited 7.1.2016.)

In Finland, population ageing and limited healthcare resources concern the Finnish diabetes association (Suomen Diabetesliitto). According to the association, the trend of new diabetes diagnoses based on the 2000-2014 statistics of people receiving special reimbursements for blood glucose medicines reached its peak in 2011, when 33,383 people began to receive the reimbursements. In 2014, the number declined to 24,682 people; it was still two times more than in 2000. Furthermore, the number of retired people receiving the reimbursements had increased threefold after 15 years. By the end of 2014, the total number of people receiving the special reimbursements was 300,708, which was 2.3 times more than in 2000. (Koski 2015, 9-11.) After 2000, researches, programs and projects have been carried out to design and promote strategies for more effective diabetes treatment and prevention (Koski 2015, 12). However, due to the resource constraints in the healthcare system, many diabetes patients still feel being left alone in dealing with the disease (Koski 2015, 16). How to implement holistic treatment for diabetes and how to carry out effective prevention with limited resources remain great challenges to the country (Koski 2015, 17-23). Furthermore, since it could take years for people to realize that they have type 2 diabetes, the number of people receiving the reimbursements only represents part of the diabetes population. The national institute for health and welfare (Terveyden ja Hyvinvoinnin Laitos) estimated that half million people have type 2 diabetes and 50,000 people have type 1 diabetes. (Terveyden ja Hyvinvoinnin Laitos 2015, cited 8.1.2016.)

In the UK, obesity and sedentary lifestyle put Britons at increased risk of developing diabetes. According to Diabetes UK, the prevalence of overweight condition and obesity had increased 13% from 1980 to 2013; as a result, nearly two third of Britons are above their normal weight. In addition, about 60% of men and 70% of women do not exercise enough. (2015a, 6.) The increase in the prevalence of diabetes seems to be associated with the increase in the overweight and obese population. In 1996, 1.4 millions of people were diagnosed as having diabetes. The figure has grown 2.5 times by 2015; almost 3.5 millions of people were diagnosed as having diabetes. In addition, it is estimated that more than half a million have diabetes but have not been diagnosed. Hence, the current diabetes population in the country is believed to be no less than four million people, of which 90% are having type 2 diabetes and 10% are type 1 diabetes. It is anticipated that the diabetes population will reach five million by 2025. (Diabetes UK 2015a, 2-3.)

For treating diabetes and reducing its complications effectively across the UK, the National Institute for Health and Care Excellence (NICE) has published clinical guidelines for diabetes. The guidelines advise people with diabetes to check at least once a year their glycated hemoglobin (HbA1c), blood pressure, cholesterol, the retina, kidneys, feet, BMI and smoking habit. In addition, these people are urged to take structured education for managing their diabetes and to work with their healthcare professionals in setting and achieving targets for lowing blood glucose, cholesterol and blood pressure, thus reducing the risk of developing complications. However, according to the national diabetes audit in 2012-2013, there were still 40% of people with diabetes in England and Wales who did not complete all the recommended checkups; the figure rose to about 70% for type 1 under 40 years old and 54% for type 2 in the same age group. The audit further indicated that the structured programmes have been offered to only about 6% of people with diabetes in England and Wales, and less than 2% of the diabetes population in England and Wales attended the programmes between 2012-2013. Finally, the majority of people with diabetes in England and Wales (about 64%) were still at high risk of developing complications, since at least one of their three values was still above the target level recommended by NICE. The audit pointed out that there is need for adjusting healthcare services to better reach and serve patients with type 1 diabetes and patients with type 2 diabetes who are under 40, and for ensuring that diabetes patients are equipped with sufficient information for managing their condition. (Health and Social Care Information Centre 2014, 9-10, 17, 21, 24.) Similar needs exist also in Scotland, where type 1 diabetes patients received fewer checkups, and the rate of attending structured programmes was also very low (Diabetes UK 2015b, 8, 13, 18).

To combat diabetes, the UK authorities have started new initiatives in the last two years. In 2014, the Scottish government (2014, cited 13.1.2016) published a diabetes improvement plan to improve its healthcare system for better diabetes control and prevention. In 2015, the National Health Service (NHS) and Public Health England have begun a pilot project for national NHS diabetes prevention programme. The pilot project offers to participants who are at high risk of developing type 2 diabetes an intensive programme which includes doing more exercise, having a healthier diet and reducing weight. (NHS England & Public Health England 2015, cited 13.1.2016.)

In the world, obesity and sedentary lifestyle are the major causes for the increasing prevalence of type 2 diabetes, which accounts for 90% of the total diabetes population (WHO 2015f, cited

13.1.2016). As estimated, one in every twelve adults in the world is having diabetes and one ninth of global healthcare cost is spent for treating diabetes and its complications (International Diabetes Federation 2015, 10). Among the complications, cardiovascular disease is responsible for at least 50% of mortality of diabetes patients. The mortality of diabetes is expected to become the 7th cause of global death by 2030. (WHO 2015f, cited 13.1.2016.) Furthermore, it is anticipated that 592 million people will have diabetes and a similar number of people will be at high risk of developing diabetes by 2035 (International Diabetes Federation 2015, 6). WHO has acknowledged the growing trend of diabetes, particularly in low- and middle-income countries; and has dedicated this year's World Health Day, in addition to the World Diabetes Day, to raise global awareness of diabetes and to promote effective control and prevention (2016).

2.4 Control and prevention

A lifestyle of eating a healthy diet, doing exercise sufficiently, drinking less alcohol and smoking cessation is believed to help patients with chronic diseases keep their condition under control, besides complying with treatment. Furthermore, for people without chronic diseases, adopting such a lifestyle can reduce the risk of developing chronic diseases. (WHO 2014a, cited 20.5.2015; WHO 2015a, cited 19.5.2015.)

A healthy diet provides humans with necessary nutrients without harming the body. Although the amount of calories in a healthy diet varies for people of different age, body mass index, level of physical activity and health conditions (NHS choices 2014q, cited 14.1.2016); there are general guidelines for the contents of a healthy diet. National Health Service in the UK has published "The eatwell plate", which illustrates a healthy diet as a balanced diet among carbohydrate, fibre and protein, while providing the vitamins and minerals needed by the body.

According to "The eatwell plate", one third of the plate should contain starchy foods like potatoes and bread, as starchy foods keep a person full longer and provide calcium, iron and B vitamins besides starch. In addition, if a person eats potatoes with their skins on or wholegrain breads, the person is also getting fibre, which helps digestion. (NHS choices 2015f, cited 14.1.2016; NHS choices 2015g, cited 14.1.2016.)

Another one third of the plate should contain fruit and vegetables, as fruit and vegetables provide vitamins, minerals and rich fibre. Specifically, taking at least 400g of fruit and vegetable every day is believed to reduce the risk of cardiovascular disease and cancer. (NHS choices 2015f, cited 14.1.2016; NHS choices 2015h, cited 14.1.2016.)

The last one third of the plate should contain the kind of food providing protein but containing very little fat (NHS choices 2015f, cited 14.1.2016), because protein is a necessary nutrient for growing and repairing muscles and other body tissues (NHS choices 2014r, cited 14.1.2016), and a little amount of fat is required for the body to receive vitamins A, D and E (NHS choices 2015i, cited 14.1.2016). Examples of foods containing rich protein and little fat are lean meat, chicken meat without the skin, white fish, pulses and low-fat milk. Furthermore, it is recommended to get protein from diverse sources thus reducing the consumption of red and processed meat to 70 grams a day. (NHS choices 2015f, cited 15.1.2016; NHS choices 2015j, cited 15.1.2016; NHS choices 2015k, cited 15.1.2016; NHS choices 2015l, cited 15.1.2016; NHS 2015m, cited 15.1.2016.)

There are two main reasons for favouring low-fat food: avoiding excessive calories and reducing the intake of saturated fat. A person gains weight by eating more than her/his body needed. A gram of fat contains 9 Kcal, whereas a gram of carbohydrate or protein contains 4 Kcal; in other word, a person who eats a high-fat diet is likely getting more calories than her/his body need thus becoming overweight or obese. Moreover, by eating more fat, a person is inevitably getting more saturated fat, which increases the amount of low density lipoprotein (LDL) (a harmful type of cholesterol) and triglycerides (a type of fatty matter) in the blood. As a result, atherosclerosis and embolism can develop and eventually trigger cardiovascular disease. (NHS choices 2015i, cited 14.1.2016.)

In addition to avoiding excessive calories and saturated fat, NHS urges people to reduce the intake of sugar and salt, because a person could easily become obese by eating excessive amount of sugar and could have hypertension by adding too much salt to her/his food. According to NHS, an adult should not consume more than 30 grams of added sugar and 6 grams of salt a day; however, many Britons eat much more sugar and salt than recommended. (NHS choices 2015n, cited 15.1.2016; NHS choices 2015o, cited 15.1.2016; NHS choices 2015p, cited 15.1.2016.)

A healthy diet is, therefore, a balanced diet which provides the body with the right amount of nutrition and without putting a person at risk of becoming overweight and developing high blood glucose, high blood pressure, high cholesterol and triglycerides. For people having chronic diseases, it could be useful to consult a dietician in order to customize a friendly diet for their specific condition (NHS choices 2015f, cited 14.1.2016; East Sussex Healthcare NHS Trust 2016, cited 15.1.2016).

However, eating a healthy diet alone is not enough for preventing or controlling chronic diseases, if a person continues living a sedentary lifestyle and/or being overweight or obese. Especially, the risk of developing cardiovascular disease and type 2 diabetes for overweight and obese people is higher than people having normal weight (NHS choices 2014f, cited 29.5.2015; Mustajoki 2015, cited 6.1.2016). Specifically, there is a negative correlation between obesity and physical health-related quality of life: overweight and obese people are more likely to experience physical pain, illness and impairment; and are more likely to have "high blood pressure, high blood glucose, low HDL cholesterol and high triglycerides", which could accelerate the development of chronic diseases like cardiovascular disease and diabetes (Korhonen et al. 2014, 71, 73). Hence, achieving and maintaining a healthy weight is integral to effective control and prevention of chronic diseases. As estimated, a 1.61 kg/m^2 reduction in body mass index could reduce the risk of having diabetes related incident by 20% for middle-aged people without diabetes; for obese people losing weight, a 30% decrease in the risk could be anticipated (Appuhamy, Kebreab, Simon, Yada, Milligan & France 2014, 13). Finally, for losing weight effectively, doing sufficient exercise is required in addition to eating a healthy but fewer calories diet (Tarnanen, Kesäniemi, Kettunen, Kujala, Kukkonen-Harjula & Tikkanen 2010, cited 15.1.2016).

The combination of healthy diet and sufficient exercise not only can reduce the risk of developing cardiovascular disease and diabetes, but also can alleviate the condition of COPD. Generally, sufficient exercise means doing at least two and half hours of moderate aerobic exercise or one hour fifteen minutes of vigorous aerobic exercise every week, plus strength exercise at least twice a week. Fast walking and cycling are considered as moderate aerobic exercise; whereas jogging and cross-country skiing exemplify vigorous aerobic exercise. Strength exercises like doing yoga and going to gym, on the other hand, emphasize on improving the body's flexibility and strengthening the muscles. For people over 64 years old, it is recommended to exercise their joints and train balance, in addition to regular aerobic exercise and strength exercise. (Tarnanen et al. 2010, cited 15.1.2016.)

For people who drink alcohol regularly, they are urged to reduce the consumption to no more than 14 units per week, the less alcohol the better; as well as to avoid drinking all the units in one day, according to the new alcohol guidelines published by the UK's Department of Health. The Department of Health indicated that although a person will not be free from chronic diseases by complying with the new recommendation, it does lower the person's risk of cancers, liver disease and other diseases. (2016, cited 18.1.2016.) The other diseases include cardiovascular disease and impairment of the nervous system (NHS choices 2014s, cited 18.1.2016).

Finally, smoking cessation can significantly reduce the risk of developing COPD, in addition to avoiding being exposed to harmful gas (WHO 2015d, cited 3.8.2015). For smokers with COPD, smoking cessation delays the deterioration of their lungs (NHS choices 2014i, cited 31.7.2015). Besides the respiratory system, the heart can also benefit from smoking cessation, as it stops the poisonous substance in tobacco from harming coronary arteries (NHS choices 2014f, cited 29.5.2015).

2.5 eHealth for chronic disease patients

Patients with chronic disease might feel unmotivated and powerless to adopt a healthy lifestyle, although they would be informed that adopting such a lifestyle is integral to the control and prevention of their condition. The feeling and perception could be affected by the patient's health conditions, personal knowledge about the disease, life circumstances, and support from family and healthcare professionals. (Kousoulis et al. 2014, 2, 5, 11-14.) The same perception and feeling could also hinder some obese people without chronic disease in losing weight; Korhonen et al. (2014, 73) found that some obese people are not motivated to lose weight because they are either content with their life or unable to do enough exercise due to physical impairment. For motivating the patients and other people who are at high risk of chronic diseases to take better care of themselves, intervention from healthcare professionals is essential. Kousoulis et al. indicated that in Europe, healthcare systems are expected to equip diabetes patients for efficient self-management as well as to intervene to reduce citizens' risk of developing diabetes. However, to facilitate efficient self-management, much work still needs to be done, particularly in improving the relevance and dissemination of educational information, in improving the quality and frequency of communication among the patients, healthcare professionals and other stakeholders; and in reducing overall cost of self-management. (2014, 11-14.) Despite the need

for more support and intervention from healthcare professionals, EU countries are in fact tightening healthcare spending due to adverse economic conditions (ibid., 13). Furthermore, it has been recognized that the challenge faced by EU, to effectively control and prevent chronic diseases with finite resources is indeed global (Wicks et al. 2014, 196).

For motivating and empowering people to take better care of themselves while alleviating the financial burden on healthcare systems, eHealth tailored to achieve specific goals might offer the solution (Wicks et al. 2014, 196, 201). eHealth is a general term that means providing health information and/or service by ICT solutions (ibid., 195). For instance, a web-based healthcare system like CDMS and video conference could improve the efficiency and quality of healthcare service; on the other hand, an online structured educational program, community and personal health record application could help individuals adopt a healthy lifestyle (ibid., 196).

Over the last few years, studies regarding eHealth have been conducted in both sides of the Atlantic. In the United States, a study suggested that a healthcare provider could lower its customers' glycated hemoglobin (HbA1c) value while saving cost, by implementing a mobile application which automatically reminds the customers about medication and self-care and sends structured educational information to the customers (Nundy, Dick, Chou, Nocon, Chin & Peek 2014, 266, 268-270). In the UK, University College London (2015, cited 20.1.2016) has developed an online self-management program for type 2 diabetes patients called "HeLP Diabetes". The program is designed to provide diabetes patients with both initial training and continuous support: it provides the user with educational information, with tools to manage medication and to adopt a healthier lifestyle, and with support from healthcare professionals and other users (Ross, Stevenson, Dack, Pal, May, Michie, Parrott & Murray 2014, 2). The program is still under evaluation (University College London 2015, cited 20.1.2016). In Finland, a study was made utilizing persuasive technology in the development of a web-based weight loss application. The application provides the user with weekly information on weight management and with tools and automatic reminders for self-monitoring. In addition, it enables the user to discuss questions related to the weekly topics with other users. The study suggested that to support the user in adopting a healthier lifestyle, an application should be easy-to-use in the first place; secondly, the application should aim to provide the user with credible and comprehensible information tailored to her/his condition; thirdly, suitable social support features should be incorporated in the application. (Alahäivälä 2013, 20-22, 29-35, 47-50, 54-55.) All the presented studies revealed that a multidisciplinary approach involving target users and healthcare experts was adopted in

tailoring self-management programs to achieving specific objectives (Alahäivälä 2013; Nundy et al. 2014; Ross et al. 2014).

According to Wicks et al. (2014, 201), "there is no one size fits all solution, and matching the right technology for a given patient population or desired clinical objective is key to ensuring sufficient perceived usefulness and uptake." In the future, it has been anticipated that by leveraging to sensors, video conference and the "Internet of Things", eHealth could make self-management and intervention from healthcare professionals more effective and efficient (Wicks et al. 2014, 199).

Nowadays, many personal health applications targeting diabetes self-management can be found from an app store, in addition to those monitoring diet and exercise. For each application, the app store would display ratings and comments from other users. (Google 2016a, cited 22.1.2016; Google 2016b, cited 22.1.2016.) However, the effectiveness of using those applications for self-management still remains to be evaluated by users themselves. Meanwhile, Apple and Microsoft have started to interface some fitness and medical applications and sensors with their health platform, in an attempt to provide their users with a holistic picture of personal health. For example, Apple provides its users with an application called Health; one feature of the application is presenting the user's health progress with a dashboard, which uses charts to display personal data collected from other iOS health and fitness applications and from sensors, e.g. iHealth Wireless Blood Glucometer. By looking at the dashboard, the user can quickly see her/his overall health conditions. (Apple 2016a, cited 22.1.2016; Apple 2016b, cited 22.1.2016; Apple 2016c, cited 22.1.2016.) Similar integration can also be observed from Microsoft. The company has released a mobile application named Microsoft Health, which works with Microsoft Band to automatically monitor its user's physical activities and sleeping and to help the user determine whether she/he has exercised enough and slept well. The data can be shared with Microsoft HealthVault, which can store other health related data in addition to exercise and sleeping quality. (Microsoft 2016a, cited 22.1.2016; Microsoft 2016b, cited 22.1.2016.)

In terms of user interface, charts and dashboards seem to be frequently used to visualize the progress of the user's health conditions in a personal health application, whether it is for general fitness or for diabetes self-management. For example, Apple Health uses line charts and bar charts to summarize the user's physical activities, weight management and the range of heart rate in a dashboard screen (Apple 2016a, cited 22.1.2016). Microsoft Health, on the other hand,

displays the cumulative amounts of the user's physical activities, burned calories and sleep in a summary screen and uses a map, a chart or a table accompanied by quantitative data to show the details of a selected activity like jogging or golfing in another screen (Microsoft 2015, cited 25.1.2016). In a highly-rated application for diabetes self-management like mySugr Diabetes Logbook, charts and relevant figures like daily highest and lowest blood glucose values, consumption of carbohydrates, intake of medication and the duration of exercise are presented in one screen to summarize the user's current conditions; the combination of charts and figures is also used to display the details of the user's diet (mySugr GmbH 2016, cited 25.1.2016). While mySugr Diabetes Logbook utilizing charts, figures, icons and images extensively; another highly-rated app, Diabetes:M contains more text and tables. Nevertheless, the application also uses charts to visualize the user's blood glucose development and the daily intake of insulin over a given period of time. (Verbanov 2015, cited 25.1.2016.)

In summary, the advantages of eHealth have the potential to empower both healthcare systems and individuals to better control and prevent chronic diseases. However, there seems to be at least two prerequisites for leveraging the advantages: the first prerequisite is possessing sufficient knowledge about the need of target users and the second is tailoring a solution to the goal with suitable technologies. For satisfying these prerequisites, a multidisciplinary approach involving patients, healthcare professionals and IT experts would be necessary. After a brief review of personal health applications, I found that there are already many mobile applications for fitness and for diabetes self-management, but the effectiveness of these applications remains to be evaluated by users themselves. On the other hand, it is more apparent that charts and dashboards are used frequently to convey overall health conditions and progress to the user. Finally, the ongoing integration across mobile applications, sensors and health platforms exemplifies the anticipation of more empowering eHealth in the future.

3 TECHNOLOGIES AND TOOLS CHOSEN FOR PROTOTYPING

The objective of this thesis project is to create a new PHR prototype tailored to the commissioner's requirements. The commissioner is responsible for communication with the company's stakeholders and providing me with software requirements, advice and feedback during the project. My responsibility is to deliver a prototype accordingly, which the commissioner can present to the stakeholders and acquire necessary information from them for future iterations.

The prototype should be responsive, interactive, using charts and tables, and able to leverage the company's existing platform. Firstly, it should be readable and easy-to-navigate on both desktops and mobile devices. Secondly, it should make the user's tasks easier, for example, by providing the user with relevant information only and by breaking an otherwise complicated form down into logical steps. Thirdly, it should convey data effectively to the user by charts and tables. Finally, it should be able to use the components from the company's platform. Above mentioned are general descriptions of the requirements. The details are not documented in the thesis report, in order to protect the business secrets of the company. Nevertheless, these general descriptions should be sufficient to serve as the criteria for choosing suitable technologies and tools.

Based on the descriptions, the main technologies and tools chosen included ASP.NET MVC5, Bootstrap, jQuery, Chart.JS and Tablesorter. ASP.NET MVC5 framework was chosen, so the prototype can leverage the company's platform which is built on ASP.NET technology and enjoy the benefits of MVC such as separation of concerns. Bootstrap was selected, since it is a well-known framework for responsive web design and is included in the ASP.NET MVC5 solution by default; in other words, by using Bootstrap, time can be focused on prototyping. jQuery, on the other hand, is a powerful JavaScript library and makes creating interactive webpages easier. It is also included in the ASP.NET MVC5 solution, thus offering the same benefit as using Bootstrap. Finally, Chart.JS and Tablesorter are open source plugins and were chosen for drawing responsive charts and creating sortable tables.

This chapter covers the features of the chosen technologies and tools which are integral to the project, namely jQuery selectors, Chart.JS line charts, Tablesorter multicolumn and attribute sorting methods; ASP.NET MVC5 partial views, resource files, and methods for examining dynamic forms and testing the prototype on the mobile device. For clarity, italic type is used to

indicate HTML tags, XML tags, the names of folders and files in the Visual Studio solution, the names of classes and methods, and the statements of programming languages.

3.1 jQuery selector

jQuery is an open source JavaScript library, which empowers developers by providing them with a wide collection of robust functions for creating interactive web and mobile applications. By utilizing the library, developers can save a lot of time from coding and testing their own functions and focus more on realizing the vision of their projects. (The jQuery Foundation 2015a, cited 17.11.2015; The jQuery Foundation 2015b, cited 17.11.2015.)

The mean which enables JavaScript and jQuery to access, animate and change a webpage is the Document Object Model (DOM). DOM is a group of objects which are created and arranged hierarchically to represent the organization of a document such as a HTML page (MDN 2015, cited 18.11.2015). A DOM is created by the browser after it loaded a webpage and is accessible via JavaScript and jQuery (W3Schools 2015a, cited 18.11.2015).

Although using JavaScript alone can manipulate the DOM, I found using jQuery DOM selectors more effective in terms of saving time and reducing the amount of code. Using JavaScript alone, a developer who is unfamiliar with the language and DOM needs to learn firstly that there are different HTML DOM methods for accessing HTML elements: *getElementById()*, *getElementsByTagName(), getElementsByClassName()* and *querySelectorAll()*; in addition, there are HTML DOM objects which can be directly accessed. Each of the directly accessible objects represents a HTML element or a collection of an element, such as forms or images (W3Schools 2015b, cited 18.11.2015; W3Schools 2015c, cited 18.11.2015). On the other hand, using a jQuery selector, the developer can access a HTML element with fewer lines of code and can leverage the knowledge of CSS selector syntax (The jQuery Foundation 2015c, cited 18.11.2015). For example, using JavaScript to get all <*p*> elements in a <*div*> element with an id "myDiv" would require two methods: *getElementById()* and *querySelectorAll()*. On the other hand, using a jQuery selector only requires the same CSS syntax for selecting all <*p*> elements under the id "myDiv" in the selector's brackets. Figure 1 illustrates how using a jQuery selector leverages CSS selector syntax and reduces the amount of code.

```
//JavaScript only
var pElementsInMyDiv = document.getElementById('myDiv').querySelectorAll('p');

//jQuery selector
var pElementsInMyDiv = $('#myDiv p');
```

FIGURE 1. Using a jQuery selector to find HTML elements

After selecting an element or a collection of elements by the jQuery selector, jQuery methods can be chained to the selector. The methods will be executed sequentially on the selected element(s). (W3Schools 2015d, cited 18.11.2015.) For example, there are two <*p*> elements in a <*div*> element with an id "myDiv", and I would like to insert greeting text to the first <*p*> element and introduce myself in the second <*p*> element. I could first select "myDiv" element, then use jQuery method *find()* to choose the first <*p*> element of "myDiv", use *css()* to style the element, and use *html()* to insert greeting text to the element. Then, instead of writing a new statement to choose the last <*p*> element, I could use a jQuery method *end()* to indicate the end of processing the first <*p*> element. After that, I could proceed to the last <*p*> element with *find()* and other methods in a single statement (see figure 2).

```
$('#myDiv').find('p:first-child').css('color','blue').html('Hello World!').end()
    .find('p:last-child').css('background-color','yellow').html('My name is Dalun.');
```

FIGURE 2. jQuery chaining

jQuery selectors and methods not only can be used to manipulate existing HTML element(s), but also can be used to create new element(s) according to a user's input or configuration; thus, it enables tailoring content and creating interactive webpages. It can be illustrated by expanding the previous example. A text input field and a button have been added to the previous example. After I typed some text in the input field and clicked the button, the text will be displayed in a new <*p*> element after the <*p*> element which contains my name (see figure 3).

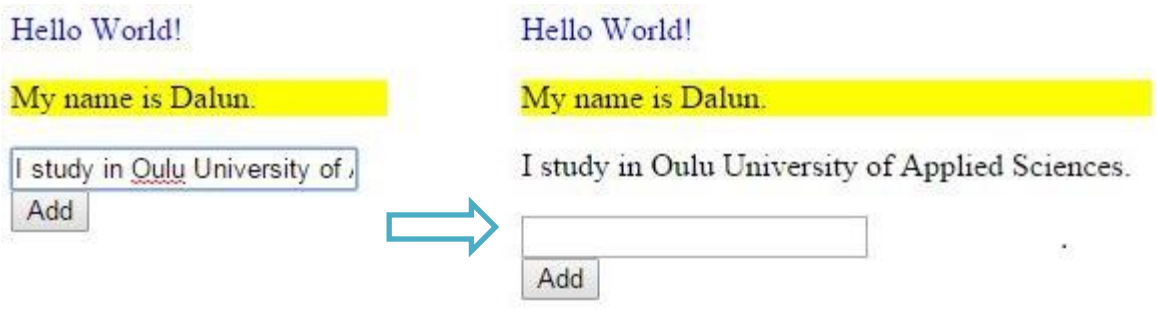


FIGURE 3. Displaying user input in a new element in the browser

The example can be achieved by the following steps: firstly, use a method *click()* to detect the event when a user clicks the button “Add”; secondly, use another method *val()* to retrieve the text entered in the input field; finally, create a new <*p*> element which contains the retrieved text by using a new selector and inserting the element after the <*p*> element which contains my name (see figure 4).

```
$('button').click(function(){
    var myText = $('input').val();
    $('<p>' + myText + '</p>').appendTo('#myDiv');
    $('input').val('');
});
```

FIGURE 4. Displaying user input in a new element by jQuery

The presented examples demonstrate how developers who are unfamiliar with DOM and JavaScript can still create interactive webpages by using jQuery selectors and methods and by leveraging the knowledge of CSS. In addition, the developers would find jQuery easy-to-learn, since there are comprehensible documentation and examples on the websites of the jQuery Foundation and W3Schools.

3.2 Open source plugins: Chart.js and tablesorter

In autumn 2014, a fellow BIT student with whom I did professional training searched for a suitable open source chart plugin for a project in which we both participated. From him, I heard about Chart.js plugin for the first time. The plugin is under the MIT license, provides animation when drawing a chart and produces in a line chart, a smooth curve between dots instead of a straight line (Chart.js 2015a, License, Line Chart, cited 25.11.2015). However, the version we used does not by default connect two dots if there is a missing value in between. The search for a chart plugin which connects two dots over missing value(s) had been continued in the group work of another degree programme course “Information Technology Project” in spring 2015. By the end of the group work, Chartist plugin met all the criteria we set: free for commercial use, responsiveness, customizable X and Y labels, and connecting two dots over missing value(s) (CHARTIST.JS 2015, cited 25.11.2015).

As charts which connect two dots over missing value(s) were required for the thesis project, Chartist plugin was utilized in the beginning. However, after comparing the appearance of a

Chartist chart with a Chart.js chart, the commissioner found Chart.js more appealing; therefore, I began searching for the solution by learning how to achieve the same appearance as Chart.js charts with Chartist plugin and by revisiting Chart.js documentation and its GitHub repository.

After I downloaded and tested Chart.js version 1.0.2, to my delight, I found the version connects two dots over missing values by default. Although the latest version was 2.0.0-alpha3 when I revisited the repository in August 2015, I decided to use the version 1.0.2 for three reasons: stability, time, and capability. Firstly, version 2.0.0-alpha3 is a testing version which might be less stable than a released version; whereas version 1.0.2 is the latest released version. Secondly, it would require time to learn version 2.0.0-alpha3, since new methods are used in the version to create and configure charts (Chart.js 2015b, 2.0.0-alpha3, cited 25.11.2015); on the other hand, I have learned in the professional training and the IT project course, how to create and configure charts with version 1.0.1; the same methods are still available in version 1.0.2. Finally and most importantly, version 1.0.2 can produce charts which not only connect two dots over missing values, but also are considered appealing by the commissioner.

Before Chart.js plugin can be used to draw charts in a webpage, the plugin file must be referred in a <*script*> tag in either the <*head*> element or the <*body*> element of the page. After that, it takes four steps to draw a responsive line chart. Firstly, create a <*canvas*> element in the <*body*>. A pixel value should be given to each "width" and "height" attributes of the <*canvas*> to set the ratio of a chart. An id would make finding the <*canvas*> straightforward in scripts, if there are more than one in the <*body*>. Secondly, create a data object which contains "labels" and "datasets" properties in a subsequent <*script*> tag to the one referring to the plugin file. Assign an array of X labels to the "labels" property. In the "datasets" property, create an object for each line in the chart. The object should contain the following properties: "label", "fillColor", "strokeColor", "pointColor", "pointStrokeColor", "pointHighlightFill", "pointHighlightStroke" and "data". An array representing a line should be assigned to the "data" property. For rendering the chart properly, the length of all the arrays assigned to the data object should be the same. Thirdly, create an option object in the same <*script*> tag and set the "responsive" property of the object to be "true". Finally, instantiate a chart object with the context of the <*canvas*>, the data object and the option object as parameters. As a result, a responsive line chart is created. (Chart.js 2015a, Line Chart, cited 25.11.2015.)

For the plugin to connect two dots over missing values, one simply replaces the missing value(s) in the "data" property of a dataset object with "null". For example, *exampleData.datasets[0].data: [1, 2, null, 3, null, 5]* replaces missing values with "null" in the "data" property of the first dataset object, which exists in the "datasets" property of a data object named "exampleData".

By default, the plugin will display a tooltip when a dot in a line chart is hovered or touched. The feature might, however, have a negative impact on the user experience in a scenario, where a line chart containing data of 30 days or longer is rendered on a mobile device, since the dots will overlap each other and consequently become difficult for the user to trigger the tooltip for the dot she/he wishes to see. A simple solution would be drawing no dots and disabling the tooltip when creating a line chart containing 30 values or more in a line. The effect can be achieved by setting "showToolTips" and "pointDot" properties in the option object to "false" (Chart.js 2015a, Line Chart, Chart options, cited 25.11.2015).

In addition, the plugin generates the same amount of X labels as the amount of data in a line by default; for example, a line chart indicating a progress over the last 30 days would have 30 date labels on the horizontal axis. Again, it would become crowded on a smaller laptop or tablet screen and unreadable on a smartphone screen. To avoid X labels crowding on the screen, one could replace each unneeded X label with an empty string. For example, an array of X labels contains seven date labels: "19.11", "20.11", "21.11", "22.11", "23.11", "24.11", and "25.11". If we wish to show only the first and the last date labels, we can change the array to *['19.11','','','','','','25.11'].* However, since the date labels between the first and the last date are replaced with empty strings, they will not be displayed in the tooltip either. If a requirement is to show the date label in the tooltip but not on the horizontal axis, it can be achieved by customizing a line chart with a Chart.js method *extend*(). Instead of replacing the values in the X label array with empty strings, the method can be used to display X labels in the tooltip but not on the horizontal axis (StackOverflow 2015, cited 25.11.2015). For fine-tuning the appearance of the 30-day line chart, one could set the property "scaleShowVerticalLines" in the chart option object to be "false", thus could remove the vertical lines from the grid displayed in the background of a line chart (Chart.js 2015a, Line Chart, Chart options, cited 25.11.2015).

Finally, as the requirements of the prototype included redrawing a line chart according to the user's selection, a Chart.js method *destroy()* was used. The method discards an existing chart object (Chart.js 2015a, Advanced usage, Prototype methods, cited 25.11.2015), so the same

variable which used to contain the old chart can be reused for storing a new chart object containing updated data and chart options.

Besides the chart, the table was another essential tool for presenting data in the prototype. A line chart can indicate trends whereas a table can organize complicated and massive data into comprehensible rows of information. However, a static table can only display rows in a fixed order; for example, from the latest to the oldest row. As a result, it cannot satisfy users who want to see the rows displayed in different orders. To delight the users who need to sort a table differently for different purposes, after receiving consent from the commissioner, I implemented sortable tables in the prototype.

Again, to save time for developing the prototype, I turned to open source. I found an easy-to-use jQuery plugin for creating sortable tables called “tablesorter”, which is also available under the MIT license. The plugin is developed by Christian Bach, a front-end developer in Sweden. He has made the documentation of the plugin accessible on a dedicated website as well as on GitHub. The plugin and its CSS files can be downloaded from its GitHub repository. (Bach 2015a, cited 27.11.2015; Bach 2015b, cited 27.11.2015; Bach 2015c, cited 27.11.2015.)

After referring to the plugin and its CSS files in a HTML document, one can select a <*table*> element in a page by a jQuery selector and chain to the selector, a method called *tablesorter(),* which transforms an otherwise static table into a sortable one (Bach 2015a, Getting started, cited 27.11.2015). Besides easy-to-use, the main reasons for choosing the plugin include the capability of sorting multiple columns in a table and the option for sorting a set of custom attribute values instead of the set of values visible in a column.

By pressing the “Shift” key, a user can click and sort more than one column (Bach 2015a, Demo, cited 27.11.2015). Since a second and more columns can be chosen and sorted if there are repeated values in the first column selected by a user, it could be useful, for example, for sorting messages by their status and by the time when the messages arrived. In addition, the multicolumn sorting by the status and the arrived time of messages can be passed as parameters to *tablesorter()*; as a result, the selected columns will be sorted by default (see Bach 2015a, Getting started).

Sorting a set of custom attribute values instead of the set of visible values in a column ensures consistent sorting results, even if the values in a table are translated in different languages or rendered in different formats. The consistency could be achieved by assigning to the attribute "data-sort-value" of each cell in a column, a number which represents the text in the cell. Assigning a number to the attribute of each cell in a column instructs the plugin to sort the numbers instead of the rendered text in the cells (Bach 2015d, cited 27.11.2015). The feature was invaluable for the prototype, as it ensures consistent sorting results in English and Finnish.

3.3 ASP.NET MVC5 partial view

In the prototype, certain parts of the content in one page were required in different pages for performing different tasks. If the pages use the same layout, the HTML markup and CSS style for rendering the shared content in these pages are often identical. Since the markup and style are identical, copy-paste them to the pages might appear to be a straightforward solution at first glance. However, it would have demanded more time and effort for editing the markup, if the markup had been copy-pasted to many pages. A more effective solution, leveraging the partial view feature in ASP.NET MVC framework was used instead.

While the partial view feature enables the same HTML markup to be reused in different pages in an ASP.NET MVC solution, it eliminates the need for making change in all the pages which reused the markup. It is achieved by creating a partial view and then rendering the partial view in different pages; as a result, changes made in a partial view will be applied to all the pages which utilize the partial view.

Creating a partial view is similar to creating an ordinary Razor C# document, which allows Razor codes, a server side markup, to be incorporated into a HTML document and uses the extension ".cshtml" (W3Schools 2015g, cited 2.12.2015; W3Schools 2015h, cited 2.12.2015). On the other hand, unlike a complete view in an ASP.NET MVC solution, a partial view does not require a method in the controller. A partial view contains only the necessary markup for rendering a part of a HTML document, for example, a table which will be reused in different pages. (Vatsa 2013, Partial views (Should know), cited 2.12.2015.)

To create a partial view in an ASP.NET MVC5 solution, firstly, right-click on the folder intended for the partial view; as a result, a list will be displayed. Secondly, hover on the "Add" in the list to see a sub-list. Thirdly, click on the "View…" to start the dialog box "Add View". Fourthly, in the box, enter the name for the partial view, select a template (select "empty" if no template is needed), and check the option "Create as a partial view". Finally, click the "Add" button in the box to create the file in the intended folder. It is recommended that the name of a partial view begins with an underscore, e.g. "_MyReusableTable", in order to distinguish a partial view from a normal view (Vatsa 2013, Partial views (Should know), Partial helper, cited 2.12.2015). Once the file has been created, the markup for reusable content can be stored in it.

After storing reusable content in a partial view, there are four methods for rendering the partial view in a parent view: *Html.Partial(), Html.RenderPartial(), Html.Action(),* and *Html.RenderAction().* Both *Html.Partial()* and *Html.RenderPartial()* take the partial view as a parameter and render the view without invoking any method in the controller, whereas *Html.Action()* and *Html.RenderAction()* take the name of a child-method in the controller as a parameter and invoke the child-method. Then, the child-method returns the partial view to be rendered in the parent view. *Html.Partial()* and *Html.RenderPartial()* are straightforward methods when no server-side process is needed, whereas *Html.Action()* and *Html.RenderAction()* are used if a certain process is required, for example, fetching specific data from a database according to a given id. (Vatsa 2013, Partial views (Should know), cited 2.12.2015.)

Since the controller logic was not implemented in the prototype, *Html.Partial()* and *Html.RenderPartial()* were sufficient for rendering reusable content. *Html.Partial()* can be called directly in a Razor inline syntax: *@Html.Partial("_NameOfAPartialView")* whereas *Html.RenderPartial()* must be invoked in a Razor code block: *@{ Html.RenderPartial("_NameOfAPartialView"); }* (Galloway, Wilson, Allen & Matson 2014, Chapter 5, Rendering helpers, cited 2.12.2015).

Html.RenderPartial() is supposed to be faster than *Html.Partial()*, as *Html.RenderPartial()* renders a partial view directly to a parent view; whereas *Html.Partial()* returns an object of *MvcHtmlString* class, which contains a HTML-encoded string and can be assigned to a variable for further processing in a Razor code block (Vatsa 2013, Partial view (Should know); Jones 2015, cited 2.12.2015; Microsoft Developer Network 2015c, cited 2.12.2015). However, it would only become

apparent when the method is used heavily (Galloway et al. 2014, Chapter 5, Rendering helpers, cited 2.12.2015).

3.4 ASP.NET MVC5 resource files

The stakeholders of the company in Finland and the U.K. are the intended audience of the prototype. Since the culture and language of the two countries are different, globalization for the prototype was necessary. Globalization is the task which enables an application to be used by users in different cultures. It consists of two sequential tasks: internationalization and localization. Internationalization designs and prepares an application to be easily rendered in different languages and culture formats; whereas localization means the actual translation. Localization can be further categorized into two levels: language level and locale level. In the language level, only text is localized; whereas in the locale level, date, number and currency are also rendered according to the end user's culture format. (Penberthy 2013, chapter 3, Objective 3.2: Plan and implement globalization and localization, cited 9.12.2015.)

For globalizing an ASP.NET MVC5 application, besides creating language specific views, the ASP.NET MVC5 framework offers an easier way: utilizing resource files. A resource file stores language specific (e.g. "en") or locale specific (e.g."en-GB") text, pictures and videos in XML format. The application can be configured to detect the language-locale preference in the user's HTTP GET request and render a page in the requested language, if a corresponding resource file exists. If such a file does not exist, the application will use the default resource file instead. (Penberthy 2013, chapter 3, Objective 3.2: Plan and implement globalization and localization; Microsoft Developer Network 2015d, cited 9.12.2015.)

Before the resource files can be utilized for localization, configuration is required. Firstly, add <*globalization enableClientBasedCulture="true" culture="auto" uiCulture="auto"*> to the <*system.web*> element in the file *Web.config* situated at the application level (see figure 5). It instructs the application to check the user's language-locale preference and then to display pages in the corresponding language, if the resource file for the preferred language exists. (Penberthy 2013, chapter 3, Objective 3.2: Plan and implement globalization and localization, cited 9.12.2015.)

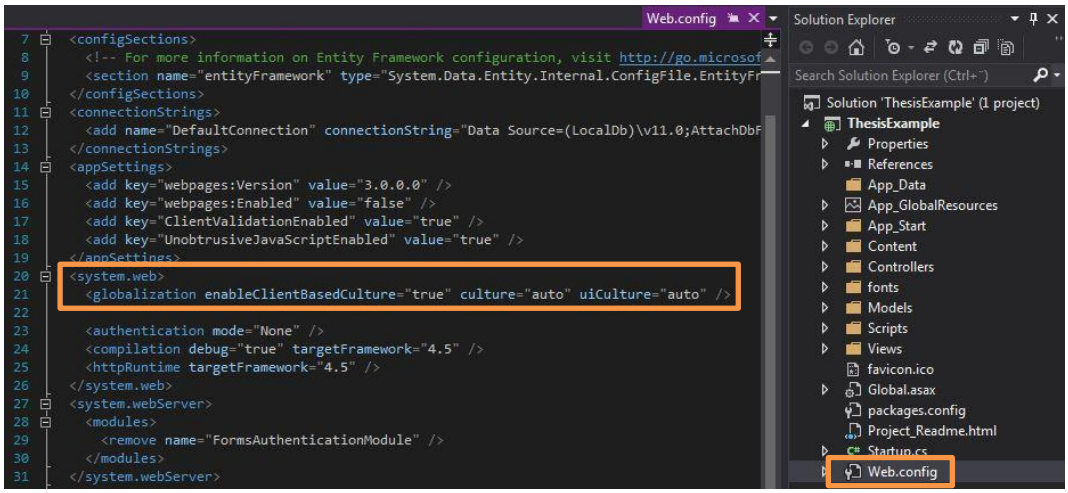


FIGURE 5. Enabling globalization in the Web.config

Secondly, add the *App_GlobalResources* folder to the application (Microsoft Developer Network 2015e, cited 9.12.2015). For example, I added the folder to a demo application by right-clicking on the application folder *ThesisExample* (see figure 5) to trigger a list; then, hovering on the "Add" in the list to open a sub-list; again, in the sub-list, hovering to "Add ASP.NET Folder" to display yet another sub-list; finally, from the latter list, selecting "App_GlobalResources".

Thirdly, create a default resource file and store key-value pairs to the file (Microsoft Developer Network 2015e, cited 9.12.2015). The application utilizes the default file when there is no file matching the language or locale requested by the user. For creating a default resource file, first, right-click on the *App_GlobalResources* folder to open a list; second, hover on the "Add" in the list to open a sub-list; third, from the sub-list, choose "Resources File", which will trigger a dialog box requesting a name for the file. Finally, enter a name or use the suggested name "Resource" and click "OK" will add a default resource file to the *App_GlobalResources* folder. After the file is created, all the words which need to be localized can be stored to the default file in key-value pairs, i.e. storing a word as value and give it a name for referencing later. For example, an application displays the word "Welcome!" when a user logs in. The word needs to be stored in the default resource file before localization. Figure 6 illustrates how to store "Welcome!" in the default resource file.

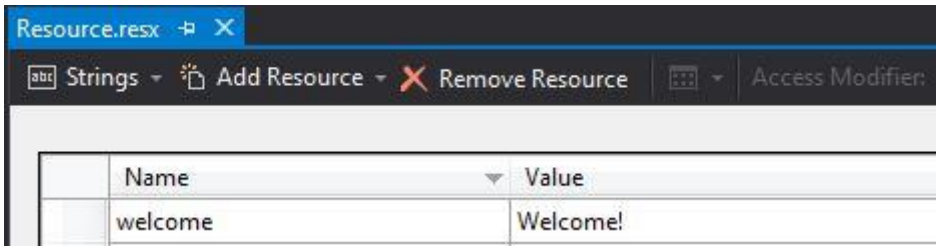


FIGURE 6. Storing a string in the default resource file

Fourthly, create a resource file containing the same keys but translated values in a specific language. An easy way for creating a new resource file containing the same keys is duplicating the default resource file and adding a language or language-locale suffix to the new file. This step can be repeated for each target language. (Microsoft Developer Network 2015e, cited 9.12.2015.) For example, for localizing the demo application in Finnish, I duplicated the default resource file and renamed it as *Resource.fi.resx.* After creating a language or language-locale specific resource file, the original words in the "Value" column can be translated into target language. Continuing the "Welcome!" example, a resource file has been created for Finnish localization; the file contains the same key "welcome" but the original word "Welcome!" in the "Value" column has been replaced with the corresponding Finnish word "Tervetuloa!" (see figure 7).

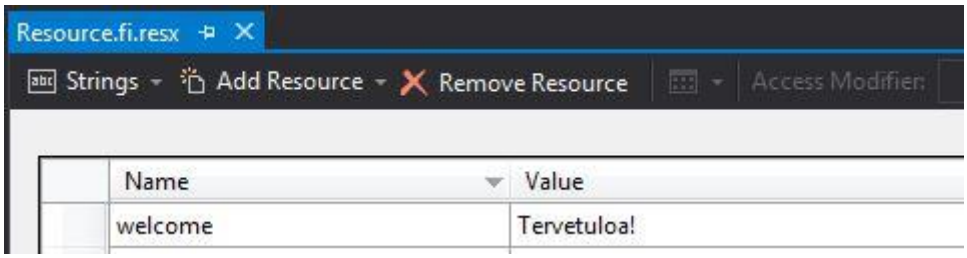


FIGURE 7. A resource file for Finnish localization

Finally, although not mandatory, adding *@using Resources* to the beginning of a view makes referencing to resources a bit easier. For example, without the statement, one would have to type *@Resources.Resource.welcome* to use the resource in a view whereas using the name space *Resources* explicitly in the view, one only needs to type *@Resource.welcome* instead.

After the resource files are created, the original words used in HTML documents such as in headers, labels, placeholders and buttons should be replaced with corresponding resources for the globalization to take effect. Figure 8 exemplifies the use of resources in a Razor HTML document. The original words "Home Page" and "Welcome!" have been replaced by corresponding resources. In this example, there are two resource files: one in English as default

and another one in Finnish. Finnish text will be displayed, if it is the most preferred language in the browser's language setting; otherwise, English text will be displayed.

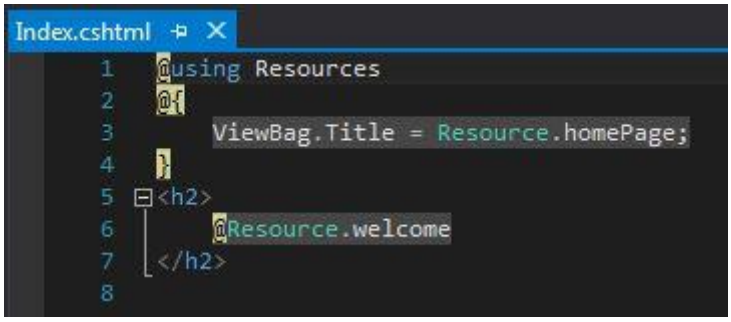


FIGURE 8. Using resources in a view

3.5 ASP.NET MVC5 Request.Form and FormCollection

The strongly typed view and model binding are two of the main features of ASP.NET MVC framework. The former defines a view model, i.e. the type of data object for a view, and the latter constructs a view model object with the data returned from a view. Both features are fully supported by Visual Studio IntelliSense and compile-time checking. IntelliSense can provide the developer with a list of the properties of a view model when the developer accesses the object, whereas compile-time checking alerts the developer if the property she/he tried to access does not exist in the view model. (Galloway et al. 2014, chapter 3, Strongly typed views, chapter 4, Model binding; Microsoft Developer Network 2015f, cited 11.12.2015; Microsoft Developer Network 2015g, cited 11.12.2015.) Therefore, leveraging both features can reduce runtime error and enhance control and maintenance of data integrity between the view and the controller.

However, for exploiting the merits of the strongly typed view and model binding, design and implementation of strongly typed view models are prerequisites (Galloway et al. 2014, chapter 3, Strongly typed views, chapter 4, Model binding). Furthermore, in a view using a dynamic form, the properties of a view model object should be mapped correctly in the scripts for successful model binding in the controller (Johansson 2015, cited 14.12.2015). Together they would have demanded more time than I had for the thesis project. Furthermore, the primary goal is delivering a prototype containing the most essential views for the commissioner to demonstrate to the stakeholders. Therefore, the design and implementation of the strongly typed view and model binding were left for the future iteration.

Without the presence of the strongly typed view and model binding, the data sent to the controller from a form, particularly a dynamic form, can still be examined to determine whether the form works. There are two basic methods to access the complete collection of form values which has been posted back from a dynamic typed view: the first method is accessing the *Request* object created by ASP.NET MVC framework, and the second by passing an instance of *FormCollection* class as a parameter to an *ActionResult* method which handles the HTTP POST request (Allen 2009, cited 15.12.2015; Gnazzo 2015, cited 15.12.2015).

The first approach utilizes a default feature of ASP.NET MVC framework. The framework creates for each HTTP request, a *Request* object, which is a property of *Controller* class and contains client-side information including the data which has been posted back from a form. In the *ActionResult* method which handles the HTTP POST request, the postback value for each input field in a form can be fetched from *Request.Form,* a property which stores postback form values as key-value string pairs. (Microsoft Developer Network 2015h, cited 15.12.2015; Microsoft Developer Network 2015i, cited 15.12.2015; Microsoft Developer Network 2015j, cited 15.12.2015.) As the postback values are stored as strings, conversion is needed before they can be assigned to non-string type variables in the *ActionResult* method (Gnazzo 2015, cited 15.12.2015).

The second approach instantiates an object from *FormCollection* class, which contains only the postback form values in key-value string pairs (Microsoft Developer Network 2015k, cited 17.12.2015; Microsoft Developer Network 2015l, cited 17.12.2015). Like *Request.Form*, conversion is also required before assigning a postback value to a non-string type variable. But, unlike the *Request* object, the framework does not instantiate a *FormCollection* object automatically. Instead, the developer should pass a *FormCollection* type variable as a parameter to the *ActionResult* method which handles the HTTP POST request; as a result, the framework's model binder will create a *FormCollection* object from postback values (Galloway et al. 2014, chapter 4, Model binding).

At first glance, the first approach might appear straightforward as the *Request* object has been automatically created. However, the second approach suits the design in which the *ActionResult* methods handling HTTP GET and POST for the same view share the same name. In addition, the *FormCollection* object can be replaced later with a view model object for intended model binding. Finally, the advantage of the second approach might become more apparent when

mimicking the *Request.Form* property in the unit testing. Instead of mocking up the *Request* object and then the property, a *FormCollection* object containing testing values can be instantiated to represent the postback values in the *Request.Form* property (Microsoft ASP.NET Forums 2008, cited 18.12.2015; Hammarberg 2009, cited 18.12.2015; Harford 2013, cited 18.12.2015).

3.6 ASP.NET MVC5 web deployment

Testing the prototype on the tablet was essential as the tablet is one of the target devices besides the desktop and laptop. Although there are free mobile device simulators and emulators on the Internet, testing on the real device could still reveal bugs, responsiveness and other user experience issues which would otherwise remain unnoticed (see Mooney 2013, cited 22.12.2015; Google 2015, cited 22.12.2015; Looper 2015, cited 22.12.2015; Microsoft ASP.NET 2015, cited 22.12.2015; Microsoft Developer Network 2015n, cited 22.12.2015; Mobile Joomla! 2015, cited 22.12.2015). Since the company has its own wireless local area network (WLAN), it provided the project with an ideal environment for testing the prototype on the real mobile device.

Before an ASP.NET MVC5 application can be tested on a WLAN, the application should be deployed to a machine on the network, which could be the developer's machine. To succeed in the deployment, correct configurations in the machine's operating system and in the Visual Studio solution are prerequisites. Since Windows 7 was the operating system and Visual Studio 2013 was the integrated development environment (IDE) I used for creating the prototype, I will present the necessary configurations in Windows 7 in this subchapter and the configurations in the Visual Studio solution in the subchapter 4.6.

The Windows 7 machine which will host an ASP.NET MVC5 application requires the Internet Information Services (IIS) feature to be enabled (Dykstra 2015, cited 5.1.2016). IIS is an interface for managing websites and applications on a Windows server (Techopedia 2015, cited 5.1.2016). After enabling IIS, the developer should check whether the IIS application pool contains .NET Framework 4, which is required for hosting ASP.NET MVC5 applications (Dykstra 2015, cited 5.1.2016).

The following steps show how to enable IIS feature on the Windows 7 machine. Firstly, open the “Turn Windows features on or off” box by clicking on the “Start” icon, typing “Turn Windows features on or off” into the “Search programs and files” field and then hitting “Enter”. Secondly, select the check box “Internet Information Services”. A blue square will appear in the check box after the check box has been selected (see figure 9). Thirdly, check “.NET Extensibility”, “ASP”, “ASP.NET” for hosting ASP.NET applications; “ISAPI Extensions” and “ISAPI Filters” will be automatically selected after “ASP” and “ASP.NET” are checked (see figure 10). Finally, click “OK” to complete the configuration. (Dykstra 2015, cited 5.1.2016.)

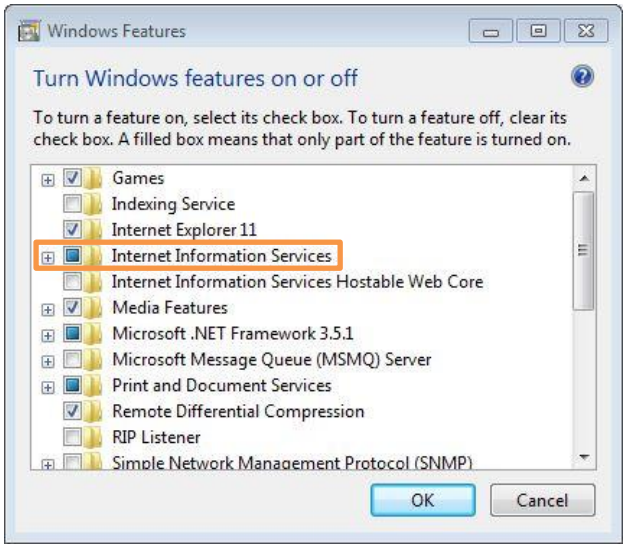


FIGURE 9. Selecting "Internet Information Services" check box

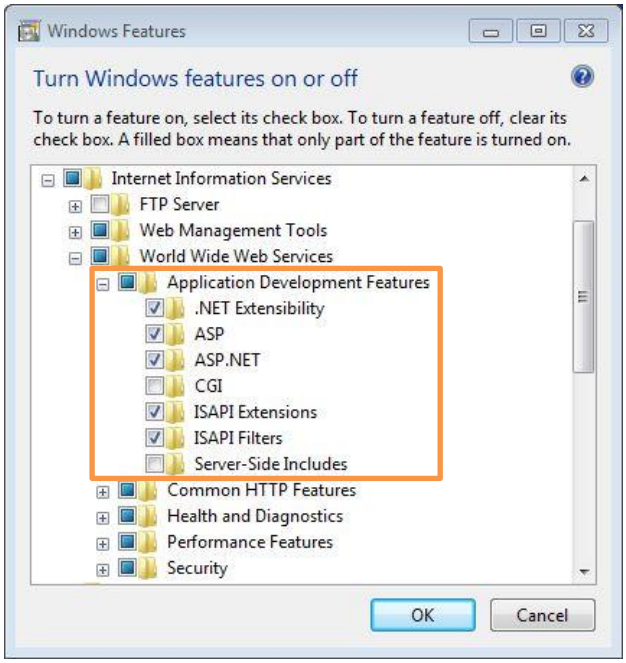


FIGURE 10. Configuring "Application Development Features" for hosting ASP.NET applications

After the IIS feature is enabled, a tool named “Internet Information Services (IIS) Manager” should be available for managing websites and applications. If the tool is not listed in the “Start” menu, it should be found after typing “iis” in the search field.

Manually registering .NET Framework version 4 with IIS will be necessary, if the framework is installed to the machine before IIS feature is activated (Microsoft Developer Network 2016, cited 5.1.2016). For example, as a result of installing Visual Studio 2013 before turning on IIS feature, IIS will not automatically register the version 4 in the application pools. Figure 11 depicts the location of IIS “Application Pools” and the use of .NET Framework version 2 by default.

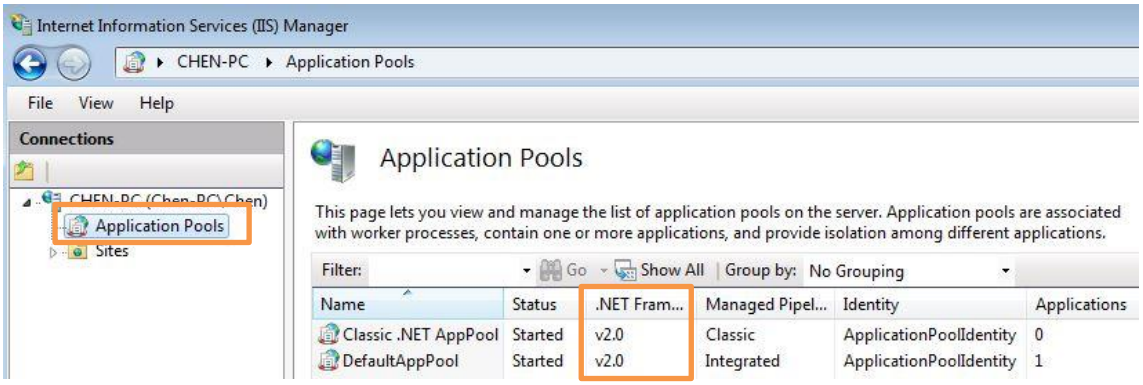


FIGURE 11. Default IIS Application Pools using .NET Framework version 2

Registering .NET Framework version 4 with IIS takes three steps. Firstly, open “Command Prompt” as administrator. Secondly, navigate to the directory of .NET Framework v4.0.30319. Thirdly, register the framework with IIS by *aspnet_regiis.exe -i* command (see figure 12). (Dykstra 2015, cited 5.1.2016.)

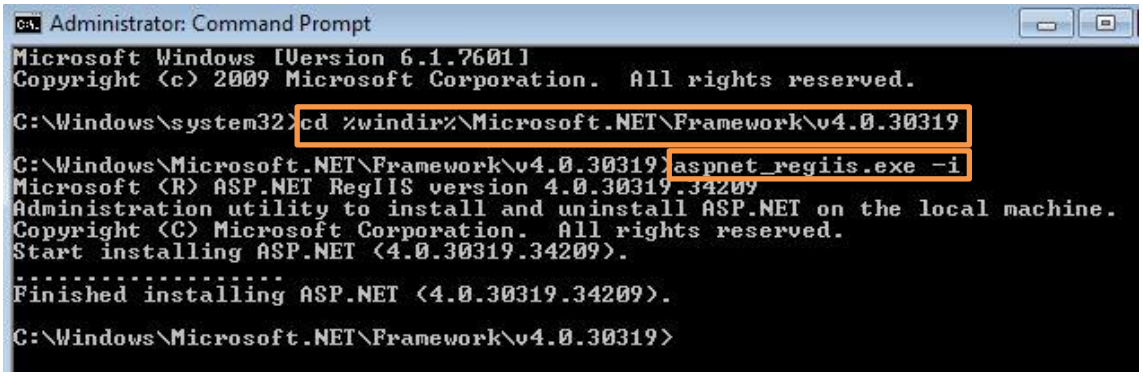


FIGURE 12. Registering .NET Framework version 4 with IIS in the Command Prompt

After registering .NET Framework version 4 with IIS, restart the IIS manager and navigate to the "Application Pools". It should now display four application pools; all of them should be using .NET Framework version 4 (see figure 13). At this point, the IIS manager is ready to host ASP.NET MVC5 applications in its "Default Web Site" folder.

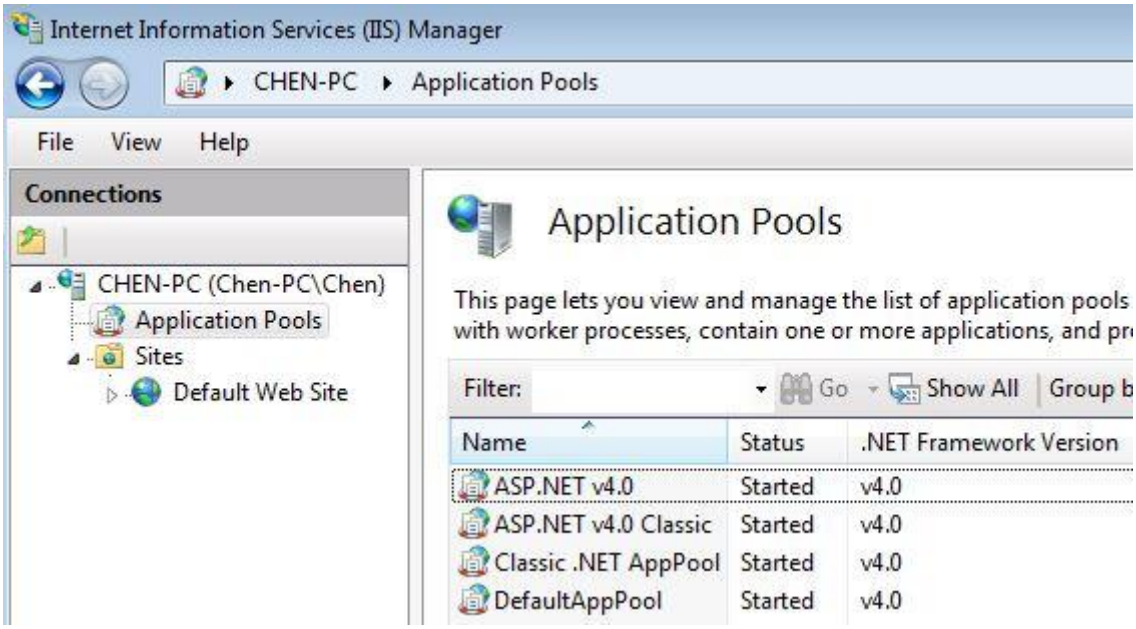


FIGURE 13. The IIS Application Pools after registering .NET Framework 4

4 IMPLEMENTATION

Proactive and regular communication with the commissioner and the instructing teacher facilitated the development of a PHR prototype. During the 13-week development phase, I had regular meetings with the commissioner and other members of the company. In the meetings, I presented my latest work and received feedback and further requirements from them. In addition to the meetings, I turned to the commissioner for advice on urgent issues. Meanwhile, I sent an email once a week to the instructing teacher to keep him updated on the development progress and sometimes, to seek technical advice. The communication helped me clarify the requirements, improve the prototype and overcome programming issues.

User interface (UI) was the focus of the development. I implemented web pages using the ASP.NET MVC5 framework for the use cases given by the commissioner. JavaScript and jQuery were used extensively to create interactive pages and demo data. Much effort had been made to implement a configurable dashboard, a step-by-step form, interactive charts and sortable tables. In addition, testing the pages on different devices was carried out frequently to identify bugs and responsiveness issues which could not be noticed on my development machine. Finally, Finnish localization was implemented in the last couple of weeks of the development phase.

This chapter records how the major UI components were implemented and how certain features in ASP.NET MVC5 framework and Visual Studio 2013 were utilized for creating reusable resources and testing. To protect the company's business secrets, the implemented user interface is concealed. Instead, web pages have been created to illustrate the main ideas of the implemented components. After that, details are given on creating reusable UI resources, implementing Finnish localization and enabling testing on different devices. For clarity, italic type is used to indicate HTML tags, XML tags, the names of folders and files in a Visual Studio solution, the names of classes and methods, and the statements of programming languages.

4.1 Creating HTML elements dynamically

Two main dynamic features I implemented by JavaScript and jQuery during the development phase were a dashboard and a step-by-step form. The dashboard can display relevant

information according to the user's configuration. The step-by-step form, on the other hand, breaks a complicated form down into sequential steps. Each step contains a group of related input fields.

The main ideas of the implemented dashboard are illustrated in figure 14 & 15. The dashboard enables the user to choose relevant content according to her/his need and thus avoids overloading the user with irrelevant information.

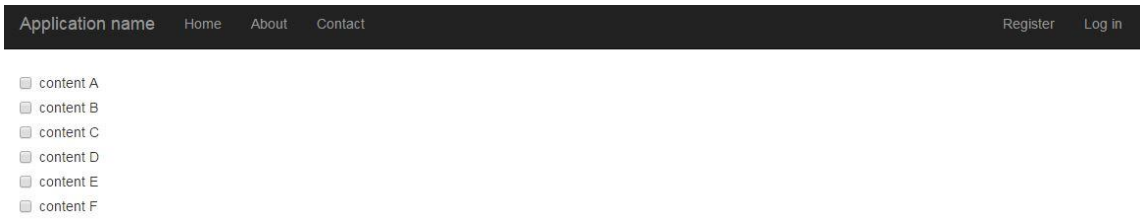


FIGURE 14. The view before a user selects any content

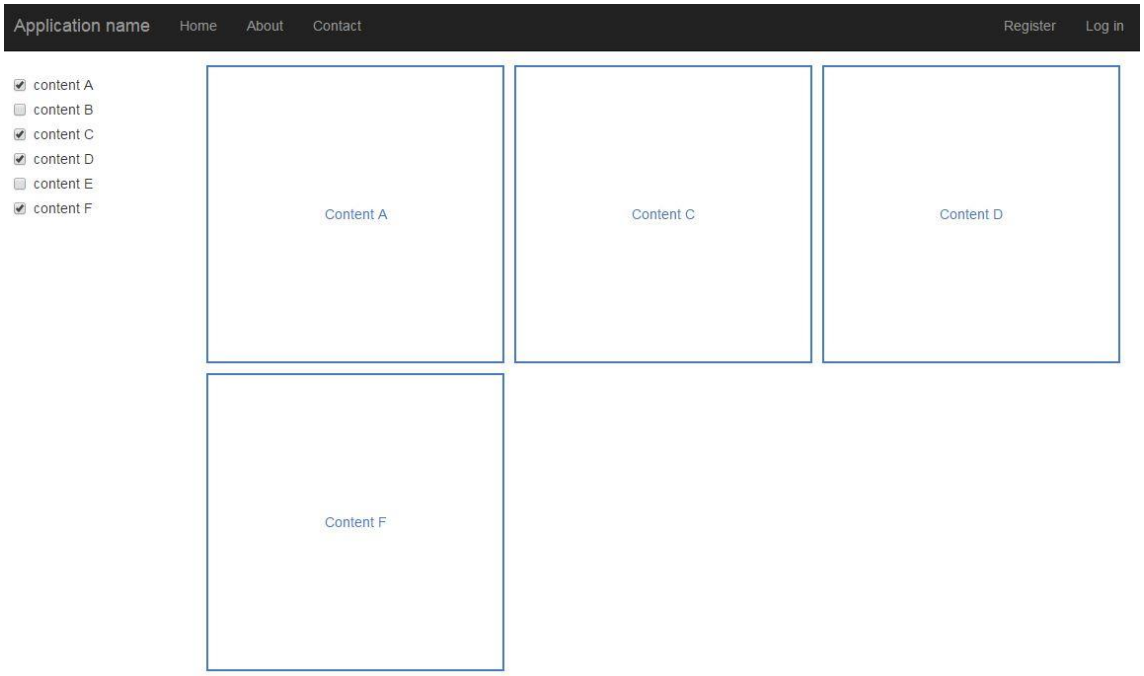


FIGURE 15. The view after a user selected the content needed

The selection can be loaded from a database or carried out by the user during runtime. In either way, the selected content and the HTML elements for displaying the content do not exist in the original HTML document; instead, the elements are created and the content is rendered by JavaScript and jQuery. For example, the content boxes in figure 15 do not exist in the original HTML document; instead, each box is created in the <*script*> tag after the user checked the

corresponding checkbox. If the user unchecks a checkbox, the corresponding content box will also be removed from the page. The code for this example is shown in figure 16.

```
        <form>
            <div class="checkbox">
                <label>
                    <input type="checkbox" value="A"/> content A
                </label>
            </div>
            <div class="checkbox">
                <label>
                    <input type="checkbox" value="B"/> content B
                </label>
            </div>
            <div class="checkbox">...</div>
            <div class="checkbox">...</div>
            <div class="checkbox">...</div>
            <div class="checkbox">...</div>
        </form>
    </div>
    <div id="containerForTailoredContent" class="col-sm-10">
        <!--No elements in the orignal HTML document-->
    </div>
</div>

@section scripts{
<script>
    $(document).ready(function () {
        //detect if a checkbox is being checked or unchecked
        $('input').change(function () {
            //get the checkbox value
            var checkboxValue = $(this).val();
            //if the checkbox is checked
            if ($(this).prop('checked')) {
                //create a square box
                //and display a title using my own JavaScript function
                var selectedContent = createContent(checkboxValue);
                //insert the new element to the DOM
                $(selectedContent).appendTo('#containerForTailoredContent');
            }
            else
            {
                //if the checkbox is being unchecked,
                //remove the HTML element and content from the DOM
                $('#' + checkboxValue).remove();
            }
        });
    });
</script>
}
```

FIGURE 16. Create HTML elements in <script>

In addition to the dashboard, a step-by-step form which makes a complicated task easier for the user was implemented. The task requires the user to complete a form containing at least 30 input fields. To make the form user-friendly, the commissioner designed sequential steps and a group of related input fields for each step. The design takes the user step-by-step to complete the otherwise overwhelming form. Figure 17-19 illustrate a simplified version of the implemented form.

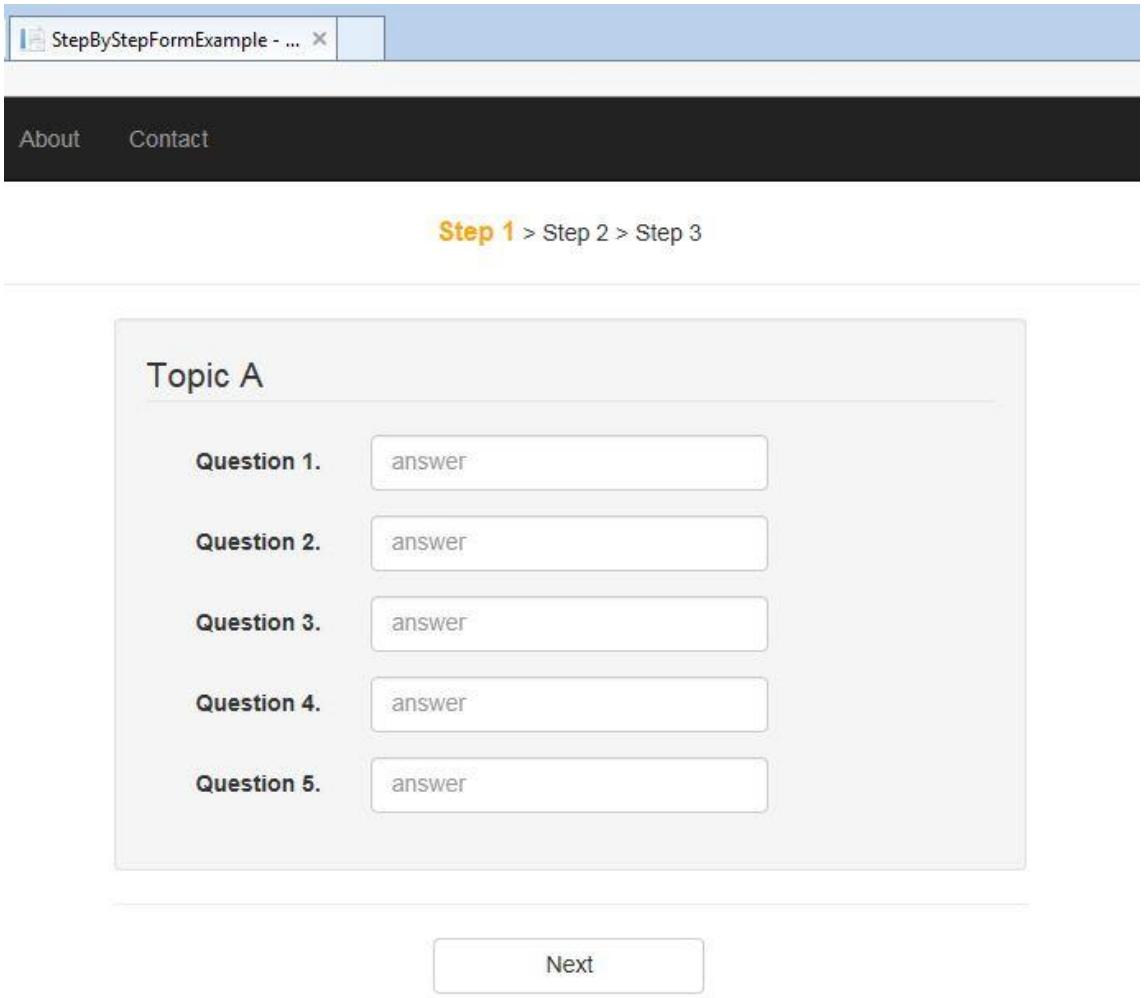


FIGURE 17. Step 1 displays question 1-5 and the button "Next"

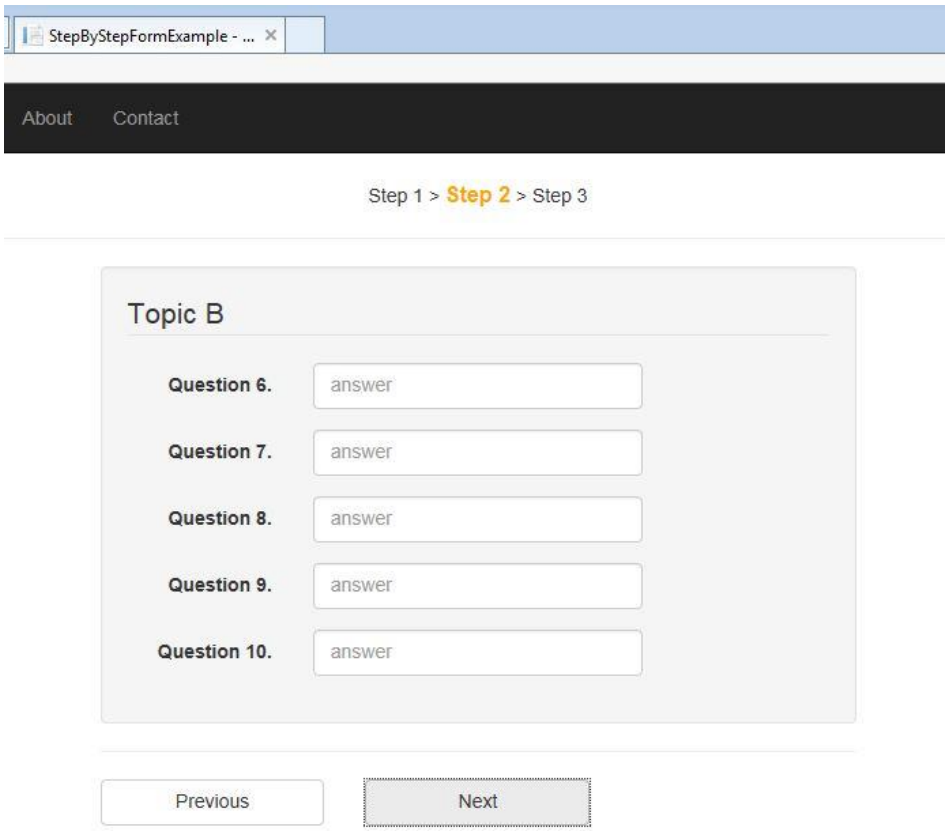


FIGURE 18. Step 2 shows question 6-10 and an additional button "Previous"

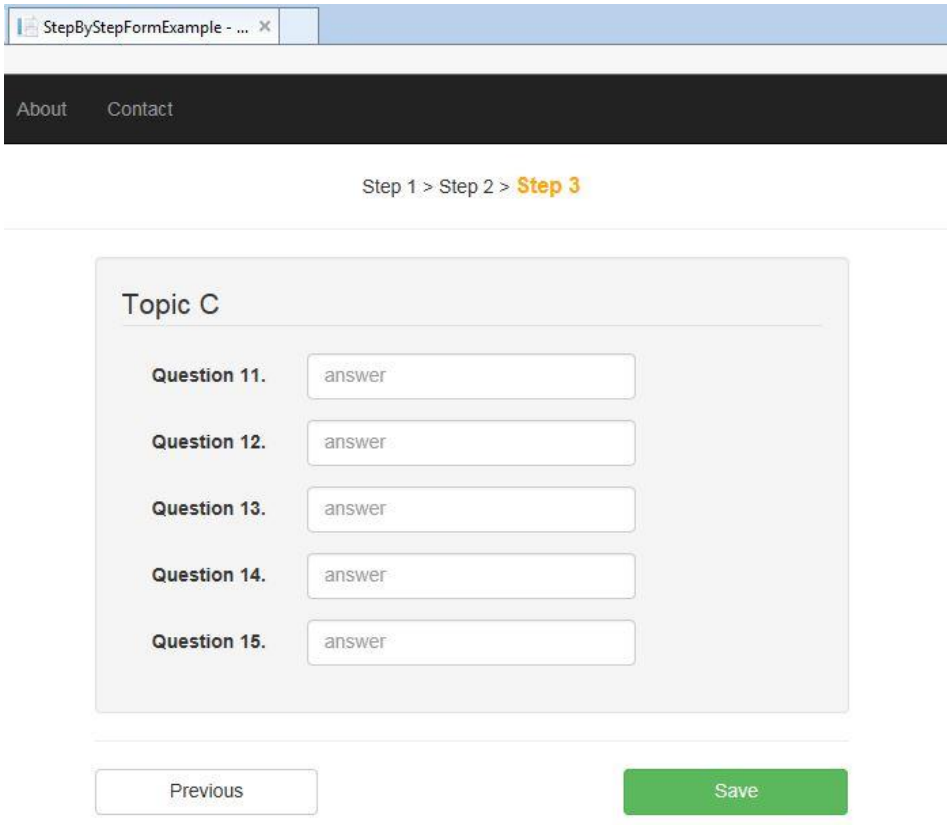


FIGURE 19. Step 3 displays question 11-15, hides the button “Next” and shows the button "Save"

The major difference between the form and the dashboard is that the HTML document of the form contains most of the HTML elements already, but displays only a part of the form to the user for each step. For example, the simplified version contains 15 questions in total but displays only a third of the questions in each step. When the user clicks the button “Next” or “Previous”, the questions and answers entered in the current step will be hidden, and the questions in the next step will be shown. For each button, a jQuery method *click()* is used to detect when the user clicks on it. Inside the method handler, jQuery selectors are used to find the group of HTML elements for the current step and for the next step, which can be a step forward or backward; then jQuery methods *addClass()* and *removeClass()* are used to hide the elements of the current step and to display the elements of the next one. The code inside the two *click()* handlers for implementing the simplified example is presented in figure 20 and 21.

```
//When button "Next" is clicked, it will hide the content of current step
//and display the content of next step as well as highlight the next step
$('#btn_next').click(function () {
    var currentStepId = $('.currentStep').attr('id').split('_');
    var nextStepId = 'step_' + (parseInt(currentStepId[1]) + 1);
    $('.currentStep').removeClass('currentStep');
    $('#' + nextStepId).addClass('currentStep');
    var numberOfTotalQuestionGroups = $('.well').length;
    var currentQuestionGroupId = $('div.well:not(.hidden)').attr('id');
    var currentQuestionGroupIndex = $('div.well:not(.hidden)').index();
    var nextQuestionGroupId = $('div.well:not(.hidden)').next().attr('id');
    $('#' + currentQuestionGroupId).addClass('hidden');
    $('#' + nextQuestionGroupId).removeClass('hidden');
    if (currentQuestionGroupIndex === 0) {
        $('#btn_previous').removeClass('hidden');
    }
    else if (currentQuestionGroupIndex === (numberOfTotalQuestionGroups - 2)) {
        $('#btn_next').addClass('hidden');
        $('#btn_submit').removeClass('hidden');
    }
});
```

FIGURE 20. Codes inside the "Next" button click event handler

```
//When button "Previous" is clicked, it will hide the current content
//and display the content of previous step as well as highlight the previous step
$('#btn_previous').click(function () {
    var currentStepId = $('.currentStep').attr('id').split('_');
    var previousStepId = 'step_' + (parseInt(currentStepId[1]) - 1);
    $('.currentStep').removeClass('currentStep');
    $('#' + previousStepId).addClass('currentStep');
    var numberOfTotalQuestionGroups = $('.well').length;
    var currentQuestionGroupId = $('div.well:not(.hidden)').attr('id');
    var currentQuestionGroupIndex = $('div.well:not(.hidden)').index();
    var prevousQuestionGroupId = $('div.well:not(.hidden)').prev().attr('id');
    $('#' + currentQuestionGroupId).addClass('hidden');
    $('#' + prevousQuestionGroupId).removeClass('hidden');
    if (currentQuestionGroupIndex === (numberOfTotalQuestionGroups - 1)) {
        $('#btn_next').removeClass('hidden');
        $('#btn_submit').addClass('hidden');
    }
    else if (currentQuestionGroupIndex === 1) {
        $('#btn_previous').addClass('hidden');
    }

});
```

FIGURE 21. Codes insides the "Previous" button event handler

4.2 Rendering demo data with responsive charts and sortable tables

Chart.js plugin version 1.0.2 can be found and installed by using “Manage NuGet Packages” in Visual Studio. The tool not only installs the plugin itself, but also ensures the solution has other necessary files, such as the right version of jQuery. The following steps show how to install the plugin by “Manage NuGet Package”. Firstly, find “NuGet Package Manager” under “TOOLS” tab in Visual Studio. Secondly, move the mouse cursor over “NuGet Package Manager” to trigger a list, then select from the list “Manage NuGet Packages for Soluction…”. As a result, a window called “Manage NuGet Packages” is displayed. Thirdly, in the window, select “Online” category on the top-left corner; then, type “Chart.js” in the “Search Online” field on the top-right corner. Afterward, search results should be visible in the central column. Fourthly, locate the version 1.0.2 of the plugin from the results. Once found, an “Install” button will also be visible on the selected result if the plugin has not been installed. Finally, click the button. Visual Studio will install the plugin and its minified JavaScript file to the *Scripts* folder in the solution. If the plugin will be used in more than one view in the solution, it should be referred in the layout view.

In the prototype, multi-line charts were implemented to indicate the progress of the user’s health conditions. The charts will be redrawn according to the day range selected by the user, e.g. a week or 30 days. If there are missing values between two dots, the dots will still be connected

with a smooth line. Figures 22 and 23 illustrate the main ideas of the implemented charts. In figure 22, a tooltip is displayed when the cursor hovers over the second dot on the gray line. Since there is also a value on the blue line for the same day, the tooltip displays the date and both values for comparison. The figure also demonstrates how the second and the fifth values on the gray line are connected by a smooth line, although there are two missing values in between. In figure 23, dots are removed and the tooltip feature is disabled when a longer day range, “Last 30 days” is selected, in order to avoid overlapped dots on a small screen. Unneeded X labels are replaced with empty strings; as a result, there are only four date labels displayed: the first, the tenth, the twentieth, and the last. The vertical lines in the background gird are also removed.

FIGURE 22. Showing the values of last 7 days by default

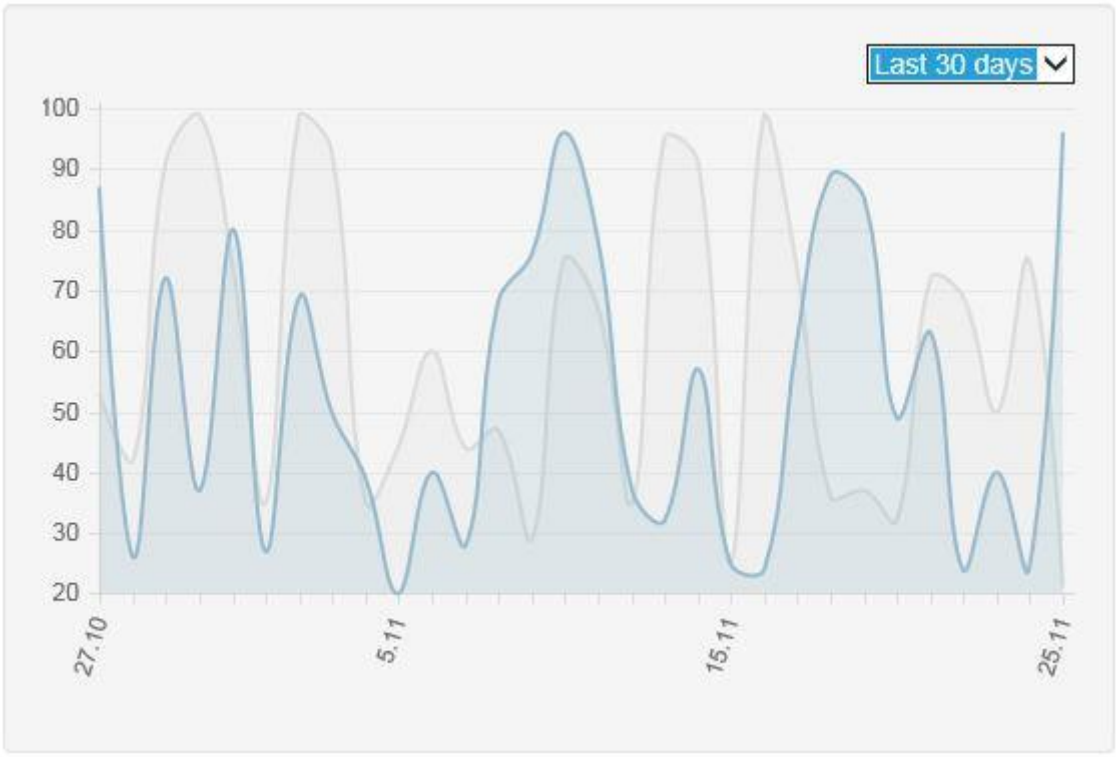


FIGURE 23. Showing the values of last 30 days

There were three ways which I used during prototyping to pass data to the data object of a chart: the first way was assigning a static array, the second one was calling a JavaScript function which returns an array, and the third way was converting a MVC5 *ViewBag* object to a JavaScript array. As the priority of the thesis project is creating a prototype which contains all required views for presentation, work was focused on the view; in other words, most of the demo data in the prototype was generated by JavaScript functions instead of rendered from the controller.

The methods are demonstrated in the code which draws the simplified chart (see figure 24). The data object of the chart requires three arrays: one for the “labels” property, one for the “data” property in the first dataset object, and one for the “data” property in the second dataset object. All three arrays must have the same length. The first array, “last7DaysDateLabels” is extracted from another array “last30DaysDateLabels”, which is a JSON object converted from a MVC5 *ViewBag* object (see the first highlight box in figure 24). The second array, the “data” array in the first dataset, is a static array; the values inside the array will remain the same unless being changed in the later part of the scripts; each “null” in the array represents a missing data (see the second highlight box in the figure). On the contrary, the third array, the “data” array in the second dataset is generated by *generateDemoData*(), a function I wrote which will create data randomly every time when the page is loaded (see the third highlight box in the figure).

```
28          //assign a controller created string array of date label
29          //to a JavaScript variable
30          var last30DaysDateLabels = @Html.Raw(Json.Encode(ViewBag.dateLabels));
31          //extract the last 7 days' date labels to another variable
32          var last7DaysDateLabels = [];
33          for (var i = 6; i >= 0; i--) {
34              last7DaysDateLabels.push(last30DaysDateLabels[(last30DaysDateLabels.length - 1) - i]);
35          }
36          var lineChart1Data = {
37              //assign an array variable to labels
38              labels: last7DaysDateLabels,
39              datasets: [
40                  {
41                      label: "My First dataset",
42                      fillColor: "rgba(220,220,220,0.2)",
43                      strokeColor: "rgba(220,220,220,1)",
44                      pointColor: "rgba(220,220,220,1)",
45                      pointStrokeColor: "#fff",
46                      pointHighlightFill: "#fff",
47                      pointHighlightStroke: "rgba(220,220,220,1)",
48                      //static array
49                      data: [65, 59, null, null, 56, 55, 40]
50                  },
51                  {
52                      label: "My Second dataset",
53                      fillColor: "rgba(151,187,205,0.2)",
54                      strokeColor: "rgba(151,187,205,1)",
55                      pointColor: "rgba(151,187,205,1)",
56                      pointStrokeColor: "#fff",
57                      pointHighlightFill: "#fff",
58                      pointHighlightStroke: "rgba(151,187,205,1)",
59                      //assigning data array by a function
60                      data: generateDemoData(7)
61                  }
62              ]
63          };
64          var lineChart1Options = {
65              responsive: true
66          };
67          var lineChart1Ctx = document.getElementById("lineChart_1").getContext("2d");
68          var lineChart1 = new Chart(lineChart1Ctx).Line(lineChart1Data, lineChart1Options);
```

FIGURE 24. Three ways to pass data to the data object

Before continuing, I would like to point out a false warning of syntax error in the older version of Visual Studio 2013 prior to Update 5 when converting a *ViewBag* object to a JSON object in the <script> tag (see figure 25). The syntax in the figure calls a system method *Json.Encode()* to create a JavaScript Object Notation (JSON) object from a MVC5 *ViewBag* object (Microsoft Developer Network 2015a, cited 26.11.2015), then uses another system method *Html.Raw()* to prevent the code in the object from being replaced with HTML entities (Kennedy 2012, cited 26.11.2015; Microsoft Developer Network 2015b, cited 26.11.2015; W3Schools 2015e, cited 26.11.2015); specifically, to prevent the quotation marks in the JSON object from being replaced with the HTML entity “"” (W3Schools 2015f, cited 26.11.2015). The output of the syntax produces a valid JavaScript array containing date labels of last 30 days.

```
var last30DaysDateLabels = @Html.Raw(Json.Encode(ViewBag.dateLabels));
```

FIGURE 25. A false indication of syntax error in Visual Studio 2013 version prior to Update 5

In the older version, a red mark would appear in the end of the statement as shown in the figure. In addition, a syntax error warning message would be displayed in the “Error List”. However, I can still build and publish the project. After investigating the situation, it appeared that many people used an older version of Visual Studio experienced the same issue, and the issue seemed to be a bug in the older versions (StackOverflow 2012a, cited 26.11.2015; StackOverflow 2012b, cited 26.11.2015; Microsoft Visual Studio 2012, cited 26.11.2015; StackOverflow 2013, cited 26.11.2015). The false warning disappeared after I installed Visual Studio 2013 Update 5.

Let us return to the dropdown list of the simplified chart. To redraw the chart when a user selects a different day range, a jQuery method *change()* is used. The method detects if a different day range has been chosen by the user; consequently, it will generate the same amount of data as the selected range and assign the data to the existing data object. In addition, if the user selects “Last 30 days”, code has been written to display only the first, the tenth, the twentieth, and the last date; as well as to configure the chart option object to show neither dots nor tooltips. In the end of the method scope, a Chart.js method *destroy()* is used to discard the old chart. Then, a new chart is created with the updated data and chart option object, and the new chart is assigned to the same variable containing the old chart object, thus replacing the old one. The code inside the *change()* method is shown in figure 26.

```
        $('select').change(function(){
            //get the selected day range
            var selectedDayRange = parseInt($('select').val());
            //prepare the data object and chart option object according to the selected day range
            switch(selectedDayRange){
                case 7:
                    lineChart1Data.labels = last7DaysDateLabels;
                    lineChart1Data.datasets[0].data = generateDemoData(7);
                    lineChart1Data.datasets[1].data = generateDemoData(7);
                    lineChart1Options = {
                        responsive: true
                    };
                    break;
                case 30:
                    //extract date labels of the first, 10th, 20th and the last date
                    var dateLabelsOfBeginningAndEvery10thDay = [];
                    for(var i = 0; i < 30; i++){
                        if(i=== 0 || i === 9 || i === 19 || i === 29 )
                        {
                            dateLabelsOfBeginningAndEvery10thDay.push(last30DaysDateLabels[i]);
                        }
                        else
                        {
                            dateLabelsOfBeginningAndEvery10thDay.push('');
                        }
                    }
                    lineChart1Data.labels = dateLabelsOfBeginningAndEvery10thDay;
                    lineChart1Data.datasets[0].data = generateDemoData(30);
                    lineChart1Data.datasets[1].data = generateDemoData(30);
                    lineChart1Options = {
                        responsive: true,
                        showTooltips: false,
                        scaleShowVerticalLines: false,
                        pointDot:false
                    };
                    break;
            }
            //discard the current chart and replace it with a new one
            lineChart1.destroy();
            lineChart1 = new Chart(lineChart1Ctx).Line(lineChart1Data, lineChart1Options);
        });
```

FIGURE 26. Redraw the chart according to a user's selection

Although a line chart enables the user to instantly see the trend of her/his health conditions, the chart might become less effective if it displays too many lines and too large amount of data in a small screen. In addition, a line chart is useful only for presenting numeric data. For presenting information which contains both numbers and text, such as a list of messages from different senders, a table would be adequate. Better yet, a sortable table provides the user with means for managing her/his messages.

For implementing sortable tables in the prototype, the version 2.0.5b of the jQuery plugin "tablesorter" was used. Since the version cannot be found by the online search in the "Manage NuGet Packages" in the Visual Studio, it should be downloaded from its GitHub repository and included to a solution manually. For utilizing the plugin and its default theme, one would need the following files from the downloaded and extracted folders: "jquery.tablesorter.js" from "tablesorter-master" folder; "style.css", "asc.png", "bg.png" and "desc.png" from either "green" or "blue" folder,

which can be found in “themes” folder. Then, in a Visual Studio solution, click on a preferred folder for storing the files, and use the shortcut “Shift + Alt + a” to find and include the files in the folder. Repeat the action to include files in a different folder. Following the default folder structure of a MVC5 solution, I added the script file “jquery.tablesorter.js” to the *Scripts* folder, and created a sub-folder under the *Content* folder and added the downloaded PNG files and the CSS file to the sub-folder. After the script file and the CSS file are referred in a layout or a view, the plugin is ready to be used.

Sortable tables were implemented in the prototype for tasks including sorting messages. Figure 27 illustrates a simplified version of an implemented message list. As shown in the figure, the column of message status is sorted in ascending order; whereas the column of received time is sorted in descending order. Therefore, the latest message will always be displayed in the first row when the page is loaded. In addition, since every column in the table is sortable, the table enables the user to sort messages according to her/his specific criteria.

Name	Status	Received Time	Subject
Cindy	new	2015-11-21 18:12	information about health
Daniel	new	2015-11-19 18:45	appointment
Frank	new	2015-10-24 12:45	information about health
Bart	read	2015-11-22 09:12	request
Emily	read	2015-11-08 16:55	request
John	read	2015-10-10 07:45	information about health
Amy	replied	2015-11-24 08:00	appointment
Grace	replied	2015-10-23 11:30	information about health
Harry	replied	2015-10-20 10:45	request
Iris	replied	2015-10-15 08:20	appointment

FIGURE 27. A sortable message list

For configuring the default sorting to sort the “Status” column in ascending order and the “Received Time” in descending order, I passed an object *{ sortList: [[1,0], [2,1]]}* as parameter to the *tablesorter()* method (see figure 28). There are two arrays in the object property “sortList”: the first array instructs the plugin to sort the second column, “Status” in ascending order; whereas the second array instructs the plugin to sort the third column, “Received Time” in descending order.

For both the simplified table and the sortable tables implemented in the prototype, the table rows and demo data were created by JavaScript and jQuery. This way, the demo data does not require a database, and the code for generating the data can be reused. As demonstrated in figure 28, the <*tbody*> element of the simplified table has no content. The content is created in the second <*script*> tag by a custom function *createMessageTable().* After that, an object containing default sorting configuration is passed to the plugin method *tablesorter()* as a parameter. The plugin method is then chained to a jQuery selector which points to the <*table*> element of the simplified table. As a result, the table has rows and becomes sortable when the page is rendered.

```
<div class="row">
    <div class="col-sm-6">
        <div class="table-responsive">
            <table class="table table-bordered tablesorter">
                <thead>
                    <tr>
                        <th>Name</th>
                        <th>Status</th>
                        <th>Received Time</th>
                        <th>Subject</th>
                    </tr>
                </thead>
                <tbody>
                </tbody>
            </table>
        </div>
    </div>
</div>
@section scripts{
<script src="~/Scripts/jquery.tablesorter.js"></script>
<script>
    $(document).ready(function () {
        //my function for creating a table
        createMessageTable();
        //make the table element sortable with default sorting orders
        $('table').tablesorter({ sortList: [[1, 0],[2, 1]] });
    });
</script>
}
```

FIGURE 28. Creating a sortable message list

Finally, since the result of sorting words alphabetically can vary in different languages, using attribute sorting instead of sorting the words in a column can ensure consistent sorting results. For example, although “new”, “read” and “replied” are in ascending order, the order will not hold true when the words are translated into Finnish: “uusi”, “luettu”, “vastattu”. Since the prototype was created in English and translated in Finnish, attribute sorting was implemented for ensuring consistent results. For demonstration, I also implemented attribute sorting in the simplified table

(see figure 29). Firstly, I assign a number to the “data-sort-value” attribute of each cell; “0” represents a new message; “1” represents a read message; and “2” represents a replied message (see the first highlight box in the figure). Secondly, either “new”, “read” or “replied” is displayed in a cell according to the value of its “data-sort-value” attribute (see the second highlight box in the figure). Later, the words can be translated in different languages. This way, no matter in which language the words are rendered, new messages will always appear on the top rows as long as the “Status” column is sorted in ascending order.

```
for (var i = 0; i < senders.length; i++) {
    $('table tbody').append(
        '<tr>' +
            //assign name from "senders" array
            '<td>' + senders[i] + '</td>' +
            //assign numeric value from "messageStatus" array
            '<td data-sort-value="' + messageStatus[i] + '"></td>' +
            //assign YYYY-MM-DD HH:mm date string from receivedTime array
            '<td>' + receivedTime[i] + '</td>' +
            //assign string from "subject" array
            '<td>' + subjects[i] + '</td>' +
        '</tr>'
    );
}
//select all rows in the <tbody> and loop through them
$('table tbody > tr').each(function () {
    //get the value of "data-sort-value" attribute of the second cell in a row
    var dataSortValue = $(this).children('td:nth-child(2)').attr('data-sort-value');
    //display new, read or replied according to the attribute value
    switch (dataSortValue) {
        case '0':
            $(this).children('td:nth-child(2)').html('new');
            break;
        case '1':
            $(this).children('td:nth-child(2)').html('read');
            break;
        case '2':
            $(this).children('td:nth-child(2)').html('replied');
            break;
    }
});
```

FIGURE 29. Assigning a custom value to the "data-sort-value" attribute of each cell in the “Status” column and displaying text in the cell according to its attribute value

4.3 Creating reusable HTML elements

Partial views were utilized in the prototype for reusing HTML elements in different pages while making editing the elements more effective than a copy-paste solution. Before creating and rendering partial views, I usually made a complete page and presented it to the commissioner first, then edited the page according to the commissioner’s feedback. Once the major change was

completed, I moved the part of HTML elements which will be reused to a partial view. After I had created a partial view, I used *Html.Partial()* to render the markup in the original page as well as in other pages which require the same elements.

For one of the use cases of the prototype, I created a partial view to store HTML markup for log entries. The logs were created on one page and read on another. Because the HTML markup for displaying the logs in both pages is identical, the markup was stored in a partial view for reusing. As a result, any change made in the partial view, e.g. using a different CSS class, changing the container id, will be applied to both pages.

To demonstrate the case above, I created two controllers representing two different contexts: *ContextAController* and *ContextBController*. An *Index* page was created for each controller. Logs can be read in both pages, but a new log can only be entered in the *Index* page of *ContextAController* (see figure 30 and 31).

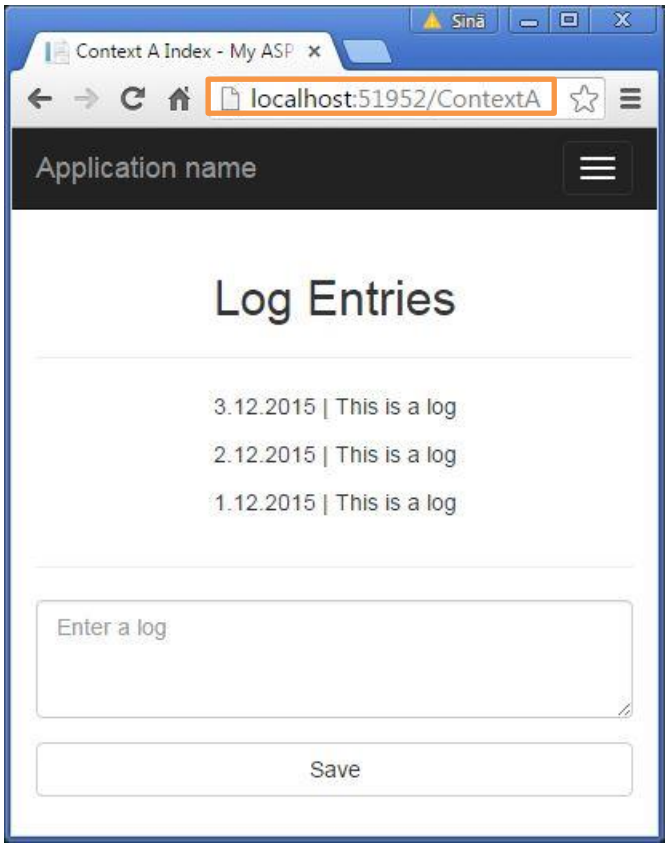


FIGURE 30. The index view of "ContextAController" displays logs and allows entering a new log

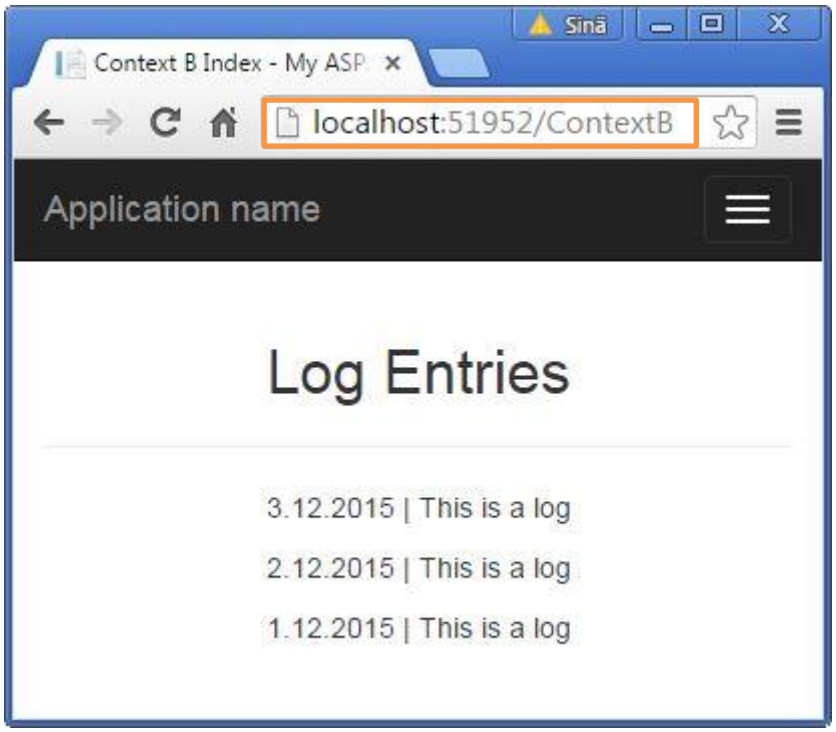


FIGURE 31. The index view of "ContextBController" displays logs only

The title "Log Entries", the horizontal line under the title, and the HTML markup for containing the logs are rendered from a partial view named *_LogEntries* (see figure 32). The partial view was created in the *Views/Shared* folder, so it can be accessed by views (pages) in different folders. Any change made in the partial view will be applied to both *Index* pages.

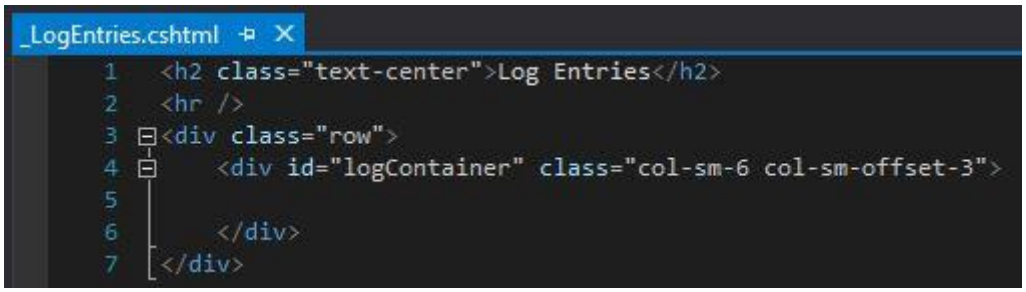


FIGURE 32. The markup in the partial view

Html.Partial() method was used here to render the partial view as in the prototype. In the *Index* page of "ContextAController", a form for entering a new log was added after the partial view method whereas in the *Index* page of "ContextBController", only the partial view was used. The date and text in the logs were generated by a custom JavaScript method *generateDemoLogs()*. See figure 33 and 34 for the use of *Html.Partial()* and for the markup differences between the two *Index* page.

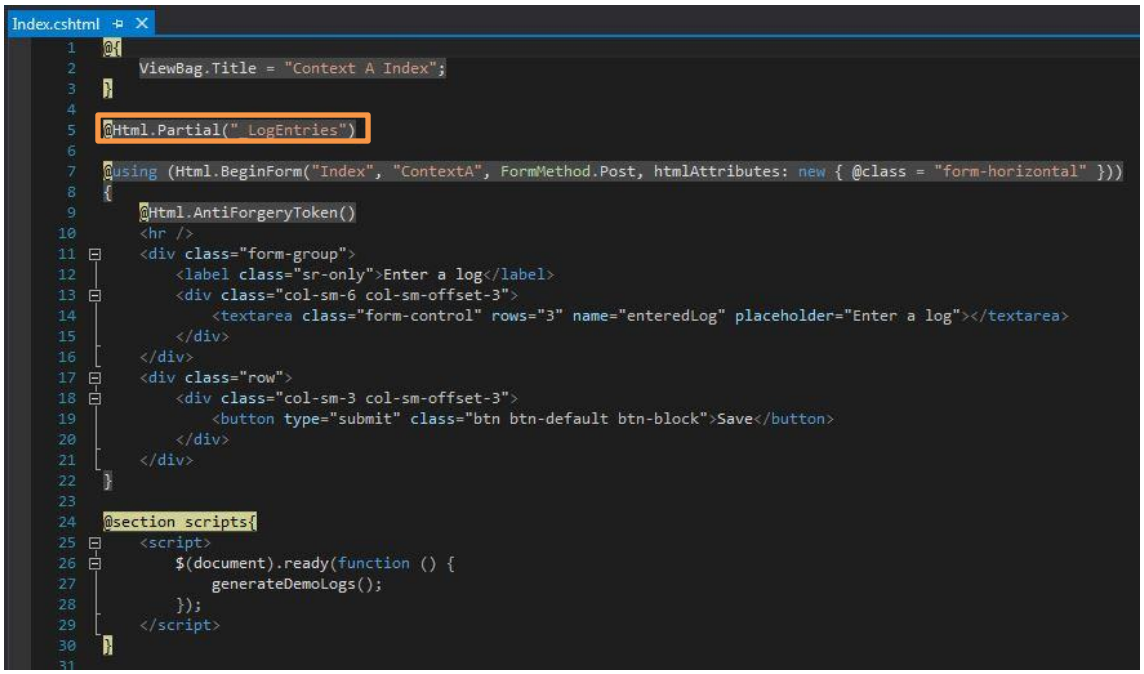


FIGURE 33. The markup in the Index view of context A

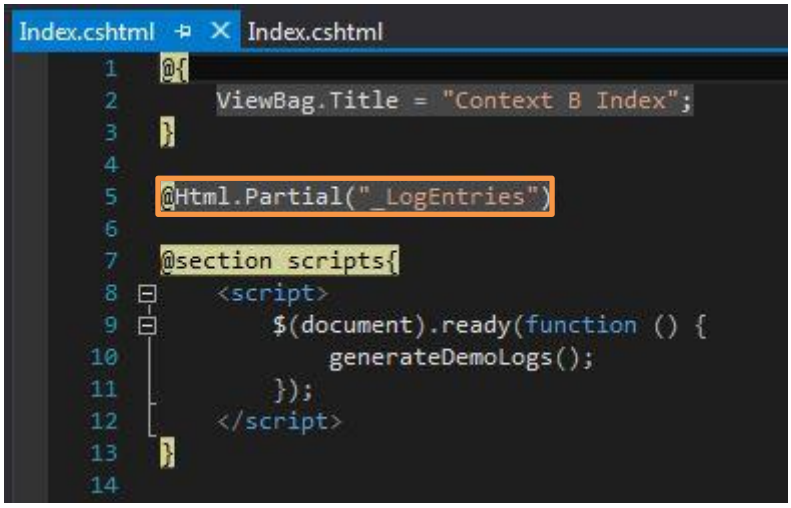


FIGURE 34. The markup in the Index view of context B

The main reasons for using *Html.Partial()* are short syntax and familiarity. The method does not need an additional code block like *Html.RenderPartial()*. Furthermore, as I am more familiar with *Html.Partial(),* I can use it with confidence thus being able to concentrate on the task at hand.

4.4 Implementing Finnish localization

As I am more familiar with English than with Finnish, I created the prototype in English first. Finnish localization was implemented in the last couple weeks of the development phase, after

major changes were completed. For the task, I was able to reuse some Finnish translations from the company's other project. In addition, I looked up general words in Finnish from an online dictionary "Sanakirja.org"; whereas for medical terminology, I consulted an online Finnish medical library "DUODECIM Terveyskirjasto". After I had translated the prototype into Finnish, I asked the commissioner for proofreading and feedback, and made final corrections accordingly.

For demonstrating the use of resource files in localization, I created and utilized two resource files for the simplified sortable table illustrated in 4.2 and the navigation bar above the table. The default resource file is in English; whereas the language specific resource file is in Finnish. As a result, the page will be displayed in either English or Finnish depending on the language setting in the browser (see figure 35 and 36).

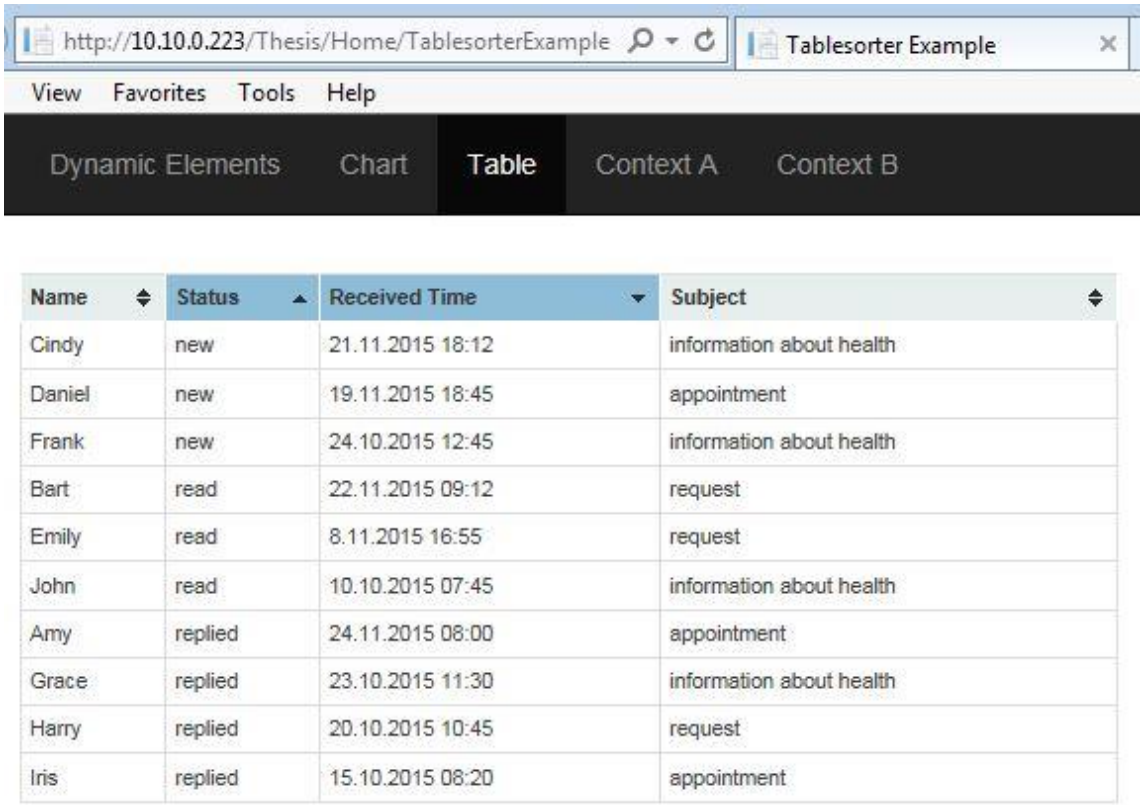
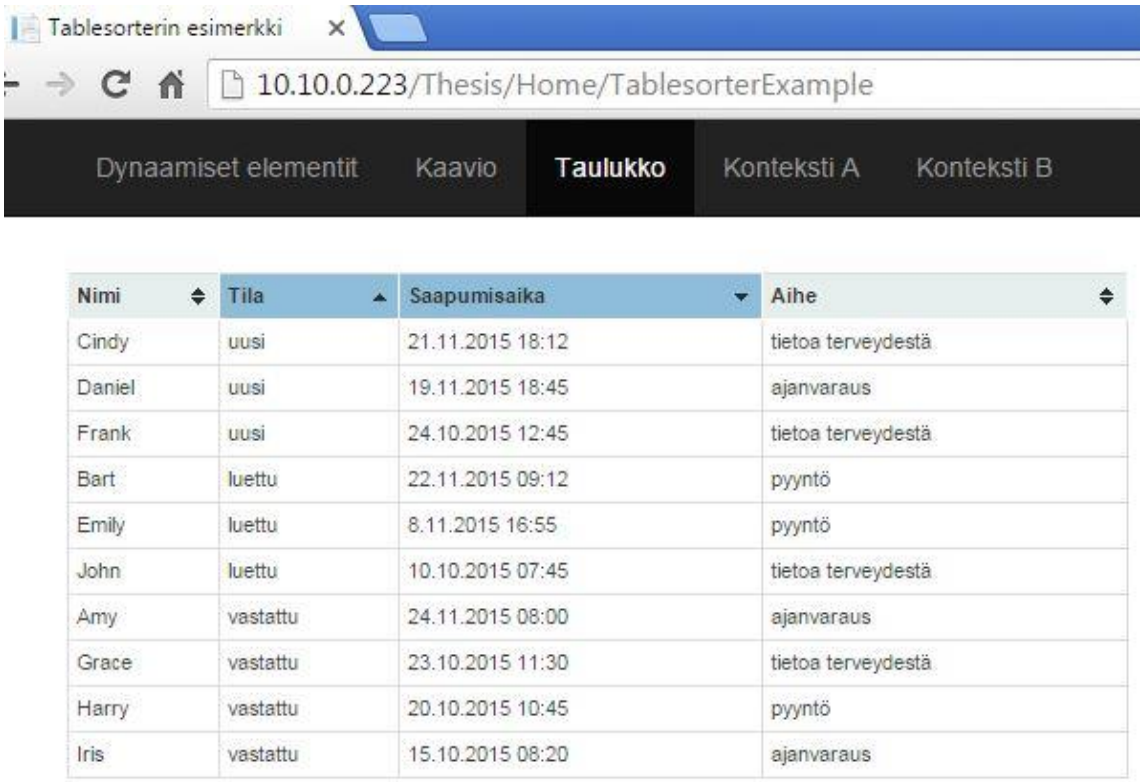


FIGURE 35. The sortable table example in English by default



Nimi	Tila	Saapumisaika	Aihe
Cindy	uusi	21.11.2015 18:12	tietoa terveydestä
Daniel	uusi	19.11.2015 18:45	ajanvaraus
Frank	uusi	24.10.2015 12:45	tietoa terveydestä
Bart	luettu	22.11.2015 09:12	pyyntö
Emily	luettu	8.11.2015 16:55	pyyntö
John	luettu	10.10.2015 07:45	tietoa terveydestä
Amy	vastattu	24.11.2015 08:00	ajanvaraus
Grace	vastattu	23.10.2015 11:30	tietoa terveydestä
Harry	vastattu	20.10.2015 10:45	pyyntö
Iris	vastattu	15.10.2015 08:20	ajanvaraus

FIGURE 36. The sortable table example in Finnish

As shown in figure 36, all the words except the names in the first column are translated into Finnish, as well as the dates are rendered in Finnish format. The text is localized by the Finnish resource file whereas the dates are formatted by *toLocaleDateString()*, a method of the JavaScript Date object.

The resource files can be accessed not only in a Razor HTML document, but also in an external JavaScript file after the resources have been converted to a JSON object. Therefore, dynamically created HTML elements, e.g. the rows in the simplified sortable table, can also leverage the resources for localization.

A basic solution for accessing ASP.NET MVC5 resources in an external JavaScript file was adopted in the prototype. The solution takes two steps: firstly, in the <*script*> tag of a Razor HTML document, create a JSON object which contains the needed key-value pairs from the resources (StackOverflow 2014, cited 8.12.2015); secondly, pass the object as a parameter to a function written in an external JavaScript file which is referred by the Razor HTML document. Instead of passing a complete resource file, the solution allows me to pass only the necessary resources to the function in an external JavaScript file.

The same solution was also used for localizing the dynamically created rows in the simplified sortable table. As highlighted in figure 37, an object called “textResources” is created and stores the necessary key-value pairs from the resources. The object is then passed to *createMessageTable(),* which is a function I wrote in an external JavaScript file.

TablesorterExample.cshtml

```
@using Resources
@{
    ViewBag.Title = Resource.tablesorterExample;
}
<div class="row">
    <div class="col-sm-6">
        <div class="table-responsive">
            <table class="table table-bordered tablesorter">
                <thead>
                    <tr>
                        <th>@Resource.name</th>
                        <th>@Resource.status</th>
                        <th>@Resource.receivedTime</th>
                        <th>@Resource.subject</th>
                        @*<th>Name</th>
                        <th>Status</th>
                        <th>Received Time</th>
                        <th>Subject</th>*@
                    </tr>
                </thead>
                <tbody>
                </tbody>
            </table>
        </div>
    </div>
</div>
@section scripts{
<script src="~/Scripts/jquery.tablesorter.js"></script>
<script>
    $(document).ready(function () {
        $('#navLi_table').addClass('active');
        //store Resource texts in an JavaScript object
        var textResources = {
            appointment: '@Resource.appointment',
            informationAboutHealth: '@Resource.informationAboutHealth',
            newMsg: '@Resource.newMsg',
            readMsg: '@Resource.readMsg',
            repliedMsg: '@Resource.repliedMsg',
            request: '@Resource.request'
        };
        //my function for creating a table
        createMessageTable(textResources);
        //createMessageTable();
        //make the table element sortable with default sorting orders
        $('table').tablesorter({ sortList: [[1, 0],[2, 1]] });
    });
</script>
}
```

FIGURE 37. Using the resource files for localization

After receiving the object “textResources” as a parameter, the function *createMessageTable()* can access the properties of the object in its scope (see highlights in figure 38 and 39). As a result, the rows created by JavaScripts and jQuery can also leverage ASP.NET MVC5 resources for localization.

```
function createMessageTable(resourceObject) {
    var senders = ['Amy', 'Bart', 'Cindy', 'Daniel', 'Em
    var messageStatus = [2, 1, 0, 0, 1, 0, 2, 2, 2, 1];
    var receivedTime = [
        '2015-11-24 08:00',
        '2015-11-22 09:12',
        '2015-11-21 18:12',
        '2015-11-19 18:45',
        '2015-11-08 16:55',
        '2015-10-24 12:45',
        '2015-10-23 11:30',
        '2015-10-20 10:45',
        '2015-10-15 08:20',
        '2015-10-10 07:45'
    ];
    var subjects = [
        resourceObject.appointment,
        resourceObject.request,
        resourceObject.informationAboutHealth,
        resourceObject.appointment,
        resourceObject.request,
        resourceObject.informationAboutHealth,
        resourceObject.informationAboutHealth,
        resourceObject.request,
        resourceObject.appointment,
        resourceObject.informationAboutHealth
    ];
```

FIGURE 38. Accessing a resource object in an external JavaScript file part I

```
    for (var i = 0; i < senders.length; i++) {
        var msgReceivedDateTimeArray = receivedTime[i].split(' ');
        var msgReceivedDate = new Date(msgReceivedDateTimeArray[0]);
        $('table tbody').append(
            '<tr>' +
                //assign name from "senders" array
                '<td>' + senders[i] + '</td>' +
                //assign numeric value from "messageStatus" array
                '<td data-sort-value="' + messageStatus[i] + '"></td>' +
                //assign YYYY-MM-DD HH:mm date string from receivedTime array to data-sort-value
                //display locale date
                '<td data-sort-value="' + receivedTime[i] + '">' +
                    msgReceivedDate.toLocaleDateString() + ' ' + msgReceivedDateTimeArray[1] +
                '</td>' +
                //assign string from "subject" array
                '<td>' + subjects[i] + '</td>' +
            '</tr>'
        );
    }
    //select all rows in the <tbody> and loop through them
    $('table tbody > tr').each(function () {
        //get the value of "data-sort-value" attribute of the second cell in a row
        var dataSortValue = $(this).children('td:nth-child(2)').attr('data-sort-value');
        //display new, read or replied according to the attribute value
        switch (dataSortValue) {
            case '0':
                $(this).children('td:nth-child(2)').html(resourceObject.newMsg);
                break;
            case '1':
                $(this).children('td:nth-child(2)').html(resourceObject.readMsg);
                break;
            case '2':
                $(this).children('td:nth-child(2)').html(resourceObject.repliedMsg);
                break;
        }
    });
    return;
}
```

FIGURE 39. Accessing a resource object in an external JavaScript file part II

4.5 Examining HTML forms which contain dynamically created input fields

Dynamic forms were implemented in the prototype to satisfy the requirement which enables the user to insert new input fields to a form by clicking an “add” button. For detecting the click event, a jQuery method *click()* was used. Once the method is triggered, the HTML elements for an input field will be generated and then inserted back to the form; as a result, a new input field will be inserted to the form every time when the user clicks the “add” button. Figure 40 illustrates a simplified example which inserts a new input field for phone number after the user clicked “Add Phone”.

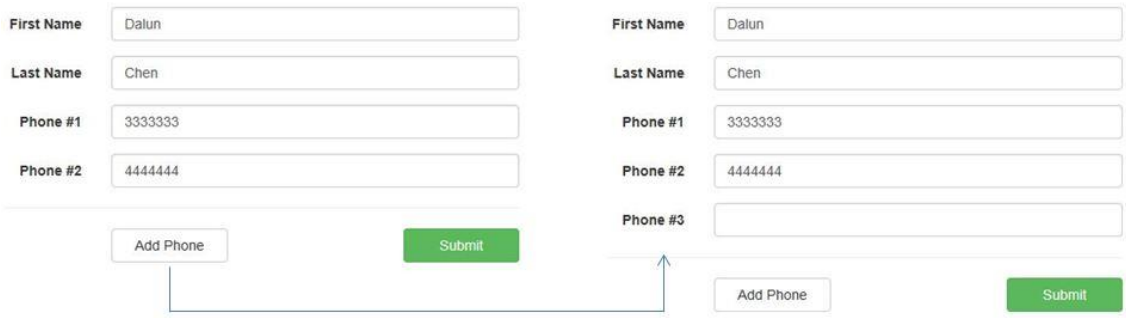


FIGURE 40. An example of a dynamic form

In order to ensure the server receives the values entered by the user, a developer should make sure that for each input field in a form, a unique name attribute is given. During the implementation of the dynamic forms in the prototype, it had concerned me that the chance of missing or repeating name attributes might increase. Hence, I felt compelled to examine whether the implemented forms send back all the input values.

Although I could inspect the postback name-value string pairs by accessing *Request.Form* in a HTTP POST *ActionResult* method, I would still need to pass a parameter to the method in order to use the same method name as the corresponding HTTP GET method. Otherwise, Visual Studio would indicate an error message and would not compile the solution (see figure 41).

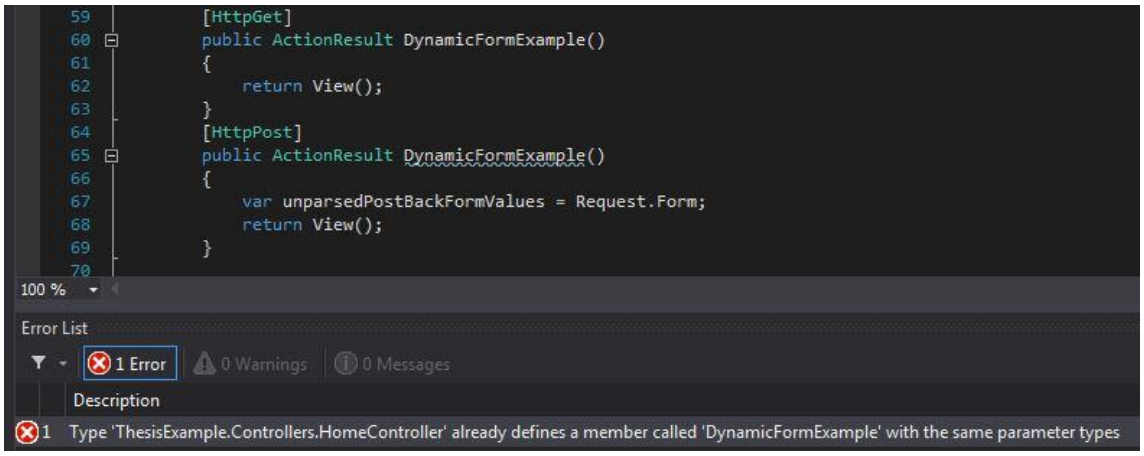


FIGURE 41. The HTTP POST ActionResult method expects a parameter

The main reason for using the same method name is to make debugging and maintenance a bit easier: other developers can quickly identify the pair of methods which handle HTTP GET and POST requests for the same view. The name conflict shown in figure 41 can be resolved by passing an object of *FormCollection* type to the HTTP POST *ActionResult* method. As a result, an object of *FormCollection* containing postback values will be created when the HTTP POST method is called. The parameter also indicates to other developers that the method expects postback form values.

One way to inspect the postback name-value pairs in a *FormCollection* object is looping through the object's property *AllKeys,* which is a string array containing all the "name" attributes of the input fields in a form (Microsoft Developer Network 2015m, cited 21.12.2015). In the loop, each name-value pair can be written to the destination page for inspection. Figure 42 illustrates the code for looping through the property *AllKeys* of the *FormCollection* object which contains the testing values I entered in the simplified form. In the loop, a system method *Response.Write()* is used to display the name-value pairs in a destination view; the comment-out code shows the same loop could be applied to *Request.Form*. As a result, the method returns a view displaying the name attributes of the input elements and the values I entered (see figure 43). The inspection helped to determine whether a dynamic form functions properly.

```
65          [HttpPost]
66          public ActionResult DynamicFormExample(FormCollection dynamicForm)
67          {
68              Response.Write("input field name: entered value <br>");
69              foreach (var key in dynamicForm.AllKeys)
70              {
71                  Response.Write(key + " : " + dynamicForm[key] + "<br>");
72              }
73
74              //foreach (var key in Request.Form.AllKeys)
75              //{
76              //    Response.Write(key + " : " + Request.Form[key] + "<br>");
77              //}
78
79              return View();
80          }
```

FIGURE 42. Accessing the name-value pairs in the FormCollection object

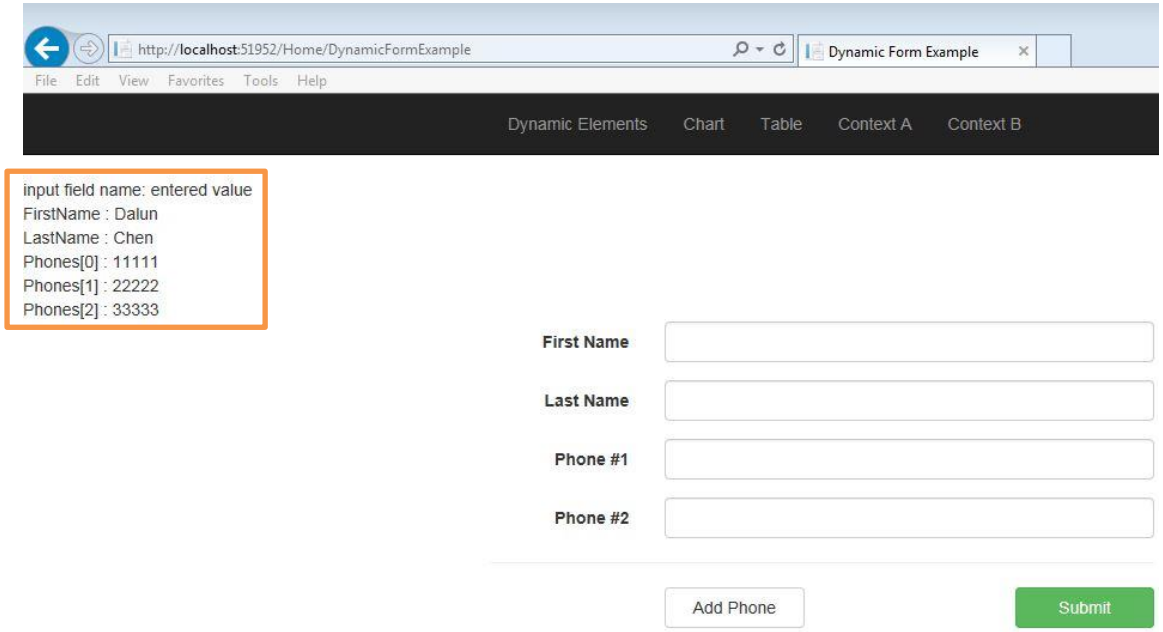


FIGURE 43. Displaying the name-value pairs in a view

4.6 Deploying the prototype to localhost

During the development phase, the prototype was regularly deployed to the localhost of my machine and tested on a smartphone, a tablet and a laptop on the company's WLAN. Testing on different devices helped me discover bugs and responsiveness issues which could not be noticed on the desktop. In this subchapter, I will show how to publish a MVC5 project to the localhost of a machine by "Web Deploy" method.

Firstly, run Visual Studio 2013 as administrator. Find “Visual Studio 2013” from the “Start” menu, then right-click on the program and select “Run as administrator”. After the program is ready, open the solution which contains the intended project for publishing. (Dykstra 2015, cited 5.1.2016).

Secondly, create a publish profile for the project. Make sure the project for publishing is selected in the “Solution Explorer”, otherwise publish option is disabled (ibid.). Right click on the project and select “Publish…” to open the “Publish Web” configuration box. For publishing a project to the localhost of the same machine, select “Custom” as the publish target (see figure 44). Then, give a name to the profile (see figure 45).

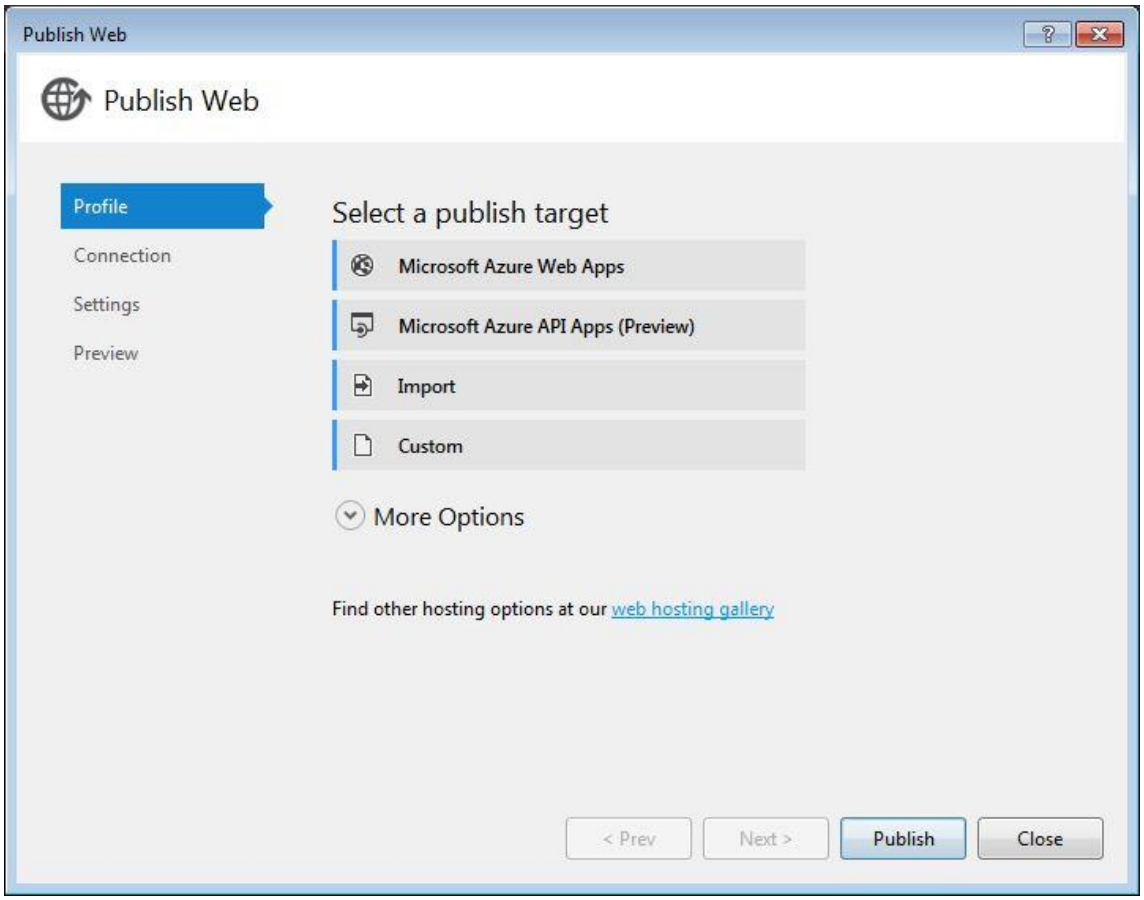


FIGURE 44. Publish Web box

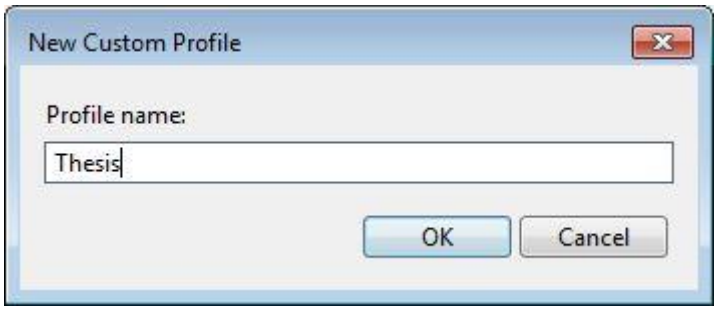


FIGURE 45. Naming the profile

Thirdly, configure the profile to publish the project automatically to IIS, known as "one-click publish" (Dykstra 2015, cited 5.1.2016). After naming the profile, the "Publish Web" box displays input fields for configuring "Connection". By default, "Web Deploy" has been selected as "Publish method". In the simplest deployment scenario where database is not included in the project, the only configurations needed are for "Server" and "Site name". Entering "localhost" in the field for Server and "Default Web Site/CustomName" will instruct Visual Studio to publish the project to a custom folder "CustomName" under the "Default Web Site" in IIS. For example, I configured the "Site name" in my publish profile to a custom folder named "MyThesis" under the "Default Web Site" (see figure 46).

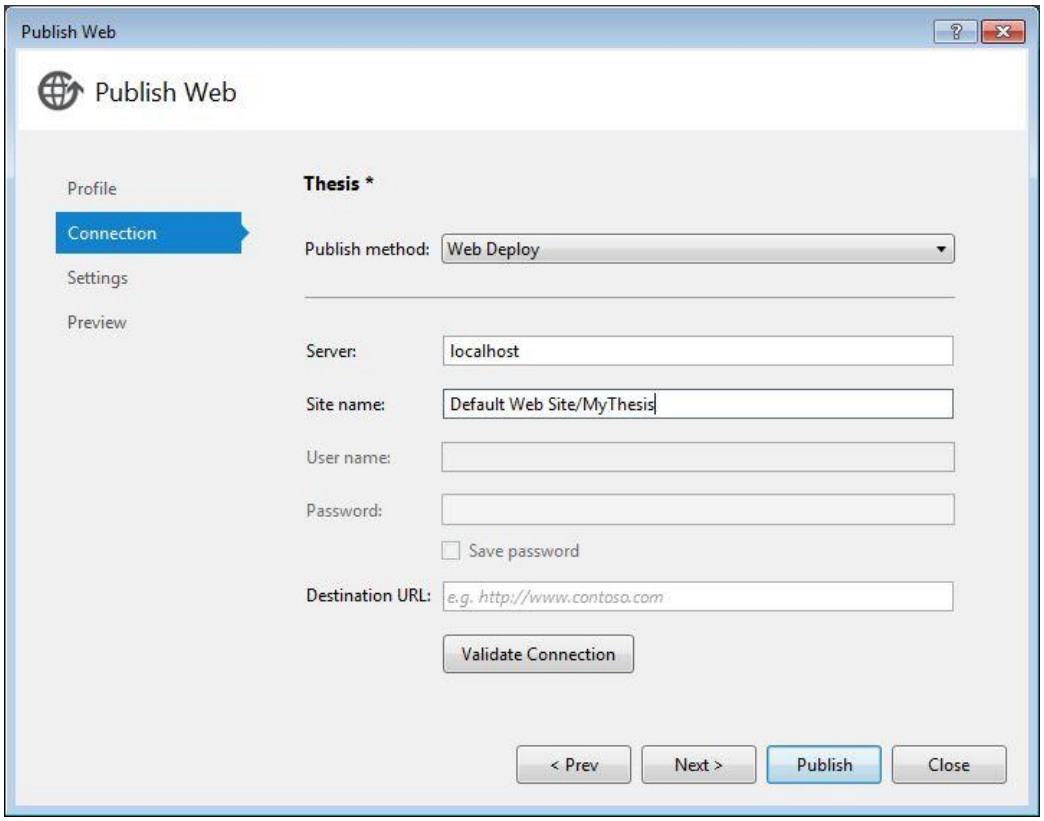


FIGURE 46. Configuring Connection

Finally, after configuring the “Connection”, the project can be deployed to IIS by clicking the “Publish” button. If a browser has not been opened automatically to display the index page of the published application, one can access the application by typing “localhost”, a forward slash, then the name of the custom folder in the address bar of the browser. For other devices on the WLAN to access the published application, simply replace “localhost” with the private IP address of the machine. For example, if I had a WLAN at home, I could access my published application on the smartphone by entering the private IP address and the subfolder name (see figure 47). Once a publish profile is created and configured, clicking on the “Publish” button is the only action needed when publishing the project next time (see figure 48).

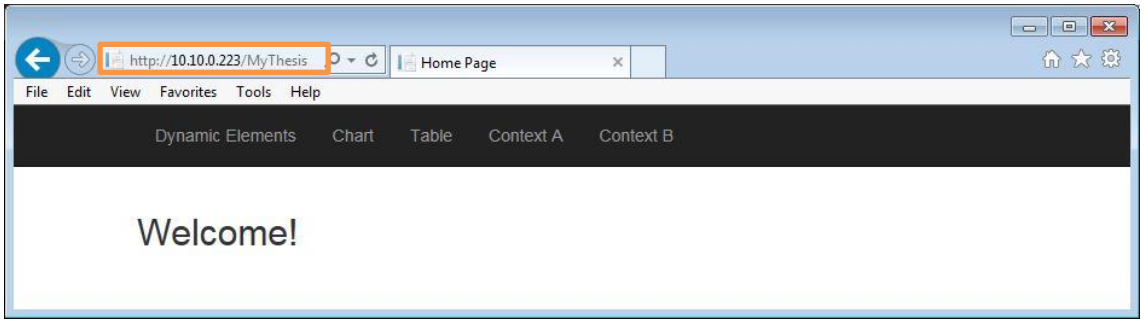


FIGURE 47. The result of a successful deployment

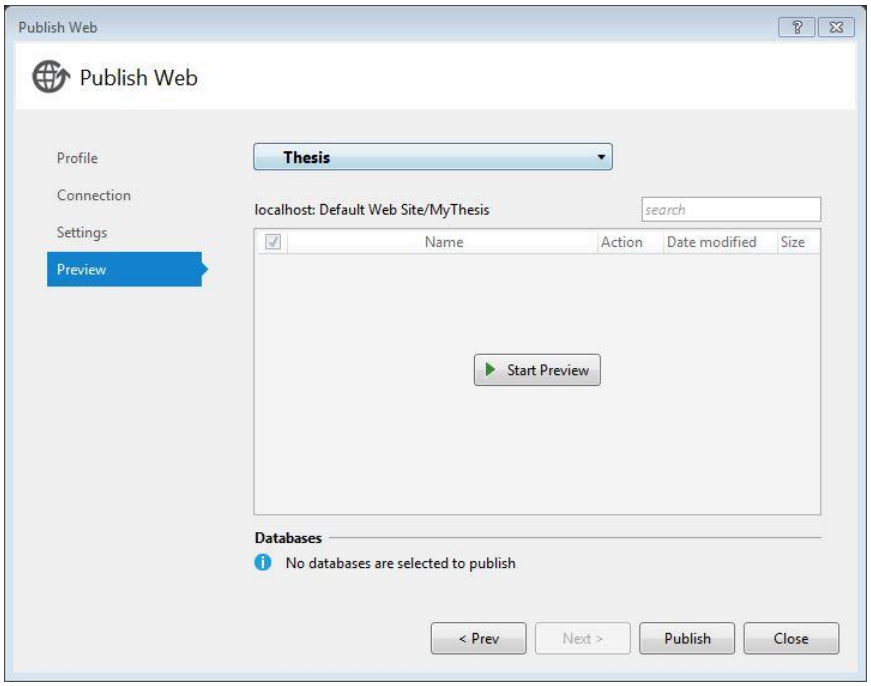


FIGURE 48. "One-click publish”

The advantage of publishing a project to a subfolder is that it can be easily removed later without affecting other files in the “Default Web Site” folder in IIS. However, if a login mechanism is implemented in a project, publishing the project to a subfolder instead of to the root level might result “page not found” error, because of the additional URL parameter, i.e. the name of the subfolder. A solution for that is creating a dedicated website folder under the “Sites” folder in IIS and publishing the project directly to the dedicated folder. Make sure the dedicated folder uses a different port than the “Default Web Site”. Since database and real login mechanism were not implemented in the prototype, the configurations described in this subchapter were sufficient.

5 CONCLUSIONS

The literature review helped me understand the requirements clearly and drew my attention to the way data is presented in the prototype. In addition, by applying the knowledge gained from the literature review, meaningful demo data, e.g. representing blood pressure and blood glucose, were created and presented by responsive charts and sortable tables.

Through regular meetings and personal discussions with the commissioner and other members of the company, the requirements for the prototype were collected and clarified. The requirements could be summarized as the following statement: delivering a prototype of a web-based PHR application; which is responsive, interactive, uses charts and tables, and is able to leverage the company's existing platform.

ASP.NET MVC5, Bootstrap, jQuery, Chart.JS and Tablesorter open source plugin were chosen for implementing the requirements. Creating the prototype using ASP.NET MVC5 framework enabled the prototype to enjoy the benefits of MVC such as separation of concerns and to easily utilize the company's components which are built on ASP.NET technology. To create responsive and interactive pages, Bootstrap and jQuery were used, since they are by default included in a MVC5 solution. I had studied Chart.JS and Tablesorter plugin before used them in the prototype. Chart.JS version 1.0.2 can draw appealing responsive charts even when there is missing data; whereas Tablesorter can easily transform a static table to a sortable one; furthermore, it offers multicolumn sorting and custom attribute sorting.

The prototype utilized the concept of a configurable dashboard which I developed in the beginning of the development phase. The configurable dashboard displays personal health measurements by charts and tables. In addition, there is a menu containing checkboxes, which enables the user to re-configure the content of the dashboard. Although the dashboard was tailored to the company's other PHR project and was not incorporated in the prototype due to time constraints, the concept of the dashboard was adopted in the development of the prototype. A page displaying a summary of personal health measurements was created in the prototype.

A step-by-step form conceptualized by the commissioner was implemented in the prototype. The form was designed to help the user complete a complicated task by breaking it into sequential

steps. As a result, the user can better concentrate on the subtasks in each step. To create reusable elements for the form, ASP.NET MVC5 partial view feature was used. Furthermore, as the form allows new input fields to be inserted, special attention was paid to ensure that the data entered in the new input fields get send back to the controller. To inspect postback values and to allow the same method name be given to the HTTP GET and HTTP POST *ActionResult* methods which serve the same page, ASP.NET MVC *FormCollection* class was used.

Responsive line charts and sortable tables were created for effective data presentation. The line charts are not only responsive, but also configurable. They were implemented to visualize the progress of personal health measurements. The user can either select a day range from a dropdown list or pick a day range from a calendar; as a result, the charts will redraw accordingly. If there is missing data between two dots, the charts will still connect the dots with a smooth line. If a selected day range is longer than 14 days, dots are removed, tooltip feature is disabled and date labels are reduced or hidden. Nevertheless, the day range is indicated on top of the chart. The tables, on the other hand, enable the user to sort multiple columns. Message List is one of the pages in which a sortable table was implemented. In the page, a table displays messages from the newest to the oldest by default. In addition, the user can sort the messages differently by choosing different columns and in either ascending or descending order.

During the development, frequent web deployment had facilitated testing on mobile devices. Testing on the tablet and the smartphone helped me discover bugs, responsiveness and other issues which cannot be detected in a simulator.

Finally, the prototype was translated into Finnish in order to be presented to the company's domestic stakeholders. Translation took place after all required pages had been created. The commissioner was asked to proofread the translation; corrections were made accordingly. As the prototype is available in Finnish and English, the commissioner can present it to both domestic and international stakeholders and obtain essential information for the next iteration.

6 DISCUSSION

The objective to create a prototype of a web-based PHR application tailored to the commissioner's requirements was achieved. The delivered prototype is responsive, contains interactive pages, presents meaningful demo data in configurable charts and sortable tables, and is able to leverage the company's platform. The prototype serves as a medium for the commissioner to collect feedback and to obtain new requirements from the company's domestic and international stakeholders.

Proactive and regular communication with the commissioner and the instructing teacher facilitated the implementation. The feedback from the commissioner and other members in the company steered the development, while the advice from the instructing teacher showed me where to find answers for technical issues. Literature review and self-study, on the other hand, supported the development. The knowledge gained from the literature review made me appreciate the role of a personal health record application in chronic disease self-management, helped me understand the requirements better, and enabled me to create meaningful demo data. Self-study was carried out to equip myself with the required JavaScript and jQuery programming skills for the implementation. Finally, 13 weeks were allocated to the development phase for ensuring that the objective was achieved.

For future iterations, I suggest recruiting target users, healthcare professionals and experts in graphical design into the project team. As revealed in other studies, a wide range of knowledge and skills is required in eHealth projects for achieving desired results.

Finally, for students who are interested in eHealth, besides participating in an eHealth project, a usability study on a collection of popular personal health applications can also be a feasible topic for the scope of a bachelor thesis. Such a study can contribute to the advance of personal health application by providing insights into the building blocks of the well-received applications.

REFERENCES

Alahäivälä, T. 2013. Software design of a health BCSS: Case Onnikka. University of Oulu. Department of Information Processing Science. Master's thesis. Reviewed 3.6.2015, http://herkules.oulu.fi/thesis/nbnfioulu-201306061567.pdf

Allen, K. S. 2009. 6 tips for ASP.NET MVC model binding. Cited 15.12.2015, http://odetocode.com/blogs/scott/archive/2009/04/27/6-tips-for-asp-net-mvc-model-binding.aspx

Apple 2016a. Health. Cited 22.1.2016, http://www.apple.com/ios/health/

Apple 2016b. Health & fitness. Cited 22.1.2016, http://www.apple.com/shop/iphone/iphone-accessories/health-fitness

Apple 2016c. iHealth wireless blood glucometer with 50 test strips. Cited 22.1.2016, http://www.apple.com/shop/product/HJ152ZM/A/ihealth-wireless-blood-glucometer-with-50-test-strips?fnode=4a

Appuhamy, J. A. D. R. N., Kebreab, E., Simon, M., Yada, R., Milligan, L. P. & France, J. 2014. Effects of diet and exercise interventions on diabetes risk factors in adults without diabetes: meta-analyses of controlled trails. Diabetology & Metabolic Syndrome 6:127. Cited 15.1.2016, http://www.dmsjournal.com/content/pdf/1758-5996-6-127.pdf

Bach, C. 2015a. Tablesorter documentation. Cited 27.11.2015, http://tablesorter.com/docs/

Bach, C. 2015b. Christian Bach. Cited 27.11.2015, https://www.linkedin.com/in/cbach

Bach, C. 2015c. christianbach / tablesorter. Cited 27.11.2015, https://github.com/christianbach/tablesorter

Bach, C. 2015d. tablesorter / docs / example-attribute-sort.html. Cited 27.11.2015, https://github.com/christianbach/tablesorter/blob/master/docs/example-attribute-sort.html

British Lung Foundation 2014. World COPD Day: key facts. Cited 4.8.2015, https://www.blf.org.uk/Page/World-COPD-Day-key-facts

Chart.js 2015a. Chart.js documentation. Cited 25.11.2015, http://www.chartjs.org/docs/

Chart.js 2015b. Release. Cited 25.11.2015, https://github.com/nnnick/Chart.js/releases

CHARTIST.JS 2015. Chartist – examples. Cited 25.11.2015, https://gionkunz.github.io/chartist-js/examples.html

Department of Health 2016. New alcohol guidelines show increased risk of cancer. Cited 18.1.2016, https://www.gov.uk/government/news/new-alcohol-guidelines-show-increased-risk-of-cancer

Diabetes UK 2015a. Facts and stats. London: Diabetes UK. Cited 11.1.2016, https://www.diabetes.org.uk/Documents/Position%20statements/Diabetes%20UK%20Facts%20and%20Stats_Dec%202015.pdf

Diabetes UK 2015b. State of the nation 2015: the age of diabetes. London: Diabetes UK. Cited 13.1.2016, https://www.diabetes.org.uk/upload/Scotland/SOTN%20Diabetes.pdf

Diabetes UK 2016. My story is a warning to others. Cited 4.2.2016, cited https://www.diabetes.org.uk/Your-stories/Type-2/My-story-is-a-warning-to-others/

Diabetes.co.uk 2016. Guild to HbA1c. Cited 7.1.2016, http://www.diabetes.co.uk/what-is-hba1c.html

Dykstra, T. 2015. ASP.NET web deployment using Visual Studio: deploying to test. Cited 5.1.2016, http://www.asp.net/mvc/overview/deployment/visual-studio-web-deployment/deploying-to-iis

East Sussex Healthcare NHS Trust 2016. Who might need to see a dietitian. Cited 15.1.2016, http://www.esht.nhs.uk/nutrition-dietetics/who/

Galloway, J., Wilson, B., Allen, K. S. & Matson, D. 2014. Professional ASP.NET MVC 5. Birmingham, UK: Wrox Press. Internal source. Cited 2.12.2015, http://proquest.safaribooksonline.com.ezp.oamk.fi:2048/book/programming/microsoft-aspdotnet/9781118794760/firstchapter#X2ludGVybmFsX0h0bWxWaWV3P3htbGlkPTk3ODExMTg3OTQ3NjAlMkZjMDVfbGV2ZWwxXzRfaHRtbCZxdWVyeT1yZW5kZXJpbmclMjBoZWxwZXI=

Gebel, E. 2009. The science of carbs: sugars, starches, and fiber, from molecule to meal. Cited 6.1.2016, http://www.diabetesforecast.org/2009/sep/the-science-of-carbs.html?loc=howthebody

Gebel, E. 2011. How the body uses carbohydrates, proteins, and fats. Cited 6.1.2016, http://www.diabetesforecast.org/2011/mar/how-the-body-uses-carbohydrates-proteins-and-fats.html?referrer=https://www.google.fi/

Gnazzo, J. 2015. ASP.NET MVC 5 – Passing data from a View to a Controller. Cited 15.12.2015, http://www.teamscs.com/2015/09/asp-net-mvc-5-passing-data-from-a-view-to-a-controller/

Google 2015. Device mode & mobile emulation. Cited 22.12.2015, https://developer.chrome.com/devtools/docs/device-mode

Google 2016a. Google Play search result of “diabetes”. Cited 22.1.2016, https://play.google.com/store/search?q=diabetes&c=apps

Google 2016b. Google Play > Sovellukset > Terveys ja kuntoilu. Cited 22.1.2016, https://play.google.com/store/apps/category/HEALTH_AND_FITNESS/collection/topselling_free

GroupHealth 2014a. How our bodies turn food into energy. Cited 6.1.2016, http://www.ghc.org/healthAndWellness/?item=/common/healthAndWellness/conditions/diabetes/foodProcess.html

GroupHealth 2014b. How insulin works. Cited 6.1.2016, http://www.ghc.org/popup.jhtml?item=/common/healthAndWellness/conditions/diabetes/insulinProcess.html

Hammarberg, M. 2009. UpdateModel, FormCollection and unit test. Cited 18.12.2015, http://www.marcusoft.net/2009/03/updatemodel-formcollection-and-unit_5466.html

Harford, R. 2013. How to use the MVC FormCollection object. Video. Cited 18.12.2015, https://www.youtube.com/watch?v=e6bN-HHaKeU

Health and Safety Executive 2014. Chronic obstructive pulmonary disease (COPD) in Great Britain in 2014. Cited 4.8.2015, http://www.hse.gov.uk/Statistics/causdis/copd/copd.pdf

Health and Social Care Information Centre 2014. National diabetes audit 2012-2013 report 1: Care processes and treatment targets. Leeds: Health and Social Care Information Centre. Cited 12.1.2016, http://www.hscic.gov.uk/catalogue/PUB14970/nati-diab-audi-12-13-care-proc-rep.pdf

International Diabetes Federation 2015. Annual report 2014. Brussels: International Diabetes Federation. Cited 13.1.2016, http://www.idf.org/sites/default/files/IDF-2014-Annual-Report-final.pdf

Johansson, P. 2015. ASP.NET MVC: Proper model binding with dynamic form. Cited 14.12.2015, https://patricjsson.wordpress.com/2015/05/25/asp-net-mvc-proper-model-binding-with-dynamic-form/

Johns Hopkins Medicine 2015a. Overview of the vascular system. Cited 18.7.2015, http://www.hopkinsmedicine.org/healthlibrary/conditions/cardiovascular_diseases/overview_of_the_vascular_system_85,P08254/

Johns Hopkins Medicine 2015b. Peripheral vascular disease. Cited 20.7.2015, http://www.hopkinsmedicine.org/healthlibrary/conditions/cardiovascular_diseases/peripheral_vascular_disease_85,P00236/

Johns Hopkins Medicine 2015c. Chronic bronchitis. Cited 31.7.2015, http://www.hopkinsmedicine.org/healthlibrary/conditions/respiratory_disorders/chronic_bronchitis_85,P01303/

Johns Hopkins Medicine 2015d. Pulmonary emphysema. Cited 31.7.2015, http://www.hopkinsmedicine.org/healthlibrary/conditions/respiratory_disorders/pulmonary_emphysema_85,P01309/

Jones, M. P. 2015. What are partial views? – ASP.NET MVC Demystified. Cited 2.12.2015, http://www.exceptionnotfound.net/partial-views-asp-net-mvc-demystified/

Kennedy, M. 2012. Understanding text encoding in ASP.NET MVC (ASP.NET MVC foundations series). Cited 26.11.2015, http://blog.michaelckennedy.net/2012/10/15/understanding-text-encoding-in-asp-net-mvc/

Korhonen, P. E., Seppälä, T., Järvenpää, S. & Kautiainen, H. 2014. Body mass index and health-related quality of life in apparently healthy individuals. Quality of Life Research 23 (1), 67-74. Internal source. Cited 15.1.2016, http://search.proquest.com.ezp.oamk.fi:2048/docview/1491226003?accountid=13030

Koskela, K. 2005. Krooninen keuhkoputkitulehdus ja keuhkoahtaumatauti. Cited 3.8.2015, http://www.terveyskirjasto.fi/terveyskirjasto/tk.koti?p_artikkeli=suo00032#refs

Koski, S. 2015. Diabetesbarometri 2015. Tampere: Suomen Diabetesliitto ry. Cited 8.1.2016, http://www.diabetes.fi/files/6203/barometri2015.pdf

Kousoulis, A. A., Patelarou, E., Shea, S., Foss, C., Knutsen, I. A.R., Todorova, E., Roukova, P., Portillo, M. C., Pumar-Méndez, M. J., Mujika, A., Rogers, A., Vassilev, I., Serrano-Gil, M. & Lionis, C. 2014. Diabetes self-management arrangements in Europe: a realist review to facilitate a project implemented in six countries. BMC Health Services Research 14:453. Cited 1.6.2015, http://search.proquest.com.ezp.oamk.fi:2048/docview/1611278590?accountid=13030

Looper, J. 2015. Through the looking glass: adventures in mobile app simulation, emulation, and device testing. Cited 22.12.2015, http://developer.telerik.com/featured/looking-glass-adventures-mobile-app-simulation-emulation-device-testing/

MDN 2015. Introduction to the DOM. Cited 18.11.2015, https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction

Microsoft 2015. Microsoft Health for android phones. Cited 25.1.2016, https://play.google.com/store/apps/details?id=com.microsoft.kapp

Microsoft 2016a. Microsoft Health. Cited 22.1.2016, https://www.microsoft.com/microsoft-health/en-us

Microsoft 2016b. Health Vault. Cited 22.1.2016, https://www.healthvault.com/fi/fi/overview

Microsoft ASP.NET 2015, Simulate popular mobile devices for testing. Cited 22.12.2015, http://www.asp.net/mobile/device-simulators

Microsoft ASP.NET Forums 2008. Form collection. Cited 18.12.2015, http://forums.asp.net/t/1313286.aspx?FormCollection

Microsoft Developer Network 2015a. Json.Encode method. Cited 26.11.2015, https://msdn.microsoft.com/en-us/library/system.web.helpers.json.encode(v=vs.111).aspx

Microsoft Developer Network 2015b. HtmlHelper.Raw method. Cited 26.11.2015, https://msdn.microsoft.com/en-us/library/system.web.mvc.htmlhelper.raw(v=vs.118).aspx#M:System.Web.Mvc.HtmlHelper.Raw%28System.Object%29

Microsoft Developer Network 2015c. MvcHtmlString Class. Cited 2.12.2015, https://msdn.microsoft.com/en-us/library/system.web.mvc.mvchtmlstring(v=vs.118).aspx

Microsoft Developer Network 2015d. ASP.NET web page resources overview. Cited 9.12.2015, https://msdn.microsoft.com/en-us/library/ms227427(v=vs.100).aspx

Microsoft Developer Network 2015e. How to: Create resource files for ASP.NET web sites. Cited 9.12.2015, https://msdn.microsoft.com/en-us/library/ms247246(v=vs.100).aspx

Microsoft Developer Network 2015f. Using IntelliSense. Cited 11.12.2015, https://msdn.microsoft.com/en-us/library/hcw1s69b.aspx

Microsoft Developer Network 2015g. Using type dynamic (C# programming guide). Cited 11.12.2015, https://msdn.microsoft.com/en-us/library/dd264736.aspx

Microsoft Developer Network 2015h. Request object. Cited 15.12.2015, https://msdn.microsoft.com/en-us/library/ms524948(v=vs.90).aspx

Microsoft Developer Network 2015i. Request.Form collection. Cited 15.12.2015, https://msdn.microsoft.com/en-us/library/ms525985(v=vs.90).aspx

Microsoft Developer Network 2015j. Controller.Request property. Cited 15.12.2015, https://msdn.microsoft.com/en-US/library/system.web.mvc.controller.request(v=vs.118).aspx

Microsoft Developer Network 2015k. FormCollection class. Cited 17.12.2015, https://msdn.microsoft.com/en-us/library/system.web.mvc.formcollection(v=vs.118).aspx

Microsoft Developer Network 2015l. NameValueCollection class. Cited 17.12.2015, https://msdn.microsoft.com/en-us/library/system.collections.specialized.namevaluecollection(v=vs.100).aspx

Microsoft Developer Network 2015m. NameValueCollection.AllKeys property. Cited 21.12.2015, https://msdn.microsoft.com/en-US/library/system.collections.specialized.namevaluecollection.allkeys(v=vs.118).aspx

Microsoft Developer Network 2015n. Emulate browsers, screen sizes, and GPS locations. Cited 22.12.2015, https://msdn.microsoft.com/en-us/library/dn255001(v=vs.85).aspx

Microsoft Developer Network 2016. ASP.NET IIS Registration Tool (Aspnet_regiis.exe). Cited 5.1.2016, https://msdn.microsoft.com/library/k6h9cz8h(v=vs.100).aspx

Microsoft Visual Studio 2012. Valid javascript/Razor syntax marked as syntax error. Cited 26.11.2015, https://connect.microsoft.com/VisualStudio/feedback/details/760339/valid-javascript-razor-syntax-marked-as-syntax-error

Mobile Joomla! 2015. How to test mobile websites on desktop: best emulators and tools. Cited 22.12.2015, http://www.mobilejoomla.com/blog/222-how-to-test-mobile-websites-on-desktop-best-emulators-and-tools.html

Mooney, G. 2013. Emulated vs. real device mobile app testing. Cited 22.12.2015, http://blog.smartbear.com/mobile/emulated-vs-real-device-mobile-app-testing/

Mustajoki, P. 2014. Keuhkoahtaumatauti (COPD). Cited 3.8.2015, http://www.terveyskirjasto.fi/terveyskirjasto/tk.koti?p_artikkeli=dlk00029#s1

Mustajoki, P. 2015. Diabetes (sokeritauti). Cited 6.1.2016, http://www.terveyskirjasto.fi/terveyskirjasto/tk.koti?p_artikkeli=dlk00011

mySugr GmbH 2016. mySugr diabetes logbook. Cited 25.1.2016, https://play.google.com/store/apps/details?id=com.mysugr.android.companion&hl=en

NHS choices 2014a. NHS Health Check / Testing for the biggest killers. Cited 18.7.2015, http://www.nhs.uk/conditions/nhs-health-check/pages/why-these-conditions.aspx

NHS choices 2014b. Atherosclerosis. Cited 18.7.2015, http://www.nhs.uk/conditions/Atherosclerosis/Pages/Introduction.aspx

NHS choices 2014c. Cardiovascular disease. Cited 20.7.2015, http://www.nhs.uk/conditions/Cardiovascular-disease/Pages/Introduction.aspx

NHS choices 2014d. Abdominal aortic aneurysm. Cited 20.7.2015, http://www.nhs.uk/conditions/repairofabdominalaneurysm/Pages/Introduction.aspx

NHS choices 2014e. Abdominal aortic aneurysm – causes. Cited 20.7.2015, http://www.nhs.uk/Conditions/repairofabdominalaneurysm/Pages/Causes.aspx

NHS choices 2014f. Cardiovascular disease - risk factors. Cited 29.5.2015, http://www.nhs.uk/Conditions/cardiovascular-disease/Pages/Risk-factors.aspx

NHS choices 2014g. High blood pressure (hypertension) - complications. Cited 29.7.2015, http://www.nhs.uk/Conditions/Blood-pressure-(high)/Pages/Complications.aspx

NHS choices 2014h. Heart failure. Cited 27.7.2015, http://www.nhs.uk/Conditions/Heart-failure/Pages/Introduction.aspx

NHS choices 2014i. Chronic obstructive pulmonary disease. Cited 31.7.2015, http://www.nhs.uk/Conditions/Chronic-obstructive-pulmonary-disease/Pages/Introduction.aspx

NHS choices 2014j. Bronchitis. Cited 31.7.2015, http://www.nhs.uk/Conditions/Bronchitis/Pages/Introduction.aspx

NHS choices 2014k. Bronchitis - Symptoms. Cited 31.7.2015, http://www.nhs.uk/Conditions/Bronchitis/Pages/Symptoms.aspx

NHS choices 2014l. Chronic obstructive pulmonary disease – causes. Cited 4.8.2015, http://www.nhs.uk/Conditions/Chronic-obstructive-pulmonary-disease/Pages/Causes.aspx

NHS choices 2014m. Diabetes. Cited 7.1.2016, http://www.nhs.uk/conditions/Diabetes/Pages/Diabetes.aspx

NHS choices 2014n. Type 2 diabetes – cause. Cited 7.1.2016, http://www.nhs.uk/Conditions/Diabetes-type2/Pages/Causes.aspx

NHS choices 2014o. Type 2 diabetes – symptoms. Cited 7.1.2016, http://www.nhs.uk/Conditions/Diabetes-type2/Pages/Symptoms.aspx

NHS choices 2014p. Healthy living with diabetes. Cited 7.1.2016, http://www.nhs.uk/Livewell/Diabetes/Pages/Healthfordiabetics.aspx

NHS choices 2014q. What should my daily intake of calories be?. Cited 14.1.2016, http://www.nhs.uk/chq/pages/1126.aspx?categoryid=51

NHS choices 2014r. Eating a balanced diet. Cited 14.1.2016, http://www.nhs.uk/Livewell/Goodfood/Pages/Healthyeating.aspx

NHS choices 2014s. The risks of drinking too much. Cited 18.1.2016, http://www.nhs.uk/Livewell/alcohol/Pages/Effectsofalcohol.aspx

NHS choices 2015a. Embolism. Cited 18.7.2015, http://www.nhs.uk/Conditions/Embolism/Pages/Introduction.aspx

NHS choices 2015b. High cholesterol. Cited 29.7.2015, http://www.nhs.uk/conditions/Cholesterol/Pages/Introduction.aspx

NHS choices 2015c. Hyperglycaemia - complications. Cited 29.7.2015, http://www.nhs.uk/Conditions/Hyperglycaemia/Pages/Complications.aspx

NHS choices 2015d. Atrial fibrillation. Cited 27.7.2015, http://www.nhs.uk/Conditions/Atrial-fibrillation/Pages/Introduction.aspx

NHS choices 2015e. Respiratory tract infections. Cited 31.7.2015, http://www.nhs.uk/conditions/Respiratory-tract-infection/Pages/Introduction.aspx

NHS choices 2015f. The eatwell plate. Cited 14.1.2016, http://www.nhs.uk/Livewell/Goodfood/Pages/eatwell-plate.aspx

NHS choices 2015g. Starchy foods and carbohydrates. Cited 14.1.2016, http://www.nhs.uk/Livewell/Goodfood/Pages/starchy-foods.aspx

NHS choices 2015h. Why 5 A DAY?. Cited 14.1.2016, http://www.nhs.uk/Livewell/5ADAY/Pages/Why5ADAY.aspx

NHS choices 2015i. Fat: the facts. Cited 14.1.2016, http://www.nhs.uk/Livewell/Goodfood/Pages/Fat.aspx

NHS choices 2015j. Meat in your diet. Cited 15.1.2016, http://www.nhs.uk/Livewell/Goodfood/Pages/meat.aspx

NHS choices 2015k. Fish and shellfish. Cited 15.1.2016, http://www.nhs.uk/Livewell/Goodfood/Pages/fish-shellfish.aspx

NHS choices 2015l. Pulses in your diet. Cited 15.1.2016, http://www.nhs.uk/Livewell/Goodfood/Pages/pulses.aspx

NHS choices 2015m. Milk and dairy in your diet. Cited 15.1.2016, http://www.nhs.uk/Livewell/Goodfood/Pages/milk-dairy-foods.aspx

NHS choices 2015n. Eight tips for healthy eating. Cited 15.1.2016, http://www.nhs.uk/Livewell/Goodfood/Pages/eight-tips-healthy-eating.aspx

NHS choices 2015o. How does sugar in our diet affect our health. Cited 15.1.2016, http://www.nhs.uk/Livewell/Goodfood/Pages/sugars.aspx

NHS choices 2015p. Salt: the facts. Cited 15.1.2016, http://www.nhs.uk/Livewell/Goodfood/Pages/salt.aspx

NHS England & Public Health England 2015. National NHS diabetes initiative launched in major bid to prevent illness. Cited 13.1.2016, https://www.gov.uk/government/news/national-nhs-diabetes-initiative-launched-in-major-bid-to-prevent-illness

Nundy, S., Dick, J. J., Chou, C., Nocon, R. S., Chin, M. H. & Peek, M. E. 2014. Mobile phone diabetes project led to improved glycemic control and net savings for Chicago Plan participants. Health affairs 33 (2), 265-272. Cited 1.6.2015, http://search.proquest.com.ezp.oamk.fi:2048/docview/1498231630?accountid=13030

Penberthy, W. 2013. Exam Ref 70-486: Developing ASP.NET MVC 4 web applications. Redmond, WA: Microsoft Press. Internal source. Cited 9.12.2015, http://proquest.safaribooksonline.com.ezp.oamk.fi:2048/book/certification/9780735677418/3dot-develop-the-user-experience/ch03s02_html?query=((asp.net+mvc)+AND+(globalization))#snippet

ProWellness Health Solutions 2013. Yhtiö. Cited 4.2.2016, http://www.prowellness.com/fi/?s=4&id=8

ProWellness Health Solutions 2014. Central London for Community Healthcare (CLCH) / Imperial College NHS Trust wins BTS Award. Cited 4.2.2016, http://www.prowellness.com/?s=news&id=32

Ross, J., Stevenson, F., Dack, C., Pal, K., May, C., Michie, S., Parrott, S. & Murray, E. 2014. Evaluating the implementation of HeLP-Diabetes within NHS services: study protocol. BMC Health Services Research 14:51. Internal source. Cited 1.6.2015, http://search.proquest.com.ezp.oamk.fi:2048/docview/1496010731?accountid=13030

StackOverflow 2012a. Razor/javascript and trailing semicolon. Cited 26.11.2015, http://stackoverflow.com/questions/12111729/razor-javascript-and-trailing-semicolon

StackOverflow 2012b. Upgrade from .net 4 to 4.5 breaks Html.Raw call in Javascript. Cited 26.11.2015, http://stackoverflow.com/questions/12275095/upgrade-from-net-4-to-4-5-breaks-html-raw-call-in-javascript

StackOverflow 2013. Razor syntax error serializing ASP.NET Model to JSON with Html.Raw. Cited 26.11.2015, http://stackoverflow.com/questions/17617263/razor-syntax-error-serializing-asp-net-model-to-json-with-html-raw

StackOverflow 2014. How to get a string from .resx file to a .js file. Cited 8.12.2015, http://stackoverflow.com/questions/26522133/how-to-get-a-string-from-resx-file-to-a-js-file

StackOverflow 2015. Show label in tooltip but not in x axis for chartjs line chart. Cited 25.11.2015, http://stackoverflow.com/questions/31604040/show-label-in-tooltip-but-not-in-x-axis-for-chartjs-line-chart

Statistics Finland 2013. Appendix table 1a. Deaths by underlying cause of death and by age in 2012, both sexes. Cited 22.7.2015, http://www.stat.fi/til/ksyyt/2012/ksyyt_2012_2013-12-30_tau_001_en.html

Statistics Finland 2014a. Appendix table 1a. Deaths by underlying cause of death and by age in 2013, both sexes. Cited 23.7.2015, http://www.stat.fi/til/ksyyt/2013/ksyyt_2013_2014-12-30_tau_001_en.html

Statistics Finland 2014b. Tilastokeskuksen PX-Web-tietokannat. >> PX-Web Statfin >> Terveys >> Kuolemansyyt >> Kuolleet ja ikävakioitu kuolleisuus peruskuolemansyyn ja sukupuolen mukaan, kaikki ja 15-64-vuotiaat 1971-2013. Cited 24.7.2015, http://pxnet2.stat.fi/PXWeb/sq/90156a70-54e7-41af-85bc-9e373118b87e

Tarnanen, K., Kesäniemi, A., Kettunen, J., Kujala, U., Kukkonen-Harjula, K. & Tikkanen, H. 2010. Liikunta on lääke (Liikunta suositus). Cited 15.1.2016, http://www.terveyskirjasto.fi/terveyskirjasto/tk.koti?p_artikkeli=khp00077

Techopedia 2015. Internet Information Services (IIS). Cited 5.1.2016, https://www.techopedia.com/definition/24953/internet-information-services-iis

Terveyden ja Hyvinvoinnin Laitos 2014. Sydän- ja verisuonitautien yleisyys. Cited 19.5.2015, https://www.thl.fi/fi/web/kansantaudit/sydan-ja-verisuonitaudit/sydan-ja-verisuonitautien-yleisyys

Terveyden ja Hyvinvoinnin Laitos 2015. Diabeteksen yleisyys. Cited 8.1.2016, https://www.thl.fi/fi/web/kansantaudit/diabetes/diabeteksen-yleisyys

The jQuery Foundation 2015a. jQuery Foundation mission and vision statement. Cited 17.11.2015, https://jquery.org/mission/

The jQuery Foundation 2015b. jQuery API. Cited 17.11.2015, http://api.jquery.com/

The jQuery Foundation 2015c. Category: Selectors. Cited 18.11.2015, https://api.jquery.com/category/selectors/

The Nemours Foundation 2012. KidsHealth > Teens > Body > Body Basic Library > Lungs and Respiratory System. Cited 31.7.2015, http://kidshealth.org/teen/your_body/body_basics/lungs.html#

The Scottish Government 2014. Diabetes improvement plan. Cited 13.1.2016, http://www.gov.scot/Publications/2014/11/6742

Titaania suonessa 18.7.2013. Vuosi infarktista. Cited 19.5.2015, http://titaaniasuonessa.blogspot.fi/2013/07/vuosi-infarktista.html

Townsend, N., Williams, J., Bhatnagar, P., Wickramasinghe, K. & Rayner, M. 2014. Cardiovascular disease statistics 2014. London: British Hearth Foundation. Cited 23.7.2015, https://www.bhf.org.uk/~/media/files/publications/research/bhf_cvd-statistics-2014_web_2.pdf

University College London 2015. HeLP Diabetes – Diabetes self management programme. Cited 20.1.2016, https://www.ucl.ac.uk/pcph/research-groups-themes/ehealth/projects/projects/dmsmp

Vatsa, A.K. 2013. Instant Razor view engine how-to. Birmingham, UK: Packt Publishing Ltd. Internal source. Cited 2.12.2015, http://proquest.safaribooksonline.com.ezp.oamk.fi:2048/book/programming/9781849696302/1dot-instant-razor-view-engine-how-to/ch01s07_html#X2ludGVybmFsX0h0bWxWaWV3P3htbGlkPTk3ODE4NDk2OTYzMDIlMkZjaDAxczA3X2h0bWwmcXVlcnk9cGFydGlhbCUyMHZpZXc=

Verbanov, R. 2015. Diabetes: M. Cited 25.1.2016, https://play.google.com/store/apps/details?id=com.mydiabetes

W3Schools 2015a. JavaScript HTML DOM. Cited 18.11.2015, http://www.w3schools.com/js/js_htmldom.asp

W3Schools 2015b. JavaScript HTML DOM Document. Cited 18.11.2015, http://www.w3schools.com/js/js_htmldom_document.asp

W3Schools 2015c. JavaScript HTML DOM Elements. Cited 18.11.2015, http://www.w3schools.com/js/js_htmldom_elements.asp

W3Schools 2015d. jQuery – Chaining. Cited 18.11.2015, http://www.w3schools.com/jquery/jquery_chaining.asp

W3Schools 2015e. HTML Entities. Cited 26.11.2015, http://www.w3schools.com/html/html_entities.asp

W3Schools 2015f. HTML ANSI (Windows-1252) reference. Cited 26.11.2015, http://www.w3schools.com/charsets/ref_html_ansi.asp

W3Schools 2015g. ASP.NET web pages – adding Razor code. Cited 2.12.2015, http://www.w3schools.com/aspnet/webpages_razor.asp

W3Schools 2015h. ASP.NET MVC – Views. Cited 2.12.2015, http://www.w3schools.com/aspnet/mvc_views.asp

WHO 2014a. Together we can prevent and control the world's most common diseases. Cited 20.5.2015, http://www.who.int/nmh/publications/ncd-infographic-2014.pdf?ua=1

WHO 2014b. The top 10 causes of death. Cited 4.8.2015, http://www.who.int/mediacentre/factsheets/fs310/en/

WHO 2015a. Noncommunicable diseases fact sheet. Cited 19.5.2015, http://www.who.int/mediacentre/factsheets/fs355/en/

WHO 2015b. Cardiovascular diseases (CVDs) fact sheet. Cited 21.7.2015, http://www.who.int/mediacentre/factsheets/fs317/en/

WHO 2015c. Chronic obstructive pulmonary disease (COPD). Cited 31.7.2015, http://www.who.int/respiratory/copd/en/

WHO 2015d. Causes of COPD. Cited 3.8.2015, http://www.who.int/respiratory/copd/causes/en/

WHO 2015e. Chronic obstructive pulmonary disease (COPD) fact sheet. Cited 4.8.2015, http://www.who.int/mediacentre/factsheets/fs315/en/

WHO 2015f. Diabetes fact sheet. Cited 13.1.2016, http://www.who.int/mediacentre/factsheets/fs312/en/#

WHO 2016. World Health Day 2016: Diabetes. Cited 13.1.2016, http://www.who.int/campaigns/world-health-day/2016/event/en/

Wicks, P., Stamford, J., Grootenhuis, M. A., Haverman, L. & Ahmed, S. 2014. Innovations in e-health. Quality of Life Research 23 (1), 195-203. Internal source. Cited 1.6.2015, http://search.proquest.com.ezp.oamk.fi:2048/docview/1491225586?accountid=13030

World Heart Federation 2015. Urbanization and cardiovascular disease. Cited 29.7.2015, http://www.world-heart-federation.org/press/fact-sheets/urbanization-and-cardiovascular-disease/

Yle. 5.6.2012. Keuhkoahtaumatauti laittaa elämän uusiksi. Cited 19.5.2015, http://yle.fi/uutiset/keuhkoahtaumatauti_laittaa_elaman_uusiksi/5671896