Abstract

This research project explains the transition from traditional real life communities to the online communities. In addition to giving a detailed description and analysis of the new way of interacting, it also shows the negative sides of this communication form. This paper also explores a new approach on how to facilitate an online community building on the Socio-semantic web. This approach is based on the object oriented programming technique, together with platform independence. Object orientation is a programming paradigm which has shown to be very efficient and popular in programming. It divides a system into different objects, which are reused further in the programming process. The main aim of this thesis is to apply object orientation to online community building. These provided objects can be reused and modified as a foundation to other objects as well. My contribution is two-fold: the identification of the objects for online community building, together with a detailed description and model of the processes within them. And also the development of a set of platform-independent open source objects.
Acknowledgments

First of all I would like to thank the person who made it possible for doing this thesis, Dino Karabeg, my mentor. Without your way of seeing things and solutions, this thesis would not be realized nor would it be possible to achieve. You gave me inspiration and motivation to carry out the whole process in finalizing this thesis. Your ideas and visions lead along this path of accomplishment and success. Thank you for the enormous knowledge and experience you have given me the past year, which I will never forget.

Thank you Tor Anders Dybing, my colleague and friend. You were always there for me and gave me clever advice, through the tough and stressful times. Your patience and smart solutions helped me through a lot of knots. Thanks to the NaCuHeal association for letting me implement and show my work on their website. Thank you David R. Woolley for publishing my work on your website and making them available for the world. I would also like to thank Tommy Bäckström for giving me good advices and solutions to problems occurred on this journey. Thank you Nicolai Aas, my former teacher, for reaching out a hand when it was needed. And of course, my dear childhood friend, Jacob Segai. Thank you for always being there for me and helping me out when I called.

Finally, I would like to thank my beloved family who has always been there for me. In particular, thank my father, without whom I would not be here nor would I be the person I am today. Your hard working effort and joy of life have given me the courage and possibility to accomplish what I have attained in my life. Last but not least, thank you my dear mother, for always giving affection and attention in good times and bad times.

Selver Tosic
Contents

1 Introduction.................................................................................................................. - 1 -

2 Background .................................................................................................................. - 5 -
   2.1 The decline of traditional communities................................................................. - 5 -
   2.2 Online community building.................................................................................... - 7 -
      2.2.1 Negative aspects of online community............................................................. - 12 -
   2.3 Object orientation.................................................................................................... - 14 -
   2.4 Socio semantic web .................................................................................................. - 19 -
      2.4.1 Socio-semantic web artefacts ........................................................................... - 20 -

3 Objects, functions and processes in online community building ......................... - 23 -
   3.1 Analyzing online communities’ facilities................................................................. - 23 -
   3.2 Describing the objects and functions.................................................................... - 25 -
   3.3 Modeling the processes ......................................................................................... - 27 -
      3.3.1 Use case overview............................................................................................. - 28 -
      3.3.2 Activity diagrams ............................................................................................. - 29 -

4 Objects - user view........................................................................................................ - 38 -
   4.1 Requirements.......................................................................................................... - 38 -
      4.1.1 Apache Web Server........................................................................................... - 38 -
      4.1.2 PHP5 server....................................................................................................... - 40 -
      4.1.3 Database server, MySQL.................................................................................. - 42 -
   4.2 The installation of the object(s)............................................................................... - 44 -
   4.3 The Poll Object....................................................................................................... - 46 -
   4.4 The Registration Object.......................................................................................... - 51 -
   4.5 The Login Object.................................................................................................... - 54 -

5 Objects - implementation.............................................................................................. - 55 -
List of Figures

Figure 1: Maslow’s hierarchy of needs ................................................................................. - 11 -
Figure 2: Illustrating an ontology domain model ................................................................. - 20 -
Figure 3: Illustrating the use of taxonomy on a site selling vehicles and dwellings ........ - 21 -
Figure 4: Showing the example of a tag cloud .................................................................... - 22 -
Figure 5: The fundamental objects, functions and processes in a SNS, OC or VC ........... - 24 -
Figure 6: Use case overview of the processes and actors in an online community system  - 28 -
Figure 7: Process flow; login of a user ............................................................................... - 29 -
Figure 8: Process flow; sending a newsletter ..................................................................... - 30 -
Figure 9: Process flow; sending of a mail and messages to users ...................................... - 30 -
Figure 10: Process flow; adding an item to web shop ......................................................... - 31 -
Figure 11: Process flow; modifying and deleting an item in web shop ............................... - 31 -
Figure 12: Process flow; creating a poll ............................................................................. - 32 -
Figure 13: Process flow; deleting a poll ............................................................................ - 32 -
Figure 14: Process flow; viewing statistics of a poll ............................................................ - 33 -
Figure 15: Process flow; registering a member to the OC .................................................. - 33 -
Figure 16: Process flow; paying membership in an OC ....................................................... - 34 -
Figure 17: Process flow; searching in an OC .................................................................... - 34 -
Figure 18: Process flow; modifying or deleting a profile .................................................... - 35 -
Figure 19: Process flow; registration of a vote ................................................................... - 35 -
Figure 20: Process flow; shopping in the OC’s web shop .................................................... - 36 -
Figure 21: Process flow; using the forum or writing a blog ................................................ - 37 -
Figure 22: Process flow; use of chat tool in an OC ............................................................. - 37 -
Figure 23: Showing what start.php looks like .................................................................... - 44 -
Figure 24: Showing the handling of the information input ................................................... - 45 -
Figure 25: Showing a successful installation ...................................................................... - 45 -
Figure 26: The local menu in the vote object ...................................................................... - 46 -
Figure 27: The form for registering a poll .......................................................................... - 47 -
Figure 28: The options available for each poll registered .................................................... - 48 -
Figure 29: Illustrating the statistics of a certain poll ............................................................ - 49 -
Figure 30: Illustrating a poll for the members of an OC ..................................................... - 49 -
Figure 31: Showing the message of a successful vote registered ........................................ - 50 -
Figure 32: Showing the index page of the register object ................................................... - 51 -
Figure 33: The registration form for a becoming member ................................................... - 52 -
Figure 34: Illustrating the button for the paying of OC services ........................................ - 52 -
Figure 35: Example of a typical payment form from PayPal .............................................. - 53 -
Figure 36: Showing the index page for login object ............................................................ - 54 -
Figure 37: Illustrating the pages only available for members of the OC, after passing the security check .................................................................................................................. - 54 -
Figure 38: The interaction between the database and administrator and the interaction between the database and the OC user .................................................................................. - 56 -
Figure 39: Illustrating the relation between the table community_poll_statistics, the user and administrator ............................................................................................................ - 58 -
Figure 40: Illustrating the storing of user input in database ................................................ - 59 -
Figure 41: Illustrating the interaction between the computer and the database regarding the login object. ............................................................................................................................... 60
Figure 42: Showing an example of the file userinfo.php ................................................................................................................................. 62
Figure 43: Sequence diagram showing the sequences when creating a poll ................................................................. 64
Figure 44: Sequence diagram showing the decomposition of the lifeline PollObject................................................. 65
Figure 45: Sequence diagram showing the sequences when registering a user’s vote ............................................. 66
Figure 46: Sequence diagram showing the decomposition of the lifeline PollObject........................................ 66
Figure 47: Sequence diagram showing the sequences when registering a member in an OC.................. 67
Figure 48: Sequence diagram showing the decomposition of the lifeline RegisterObject .................. 68
Figure 49: Sequence diagram showing the sequences when a user is logging into an OC ........... 69
Figure 50: Sequence diagram showing the decomposition of the lifeline LoginObject................................. 69
Figure 51: Showing the choices of objects available on NaCuHeal.org .................................................. 74
Figure 52: Showing the registration form on NaCuHeal.org portal ................................................................. 75
Figure 53: Showing the log in form for users on NaCuHeal web site ................................................................. 75
Figure 54: Showing the start page after a user log in ...................................................................................... 76
Figure 55: Showing the log in form for administrators on NaCuHeal web site ......................................... 77
Figure 56: Showing the start page of the administrator, after logging in ................................................. 77
Figure 57: Showing the Make Conference Form ............................................................................................. 78
List of Tables

Table 1: Listing the reasons why people gather in online communities, (Boetcher, Duggan et al. 2002). ................................................................................................................................................................................................................................. - 8 -
Table 2: Showing the community_poll table and its distribution.................................................. - 55 -
Table 3: Showing the distribution in the community_poll_statistics table and its fields ............ - 57 -
Table 4 below: Showing the distribution in the table community_users....................................... - 59 -
1 Introduction

Human beings are social beings, and have an enormous need for human connection and interaction. As Putnam showed us, the communal life is undergoing a profound change. On the one hand the traditional community life has been declining, where several traditional ways of social interaction have diminished through the past three decades. In 1992 three quarters of the U.S. workforce indicated that “the breakdown of community” and “selfishness” were “serious” or “extremely serious” problems in America, says Putnam. He points at a variety of consequences which this decline has brought about. In areas where social capital is lacking, the effect of poverty, adult unemployment and family breakdown are magnified. Even human health and happiness have suffered from the lack of social capital, becoming a serious public health challenge. According to Putnam, socially isolated people are more likely to smoke, drink, overeat, and engage health damaging behaviors. In addition to this, people lacking social capital have a larger tendency to have heart attacks, cancer, depressions and premature deaths of all sorts. If this is not enough, Lisa Berkman, one of the leading researches in the field, has speculated that human loneliness is a chronically stressful condition to which the organism responds by aging faster, (Putnam 2000).

A few of us still share plowing chores with neighbors, while many more pitch in to wire classrooms to the Internet.

Robert D. Putnam – Bowling Alone

On the other hand, Sulin Ba exposed that there is a strong move towards the online community as a social structure, (Ba 2001). Boyd and Ellison’s research showed the rapid growth of the online communities’. From having only one major social network site in 1997, today there are over 1900 registered in the Online Communities Directory, (FIC 2008). These numbers are encouraging. On the 9 of August 2006, the social network online newspaper Mashable.com wrote that Myspace.com had reached 100 million accounts in their community, (Cashmore 9 August 2006). In addition to this Facebook.com has, according to the Norwegian newspaper Dagbladet.no, attained 45 million users within a three year period, (Omdahl 30 October 2007). These communities are not only helping rebuilding social capital, they may also promote healthy living. While social isolation can lead to deteriorating health,
people are using online communities to cure health problems, (Crespo 2007). Wilson and Peterson’s research, plus Putnam’s book, confirms this growth of such internet usage. They explain that Internet as a social interaction medium has grown in the past years and that it will have an enormous influence on the future state of social capital, (Putnam 2000; Wilson and Peterson 2002). Since 1991, computer networks have received enormous public attention. Today, 510 million computers are online in Asia, 348 million in Europe, 238 million in North America, 126 million in Latin America, 44 million in Africa, 33 million in the Middle East and 19 million in Oceania. Back in the early 90s, very few outside the US had Internet access. In 2007, approximately 20% of the World’s population were using Internet, and the figure is still growing, (Internet World Stats 2007). So, while local communities are bleeding members, their values and traditions forgotten, the development of numbers of online communities and Internet users has increased dramatically recently. This can make us ponder if local communities are moving towards online communities.

The purpose of this thesis is to introduce object orientation (OO), an approach which has proven to be very efficient in computer programming. In order to introduce this I will use its effective properties to support online community building. By first dividing an online community and its functions into objects, I have developed platform-independent objects as open source, primarily meant to serve the facilities of an online community. Offering the newest technology and technique for online community builders is very important in order to achieve a long living online community, (Butler, Sproull, Kiesler and Kraut in Weisband 2007). Making these objects open source entail making the source code available to anyone. In combination with the newest technology for web programming and object oriented technique, maintaining, customizing and further developing is easier. Earlier there have been open source objects created similar to the ones provided here and a collection of these are available at David R. Woolley’s site http://thinkofit.com/webconf/forumsoft.htm. To compare a few, a discussion forum for online communication is linked to the web site http://beehiveforum.sourceforge.net/about.html and a message board can be found at http://zorum.phpoutsourcing.com. However, there are few open source objects that are platform-independent. The objects provided with this thesis are platform-independent, as well as customizable. By using “cut and paste”, these objects provided can be added to any site and with small modifications work perfectly in any online community, this makes them available at anytime from anywhere. Together with this independency and flexibility they will increase the small number of such objects available in the free software community. Broadcasting of
the object is obtained in cooperation with David R. Woolley, a pioneer in online conferencing for over 30 years, who has given the objects mentioned above plus my developed objects, publicity and recommendation on his online communication web site http://thinkofit.com/webconf/forumsoft.htm#tosic.

*Technology has its most profound effect when it alters the ways in which people come together and communicate.*

Kollock and Smith - Online Communities

By doing this I have also undertaken to explore the use of object orientation in web programming and modeling, in hope that the object orientation paradigm will show its efficiency in web programming. I wish that online community builders will use this approach to build more online communities and fulfilling people’s social needs. Through extension and modification, which is easily done with the object oriented technique, online community builders can extend existing objects to create other objects with different purposes. This approach can be used develop all objects facilitating an online community. By increasing the numbers of online communities, I covet that people will use them and interact. Hopefully, this will benefit mental and physical health and strengthen social capital.

My motivation for doing this thesis is to help others and contribute to keep and maintain social capital. The other motive is the opportunity to enquire whether object orientation can help increase social capital on the social-semantic web, specifically towards online communities. As an information scientist I sincerely hope that social capital can be rebuilt through the object oriented approach.

My questions when working on this thesis have been: What is the basic set of objects that can fulfill the main functions and facilitate the main processes in an online community? How can these objects be implemented and deployed in the right way? Can the source code of a few objects be used as a foundation to create other objects with different purposes?
In addition to this introduction, this thesis has eight chapters plus the appendix. The presentation of these chapters is similar throughout the whole paper with an informative text at the very beginning of every chapter. This will help the reader to be prepared for the coming content. First I emphasize the background of this research which is the foundation of the thesis. Before I present the setup, development and demonstration of the developed objects, the analysis of the online community and its processes is described in chapter three. Finally, the conclusion is presented together with bibliography and source code in the appendix.

Words like he, his and him may be read as she, hers and her.

The words online community (OC), virtual community (VC) and social network sites (SNS) may all be read as online community and vice versa through the whole paper.

Online community, virtual community and social network sites in this thesis are set to mean a site on the Internet where people come together, communicate and interact through different tools provided by the online community.
2 Background

In this chapter I introduce some background themes which are important for understanding this thesis.

2.1 The decline of traditional communities

Bonding social capital is far more often a good thing than not, Putnam highlights in his book. Besides explaining the importance of social capital, local communities and togetherness, the book also emphasizes how these approaches psychologically support human beings. It also shows that the foundation of social capital and strong bonds are mainly achieved through interaction, which again may be achieved through community activities and get-togethers. However, many activities that America’s ancestors established are diminishing. The religious values and community which topped the American society until after the Second World War are also to a large extent forgotten. The religious feelings in America were so profound at this time that the US was the most fundamentalist and religiously traditional country in Christendom, (Putnam 2000). People involved in organized religion knew more people than those who were not involved and 60% of members of churches volunteered compared to 30-35% of the non-members. How could such a strong commitment and feeling of community diminish so quickly?

*No sector of American society will have more influence on the future state of our social capital than the electronic mass media and especially the Internet.*

Robert D. Putnam – Bowling Alone

Putnam’s answer to this question is that they have been pulled apart from one another and from their local communities over the last third of the century, without first noticing it. The situation that America and the western World are in today, is frightening and catastrophic. A survey conducted amongst infant parents in 1987, concluded that 77 percent said that the nation was worse off because of “less involvement in community activities”. Some of the side effects due to the decline of traditional communities are less political participation, worse health, unhappiness, less people volunteering, honesty and trust decreasing, while pressure on time and money increasing, (Putnam 2000). Going from being the people of a nation without
worries and a life where people took care of each other to the exact opposite we see today. From taking the leaves before they blow onto the neighbor’s yard and lending money to a stranger for a parking meter, America’s people now worry about whether they got the right change from the clerk or double-checking if they locked the car, (Putnam 2000). Modern habits, interests and trends which involve less human contact and minimum involvement in the lives and activities of previous generations, are the main reasons for this collapse. Although several factors need to be considered in this development, crossing the bridge from “the old way of life” to contemporary information technology is possibly the most important. Local communities of today seem to move towards the virtual ones, i.e. establishing grip on the internet. In other words, they are seeking social capital through online interaction.

*If we are to reverse the adverse trends of the last three decades in any fundamental way, the electronic entertainment and telecommunications industry must become a big part of the solution instead of a big part of the problem.*

Robert D. Putnam – Bowling Alone
2.2 Online community building

*We are made for conversation with our kind. ... and to/ communicate and share the communications of others.*

John Dewey - Characters and Events Vol 1 (as quoted in Preece 2000)

Communities have existed as long as there has been life on this planet. Social interaction has always been in the interest of every man and woman. Whether it is virtual or real life, we all have some kind of understanding of this term. The definitions of a community are many, but related to each other; “A group of people having common interests”, “Sharing, participation, and fellowship”, (thefreedictionary.com 2008).

An online community (OC) is – obviously - a community which can be found online. It is also known as virtual community or social software, which may be described as software helping people to socialize on the web. Just like ordinary people meet at the gym or cafés, online communities are virtual places where people meet when connected to the Internet. In these online communities they meet for various reasons, depending on the purpose of the OC. Some people intend to discuss, some to bond friendships and others to connect with business partners. Boyd and Ellison describe in their article how these communities try to gather people with a specific purpose and / or goal. Other online communities, however, are gaping for wider approaches with several purposes and goals, tempting the users with various communication offers like blog and photo / video sharing. By capturing more people from the audience, they obtain marketing as well as the members operate the community through using the tools provided for them. Boetcher, Duggan and White made an overview of reasons why people are members of online communities.
| **Socialize** | Meeting people, playing around, sharing jokes, stories and just taking interest in each other. Communities like this often focus around bulletin boards and chat rooms. |
| **Work together (business)** | Distributed work groups within companies and between companies use online community to build their team, keep in touch and even work on projects together. |
| **Work together (community – geographic)** | Community groups such as soccer teams, school groups and others have used online community to provide forums for information and discussion, helping bring groups together. |
| **Work together (issues)** | People who share interests in issues and causes. Support groups for people dealing with certain diseases, causes such as politics or the environment, or people studying together, all can form a nucleus for an online community. |
| **Have topical conversations** | Communities of people who enjoy conversations about topics and shared interests. That could be anything, e.g. relationships, business and finance, health, hobbies, religion, music, international. |

Table 1: Listing the reasons why people gather in online communities, (Boeticher, Duggan et al. 2002).

Like stated above, more than 1900 different online communities are registered in the Online Communities Directory. A minor part of these are key SNSes which have shaped the destiny of the others, (Boyd and Ellison 2007). In addition to these, several online communities are not registered on this site, making the total number even higher. One community may gather people trying to connect to old classmates, e.g. [http://www.facebook.com](http://www.facebook.com), while others are
embracing nature enthusiasts, e.g. http://www.nacuheal.com. Spending a lot of time within the community is a contribution to the community itself. By using the tools available, e.g. sending messages, sharing photos, debating etc, the user becomes a part of the community.

How different online communities work varies from community to community, but Boyd and Ellison describe a general understanding of social network sites in their article. They emphasize that most SNSes allow users to create a public or semi-public profile within a bounded system, which can be related to them as their “home area”. In addition to this, the users have a list of other users who they somehow are connected to. This could be “friends” within the community or friends from the real life, (Boyd and Ellison 2007).

_The Internet offers a variety of technical tools and mechanisms to support online social interactions in groups._

Boyd and Ellison - Social Network Sites: Definition, History, and Scholarship

In order to make online community users collaborate and interact with each other in hope of building social capital, communication tools need to be provided. Therefore, it is important to define the various technologies regarding this comprehensive and important communication function in an OC. Most noteworthy is Amy Jo Kim who in her book, “Online Community Building on the Web”, gives a lucid overview of different aspects of communication within OCs. Still, I want to emphasize that there may exist more communication tools today.

**Mailing Lists** allows sending a message to several email addresses. First the administrator creates a list of email addresses and gives this list an address (ID), which is used to send the same message to all the email addresses on the list. The recipients of this message may then reply to the email, and the reply would be seen by everyone on the list. Thus, interaction between the users of the email addresses is possible. The mailing list software is the easiest kind of online gathering and also the easiest to maintain, create and participate in. This solution is superb for small groups, temporary conversations, newsletters, announcements and new-started communities.

**Message Board** is also known as newsgroup, bulletin board, conference, discussion or forum. Like mailing lists, this type of interaction is asynchronous, which means that participants do
not need to be at the same (virtual) place at the same time to have a conversation. Because of this, the endurance can vary from minutes to months. They are great and often used for question answering and topic conversations, like a support forum, specified topic discussions etc. The message boards are especially good for providing context, history and a sense of place.

**Real Time Chat** is a vast type of synchronous interaction, which means that people in different physical locations are communicating with each other at the same time. It enables anyone who is connected to the system to correspond instantly with any other participant. This type of live interaction has many subgroups which could work independently and towards a different audience. It is emphasized that it is specially suited for holding scheduled events, preparing for live events, hanging out (relaxing, flirting, gossiping, etc). Real time chat can be from black and white web based text chat to graphical chats and voice chats, including characters, arts, sounds and voices of the participants.

**Virtual world** can create an interactive, navigable environment using graphics, sounds, and animation. It also includes customizable look-a-like characters which represent the members in a virtual world. The inhabitants interact with each other through communication tools offered, e.g. text or audio. In addition to this they are also provided a variety of games and activities, depending on the environment of the virtual world.

Why do people bother interacting with each other, and why has there always been a human interest in socializing? The needs of the humans are many and evolving, (Kim 2000). By taking a look into Abraham Maslow’s hierarchy of human needs, given in Amy Jo Kim’s book, it is clear why human beings in fact are social beings.
Maslow believed that people are urged to satisfy needs starting at the most basic ones to achieve self fulfilment. Without satisfying the lower ones, the needs above can never be fulfilled. Psychological needs - like water, food and health - are the most basic: deprived of these, a person cannot satisfy any other needs and will eventually die. Safety – the need to feel secure from crime and war, plus the need to live in a fair society - can never be fulfilled until we have food, water etc. The social need is the most relevant in this case. It is the ability to give and receive love; the feeling of belonging to a group or family cannot be achieved unless we feel safe away from danger. In the case of online communities, this means that people need to belong to a community as a whole, where they socialize with other human beings. By satisfying the social needs humans can please the self esteemed needs, like self-respect, the ability to earn the respect of others and contribute to society. At the top we have the needs which demand all the other needs to be fulfilled in order to be realised, (Kim 2000).

Because socializing is such a fundamental need to the humans, it has to be considered and given high priority when building an online community. Finding these needs and customizing online communities to them, takes effort and time. Unless the needs are real and fulfilled, the users will not be motivated to come back, (Kim 2000).
On the other side, bonding and online interaction can have a negative impact on human beings, (Ba 2001). Media has given a lot of attention to questions towards trustworthiness, personal safety and Internet fraud. The most important issue here is the differences between real communities, where you know, and online communities where you often do not know, people you interact with. Talking and discussing over a screen with someone you do or do not know, sharing information with people that could be your neighbour or from another continent, is a security issue which needs to be mentioned in this paper. The need to feel secure is also one of the most fundamental needs in Maslow’s hierarchy.

2.2.1 Negative aspects of online community

*On the Internet, nobody knows you are a dog.*

Peter Steiner’s drawing - the New Yorker July 5, 1993 (as quoted in Ba 2001)

What kind of effects can we expect by gathering people on the web, especially in terms of relations between those that have not met before? Even though most are positive, there has been an unfortunate increase of negative effects. As early as 1999, there was a 38% increase in Internet fraud complaints, (Ba 2001). Interactions introduce to misbehaviour without paying reputational consequences. This misbehaving can draw parallel to Putnam’s book saying people have moved away from traditional to newer forms of negative behaviour. The amount of unwanted behaviour over the Internet and SNSes varies a lot. Parameswaran, for instance, points out in his research how many of these environments that facilitate and rely on exchange of multimedia content, which are potential carriers of viruses and other threats. Granted access to a public domain if carrying a virus or some kind of threat is a potential nightmare in terms of security, but is still just a minor part of the problem, (Parameswaran 2007).

Online communities have also bred some fear and criticism regarding the security of the most valuable and vulnerable people, our children. The communities can serve as dangerous hunting grounds for online criminals, such as identity thieves and stalkers. While the popularity of the online communities has grown, the number of online crimes has also increased dramatically. Concerning this development, some schools and parents have banned online communities like MySpace, where paedophiles and other criminals may masquerade
under other identities, (Parameswaran 2007). OFCOM, a research team from the UK concluded in their research that 41% of the children and 44% of the adults attending an online community, are leaving their privacy settings too open – visible to anyone, (OFCOM April 2008). These alarmingly large numbers are wake-up calls to the parents as well as to online community builders. Some users even say they derive enjoyment from ‘collecting’ lists of people with whom they have an online connection but often have never met, (OFCOM April 2008). These people may be “masquerading criminals” as Parameswaran calls them. This is frightening, considering the fact that nearly one third of children from eight to eleven who have Internet access, also have an online community profile, (OFCOM 2008).
2.3 Object orientation

*Computers follow instructions. Computers are stupid. They only do what you tell them to do, or what someone told them to do-- Not what you want them to do-- Whether you want them to do it or not.*

Robert J. Banis, PhD - Teaching Professor of Logistics and Operations Management

Programming languages are used to make applications or programs, in other words the programming language is the engine of an application running on a computer. Therefore it is implied that it is used with the exact right syntax in order to run a program properly.

A computer program is the instructions designed particularly to tell the computer what it should do. It may be compared to phenomena we are familiar with; a food recipe or a do-it-yourself manual. Because the computer does not understand our natural languages, the instructions have to be written in programming languages, e.g. PHP, C++, Java, etc.

Object Oriented Programming (OOP) is a term that popped up in the 1960s as the computer world was overwhelmed by several programming languages, also called *spaghetti code*. This was because of the unstructured and messy syntax. This was a huge problem, not only for the programmer himself, but also for the people that later would have to read this code, maybe for maintenance or expansion. The need for a structured, organized code which would make maintaining easier, was obvious. Ole-Johan Dahl and Kristen Nygaard of the Norwegian Computing Centre in Oslo came up with a programming technique called *object oriented programming (OOP)*. Its purpose was to be a tailor-made technique for creating applications, especially large applications, in a structured and organized way. Very roughly, OOP groups things into different classes of objects or modules where each class of object is responsible to define its own data. The structure is hierarchical. We can draw parallels with the real life, saying an object could be a person which contains the properties of this person, how he behaves, what he does, etc. In principle each object may be viewed as an independent little machine with its own data and a distinct role or responsibility.
However, the object oriented programming technique has many concepts and are well described by Ivar Jacobson in his book, which I will consider when describing them below, (Jacobson 1992).

**Class**

A class is like a container of the data it is responsible for, saved in attributes. Within the class we describe the attributes with what kind of data it will store and with the operations that we want to do with this data. A class should be recognizable to a non-programmer familiar with the problem domain, meaning that the characteristics of the class should make sense in context. For example we could have a class named Car which would consist of data shared by all cars, like colour, seats, etc.

**Method**

Methods are operation(s) that the class can do, given the right syntax and parameters. The methods describe exactly the behaviour of the class and its properties. The abovementioned class Car has a method `drive()`, which would describe what is needed to use this operation. A class may have many methods, the class Car would have a method `stop()` too, which obviously would give the car instructions to stop. In other words, it describes the behaviour of the class:

- **Class**
  - Attributes
  - Method1()
  - Method2()
Inheritance

In nature and society it is common to divide things into hierarchies. This is in order to understand them better. For example:

- **Animal** (parent / super class)
  - **Mammal** (child / subclass)
  - **Birds** (child / subclass)
  - **Fish** (child / subclass)

- **Vehicle** (parent / super class)
  - **Car** (child / subclass)
  - **Truck** (child / subclass)

All the subgroups (**subclasses**) have the properties of the group above (**parent**) in addition to the properties the groups themselves got. Every animal is not a mammal, a bird and a fish, it is grouped into one of these subclasses, with its own special properties that the other groups do not have. These properties are attributes and methods and other aspects described in the parent class.

The grouping can go on and on. We could for example divide the Fish class above into Freshwater and Saltwater, which would inherit the properties of its parent Fish. The Fish class would again inherit the properties of its parent Animal.

Encapsulation

This property conceals the exact details of what is happening in a class from an objects’ point of view. If we take the example above with a method `drive()` in the Car class, it obviously needs a method `pressGaspedal()`, used to describe the pressing of a gas pedal in order to make a car run. However, the main reason for using this feature is to prevent others from changing the code, which could make the program useless. This is done by giving other classes, related to the class Car, read access to specific attributes (information) and methods (operations).
This is done by writing the right code syntax. For example: if we in the future get a car that does not have pedals, a class `NopedalCar` is not necessary. Then it is not needed for class `Car` to have read access to this method, which could also work as a security feature. The programmer can do this by making the attributes and methods public, protected or private. This would determine whether they are available to all classes, sub-classes or only the defining class.

**Abstraction**

In order to simplify the complexity, abstraction is used. This can be done by a further grouping within the hierarchy, in order to work with only one object. If we take the abovementioned class car, and something has gone wrong in the engine, then we want to make an object of that part only: we want to work on the engine and not e.g. on the gearbox. A car is made up of several different parts and we do not need to know how the other parts are working, except when they are somehow connected (*messages ending/receiving*).

- **Vehicle** (parent / super class)
  - **Car** (child / subclass)
    - **Engine** (child / subclass of Car)
    - **Gearbox** (child / subclass of Car)
    - **Fueltank** (child / subclass of Car)

In this case we do not want to make an object of the parent class Car, but only the subclass Engine, which we can see is the sibling to Gearbox and Fueltank. In other words, they are somehow related. Then we can focus on the engine part, without touching the other parts, which can communicate through messages.
Polymorphism

There are several types of polymorphisms. Polymorphism is basically the ability of using the objects of classes from the parent class as well as referring objects to the existing subclasses and the subclasses we may add later. In short terms, polymorphism allows you to treat derived class members just like their parent class' members. If the class Vehicle had a method called tankFuel(), we would first have to find out the type of fuel the car or truck is using. The polymorphism lets us get information from our subclasses easily, in this case to prevent us from tanking the wrong fuel.

Object

An object can be seen as an instance of one class which is described above. We can create an object if we need the properties of a class or simply just want to register another sample of that class. Programmers create objects in order to use information of a specific class, and then they automatically get access to the class information that is stored. Let us consider a class Person, parent to the class Jerry, and we want to know how long Jerry’s arm is because we are buying him a jacket. Then we create an object of the class Jerry, which contains information about the length of Jerry’s arm. By using the right syntax (100% accurate writing language), we can make many different objects of one class and then use them to our purpose. If we later find out that we need to save similar, but not exactly the same information, we can simply use a certain object and expand and modify it according to our purpose.

In short, OOP is a programming paradigm that uses objects to design applications and computer programs. It is a structured way to program and a major advantage is the reuse of the code instead of rewriting. Instead of using the whole program or start programming from scratch every time you need something similar, one can use one part of a program and modify it to ones satisfaction. So in fact, a whole program written with object oriented technique, is built up by several objects that are working together in order to make the program run as one. This opens for using one object for one purpose instead of using a whole program. Imagine if you only need a steering wheel, would you buy the whole car?
2.4 Socio semantic web

*Socio-semantic web is relying on the pace-layering of ontologies, taxonomies, and folksonomies to learn and adapt as well as teach and remember.*

Peter Morville – Ambient Findability

L’Hédi Zaher, Jean-Pierre Cahier and Manuel Zacklad explain the Socio-semantic web (S2w) in their article as an extension to the Semantic Web, (Zaher, Cahier et al. 2007). Cahier and Zacklad were the first to coin this term in 2003, pointing out that this extension focuses on the user of the web with respect to Information Technologies. Instead of involving a high level of automation of meaning, the Socio-semantic web rather focus on the situations where human beings are involved and interacting during the whole lifecycle of applications, (Zaher, Cahier et al. 2007). In other words, S2w emphasizes the importance of human created loose semantics as means to fulfill the version of the semantic web. Instead of relying on automated semantics, humans are collaboratively building semantics aided by socio-semantic information systems.

A socio-semantic view is used in a management point of view, to support communities that need to continuously and collectively elicit information. It is used in order to make it visible and structured, (Zaher, Cahier et al. 2007). This may be obtained through the artefacts associated with Socio-semantic web. By using the S2w artefacts, the user is in focus all the time and is the main source for a running artefact. So deploying Social semantic web artefacts in a user driven society is very informative to the user. Because both Socio-semantic web and Web 2.0 are created to make information lucid for users. They also focus on social aspects, users participation and a user driven society, (O'Reilly 2005), the artefacts and properties of these two approaches may be coherent in many places.
2.4.1 Socio-semantic web artefacts

Ontologies

An ontology defines a set of representational primitives with which to model a domain of knowledge or discourse.

Tom Gruber - Ontology

In Tom Gruber’s publishing “Ontology” from 2008, he explains the ontology in computer science relation. He describes it as a technical term that is designed for a purpose, which is to enable the modeling of knowledge about some domain, real or imagined. In his article it is also indicated that ontologies’ primitive representation typically are classes (or sets), attributes (or properties), and relationships (or relations among class members). These primitives contain information about their meanings so that the whole context of the sets becomes obvious, (Gruber 2008). The ontology can be used, in this case, for modeling knowledge about classes (or sets), their attributes, and their relationships to other classes (or sets).

By using ontology in information science one can divide information in different sets with describing attributes as their special properties. By dividing this information it is also easy to understand the relations between the different sets and properties. This can help readers in general to understand the whole domain by a describing model, as shown below.

![Ontology Diagram](image)

**Figure 2**: Illustrating an ontology domain model.

This illustration shows that both Car and Truck are some type of Vehicle, whereas Automatic and Manual transmission cars are not types of Truck in this case. The illustration could be
expanded by dividing vehicles further down the hierarchy. In our case we can classify vast online communities into sets, attributes and relationships which give a clear picture of the OCs and their activities. Modeling online community facilities can also be a good foundation for building an online community. I will get back to this in the next chapter.

**Taxonomies**

Taxonomy, in this case, is a method of organizing content on a site. For example, classifying sport in different divisions could generate this list: football, swimming, running. “Football” might for instance be further classified as English Premier League, Italian Series A, Spanish Primera Division, and so on. The exact definition of taxonomy is: “Division into ordered groups or categories” (thefreedictionary.com 2008).

By classifying information in this way, the information is more lucid for a user visiting a site. Instead of having too much information on a site misleading the users, taxonomy can help the users not to get lost in the information jungle.

![SampleSalesment.com](image)

**Figure 3:** Illustrating the use of taxonomy on a site selling vehicles and dwellings
Folksonomies

Folksonomy is a term that focuses on people and the collaboration between people. Unlike taxonomy, there are no hierarchy, no parent-child or sibling relations between these terms. These folksonomies are simply the set of terms that a group of users tagged content with, (Mathes 2004). In our case, its purpose is to help people get to the information they are looking for, e.g. on a web site with lots of links where the user often gets lost. One technique using this approach is the tag cloud; it makes the most used words light in the dark for users.

Figure 4: Showing the example of a tag cloud

Here we can see a tag cloud which contains keywords in different font sizes. This is usually found on the starting page of a web site. The fonts grow as more people search for the same word or click on the words. This way, the users can easily get to the most popular topics.
3 Objects, functions and processes in online community building

Like mentioned in introduction, this chapter will answer the questions asked in the introduction. The analysis of objects, functions and processes common in an online community, will be shown here. Through the use of object oriented modelling, I will give a lucid picture of the online community facilities and the processes within them.

3.1 Analyzing online communities’ facilities

The task I will consider here is to identify the collection of objects, functions and processes existing in most of today’s online communities. By doing this analysis I will model the collection of objects, functions and processes which are needed to facilitate online community building.

By attending myself to SNSes, as well as comparing different SNSes, I have made an overview of objects within an OC. Although this may not synthesize to all online communities, the key technological features are fairly consistent in most SNSes, (Boyd and Ellison 2007). I also want to point out that not all of these objects are necessary to make an OC run, but the framework is likely to be the same.
Figure 5: The fundamental objects, functions and processes in a SNS, OC or VC.

While the processes connected to the rectangles at the concrete level are shown as circles, the objects and functions one can find within an OC are represented by rectangles with a describing topic. In this way it is easy to see what an OC consists of, as well as it points out the operations one can do within it. E.g. the user can participate in a poll by voting, while the administrator can create, delete and view the statistics of a poll. As it is easy to see from this model, online communities are divided into two parts; one for the user or member of the OC, and one for the administrator(s) or staff. Although they are separated, where the members can use the facilities of the OC and the administrator runs it to a certain point, they are not so very different from real life places like the gym, where the members use the machines available while the administration or the staff takes care of the rest. To take a practical example
regarding a process within both of these systems, the gym and the OC, the member has to give his personal information and somehow be registered. This information is then used for something for notifying users with newsletters, notifications, payment reminding, etc.

3.2 Describing the objects and functions

As mentioned above, the functions or objects are different from OC to OC, especially the ones related to administrator(s). This is because the part ordinary members do not see. However, I will describe the most common ones and the functions that are normal to have regarding this theme. By describing the different ones I want the reader to get an idea what kind of objects are common to have and why they do have them, in this way he should also be able to understand how the members use them.

**Broadcast**

This feature is most used by administrators to keep in touch with the users of the OC in a simple way. By one transmission, also known as mass sending, administrators can send emails or messages to users to notify them about social events, payment reminding, password renewal, newsletter, etc.

**Webshop – administrators view**

When having an online shop, like most online communities do have, someone should be responsible for it. Taking care of orders, adding new articles, removing old ones and modifying existing articles is a major job. By maintaining the web shop you are giving the best offers for your members, keeping them up to date within your community branch and helping your OC become a known branch as well as having an income.

**Poll**

Polls can give reliable information about the opinions and makeup of the target audience. In order to have various types of questions every day available for the users in an OC, the administrators have to make new polls which could later be used in advance for the OC. By collecting the answers from the users, the administrators are also allowed to see the statistics of the poll, which again can help them to draw some conclusions in advance to the OC.
Registration

Like in real life community, e.g. the gym, where you have to give up your personal information to become a member, so is the deal in online communities. Normally you give your personal information in addition to community information, i.e. username, password, whether you want newsletter, etc. Later, this is your identity in the OC, which is also used to log into the OC as well as it is used to create “your face” in the community, i.e. your profile.

Search

Search is a common and extreme helpful function that allows administrators and users to search on members in the community, as well as blogs and forum topics. By having this mechanism the job is easier for administrators and users when they want to gather information about specific issues or searching for members to contact.

Profile

This is the homepage for a member, the face of a single member in an OC. Usually when members are browsing through members, this is the first thing they see and read. Together with the information entered at registration, a profile often contains information that the owner of the profile has written about himself. The user profile is often decorated with a photo, albums and colours to capture other member’s attention. The owner of the profile is often asked if it should be readable by anyone, only the OC users or just his friends.

Webshop – user view

A web shop is like an ordinary shop that occurs online, but normally without traditional payments like cash. This feature is often available for both members and non-members, letting them buy articles given by the administrators of the OC. The products in a web shop vary from OC to OC, but one will often find articles and merchandises related to the OC and their purposes.

Communicate

This is the most important function in an OC, mainly because the interaction and socializing is the reason members are active and the OC is running. Although the communication tools provided are extremely various from OC to OC, some kind of communication tools has to be
provided in order to get the online community stamp. An overview of the different types of communication functions in an OC is described in chapter 2.2.

3.3 Modeling the processes

Although the processes in an OC may vary, I have analyzed the processes with respect to the analyzed functions and provided objects. In this way I found it most adequate to make an overview use case diagram with separate activity diagrams for each process, describing the process flow. By using features provided by the UML 2.0 frame, the use case overview is supposed to catch the functions of the system, the OC in this case, and also the actors executing them. The activity diagram is used because of its uniqueness in showing the behavior in a process, i.e. the processes flow. The process analysis and activity diagrams can deviate from other process analysis, depending on the OC and its purpose. It is also worth to mention that the flow within a process may work differently depending on the process’ scope, particularly the lowest level\(^1\). The functions of the processes are described above in the subchapter “Describing the objects and functions”.

\(^1\) This can only be exactly shown if you have the most detailed information about the process, in order to show the exact flow.
3.3.1 Use case overview

By examining this overview it is clear who is using the features of the online community and who is providing them. Use cases, which represent the operations (processes) one can do inside the community, are represented by an oval with explaining text under it. However, the “owner” of the process is represented by a human figure, connected to the process. As we can see, the activity of the non member is limited with a few facilities he can use provided by the OC’s administration. On the other hand, after becoming a member and logging in, the opportunities are many, opening for interaction with other members and modifying their “mien” in the OC. The administrator’s operations within the OC are mainly towards “cleaning” and maintaining the OC and making it comfortable to their members. As we can see, there are some processes which are demanded so users can benefit from them. E.g. a poll has to be registered before he can vote, which is shown through the extend line.

---

2 Extend means that the process has to be done in order to make another work, it is like a condition.
3.3.2 Activity diagrams

The process flow within a process is shown through these activity diagrams. The diagrams are showing behavior step by step in the different processes, from the very start of the process to the end of the process, with possible outcomes or failures.

Login

![Diagram](image)

Figure 7: Process flow; login of a user

This diagram describes the process flow when an administrator or user is logging into the OC system using his personal username and password. Then a check is done towards the information stored in the database. If the combination is the same, the administrator or user is granted pass to the OC’s facilities and receives a notification of this. In a negative case, the username and password combination typed does not match the information stored in the database, and the he gets an error message.
**sendNewsletter**

Here an administrator is sending a newsletter. In that case, the newsletter is written and a search for all members’ email addresses is done in the database\(^3\). These email addresses are later used for sending this newsletter to all online community users, after having a check of entered fields.

![Process flow: sending a newsletter](image)

**sendMail & sendInternMessage**

In this case the administrator or an OC user writes an email or message to specified user(s), by using the communication tools provided. A check is done, determining whether all required fields are filled in. If everything is in order, the email or message is sent to the recipient’s and the administrator or user gets a confirmation of this.

![Process flow: sending of a mail and messages to users](image)

\(^3\) Note that only the members wanting newsletter are added to the list of recipients.
addItem

A web shop has to contain articles in order to exist at all and offer something to the audience. When a new article has arrived, information about the article has to be described. Before saving the information into the database, i.e. details, quantity, etc, a check of required fields is done. After a successful database saving, the administrator receives a notification.

Figure 10: Process flow; adding an item to web shop

modifyArticle & deleteArticle

When an article in the web shop needs to be modified or deleted, the process is very often similar, and it is here shown in one diagram. The administrator modifies or deletes a particular article, then the database is updated with the new information, and a notification is sent to the administrator telling that the process is done.

Figure 11: Process flow; modifying and deleting an item in web shop
createPoll

The administrator creates a poll by typing the date, one question and alternatives. The validation of the fields is then connected to the process, in order to have the right date format, etc. Later, another check is done to check for existing polls with the same date. If there are no polls registered on this date, the poll is registered in the database and administrator is notified. If a poll already exists, the administrator has to choose another date.

Figure 12: Process flow; creating a poll

deletePoll

When deleting a poll, the administrator first chooses which poll to delete and the database is updated automatically. The confirmation of a successful process end is shown to the administrator.

Figure 13: Process flow; deleting a poll
viewStatistics

If an administrator wants to view the statistics of a poll, the database is contacted with information regarding the exact poll, which returns the information stored. The results is then received from the database and shown to the administrator, often in form of percentage or through diagrams.

Figure 14: Process flow; viewing statistics of a poll

Register

When a user wants to become a member of an OC, he has to fill in information in the fields required. The check for valid information is then activated. If it passes this test, another check is performed, searching for users already occupying the entered username. If the username is taken, another username has to be chosen. If not, the user is registered in the database and receives a notification of a successful registration.

Figure 15: Process flow; registering a member to the OC
Pay membership

This diagram describes the process of the membership payment. As seen from this diagram personal information, i.e. name, address, etc, has to be correctly filled in, together with information about the credit card. A check whether illegally characters have been used is preformed first, and then the mandatory verifying of the credit information is done. If both tests are passed, the information about the payment is saved into the database and user gets a notification of a successful transaction.

Figure 16: Process flow; paying membership in an OC.

makeSearch

This process flow is showing when an administrator or user is using the search mechanism in the OC. The search for a specific word, username, etc, typed in the search field, has to be done first. After looking for this string in the database, forum topics and blogs, a result of the findings is represented to the administrator or user.

Figure 17: Process flow; searching in an OC.
modifyProfile & deleteProfile

Because the process flow regarding these two operations is the same, they will be shown in the same activity diagram. Because modifying or deleting a profile is a crucial operation, a confirmation from the user is always asked. If the user did this by an accident, the process is aborted. If not, the fields in the database regarding this user are modified / deleted. At the end of this process the user receives a confirmation.

Figure 18: Process flow; modifying or deleting a profile.

Vote

This vote process requires that a poll is registered on the day the user is voting. A user then chooses an alternative he wants to vote on. The alternative is then registered in the database, and a message telling the vote is successfully registered is shown.

Figure 19: Process flow; registration of a vote.
Webshop for user(s)

In this case a person, who may or may not be a member of the OC, is shopping in the OC’s web shop. The first thing shown here is the loop of adding or removing articles from the chart. This loop goes on until the user is done adding and removing article(s) from his chart, and proceeds to checkout by confirming the chart’s item(s). Then he is asked to enter personal and credit card information, which he enters. Before credit verification is started, the information is validated, checking for illegal characters. If the credit card is valid and balance is higher than the chart’s cost, the information about the payment is saved in database. The database is then updated, subtracting items from the database, before sending the buyer a confirmation of a successful purchase.

Figure 20: Process flow; shopping in the OC’s web shop.
**useForum & writeBlog**

When a user uses the communication facilities in an OC, he has to write something. If he uses a forum or writes a blog, the process flow is the same. After typing the message he needs to publish it. Then the language is checked for infelicitous words and language. If the text is approved it is published and available to the World, if not he is taken back to the first step of the process.

![Process flow; using the forum or writing a blog.](image)

**joinChat**

This diagram shows how the user is connecting to the live chat an OC is offering. By establishing connection, he may interact with other connected users as well.

![Process flow; use of chat tool in an OC.](image)
4 Objects - user view

This chapter describes the developed objects from the user’s point of view, a sort of product documentation. Because there are two main users of the OC, administrator and user, the description will cover both viewpoints. First a description of the requirements of the objects is presented, explaining in details how they can be installed and work. Further, the installation of the objects is explained, before giving a step by step documentation of the different objects.

4.1 Requirements

These objects that I have created have some requirements in order to work properly. It is demanded that some installation is done on the server\(^4\) before starting the installation of the object(s). The server where the object(s) are supposed to be installed should have the following features installed. This is just an easy explanation of how to setup the different servers in order to make my objects work. However, most servers have these features installed. If that is the case then ignore this server setup and go to the installation of the object(s).

4.1.1 Apache Web Server

On the computer where the objects will be installed, there need to be a web server installed in order to show the objects visually. I have chosen the freely available web servers from Apache.

1. Download and install Apache from [http://httpd.apache.org/download.cgi](http://httpd.apache.org/download.cgi)

---

\(^4\) A server is a computer which is connected to a network, containing the installed features available for clients (users). One server can have several features installed, and should have these features installed together with the objects provided.
NOTE: You should get the msi installer version, i.e. if the server has Microsoft Windows installed, the file should look something like: apache_2.2.X-win32-x86-no_ssl.msi

2. When you install it you will be asked to fill in server information, if you have this than fill in correct information. If you are planning on using Apache locally\(^5\), type:

- Network Domain: localhost
- Server Name: localhost
- Admin Email: Your email address

NB! Do NOT forget to check the option: “for All Users”, “on Port 80” and “as a Service”

In order to use the objects, the Apache server has to be on the whole time. Apache web server can easily be switched on, off and rebooted with simple clicks on the system tray icon down in the right corner of your screen. By right clicking on the Apache icon you get several options to do with Apache.

**Testing**

To see if everything is working so far and that it is installed the right way, open your browser and type: [http://localhost/](http://localhost/) or the server address entered in the step above.

If it shows the text: "It works!", the Apache server is installed and is ready for use. If not, go back and try again.

**Making Apache point to your files**

These steps are making Apache aware of where the objects are located and also rest of your web pages.

3. Find the location where you installed Apache, typically C:\Apache2_2\ on Microsoft Windows.

\(^5\) Using it locally is used most for testing and developing, one can write the domain adress if desired.
4. Using Notepad or other text editors and open the file C:\Apache2_2\conf\httpd.conf and search 6 for DocumentRoot. Change it from something like DocumentRoot "C:/Apache2_2" to the location where your HTML files are and where you will place the object(s) too, e.g.

DocumentRoot "C:/OnlineCommunity/"

NOTE: The httpd.conf could be a little messy, so make sure you change the correct line which is DocumentRoot and which is down7 about line 225.

5. Then scroll down about one page or search for <Directory "C:/Apache2_2/htdocs"> and change it to point to the same location you set DocumentRoot to in step four.

6. Restart apache to make the changes take effect

4.1.2 PHP5 server

Because the objects are written in PHP5 programming language, the server must understand the code written in order to run the objects.


Notice: If the server runs on Windows platform, then you should download the latest version of PHP5, something like php-5.2.X-Win32.zip. Keep in mind to download the Windows Binary zip package that will work with Apache.

2. Unzip PHP on your computer, for example C:\php\ 

3. Rename the file C:\php\php.ini-dist to php.ini

4. Open php.ini in Notepad or other text editors and scroll down about halfway through the file. Then look for doc_root and change it to point to whatever your Apache DocumentRoot is set to (step 4 in Apache). Example: doc_root = "C:\OnlineCommunity"

6 A search within a document can be obtained by normally holding Ctrl down and pressing f button, (Ctrl+f).

7 Searching for an exact line can be achieved through the Go To – function, (Ctrl+g).
5. Also search for the line: **;extension=php_mysql.dll**
   and remove the ; in front of that line.

6. Scroll down about seven more lines and change the **extension_dir = "/"**
   to the location of the ext folder after you unzipped PHP (in step 2 from above), e.g.
   **extension_dir = "C:\php\ext"**

7. Using Notepad open **C:\Apache2_2\conf\httpd.conf** - see the Apache tutorial above
   (step 4). Either at the very beginning or end of the file copy and paste these three
   lines:

   ```
   LoadModule php5_module "C:/php/php5apache2_2.dll"
   AddType application/x-httpd-php .php
   PHPIniDir "C:/php"
   ```

   **NOTE:** Be sure to change BOTH C:/php parts to the directory you installed your php
   to.

**Testing**

Restart Apache if it is already running. If it doesn't start or you get errors, use your Apache
"Test Configuration" shortcut in the Start Menu to see why and solve the problem.

To test your PHP simply create a test.php file in your Apache "DocumentRoot" folder, e.g.
C:\OnlineCommunities\. Open your test.php file with Notepad or other text editor, copy these
three lines and then load the file in your browser like http://localhost/test.php

```
<?php
phpinfo();
?>
```

If you get a long list of text, php variables and settings, etc, then the PHP server and Apache
server are working perfectly together.
4.1.3 Database server, MySQL

The last step of the requirements is to install and set up the database where the information should be stored. The free database working best together with apache and PHP5, is MySQL.


   NOTE: If the server runs on Windows platform, make sure you get a Windows Essentials (x86) binary version. The filename should be something like: mysql-essential-4.1.22-win32.msi

2. Install the file you downloaded with the following settings when asked:

   - Typical Setup
   - Skip Sign-Up
   - Make sure "Configure the MySQL Server now" is checked
   - Choose detailed Configuration
   - Choose developer Machine
   - Choose multifunctional Database
   - Choose “InnoDB Tablespace Settings” - leave everything default
   - Choose decision Support (DSS)/OLAP
   - Make sure "Enable TCP/IP Networking" is checked and leave the port number at 3306 (at this point, if you have a firewall, it will usually try to access itself on the localhost)
   - Choose standard Character Set
   - Check "Install As Windows Service"
   - Enter root password and it is recommend leaving "Enable root access from remote machines" unchecked
   - Press "Execute" button and it will install and set it up
Getting PHP5 to work with MySQL - Official information

In PHP5 the built-in support for MySQL is unfortunately removed. To get this to work, the easiest way is to copy the mysql library file by hand.

3. Open the folder you unzipped your PHP to (Step 2 in PHP setup). Copy the `libmysql.dll` file (should be located in C:\php\) into your Windows' System folder (on XP it is usually C:\Windows\System32\ although might be C:\WinNT\System\ on other Microsoft Windows versions).

4. Restart Apache and see if you get any errors. If it complains about "php_mysql.dll", either your extension directory isn't correct or Windows can't find libmysql.dll, i.e. you have not copied it into the right folder, so try step 3 and 4 again.

Testing

If you have setup the servers correctly you can start the installation of the open source object(s). By running the installation of the object(s) described below in this paper, you can easily see if everything is installed correctly. If the installation is successful and everything is working as it should, then you have connection between the different servers.
4.2 The installation of the object(s)

Before one can use the objects provided, an installation file for the connection to the MySQL database server has to be set up in order to establish connection. It is the same easy procedure for every object I have created so that the developer does not get confused in any way. The installation file, which is called “start.php”, is placed in PollObject/start for the poll object, RegisterObject/start for the register object and LoginObject/start for the login object. This installation file should be loaded in your browser and installed before you can use the objects available.

To be able to set it up successfully, you need correct information about your MySQL server, i.e. your MySQL server’s address, username and password. If you are planning on using the objects locally, the information in the fields should be:

MySQL server: localhost

Server username: the username typed at the MySQL server setup in previous chapter

Server password: the password typed at the MySQL server setup in previous chapter

![Server information](image)

Figure 23: Showing what start.php looks like.

---

Notice that this is only done once at the very start of the installation of the objects. Doing this several times with wrong information can cause errors in the connection settings which could lead to losing information.
By entering the information in the three fields and pressing button *Next*, the information will be stored in a file called userinfo.php which contains the information you just entered. This file is editable if the information entered should differ later. In this way the user is not asked for this information every time the objects need it.

**Figure 24: Showing the handling of the information input**

Later the user is automatically redirected to the last step of the installation where the creation of the database and table for the object are done through the information entered by the user. If the installation file gets access to the database, you would get a successful message. If it does not, you will have to go back and fill in the information again.

**Figure 25: Showing a successful installation**

After a successful installation, you are automatically taken to the front page of the actual object, i.e. index.html. From here you can start using the object installed and customizing it for your OC.
4.3  The Poll Object

Because this object both affects the administrator side of an OC as well as the user of the OC, the object will be presented separately in order to distinguish those two views.

Poll object: Administrator’s view

In order for the members of the OC to have a poll, an administrator of the OC has to create one. To be able to make a poll, the administrator(s) of the OC should first come up with an issue that is relevant to the OC as well as the information they gather from the members, should be valuable. Depending on the purpose of the OC and their goals, the polls could vary a lot, e.g. let say the staff of the OC are planning a party for their members. To be able to gather most people on the party the staff should come up with a couple of alternative days. Then the members of the OC that are attending to this party vote on the day that is most suitable for them. In this way the OC staff can plan a party on the day most participants are available, as well as the members have influenced a decision in their community through participating and democracy.

Local menu

It is easy for inexperienced users to get lost in a new system, especially if it is too much information at one time. In real life we have signs to show us the directions to locate ourselves in unfamiliar places when we need them. However in the virtual world we have a local menu which works as our signboards telling us where we are, as well as they give us the opportunity to move back and forward. The local menu in this case is placed in the middle at the very top where it is visible all the time, helping the administrator to Create Question, Show Options or locates him if he gets lost. Although it may not be necessary when this object only have two directions from administrators view, the object could be expanded and more directions could be easily added, which would make the objects more complex and not as lucid as it is now.

Create Question,  Show Options

Figure 26: The local menu in the vote object
Create question

By letting all members of an OC vote, the administration would have a good basis to bring a conclusion regarding one actual issue. That is why the question as well as poll is limited to one per day. In addition, this gives every member the chance to vote within a day and it also makes it lucid for the administration to later find a specific poll to look or compare the results.

The administrator chooses himself a date which he wants to register the actual poll, by typing it in the date field on the top in the right format, YYYY-MM-DD, e.g. 2008-05-01. Later he fills in the question he wants the members of the OC to vote on and at least two alternatives and maximum six, giving the members different choices for answering the question. The date field, one question and two vote alternatives are mandatory to register the poll into the database and save it. By clicking the Add button, the poll will be registered.

Figure 27: The form for registering a poll
Show options

This feature, which is located to the right on the local menu at the top, brings out a listing of all the polls registered in the database and options available. Here the administrator is able to see questions and alternatives he has previously registered, given in chronological order. If a poll is very old, or he simply has registered wrong date or mistyped a question, he can easily remove the question from the database by clicking *Delete* button\(^9\).

![Figure 28: The options available for each poll registered](image)

Besides removing registered polls, the administrator(s) is also able to view the statistics of the polls registered by clicking the button *Show statistics*. Here he is taken to a new page (see figure 29 on next page) where the statistics of the actual poll are shown in form of percentage of the different alternatives and the total votes of the members. With this information, the administration of an OC can collect information that may be valuable for the OC as well as for the members, in the future, past or presence.

\(^9\) Note that all data about the poll will be deleted, including the statistics.
Figure 29: Illustrating the statistics of a certain poll

**Vote: user view**

The user’s approach regarding polls is relatively different than administrator’s that make polls. As we can see in the example below (PollObject/vote.php), the user chooses an alternative he feels is the most correct for him in order to give his opinion, influence the community he is a member of and simply become an active member in his community. The member of an OC simply chooses the alternative he wants to vote for and presses the *Vote* button, which is then sent to the database for registration.

---

**Online community Building**

**What, Why, How ?**

This is a test of an online community to show how objects can be implemented on any web site. It is a result of work done by Selver Tosić, with Dino Karabeg as his mentor, to give a contribution to people wanting to build an online community. This is just one of the object oriented objects done with respect to this project.

Figure 30: Illustrating a poll for the members of an OC
When a user clicks the Vote button, the vote is registered successfully, i.e. if he has not voted before. Then a green message confirming the successful registration of the vote is shown.

**Online community Building**

**What, Why, How?**

This is a test of an online community to show how objects can be implemented on any web site. It is a result of work done by Selver Tosić, with Dino Karabeg as his mentor, to give a contribution to people wanting to build an online community. This is just one of the object-oriented objects done with respect to this project.

Figure 31: Showing the message of a successful vote registered
4.4 The Registration Object

After a successful installation, see chapter 4.2, you are taken to the index page of this object. Here the users of an OC have the opportunity to sign up to the OC or subscribe, i.e. pay membership. But, first you need to register some information. This can be done by clicking the *Sign up* button.

![Image](image.png)

*Figure 32: Showing the index page of the register object.*

In order to register every user that wants to join an OC, he has to give up his personal information as well as information regarding the OC, i.e. username, password, newsletter, etc. All the fields in the form are required, as well as there are some prevailing conditions, which can also be related to the real World. The real name of an individual do not contain anything else than letters, as well as the age do not contain anything else than numbers. In addition to this, the phone number must not carry anything else than digits and the email address has to be written in correct order. It is not possible to sign up for an account unless all the fields are correct.
After a successful registration, the user is redirected to the index-page, which will give him the opportunity to pay a fee for using the OC’s services. Although this is just a sample of a possible solution, one can make an account and accept payment through different methods from the OC members. However, in this example you can click on the *Subscribe* button marked with a red circle where you are redirected to Paypal’s secure site for payment\(^\text{10}\) for joining the OC.

\(^{10}\)This is just a sample with my billing information and could be altered in order to be used by anyone or excluded if wanted.
Figure 35: Example of a typical payment form from PayPal.

When a payment is successful, the administrator of the OC receives an email, on registered mail address, that verifies the transaction with the user’s personal information.
4.5 The Login Object

When the installation setup of the login object is completed, the index page is shown. It contains a login form where users of the OC can log in and enter the community. By typing the right username and password combination and pressing the Log in button, the user is granted access to the OC and the functions available for his subscription.

![Login Object Visual Representation](image)

**Figure 36: Showing the index page for login object**

The user is taken to pages which only he and other members are allowed to see, where the information is merely for the members. In this way the users are not disturbed by surfers, spam users or non-members, keeping him in a safe place among other authorized users. This page is just a sample showing that the login function and security are working properly, by letting only the people that need to see this information enter this page. Later one can add more information and link several pages together.

![Sample Page for Members](image)

**Figure 37: Illustrating the pages only available for members of the OC, after passing the security check**

When the user wants to leave the OC, he simply presses the Log out link, which will then destroy the information stored\(^{11}\) and redirect him to the login page again.

---

\(^{11}\) This information is cryptic (md5) stored in sessions that are removed automatically when logging out so that no misusing the information is very hard.
5 Objects - implementation

In this chapter I present and discuss the details of the objects’ solution. The details of the objects and the database will be presented together with the database design. Going from the tables of the poll object to the register and login object, a presentation of the interaction and message sending inside the objects are modeled and explained. In the end of this chapter advantages and disadvantages of the developed objects are discussed with respect to today’s Internet and technology towards the objects provided.

5.1 The database design

5.1.1 The Poll Object

All the information gathered from both the user of an OC and the administrator(s) has to be stored somewhere. In order to somehow keep the different information in a secure place, the information is stored in different tables in the same database called signup, which is in common for all the objects.

When an administrator creates a poll; the date, question and alternatives are stored in a table called community_poll. By having this information stored in a database, the administrator(s) can make polls in advance as far as he wants and also look at previous polls.

<table>
<thead>
<tr>
<th>Name of field</th>
<th>Type</th>
<th>Key(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
<td>Date</td>
<td>Primary key</td>
</tr>
<tr>
<td>Question</td>
<td>Varchar(200)</td>
<td></td>
</tr>
<tr>
<td>Alternative 1</td>
<td>Varchar(50)</td>
<td></td>
</tr>
<tr>
<td>Alternative 2</td>
<td>Varchar(50)</td>
<td></td>
</tr>
<tr>
<td>Alternative 3</td>
<td>Varchar(50)</td>
<td></td>
</tr>
<tr>
<td>Alternative 4</td>
<td>Varchar(50)</td>
<td></td>
</tr>
<tr>
<td>Alternative 5</td>
<td>Varchar(50)</td>
<td></td>
</tr>
<tr>
<td>Alternative 6</td>
<td>Varchar(50)</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Showing the community_poll table and its distribution
The left side of this table shows the name of the fields in the table community_poll, while next to the name(s) is the description of the fields. While Varchar(50) is a text field which can save a string up to 50 characters, the type Date emphasizes that this field is a date and can only store information in date format. The right column shows the key in a table, which can vary from none and above. A primary key indicates that this field is unique and in this table there can only be one instant of one primary key, i.e. you cannot save information in this table if an instant with the same date already exists. In our case it is not possible to register two polls on one day.

![Diagram](image)

**Figure 38:** The interaction between the database and administrator and the interaction between the database and the OC user.

This drawing explains how the interaction between the computers and the database server is taking place. When an administrator makes a poll, the information is stored in the database signup into the fields listed in table community_poll, shown above. This information stored is also used when an OC user is voting, in order to show him the poll that the administrator has registered.
When the user has voted, the alternative chosen is saved in a table called `community_poll_statistics`, which is responsible for keeping the number of votes.

<table>
<thead>
<tr>
<th>Name of field</th>
<th>Type</th>
<th>Key(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
<td>Date</td>
<td>Primary key</td>
</tr>
<tr>
<td>Question</td>
<td>Varchar(200)</td>
<td></td>
</tr>
<tr>
<td>Alternative 1</td>
<td>int(11)</td>
<td></td>
</tr>
<tr>
<td>Alternative 2</td>
<td>int(11)</td>
<td></td>
</tr>
<tr>
<td>Alternative 3</td>
<td>int(11)</td>
<td></td>
</tr>
<tr>
<td>Alternative 4</td>
<td>int(11)</td>
<td></td>
</tr>
<tr>
<td>Alternative 5</td>
<td>int(11)</td>
<td></td>
</tr>
<tr>
<td>Alternative 6</td>
<td>int(11)</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>int(11)</td>
<td></td>
</tr>
</tbody>
</table>

*Table 3: Showing the distribution in the community_poll_statistics table and its fields*

This table is similar to the previous table, but in addition it has the total field at the bottom. The main difference between these two tables is the type of fields, where the fields storing the alternatives and the total of votes are now of the type `int`. This means that the information in these fields is numeric and not textual. At the maximum, this numeric information can be up to 11 digits long, which is enough when considering votes.
After the vote has been saved, the table is automatically updated and it keeps track of the different votes as well as the total votes in the poll. By adding the votes in the corresponding fields as well at the total votes, the calculation of the polls are easier to determine when it comes to statistics. If the administrator(s) wants to view the statistics of a specific poll, the numbers stored are used and a calculation is done in order to show the administrator the result in form of percentage.

5.1.2 The Registration Object

The information of the different users is registered in a table in the database, as well as everything else regarding these objects. When a user successfully registers, the information is stored in a table called `community_users`. This table saves the information in different fields, which makes it lucid for the administrator to review the information of the OC user(s).
Table 4 below: Showing the distribution in the table community_users

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Key(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Username</td>
<td>Varchar(30)</td>
<td>Primary key</td>
</tr>
<tr>
<td>Name</td>
<td>Varchar(30)</td>
<td></td>
</tr>
<tr>
<td>Address</td>
<td>Varchar(50)</td>
<td></td>
</tr>
<tr>
<td>Age</td>
<td>Int(3)</td>
<td></td>
</tr>
<tr>
<td>Email</td>
<td>Varchar(50)</td>
<td></td>
</tr>
<tr>
<td>Phone</td>
<td>Bigint(20)</td>
<td></td>
</tr>
<tr>
<td>City</td>
<td>Varchar(20)</td>
<td></td>
</tr>
<tr>
<td>Country</td>
<td>Varchar(25)</td>
<td></td>
</tr>
<tr>
<td>Registered</td>
<td>date</td>
<td></td>
</tr>
<tr>
<td>Newsletter</td>
<td>Varchar(3)</td>
<td></td>
</tr>
<tr>
<td>Password</td>
<td>Varchar(100)</td>
<td></td>
</tr>
<tr>
<td>Payed</td>
<td>Varchar(5)</td>
<td></td>
</tr>
</tbody>
</table>

The fields the user enters in the registration form are all stored in the table community_users in addition to the date registered. The Payed field keeps track of the pay status of every single user in the OC. This value is set to “no” as default because the users cannot pay before they register, and the value can be modified later by the administrator. The primary key here is the username, which is obvious because the OC cannot have several instances of the same identity within one OC. However, if we take a look at the password type, we can see that it has been reserved for 100 characters. The reason for this is because the password in this field is saved with cryptic algorithm, which is a security measure, and normally contains significantly more characters than ordinary strings.

The communication between the computer and the database is established when a user is registering. Later, the administrator(s) can browse through the stored information as much as they want, or simply add functions to their purpose, e.g. to remove old users from the database. In order to verify the user, the information registered in this table is also used in the login object in this thesis. This is described in next subchapter.

![Figure 40: Illustrating the storing of user input in database](image-url)
5.1.3 The Login Object

The security measures and procedures that are checking whether a user is allowed to enter a specific page is taking place in the specific code of the object Login. This object uses information already registered in the database ¹² by the user himself, see RegistrationObject.

The Login object then compares if the typed username and password, matches the combination of the username and password registered in the table community_users in the database. If it is a match, the user is granted access. If not, then he gets an error message.

![Diagram of Login Object interaction](image)

**Figure 41:** Illustrating the interaction between the computer and the database regarding the login object.

In this illustration the user is typing his username and password, which then is sent to the server for comparing it with the information stored in the database. The user then receives a permit access to the OC or is denied access.

---

¹² The LoginObject uses the database when registering users, i.e. community_users, described above.
5.2 Objects’ construction

Like mentioned through the whole thesis, there are three objects developed as open source software; Poll, Register and Login. In common they have the following features:

- Install script
- SQL Queries
- PHP5 functions
- HTML functions

These technological approaches, which are melted together, are all part of the functionality of the objects and will be described below. However all of the objects are armed with some kind of security and validation features which are also worth to show;

- Error handling
- Validation
- Md5 encryption (only Login object)
- Session handling (only Login object)

**Install script** is used to set up and create a file used to connect to the database. This file is located in the main folder of the object, i.e. PollObject/userinfo.php, RegisterObject/userinfo.php and LoginObject/userinfo.php. Every time a connection or signal is sent to the database, this file is opened and the information saved within this file, is used. To make an example of the information saved in that file; if I at the installation of the object wrote mysql.ifi.uio.no as my MySQL server address, root as my username and 123456 as my password, the userinfo.php file would look like this:
By having this information in a text field, the objects are allowed access to the database at any time, letting them read and write information when needed.

**MySQL queries** are used to get information from the database, as well as they are used to write or update the database. MySQL queries have their own syntax, e.g. when we want to receive the address of the user James in our OC, a MySQL query towards the table community_user, would look like this:

```
SELECT address FROM community_users WHERE username="James";
```

**PHP5 functions** can have several purposes, depending on the functionality one want to achieve. Functions in this case are instructed to meet the requirements of the purposes the developed objects have, i.e. create a poll, register a vote, validate input fields, check username and passwords, etc. However, sequence diagrams showing the functionality and interactions of the main functions are shown later in this paper. In addition to this, PHP5 built in functions like, mysql_connect() and md5() are also used to connect to the database and encrypt passwords, which will also be described further down.

**HTML functions** are used through two types in these objects. The one is the html_header() which is used to forward a user to another page, often the start page of the object. And the second is html_select() which is used to offer a drop down menu to user with various alternatives he can choose.
**Error handling** takes action whenever an error occurs, this could be errors caused by user input, connection error, database error, etc. However, whenever this is the case, the user is somehow notified. Within every object there is a function handling errors and notifying the user. E.g. if a user enters a site he is not allowed to see the function `security_check()` found in the LoginObject.php (see Appendix) is activated. This function is controlling the user, granting him access if he is a user of the community. If a user does not have access to this site, the user would just get redirected to the login form.

**Validation** is activated every time the user is entering information in the input fields. If validation requirements are not fulfilled, an error occurs and user is notified, e.g. a validation of the user’s name when a user wants to register. The function `validateName($name)` in the class.User.php is then called, see Appendix. This function only allows letters a-z in both lower and upper case to be entered. If this requirement is not fulfilled the user will not be registered to the OC.

**Md5 encryption** converts the password into cryptic messages, especially used when transferring passwords to different entities. This is done in the LoginObject with the function `login($username, $password)` with the purpose of making the password invisible for unauthorized users. E.g. a password 123456 could look something like e10adc3949ba59abb.

**Session handling** is used when a user is logging into the OC with the right username and password. The cryptic password is then saved into sessions which are used for storing information. These sessions are holding the cryptic password until the user decides to log out and leave the OC. The sessions are then destroyed together with the saved information.
In order to show how the interaction is travelling inside the objects and the signals sent within the objects, as well as to the database and the administrator / user of the OC, the use of sequence diagrams in UML 2.0 is most applicable. The description and illustration will consider one object at the time, going into the respective lifelines to the lowest level, i.e. decomposing\(^{13}\). From these sequence diagrams it is easy to see, with the additional text, what kind of tasks the main functions within the provided object have. The actors in a sequence diagram are represented by a lifeline, sending signals with information as a form of action and trigger.

**Poll: Administrator**

![Sequence Diagram](image)

*Figure 43: Sequence diagram showing the sequences when creating a poll*

The administrator creates a poll by typing the date, one question and alternatives. The PollObject is then contacted with this information attached. In this model, the outcome has three alternatives distinguished with separated lines. Either the PollObject returns a Poll Error because of too few alternatives or a Date Error due to wrong date entered or duplicate date registered. In a successful creation of a poll, the PollObject registers the poll in the database and returns a confirmation to the administrator.

---

\(^{13}\) Decomposing a lifeline in a sequence diagram is to go into the actual lifeline and show with sequence diagram(s) what signals are going inside of it. This can only be shown if you have detailed information of the sequence diagrams and actual lifelines (actors).
This is a decomposing sequence diagram of the lifeline PollObject from previous sequence diagram. The function setQuestion($date, ….) receives a signal from Admin that wants to create a poll. This function then sends it to the function validate($date) with the date as the attachment to identify the poll, which validates the poll and its alternatives. If the poll is not valid, the function validate($date) sends an error back which is forwarded to the administrator by the function setQuestion($date, ….). In the other alternative, if the poll is valid, the function setQuestion($date, ….) validates the entered date by sending it to a function called doubleDate($date), which searches in database for a poll already registered on this date. A poll is already registered on this date and a date error is shown to the administrator. In the successful alternative there is no poll registered in the database on this date and the function setQuestion($date, ….) registers the poll in the database and gives a confirmation to the administrator.

Figure 44: Sequence diagram showing the decomposition of the lifeline PollObject
Vote: User

Figure 45: Sequence diagram showing the sequences when registering a user’s vote

This process requires that a poll is registered on the day the user is voting. The assert frame in this sequence diagram demands that these steps in the processes are the actual processes, i.e. that the condition is conformed and there are no negative signals. First the user chooses an alternative available, which is sent to the PollObject. This again, resends the vote in the database and notifies the user that the vote is registered. The negative outcomes and messages in this case are not relevant to outline, because there cannot be any with respect to the PollObject developed in this thesis.

Decomposition: userdePoll

Figure 46: Sequence diagram showing the decomposition of the lifeline PollObject

When a vote needs to be registered, the function registerAnswer($date, $alternative) is receiving a signal with the date of the poll and the alternative chosen as attachments. This
function then contacts the function validate($date) which returns a result whether the date of
the poll is valid or not. Then the function registerAnswer($date, $alternative) performs a
check on the alternative to see if it is valid and registers the vote in the database. At the end,
this function sends a notification to the user telling that the vote is registered.

Register

Figure 47: Sequence diagram showing the sequences when registering a member in an OC

When a user wants to become a member of an OC, he has to fill in information in the fields
required. The RegisterObject then receives a signal with the entered information as
attachment, and a validation of the fields are done. Later the username and email address is
sent to the database checking for existing fields. The database then returns the result to the
RegisterObject, which could either be that this username or email address already is registered
or that everything is ok. The RegisterObject then translates this message which can have three
outcomes and forwards it to the user.
Decomposition: deRegister

![Sequence Diagram](image)

Figure 48: Sequence diagram showing the decomposition of the lifeline RegisterObject.

When the RegisterObject is contacted, the signal is sent to a function called register($name, $age, ….). This function sends the information as an attachment to function checkFields($name, $age, ….), which validates the fields, and this is also the function that sends the username and email address to the database to make a check for existing registrations. It is also responsible for forwarding the result received by the database to function register($name, $age, ….), which could either be an error or a confirmation of a successful registration. Then the function register($name, $age, ….) resends this message to the user which is either saying that some fields are not correctly filled in, that the username or email address is already registered or in last case the registration was successful.
Login

Figure 49: Sequence diagram showing the sequences when a user is logging into an OC

This diagram describes the interactions and signals when a user is logging into the OC system using the LoginObject provided with respect to this thesis, by typing his username and password into the login form. Then the LoginObject checks that the entered information is right according to the information stored in the database. If it is right, the user is granted pass to the OC’s facilities and receives a notification of this. In a negative case, the username and password combination of the typed information does not match the information stored in the database, and the he gets a Login error message.

Decomposition: deLogin

Figure 50: Sequence diagram showing the decomposition of the lifeline LoginObject
The username and password entered by the user is received by the function called login($username, $password). This function encrypts the password in md5 for security reasons when transferring passwords. After this converting, the function sends the username and password to the database to check if the combination is right. If the username and password combination match the ones stored in the database, the function login($username, $password) receives a confirmation and saves the cryptic information into cookies. Cookies are used to identify the user during his stay in the OC. In the last step, the user receives a confirmation telling that he is logged in. The other alternative in this case, is if the function login($username, $password) receives a mismatch from the database, regarding the typed username and password, and the ones found in database. Then the function login($username, $password) receives a “pass denied”- message, and sends a login error message to the user.
5.4 Advantages and disadvantages of the objects and solution

In the start phase of the objects’ development I decided to make the objects platform-independent, for several reasons, which is why I chose PHP5 computer scripting language. Making objects platform-independent gives the software provided the ability to run in any environment, i.e. Windows, Linux, UNIX, etc. Another advantage to the platform independence is the long living technology. When using platform-independent web-based software, the worries of the “next big thing” are not there. Regardless what kind of technology is coming, the users of your OC will always have access to it through the Internet. This is a huge advantage in today’s Internet, overloaded by new technologies and better solutions born every day. Both the objects and requirements for the developed objects are also freely available, which makes them even more attractive in today’s collection of highly charged software. In addition, by using open source software the advantages for the users are many. The main advantage is the reduced dependence on software vendors, which avoids vendor lock-in and gives people the freedom to choose what to do with the code and when to do it. The open source feature implicitly makes the source code available for others that may need it, this also opens for modification of the objects to any purpose and desire. Not needing to program something from scratch but adding and/or deleting something from the code, is a time saving and energy saving process. A positive feature regarding the object orientation is the structured code, which is easy to maintain, modify and read. Expansion when it comes to object oriented programming technique is also a feature that is easy to implement, because of the structured way of programming and clear program code.

Although the advantages regarding these objects are many, there are also some disadvantages. Regarding the platform-independent software’s security, it is far more threatened because of the availability from anywhere and anytime. People with any kind of computer and software platform can try to access or even hack into the platform-independent software. In combination with open source software, people can easily get their hand on the source code. By letting people with bad intensions get the source code, which cannot be easily controlled, they may exploit security gaps and break into the site or in worse case into the database where information is saved. Another disadvantage with the platform-independent software, especially the objects provided, is the browser interface. Because the outlook is not perfect and different browsers translate source code differently, the outlook of the objects may be slightly different some places. Open source objects are also available to ones competitors,
which may use these objects to get a competitive advantage. Modifying the objects and making theirs OC more attractive and appealing, may lead the OC users to leave a community in advance for another. Because open source objects are often made and offered gratis and no requirement is determined for the software, the documentation, updates and support are often not well considered. This could be a big minus if the software is not well developed, security gaps detected or difficult to use.
6 Examples of implementation

To show the objects in practice, the objects have been implemented on an online community. This chapter describes the site which is used as an example of the objects’ functionality and how it is serving this particular online community.

6.1 Nacuheal (http://www.nacuheal.org)

The provided objects have been implemented and tested on the Nature Culture Health International community portal, http://nacuheal.org. The results of deploying the provided objects with cut and paste method are shown below.

Nature Culture Health International is an international NGO originating in Norway whose purpose is to spread salutogenic (health bringing) ideas and activities, not only locally but regionally, nationally and internationally. By introducing salutogenic ideas and activities, they believe that public health, safety, environments and peace can be promoted.

The NaCuHeal association is compounded by members and cooperators which are interested in health promoting nature and culture activities. They are working through inspiring open conversations. Their goal is to help building a life-giving society based on human happiness and growth. Focusing and introducing ideas and activities that can have positive effect on the human being, they claim a positive impact can be achieved on health, safety and life environment.

I illustrate the functioning of my objects on the example of registration and poll. In addition to this, I have modified and reused the source code of the RegistrationObject to make another object which is valuable for NaCuHeal, conference registration. By customizing and reusing this code, it is shown that any object with any purpose can be made with the same paradigm.
6.2 Implementation of objects on NaCuHeal.org portal

The objects provided with this thesis have been implemented on his portal in order to show their functionality, as well as to show that they are made for serving the facilities of an online community. The method used to implement these objects is “cut and paste” with small modifications to fit the design of NaCuHeal’s web site.

Nature Culture Health International

Online community building tools (under development)

Welcome to the online community tools test page. Please try out the set of tools, developed to serve the online community facilities.

Register

In order to be a member of the online community and interact with other members, you have to register.

Log in - User

If you have registered to our online community, you are more than welcome to log in.

Log in - Administrator

For administrative privileges and rights, a root account is needed. If you have one, please click here to perform administrative tasks.

©2008 Nature Culture Health International

Figure 51: Showing the choices of objects available on NaCuHeal.org

This page is the start page of the objects provided with this thesis. Here the user can choose to either register to the community or log in, if he is already registered as a member. For the administrators it is also possible to log in through the LoginObject provided.

6.2.1 Register

In order to use the facilities the online community is offering, registration is demanded. It is therefore most applicable to show the registration form implemented on this web site.
After a successful registration, the user is redirected to the log in page. Here he is able to enter the online community with the username and password entered in the registration form.

### 6.2.2 User - Log in

![User Log in Form](image)

**Figure 53: Showing the log in form for users on NaCuHeal web site**
If the username and password combination is right, the user is allowed to enter the area only available for members of the community. This would give the user a number of activities to do through the tools provided by the OC. In this case the start page is showing links to the activities that could be found in an online community, i.e. profil, chat, blog, gallery etc.

6.2.3 Poll – user view

In this page a poll is shown on the right side, which is the poll object provided with this thesis. This poll is shown if an administrator has registered a poll on today's date. The user is then able to participate in the poll or attend to other activities shown on this page, i.e. sign up to a conference (described below), use the web shop etc.
6.2.4 Administrator – Log in

The log in form for the administrator is the exact same as the log in form provided for users. The difference is that only administrators are able to log in through this form and not the users.

![Image of Administrator Log in Form]

Figure 55: Showing the log in form for administrators on NaCuHeal web site

6.2.5 Poll – administrator view

![Image of Poll Form]

Figure 56: Showing the start page of the administrator, after logging in
After a successful username and password combination, the administrator is able to do administrative tasks, i.e. create polls, view options or make a conference form.

6.2.6 Customized object – Make conference form

Like mentioned above, the objects provided are customizeable and may be modified to serve other purposes. I have customized the RegisterObject to make a object called *Make Conference Form*. This tool gives administrator the opportunity to make a conference form used to sign up members to a certain conference. Because NaCuHeal often offer conference and get-togethers to their members, this kind of tool was needed.

![Nature Culture Health International](image)

*Figure 57: Showing the Make Conference Form*

Here the administrator is able to define the name of the conference, together with fields he wants the members to fill in. Later, a database is created with the information entered by the administrator and the users are then able to sign up for the conference. In this way the administrators do not need to make a new page every time a new conference is offered, they just fill in this form provided.
7 Conclusion

As we have seen in the introduction and chapter 2.1, the traditional communities and activities are declining and substituted by new trends and technology. In parallel with this decline, we have seen that the numbers of Internet users have increased dramatically through the past sixteen years. Chapter 2.2 also showed us an increase in online community building, which can tell us that they are becoming more and more popular. In addition, this chapter showed us that this way of interacting has brought out new habits and trends, as well as new forms of misbehaviour among humans. Nevertheless, online community building is necessary to fulfill the social needs described by Maslow, (chapter 2.2). As I mentioned in chapter 2.2, people attend to online communities with different goals and expectations. In order to fill these expectations and needs, more online communities are needed so that people may use online communities to interact with each other and re-strengthen the social capital that has diminished through the past three decades, as Putnam highlighted. This again will lead to avoid the health and poverty problems caused by isolation, loneliness and social flaws, emphasized in the beginning of this thesis. The problem, however, is that online community builders are not necessarily programmers but often domain experts. One purpose of this thesis was to make their job easier, by providing customizable tools for online community building.

I have endeavored to do that by using the object orientation approach in web programming. This has been applied to certain objects, primarily developed for facilitating an online community. These objects can be modified and customized to develop any object facilitating an online community. Although the features class, method and object were primarily used to develop these objects for online community building, they are also open for other features highlighted in chapter 2.3, i.e. inheritance, encapsulation, abstraction and polymorphism. I have implemented this approach into three objects, which has made it easier to reuse, modify and customize the source code to any purpose. The three objects are “Register”, “Login” and “Poll”, and have been analyzed through a domain model in chapter 3. The objects provided are also made with an installation script, and does not require code interference at all. This makes it possible for non-programmers to use these objects, without looking into the programming code, that saves them both time and energy. By supporting object orientation and platform independence, these objects may also easily be used as starting points or foundation when developing any other object in the online community building process. The
structured code has made it possible to easily expand the source code and reuse it instead of rewriting it, as emphasized in chapter 2.3. Because of this, these objects can easily be “cut and pasted” into any web site. Again, by using the “cut and paste” method, the online community builders do not need to have any programming skills and still build online communities.

These objects are now freely available to anyone from the collection site of online interaction software [http://thinkofit.com/webconf/forumsoft.htm#tosic](http://thinkofit.com/webconf/forumsoft.htm#tosic).

This paper also show the objects implemented in practice, not just by giving a step by step documentation and detailed description as found in chapters 4 and chapter 5, but also adding them and showing their functionality in a real online community (chapter 6). By having these objects facilitating NaCuHeal association’s web sites, I conclude that these objects are in fact made with the right purpose: facilitating online communities and contributing to build social capital through new trends. In addition it is possible to customize object oriented objects for other purposes. Because of the object orientation approach, they can be modified, extended and used as a foundation for developing other objects facilitating an OC. I also conclude that these objects are easy to use and do not require any programming skills for the user to benefit from them. Because the object orientation technique has made it possible to use these objects with no programming knowledge, an additional conclusion is that it is easier to build an online community through these objects and the object oriented approach than through other, more traditional,

### 7.1 Further development

I believe that object orientation will prove to be fertile in web programming, particularly in online community building. Further development of this approach is, however, left for future research. I have provided three objects that may be used to facilitate an online community, and they may also be used for developing other objects facilitating an online community. These objects are only a small number of objects needed to create an entire online community. Nevertheless, these objects have shown the direction for others who may use them as models when using the same effective approach in online community building. The question of a complete set of customizable objects, one that may be used to implement all relevant processes that occur in online community building, is also left for future research.
8 Bibliography


Appendix

9.1 Source code – Poll object

```php
<?php

class VoteObjects
{
    public function setQuestion($date, $question, $alternative1, $alternative2, $alternative3, $alternative4, $alternative5, $alternative6)
    {
        include('config.php');

        $statistic3 = 0;
        $statistic4 = 0;
        $statistic5 = 0;
        $statistic6 = 0;

        if( $alternative3 == "" )
        {
            $alternative3 = -100;
            $statistic3 = -100;
        }
        if( $alternative4 == "" )
        {
            $alternative4 = -100;
            $statistic4 = -100;
        }
        if( $alternative5 == "" )
        {
            $alternative5 = -100;
            $statistic5 = -100;
        }
        if( $alternative6 == "" )
        {
            $alternative6 = -100;
            $statistic6 = -100;
        }
        if( $date != "" && $question != "" && $alternative1 != "" && $alternative2 != "" )
        {
            if( !$this->validate($date) )
            {
                //Validates the date
                echo "<div id=errorstring>$invalidDate</div>
            }
        }
    }
}
```
elseif( !$this->doubleDate($date) )
{
    //Checks for already registered date
    echo "<div id=errorstring>A question on date $date already exists</div>";
}
else
{
    $query = "INSERT INTO $vote_table (date, question, alternative1, alternative2, alternative3,
alternative4, alternative5, alternative6) VALUES ('$date', '$question', '$alternative1', '$alternative2', '$alternative3', '$alternative4',
'alternative5', 'alternative6')";
    $result = mysql_query($query);

    $query2 = "INSERT INTO $vote_statistic (date, question, alternative1, alternative2, alternative3,
alternative4, alternative5, alternative6, total) VALUES ('$date', '$question', '0', '0', '$statistic3', '$statistic4', '$statistic5',
'$statistic6', '0')";
    $result2 = mysql_query($query2);

    echo "<div id=goodstring> Question added </div>";
    return true;
}
else
{
    echo "<div id=errorstring>$blankField</div>";
}
}

public function showQuestion( $date )
{
    //The date must be in the form YYYY-MM-DD
    include('config.php');

    $query = "SELECT question FROM $vote_table WHERE date = '$date'";
    $result = mysql_query($query);

    if( $date != "" )
    {
        if( !$this->validate($date) )
        {
            //Validates the date
            echo "<div id=errorstring>$invalidDate</div>";
        }
        else
        {
            if( $row = mysql_fetch_array($result) )
            {
                echo htmlspecialchars($row['question'],ENT_QUOTES);
            }
        }
    }
else
{
    return -1;
}
}
}
else
{
    echo "<div id=errorstring> You must enter a date </div>";
}

public function showAlternatives( $date )
{
    include('config.php');

    if( !$this->validate($date) )
    {
        //Validates the date
        echo "<div id=errorstring>$invalidDate</div>";
    }
    else
    {
        for( $i = 1; $i < 7; $i++ ) //7 Because we have 6 alternatives
        {
            $query = "SELECT * FROM $vote_table WHERE date = '$date'";
            $result = mysql_query($query);
            $row = mysql_fetch_array($result);

            if( $this->checkAlternative($date, "alternative$i") )
            {
                echo ' <INPUT TYPE="RADIO" NAME="alternative" VALUE="alternative'.$i.'"> '.htmlspecialchars($row["alternative$i"],ENT_QUOTES);
                echo "<BR>";
            }
        }
    }
}

public function showAllQuestions()
{
    include('config.php');

    $query = "SELECT * FROM $vote_table ORDER BY date DESC"; //Chronological sorting by date
    $result = mysql_query($query);
    $rownumbers = mysql_num_rows($result);
if ($rownumbers > 0 )
{
	//echo "<div id=allquestions>";
	while( $row = mysql_fetch_array($result) )
	{
		//echo '<BR>";
		echo htmlspecialchars($row["date"],ENT_QUOTES) ."<BR>";
		$date = $row["date"];
		echo "Question: <strong>". htmlspecialchars($row["question"],ENT_QUOTES) ."</strong> <BR>";
		echo "Alternative 1: " . htmlspecialchars($row["alternative1"],ENT_QUOTES) ."<BR>";
		echo "Alternative 2: " . htmlspecialchars($row["alternative2"],ENT_QUOTES) ."<BR>";
		if ($row["alternative3"] != -100 )
		{
			echo "Alternative 3: " . htmlspecialchars($row["alternative3"],ENT_QUOTES) ."<BR>";
		}
		if ($row["alternative4"] != -100 )
		{
			echo "Alternative 4: " . htmlspecialchars($row["alternative4"],ENT_QUOTES) ."<BR>";
		}
		if ($row["alternative5"] != -100 )
		{
			echo "Alternative 5: " . htmlspecialchars($row["alternative5"],ENT_QUOTES) ."<BR>";
		}
		if ($row["alternative6"] != -100 )
		{
			echo "Alternative 6: " . htmlspecialchars($row["alternative6"],ENT_QUOTES) ."<BR>";
		}
		echo "<div id=buttons> <div id=buttonLeft> <form name='form2' method='post'
action='options.php?action=stat'>
<input type='submit' value='Show statistics'>
<input type='hidden' name='date_stat' value='$date'>
</form> </div>";
		echo "<div id=buttonRight> <form name='form3' method='post'
action='options.php?action=delete'>
<input type='submit' value='Delete'>
<input type='hidden' name='date_stat' value='$date'>
</form> </div> <div id=empty> </div> </div>";
		//echo "</div>";
	}
	else
	{
		echo "<div id=errorstring>There are no question registered</div>";
	}
	}

//Validates the date
if (!this->validate($date) )
{
	//Validates the date
	echo "<div id=errorstring>$invalidDate</div>";

public function registerAnswer( $date, $alternative)
{
	include('config.php');
	$remember = @$COOKIE['remember'];
	if (!this->validate($date) )
	{
		//Validates the date
		echo "<div id=errorstring>$invalidDate</div>";
	}
public function checkAlternative($date, $alternative)
{
    include('config.php');

    
    elseif ($alternative == "")
    |
    |
    return -1;
    
    elseif ($remember == "voted")
    |
    |
    return -2;
    
    else
    |

    //Checks what the answer is, it has to match the alternative
    
    for( $i = 1; $i < 7; $i++ )
    |

    if( $alternative == "alternative" . $i )
    |

    $query = "UPDATE $vote_statistic SET alternative$i = alternative$i +1 WHERE
    |
    |
date = "$date"; |

    $result = mysql_query($query);
    $query2 = "UPDATE $vote_statistic SET total = total +1 WHERE date = "$date"; |

    $result2 = mysql_query($query2);

    session_start();

    // remove all the data from the session (auto logoff)
session_unset();

    // remove the session itself
    session_destroy();

    // put the password in the session
    @ session_register("me");
    $_SESSION["me"] = "voted";

    //set a password in the cookie, the name is unrecognizable
    setcookie("remember ", "voted", (time())+640000);
    return 1;

    
    
}
if ( !$this->validate($date) )
{
    //Validates the date
    echo "<div id=errorstring>$invalidDate</div>";
}
else
{
    $query = "SELECT $alternative FROM $vote_table WHERE date='$date';";
    $result = mysql_query($query);
    if( $row = mysql_fetch_array($result) )
    {
        {
            return true;
        }
        return false;
    }
    return false;
}

public function calculate( $date )
{
    include('config.php');
    if ( !$this->validate($date) )
    {
        //Validates the date
        echo "<div id=errorstring>$invalidDate</div>";
    }
    else
    {
        $query = "SELECT * FROM $vote_statistic WHERE date = '$date';";
        $result = mysql_query($query);
        $row = mysql_fetch_array($result);
        $total = htmlspecialchars($row['total'],ENT_QUOTES);
        for( $i = 1; $i < 7; $i++ )
        {
            $query = "SELECT * FROM $vote_statistic WHERE date = '$date';";
            $result = mysql_query($query);
            $row = mysql_fetch_array($result);
            if( $this->checkAlternative($date, "alternative$i") )
        }


```php
{
    $row = mysql_fetch_array($result);
    $alt = htmlspecialchars($row["alternative$i"], ENT_QUOTES);
    $percent = round(($alt / $total) * 100);
    echo "<div id=vote_stat>" . $this->getAlternative($date, "alternative$i") . $percent . "</div> <BR>";
}

public function getAlternative( $date, $alternative )
{
    include('config.php');

    if( !$this->validate($date) )
    {
        //Validates the date
        echo "<div id=errorstring>$invalidDate</div>";
    }
    else
    {
        $query = "SELECT * FROM $vote_table WHERE date = '$date';
        $result = mysql_query($query);
        $row = mysql_fetch_array($result);
        
        echo htmlspecialchars($row["$alternative"], ENT_QUOTES);
    }
}

public function doubleDate($date)
{
    include('config.php');

    $query = "SELECT question FROM $vote_table WHERE date = '$date";
    $result = mysql_query($query);
    $row = mysql_fetch_array($result);

    if( $row = mysql_fetch_array($result) )
    {
        return false; //exists
    }
    else
    {
        return true; //doesnt exist
    }
}
```
public function validate($validate)
{
    $goodChars = "^[0-9_.-]";
    if (ereg($goodChars, $validate))
    {
        // we split the date in 3 parts, we break it at the - sign
        $partsNumber = split("-", $validate);

        // if there aren't 3 parts, then the format is incorrect
        if (count($partsNumber) <> 3)
        {
            return false;
        }
        else
        {
            // else we save the 3 parts in $year and $month and $day
            list($year, $month, $day) = split("-", $validate);

            // and we check if the user entered a year that's too early.
            if (strlen($year) <> 4 || $year < 2007)
            {
                return false;
            }

            if (strlen($month) > 2 || $month > 12 || $month < 1)
            {
                return false;
            }

            if (strlen($day) > 2 || $day > 31 || $day < 1)
            {
                return false;
            }
        }
        return true;
    }
    else
    {
        return false;
    }
}

public function deleteQuestion($date)
{
    include('config.php');

    $query = "SELECT * FROM $vote_table WHERE date='$date';";
$result = mysql_query($query);
$rownumbers = mysql_num_rows($result);

if ( $rownumbers > 0 ) {
    $query2 = "DELETE FROM $vote_table WHERE date='$date';"
    $result2 = mysql_query($query2);
    echo "Personen har blitt slettet fra databasen"
    $query3 = "DELETE FROM $vote_statistic WHERE date='$date';"
    $result3 = mysql_query($query3);
} else {
    echo "No question registered on $date";
}

9.1.1 Source code – Installation file (Poll object)

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<link rel="stylesheet" type="text/css" href="colors.css" />
<title>Start installation</title>
</head>
<body>
<?PHP
//Connecting to DB through config file
include('config.php');

//Then create the tables needed
$query1 = "CREATE TABLE $vote_table(
'date' DATE,
'question' VARCHAR(200) CHARACTER SET utf8 COLLATE utf8_unicode_ci NOT NULL,
'alternative1' VARCHAR(50) CHARACTER SET utf8 COLLATE utf8_unicode_ci NOT NULL,
'alternative2' VARCHAR(50) CHARACTER SET utf8 COLLATE utf8_unicode_ci NOT NULL,
'alternative3' VARCHAR(50) CHARACTER SET utf8 COLLATE utf8_unicode_ci NOT NULL,"
`alternative4` VARCHAR(50) CHARACTER SET utf8 COLLATE utf8_unicode_ci NOT NULL,
`alternative5` VARCHAR(50) CHARACTER SET utf8 COLLATE utf8_unicode_ci NOT NULL,
`alternative6` VARCHAR(50) CHARACTER SET utf8 COLLATE utf8_unicode_ci NOT NULL,
PRIMARY KEY (`date`))
ENGINE = MYISAM CHARACTER SET utf8 COLLATE utf8_unicode_ci;

$result1 = mysql_query($query1);

$query2 = "CREATE TABLE $vote_statistic(
`date` DATE,
`question` VARCHAR(200) CHARACTER SET utf8 COLLATE utf8_unicode_ci NOT NULL,
`alternative1` INT,
`alternative2` INT,
`alternative3` INT,
`alternative4` INT,
`alternative5` INT,
`alternative6` INT,
`total` INT,
PRIMARY KEY (`date`))
ENGINE = MYISAM CHARACTER SET utf8 COLLATE utf8_unicode_ci;";

$result2 = mysql_query($query2);
$errornr= mysql_errno();

//Do a check to see if the creating was sucessfull
if( $result2 == 1 )
{
    echo "<p>Created table $users_table in datababase successfully</p>";
    echo "<p>Please wait ... You are being redirected.</p>";
    echo "<meta http-equiv='refresh' content='5;url=index.php' />
;
}
elsif ($errornr == "1050")
{
    echo "<p>This database already exists (Errornumber .mysql_errno()). \"</p>";
    echo "<p>But no harm ... You are being redirected.</p>";
    echo "<meta http-equiv='refresh' content='5;url=index.php' />
;
9.2 Source code - Register object

```php
<?PHP

include('config.php');

class User
{

protected $username;

public function register($name, $adress, $age, $email, $phone, $country, $city, $newsletter, $username, $password, $password2)
{
    include('config.php');

    $date = date('Y-m-d');

    $error = $this->checkFields($name, $adress, $age, $email, $phone, $country, $city, $newsletter, $username, $password, $password2);

    if( $error == "" )
    {
        $query = "INSERT into $users_table (username, name, adress, age, email, phone, city, country, registered, newsletter, password, payed) VALUES ('$username','$name', '$adress', '$age', '$email', '$phone', '$city', '$country', '$date', '$newsletter', md5('$password'), 'No' )";

        $query2 = "INSERT into $statistic_table (username, lastSignedIn) VALUES ('$username', NOT NULL)";
```
$result = mysql_query($query);
$result2 = mysql_query($query2);

//echo "<div id=goodstring>You were successfully registered</div>";
return true;

}
else
{

//echo "<div id=errorsignup>You could not be registered because of the following reason(s):
//<ul>$error</ul></div>";

    echo "<div id=midlemessage> <div id=errorstring> You could not be registered because of the following reason(s):
<ul>$error</ul></div> </div>";

}

private function checkFields( $name, $adress, $age, $email, $phone, $country, $city, $newsletter, $username, $password,
$password2 )
{

    include('config.php');

    if( $username == "" || strlen($username) < $nameLengthMIN )
    {
        $error = "$error<li>You have not written any username or less than 2 letters</li><br>

    }

    if ( !$this->validateName($name) || $name == ""
    {
        $error = "$error<li>You have used illegal characters in your name</li><br>

    }

    if ( !strlen($adress) || $adress == ""
    {
        $error = "$error<li>The adress has to be at least $adressLengthMIN letters</li><br>

    }

    if ($age == "" || strlen($age) < $ageLengthMIN )
    {
        $error = "$error<li>The age has to be at least $ageLengthMIN years</li><br>

    }

    if ( !strlen($email) || $email == ""
    {
        $error = "$error<li>The email has to be written</li><br>

    }

    if ( !strlen($phone) || $phone == ""
    {
        $error = "$error<li>The phone has to be written</li><br>

    }

    if ( !strlen($country) || $country == ""
    {
        $error = "$error<li>The country has to be written</li><br>

    }

    if ( !strlen($city) || $city == ""
    {
        $error = "$error<li>The city has to be written</li><br>

    }

    //echo "<div id=goodstring>Your registration was successful</div>";
    return true;

}

else
{

//echo "<div id=errorsignup>You could not be registered because of the following reason(s):
//<ul>$error</ul></div>";

    echo "<div id=midlemessage> <div id=errorstring> You could not be registered because of the following reason(s):
<ul>$error</ul></div> </div>";

}

private function validateName( $name )
{
    if ( !strlen($name) || $name == ""
    {
        return false;
    }

    // Regular expression to check if name contains only letters
    $regex = '/^[a-zA-Z]+$/';
    if ( !preg_match($regex, $name) )
    {
        return false;
    }

    return true;

}

private function validateAge( $age )
{
    if ( !strlen($age) || $age == ""
    {
        return false;
    }

    // Regular expression to check if age is a number
    $regex = '/^[0-9]+$/';
    if ( !preg_match($regex, $age) )
    {
        return false;
    }

    return true;

}
if ( $email == "" || strlen($email) < $emailLengthMIN )
{
    $error = "$error<li>The email adress is less than $emailLengthMIN letters<BR>
    
};

if ( !$this->validatePhone($phone) || $phone == "" )
{
    $error = "$error<li>You have not written correct phone number<BR>
    
};

if ( $city == "" || strlen($city) < $cityLengthMIN )
{
    $error = "$error<li>The city you live in has to be at least $cityLengthMIN letters<BR>
    
};

if ( $country == "" || strlen($country) < $countryLengthMIN )
{
    $error = "$error<li>Your country has to be at least $countryLengthMIN letters<BR>
    
};

if ( $this->userExists($username) )
{
    $error = "$error<li>The username is already taken<BR>
    
};

if ( $password <> $password2 )
{
    $error = "$error<li>Your password does not match<BR>
    
};

if ( strlen($password) < $passwordLength )
{
    $error = "$error<li>The password has to be at least $passwordLength characters long<BR>
    
};
if ( !$this->emailCheck($email) )
{
    //Validates the date
    $error = "$error<li>This mail adress is already registered<br>in";
    //return false;
}
else if ( !$this->validateMail($email) )
{
    //Checks for already registered date
    $error = "$error<li>You have entered an invalid mailadress<br>in";
    //return false;
}

return $error;

private function userExists($username)
{
    include('config.php');
    $query = "Select * from $users_table WHERE username='$username';
    $result = mysql_query($query);
    if ( $row = mysql_fetch_array($result) )
    {
        return true;
    }
    return false;
}

public function emailCheck($email)
{
    include('config.php');

    $query = "SELECT email FROM $users_table WHERE email = '$email';
    $result = mysql_query($query);
if ($row = mysql_fetch_array($result) )
{
    return false; //exists
}
else
{
    return true; //doesnt exist
}
}

//USed to validate mail adress, acceptable chars A-Z, a-z and 0-9
public function validateMail($mailadress)
{
    $goodChars = "^[a-zA-Z0-9_.-]+";
    if( ereg($goodChars,$mailadress) )
    {
        // we split the date in 3 parts, we break it at the - sign
        $partsNumber = split("@",$mailadress);

        // if there aren't 3 parts, then the format is incorrect
        if (count($partsNumber) <> 2)
        {
            return false;
        }
        return true;
    }
    else
    {
        return false; //Only A-E and 0-9
    }
}

public function validateAge($validate)
{
    $goodChars = "^[0-9]+$";
    if( ereg($goodChars,$validate) )
// if there are more digits than 3, then it's not correct, or the age is higher than 110.
if( strlen($validate) > 3 || $validate > 110 )
{
    return false;
}
else
{
    return true;
}
}
else
{
    return false;
}
}

public function validatePhone($validate)
{
    $goodChars = "^[0-9]+$";
    if( ereg("$goodChars",$validate) )
    {
        // if there are less than 8 digits, then it's not correct.
        if( strlen($validate) < 8 )
        {
            return false;
        }
        else
        {
            return true;
        }
    }
    else
    {
        return false;
    }
}
public function validateName($name)
{
    $goodChars = "^[a-\-A-Å\ ]+\$";
    // $goodChars = "^[a-zA-z]+([ ]?[a-zA-z]+)\$";
    if( eregi($goodChars,$name) && strlen($name) > $nameLengthMIN )
    {
        return true;
    }
    else
    {
        return false;
    }
}
?>

9.2.1 Source code – Installation file (Register object)

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<link rel="stylesheet" type="text/css" href="colors.css">
<title>Start installation</title>
</head>
<body>
<?PHP
//Connecting to DB through config file
include('config.php');

//Then create the tables needed
$query2 = "CREATE TABLE $users_table("
`username` VARCHAR(30) CHARACTER SET utf8 COLLATE utf8_unicode_ci NOT NULL,
`name` VARCHAR(100) CHARACTER SET utf8 COLLATE utf8_unicode_ci NOT NULL,
`adress` VARCHAR(50) CHARACTER SET utf8 COLLATE utf8_unicode_ci NOT NULL,
`age` INT(3) NOT NULL,
`email` VARCHAR(50) CHARACTER SET utf8 COLLATE utf8_unicode_ci NOT NULL,
`phone` BIGINT(20) NOT NULL,
`city` VARCHAR(20) CHARACTER SET utf8 COLLATE utf8_unicode_ci NOT NULL,
`country` VARCHAR(25) CHARACTER SET utf8 COLLATE utf8_unicode_ci NOT NULL,
`registered` DATE NOT NULL,
`newsletter` VARCHAR(3) CHARACTER SET utf8 COLLATE utf8_unicode_ci NOT NULL,
`password` VARCHAR(100) CHARACTER SET utf8 COLLATE utf8_unicode_ci NOT NULL,
`payed` VARCHAR(5) CHARACTER SET utf8 COLLATE utf8_unicode_ci NOT NULL,
PRIMARY KEY (`username`))
ENGINE = MYISAM CHARACTER SET utf8 COLLATE utf8_unicode_ci;";

$result2 = mysql_query($query2);

$errornr= mysql_errno();

//Do a check to see if the creating was successfull
if( $result2 == 1 )
{
    echo "<p>Created table $users_table in datababase successfully</p>" ;
    echo "<p> Please wait ... You are beeing redirected. </p>";
    echo "<meta http-equiv='refresh' content='5;url=index.php'/";        
}
elseif ($errornr == "1050")
{
    echo "<p>This database already exists (Erronumber ". mysql_errno() .":\\n". mysql_error() ."\n\br’. You are beeing redirected. </p>";

- 101 -
9.3 Source code – Login object

```php
<?PHP
class LoginObject
{
    public function login($username, $password)
    {
        include('config.php');

        // check if the username exist
        $query = "Select * from $users_table WHERE username='$username';
        $result = mysql_query($query);

        //First we check if there are any results at all
        if ($row = mysql_fetch_array($result))
        {
            // finally we check the database to see if the password is correct, if not we give an error
            if ($row["password"] == md5($password))
            {
                // tell we want to work with sessions
            }
        }
    }
}
?>
</div>
</body>
</html>
```
session_start();

// remove all the data from the session (auto logoff)
session_unset();

// remove the session itself
session_destroy();

// put the password in the session
@ session_register("me");
$_SESSION["me"] = md5($password);

// sets a password in the cookie, the name is unrecognizable
setcookie("rememberMe", md5($password), (time()+604800)); // +3600 is better?

// go to the secured page.
//echo "<meta http-equiv='refresh' content='5;url=index.php' />";
header("Location: $loginHeader");

}else
{

// and of course we tell the user that his login failed.
echo "<div id=errorstring> $incorrectLogin </div>";

// empty the fields
$Username = "";
$password = "";

}


```php
} else {
  // if the form is filled it means that the username does not exist. Therefore we show the form
  // with an error.
  echo "<div id=errorstring> $incorrectLogin </div>";

  $username == "";
  $password == "";
}
```

// This method should be executed on every page that needs authorization

```php
public function security_check() {

  include('config.php');

  // get the password if it exists
  $rememberMe = @$_COOKIE['rememberMe'];

  // tell we want to work with sessions
  session_start();

  // the $HTTP_SESSION_VARS[id] in this query indicates that we want to retrieve the password from the
  // session.
  $query = "Select * from $users_table WHERE password = ":@$_SESSION[me].";"
  $result = mysql_query($query);

  // if there are results check if the access level is high enough. If there aren't results tell the user to log-in and
  // stop
  // (die) after that.
  if ( $row = mysql_fetch_array($result) ) {
    // check the password
  }
```

- 104 -
if ($row['password'] == $rememberMe)
{
    // if the user has a valid cookie, set the session vars:
    // remove all the data from the session (auto logoff)
    session_unset();

    // put the password in the session
    @session_register("me");
    $_SESSION['me'] = $rememberMe;
}
else
{
    // if there are no results, check for an cookie
    if ($rememberMe != "")
    {
        // validate the cookie
        $query3 = "Select * from $users_table where password='".$rememberMe.'";"
        $result3 = mysql_query($query3);
        // and stop (die) after that.
        if ( $row3 = mysql_fetch_array($result3) )
        {
            // check the password
            if ( $row3['password'] == $rememberMe )
            {
                // if the user has a valid cookie, set the session vars:
                // remove all the data from the session (auto logoff)
                session_unset();

                // put the password in the session
                @session_register("me");
                $_SESSION['me'] = $rememberMe;
            }
            else
            {
                header("Location: $frontpage");
            }
        }
    }
public function loggOff() {

    include('config.php');

    $rememberMe = @$_COOKIE['rememberMe'];
    session_start();

    //sets the cookie to expire by giving it a time that has passed
    setcookie("rememberMe", "X@X", (time())-3600);

    //remove all the data from the session
    session_unset();
    unset ($_SESSION['rememberMe']);

    //remove the session itself
    session_destroy();

    //redirect to main page
    //echo "<div id=errorstring> You are now logged out</div>";
    header("Location: $frontpage");
    echo "<div id=errorstring> You are now logged out</div>";
}
9.4 Source code – Configuration file

```php
<?php

include ('userinfo.php');

$database = "signup";
$dbprefix = "community_";

//tables that are used by the objects
$users_table = $dbprefix."users";
$vote_table = $dbprefix."poll";
$vote_statistic = $dbprefix."poll_statistics";

// Login / Logout features
$incorrectLogin = 'Incorrect login'; //Incorrect login message
$loginHeader = 'redirect.php'; //Page to redirect after successful login
$frontpage = 'index.php'; //Redirect page after successful logout

//errormessages
$couldNotConnectMysql = "<BR>Error!<BR>Could not connect to MySQL.<BR>Please check your settings in config.php";
$couldNotOpenDB = "<BR>Error!<BR>Could not open database.<BR>Please check your settings in config.php";
$invalidDate = "<BR>Error!<BR>Invalid date format OR early year, must be YYYY-MM-DD";
$blankField = "<BR>Error!<BR>You must enter a date and at least two alternatives";
$noSelection = "<BR>Error!<BR>You must select one alternative";

//................. Connecting to database ............................................///

$connect2 = mysql_connect($server,$uname,$pw);
$squery = "use $database";

if( mysql_query($query) == null )
{
}

```
$query = "CREATE DATABASE $database";
$result = mysql_query($query);

if( $result == 1 )
{
    $connect = mysql_connect($server, $uname, $pw) or die ("Could not connect to mysql");
    mysql_select_db($database, $connect) or die ("Could not open database");
}
else
{
    echo "Error in database (Error number \n" . mysql_errno() . ": " . mysql_error() . "\n"");
}
else
{
    // It already exists, so we connect
    $connect = mysql_connect($server, $uname, $pw) or die ("Could not connect to mysql");
    mysql_select_db($database, $connect) or die ("Could not open database");
}
?>