

AUTOMATED CUSTOMER SUPPORT SYSTEM

Rikhard Nousiainen

Master's Thesis
June 2012
Information Technology

TAMPEREEN AMMATTIKORKEAKOULU
Tampere University of Applied Sciences

ABSTRACT

Author(s)	Rikhard Nousiainen
Master's thesis	Automated customer support system
Number of pages	64
Graduation time	June 2012
Thesis supervisor	Tony Torp
Commissioning company	EDI-Soft Finland Oy, Petri Karjalainen

When software or program is distributed to the world, support is needed for customers. Customers require help for many different tasks from basic installation to very detailed error messages. As capacity for support is usually limited in software companies, automation can help customers to find solutions into their problems without contacting support technicians. This requires a system where customers can find solutions into their problems through search engine by entering few keywords about their problem. As more customers are going to use software, program or web service, the more same questions keeps on rising and that causes frustration and extra work for people whom are working as supporting roles.

System behind this automated support system is done via web service. This service is providing easy to access communication to interface where solutions to all customers questions exists. This system is also defined as customers' self-service portal.

Keywords: automation, customer support, supporting system, self-service portal

TIIVISTELMÄ

Tekijä	Rikhard Nousiainen
Työn nimi	Asiakastuen automatisointi järjestelmä
Sivumäärä	64
Valmistumis aika	June 2012
Työn valvoja	Tony Torp
Työn tilaaja	EDI-Soft Finland Oy, Petri Karjalainen

Kun ohjelmisto laitetaan jakoon koko maailmalle, asiakastukea tarvitaan käyttäjille. Asiakkaat tarvitsevat apua moniin eri ongelmiin aina asennuksista hyvinkin yksityiskohtaisiin virheilmoituksiin. Asiakastuki on yleensä kuormitettu niin täyteen, että asiakas joutuu jonottamaan ja odottamaan omaa vuoroaan ongelman korjauksessa. Automaatiolla voidaan vähentää tätä odottamisaikaa, sillä asiakas voi löytää ongelmaansa ratkaisun ennen yhteydenottoa tukihenkilöön. Tämä tosin edellyttää järjestelmää, jossa asiakkaat voivat löytää ratkaisuja ongelmiinsa hakukoneen avulla antamalla ongelmansa avainsanoja hakusanoiksi. Usein asiakastuessa ratkaistaan samaa ongelmaa useita kertoja ja tämä kuormittaa ja työllistää asiakastukea turhaan. Kun automaattinen järjestelmä on olemassa, asiakas löytää ongelmaansa ratkaisun suoraan.

Järjestelmä pohjana toimii verkkopalvelu, joka tarjoaa käyttäjille helpon pääsyn tietokantaan josta vastaukset heidän esittämiin kysymyksiinsä palautetaan. Käyttöliittymä toimii itsepalvelu portaalina asiakkaille, joten asiakastuen ei tarvitse kuin ylläpitää palvelua ja ohjata käyttäjät käyttämään järjestelmää.

Avainsanat: automatisointi, asiakastuki , tukijärjestelmä, itsepalvelu

FOREWORD

My interest into this topic comes from my work as a technical consultant at EDI-Soft Finland Oy. I have experience in customer support in the area of IT and many of the questions from customers are very technical and can sometimes be really hard to answer without knowing own system throughout. Many times I have faced same issues coming from different customers and noticed that same solutions work with them also. This gave me an idea to move customer support towards web service and automate parts of the support work.

I also want to give special thanks to my wife Ulla who has been supporting me during this project and Tony Torp whom has given excellent directions to this work.

Tampere June 2012

Rikhard Nousiainen

TABLE OF CONTENTS

1	INTRODUCTION	8
2	AUTOMATION	10
2.1	Advantages for support	10
2.2	Disadvantages for support	11
2.3	Limitations for support	11
3	CUSTOMER SUPPORT	13
3.1	Technical customer support	13
3.2	Automated self-service support	14
3.3	Automated customer support	14
4	BASIC COMPONENTS OF AUTOMATED CUSTOMER SUPPORT SYSTEM	16
4.1	Web service.....	17
4.2	Database	17
4.3	Solution interface and HTML template	18
5	USE-CASES	20
5.1	Successful search.....	20
5.2	New error report.....	21
5.3	New solution page	22
5.4	Benefits from automated system.....	23
6	USABILITY AND USER EXPERIENCE.....	25
6.1	User experience design principles for implementation.....	26
6.2	Web service usability	27
6.3	User experience of system	30
6.4	Using support service as a mashup service.....	31
7	BUSINESS VALUES AND MARKETING SEGMENTS.....	33
7.1	Business values	33
7.2	Marketing segments	34
8	DESIGNING FUNCTIONING DRAFT OF AUTOMATED CUSTOMER SUPPORT SYSTEM.....	35
8.1	Design of web service	35
8.2	Design of database	39
8.3	Design of solution interface and HTML template	41
8.4	File system architecture	43
9	CONCLUSIONS	46
	REFERENCES.....	48
	ATTACHMENTS	50

PICTURES

Automated customer support system overview	16
Self-service truth table	19
Successful search	21
New error report	22
Technician creates a new solution to a problem	23
Why, What and How to consider when designing technology-mediated experiences...	25
User finds topics with few keywords at web service	27
Solution interface where guides to issues exists	28
New problems are registered through report page	29
Solution interface provides HTML template to create or modify solutions	30
File relation map	45

LIST OF ABBREVIATIONS

Script	Simple and small program which helps in automation
AJAX	Asynchronous JavaScript and XML
JavaScript	Scripting language that is dynamic
XML	Markup language that defines a set of rules for encoding documents into readable for human and machine.
HTML	Main markup language for web pages
PHP	Server-side scripting language to produce dynamic web pages
WS	Web service
DB	Database
SI	Solution interface
CAPTCHA	computer asking a user to complete a simple test which the computer is able to grade to make sure that input is given by a person
WYSIWYG	”What You See Is What You Get” HTML editor to help see HTML page during editing how it will be shown to end-users
APACHE	HTTP Server software to host static file share through HTTP protocol
MD5	Message-Digest Algorithm is a widely used cryptographic hash function that produces a 128-bit (16-byte) hash value
API	Application programming interface is a specification intended to be used as an interface by software components to communicate with each other

1 INTRODUCTION

Automated customer support system is generally used as a web service providing users self-service system to find out solutions into different kind of problems. Automated customer support system can be used in all kind of industry from computer technologies to car manufacturers as the basic idea behind the system stays the same all the time. Only the content of the support system is the only thing that varies between different industries.

This thesis will specify plans how to implement working automated customer support system that is functioning as an independent web service. As it works by itself, it can easily to be merged at any web service what could require automated customer support system. Techniques that are used to establish this automated customer support system are HTML, PHP, JavaScript, Apache, MySQL and AJAX. Commonly known acronym for having these techniques in a bundle is WAMP or LAMP, depending if it's meant for Windows or Linux operating systems. Automated customer support system will contain web service (WS), database (DB) and solution interface (SI). With these mentioned modules and components, knowledge base can be set up running to support customers' most effective way at their issues.

When knowledge base is built and it is containing all relevant supporting material, it is also working as information database to resolve issues before they turn into actual problems. With this system it's also possible to rate and comment material so that both support technicians and customers can benefit from the system. Technicians get valuable information about existing material and up to date information and customers' receive better view about relevance into their problems.

It is easy to establish such automated customer support system at any web service to support customers. Whenever customers are facing issues of any kind related to the product or service they are using, they can head straight away to use automated self-

support service and find solutions from a system that is running on either semi- or full automation. Only requirement for installing automated customer support system at the web service is that it must contain packages to support AJAX, PHP, JavaScript, HTML and MySQL.

2 AUTOMATION

Automation means that system is controlled by computers, most usually some scripts to do the desired work. Automation robots at industry are handling many complex simultaneous tasks at rapid speed and very accurately. Most scripts of automated systems are doing one simple task very well and these scripts can be put to parallel or serial to work together. Main idea by using automated scripts is to reduce the need of human work during production and provide automated service for users.

During 20th and 21st centuries automation has been increasing in world economy but still technology is not able to automate all the desired tasks. Most commonly automation is used to increase quality of manufacturing process and do simple repeatable actions. Any system that is using automation is decreasing operational time and handling time. Operational time is period during which a system is working in a manner acceptable to its operator or user and handling time is a period needed to transport parts or materials to or from a work area. Also automation allows employees to concentrate more on other tasks such as development and maintenance of automation processes. (The Boston Globe. 2008-03-24)

2.1 Advantages for support

While automation is decreasing human work, it is also decreasing errors in simple monotonous tasks. In support, automation is used to control machinery such as telephone switchboards and answering machines. Advantages of automation started first in 20th century from reducing production costs. It developed fast to bring more quality, accuracy and reliability into production and also into support. As automation at mid of 20th century was precise and repeatable it also gave benefits for support technicians to develop common knowledge base to share information for any system users. This leads to

a point where one problem needs to be solved only once when everyone can follow up all solved cases at the common knowledge base.

2.2 Disadvantages for support

When any system is automated, it is easily noticed that disadvantages of automated systems are that automation scripts are mostly very poor at handling error situations and also that automation requires investment, development, maintenance and full clear understanding on how system is working. It requires a lot of effort to start up automation it can easily be dismissed.

When a person is getting support at poorly handled automated customer support system, it can cause customer to become unsatisfied with solutions when customer cannot understand the main idea of the solution or cannot follow up the guide. Support should always be made very human way for customers and usually it is good that at least someone is observing how automated supporting system is handing and helping out each customer. This will require constant follow up which can be hard to fulfill as automation is working constantly 24 hours 7 days a week. Sometimes this problem can be resolved by using watchmen whom are constantly aware of the current situation and are capable of fixing problematic situations at any time.

2.3 Limitations for support

As automation is poor at handling error situations, automated tasks should not be set to areas where errors are occurring most often. Also when automation is doing tasks related to user experience, it is not good to let scripts do the decisions how to set layout of page or how to visualize system. Also systems where human consciousness is required are not good places to use automation. This can be for example courtrooms where judges and jury are deciding if something should be allowed or not.

When automated system is running constantly in the background, it is then very dependent on network and electricity. If a system fails or requires maintenance, new system needs to be taken into usage during the maintenance. This kind of system requires at least technical supports to back up whole system in case of system breakdown or unexpected results.

3 CUSTOMER SUPPORT

Customer support means that there exists a product which might require assistance during its usage. Most commonly assistance is regarding trouble shooting and finding out solutions for customers problem. Different kind of issues might rise during different phases such as installation, usage and upgrading the product. Idea of customer support is to give customers more value of the product.

If issue is related to correct usage of the product it is considered to be common customer support, but when customer is facing issues what requires knowledge about how product is working in technical way or at hardware level, it will require help from technical customer support.

3.1 Technical customer support

When customer support is called technical support it means that support is given to solve problems within a product. Most commonly technical support is given through a technical support service which can be through a phone, e-mail, consultancy, tool or a web service. Most common ways of giving technical support is giving consultancy for people working with a product or system which requires knowledge how to use it and how it works. Web service is another very popular as it can be used at all times and it's available everywhere. Only problem is that building up a web service requires investment and maintenance and it is very dependent on automation.

Technical support is usually divided into multiple tiers. These are levels from one to four. Tier one is first in line with customers to record and solve their issues. If problem is too tricky or hard to be solved by the support technician, it will be forwarded to next tier. At third tier technical support person can be architect of the system whom knows very detailed how the system is working. Beyond this point issue cannot be solved in-

side organization and issue will be forwarded to tier four to for example hardware manufacturer.

3.2 Automated self-service support

When a product is requiring constant support, it can be done through automated self-service support. This means that customer can do everything by himself without having to contact support person. Self-service support is proven to be easier, quicker and more productive for customers or system users. This kind of system requires easy usage at all times and how-to guides should be easy to access.

To build up fully automated self-service support, it requires knowledge base where exists pretty much all information regarding whole system. This information is accessed by simple keywords or questions to find out different kinds of answers. These keywords can generate further questions regarding the issue and user can select most suitable options. At the end system generates most suitable answer or solution from information that user described.

3.3 Automated customer support

Automated customer support means that there is a system what is working on semi or fully automated for every customers and users. This requires everything from creating new issues, updating or deleting old ones and getting feedback from users. Support is handled by knowledge base what contains solutions from all around the system. Search is done via guidance or search bar. Usually automated customer support can be offered through a web service but at some rare cases also phone support can be automated at certain limitations. If automated customer support is unable to solve users problem it can be either marked as new issue or offered possibility to contact technical support person via mail or phone.

Supporting at fully automated systems also require good backup system as if service breaks down for any reason, users will require other ways to solve their issues - most likely phone for support. This can lead easily to overloading customer support center and increase amount of unsatisfied customers. It would be wise to use for example cloud type services where systems can run at all times without fear of system crashing. Of course if support system is poorly designed it can become unstable at constantly growing knowledge base. This leads to a situation where working system must be maintained in case of system breaks at cloud.

4 BASIC COMPONENTS OF AUTOMATED CUSTOMER SUPPORT SYSTEM

The web service which can handle automated support has three main components: Web service (WS), Database (DB) and Solution interface (SI) and SI also includes sub component which is called HTML template. Together these components form the automated customer support system which is capable of finding solutions to rapidly asked questions easily and lead support users to desired conclusions. Connections and relationships between these components have been described in figure 1 what is architectural overview of this automated supporting system. At this system user is using the support system as a customer and technical support is working as maintaining the solutions through web service. System at this point is semi-automatic as there is a possibility to contact technical support also through phone or mail.

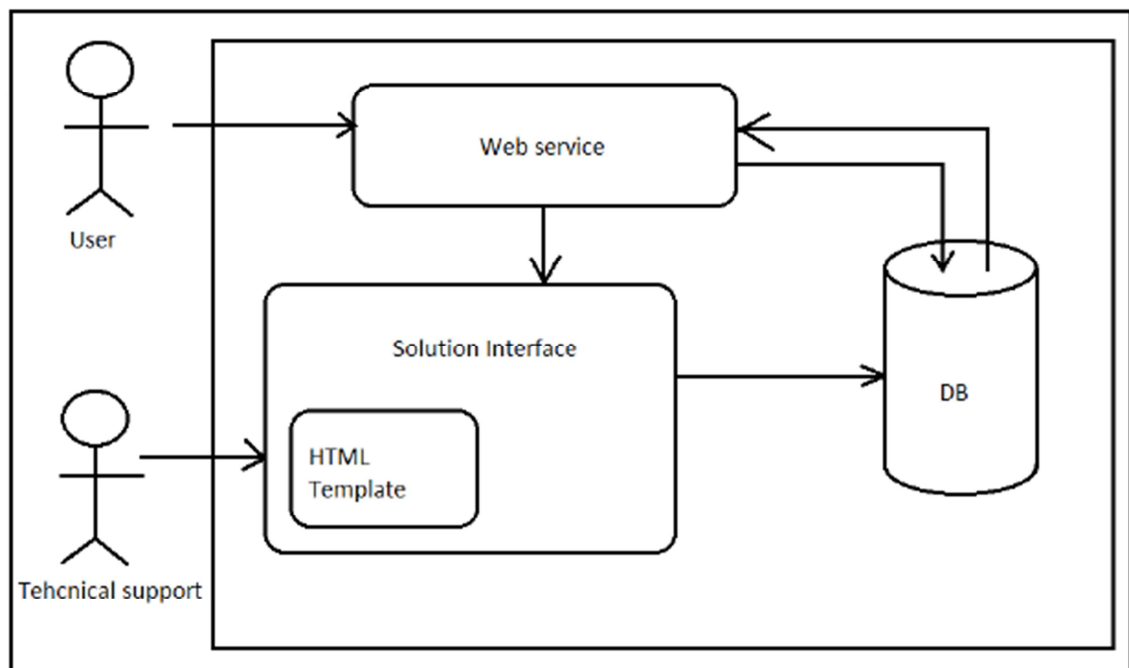


Figure 1 - Automated customer support system overview

4.1 Web service

Web service is the interface for users to search for answers into their problems. It contains a web page with simple input field where users can type the problem they are facing. Based on what user has typed the service already suggests suitable topics for the problem. All topics with keywords are set in database and input field is constantly sorting closest matches also showing rating of each page. The suggestions appear underneath the input field in a drop-down menu. The suggested topics are actually links to the solution pages so when user has done the search or selected one link directly from the drop down menu the page will be redirected to the referred solution page. The solution page contains all the needed information that is needed to fix the problem what customers are facing.

Web service is connected straight to database so all search queries are checked against keywords set in database. Whenever keywords match with users search query, database will send matches back to web service. To make this connection work in real time while typing, AJAX needs to be used for asynchronous connection. This enables rapid search and saves good amount of users' time.

Web service is the main tool for users so it must be kept very simple and with as less unrelated information as possible. If web service is doing what users expect it to do it can become commonly used tool inside organizations. This can lead it to become popular and easily approached tool. By keeping web service simple, it can be easily integrated into other systems as well as a third party plugin.

4.2 Database

The database is MySQL database which contains the links to the solutions. Database also has some additional information which includes the keywords, topic, date and the rating of the solution. The database is the place which is searched while the user types the information in to the input field. When user is making a search query at web service,

database is already sorting out possible matches from solutions keywords through AJAX. To make database most efficient, indexing is needed to sort out solutions faster. Also rating of the solution page will be returned to search results. This helps users to select links that have been confirmed to be useful for other users before.

Database is maintained by technical support personnel whom have their own access to it through a tool to create new solutions for customers. This tool is called HTML template. To get access to this tool requires also authentication where database is also very useful.

4.3 Solution interface and HTML template

The view where the solution page is opened is actually called the solution interface. The interface is an HTML page that will have text, video and images to guide the user how to fix the problem at hand. The same solution interface can also be used to list new problems and also rate existing ones. Rating system is built in so users can easily see if guide is working as expected and it gives support technicians' feedback if solution needs to be updated or fixed. If the users aren't happy with the solution they can rate the solution example from for one to five at this view and it would also be possible to leave comments and feedback that will help support technicians to know how solution is working and what area could use improvement - such as extra images or more detailed texts. If some topic starts to constantly get low scores the system technician who is responsible of that solution will know that the solution needs to be tested and improved.

If there wasn't any solution for the problem users can create a new error through web service. When creating the error users writes down as much details as possible about the problem that had occurred. Users can also use checkboxes to narrow down the area where the problem occurred. If the company is working for example in IT industry those checkboxes might be something like "System startup", "Display", "Hardware" and so on. Also email address is required to diminish false errors and also so that users

will get information as soon as the solution into their problem is available. It would be also wise to add CAPTCHA to ensure that user's response is generated by a person and not a computer bot. This helps out at getting false input from users problems.

System technician will also use the solution interface. When they get new problems that has not been solved yet they will create a new HTML page. The tool for that is the HTML template which can be used to add text, video and pictures to the page. When creating new page the technician also thinks about the possible keywords that will be attached to the page so that it will be shown in search when user looks for a solution to that kind of a problem. The technician also checks from checkbox lists that which category the solution is most suitable. When the page is ready it will be stored in the file system with the topic name and unique id. These together will form the link which will be stored in the database where search query is constantly sorting matches from users input field at web service.

HTML template is also using WYSIWYG HTML editor to make it possible to create pages simply by using commonly known layouts and settings. This makes it possible for everyone to create new pages without knowing any HTML coding conventions. Also header and footer layout for each page can be set so each company using this kind of system can have solution pages with their own look and feel experience.

	Did not contact company	Contacted company
Viewed answer	Self service success	Self service failure
Did not view an answer	Unknown. Customer may have accidentally visited support home or was just browsing	Need to encourage use of self service

Figure 2 – Self-service truth table (Werner, Fulton 2010)

5 USE-CASES

The first use-case is the most common thing that user faces. User searches for a solution into a problem at web service and finds a suitable answer for it. After this user can rate or comment about the solution.

5.1 Successful search

The first thing what user does is that he opens the Web service page as shown in Figure 2. After this he starts typing to the input field a query which is including couple of keywords from the problem he is facing. As the letters appear in the input field search is started in real time through AJAX and database is sorting out possible matches according to keywords.

After user has completed the search there is a few solution links from the database listed in the page sorted by matching keywords. User selects the one that seems to be similar to the problem he is facing and clicks the topic to open the link into solution interface.

The solution page is opened in the solution interface view and user finds a working solution to the problem. After the problem is fixed user rates the solution by giving it rating 5 as solution was just what he was looking for. This rating leaves a mark into database so technical support can see the solution is working as expected.

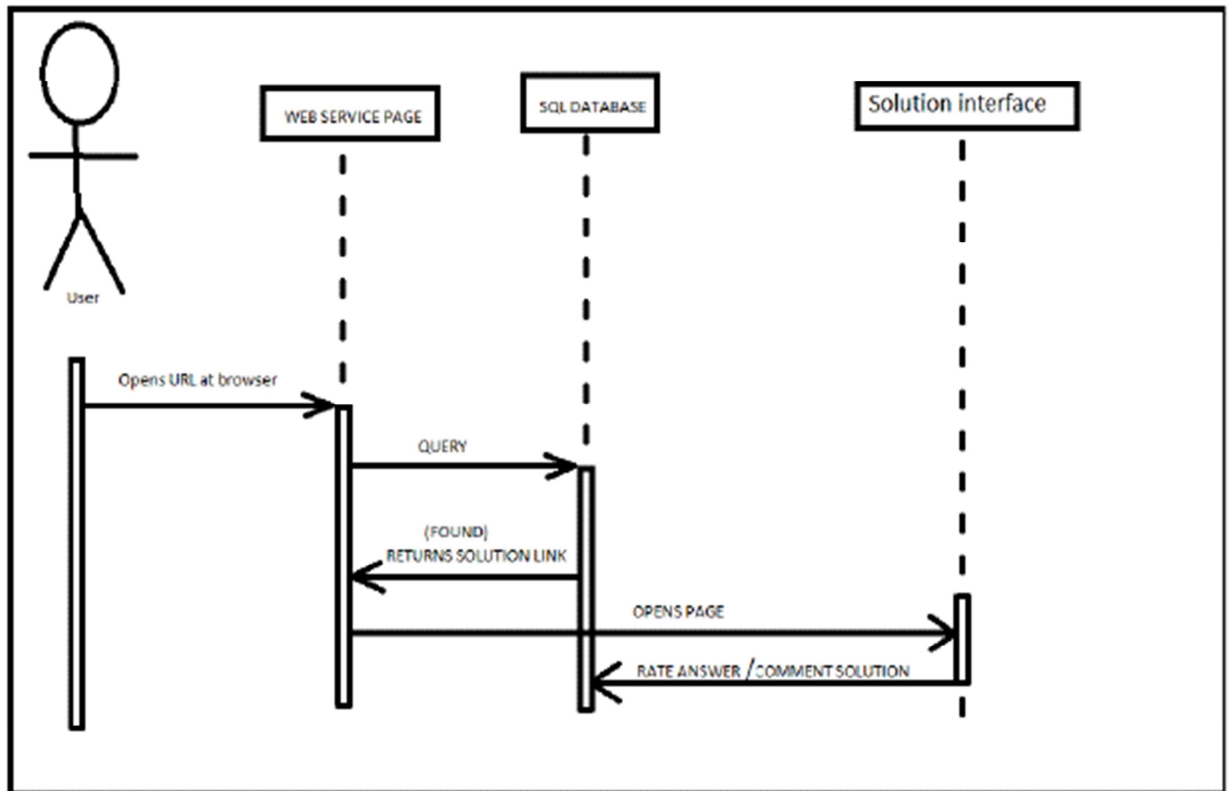


Figure 3 - Successful search

5.2 New error report

The second use-case is when user is not able to find a solution to the problem at hand. It starts just the same way as in the first scenario but now when user completes the search and opens a couple of links, none seems have had the same problem he is facing. The next step user needs to do is to list a new problem and send the report to the system technician. This is done through a link at main page to report a new issue.

When user drafts a new problem he has to select the category to a problem from a checkbox list. If the problem occurred for example in the system start up user check the “system start up” box. That automatically narrows down the area and sets up keywords when the support technician tries to find out what is causing the problem. User also includes his email address when creating the problem so that he will be immediately contacted when the solution for the problem is ready. Email address can also be used to in cases where the support technician needs more information about the problem from the

person who created the issue. Also log files can be left as attachment which helps out when figuring out where problem is.

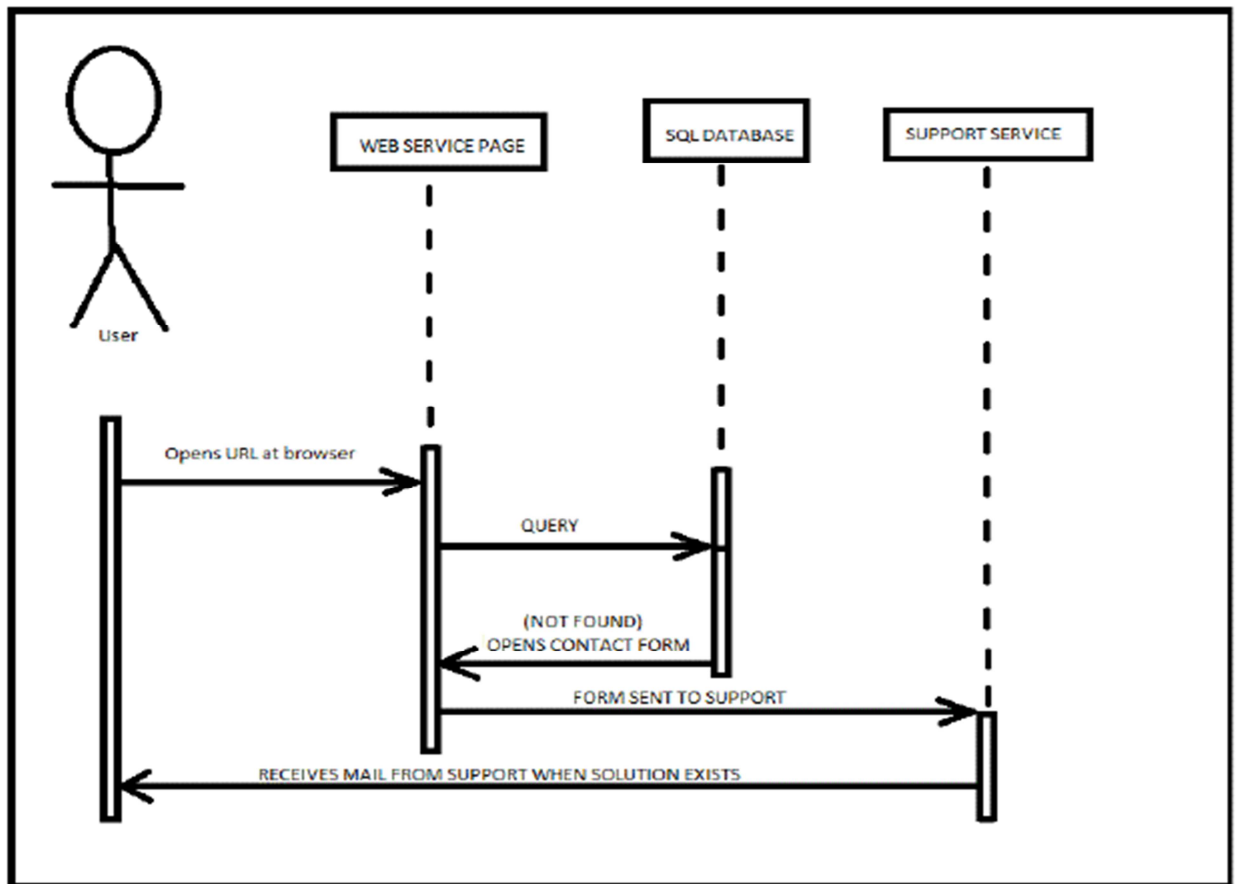


Figure 4 - New error report

5.3 New solution page

Third use-case is when technician creates a new solution to a problem that was requested and listed by a user.

The user has now listed a new problem. System technician receives an email with detailed description about the problem. After technician has studied the problem and found out solution how to fix it he opens the solution interface with support technician or admin rights and fills in the HTML template that is used to create the solution pages. With the template technician chooses from the checkboxes the correct area where the problem occurs and he also lists the keywords which can be used to find the solution.

Template allows the technician to add text fields, pictures, links to videos etc. so that the description of how fix a problem will as easy to understand as possible. This can also include WYSIWYG HTML editor so it helps out at layout of the solution page. After the page is ready it will stored in the servers file system and the link that includes the topic and unique id will be stored in the database with the keywords. When the page is stored the user who listed the problem will automatically receive an email which contains the direct link to the solution.

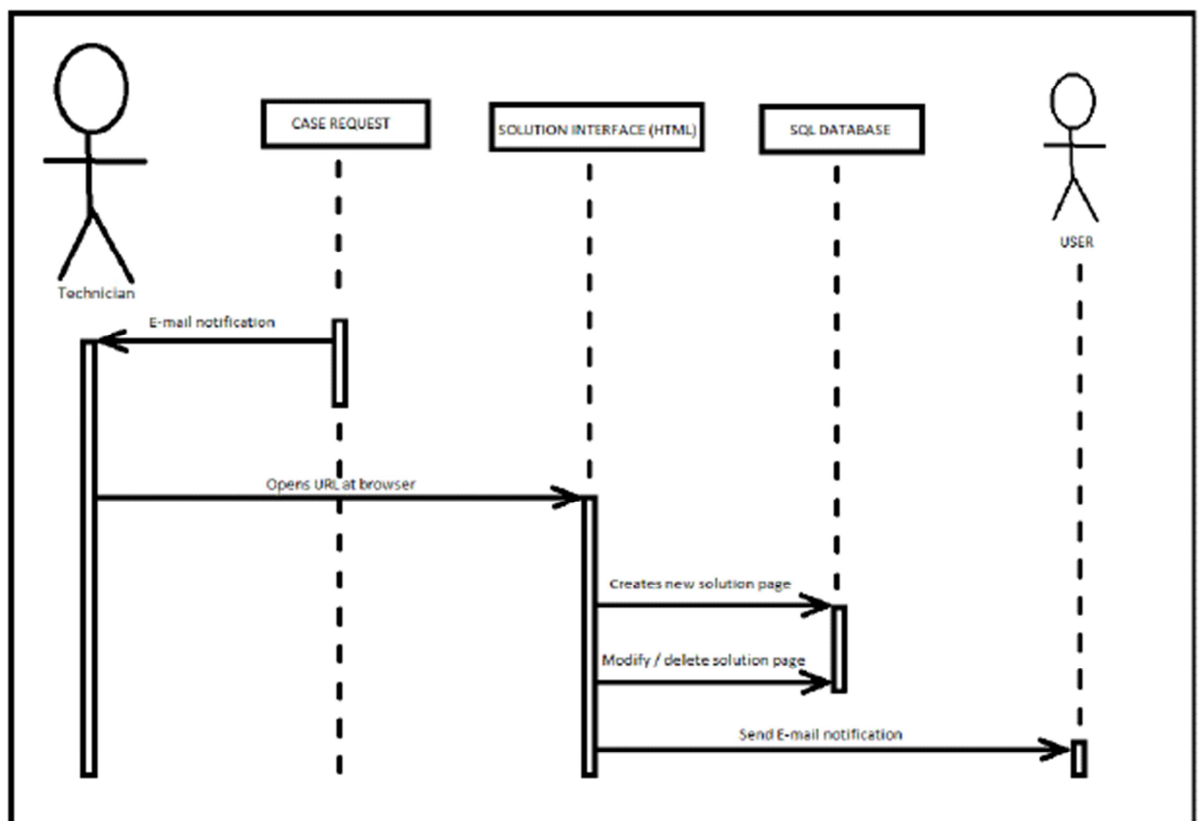


Figure 5 - Technician creates a new solution to a problem

5.4 Benefits from automated system

As distributed program evolves, automated support can easily help new customers to find solutions for many problems what old customers have already faced before. This saves good amount of support technician's man hours and is very much worth of investing money, efforts and capacity to maintaining automated supporting system. In the beginning it will be tricky to maintain automated support system as issue database doesn't include much data, but after a while customers find solutions into their problems

straight from the system and support technicians can concentrate more into maintaining the data and upgrading existing solutions.

6 USABILITY AND USER EXPERIENCE

Satisfied customer is always the main goal of whole support so automation needs to be developed as user friendly as possible. This requires good understanding of usability of how customers use web service. If web service can offer correct choices for customer the most obvious way, then web service can be stated as successful. Also setting all texts, links and images in the right position of the page is important for readability and for customer to understand guidance. Also layout template of the HTML template needs to be created as simple as possible and all pages needs to be in harmony with each other's.

Designing any service starts up by need and need requires experience that is causing the need. Experience can be described as conceptual model which includes three levels: why, what and how. What is part what describes the possibilities of an interactive product and is very much tied to the product itself. How is operational way like touching buttons and is more detailed by possibilities of the product. Why is the experience part clarifying the need of service which is causing people to use the product. These have to kept in mind when designing technology-mediated experiences. (Hassenzahl, 2011)

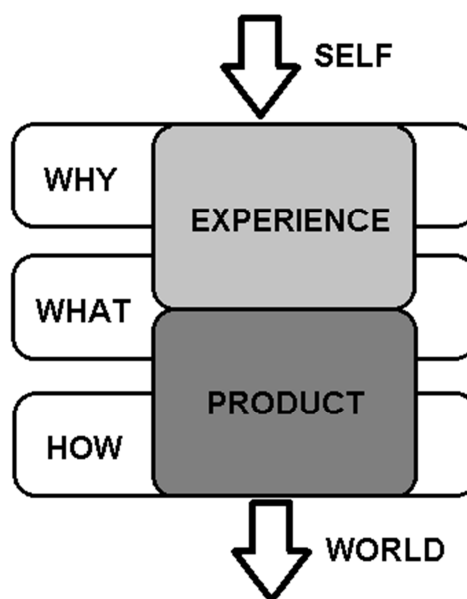


Figure 6 – Why, What and How to consider when designing technology-mediated experiences

When using why, what and how in automated customer support at very high level, why part is customers demand for solution at their issues through support service. What is the customer support service which is interacting with the customers. How is the service running in the background and analyzing the data customer is giving. This includes for example database and guiding customers towards correct solutions.

When using why, what and how methodology, implementation at users experience point of view can be defined quite straight forward as light web service doesn't require deep insight of each component and modules. For instance at solution interface what describes possibility of finding correct solution for the problem at hand, how is very much tied to the usability and search engine and why is the cause and effect that there exists a problem that needs to solved.

6.1 User experience design principles for implementation

As user experience plays big part at simple web services, it is considered good way to go through basic principles during whole implementation process. This allows multiple qualities that are affecting positive way for customers. According to Microsoft's User Experience Guide, these user experience design principles follow rules that help at reducing concepts, implementing more small details, adding look and do at UI, reducing distractions, leaving out unnecessary questions from end users, adding more personalization, giving more value to life cycle and lowering implementation time. (Microsoft, 2010)

For a light web service these principles can be followed easily and it allows creating very understandable service to provide clear and understandable content and ensuring that service is accessible and available for everyone whom might have need for it.

By following principles it can lead to a point that customers are able to resolve their problems quickly and effectively and the organization is able to build the customer relationship while saving significant costs.

6.2 Web service usability

When customer is facing issue with the system, he browses to web service where can be found a simple input field for search query. When few keywords have been entered, web service connects to database and returns solutions topics that match to entered keywords. Connection is done with PHP, AJAX, JavaScript and MySQL so solutions are seen in real time and sorted out.

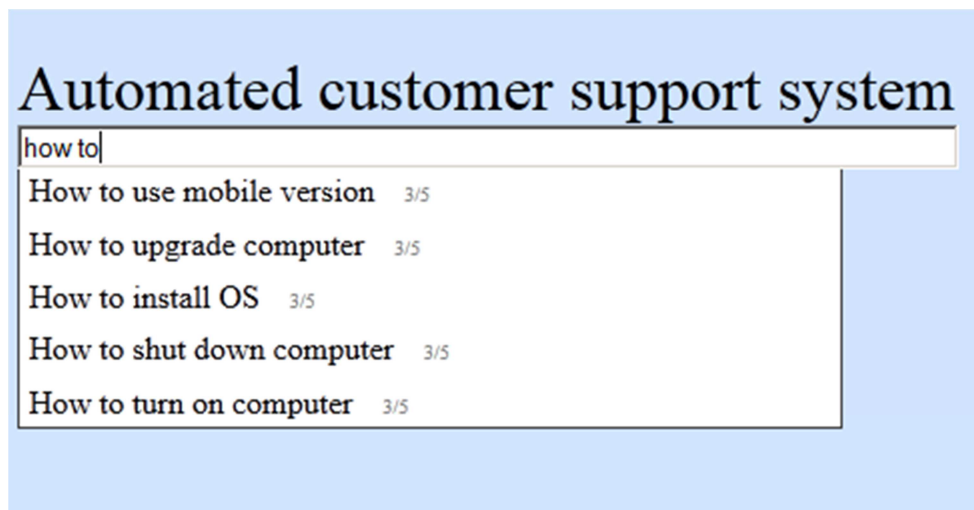



Figure 7 - User finds topics with few keywords at web service

After topic has been selected, user is taken into solution interface where solution pages exist. There are three kind of fields – texts, links and images to guide users how to solve their issues. All pages also have search bar to find another solution pages or to rate the current solution. Rating will help both support technicians and customers in the way that customers know that guide is working as expected and support technicians know if something needs to be improved at the page. Also comments can be left at the end of each solution page.

How to use WYSIWYG editor (★★★)

Using WYSIWYG



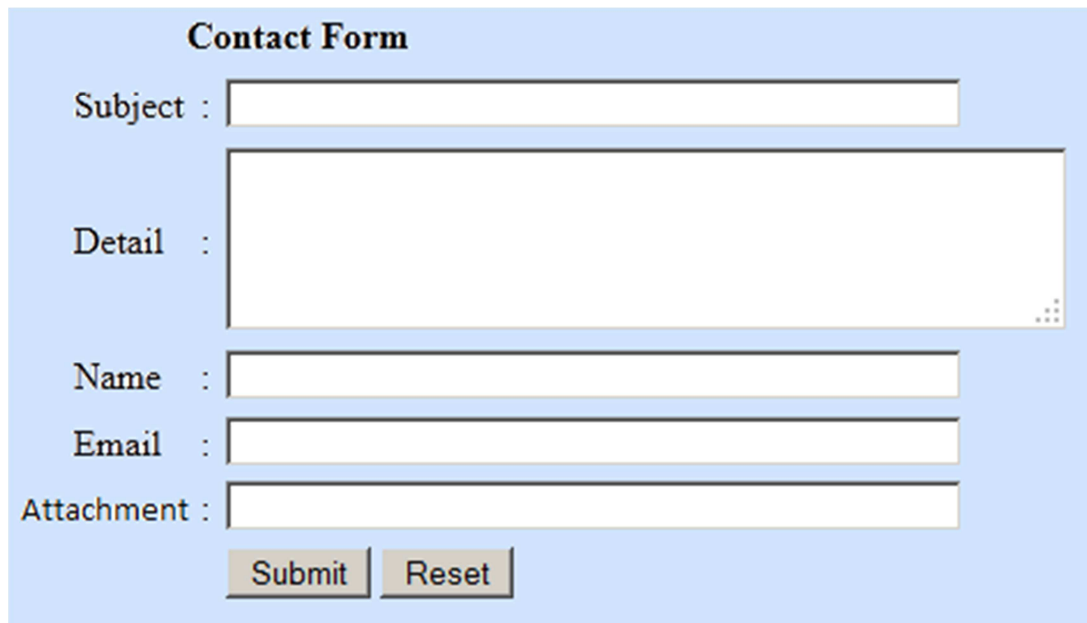
WYSIWYG implies a [user interface](#) that allows the user to view something very similar to the end result while the document is being created. In general WYSIWYG implies the ability to directly manipulate the [layout](#) of a document without having to type or remember names of layout commands. ^[3] The actual meaning depends on the user's perspective, e.g.

Start using it by...

0 5

Figure 8 - Solution interface where guides to issues exists

If user does not find solution into their problem, at first page of web service there is a link to register new issues. There user describes their issue as detailed as possible, adds screen-shots or log files as attached documents for support technicians and enters contact information so that user can be contacted once solution has been solved or if support technician needs further details about the occurred error.



Contact Form

Subject :

Detail :

Name :

Email :

Attachment :

Figure 9 - New problems are registered through report page

When new issue has been registered, support technician receives notification through e-mail. With the information described in the mail technician starts to create either new issue, or modify/delete existing ones. This is done through a HTML template which helps to create new pages easily. Support technician needs to write down topic for solution, add keywords how customers can find this page and add content with attachments. New pages are created dynamically by adding attachments, texts and pictures so each solution page can be differently set.

are working. Also by creating new issues from customers side is made so that customer doesn't get frustrated at complex problematic situations.

6.4 Using support service as a mashup service

Mashup service is defined as “a combination of preexisting, integrated units of technology, glued together to achieve new functionality, as opposed to creating that functionality from the scratch.” This means that many services can be used through one service and this is generating more services where existing functionalities are working together. (Väänänen-Vainio-Mattila, Wäljas 2011)

As customer support system is just a plain web service, it would be best working as mashup service by adding support system straight to the system where it is needed. By doing this there can be future possibilities to lead users of the service straight into the correct solution pages at error situations. Error situations can be coded into error codes and that code can be a key for customer support system to start finding correct solution immediately.

To enable system to become combined by other systems it requires APIs so other programs can interact and integrate data between the services. Architecturally, there are two styles of mashups: web-based and server-based. Whereas web-based mashups typically use the user's web browser to combine and reformat the data and server-based mashups analyze and reformat the data on a remote server and transmit the data to the user's browser in its final form. (Bolim, 2005)

At web service side currently used technologies are XMLHttpRequest, XML-RPC, JSON-RPC, SOAP and REST. These are technologies are used to send HTTP or HTTPS requests directly to a web server and load the server response data directly back into the script. At interface side where mashup service is used the technologies currently in use are HTML/XHTML, CSS, Javascript, and AJAX. These enable web service to be

established just like automated customer support system and by using web service side technologies it makes it possible to communicate between other web services to provide mashup services.

7 BUSINESS VALUES AND MARKETING SEGMENTS

This kind of system can be seen as great business value as customers can find solutions into their problems without having to contact support technicians and system can help program managers to find out areas where customers are facing their issues. This leads to the situation where program or software flaws at certain areas can be fixed before distribution. As customers are also rating solutions and leaving comments, this information can be used as valuable data for further development for upcoming releases. As this system is helping customers to solve their issues, marketing value increases as customers know there is generic helping system which is equal for all the users.

7.1 Business values

Every even a little bit complex system must have some kind of a support system to back up all the users using their product. As automated customer support system is working constantly in the background, it provides value to business and customers at all times. All stakeholders related to supporting benefit from the automation in long term. At establish phase there is lack of content but in the long run solutions are providing to be great asset for people using the system and people whom used to do support case-by-case manually.

Automated customer support system is also very accurate, reliable and precise so it is worth of investing at early phase when developing a system. At later times after continuous improvement most of the issues related to problematic situations within the product are handled by huge database. This allows customers get their work done much easier and faster than at contacting customer support service. Also quality of service increases over time when using automated support.

7.2 Marketing segments

It's easy to market a system that is light, simple and provides automated support. Also automated customer support is used widely at all business areas so automated customer support system can be used at broad and narrow markets. Potential customer can be anyone since support is used by everyone and everywhere. Support does not compete with anything as it is basically just a service for customer to use product or system at error situations. Main idea of course is to improve customer satisfaction and loyalty towards service.

8 DESIGNING FUNCTIONING DRAFT OF AUTOMATED CUSTOMER SUPPORT SYSTEM

Design contains fully working web service including database, solution interface and HTML template. Some features such as creating new solution pages, web service searching, logging and managing user accounts are included as well. There is also option at HTML template to use WYSIWYG editor to create pages more easily and seeing results immediately before distributing solution.

8.1 Design of web service

Web service is the part where users are doing their searches for different solutions. It works as a powerful search engine by returning links from MySQL database that matches closest to search query. Web service can be defined as simple search box and fast response to give solutions in a list of topics functioning as links into solutions.

Program listing 1: Search box is done with a simple form by using HTML, JavaScript, AJAX, PHP and MySQL query.

```

1      <?php session_start();
2      if (isset($_POST['search']))
3      {
4          $search = htmlentities($_POST['search']);
5          require_once("connectsql.php");
6          $sql = "SELECT * from acskeydata WHERE keywords LIKE '%$search%' OR
7              title LIKE '%$search%'";
8          $req = mysql_query($sql) or die();
9          echo '<ul>';
10         while ($data = mysql_fetch_array($req))
11         {
12             echo '<li><a href="'.htmlentities($data['link']).'"
13                 onclick="selected(this.innerHTML);">'.htmlentities($data['title']);
14             echo '<font size=1 color=grey> &nbsp; &nbsp; &nbsp;';
15             echo $data['rate'].'/5</font></a></li>';
16         }
17         echo '</ul>';
18         exit;

```

```

19     }
20     ?>
21     <form method="get" id="searchform" action="list_solutions.php">
22     <div><input autocomplete="off" type="text" value="Enter few keywords for
23     search query" name="s" size="75" id="s" onFocus="this.value=''"
24     onkeyup="request(this.value);" />
25     </div><div id="tag_update"></div>
26     </form>
27     <?
28     function selfURL()
29     {
30     $s = empty($_SERVER["HTTPS"]) ? ''
31     : ($_SERVER["HTTPS"] == "on") ? "s"
32     : "";
33     $protocol = strtolower($_SERVER["SERVER_PROTOCOL"]), "/" . $s;
34     $port = ($_SERVER["SERVER_PORT"] == "80") ? ""
35     : (":" . $_SERVER["SERVER_PORT"]);
36     return
37     $protocol . "://" . $_SERVER['SERVER_NAME'] . $port . $_SERVER['REQUEST_URI'];
38     }
39     function strleft($s1, $s2)
40     {
41     return substr($s1, 0, strpos($s1, $s2));
42     }
43     ?>

```

The program listing 1 is showing how easy it is to create SQL query to match title or keywords according to data in the database. At line 24-25 forms input field contains onkeyup method which does asynchronous AJAX call. When this field contains any letters, JavaScript will capture content from search field and forward SQL query into MySQL database. After results are back, listing will appear underneath the search box. Lines 12 – 14 show how the listing is built. At first there is link, which contains text of the topic and at the end gives current rating of the solution.

Program listing 2: AJAX is used through separate JavaScript file to capture content from search box and forwarding it into SQL query.

```

1 var myAjax = ajax();
2 function ajax()
3 {
4     var ajax = null;
5     if (window.XMLHttpRequest)
6     {

```

```
7     try
8     {
9         ajax = new XMLHttpRequest();
10    }
11    catch(e) {}
12 }
13 else if (window.ActiveXObject)
14 {
15     try
16     {
17         ajax = new ActiveXObject("Msxml2.XMLHTTP");
18     }
19     catch (e)
20     {
21         try
22         {
23             ajax = new
24             ActiveXObject("Microsoft.XMLHTTP");
25         }
26         catch (e) {}
27     }
28 }
29 return ajax;
30 }
31
32 function request(str)
33 {
34     myAjax.open("POST", "list_solutions.php");
35     myAjax.onreadystatechange = result;
36     myAjax.setRequestHeader("Content-type",
37     "application/x-www-form-urlencoded");
38     myAjax.send("search="+str);
39 }
40 function result()
41 {
42     if (myAjax.readyState == 4)
43     {
44         var liste = myAjax.responseText;
45         var cible = document.getElementById('tag_update').innerHTML
46         = liste;
47     }
48 }
49 function selected(choice)
50 {
51     var cible = document.getElementById('s');
52     cible.value = choice;
53     document.getElementById('tag_update').style.display = "none";
54 }
```

As search result is sent to MySQL database, list of closest matches are returned to a list showing topic and rating of each match and then user can pick the desired solution and moving to solution interface by clicking the topic.

Program listing 3: CSS Style contains definitions how results are seen at screen as well as layout is set to each page.

```
1     body
2     {
3         background-color:#d0e4fe;
4     }
5     #tag_update
6     {
7         display: block;
8         border-left: 1px solid #373737;
9         border-right: 1px solid #373737;
10        border-bottom: 1px solid
11        #373737;
12        position:absolute;
13        z-index:1;
14    }
15    #tag_update ul
16    {
17        margin: 0;
18        padding: 0;
19        list-style: none;
20    }
21    #tag_update li
22    {
23        display:block;
24        clear:both;
25    }
26    #tag_update a
27    {
28        width:400px;
29        display: block;
30        padding: .2em .3em;
31        text-decoration: none;
32        color: #000;
33        background-color: #FFFFFF;
34        text-align: left;
35    }
36    #tag_update a:hover
37    {
38        color: #fff;
39        background-color: #373737;
40        background-image: none;
```

```

40     }
41     div#chat
42     {
43         text-align:center;
44         width:303;
45         border:2px solid;
46         border-radius:25px;
47         -moz-border-radius:25px;
48     }

```

Most important part here is *tag_update* which is gathering all data through JavaScript by using AJAX and setting data into *<div>* section at main search file. This solution makes it possible to return solution links at any form. Also as every solution contains possibility to leave feedback and comments there is chat style to create borders with radius to each comment. This makes it easy to see name, timestamp and comment at one simple border view.

8.2 Design of database

Database contains four tables. First table is *acskeydata* which contains automatic customer support data to provide links into solution interface and also stores rating of each solution.

Program listing 4: MySQL table *acskeydata* contains record for id, title, keywords, link, rates, points, rate and datecreated.

```

1 CREATE TABLE "acskeydata" (
2 "id" int(6) NOT NULL AUTO_INCREMENT,
3 "title" varchar(200) NOT NULL,
4 "keywords" varchar(200) NOT NULL,
5 "link" varchar(200) NOT NULL,
6 "rates" int(10),
7 "points" int(10),
8 "rate" float(10),
9 "datecreated" date NOT NULL DEFAULT '0000-00-00',
10 UNIQUE KEY "id" ("id") );

```

At this table, title and link are the same except at link white spaces are replaced by underscores, current URL added to front of the link and adding *PHP* file extension to the end. Each page rating is calculated by given points divided by how many rating that specific solution has gotten and also value of the calculated rate is stored into the table with one decimal to provide numerical value during search results query.

Second and third tables at the database are intended for administrator privileges to create new access rights for technical support and to monitor login attempts. These tables are *acstrack* and *acsusers*.

Program listing 5: SQL table *acstrack* to track login attempts. It contains record for id, username, ip, tm, login.

```

1 CREATE TABLE "acstrack" (
2   "id" int(6) NOT NULL AUTO_INCREMENT,
3   "username" varchar(15) NOT NULL,
4   "ip" varchar(20) NOT NULL,
5   "tm" varchar(20) NOT NULL,
6   "login" tinyint(1) NOT NULL DEFAULT '0',
7   UNIQUE KEY "id" ("id") );

```

This table is simply recording each attempt trying to login into the management view. It stores used username, IP address of the computer, timestamp and boolean value whether login was successful or not. This information can help out management to block computers trying to hack into the system through login.

Program listing 6: There can be different kind of access rights set for automated customer support system. At least administrative and moderator rights should exist but also for each user or customer there can be set different access rights.


```

1 CREATE TABLE "acsusers" (
2 "ID" int(11) NOT NULL AUTO_INCREMENT,
3 "role" varchar(50) NOT NULL,
4 "username" varchar(50) NOT NULL,
5 "email" varchar(50) NOT NULL,
6 "password" varchar(50) DEFAULT NULL,
7 "datecreated" date NOT NULL DEFAULT '0000-00-00',
8 "datelastlogin" date NOT NULL DEFAULT '0000-00-00',
9 PRIMARY KEY ("ID"),
10 UNIQUE KEY "user_name" ("username") );

```

This table contains records for id, role, username, email, password, datecreated and datelastlogin. Role can be administrator, technical support etc. and password should be encrypted by some algorithms in case of intruders managing to see content of database.

Program listing 7: Fourth table contains data for storing feedbacks and comments from each solution page. Feedback is always implemented at the end of each solution.

```

1 CREATE TABLE acsfeedback (
2 id int(110) NOT NULL auto_increment,
3 topicid int(110) NOT NULL default '0',
4 login varchar(20) NOT NULL default '',
5 message varchar(255) NOT NULL default '',
6 itstime varchar(30) NOT NULL default '',
7 PRIMARY KEY (`id`)
8 );

```

Table has records for id, topicid, login, message and itstime. At this table topicid is used to refer into correct solution at acskeydata table as a foreign key. This makes it possible to leave different feedback at each solution page.

8.3 Design of solution interface and HTML template

Solution interface is area where pages have been created by HTML template. There exists a header and footer file for each solution page where header contains includes for web services search engine, topic and rating of that specified page and footer contains rating solutions form and feedback.

Program listing 8: HTML template contains form for topic, keywords and content. Topic field will also be the name of the file except that white spaces are replaced by underscores and file extension will be *.PHP*.

```

25 ...
26 else
27 {
28     if (isset($_POST['filename']))
29     {
30         $removeslashes = stripslashes($_POST['html_code']);
31         $fh = fopen($filename, 'w') or die("can't open file");
32         $headerstr = '<?php session_start(); include("solutionheader.php");
33         ?><br><br>';
34         $footerstr = '<br><br><?php include("solutionfooter.php"); ?>';
35         $fileStr = $headerstr . $removeslashes . $footerstr;
36         fwrite($fh, $fileStr);
37         fclose($fh);
38     }
39 }
40 ...

```

At line 30 written solution text is first being parsed through by stripping slashes away from the text to enable link and image functionality. Line 32 and 33 contains what each solution header and footer will contain. At line 34 solution is merged together and after that stored as a new solution page.

Program listing 9: HTML template contains fields for, title, keywords, solution text area and image upload through AJAX. By adding AJAX to this upload functionality it makes it possible to upload images on the fly while writing the solution. Solution text can be written with pure HTML code or by using WYSIWYG editor for faster writing. After solution has been written, there is submit button to create the solution page.

```

40 ...
41 <html>
42 <head>
43 <title>HTML Template</title>
44 <link rel="stylesheet" type="text/css" href="style.css" />
45 <script type="text/javascript"
46 src="scripts/jquery-1.3.2.min.js"></script>
47 <script type="text/javascript"
48 src="scripts/jquery-ui-1.7.2.custom.min.js"></script>
49 <link rel="Stylesheet" type="text/css"

```

```

50 href="style/jqueryui/ui-lightness/jquery-ui-1.7.2.custom.css" />
51 <script type="text/javascript"
52 src="scripts/jHtmlArea-0.7.0.js"></script>
53 <link rel="Stylesheet" type="text/css" href="style/jHtmlArea.css" />
54 </head>
55 <body>
56 <? include("fileupload.php"); ?><br>
57 Title: <font color=grey>(ex. How to solve issue x)<font color=red> use
58 letters A-Z (no åöä etc)</font><br>
59 <form id="form1" method="post" action="<?php $php_self ?>">
60 <input name="filename" type="text" value="" size="30" maxlength="60" />
61 <p></p><font color=black>Keywords: <font color=grey>(ex.
62 connect|remote|support ) <font color=red>separator <font size=5><b> |
63 </b></font><br>
64 <form id="form2" method="post" action="<?php $php_self ?>">
65 <input name="keywords" type="text" value="" size="40" maxlength="100" />
66 <p><font color=black>Write solution here: <font color=grey>(include HTML
67 tags)</font><br><input type="button" id="hideme" value="Use WYSIWYG"
68 onclick="$('#html_code').htmlarea('forecolor', 'blue');
69 document.getElementById('hideme').type='hidden';" /><br>
70 <textarea name="html_code" cols="70" rows="30" wrap="physical"
71 value="html/html" id="html_code" cols="50" rows="15"><p><h3>To solve
72 this
73 problem do the following:</h3><br>Check <b>settings</b>.</p></textarea>
74 <br>
75 <input type="submit" name="Submit" value="Create page" />
76 </form>
77 ...

```

8.4 File system architecture

Automated customer support system contains five groups. Main page (index.php) connects to administrator view (admin.php), issue reporting tool (reportissue.php), solution finder (list_solutions.php) and issue creator (HTML_template.php). At last group there are common files for all these groups like layout, SQL connection credentials, login functionalities etc.

Administrator view contains three files where md5creator.js is used to secure password encryption by adding MD5 hash encryption into password and action.png to enable click functionality in user management window. MD5 hash encryption is a good way to

store credentials into SQL database as content in the table is then shown as MD5 hash and it is not in human readable format.

Issue reporting tool is basically a simple form where can be reported new issues. It contains a folder for attachments and e-mail that is sent from submitting the form contains just a link to this attachment. This reportissue.php file contains e-mail address where form will be sent to. It also supports sending form into multiple e-mail addresses.

Solution finder is the search engine what contains one input field and AJAX call to return matching solutions. MySQL connection and linking is done at search.php and at search.js is done the AJAX call for search keywords.

Issue creator is the place where technical support persons are creating new solution pages. It contains two files for uploading: fileupload.php and ajaxupload.php and these make it possible to upload attachments on the fly so input fields can contain already written data while doing uploads. All uploaded files will be put in files-folder. Leaving feedback and rates for each solution is done at solutionfooter.php. At solutionheader.php there exists actual rate calculation by giving each solution star.png picture per rate point. At scripts- and style-folders there are needed files for enabling WYSIWYG editor.

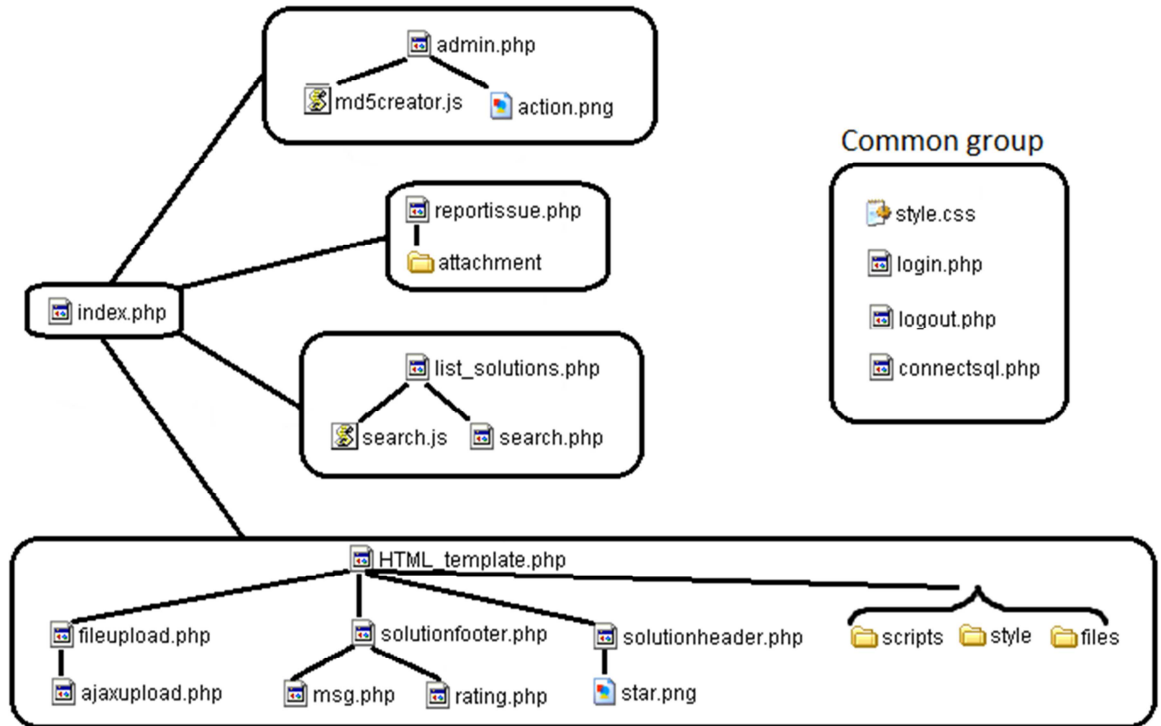


Figure 11 - File relation map

9 CONCLUSIONS

Automated customer support system is easy to establish at any kind of industry. It provides easy to access for all users and can serve multiple users simultaneously and accurately. Supporting technicians are responsible at keeping data up to date and creating new issues if it's required. By automating customer support, it gives advantages for customer by easiness and fastness of the system and also technical support persons by lowering incoming calls.

Minimum requirements when starting up automated customer support system is to have a database where to collect data of each solution and a web server to store the content. Required bundle to establish automated customer support system is called LAMP/WAMP which contains principal components (PHP, MySQL, JavaScript and APACHE) to run web server.

Each solution should have its own page to make maintenance easier at technical supporting side. Users of the support will also require interface to easily connect into a solution they need help with and this interface should be very simple and should also give good user experience. As solutions need to be consistent, technical support persons maintaining the pages should have common tool to add, modify and delete solutions. This can be a tool such as HTML template where exists common header and footer for each page and editor to easily write solutions by using WYSIWYG editor. These header and footer files can be used to bring look and feel user experience as any company desire.

If automated customer support is established as a web service, it requires stable environment and server. In case business is only relying into automated customer support, at possible error situation such as data corruption, customers will require alternative backup route to access support which can halt customers' production. Also if web service is using mashup service to exchange data between other web services it requires APIs to

provide integration. Advantages of mashup services are that they can be integrated into other services and by this way providing more information than those services were originally planned.

REFERENCES

Books, articles and web sites

Jacobs, Macfarlane, 1990. The Vital Corporation

<http://www.mirainternational.com/books/corporation/CHAP07.htm>

Otlacan, 2005. Marketing Strategy: 7 Steps to Market Segmentation

<http://ezinearticles.com/?Marketing-Strategy:-7-Steps-to-Market-Segmentation&id=82831>

EN Wikipedia Customer service, 2012.

http://en.wikipedia.org/wiki/Customer_service

EN Wikipedia Technical support, 2012.

http://en.wikipedia.org/wiki/Technical_support

Violino, 2005. Research: Automated Customer Service Takes Off

<http://www.informationweek.com/news/166403162>

Computer support for dummies by experts, 2012.

<http://computersupport360.com/talk/>

Rubens, 2005. Technical support for the neighbours

http://news.bbc.co.uk/2/hi/uk_news/magazine/4387525.stm

EN Wikipedia Automation, 2012

<http://en.wikipedia.org/wiki/Automation>

The Boston Globe, 2008. 30 Of The Fastest Declining Occupations

http://www.boston.com/bostonworks/galleries/30fast_declining_occupations?pg=10

Kongthon, Sangkeettrakarn, Kongyoung, Haruechaiyasak, 2009. Implementing an online help desk system based on conversational agent

<http://dl.acm.org/citation.cfm?id=1643823.1643908>

Hassenzahl, 2011. User Experience and Experience Design

http://www.interaction-design.org/printerfriendly/encyclopedia/user_experience_and_experience_design.html

Väänänen-Vainio-Mattila, Wäljas, 2011. *Moving Towards User-Centered Mashups: Exploring User Needs for Composite Web Services.*

http://www.cs.tut.fi/~kaisavm/CompositeUX_WIP_final_update_KVVM_060311.pdf

Windows User Experience Interaction Guidelines, 2010.

<http://msdn.microsoft.com/en-us/library/windows/desktop/aa511258.aspx>

Hess, 2010. Guiding Principles for UX Designers

<http://uxmag.com/articles/guiding-principles-for-ux-designers>

Werner, Fulton, 2010. User experience best practices for web self-service

<http://www.rightnow.com/files/whitepapers/RightNow-Web-Self-Service-Best-Practices-White-Papers.pdf>

Bolim, 2005. End-User Programming for the Web, MIT MS thesis

http://bolinfest.com/Michael_Bolin_Thesis_Chickenfoot.pdf

ATTACHMENTS

admin.php (1/3)

```

<?php session_start();
if ($_SESSION['role'] != 'admin')
{
    header('Location: index.php');
    exit();
}
?>
<html>
<head>
<link rel="stylesheet" type="text/css" href="style.css" />
<title>administrator</title>
<script src="md5creator.js" type="text/javascript"></script>
<SCRIPT language="javascript">
function generatePWD()
{
    document.getElementById("pwd").type="text";
    document.getElementById("pwd").value="";
    document.getElementById("nappi").type="button";
    document.getElementById("resetti").type="button";
}
</SCRIPT>
</head>
<body>
<FORM METHOD="LINK" ACTION="index.php" >
<INPUT TYPE="submit" VALUE="Back to main page">
</form>
<font size=5>Manage user accounts for solution interface</font>
<?php
require_once ("connectsql.php");
if (isset($_POST['hdnCmd']))
    if($_POST["hdnCmd"] == "Add")
    {
        $strSQL = "INSERT INTO acsusers ";
        $strSQL .= "(role,username,email,password,datecreated) ";
        $strSQL .= "VALUES ";
        $strSQL .= "(".$_POST["txtAddrole"]." ";
        $strSQL .= ",'"$_POST["txtAddusername"]." ";
        $strSQL .= ",'"$_POST["txtAddemail"]." ";
        $strSQL .= "MD5('".$_POST["txtAddpassword"]."', CURDATE()) ";
        $objQuery = mysql_query($strSQL);
    }
    if(isset($_POST['hdnCmd']))
        if($_POST["hdnCmd"] == "Update")
        {
            $strSQL = "UPDATE acsusers SET ";
            $strSQL .= "ID = '".$_POST["txtEditID"]." ";
            $strSQL .= ",role = '".$_POST["txtEditrole"]." ";
            $strSQL .= ",username = '".$_POST["txtEditusername"]." ";
            $strSQL .= ",email = '".$_POST["txtEditemail"]." ";
            $strSQL .= ",password = '".$_POST["txtEditpassword"]." ";
            $strSQL .= "WHERE ID = '".$_POST["hdnEditID"]." ";
            $objQuery = mysql_query($strSQL);
        }
        if (isset($_GET['Action']))
            if($_GET["Action"] == "Del")
            {
                $strSQL = "DELETE FROM acsusers ";
                $strSQL .= "WHERE ID = '".$_GET["ID"]." ";
                $objQuery = mysql_query($strSQL);
            }
        $strSQL = "SELECT * FROM acsusers";
        $objQuery = mysql_query($strSQL) or die ("Error Query
        [ ".$strSQL." ]");
?>

```

admin.php (2/3)

```

<form name="frmMain" method="post" action="<?=$_SERVER["PHP_SELF"];?>">
<input type="hidden" name="hdnCmd" value="">
<table border="1">
<tr><th></th><th><div align="center">
User name </div></th><th><div align="center">
E-Mail </div></th><th><div align="center">
Role </div></th><th><div align="center">
Password </div></th><th><div align="center">
Edit </div></th><th><div align="center">
Delete </div></th></tr>
<?php
while($objResult = mysql_fetch_array($objQuery))
{
    if (isset( $_GET["ID"] )) {} else
    $_GET["ID"] = 'temp';
    if($objResult["ID"] == $_GET["ID"] and $_GET["Action"] == "Edit")
    {
?>
        <tr>
        <td><div align="center">
        <input type="hidden" name="txtEditID" size="5"
        value="<?=$objResult["ID"];?>">
        <input type="hidden" name="hdnEditID" size="5"
        value="<?=$objResult["ID"];?>"></div></td>
        <td><input type="text" name="txtEditusername"
        value="<?=$objResult["username"];?>"></td>
        <td><input type="text" name="txtEditemail"
        value="<?=$objResult["email"];?>"></td>
        <td><input type="text" name="txtEditrole"
        value="<?=$objResult["role"];?>"></td><td>
        <INPUT id="resetti" type="button" value="Reset Password"
        onclick="generatePWD()"><br>
        <input id="pwd" type="hidden" name="txtEditpassword"
        value="<?=$objResult["password"];?>">
        <input type="hidden" id="nappi"
        onclick="document.getElementById('pwd').value =
        hex_md5(document.getElementById('pwd').value)" value="generate new
        password"></td>
        <td colspan="2" align="right"><div align="center">
        <input name="btnAdd" type="submit" id="btnUpdate" value="Update"
        OnClick="frmMain.hdnCmd.value='Update';frmMain.submit();">
        <input name="btnAdd" type="submit" id="btnCancel" value="Cancel"
        OnClick="window.location='<?=$_SERVER["PHP_SELF"];?>';">
        </div></td></tr>
<?php
    }
    else
    {
?>
        <tr><td></td><td>
        <?=$objResult["username"];?></td><td>
        <?=$objResult["email"];?></td><td>
        <?=$objResult["role"];?></td><td><div align="center">
        <?=$objResult["password"];?></div></td><td align="center"><a
        href="<?=$_SERVER["PHP_SELF"];?>?Action=Edit&ID=
        <?=$objResult["ID"];?>"></a>
        <td align="center"><a
        href="<?=$_SERVER["PHP_SELF"];?>?Action=Del&ID=
        <?=$objResult["ID"];?>"><?>
        if ($objResult["role"]!='admin')
            echo '';?></a></tr>
<?php
    }
}
?>

```

admin.php (3/3)

```

<tr><td><input type="hidden" name="txtAddID" ></td>
<td style="background-color:black;" >
<input type="text" name="txtAddusername"></td>
<td style="background-color:black;" >
<input type="text" name="txtAddemail"></td>
<td style="background-color:black;" >
<input type="text" name="txtAddrole"></td>
<td style="background-color:black;" >
<input type="text" name="txtAddpassword" size="30"></td>
<td style="background-color:black;" colspan="2" align="right">
<div align="center">
<input name="btnAdd" type="submit" id="btnAdd" value="Add"
OnClick="frmMain.hdnCmd.value='Add';frmMain.submit();" >
</div></td></tr></table></form><br>
<font size=5 color="darkred">Latest 15 unsuccessful logins</font>
<table>
<?php
echo "<th>User name</th>";
echo "<th>IP</th>";
echo "<th>Time</th>";
$query = "select username,ip,tm from acstrack WHERE login='0' ORDER BY
          id desc LIMIT 15";
$result = mysql_query($query);
while ($row = mysql_fetch_array($result))
{
    echo "<tr>";
    echo "<td>{".$row['username']."}</td>";
    echo "<td>{".$row['ip']."}</td>";
    echo "<td>{".$row['tm']."}</td>";
    echo "</tr>\n";
    $rows++;
}
?>
</table><br>
<font size=5 color="darkgreen">Latest 15 successful logins</font>
<table>
<?php
echo "<th>User name</th>";
echo "<th>IP</th>";
echo "<th>Time</th>";
$query = "select username,ip,tm from acstrack WHERE login='1' ORDER BY
          id desc LIMIT 15";
$result = mysql_query($query);
while ($row = mysql_fetch_array($result))
{
    echo "<tr>";
    echo "<td>{".$row['username']."}</td>";
    echo "<td>{".$row['ip']."}</td>";
    echo "<td>{".$row['tm']."}</td>";
    echo "</tr>\n";
    $rows++;
}
?>
</table>
<FORM METHOD="LINK" ACTION="index.php" style="display:inline;" >
<INPUT TYPE="submit" VALUE="Back to main page">
</form>
</body>
</html>

```

ajaxupload.php

```
<?php
$destination_path = getcwd().DIRECTORY_SEPARATOR;
$result = 0;
$target_path = $destination_path . '/files/' . basename(
$_FILES['myfile']['name']);
if(@move_uploaded_file($_FILES['myfile']['tmp_name'], $target_path))
{
    $result = 1;
}
sleep(1);
$upfile = basename( $_FILES['myfile']['name']);
?>
<script language="javascript" type="text/javascript">
window.top.window.stopUpload(<?php echo $result; ?>);
</script>
<script language="javascript" type="text/javascript">
window.top.window.printName("<?php echo $upfile; ?>");
</script>
```

connectsql.php

```
<?php
$db = mysql_connect("databaseurl.fi:3306", "login", "password");
$objDB = mysql_select_db("database");
?>
```

fileupload.php

```

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script language="javascript" type="text/javascript">
function stopUpload(success)
{
    var result = '';
    if (success == 1)
    {
        document.getElementById('filename').innerHTML = "File uploaded to
        http://yoururl.com/files/";
    }
    document.getElementById('upform').innerHTML = result + '<label>File:
    <input name="myfile" type="file" size="30" /></label><label>
    <input type="submit" name="submitBtn" class="sbtn" value="Upload"
    /></label>';
    return true;
}
function printName(upfile)
{
    document.getElementById('filename').innerHTML =
    document.getElementById('filename').innerHTML + upfile;
    return true;
}
</script></head><body>
<div id="filename"></div>
<div id="container">
<div id="content">
<form action="ajaxupload.php" method="post" enctype="multipart/form-data"
target="upload_target" >
<p id="upform"><br/>
<label>Upload file:<br>
<input name="myfile" type="file" size="20" /></label>
<label>
<input type="submit" name="submitBtn" class="sbtn" value="Upload" />
</label></p>
<iframe id="upload_target" name="upload_target" src="#"
style="width:0px;height:0px;border:0px solid #fff;"></iframe>
</form>
</div>
</div>
</body>
</html>

```

HTML_template.php (1/2)

```

<?php session_start();
if ( $_SESSION["login"] != "1" )
{
    header("Location: index.php");
}
echo '<div align=right><form action="logout.php" method="get">
<input type="submit" value="Log out" />
</form>';
if ( $_SESSION["role"] == "admin" )
{
    echo '<form action="admin.php" method="get">
    <input type="submit" value="User management" />
    </form>';
}
echo '</div>';
if (isset($_POST['filename']))
{
    $savelink = str_replace(" ", "_", $_POST['filename']);
    $filename = "" . $savelink . ".php";
}
if (file_exists($filename))
{
    echo "The file $filename exists go back and rename it.";
    exit;
}
else
{
    if (isset($_POST['filename']))
    {
        $removeslashes = stripslashes($_POST['html_code']);
        $fh = fopen($filename, 'w') or die("can't open file");
        $headerstr = '<?php session_start(); include("solutionheader.php");
        ?><br><br>';
        $footerstr = '<br><br><?php include("solutionfooter.php"); ?>';
        $fileStr = $headerstr . $removeslashes . $footerstr;
        fwrite($fh, $fileStr);
        fclose($fh);
    }
}
$php_self = $_SERVER['PHP_SELF'];
?>
<html>
<head>
<title>HTML Template</title>
<link rel="stylesheet" type="text/css" href="style.css" />
<script type="text/javascript"
src="scripts/jquery-1.3.2.min.js"></script>
<script type="text/javascript"
src="scripts/jquery-ui-1.7.2.custom.min.js"></script>
<link rel="Stylesheet" type="text/css"
href="style/jqueryui/ui-lightness/jquery-ui-1.7.2.custom.css" />
<script type="text/javascript"
src="scripts/jHtmlArea-0.7.0.js"></script>
<link rel="Stylesheet" type="text/css" href="style/jHtmlArea.css" />
</head>

```

HTML_template.php (2/2)

```

<body>
<? include("fileupload.php"); ?><br>
Title: <font color=grey>(ex. How to solve issue x)<font color=red> use
letters A-Z (no åöä etc)</font><br>
<form id="form1" method="post" action="<?php $php_self ?>">
<input name="filename" type="text" value="" size="30" maxlength="60" />
<p></p><font color=black>Keywords: <font color=grey>(ex.
connect|remote|support ) <font color=red>separator <font size=5><b> |
</b></font><br>
<form id="form2" method="post" action="<?php $php_self ?>">
<input name="keywords" type="text" value="" size="40" maxlength="100" />
<p><font color=black>Write solution here: <font color=grey>(include HTML
tags)</font><br><input type="button" id="hideme" value="Use WYSIWYG"
onclick="$('#html_code').htmlarea( 'forecolor', 'blue' );
document.getElementById( 'hideme' ).type='hidden';" /><br>
<textarea name="html_code" cols="70" rows="30" wrap="physical"
value="html/html" id="html_code" cols="50" rows="15"><p><h3>To solve this
problem do the following:</h3><br>Check <b>settings</b>.</p></textarea>
<br>
</p>
<input type="submit" name="Submit" value="Create page" />
</form><br><br><br><br>
<form id="form1" method="post" action="index.php">
<input type="submit" value="Back to main page"/>
</form>
<?
if (file_exists($filename))
{
    require_once("connectsql.php");
    $topic = $_POST['filename'];
    $start = strpos(selfurl(), 'HTML_template.php');
    $r = substr(selfurl(), 0, $start);
    $r = $r . $filename;
    $s = $_POST['keywords'];
    $sql = "INSERT INTO acskeydata (title, keywords, link, rates, points,
rate, datecreated) VALUES ('$topic', '$s', '$r', 1, 3, 3, CURDATE())";
    $req = mysql_query($sql) or die();
    echo "Solution page $filename was created";
}
function selfURL()
{
    $s = empty($_SERVER["HTTPS"]) ? ''
    : ($_SERVER["HTTPS"] == "on") ? "s"
    : "";
    $protocol = strtolower($_SERVER["SERVER_PROTOCOL"]), "/" . $s;
    $port = ($_SERVER["SERVER_PORT"] == "80") ? ""
    : (":" . $_SERVER["SERVER_PORT"]);
    return
    $protocol . "://" . $_SERVER['SERVER_NAME'] . $port . $_SERVER['REQUEST_URI'];
}
function strleft($s1, $s2)
{
    return substr($s1, 0, strpos($s1, $s2));
}
?>
</body>
</html>

```


index.php

```

<?php session_start(); ?>
<html>
<head>
<link rel="stylesheet" type="text/css" href="style.css" />
</head>
<body>
<font size="6">Automated customer support system<br></font>
<?php
include ("list_solutions.php");
if( $_SESSION['login'] != 1 )
{
    require( 'login.php' );
}
if ( $_SESSION["login"] == "1" )
{
    echo '<div align=right><form action="logout.php" method="get">
<input type="submit" value="Log out" />
</form><form action="HTML_template.php" method="get">
<input type="submit" value="HTML template" />
</form> ';
    if ( $_SESSION["role"] == "admin" )
    {
        echo '<form action="admin.php" method="get">
<input type="submit" value="User management" />
</form>';
    }
    echo '</div>';
}
?>
<br>
<form id="form1" method="post" action="reportissue.php">
<font color=red size=3>Didn't find what you were looking for? </font><br>
<input type="submit" value="Report new issue here"/>
</form>
</body>
</html>

```

list_solutions.php

```

<?php session_start();?>
<html>
<head>
<link rel="stylesheet" type="text/css" href="style.css" />
</head>
<body>
<script type="text/JavaScript" src="search.js"></script>
<?php include("search.php"); ?>
</body>
</html>

```

login.php

```

<?php session_start();
$username = $_POST['name'];
$password = $_POST['pass'];
$tm=date("Y-m-d-H:i:s");
$ip=$_SERVER['REMOTE_ADDR'];
if( isset($username) || isset($password) )
{
    if( empty($username) )
    {
        die ("ERROR: Please enter username! Press back
        to try again.");
    }
    if( empty($password) )
    {
        die ("ERROR: Please enter password! Press back
        to try again.");
    }
    require_once ("connectsql.php");
    $sql = "SELECT * FROM acsusers WHERE username='$username' AND
        password=MD5('$password')";
    $query = mysql_query($sql);
    if (mysql_num_rows($query))
    {
        $result = mysql_query("SELECT role FROM acsusers WHERE
            username='$username'");
        $row = mysql_fetch_array( $result );
        $_SESSION['role'] = $row['role'];
        $_SESSION['login'] = '1';
        mysql_query("UPDATE acsusers SET datelastlogin=CURDATE() WHERE
            username='$username' ");
        mysql_query("INSERT INTO acstrack(username, ip, tm, login) VALUES
            ('$username', '$ip', '$tm', '1')");
    }
    else
    {
        echo '<DIV ALIGN="right">ERROR: Incorrect username or password!
        Press back to try again.</div>';
        mysql_query("INSERT INTO acstrack(username, ip, tm, login) VALUES
            ('$username', '$ip', '$tm', '0')");
    }
}
else
{
    ?>
    <html>
    <head></head>
    <body>
    <DIV ALIGN="right">
    <form method="post" action="<?php echo $_SERVER['PHP_SELF']; ?>">
    <font size="1"> User: <input type="text" name="name" size="1"
    value="<?php echo $_COOKIE['username']; ?>">
    Pass: <input type="password" name="pass" size="1">
    <input type="submit" name="submit" value="Login"></form>
    </font>
    </div>
    </body>
    </html>
    <?php
}
?>

```

logout.php

```

<?php session_start();
unset($_SESSION['login']);
unset($_SESSION['role']);
$role = '0';
header('Location: index.php');
?>

```

msg.php

```

<html>
<head>
<link href="style.css" rel="stylesheet" type="text/css" />
</head>
<body>
<?php
if ( isset( $_POST['send'] ) )
{
    $login = $_POST['login'];
}
else
{
    $login = "";
    $message = "";
}
?>
<form method="POST">
Name:<br><input type="text" name="login" size="20" maxlength="20"
value="<?php echo $login; ?>"><br><br>
Message:<br><textarea name="message" size="30"
maxlength="255"></textarea><br><input type="submit" name="send"
value="Send"><br><br>
</form>
<?php
require_once("connectsql.php");
$title = selfURL();
$sql = "SELECT id from acskeydata WHERE link='$title'";
$req = mysql_query($sql) or die();
while ( $data = mysql_fetch_array($req) )
{
    $titleid = $data['id'];
}
function addMessage( $login, $message, $titleid )
{
    $login = mysql_real_escape_string(strip_tags( $login ));
    $message = mysql_real_escape_string(strip_tags( $message,
    "<a><b><i><u>"));
    $itstime = date("F j, Y, g:i a");
    mysql_query( "INSERT INTO acsfeedback ( topicid, login, message,
    itstime ) VALUES ( '$titleid','$login','$message',
    '$itstime' )" );
}
if ( isset( $_POST['send'] ) )
{
    addMessage( $_POST['login'], $_POST['message'], $titleid );
}
if ( isset( $_POST['send'] ) )
{
    $login = $_POST['login'];
}
else
{
    $login = "";
    $message = "";
}
include("connectsql.php");
function printMessages($titleid)
{
    $rs = mysql_query( "SELECT * FROM acsfeedback WHERE topicid=$titleid
    ORDER BY id DESC LIMIT 0,80" );
    while ( $msg = mysql_fetch_array( $rs ) )
    {
        echo "<div id=\"chat\">";
        $msg['message']=wordwrap($msg['message'], 75, "\n", 1);
        echo "<span style=\"color:navy\"><span style=\"color:blue\">";
        echo "<font size=2><b>" . $msg['login'] . "</b> <font size=1>(" .
        $msg['itstime'] . ")<br><font size=2></span><span
        style=\"color:navy\"> " . $msg['message'] . "</div></span><br />";
    }
}
printMessages($titleid); ?>
</body>
</html>

```

rating.php

```
<?php
require_once("connectsql.php");
$rate = $_GET["rate"];
$id = $_GET["id"];
$rating = $_GET["rating"];
mysql_query ("UPDATE acskeydata SET points = points+$rate, rates =
              rates+1, rate = $rating WHERE id = $id");
header("Location: index.php");
?>
```

reportissue.php

```
<?php session_start();?>
<html><head>
<link rel="stylesheet" type="text/css" href="style.css" />
</head>
<body>
<tr><td>
<strong>Report new issue </strong></td></tr><br>
<table width="400" border="0" align="left" cellpadding="0" cellspacing="1">
<tr><td>
<form id="form1" method="post" enctype="multipart/form-data"
action="<?php $php_self ?>">
<table width="100%" border="0" cellspacing="1" cellpadding="3">
<tr><td width="16%">
Subject</td>
<td width="2%"></td>
<td width="82%"><input name="subject" type="text" size="50" /></td></tr>
<tr><td>
Detail</td><td></td><td>
<textarea name="detail" cols="50" rows="4"></textarea></td></tr>
<tr><td>
Name</td><td></td><td>
<input name="name" type="text" size="50" /></td></tr>
<tr><td>
E-mail</td><td></td><td>
<input name="customer_mail" type="text" size="50" /></td></tr>
<tr><td>
Attachment (zip-file)</td><td></td><td>
<input name="attachment" type="file" /></td></tr><tr>
<td>&nbsp;</td><td>&nbsp;</td><td>
<input type="submit" value="Report issue" /></td></tr>
</table></form></td></tr></table>
<?php
if (isset($_POST['subject']))
{
    $uploaddir = 'attachment/';
    $uploadfile = $uploaddir . basename($_FILES['attachment']['name']);
    move_uploaded_file($_FILES['attachment']['tmp_name'], $uploadfile);
    $to = 'yoursupport@mail.com'
    $subject=$_POST["subject"];
    $detail=$_POST["detail"];
    $email=$_POST["customer_mail"];
    $name=$_POST["name"];
    $message = 'from: ' . $name . ' attachment:
http://youremail.com/' . $uploadfile . ' reported issue:
' . $detail . ' Link: http://youremail/HTML_template.php';
    $send_mail=mail($to,$subject,$message,$email);
    if($send_mail)
    {
        echo "Thank you. You will receive mail once solution exists.";
    }
}
}
$php_self = $_SERVER['PHP_SELF'];
?>
<form id="form1" method="post" action="index.php">
<input type="submit" value="Back to main page"/>
</form></body></html>
```

search.js

```

var myAjax = ajax();
function ajax()
{
    var ajax = null;
    if (window.XMLHttpRequest)
    {
        try
        {
            ajax = new XMLHttpRequest();
        }
        catch(e) {}
    }
    else if (window.ActiveXObject)
    {
        try
        {
            ajax = new ActiveXObject("Msxml2.XMLHTTP");
        }
        catch (e)
        {
            try
            {
                ajax = new
                    ActiveXObject("Microsoft.XMLHTTP");
            }
            catch (e) {}
        }
    }
    return ajax;
}

function request(str)
{
    myAjax.open("POST", "list_solutions.php");
    myAjax.onreadystatechange = result;
    myAjax.setRequestHeader("Content-type",
        "application/x-www-form-urlencoded");
    myAjax.send("search="+str);
}

function result()
{
    if (myAjax.readyState == 4)
    {
        var liste = myAjax.responseText;
        var cible = document.getElementById('tag_update').innerHTML
            = liste;
    }
}

function selected(choice)
{
    var cible = document.getElementById('s');
    cible.value = choice;
    document.getElementById('tag_update').style.display = "none";
}

```


solutionheader.php

```

<html>
<head>
<link rel="stylesheet" type="text/css" href="style.css" />
</head>
<body>
<script type="text/JavaScript" src="search.js"></script>
<?php include("search.php");
require_once("connectsql.php");
$title = selfURL();
$sql = "SELECT id, title, rates, points from acskeydata WHERE
link='$title'";
$req = mysql_query($sql) or die();
while ($data = mysql_fetch_array($req))
{
    echo '<font size="5"><b>';
    echo $data['title'];
    echo '</font></b>';
    $id = $data['id'];
    $rates = $data['rates'];
    $points = $data['points'];
}
$current = $points / $rates;
$stars = round($current, 0);
$setrating = round($current, 1);
echo " (";
while ( $stars >= 1 )
{
    echo '';
    $stars--;
}
echo ")";
?>

```

style.css

```
body
{
  background-color:#d0e4fe;
}
#tag_update
{
  display: block;
  border-left: 1px solid #373737;
  border-right: 1px solid #373737;
  border-bottom: 1px solid #373737;
  position:absolute;
  z-index:1;
}
#tag_update ul
{
  margin: 0;
  padding: 0;
  list-style: none;
}
#tag_update li
{
  display:block;
  clear:both;
}
#tag_update a
{
  width:400px;
  display: block;
  padding: .2em .3em;
  text-decoration: none;
  color: #000;
  background-color: #FFFFFF;
  text-align: left;
}
#tag_update a:hover
{
  color: #fff;
  background-color: #373737;
  background-image: none;
}
div#chat
{
  text-align:center;
  width:303;
  border:2px solid;
  border-radius:25px;
  -moz-border-radius:25px;
}
```