

RANIPPETTAI ENGINEERING COLLEGE

Name - N. DHINESH

Register No - 512121104302

PERSONAL BLOG ON IBM CLOUD STATIC WEB APPS

PHASE- 05

Department Of computer science Engineering

ABSTRACT

To increase an efficiency of a product, nowadays many web development companies are using different project management systems. A company may run a number of projects at a time, and requires input from a number of individuals, or teams for a multi level development plan, whereby a good project management system is needed.

Project management systems represent a rapidly growing technology in IT industry. As the number of users, who utilize project management applications continues to grow, web based project management systems enter a critical role in a multitude of companies. Thus, a proper project management system plays a distinctive part in ensuring reliable, robust and high quality web applications for customers. Developing a web based project management system and showing how, in turns, it helps users to handle projects. These processes in everyday's working life, is the scope of the thesis. The reliability and robustness of a web based project management system has also been set as the structure of the current thesis. Finally, a web based project management system has been developed, which highly meets the standards and requirements set by the company. The web based project management system uses an already integrated TRAC application that has improved to suite companies needs.

Keywords Project management, Resource management, Quality assurance

CONTENTS

WEB BASED PROJECT MANAGEMENT SYSTEM	1
1. INTRODUCTION	4
1.1 Background of Project Management	6
1.2 Motivation	8
1.3 General description of the project	10
2. PROJECT SPECIFICATION	12
2.1 Analysis and design	13
2.2 Limitations.....	18
2.3 General Function Description	18
2.3.1 Requirements	19
2.3.2 Requirements for a Web Development Process	20
2.4 Modelling	21
2.4.1 Main functionalities	22
2.4.2 Detailed description of main functionality	23
2.5 Architecture	29
3. IMPLEMENTATION.....	32
3.1 Functional requirements	33
4. TESTING	49
5. CONCLUSION	55

REFERENCES.....	
57	

1. INTRODUCTION

Web based project management systems are designed to manage and store project information that are used in web-based applications. By different groups of people such as, sales department, programmers or project managers will be let by project applications a controlled access to information and automated distribution of information.

The objective for collaboration has been: getting thing done faster, cheaper and better by applying their common knowledge, bringing together a selection of resources and attainments in a project. Because valid collaboration with teams improves productivity, speeds up result-making and optimizes of making a right decisions, it also helps to intercept precious intellectual fortune and time. Web- based project management system can surprisingly increase performance, productivity and efficiency within an organization. Since web-based applications can be accessed through any web browser, no desktop installation or updates are required. Moreover, developers, who write great code while staying out of the way are able to use it along the distance, while they stay in geographically different place and collaboration between team still exists. Please find a short overview of the system as described in Figure 1-1 below. The aim of the Figure is to provide the background of the system conducted. The background of the system includes an introduction to the system area and the motivation behind the development and research.

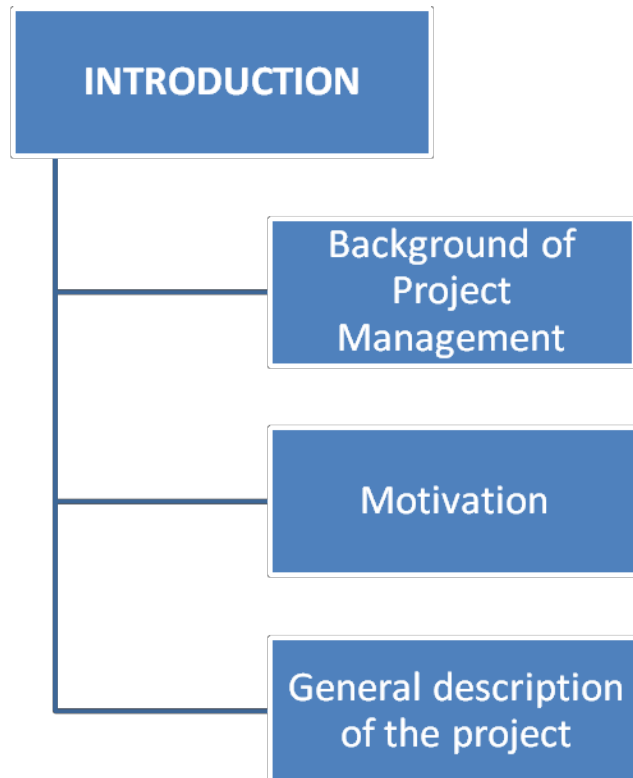


Figure 1-1 Structure of Introduction chapter

The section describing the aim and the method of the system illustrates the process which led to the final method and aim of the study through the study. The limitations and the value of the system are also described. The chapter ends with a general description of the project of the thesis.

1.1 Background of Project Management

Project Management era dates back to 1950-s, that was marked as a date for the beginning of modern project management. As before 1950's, projects were managed mostly by **Gantt Charts**, and informal techniques and tools. Later on project management tools and techniques were formalized to more professional and modern solutions. Today's rapid technological advancement, of IT industries, and globalization, project management solutions are in demand throughout the world as a fundamental force to complete projects within a defined scope, time, and within cost constraints. Today's most modern project management systems deliver innovative solutions and its management process has the latest tools, techniques, systems and schemes in use. /Project management history, 11/

But what does project management by itself mean?

Project management is like a series of actions added to a process of getting things done on a project by working with project team members to reach project schedule, cost and technical performance objectives. Definitely we could say that project management is a carefully planned and organized effort to accomplish a specific one-time objective. It doesn't matter if it is for constructing a building or implementing a major new computer system. What especially does it include then? To define and confirm the project goals and objectives we need first to develop a project plan, after that we could easily identify tasks and achieve goals. Later on, quantifying the resources is needed, determining budgets and timelines for completion. We can't forget to mention, that project management also includes managing the implementation of the project plan, along with operating regular controls to ensure that there is accurate and objective information relative to the plan, and the mechanisms to implement recovery actions where necessary. Projects usually follow major stages, including feasibility, definition, project planning, implementation, evaluation and maintenance.

For last and maybe the **one** important thing that project management

includes is risk management of project. In many projects, risks are identified and analyzed in a random. This is fatal, because unexpected risks arise, which have not been

planned for and have to be dealt with on an emergency basis. Rather than look at each risk independently and randomly, it is much more effective to identify risks and then group them into categories, and then to identify potential risks within each category. This way, common influences, factors, potential impacts and potential preventative for corrective actions, can be discussed and agreed on.

Categorizing risks is a way to systematically identify the risks and provide a foundation for awareness, understanding and action. Each potential risk needs to be carefully analyzed and the project team, the supporting teams, the organization involved in managing the project, all need to be evaluated to determine whether there is the capability to manage that risk successfully, should it arise. There are namely many different sorts of risks, and we have to decide on a project by project basis what to do about each type. Here I would like to show the breakdown, presented by Barry Boehm in his Tutorial on Software Risk Management, IEEE Computer Society, 1989. In Figure 1-2 below. (Bob Hughes and Mike Cotterell c2002, 138.)

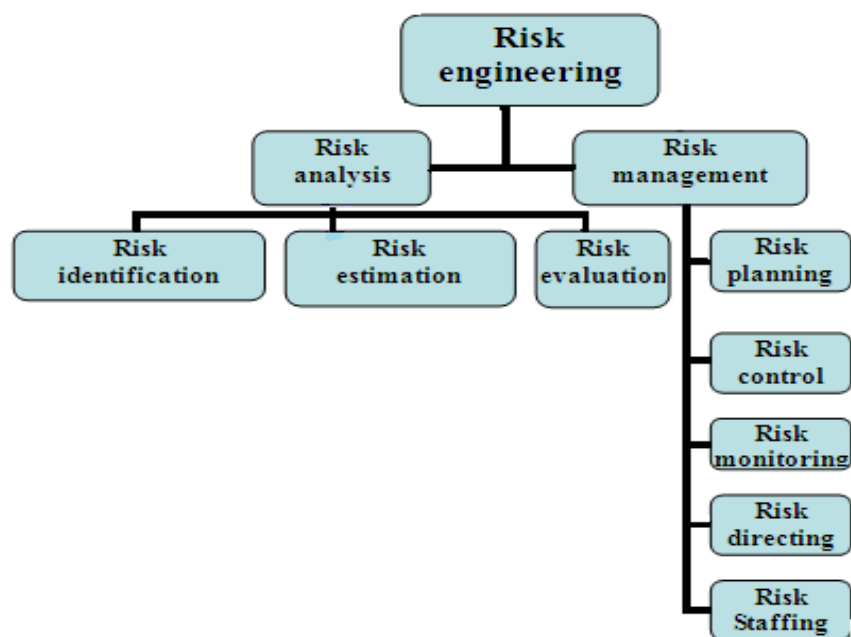


Figure 1-2 Boehm's risk engineering task breakdown (Source: Software project management 2001)

1.2 Motivation

Web based project management systems are designed to manage and store project information used as web-based applications. By different groups of people such as, sales department, programmers or project managers will be let by project applications a controlled access to information and automated distribution of information. The objective for collaboration has been: getting thing done faster, cheaper and better by applying their common knowledge, bringing together a selection of resources and attainments in a project. Valid collaboration with teams improves productivity, speeds up result-making and optimizes of making a right decisions, it also helps to intercept precious intellectual fortune and time. To prove such kind of improvement to productivity and to make easier our everyday working life, it was needed from the company to make an inside system for project management. Namely, having troubles of finding right files and wasting useful time for sending and searching documents, describing and instructing new employers of the whole system and steps that needs to be done, before beginning to make changes in projects or code.

For consuming necessary time, keeping all troubles less, and to organize all documents into one place and most importantly, to keep track of projects that are in production for customers or for keeping an eye on errors or mistakes that occur during the work process, then a good web based project management system was under consideration. To consider everyday use and needs, the aim was to make an inside system for the company. The system is for helping workers (namely programmers, project managers, developers) to deal with some specific project and its errors occurred. Project management system is needed, for helping to organize and keep an eye on the project and its process. The system is web-based; there are possibilities to add documents and specifications for specific project. Documentation can consists of different graphs, database diagrams and graphical diagrams, which are needed for project development. The most important part is that, the system has an issue tracking system, a system where can be added comments, bugs and other

related questions for specific project.

At the moment, the company has no documentation management system; all information related to one project is in different places. For example, specification requirements, application documents and other kind of documents related to one specific project lay on many places. For example project managers and sales department are keeping the documentation on their hands. When the programmer starts to create a web page, occurs the need to read the documentation of the project. The programmer basically needs to see what kind of modules for that project will be needed and get the main idea of the whole structure. So it is better, less time consuming and comfortable to get all the documents from one place. Besides that, the system has to have all documents related to that specific project, but it also has to have the code, right paths, folders and links. To keep eye on the process, find and search for bugs and take a look of the documentation of requirements.

An issue tracking system is useful while starting completely new project for another company. When this project has quite similar module of some another already implemented project, then the basic idea can be used for new project from the old project that has already the same modules and user can implement it to new project. By issue tracking, it is easy to search the issue under a project and related code or changes made to it. Check out of a project, can be done easily, while just opening the issue part for a project and checking out the bugs, comments or modifications related to it. While there is needed some specific information or comments for the code, then it can be find from bugs. Those have been added with changes of working code, which has been committed to web through user interface, with all the errors solved and described the problem itself. In a bug there should be described every problem or mistake that occurred during the process. During the project development phase, some difficulties occurred, that were caused by the programmer who was currently working on a project. For example, some problems, for running or adding some data, or maybe some importing questions or CRON problems occurred. Programmer modifications in a project and other kind of changes made in a system had to be described, not just adding comments to the code, but also describe shortly in a message – then it is

easier to find it by others. Committing code to web is needed, because it helps new developers or programmers find easily the main point and start implementing new project. The idea of tickets will be that, they have a status. Status if they have been solved or in process. Programmer can easily search tickets by its date that makes the system easier to administrate, beside a date of a ticket, we can easily see who created a specific ticket. The system helps to understand the projects structure and helps quickly to resolve bugs in a project. Finally project is using already developed TRAC system, where has been done implementations to the system. The implementations to the system are as following: file upload, deleting updated files throw web application, user management, with adding new users, data related to it, adding user to specific group, change password, or delete user, next implementation to TRAC, was a project management, with adding new project and specify administrator for the project, also project based views, were implemented – which means, that every project has its own TRAC project view. System is using an open source TRAC, which has been improved and modified by customer needs.

1.3 General description of the project

Web based project management system development is not just about writing code, that is only a part of the overall process. The customer needs are basically the most important to understand, how to analyze the requirements, produce a design and go about development and testing so that the system you deliver is a high quality and does what the client wants it to do.

As a programmer, you are unlikely to get involved in the entire system development process on every project you undertake, but regardless of how much it you are involved in, you need to understand the entire process for creating system. (Kieron Conway 2001, ixi.) For that, it is needed to have a system that helps users to make accurate and well functioned projects for clients. Web based project management system helps programmers and other users to create well functioning web applications

to clients, where all needed documentation, coding,

testing and bugs related to project are in one place for one specific project.

Projects that will be created as web applications for clients are only worthwhile if they satisfy real needs and so will be examined how can be identified the stakeholders in a project and their objectives. Having identified those objectives, ensuring that they are met is the basis of a successful project. This, however cannot be done unless there is accurate information that can be successfully added to web based project management system under user information. Web-based project management system can surprisingly increase performance, productivity and efficiency within an organization. Since web-based applications can be accessed through any web browser, no desktop installation or updates are required. Web-based applications require to be installed on a server, which is most of the time hosted by the software developer. Moreover, certain providers even offer Intranet solutions, which can be installed on your own server. (John McManus and Trevor Wood-Harper 2003, 15.)

2. PROJECT SPECIFICATION

The primary goal for the thesis was to make a complete project for daily use in one small company, which should confirm all requirements. The demands and requirements for the system come from the system structure used in our company.

The project specifications of the system are described in Figure 2-1.

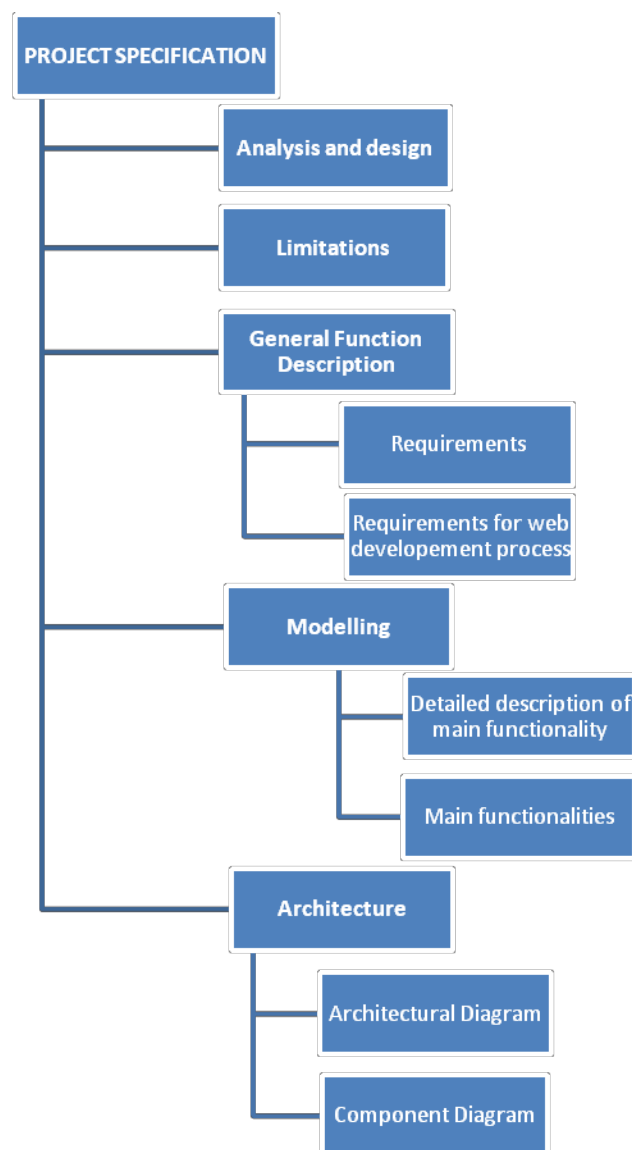


Figure 2-1 Structure of the project specification chapter

Analysis is the process whereby we break down a complex entity into its simpler, components parts. The section describing the aim and the method of the system illustrates the process which led to the final method and aim of the study through the study. Analyze of the design and the general function description is also described. The chapter ends with a description of architecture of the thesis.

2.1 Analysis and design

The primary goal for the thesis was to make a complete project for daily use in one small company, which should confirm all requirements. The web based project management system is written in PYTHON programming language, and has some PHP and HTML inside. The demands and requirements for the system come from the system structure used in our company, namely everything connected with web application development in a company has been done in PHP, and it includes all projects and systems: CRM for our clients etc. So the idea and the structure of the system came from the company. Company prefers PHP, because it is widely-used and best suitable for dynamic and interactive Web development, it is easy and can be easily modified or added to rest of the system. So generally PHP is available free of charge, deployed mostly on web servers, used on many operating systems and platforms. As the PHP is free it corresponds to our requirements for making free and open source project system. As said before, for programming web-based project management system there is used PYTHON programming language, as known, PYTHON is powerful, dynamic programming language, which is widely used in application domains. PYTHON run everywhere; it runs even on Nokia Series 60 cell phones. It is also friendly for use by everyone and easy to learn. To conclude, definitely python is that its open source, that makes it freely usable and distributable.

But why PYTHON in our web-based project management system? The answer here is quite simple. Namely, starting to create a web based project

management system, have been beforehand taken a brief look of application that already has

been developed. As company is using PHP, it was their idea to use PHP support on this project either. Finally the most suitable and obvious application was found, but it is already made, implemented and the application is widely used all over the world. The readymade application was chosen for implementation, than starting to create whole application from zero. Modifying and deploying the application was quite challenging, because to improve an already made application and to suite it to companies requirements, needs by developer more concentration and understanding of the whole structure of a system. The project is called TRAC. TRAC is already developed and fully working open source application, which corresponds to our needs; TRAC is made for software development projects, which has got integrated wiki and issue tracking system inside TRAC. /TRAC,

12/ TRAC uses a minimalistic approach to web-based software project management. The main reason why TRAC was chosen was that it provides an interface to Subversion (SVN), what is in other words, version control system. To be exact: version control system is used to maintain current and historical versions of files such as source code, web pages, and documentation. Subversion is also well-known and open source – that responds to our needs, as using open source. But in company there is also used Version control system, that is most commonly stand-alone application, but it is also embedded in various types of software. Version control or in other words revision control is the management of changes to documents, programs and other information stored as computer files. It is commonly used in system development, where many people who work on the same project are changing the same file. The main idea to use revision control system in application is to maintain documentation and configuration files as well as source code.

Version control is used in systems that are designed, developed and deployed, commonly for multiple versions of the same system to be deployed in different sites, and for the system developers to be working simultaneously on updates. Bugs and issues with system are often only present in certain versions. Therefore, for the purposes of locating and fixing bugs, it is vitally important to be able to retrieve and run different

versions of the system to determine in which version the

problem occurs. It may also be necessary to develop two versions of the system concurrently. Developers could simply retain multiple copies of the different versions of the program, and number them appropriately. This simple approach has been used on many large application projects. While this method can work, it is inefficient as many near-identical copies of the program have to be maintained. This requires a lot of self-discipline on the part of developers, and often leads to mistakes. Consequently, systems to automate some or all of the revision control process have been developed. Moreover, in system development and other environments, including in legal and business practice, it is increasingly common for a single document or snippet of code to be edited by a team, the members of which may be geographically dispersed and/or may pursue different and even contrary interests. Sophisticated revision control that tracks and accounts for ownership of changes to documents and code may be extremely helpful or even necessary in such situations. Another use for revision control is to track changes to configuration files, such as those typically stored in /etc or /usr/local/etc on UNIX systems. This gives system administrators another way to easily track changes to configuration files and a way to roll back to earlier versions should the need arise. On Figure 1-3, you are able to see how the “commit” command acts in server and client side.

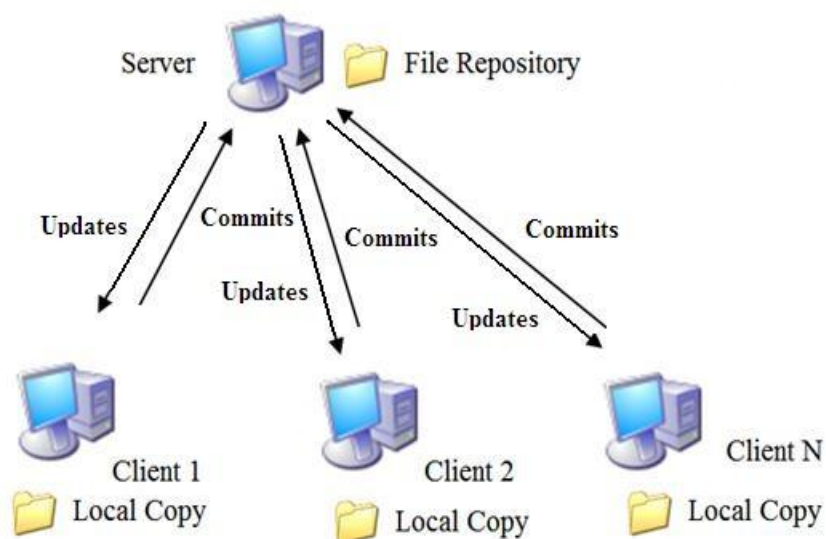


Figure1-3 Commit action through user application

When files are retrieved from the server to the client, it is called an “update”, and when new versions of the files are sent to the server from the client, it is called a “commit”. But why Subversion not CVS, which seem to be identical? Subversion has been done with an effort to write an open source version control system which operated much like CVS but which fixes the bugs and supplies the features missing in CVS. Subversion's file system is three dimensional, the third dimension is revision and each revision in a Subversion file system has its own root, which is used to access contents at that revision. Subversion repository is quite compact, while it is storing files as links to the most recent change. The Subversion file system uses transactions to keep changes atomic. A transaction is begun from a specified revision of the file system, not necessarily the latest. The transaction has its own root, on which changes are made. It is then either committed and becomes the latest revision, or is aborted. The transaction is actually a long-lived file system object; a client does not need to commit or abort a transaction itself, rather it can also begin a transaction, exit, and then can re-open the transaction and continue using it. Multiple clients can access the same transaction and work together on an atomic change.

Systems are designed to improve collaboration between the teams working on a project, reducing potential risks and helping to ensure that the project is obtained on time. As a repository they adapt for all the documents, graphs and communications relating to some specific project and are used by all co-workers in a project to access, modify, print out, and edit matter according to authorizations set up by the project creator. Programmers often confuse the terms analysis and design. Determining where analysis ends and design begins is sometimes quite difficult. As analysis proceeds, design consideration keep popping up, making it easy to get sidetracked into following up in depth on such issues. Dealing with design issues at a superficial level at this stage helps minimize the technical risks, but you must temper any time you schedule at risk. Like so many things in system development, a fine balance is essential. Based on the requirements and the detailed analysis of a new system, the new system must be designed. It is a most crucial phase in the development of

a system.

Normally, the design proceeds in two stages:

- preliminary or general design
- structure or detailed design

In the preliminary or general design, the features of the new system are specified. The objective of the detailed design phases is to create a design that will correctly and completely implement the requirements. For the preliminary phase, the main goal is to map out how the web-based project management system will perform the functions specified in the requirements, within the defined interfaces, and the environment. At this phase, the designer needs to maintain a systems perspective and look at the system operations in concert with the rest of the operations. The objective of design assurance is to verify that the design does implement all the requirements, and that it implements nothing but the requirements. The main design activities for the preliminary design phase are:

1. Create the high-level design description.
2. Any derived requirements that result from the process are fed back to the requirements engineering process
3. Any omissions or errors are resolved
4. Include reliability, maintenance, and test features that are necessary to meet performance and quality requirements, and to ensure that testing can be performed to verify the requirements.
5. Identify constraints on other system elements that are a result of this high-level design

Analysis emphasizes an investigation of the problem and requirements, rather than a solution. For example, if a new online project management system is desired, how will it be used? What are its functions? Analysis is more a board term, best qualified, as in requirements analysis an investigation of the requirements. Design emphasizes a conceptual solution (in software and hardware) that full-fills the requirements, rather than its implementation. For example, a description of a database schema and software objects. Design ideas often exclude low-level details – obvious to the intended consumers. Ultimately,

designs can be implemented, and the implementation (such as code) expresses the true and complete realized design. With analysis, the term is best qualified, as in database design. Useful analysis and design have been summarized in the phrase

„do the right thing (analysis), and do the thing right (design).“

2.2 Limitations

There occurs some limitations and problems in Subversion, namely SVN allows only directory access control and has less detail file access control. Problem occurs in Subversion and in projects where directories are not structured to address the functional separation among various objects. For example, directories like lib, src, and bin do not address security and access control in most cases. For a second case, is the implementation of the file and directory rename operation. Subversion currently has the way to implement the renaming of files and directories as a “copy” to the new name followed by a “delete” of the old name. This means that only names are changed and all data related to the edit history remains the same and SVN will still use the old name.

2.3 General Function Description

For us the more helping way to analyze the big picture and its relations between system elements are through diagrams, which basically helps to discover or explore the relations, while allowing us to ignore or hide uninteresting details. To conclude, we can say that it is the most essential value of the UML and can be said that also the simplest value of the UML or any diagramming language.

2.3.1 Requirements

To determine the requirements for the development of a project management system for company as an inner system, that aids the project officers in the daily task and responsibility of effectively and professionally managing each of the projects as well as the program. It was the task of to run a series of requirements analysis sessions. There has been found that the analyses shows that , it is more time consuming and gives a better outcome for the client or the end-user, while we use a project management system in our daily working life. Each of these outcomes will be supported by evidence collected by the individual projects that showed exactly the same result, for collecting and gathering information for everyday working life.

Was identified the following main project management system requirements:

- User management integration to the system
- Security integration on application
- To track projects and its issues / bugs related to specific project
- Upload and delete files in a project
- Create new users who belong to specific group (Admin, SVN)
- Generate project and defined admin rights to it
- Create a shared database for information

It was agreed that the project management system would be developed as a system to match the requirements of the program that would integrate seamlessly with the security infrastructure of the existing portal website CMS. This will enable authorized users to log in to the system and see a summary of relevant project management system information. From logged in users who have access will be able to seamlessly access to project management system. A system requirements specification is a complete description of the behaviour of the system

to be developed. It includes a set of use cases that describe all of the

interactions that the users will have with the system. In addition to use cases, the system requirements specification contains functional requirements, which define the internal workings of the system: that is, the calculations, technical details, data manipulation and processing, and other specific functionality that shows how the use cases are to be satisfied. It also contains non-functional requirements, which impose constraints on the design or implementation (such as performance requirements, quality standards or design constraints). The objective for collaboration has been the same: getting thing done better, faster and cheaper by bringing together a variety of resources and apply their collective knowledge and abilities in a project. Because valid collaboration with teams improves productivity, speeds up result-making and optimizes of making a right decisions, it also helps to intercept precious intellectual fortune and time.

2.3.2 Requirements for a Web Development Process

From experience in developing Web applications, have derived a list of requirements for Web development process. The most important requirements are to provide end-user involvement, prototyping, change management, immediate response, risk minimization, no administrative overhead and transparency and guidance. Knowing the end users requirements is essential for the development of successful Web applications. Defining the main goals for the development of a Web application, then customer is not the actual end-user and, therefore, he or she is not able to define all the requirements that are important to end-user. Prototyping is used to leverage the involvement of end-users in Web application development. Prototyping produces a preliminary version of the required system that can be reviewed by end-users. After review, the prototype is added to and altered to produce another version closer to the one that is wanted. Figure 2-2 gives a diagram of prototyping process.

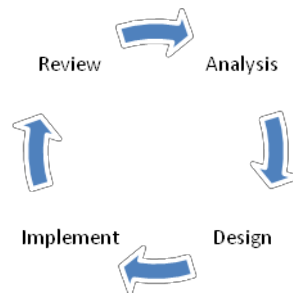


Figure 2-2 Prototyping process

Project development prototyping is essential, because finding the way to solution is much faster and more effective way to speed up the process of the project and find the best option for best results. Prototyping in our case influences the project development process.

2.4 Modelling

Modelling is like building a representation of things in the „real world“ and allowing ideas to be investigated. In fact, model is more likely a way of expressing a particular view of a system. Mainly modelling is used to:

- understand the problems involved in building some system
- an aid to communication between those involved in the project
- a component of the methods used in development activities such as the analysis of the requirements

The way modelling is used in this project is called Unified Modelling Language (UML) that is a standard language for specifying, visualizing, constructing, and documenting the artefacts of systems, as well as for business modelling and other non-software systems. The UML represents a collection of best practices that have proven successful in the modelling of large and complex systems. It is an important part of developing system and their development process. The UML uses mostly graphical notations to express the design of projects, it helps project

teams communicate, explore potential designs, and validate the architectural design of the system.

The primary goals in the design of the UML were:

1. Provide users with a ready-to-use, expressive visual modelling language so they can develop and exchange meaningful models.
2. Provide extensibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development processes.

Provide a formal basis for understanding the modelling language

2.4.1 Main functionalities

Use Cases are text stories, widely used to discover and record requirements. Use cases need to be more detailed or structured and emphasize the user goals and perspective. A use case diagram in the Unified Modelling Language (UML) is a type of behavioural diagram defined by and created from a use case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals, and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted. On the Figure 2-3 is shown a functionality of project management system, where user and administrator have different functionalities to run. While user is modifying or upload / download or delete files in a system while user is logged in, then for our administrators has more rights to control in the system. Namely administrator has the right to add, modify or delete users in a system or add new projects and definitely available to modify projects as the user or add new projects to the system. The Figure below illustrates exactly what kind of possibilities or options are for user and administrator of the system.

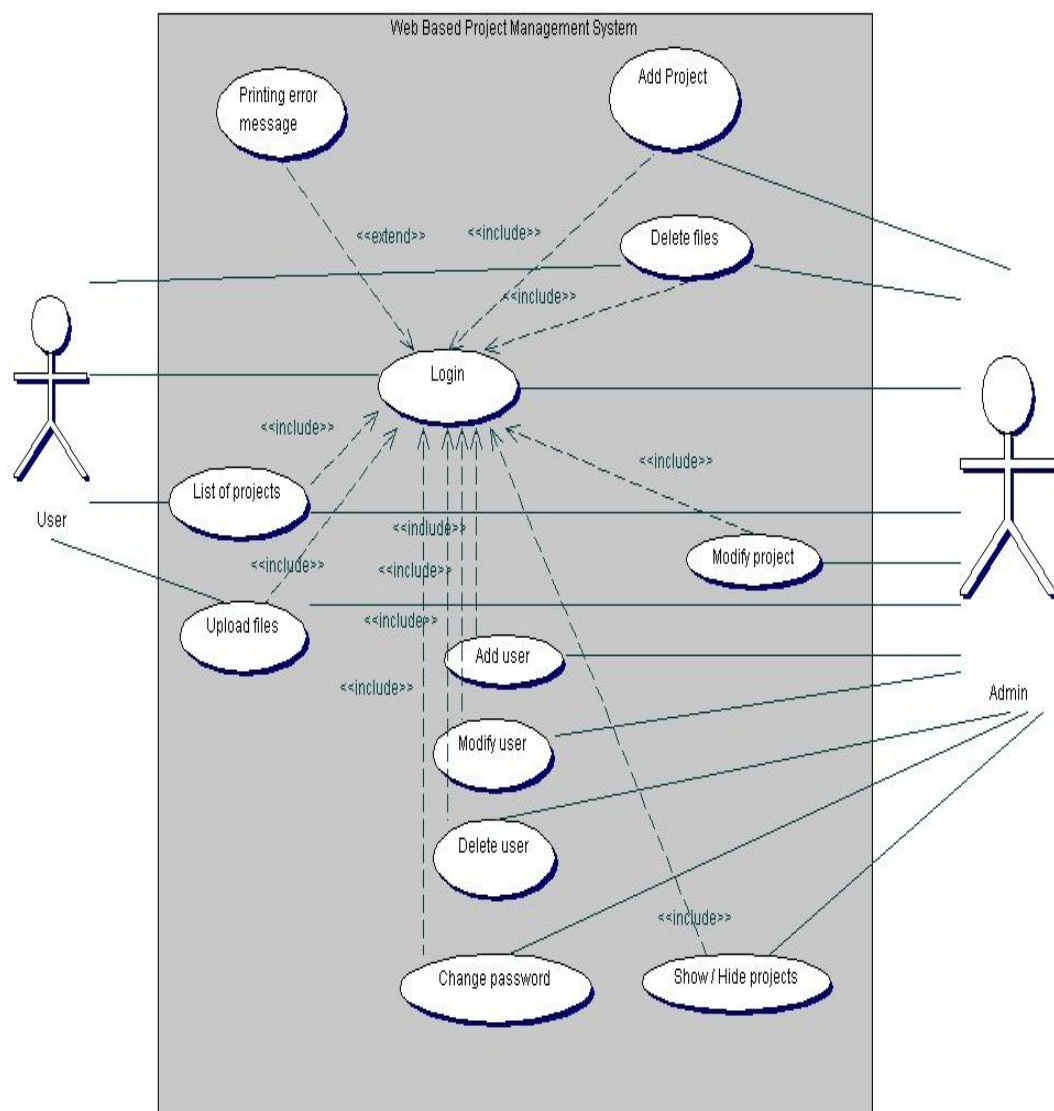


Figure 2-3 Functionality of project management system

2.4.2 Detailed description of main functionality

The term interaction diagram is a generalization of more specialized UML

diagram type:

- Sequence diagram

Sequence diagrams illustrate interactions in a kind of fence format, in which each

new object is added to the right. A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. Earlier I stated that sequence diagrams are effectively a form of visual coding, or perhaps another way to think of it is that sequence diagrams can be used for very detailed design.

The flow of messages, events and actions between the objects or components of a system have been easily used to represent or modelled by UML sequence diagrams. On the vertical direction is showed the time that you can find on Figure

2-4 and on 2-5. The reciprocal effect of sequence is the header elements, which are indicated horizontally at the top of the diagrams on the Figures 2-4 and 2-5. Sequence diagrams are mainly used for finding the logic of the system. Document, model the design and displaying the architecture of the system, by describing the actions that need to be performed for completing a task. UML provides a dynamic view of the behaviour of the system that can be extremely complicated to read from diagrams or work description, that's why sequence diagrams are called as powerful designing tools.

To conclude, sequence diagrams are useful in system architecture, as really good engineering tools to design appropriate system, they have been used also in describing object-oriented systems. In other systems we use the tool for showing the system architecture with flow diagrams and protocol stack design with analysis.

On Figure 2-4 Sequence diagram represents a file upload structure in a system used by administrator. First user goes to main page where user gets a login window to authenticate him. If log-in info is an accurate user will be lead to admin page, where user is able to see a list of projects, a links which will lead a user to users management page, which makes a new project and which show / or hide a projects. While user chooses a project, the system will lead user to TRAC page, on the page there is a link "File Upload" inside system is reading and configuring TRAC conf file. Meanwhile the system is also accessing to DB. While user wants to upload

a file, system is browsing and reading SVN repo,

checkouts if there already similar file exists, if not system commits changes, inserts file to TRAC and returns back message that the file is successfully added to system.

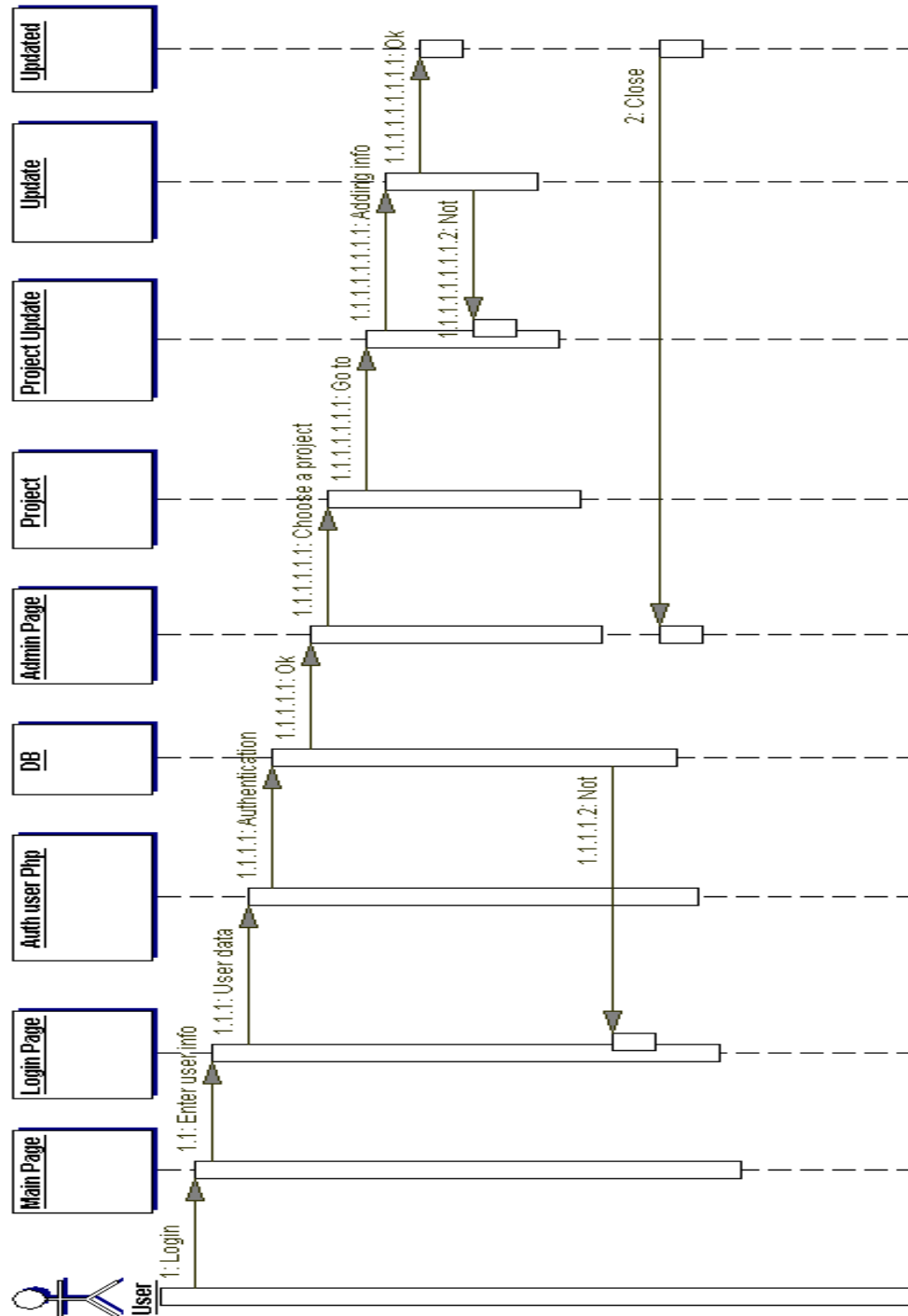


Figure 2-4 Sequence Diagram of File Upload

On Figure 2-5 is shown a sequence diagram of adding users in administrator view through users" management. Firstly user goes to main page and will get a log in window, after adding a username and password, the system checks if the information is correct. If it is correct the user will be lead to main page, where user is able to see a list of projects, links to user"s management page, add project page and show/hide project. User moves to users management and under there is a add user field, where actually TRAC will get as a root user to work under SUDO that adds contact information to a file. System sends back a replay to user, if the process was successful. The main thing on a Figure to put an attention is that the system is divided as one part belongs to www-data and another one to root.

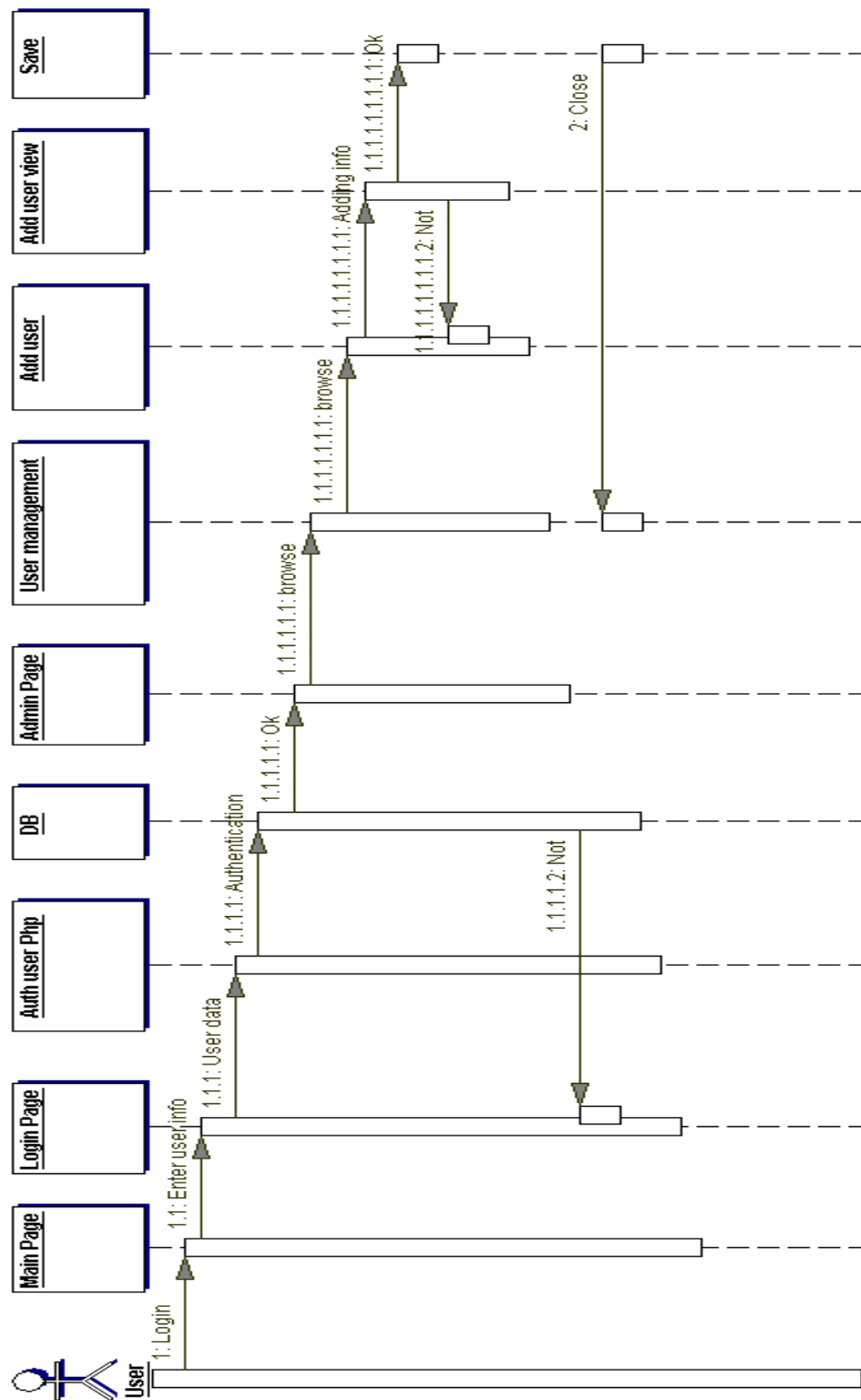


Figure 2-5 Sequence Diagram of adding user

On Figure 2-6 we are able to see more specific view of the system how it reads

data from database or from file.

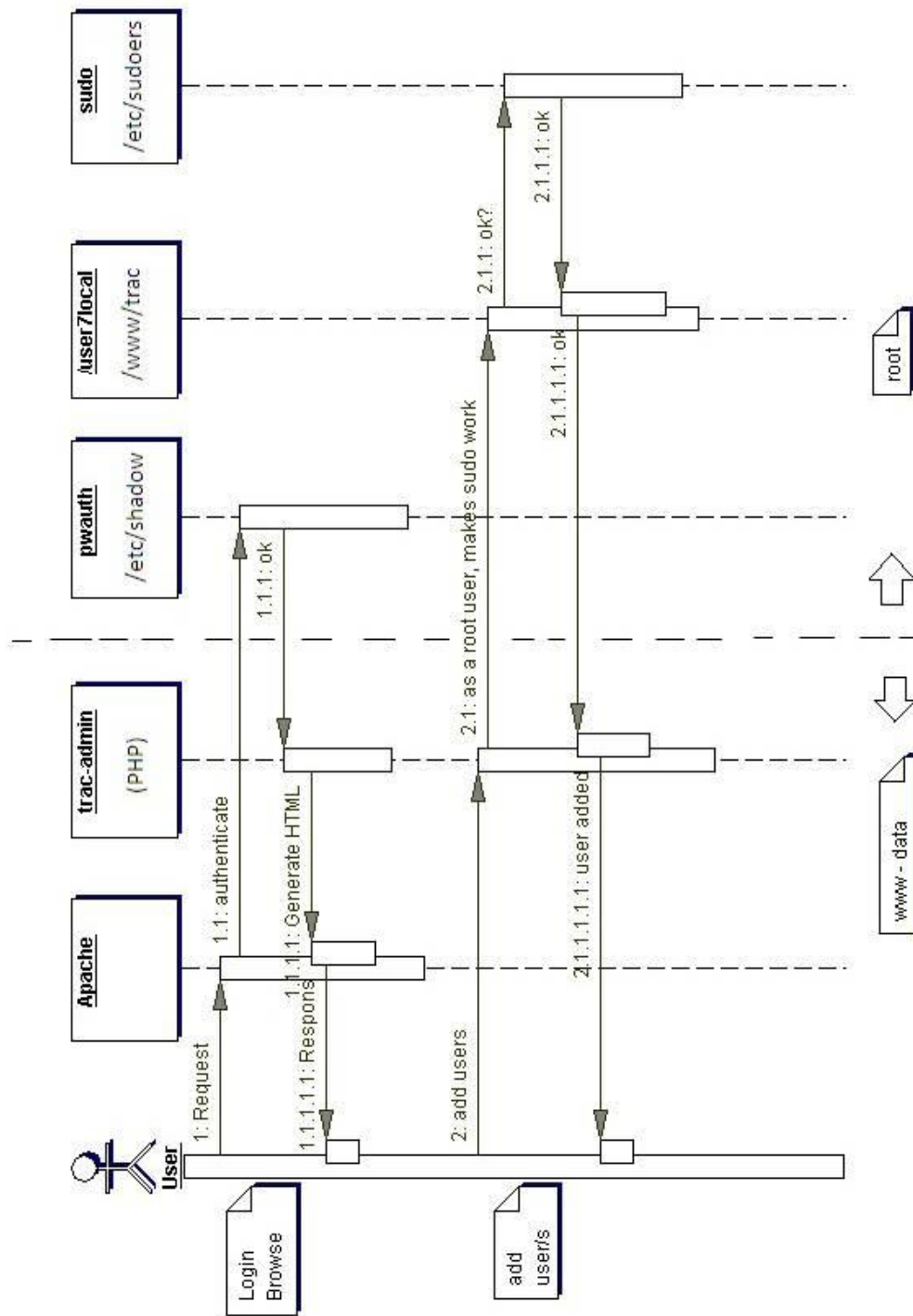


Figure 2-6 Sequence Diagram of adding user (Detailed description)

2.5 Architecture

System architecture is the conceptual design that defines the structure and/or behaviour of a system. An architecture description is a formal description of a system, organized in a way that supports reasoning about the structural properties of the system. It defines the system components or building blocks and provides a plan from which products can be procured, and systems developed, that will work together to implement the overall system. /System Architecture, 11/ Defining the architecture as the set of relationships between the components of a system, that jointly ensures emergent properties of the system as a whole. The architecture of a system is the set of relationships between its components that cause the system to have desired properties, such as desired functionality, behaviour, semantics and quality of service. Architecture is the central problem in web applications because these applications should enable distributed coordination between people and the architecture of these coordination mechanisms evolves by itself as well as is designed by people. As shown on Figure 2-6, we are able to see the architectural diagram of our system that corresponds to our needs.

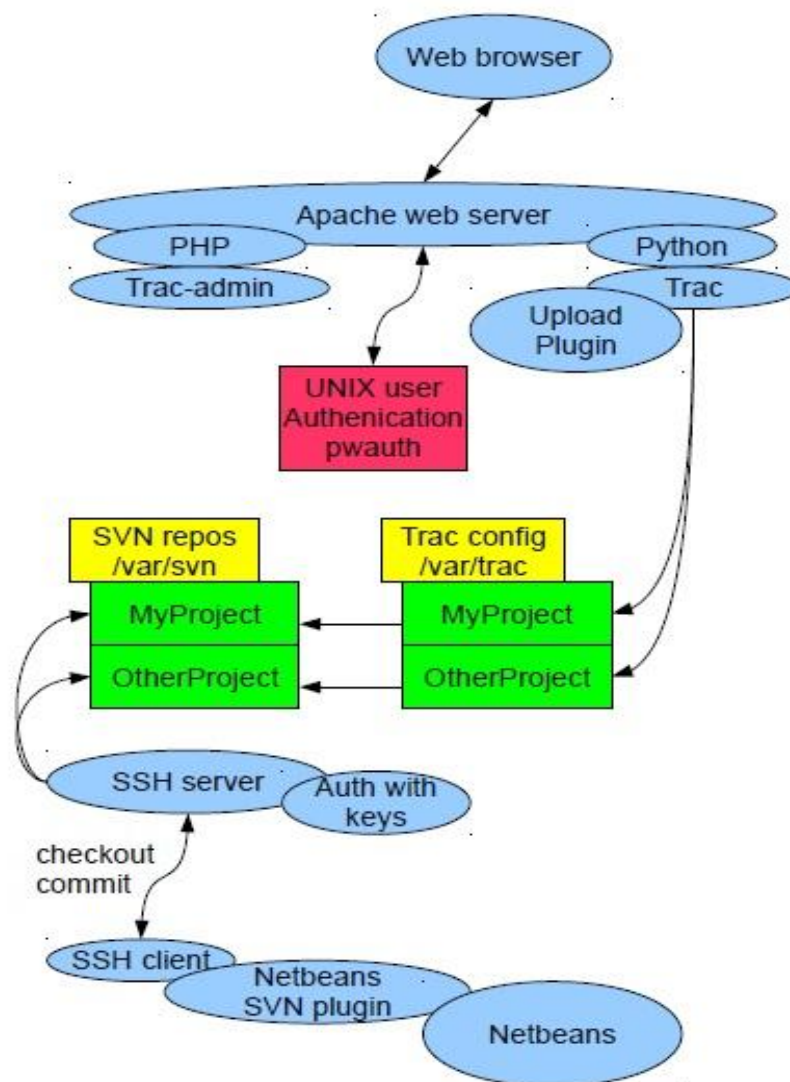


Figure 2-6. Architectural Diagram

While starting to read the chart from top, we start firstly with web browser, that is mainly used as a web application for retrieving and presenting information resources on the web that can be used all over the world. Some browsers can be also used to save information resources to file systems. For next step we connect web browser to Apache web server, what generally is recognized as the world's most popular Web server (HTTP server). Originally designed for UNIX servers, the Apache Web server has been ported to Windows. The Apache Web server provides a full range of Web server features, including CGI, SSL, and virtual

domains. Apache is reliable, free, and relatively easy to conFigure – which corresponds to our requirements. The goal of this project is to provide a secure, efficient and extensible server that provides HTTP services in sync with the current HTTP standards. In our project we use Apache to serve dynamic Web pages, where the content is available in a secure way. To Apache server have added a UNIX authentication pwauth program, that lets authnz-external module, from which can be called out pwauth program, where TRAC can authenticate itself contrary to UNIX users. Pwauth is an authenticator designed to be used with mod_authnz_external and the Apache HTTP Daemon to support reasonably secure web authentication out of the system password database on most versions of UNIX. Particularly - secure authentication against PAM.

Below there is a Figure, that shows component structure of the system, on the

Figure 2-7.

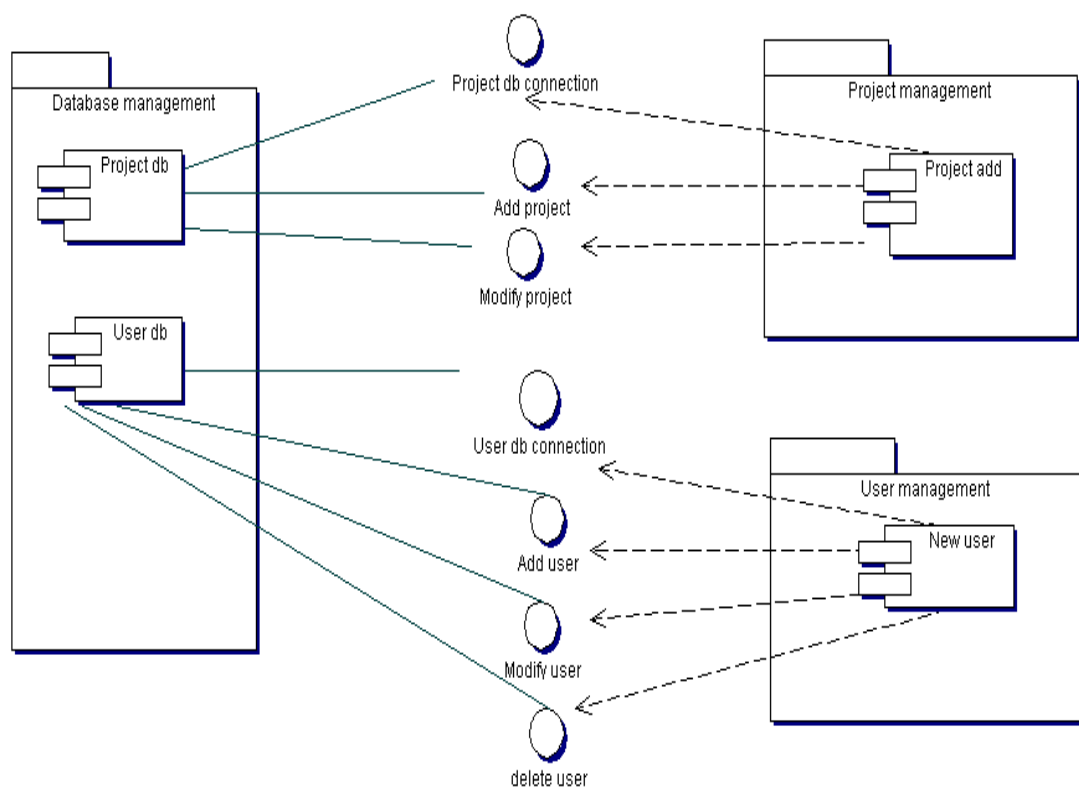


Figure 2-7. Component Diagram

3. IMPLEMENTATION

Project management system is based on these tools:

- TRAC - It was designed to be simple, modular, secure, and reliable. It is easy to understand, and easy to integrate with other programs.
- Apache - is web server software, that playing a key role in the initial growth of the World Wide Web.
- MySQL - is a relational database management system. The program runs as a server providing multi-user access to a number of databases.
- SVN - Subversion (SVN) is a version-control system. Subversion is used to maintain current and historical versions of files such as source code, web pages, and documentation.
- SSH - is a network protocol that allows data to be exchanged using a secure channel between two networked devices. Used on Linux based systems to access shell accounts.
- Netbeans – is a platform framework for Java desktop applications.

TRAC is mainly used for retrieving and presenting information resources on the web that is accessible to all users all over the world. TRAC is connected with Apache web server that provides a full range of Web server features. Apache is reliable, free, and relatively easy to conFigure and used for dynamic Web pages, where the content is available in a secure way. To Apache server will be connected UNIX authentication pwauth program, where TRAC can authenticate itself contrary to UNIX users. To support reasonably secure web authentication out of the system password database on most versions of UNIX. Subversion uses the Apache Web server as one of the two servers it supports for allowing remote access to the repository. TRAC ini will be conFigured with SVN repo that creates special projects under /var/svn. For commits, that has been done from user application and in this case, from Netbeans. System is using SSH server to update commits to web server. It provides secure encrypted

communications between two

no trusted hosts over an insecure network. For accessing to servers by clients, that are properly configured, it is needed a secure shell public key authentication. Netbeans user application, for changing, adding and committing data to web browser, is using SVN plug-in. Netbeans works with the SVN command line client and provides useful tools to manage the SVN operation on the source code. Basically, SVN is a server/client system.

3.1 Functional requirements

To improve productivity of developing web applications for customers and to make easier our everyday's working life, the company has set up requirements for making an internal system that is called web based project management system. To consume necessary time and for keeping all troubles less than good project management system helps company out. The project management system will help to arrange documentation management inside company, by keeping documents in one place. Also, the most important it helps to keep track on new projects that are implemented and for those that are under implementation for customers and also to keep eye on errors or mistakes that occur during our work process for some projects. The idea come from everyday use and working life, the aim was to make an inside system for the company. The system is for helping workers (namely programmers, project managers, developers) to deal with some specific project and its errors occurred.

The system is web-based; there is a possibility to add documents/specifications for the specific project. Documentation can consists of graphs, database diagrams and sequence diagrams. The most important part is that, the system has the issue tracking system, system where can be added comments, bugs and other related questions for specific project. Besides that, the system has all documents related to that specific project, but it also has the code, right paths, folders and links. Issue tracking inside system, helps programmers to keep their eye on the process, find and search for bugs related to some specific issue or project.

Additionally the

system has a functionality to add new members, new projects and ability to delete

or lock / unlock users, who later on don't have access to the system any more.

The system will utilize:

- MySQL database
- Python
- PHP
- Version control system (Subversion)
- SVN plug-in
- SSH (Secure Shell) authorized key
- Extensible Hypertext Markup Language 1.0
- Cascading Style Sheets (CSS) 2.0 Compliant
- User application (Netbeans)

System utilize MySQL database to store information. PYTHON programming language is used as a dynamic programming language, which is widely used in application domains. PHP is widely-used general –purpose scripting language. For next, our system utilize Version control system, the basic functionality of any

/Version control system is to keep track of the changing states of files over time and merge contributions of multiple developers. They support this, for the most part, by storing a history of changes made over time by different people. In this way, it is possible to roll back those changes and see what the files looked like before they were applied. Additionally, a version control system will provide facilities for merging the changes, using one or more methods ranging from file locking to automatic integration of conflicted changes. (William Nagel, 2005:4)

Configurations of the system:

To start with application, it is first needed to configure, install and run all packages. While running it in LINUX op system, it is needed to install these packages on Figure 3-1:

```
sudo apt-get install
\ TRAC \
python-mysqldb \
python-psycopg2
apache2 \ libapache2-
mod-wsgi \ libapache2-
mod-python \ libapache2
-mod-php5 \
libapache2-mod-authnz-
external \ libapache2-mod-
authz-unixgroup \ pwauth \
php5-mysql
\ php5-svn \
mysql-
server
```

Figure 3-1 Packages that needs to be install

In Linux `sudo apt-get` is used to install, applications. In this case, it is needed to install TRAC – that is an enhanced wiki and issue tracking system for software development projects /TRAC, 12/. TRAC uses a minimalistic approach to web- based software project management. For next, `python-mysqldb` is installed. MySQLdb is a thread-compatible interface to the popular MySQL database server that provides the Python database API. MySQLdb is a thin PYTHON wrapper around `_mysql` which makes it compatible with the Python DB API interface. PYTHON-psycopg2 `apache2` is a PYTHON and Apache configuration. `Mod_python` is an Apache module that embeds the Python interpreter within the server. With `mod_python` can write web-based applications in Python that will run many times faster and will have access to advanced features such as ability to retain database connections and other data between hits and access to Apache internals. Quite simply - it is integration of PYTHON and Apache. Apache is a sort of a Swiss knife of web serving, especially the upcoming 2.0 version, which does not limit itself to HTTP but can serve any protocol

for which there exists a

module. Mod_python aims to provide direct access to the riches of this functionality for PYTHON developers.

Libapache2-mod-wsgi: The mod_wsgi adapter is an Apache module that provides a standard interface between web server software and web applications written in PYTHON. Compliant interface for hosting PYTHON based web applications within Apache. The adapter provides significantly better performance than using existing WSGI adapters for mod_python or CGI. /Ubuntu packages, 7/

Libapache2-mod-python: The mod_python module supports web applications written in Python. Because the parser is embedded in the server as an Apache module, it will run much faster than traditional CGI.

Libapache2-mod-php5: This package provides the PHP5 module for the Apache 2 web server. This package works only with Apache's prefork MPM, as it is not compiled thread-safe.

Libapache2-mod-authnz-external: Mod_Auth_External can be used to quickly construct secure, reliable authentication systems. It can also be miss-used to quickly open gaping holes in your security.

Libapache2-mod-authz-unixgroup: Mod_Authz_Unixgroup is a UNIX group access control module for Apache 2.1 and later. If users authenticate with real UNIX login ID over the net, using something like my mod_authnz_external / pwauth combination, and it is needed to access control based on UNIX group membership, then mod_authz_unixgroup.

Pwauth: Pwauth is an authenticator designed to be used with mod_auth_external or mod_authnz_external and the Apache HTTP Daemon to support reasonably secure web authentication out of the system password database on most versions of UNIX. Particularly - secure authentication against PAM.

Php5-mysql: This package provides modules for MySQL database connections directly from PHP scripts. It includes the generic "mysql" module which can be used to connect to all versions of MySQL, an improved "mysql" module for MySQL version 4.1 or later, and the pdo_mysql module for use with the PHP Data Object extension. PHP5 is an HTML-embedded scripting language. The goal of the language is to allow web developers to write dynamically generated pages quickly.

Php5-svn: These bindings provide a method for manipulating Subversion working copies or repositories with PHP.

Mysql-server: relational database management system.

Next, we let authnz-external module, from which we could call out pwauth program, that TRAC could authentic itself from UNIX users on Figure 3-2.

```
sudo a2enmod
authnz_external sudo
a2enmod authz_unixgroup
```

Figure 3-2 TRAC authentications against UNIX users

With SVN, we have to greater a SVN group on Figure 3-3:

```
sudo groupadd
svn
```

Figure 3-3 Creating SVN group

Add SVN users to group, its needed to do in this way: like on Figure 3-4:

```
sudo gpasswd -a anne
svn
```

Figure 3-4 adding SVN users to

group

While there is a need to add new users (For those users who can access through

SSH in), have to follow the steps, Figure 3
-5:

```
sudo useradd
username sudo
passwd username
```

Figure 3-5 Adding new users through LINUX command prompt

Add Apache2 to SVN users group. That's for with TRAC-admin created SVN repositories, can be set „SVN“, on Figure 3-6:

```
sudo gpasswd -a www-data svn
```

Figure 3-6 Setting repository name

It is needed to make directories, on Figure 3-7:

```
sudo mkdir /var/svn
sudo chown www-data:svn -R
/var/svn sudo chmod 775 -R
/var/svn
sudo mkdir /var/TRAC
sudo chown www-data:www-data -R /var/TRAC
sudo chmod 755 -R /var/TRAC
```

Figure 3-7 making specific directories for system use

Tooling Apache: Open /etc/apache2/sites-enabled/TRAC, on Figure 3-8:

```
sudo nano /etc/apache2/sites-enabled/TRAC
```

Figure 3-8 Tooling Apache

Adding Apache conf to system, on Figure 3-9:

```
AddExternalAuth pwauth /usr/sbin/pwauth
SetExternalAuthMethod pwauth pipe
```

```
AddHandler mod_python .py
PythonHandler
mod_python.psp
```

PythonDebug On

*<Location /TRAC-
admin> AuthType Basic
AuthName „SVN
admin“*

```

AuthBasicProvider
external AuthExternal
pwauth AuthzUnixgroup
on
Require valid-user
Require group admin
</Location>

<Location /TRAC>
SetHandler
mod_python
PythonHandler TRAC.web.modpython_frontend
PythonInterpreter main
PythonOption TRACEnvParentDir
/var/TRAC PythonOption TRACUriRoot
/TRAC/
SetEnv PYTHON_EGG_CACHE /tmp
</Location>

<LocationMatch
/TRAC/[[[:alnum:]]+]/login> AuthType
Basic
AuthName TRAC
login" AuthBasicProvider
external AuthExternal
pwauth AuthzUnixgroup
on
Require valid-user
Require group svn
</LocationMatch
>

```

Figure 3-9 Apache
conf

This will make clear, that under <http://localhost/TRAC> will locate repository list, it is possible to do this http://localhost/TRAC/**STRING**/login authenticate users. Users who belong to "SVN" group can login only. This authentication goes in return to UNIX users (those, who can login with SSH), not against to SQL database table.

To install UNIX authentication, on Figure 3-10:

```

sudo apt-get
install \

```

*libapache2-mod-authn-
external *
*libapache2-mod-authn-
unixgroup *

Figure 3-10 UNIX
authentications

Next, authnz-external module, from which could call out pwauth program, that

TRAC could authentic itself from UNIX users, on Figure 3-11.

```
sudo a2enmod
authnz_external sudo
a2enmod authz_unixgroup
pwauth \
```

Figure 3-11 authnz-external modes

In SSH:

SSH login with no password and with a key:

With this command, it will create generated key files. In this case passphrase is empty!

```
ssh-keygen -t rsa -f ~/.ssh/id_rsa -
N "" Now I have in my home
directory:
```

~/.ssh/idrsa – This is a private key

~/.ssh/idrsa.pub – This is a public key, with this one, I can be authenticated.

If we would like to log in to the server without a password, we have to add a public key to the server in this way, on Figure 3-12:

```
ssh-copy-id kasutaja@serverihostname
```

Figure 3-12 Adding public key to server

After this command line, it should not ask any password, while log in to SSH, the system will authenticate you, in this way: ~/.ssh/idrsa, on Figure 3-13

```
ssh username@serverihostname
```

Figure 3-13 System authenticates the

user.

But, we have to put attention to ssh-copy-id that will add public key in server: ~/.ssh/authorized_keys, keeping its old content!

Under common.php is added a function, that will authenticate user and its group and user has to belong to im_admin, to modify or add changes, on Figure 3-14:

```
function get_auth_user_groups() { //authenticated user groups
    $invalid_users = array("root","anonymous","nobody","www-data");
    $authuser = $_SERVER['PHP_AUTH_USER']; //server authenticates
    user if (in_array($authuser, $invalid_users)) { //if user is invalid
        return array();
    } else { //user will be authenticated and added to group
        $users = unix_users();
        return $users[$authuser][groups];
    }
}

$authuser = $_SERVER['PHP_AUTH_USER']; //user authentication
$im_admin = in_array("admin", get_auth_user_groups()); //if user is
admin function unix_users() {
    $users = array();
    $valid_shells = array("/bin/sh", "/bin/bash"); //finds folder in shell
    $lines = file('/etc/passwd'); //finds file in shell, to
    log in foreach ($lines as $line_num => $line) {
        list($raw_username, $x, $uid, $gid, $raw_contact, $home, $shell) =
        explode(":", $line); //displays the fields while adding new contact
        $username = trim($raw_username); //takes the username
        $fullname = trim($raw_contact); //takes contact name
        @list($fullname, $location, $phone, $email) = explode(" ",
        trim($raw_contact)); //lists data to shell file
        # User ID has to be 1000-65xxx, for example Apache runs as
        www-data (uid=33)
        # The shell has to be /bin/sh or /bin/bash in order to log in
        via
        SSH
        //
        if (($uid >= 1000) && ($uid < 65000) && in_array($shell,
        $valid_shells)) {
            if (($uid >= 1000) && ($uid < 65000)) {
                $users[$username] = array();
                $users[$username]["uid"] = (int)$uid;
                $users[$username]["gid"] = (int)$gid;
```



```

        $users[$username]["fullname"] = $fullname;
        $users[$username]["location"] = $location;
        $users[$username]["phone"] = $phone;
        $users[$username]["email"] = $email;
        $users[$username]["home"] = trim($home);
        $users[$username]["shell"] = trim($shell);
        $users[$username]["groups"] = array();
        $users[$username]["locked"] = !in_array(trim($shell),
$valid_shells);    }

    }

    $lines = file('/etc/group');//adding a group to user,
    later on foreach ($lines as $line_num => $line) {
        list($group, $x, $gid, $raw_users) = explode(":", $line );
        if (trim($raw_users) != "")
            foreach (explode(",", trim($raw_users)) as $username)
                if (array_key_exists($username, $users))
                    $users[$username]["groups"][] =
$group;

    }

    return $users;
}

```

Figure 3-14 Code that authenticates the user

Web Server is divided into two interfaces; one is TRAC and another one TRAC- admin. TRAC is used for usual workers, workers who have access to all projects and who have the ability to update, download files, delete added files through web application, to see all changes committed to web-service. TRAC admin has a different view from a usual worker. Admin has the ability to see all projects that are locked and unlocked and also possibility to make them hide or shown by TRAC user, admin can easily add new users and add users to special group that user could have the right to access or

update commits to web browser. Admin definitely has the right to delete⁴³
user or change password of a user. What's

interesting is that TRAC-admin has the right to lock or unlock a user, which means that user can or can't access to the site, which is meant for users who have finished contract with a company.

Before we could start to use TRAC and TRAC-admin, it is needed to install also

TRAC, on Figure 3-15:

```
sudo apt-get install
\ TRAC \
```

Figure 3-15 Installing
TRAC

For Web development I used a PHP scripting language and PYTHON programming language that has very clear and readable syntax, intuitive object oriented to produce dynamic web pages. For Subversions, who use the Apache Web server as one of the two servers it supports for allowing remote access to the repository. For next step we have to make sure that Apache has read/write access to any repositories that I want it to serve. Once basic location block is conFIGured, there can be created an access file and define some authorization rules in it. The syntax of the access file is the same familiar one used by svnserve.conf and the runtime configuration files. Lines that start with a hash (#) are ignored. In its simplest form, each section names a repository and path within it, and the authenticated usernames are the option names within each section. The value of each option describes the user's level of access to the repository path: either r (read-only) or rw (read-write). If the user is not mentioned at all, no access is allowed. There is also another possibility; the access file also allows defining whole groups of users, much like the UNIX /etc/group file: Add Apache to SVN users group.

TRAC-admin created SVN repositories can be set "SVN", on Figure 3-16:

```
sudo gpasswd -a www-data
svn
```

Figure 3-16 Group SVN has been
created

It is needed to make directories, on Figure 3-17, below:

```
sudo mkdir /var/svn
sudo chown www-data:svn -R
/var/svn sudo chmod 775 -R
/var/svn
sudo mkdir /var/TRAC
sudo chown www-data:www-data -R /var/TRAC
sudo chmod 755 -R /var/TRAC
```

Figure 3-17 Read / Write rights to directories

Under config.php file, it is possible to see groups that have been made, on Figure

3-18:

```
$config["apache_user"] = "www-data";
$config["group_svn"] = "svn";
$config["group_svn_admin"] = "admin";
```

Figure 3-18 Groups and special rights on specified groups

Under svngroup.php it is able to find svn users, who belong to specific group, on

Figure 3-19:

```
function svn_users() {
    $lines = file('/etc/group');
    foreach ($lines as $line_num => $line) {
        list($group, $x, $gid, $raw_users) = explode( ":", $line );
        if ($group == "svn")
            if (trim($raw_users) == "")
                return array();
        else
            return explode( ":", trim($raw_users));
    }
    return NULL;
}
```

Figure 3-19 for finding a group

TRAC and TRAC - admin has two different user interfaces for different users with different rights, who log in to the system. Under TRAC, user can see list of projects and under TRAC-admin user, who has admin rights is able to see all projects that belongs to this user and whole list of projects. Projects that are made for web interface are saved from Apache Web server, and then conFigured by TRAC conf, folder on /var /TRAC. The TRAC.ini configuration file is writable by the web server, as TRAC currently relies on the possibility to trigger a complete environment reload to flush it caches. TRAC ini will be conFigured with SVN repo that creates special projects under/var/svn – repository on local machine. For commits, that has been done from user application and in our case, from Netbeans. SSH server has been used to update commits to web server, for later on reading on Web Browser. SSH is a program for logging into a remote machine and for executing commands on a remote machine. It provides secure encrypted communications between two entrusted hosts over an insecure network. SSH connects and logs into the specified hostname (with optional user name). The user must prove his/her identity to the remote machine using one of several methods depending on the protocol version used. For accessing to server by clients, there is need for a secure shell public key authentication. The remote host can authenticate itself using either traditional public-key authentication or certificate authentication. At the beginning of the connection the server sends its public host key to the client for validation. If certificate authentication is used the public key is included in the certificate the server sends to the client. The connection between Client and Server on SSH is shown on Figure 3-21, below.

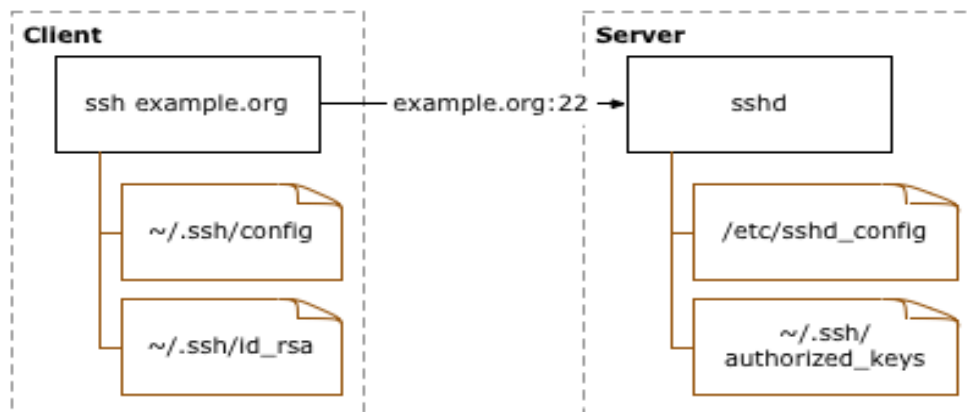


Figure 3-21 SSH public key authentication
(Source:

<http://sial.org/howto/openssh/publickey-auth/>)

Definition of the Figure:

- **Client:** the system one types directly on, such as a laptop or desktop system.
- **Server:** anything connected to from the client. This includes other servers accessed through the first server connected to. /Open SSH public key authentication, 8/

A key pair must be generated on the client system. The public portion of this key pair will reside on the servers being connected to, while the private portion needs to remain on a secure local area of the client system, by default in ~/.ssh/id_rsa. The public portion of the key pair must be copied to any servers that will be accessed by the client. The public key information to be copied should be located in the ~/.ssh/id_rsa.pub file on the client. Assuming that all of the servers use SSH instead of a different SSH implementation, the public key data must be appended into the ~/.ssh/authorized_keys file on the servers. SSH agent in client to server connection forwarding will not be a trouble. We enable the SSH agent forwarding, if the server connected to, user logs into other servers. On the proxy system a private key must be available, because it is recognized by the server that is being connected to it, while SSH agent forwarding is disabled. Setting ForwardAgent in an SSH config file, such as

~/.ssh/config, or to use the same configuration, if its

need to enable forwarding, to SSH when connecting. User-specific configuration file will be overwritten by the command line arguments, which can override the global `ssh_config` configuration, file. To conclude, our netbeans user application, for changing, adding and committing data to web browser, is using SVN Plug-in. Netbeans works with the SVN command line client and provides useful tools to manage the SVN operation on the source code. Basically, SVN is a server/client system. The server that contains the repository (source code of our application). The clients get the source code (Check out) or only the new files (Update) from the server. The opposite operation is called commit, and consists in sending new files to the server. Every commit generates a new repository revision; SVN maintains the content of the files in every revision with the aim not to lose code and the possibility to undo code changes. Figure 3-22, below.

Repositories are created in way, this will create SVN repository, on Figure 3-23:

```
svn_repos_create($svn_path);
adjust_svn_repo_permissions($svn_path, "svn", 0775);
```

Figure 3-23. Creating SVN repository

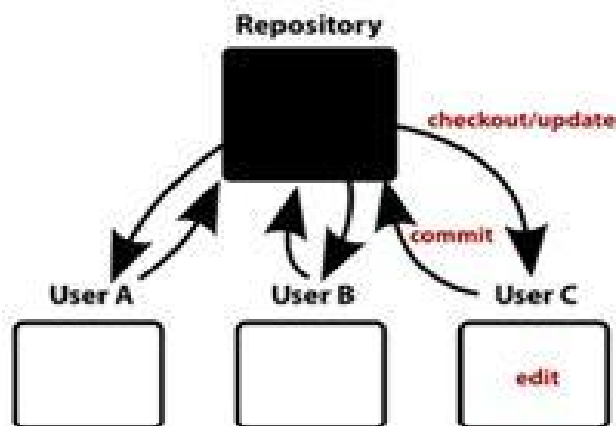


Figure 3-22, SVN work.

After Netbeans application has connected with SSH client and all authentications have been successfully, then we could easily commit our changes to Web browser. To conclude, system starts from user application in our case Netbeans while committing changes to web browser, we connect our application that has SVN plug-in to SSH Server and authenticate the user, we save our project to SVN repository. Next step TRAC conFigures changes to Web browser and our commits will be seen by every user on web.

4. TESTING

The goal of testing was to demonstrate that the program under control contains bugs. Testing must not be confused with debugging, which is the process of detecting and reducing the number of existing errors. Testing can never prove that a code is error-, but rather verify that errors exist. Therefore need to consider the fact that the error might come from the test itself, while the tested code might be correct. Have to know what the results of the test will show before it has actually been performed. The one who is responsible for doing the testing has to be able to define what the outcome should be, if not, this will lead to bugs in either the program or the test or in both the program and the test. The good thing about system testing is that it can be carried out without any prior knowledge about the program design and can thus be performed by „outsiders“. To maintain the quality of a system, it is definitely needed to conFigure a system testing. Detecting and fixing the errors in a system is known as one of the main objectives behind testing of a system in a development cycle. Here is described a set of test cases; some of them have been shown below:

1. For checking the conjunction in the system that is developed, couples of users are connected to the Application simultaneously through the Internet browser. Different parts of system, like TRAC, project management and user management have been monitored simultaneously on these client machines. To conclude, the testing showed that system components could be monitored conjunctionally from multiple clients through Internet browser.
2. It has also been noticed that data regarding system modifications are easily available with for all users. This affirms that the data recording feature of the developed system is working correctly.
3. The SVN SSH connection has been tested from user application to commit code or make changes on code and later on to be seen on web browser. Commits and changes are nicely illustrated with differences made on the code on the web browser. These changes are seen for all users.

4. Creating a new project and its administration, description and group.
To add new project from admin panel to system, then we had to see, if the projects have made their own SVN repository. Where later on all updated files and commits have been added.
5. Definitely, user management has been tested, while adding users, with different rights – belong to different groups.

To be more specific, testing system started with an admin adding new user with

Different rights, belonging to different groups, on Figure 3-24.

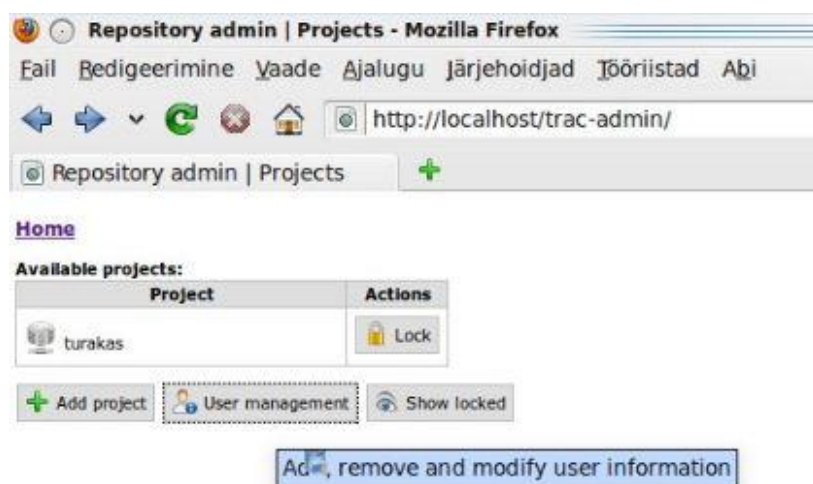


Figure 3-24: User management

Admin has to fill up all fields, as shown on Figure 3.25: user management view. All the fields are needed to fill, because it helps to get an accurate data for every user. Admin fills up username that is users name used for log-in to system and password which should consist of letters and numbers. Password is protected, which avoids unexpected people to hack in to system and read others passwords. It would be nice to also fill up fields like: user's full name, Location, Phone and E-mail. Choosing a group is a required field which admin must choose.

Repository admin: User Management - Mozilla Firefox

Fail Bedigeerimine Vaade Ajalugu Järjehoidjad Tööriistad Abi

http://localhost/trac-admin/users-add.php

Home | User Management | A...

Home | User Management | Add User

Username (only A-Za-z0-9): Teacher

Password: ●●●●●●●●

Confirm: ●●●●●●●●

Full name: Mr Teacher

Location: Room 314

Phone: 06787686

Misc: teacher@zoo.org

Groups: ☐ svn ☐ admin ☒ users

Add

Figure 3-25 Add user management

Next step, admin logs out from the system and for testing purpose with new user account trying to log in to the system, which has been made previously, user will see a login authentication as shown on Figure 3-26, below.

Autentimine on vajalik

Veebileht aadressil http://localhost nõuab kasutajanime ja parooli. Veebileht ütleb: "SVN admin"

Kasutajanimi: Teacher

Parool: ●●●●●●●●

Sobib Loobu

Figure 3-26 Authentications in login

After login, user is able to see a user's view, as shown on Figure 3-27. Second concern was at the beginning related with view. Namely we have two different types of users, ones are just users and to second group belongs administrators. The

difference here is the accessibility to everything. Views are different, by user and its rights. Namely users are able to see only list of project and they are not able to add any project or user to system. It is important for security reasons.

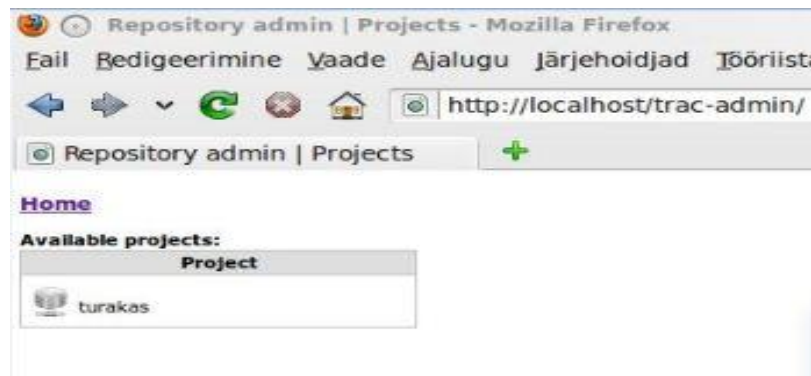


Figure 3-27 Users view

To conclude, all users worked correctly with no mistakes or errors occurred. While have been logged-in to system with different users. But it was not enough, had to check if users have right access and rights, depending on users group that admin has added to user.

Administrators see, different view, as said before, namely: admin has rights to add

projects, users and locked/unlocked projects view, as shown on Figure 3-28.

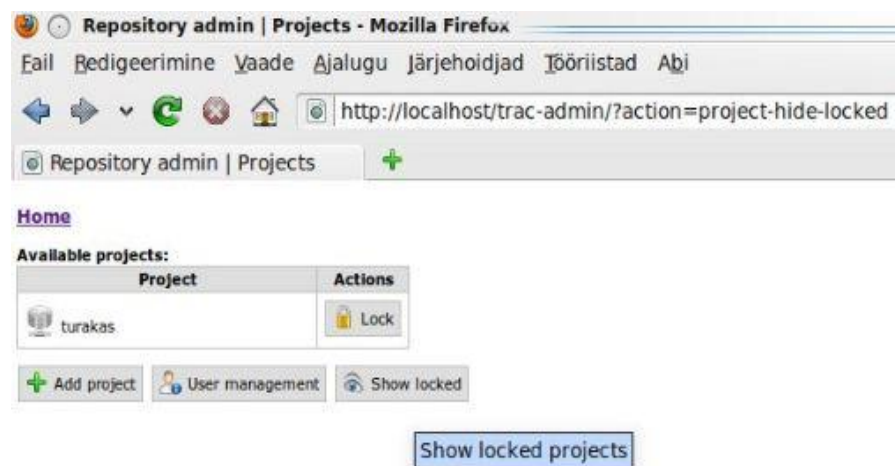


Figure 3-28 Administrator view

“Show locked” is the functionality that helps user easily handle projects in system. Namely users would not like to see all projects in one list, because it is annoying to search from long list some project. But while admin will click “Show locked”, then administrator can easily see all locked projects on the list through simple click. In a mysql database, there can be seen the difference, when project is

locked and when project is unlocked as shown on Figures 3-29 and 3-30.



































			username	action
<input type="checkbox"/>			anonymous	LOG_VIEW
<input type="checkbox"/>			anonymous	FILE_VIEW
<input type="checkbox"/>			anonymous	WIKI_VIEW
<input type="checkbox"/>			authenticated	WIKI_CREATE
<input type="checkbox"/>			authenticated	WIKI_MODIFY
<input type="checkbox"/>			anonymous	SEARCH_VIEW
<input type="checkbox"/>			anonymous	REPORT_VIEW
<input type="checkbox"/>			anonymous	REPORT_SQL_VIEW
<input type="checkbox"/>			anonymous	TICKET_VIEW
<input type="checkbox"/>			authenticated	TICKET_CREATE
<input type="checkbox"/>			authenticated	TICKET_MODIFY
<input type="checkbox"/>			anonymous	BROWSER_VIEW
<input type="checkbox"/>			anonymous	TIMELINE_VIEW
<input type="checkbox"/>			anonymous	CHANGESSET_VIEW
<input type="checkbox"/>			anonymous	ROADMAP_VIEW
<input type="checkbox"/>			anonymous	MILESTONE_VIEW
<input type="checkbox"/>			anne	TRAC_ADMIN

Figure 3-29 shows unlocked projects

On the Figure Locked projects, can be seen that users are not able to create or modify any project information either upload or delete files on the project. While project is locked, it means that we are just able to read the info on the Project.



































			username	action
<input type="checkbox"/>			anonymous	LOG_VIEW
<input type="checkbox"/>			anonymous	FILE_VIEW
<input type="checkbox"/>			anonymous	WIKI_VIEW
<input type="checkbox"/>			authenticated	DISABLED_WIKI_CREATE
<input type="checkbox"/>			authenticated	DISABLED_WIKI_MODIFY
<input type="checkbox"/>			anonymous	SEARCH_VIEW
<input type="checkbox"/>			anonymous	REPORT_VIEW
<input type="checkbox"/>			anonymous	REPORT_SQL_VIEW
<input type="checkbox"/>			anonymous	TICKET_VIEW
<input type="checkbox"/>			authenticated	DISABLED_TICKET_CREATE
<input type="checkbox"/>			authenticated	DISABLED_TICKET_MODIFY
<input type="checkbox"/>			anonymous	BROWSER_VIEW
<input type="checkbox"/>			anonymous	TIMELINE_VIEW
<input type="checkbox"/>			anonymous	CHANGESSET_VIEW
<input type="checkbox"/>			anonymous	ROADMAP_VIEW
<input type="checkbox"/>			anonymous	MILESTONE_VIEW
<input type="checkbox"/>			anne	DISABLED_TRAC_ADMIN

Figure 3-30 Shows projects that are locked

To conclude, all testing related with hiding projects that have been locked, and later on trying to access to those projects and change information, what was added all the information was working correctly. No errors occurred during procedure. The interface of the project management system is also flexible. In order to reach a large number of users, the interface of the system is web based, running inside a standard web browser. This allows the users to work from own computer whenever it is needed, and it does not imply special software to be installed. Whatever the user updates, the system keeps track of projects and their revisions, tests, sessions and results. This is done via apache server that is connected through SSH to SVN. Flexibility is again built in the users and project management.

Install

Install the Azure Static Web Apps CLI
using **yarn** or **npm**.

Init

Setup your project
using **swa init**

Login

Authenticate with Azure services
using **swa login**

Start

Preview and debug your static web app
using **swa start**.

Deploy SWA (cloud)

Build/deploy to multiple environments
using **swa deploy**

Configure

Setup **environment variables**
and **config settings**.

<https://github.com/equinor/az-static-web-app-docs-template/blob/main/README.md#host-documentation-with-granular-access-control-using-azure-static-web-app>

This template repository is designed as a starting point for those who want to host their documentation in a way that provides:

1. Fast, simple and downright gorgeous static web sites using documentation builders like MkDocs, Sphinx, or the static site generator of your choice.
2. CI/CD pipeline that automatically builds and re-deploys your documentation whenever you commit changes to it.
3. Lets you to keep the documentation "physically" close to your code, since you can have both the code and documentation in the same repo.
4. Makes the documentation easily accessible through a (customizable) URL defined in Azure.
5. Authentication using Azure Active Directory, allowing e.g. only members of your organization to access the documentation.
6. Granular access control, where you can limit access for parts of the site to members granted specific roles.

Check out the live demo [here](#).

The tutorial is split into several sections, adding on more and more of the features mentioned above. If you or your organization do not require all these features, feel free to follow along for as long as it makes sense for you. **PS:** Several chapter ends with an **output**-sections, which lists all the keys, secrets, values etc. that you should have written down before proceeding to the next step.

Abbreviation

Abbreviation	Description
AAD	Azure Active Directory
SWA	Static Web App

Step 1: Prerequisites

This guide uses poetry to manage dependencies and virtual environments, but any package manager should work with some configuration. Please note that you will need the following to complete this setup:

1. A GitHub-account
2. An Azure account and subscription

Step 2: Set up the Azure Static Web App

Goal: After this section, you will be hosting a demo landing page for our documentation web page.

1. Fork this repository or click the template button above.



2. In [Azure Portal](#), navigate to [Azure Static Web App in Azure](#) and create a new resource. Go through the setup wizard and connect it to your forked repository, as follows:
 - i. Select a resource group. If you haven't already created a Resource Group, see the point below.
 - ii. (Optional) To create a new resource group, click the "Create new"-button underneath the drop-down for selecting resource groups. Alternatively, from the [Azure Portal](#), go to "Resource Group" and click "Create". Go through the setup wizard to create your resource group.
 - iii. Give your SWA Resource a descriptive name.
 - iv. Hosting plan: If you want to enable custom authentication and follow along for the last steps in this tutorial, you must choose the **Standard** plan type.
 - v. Select a desired region for hosting your resource. If you are part of an organization, they might have a policy for which you should choose.
 - vi. Press the button "Sign in with GitHub" and follow the instructions in the pop-up window.
3. Connect to GitHub (image below):
 - i. After completing the GitHub login, use the "organization", "repository" and "branch" drop-down menus to select the repository you forked. In this tutorial, we connect the web app to the main-branch. If you want to keep a separate branch for built documentation, feel free to do so.
Important note: Do not connect to this template repository directly. Connect to your own fork of this template.
 - ii. Select "Custom" in the "Build Presets"-dropdown and fill in the following fields:
 - App location: /
 - Api location: /api
 - Output location: /docs/build
4. Finish the setup wizard by clicking the "Review + create"-button and then the "Create"-button.
5. Once completed, go to your newly created resource. Your SWA has a **deployment token** which (see image below). Please write this down.
6. Your SWA also has an **URL** which we will refer to as YOUR_SITE_URL. Please write this down as well.

Under-the-hood: What just happened?

1. First, Azure automatically registers the **deployment token** as a secret in your repository, which you can view by visiting your **GitHub Repo > Settings**



- > **Secrets > Actions**. This gives the Web App access to your repo.
- 2. Second, Azure commits a workflow-file to your repo which contains the necessary GitHub Action for deploying the website located in /docs/build. This is triggered automatically when anything is pushed to main, and will handle the connection and deployment of content to your SWA.
 - o You can view triggered actions by clicking on "Actions" in your GitHub repo. (**NB!** The deploy-site.yml-action will fail at this point since we haven't yet created the GitHub secret that it expects. This will be fixed later).
 - o You can view the newly created GitHub Action file in the folder .\github\workflows.
- 3. After the GitHub Action has completed, you can visit the **URL** found to your SWA. If deployed successfully, you should be welcomed by the following demo page:

Feel free to click the "documentation examples" button and browse the two examples. The other buttons will not work properly yet.

Note: The text printout at the bottom of the page is only for debugging purposes, which will be useful in the sections related to authentication and role management..

Output

- deployment token
- YOUR_SITE_URL

Step 3: Automate the compilation of docs

***Goal:** After this section, the documentation examples will be built and deployed, making them visible on your SWA. Further, a Github Action workflow will ensure that the documentation automatically re-builds and re-deploys itself when new changes are committed to the branch.*

NB! If you are working on integrating this template into an existing repository, follow along until you have completed the sub-chapter "Under-the-hood: What just happened?", then go to the chapter [(Optional) Implement this solution into an existing repo]([Optional) Implement this solution into an existing repo]).

SWAs does not support building non-Javascript projects and therefore you have to compile the docs before it gets deployed. Thankfully, [Github Actions](#) allows us to build and compile the HTML files as part of the action that deploys them to the web app. Doing it this way allows us to delete the build files, so that they don't clutter our repository. Neat! Furthermore, it ensures that we do not have to run any manual steps to update the documentation. Double-neat! This also means that we can use almost any type of documentation compiler as long as it is possible to install on the Github Action build server and it can compile to HTML. Triple-neat! For this repository, sphinx and mkdoks is used, but it can modified to work with your preferred build tool.

To build the docs using sphinx and mkdoks, we need to add some custom build steps to the workflow file that the SWA created during its setup. Rather than editing the newly created workflow file, we can use the file that comes with this repo.

NB! In the next steps we will delete the github action that was created by our SWA, and replace it with the `deploy-site.yml` that already exist in the template repo. However, this will break the link in your SWA's overview page which link to the github action it created itself. Currently, there is no way of updating this link. It will not affect operation, but if you would like to keep this link intact, you should copy the content from `deploy-site.yml` into the newly created yml-file, and delete the `deploy-site.yml`-file instead.

1. Navigate to your GitHub Repo > Settings > Secrets > Actions > Repository secrets.
2. There should only be one secret called `AZURE_STATIC_WEB_APPS_API_TOKEN_<URL-name>` under "Actions secrets". Delete this secret by pressing "Remove".
3. Create a new repository secret by clicking "New repository secret" and use the following values:
 - o Name: `"DEPLOYMENT_TOKEN"`
 - o Value: deployment token (output from previous chapter)
4. Locate your workflow files in `.\.github\workflow`. There should be three files there, where you should delete the first one:
 - i. `azure-static-web-apps-<YOUR_SITE_URL>.yml`: *Delete this and commit changes. This file was created when you created the SWA*
 - ii. `deploy-site.yml`: *This will replace the file above*
 - iii. `lint-and-format.yml`: *This file is optional, feel free to delete it. It runs some linting and syntax checks on your code*

Under-the-hood: What just happened?

The Github workflow committed by the SWA only contains the actions necessary for deploying the already built files that the template repo provided. The `deploy-site.yml` file contains additional steps that enable automatic building of the documentation using both MkDocs and Sphinx (in production, you would probably only use one).

```
- name: Set up Python 3.10
  uses: actions/setup-python@v2
  with:
    python-version: 3.10.8
```

```
- name: Build install poetry
  run: |
    curl -sSL https://raw.githubusercontent.com/python-poetry/poetry/master/get-poetry.py | python -
    echo "$HOME/.poetry/bin" >> $GITHUB_PATH
```

```
- name: Install Poetry
  run: poetry install
```

```
- name: Build docs with Sphinx
  run: poetry run sphinx-build -b html docs/source/sphinx-example
  docs/build/sphinx-example
```

```
- name: Build docs with MkDocs
  run: poetry run mkdocs build --config-file docs/mkdocs.yml
```

Rather than modifying the workflow-file created by the SWA, we just renamed the

github secret token to DEPLOYMENT_TOKEN (which is what the deploy-site.yml-file expects) and deleted the file created by the SWA. By modifying the secret *before* deleting the file, we ensured that the workflow was not triggered by a new commit before the renaming of the secret was handled.

(Optional) Implement this solution into an existing repo

To implement automatic doc-building into your own, existing repository, follow the next steps

Ok, the time has come to decide which doc-builder you want to use. For the next few steps, we will be using MkDocs. Other builders will have slightly different configurations that you will have to read up on yourselves.

Since this template repo uses *two* documentation builders, we have two sub-folder in the docs-directory. We will now only be using *one* builder, and should therefor restructure our repo to a more common folder structure:

1. Create two empty folders in the repo's top directory called docs and build.

Note: Git is often reluctant to stage empty folders. You might want to add an empty file if you are unable to stage the changes.

2. If you want to use the Equinor theme, copy the docs\source\equinor-example\stylesheets-folder into your newly created docs-folder.
3. Move the mkdocs.yml-file into the top folder. Open it add change the following two attributes:
 - docs_dir: 'docs'
 - site_dir: 'build'

Note: You will have to update your navigation bar based on the documents and file structure you want. See the [MkDocs documentetion](#) for more information.

Note: You might want to change other attributes in this file to better match your GitHub and SWA information.

4. Since you will only be using *one* doc-builder, please apply the following changes:
 - i. In the deploy-site.yml file, remove the action related to doc-builder that you *will not* be using. If you e.g. want to use MkDocs, remove the Sphinx-action.
 - ii. If you are using MkDocs-case: since we moved the mkdocs.yml-file into the root folder, we no longer have to provide a specific path. Therefor, modify the action as follows: run: poetry run mkdocs build
 - iii. If you are using Poetry, navigate to the pyproject.toml file and remove the dependencies related to the documentation builder that you are no longer using (in our case sphinx and myst-parser).
 - iv. Update the output_location in the **Build And Deploy**-step to **"/build"**

You may now continue on the rest of the original tutorial.

(Optional): Editing the documentation

Sphinx:

Sphinx expects .rts-files as default, but can be extended to support a range of



different file formats (like e.g. markdown). Locate the example in the `.\docs\sources\sphinx-example-folder`. To add a new pages, create the file and add it to the `index.rst`-file. Then re-build the documentation page by either pushing a new commit or by building it manually (see the next chapter).

MkDocs

MkDocs only supports markdown-files. An example is located in the `.\docs\sources\equinor-example-folder`. To add a new pages, create the file and add it to the `mkdocs.yml`-file. Then re-build the documentation page by either pushing a new commit or by building it manually (see the next chapter).

NB! The `mkdocs.yml` file would normally be placed at the top folder for the documentation files, but is currently placed on level above in this demo to make it easier to work with two documentation frameworks.

Note: In the `mkdocs.yml`-file, please update the keys `site_url`, `repo_name` and `repo_url` to match your own project. The builder inserts these links into your page (in the logo and the edit-button)

(Optional): Building the documentation manually

1. [Install poetry](#) and then run `poetry install` in the project folder OR use any package manager of your choice and ensure that you have sphinx and/or mkdocs installed.
NB! If you experience trouble installing poetry (especially if you're an Equinor employee on a Windows), try the following:
 - i. Don't install Python using the Windows Store. Instead, download and install [Python 3.10](#) (others have experienced issues with the MS Store version, see [here](#) and [here](#)).
 - ii. As an Equinor employee, following the [documentation](#) might result in a socket-error: "socket.gaierror: [Errno 11001] getaddrinfo failed". This is probably a proxy-issue that occurs when you are on the work-network. Try repeating the installation step from a different network (e.g. hotspotting from your phone), or modify the relevant proxies.
 - iii. Despite the Poetry documentation explicitly stating that it add the relevant PATH environment variables for you, this does not happen in all cases. If this becomes an issue, try manually adding `%USERPROFILE%\poetry\bin`. When this is done, verify that it works by running `poetry --version` in your terminal.
 - iv. By default, Poetry creates a virtual environment in `{cache-dir}\virtualenvs` (Windows). If you instead want it to be placed in the same folder as your project, enter the following command in you terminal `poetry config virtualenvs.in-project true`. If you now run `poetry install`, the relevant files should now be placed in your current working directory.
2. Depending on which documentation compiler you are using, choose either the first, the second or both options below. Make a change in the docs and see that is included in your build:
 - o Sphinx:
 - a. Recompile the documentation by running `poetry run sphinx-build -b html docs/source/sphinx-example docs/build/sphinx-example`.
 - b. Open the `index.html`-file inside the `build/sphinx-example`-

folder.

- o MkDocs:
 - a. Recompile the documentation by running `poetry run mkdocs build --config-file docs/mkdocs.yml`.
 - b. You can serve the page locally as follows: `poetry run mkdocs serve -f docs/source/equinor-example/mkdocs.yml`.

Step 4: Set up authentication using Azure Active Directory

Goal: After this section, the user should be able to click the "login" button on the SWA and then have access to the content behind the "Verify authenticated role" button. This can be useful if you want to restrict content to employees of your company only.

For the next two steps, we will follow along with Microsoft's own [tutorial](#).

By default, every user belongs to the built-in *anonymous* role, and all logged-in users are members of the *authenticated* role.

Also by default, SWAs allow authentication through Azure AD, github, Twitter, Facebook and Google. In this example, we only want employees of our organization to be able to authenticate, and other options has therefor manually been disabled in the `staticwebapp.config.json`-file.

Create an Azure Active Directory application

The current chapter is based on [this section](#) in Microsoft's own documentation.

App registration

1. In the Azure portal, search for and navigate to *Azure Active Directory*.
2. In the menu bar, select **App registrations**.
3. Select **+ New registration** to open the *Register an application page*.
 - o NB! Your organization might require you to activate a role in **Azure Privileged Identity Management** with elevated privileges to do this.
4. Enter a name for the application. For example, **MyStaticWebApp**.
5. For *Supported account types*, select **Accounts in this organizational directory only**.
6. For *Redirect URIs*, select **Web** and enter the AAD login [authentication callback](#) of your static web app. For example, `<YOUR_SITE_URL>/.auth/login/aad/callback`.

Replace `<YOUR_SITE_URL>` with the URL of your static web app from a previous section.

7. Select **Register**.
8. After the app registration is created, write down the **Application (client) ID** and **Directory (tenant) ID** in the *Essentials* section. You'll need these values to configure AAD authentication in your static web app.

We will now create a client secret:

1. Select *Authentication* in the menu bar.
2. In the *Implicit grant and hybrid flows* section, select **ID tokens (used for implicit and hybrid flows)**



This configuration is required by Static Web Apps to authenticate your users.

3. Select **Save**.
4. Select *Certificates & secrets* in the menu bar.
5. In the *Client secrets section*, select **+ New client secret**.
6. Enter a name for the client secret. For example, **MyStaticWebApp**.
7. Choose a duration for the Expires field (default is 6 months).

Note: *You must rotate the secret before the expiration date by generating a new secret and updating your app with its value.*

8. Select **Add**.
9. Write down the **Value** of the client secret you created, which we will refer to as your **client secret value**. You'll need this value to configure AAD authentication in your static web app.

Output

- Application (client) ID
- Directory (tenant) ID
- Client secret value

Configure Active Directory authentication

The current chapter is based on [this section](#) in Microsoft's own documentation.

1. In a browser, open the GitHub repository containing the static web app you deployed. Navigate to the app's configuration file at *staticwebapp.config.json*. It contains the following section:

```
"auth": {
  "rolesSource": "/api/GetRoles",
  "identityProviders": {
    "azureActiveDirectory": {
      "userDetailsClaim":
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/name",
      "registration": {
        "openIdIssuer":
"http://login.microsoftonline.com/<YOUR_AAD_TENANT_ID>",
        "clientIdSettingName": "AAD_CLIENT_ID",
        "clientSecretSettingName": "AAD_CLIENT_SECRET"
      },
      "login": {
        "loginParameters": [
          "resource=https://graph.microsoft.com"
        ]
      }
    }
  }
}
```

Note: To obtain an access token for Microsoft Graph, the loginParameters field must be configured with resource=https://graph.microsoft.com.

2. Select the **Edit** button to update the file.
3. Update the *openIdIssuer* value of `https://login.microsoftonline.com/<YOUR_AAD_TENANT_ID>` by replacing `<YOUR_AAD_TENANT_ID>` with the **directory (tenant) ID** of your AAD that you wrote down earlier.
4. Select **Commit directly to the main branch** and select **Commit changes**.
5. A GitHub Actions run triggers to update the static web app.
6. Navigate to your SWA resource in the Azure portal.
7. Select **Configuration** in the menu bar.
8. In the *Application settings* section, add the following settings:

Name	Value
AAD_CLIENT_ID	Your AD application (client) ID
AAD_CLIENT_SECRET	Your AD application client secret value

9. Select **Save**.

If you visit our landing page and click the **Verify "authenticated" role** button, you will be met with an "401: Unauthorized" page.

If you instead clicks the Login-button, you will be offered to login in via an AAD portal, and then redirected back to your page. You should now have access to the content placed behind the **Verify "authenticated" role** button, which should look like this:

Note: Your organization might have special rules set up to manage who get's access. If you are not able to complete the login, you can try to first elevate your privileges in Azure's **Privileged Identity Management** and the retry logging in.

Note: Some organizations have extra safety steps in place in order to successfully set up authentication. This might include:

1. Having to add two or more owners to the App Registration
2. The API permissions of your App Registration might require you to forward a consent to an administrator to be evaluated, with a justification for requesting this app. These requests will typically go into an admin consent workflow. When accepted, users should no longer be prompted with this message.

Output

- AAD_CLIENT_ID
- AAD_CLIENT_SECRET

Step 5: Role management

Goal: This section will show how to obtain even more granular access control by assigning access roles to a subset of your organization.

The current chapter is based on [this section](#) in Microsoft's own documentation.

In the previous chapter, we enforced that users had to log in to access the content behind the **Verify "authenticated" role** button. In this chapter, we will create a custom "reader" role that we can assign to users who should be able access the content behind the **Verify "reader" role** button. If a user is not already given the "reader" role, they will still not have access to the content behind the **Verify "reader" role** button, and will be met with the following page:

To achieve this, the template repo contains a serverless function (*api/GetRoles/index.js*) that queries Microsoft Graph to determine if a user is in a pre-defined group. Based on the user's group memberships, the function assigns custom roles to the user. The application is configured to restrict certain routes based on these custom roles. You can read more about Azure Functions in SWA [here](#).

1. In AAD, navigate to "groups" and find a group to which you want to assign the reader role. Write down the **Object Id**.
2. In your GitHub repository, navigate to the *GetRoles* function located at *api/GetRoles/index.js*. Near the top, there is a *roleGroupMappings* object that maps custom user roles to AAD groups.
3. Click the **Edit** button.
4. Update the object with your custom role name (e.g. "reader") and the group ID(s) from your AAD tenant.

```
const roleGroupMappings = {
  'reader': '<Object ID>'
};
```

The *GetRoles* function is called whenever a user is successfully authenticated with Azure Active Directory. The function uses the user's access token to query their Active Directory group membership from Microsoft Graph. If the user is a member of any groups defined in the *roleGroupMappings* object, the corresponding custom roles are returned by the function.

Note: The current function app only supports the assignment of *one* AAD Group per role. If you would like to assign the same role to multiple groups, you must rewrite the function a little. Feel free to open a PR if you decide to do so ;)

In the above example, if a user is a member of the Active Directory group that you selected in the first step, they are granted the reader role.

5. Select **Commit directly to the main branch** and select **Commit changes**.
6. A GitHub Actions run triggers to update the static web app.
7. When the deployment is complete, you can verify your changes by navigating to the app's URL.



8. Log in to your static web app using AAD.
9. You should now be able to see the content behind the **Verify "reader" role** button, but not the content behind the **Verify "admin" role** button.

Step 6: Routing and role authentication

See the [Configure Azure Static Web Apps](#) for more information about routing and setting the role requirements for the different part of the website. This is all configured in the `staticwebapp.config.json`-file.

Note: `routes.json`, which was previously used to configure routing, is deprecated.

Hide the complete web app behind a "login wall"

If you want to require that users log in before even landing on your page, consider adding e.g. the following routes rule `staticwebapp.config.json`

```
{
  "route": "/*",
  "allowedRoles": [
    "authenticated"
  ]
}
```

This will require all users to have the role "authenticated" (i.e. be logged in), and will by default great them with 401: Unauthorized. To handle this, add the following response override to the same config file:

```
"responseOverrides": {
  "401": {
    "redirect": "/.auth/login/aad?post_login_redirect_uri=.referrer",
    "statusCode": 302
  },
}
```

Which automatically will redirect all unauthorized users to the login portal.

Architecture

A graphical overview of where the different keys, secrets, IDs, etc. goes.

File content explanation

Here is an description of the main files and folders in this project:

```
Project
├── .github
│   ├── workflows
│   │   ├── deploy-site.yml          - Github Action workflow for building
│   │   │                           documentation and deploying the service to Azure
│   │   ├── lint-and-format.yml      - Github Action workflow for linting and
│   │   │                           validating syntax
│   └── api
│       ├── (...)                  - Several config files for the serverless function
│       │                           powered by Azure Function)
│       └── GetRoles
```

```

|   |   | function.json      - Azure Function config
|   |   | index.js         - Serverless Function that assigns custom roles
to users
|   |   |
|   |   | docs
|   |   | | source         - Contains source files for documentation.
|   |   | | | equinor-example - Source folder for the Sphinx example
documentation code.
|   |   | | | sphinx-example  - Source folder for the MkDocs example
documentation code.
|   |   | | |
|   |   | | | build          - Custom and compiled HTML files than can be
manually configured and will be hosted through the Azure Static Web Apps.
|   |   | | | index.html     - Landing page
|   |   | | | doc_index.html - Page for choosing which documentation
example you would like to see
|   |   | | | authenticated.html - Demonstrating the "authenticated" role
|   |   | | | reader_role.html  - Demonstrating the "reader" role
|   |   | | | staticwebapp.config.json - Azure Static Web Apps
routes/authentication configuration (see: https://docs.microsoft.com/en-us/azure/static-web-apps/configuration)
|   |   | | |
|   |   | | | sphinx-example  - Build folder for the Sphinx example
documentation code.
|   |   | | | equinor-example - Build folder for the MkDocs example
documentation code.
|   |   | | |
|   |   | | | img            - Images used in this readme-file
|   |   | | |
|   |   | | | pyproject.toml  - (Optional) Project config used by the [Poetry
package manager](https://python-poetry.org/).
|   |   | | | poetry.lock     - (Optional) Package description used by the
package manager.

```

5. CONCLUSION

The result of the project is described from the perspective of the aim and scope set in the beginning of the thesis. The ideas for the future web-based project management system are also described here.

The aim of the project was to make a complete, fully working web based project management system for the company. Requirements from the company has been gathered and taken into account. In web based project

management system there has been used an already implemented TRAC system to improve company's everyday use and to increase performance, productivity and efficiency. As a good project management system it has a possibility to upload, download and delete files and uniformly gives change for developers to be in constant contact with the customer requirements and expectations for the project. User management tool in web based project management system is a good appliance for keeping eye on the project and for giving rights to different users by system administrator in company. This all makes a complete and good communication system inside company, all data and material will be accessible from one place, to facilitate the solution of a project and contact communication with a client. Finally, the whole system has been tested to ensure that everything functions correctly before the system processes actual data and produces information that people will relay on. The features that were implemented are listed below.

General principles of system,
implemented:

1. Users management

- ✓ Adding new users
- ✓ Adding reliable data to user
- ✓ Ability to add different rights to users (admin)
- ✓ Ability to change password
- ✓ Ability to delete users
- ✓ Ability to lock users (user don't have possibility to log-in to system)

- ✓ Ability to modify data and further on to change rights

2. File upload

- ✓ Users are able to delete files
- ✓ Upload files through Trac interface
- ✓ Download files

3. Project management

- ✓ Adding new projects to system
- ✓ Adding projects logo to Trac system
- ✓ Define a admin for a specific project
- ✓ Adding needed data to project
- ✓ Updating files to project
- ✓ Deleting updated files in project

4. Project based views (Every project has its own Trac project)

5. Security login added to system (authentication of a user)

To conclude, web based project management system is an improved TRAC system and completed as one fully functioning system that corresponds to the companies needs and helps to produce a good quality web application projects for the clients. The result of the project responded to the customer"s expectations. The company was satisfied with the features implemented and their reliability and robustness. Through the thesis and development process I gained quite good experience, of an overall structure of different systems and the basic concept of the system as whole. It was quite challenging to improve already made system as TRAC, while adding there new features and to put TRAC to work with new functions. New ideas of what more to improve or how to improve the system and what kind of new features to add, come up through the development of thesis. For example there has been an idea, to make search functionality to project, which will help users to search projects by date or by creator of a project. During the process of implementation, wonderful ideas have been got and hopefully in a near future, there is a possibility and time allocated to improve the system.

REFERENCES

1. Software project management: from concept to deployment / Kieron Conway. Scottsdale (Ariz.) : Coriolis, c2001
2. Software project management / Bob Hughes and Mike Cotterell, London [etc.]: McGraw-Hill, c2002, 3rd ed.
3. Information systems project management: methods, tools and techniques / John McManus and Trevor Wood-Harper, Harlow [etc.] : Prentice Hall, c2003
4. Subversion version control: using the Subversion version control system in development projects / William Nagel, Upper Saddle River (N.J.): Prentice Hall/PTR, c2005
5. Systems Analysis and Design Shelly Cashman Adamski Boston 1991
6. Software Engineering Roger S.Pressman UK, c2000, 5th ed.
7. Ubuntu - <http://packages.ubuntu.com/intrepid/libapache2-mod-wsgi>
8. Open SSH - <http://sial.org/howto/openssh/publickey-auth/>
- 9..<http://ezinearticles.com/?Project-Management:-History-and-Evolution&id=340860>
10. TRAC - <http://trac.edgewall.org/>
11. System Architecture - http://en.wikipedia.org/wiki/Systems_architecture