

CENG 491

D&D SOFTWARE

**DETAILED DESIGN
REPORT**

Prepared by:

Firat Alpergin

Dogan Yazar

Tuncay Namli

Mehmet R. Dogar

TABLE OF CONTENTS

1. INTRODUCTION.....	2
1.1 PROBLEM DEFINITION.....	2
1.2 GOALS & OBJECTIVES	2
1.3 STATEMENT OF SCOPE	4
1.4 DESIGN CONSTRAINTS	4
1.5 WORK BREAKDOWN STRUCTURE	5
2. ARCHITECTURAL & MODULAR DESIGN	8
2.1 DEPLOYMENT VIEW	8
2.2 MODULAR VIEW	10
3. SYSTEM DESIGN.....	15
3.1 USE CASE VIEW	15
3.2 CLASS VIEW.....	26
3.2.1 CLASSES	26
3.2.2 CLASS ASSOCIATIONS	56
3.3 DYNAMIC VIEW	60
3.4 ACTIVITY VIEW	85
3.5 STATECHART VIEW	88
3.6 DATABASE DESIGN.....	95
3.6.1 ER DIAGRAM	95
3.6.2 DATABASE TABLES	96
4. INTERFACES	102
5. TESTING ISSUES.....	108
6. PROJECT SCHEDULE.....	109

1. INTRODUCTION

1.1 PROBLEM DEFINITION

Due to the importance of a detailed design for this project (DProject), this document tries to deduce a final design, step by step, starting with the very basic goals and objectives of the project and then presenting the design details in an incremental manner. This document will be helpful for the development of all parts of the project, including implementation phase, testing phase, debugging phase, and documentation phase.

The resulting product of this development process will be a web-based project management tool, DProject. DProject lets its users define new companies in the system, manage the projects of a company, perform task management operations, perform user operations, perform user time-management operations, perform resource management, perform notification operations within the system, create and export/import files & statistics, perform meeting arrangement operations, use project planning facilities, view forums and compose forum threads. The clients of the project also have the opportunity to view the overall progress of the project they are purchasing. The tool's ultimate aim is to ease the development of a project by all means.

The main functionalities, goals and objectives of DProject can be found in the section that follows.

1.2 GOALS & OBJECTIVES

DProject sets its limits to the level where the aim of easing the management of projects can be fully satisfied. The main goals and objectives of DProject are as follows:

- To provide easy and secure access to its users. The ease of access is accomplished by the web-based nature of DProject. To be able to provide enough security to its users, DProject will have additional security issues that will provide the secure environment to any of its users.
- To provide consistency into the system among the members. In the real world projects, there is hierarchical decomposition among the project team (and generally in the company). This should also appear in a project management tool and DProject accomplishes that by defining different levels of access rights that can simulate the real world hierarchy (e.g. administrator rights, project manager rights, ordinary user rights, etc.).
- To provide efficient task management operations. Task management is one of the most important features of a project management tool and DProject offers advanced task management features to its users. Users, depending on their access rights, can create tasks, assign users to tasks, assign reviewers to tasks, can view

task history trails, monitor task progress, perform critical path management, work on tasks, etc.

- To provide features for efficiently managing meetings. Meeting management is one of the most problematic issues of a typical project development process, especially in major ones. DProject uses a special system, in which the arranger of the meeting provides options for the meeting and notifies them. Then according to the feedback from the potential attendants, DProject lets the meeting arranger choose the optimum meeting details, also taking the preferences of the arranger into account.
- To provide users with a personal time-schedule/calendar, by which they can plan their time more efficiently. This calendar will automatically include task-deadlines, project deadlines, project milestones, and meeting dates; and also the user will be able to create her/his items. Also, the user will be able to mark these items so that they will be automatically reminded to her/him on time.
- To provide communication means among the users. Communication is very important in large scale projects and DProject provides notifications within the system to satisfy the communication needs of its users. Another important communication feature is forums, which can be used for any purpose among the members of a company.
- To provide human management features. Human factor is an important variable projects so they are treated separately in DProject. The users working in a project, the amount of work done by each member, the payment information of members, and many other features can be monitored and controlled in DProject.
- To provide resource management features. Resources of a management are very important entities and efficient ways should be developed for handling the management of them. In DProject, different resources can be attached to different projects (or companies, more generally), their necessary information (e.g. unit price, seller address, etc.) are kept, resources can be attached to tasks, budget information of a project is kept and updated accordingly, etc.
- To provide features for report & statistics generation and their exportation/importation. Reports and statistics are vital for any project development because they are useful both within the project and also among different projects because they are used for various purposes including efficient project planning, user capability analyses, etc. DProject has a number of important features for efficient report & statistics operations. These include the importation/exportation of reports from/to different formats, the importation/exportation of a project as a whole from one system to another, statistic generations for specific subset of tasks, for the overall project tasks, for user teams, for individual members, for a duration of time, for the whole project life span, etc.

- To provide efficient means of project scheduling. Scheduling is one of the most problematic issues of a project development process that can occur in serious conflicts between the developer side and the client side. To be able to prevent such inconsistencies, DProject offers sophisticated features for project scheduling. The users can see task creation times, the estimated hours spent on tasks or the whole project, Gantt charts created automatically, etc.
- To provide features for the clients to follow the progress of the project. The clients naturally want to view the project they are purchasing, so DProject lets its clients see the necessary information for them to understand that whether the project is progressing as they wish or not.

1.3 STATEMENT OF SCOPE

The following general requirements apply to DProject:

- A way to define new company and set up new company information
- A way to add new users to the system
- A way to define new projects and set up new project information
- A way to define tasks, assign users to tasks, assign reviewers to tasks, work on tasks, attach resources to tasks, review tasks, confirm/reject tasks, view tasks
- A way to supply users with time-schedules/calendars which include important dates
- A way to handle critical path management
- A way to arrange meetings
- A way to handle communication among users
- A way to handle human management
- A way to handle resource management
- A way to create, import/export statistics & reports
- A way to perform project planning
- A way to perform project progress monitoring for clients

1.4 DESIGN CONSTRAINTS

To be able to work efficiently, satisfying the requirements imposed, DProject should be carefully designed. However, there are some design constraints which should be taken into account while designing the system.

Web-based applications are the ones which decay fastest, due to the changing nature of the Web. DProject has to be designed to use the most new technologies in terms of web-server, application-server, and database-server integrity; has to be designed to use the most new protocols, and standards; giving it the chance to live the longest life it can, in this short-living applications world: WWW.

The security issue is another problem that a web-based application must be aware of. Since our tool will serve to companies in developing their projects, the protection of the information about these projects is a crucial issue. To resist to security attacks successfully, three points must be considered. First, the architectural design of DProject should be resistant to attacks; and this requires good design. Second, one should be aware that these new technologies of web-based applications that DProject promises to use and support, are also the ones that create security risks, since they are new and has gone through less testing than their older counterparts. This awareness should bring good choice of technology, not taking the 'newest' one, but taking the 'newest and most secure' one. And as third point, DProject offer to its customer two modes of use: a WWW-based use, or a LAN-based use. Our customers can use DProject on our servers via WWW, or can buy the application and install it to their own LAN-server, which brings more security.

Installation of computer applications had always been a problem for big companies when they have hundreds of computers, and when every single computer will use the new application. In such a case, the 'Installation CD' may need to travel hundreds of computers. To prevent such a thing to happen, DProject needs only a browser on the client-side of the application; so that an installation only to the server will suffice, no matter how many client computers will use it. While this brings more burden to the development team in terms of not using the components offered for applications, but trying to fix everything so that it will work on a browser, this will prevent our customers from much more pain when they are installing DProject to one server but not hundreds of machines.

DProject is a web-based system and that adds an overhead because of the possible problems with the Internet connection. To be able to minimize the effect of this overhead, the communication within the system modules should be minimized avoiding the unnecessary interactions that can further delay processing.

DProject is a system that heavily interacts with the database behind it. Nearly all the necessary information for processing is maintained in the database. There is a heavy load of fetching/storing data from/to the database. This makes the efficiency of the DBMS an important constraint that must be taken into account seriously. An efficient DBMS should be used and the database should be carefully designed, preventing any unnecessary burden put on the DBMS. Also the queries should be designed efficiently to minimize the cost of database operations. By this way, the overhead caused by the DBMS can be minimized.

1.5 WORK BREAKDOWN STRUCTURE

This part of the document presents the work breakdown structure for DProject. Note that the work-package definitions for 'implementation' and 'test & debugging' sub-projects are tentative, and will be revised in each successive document. Also note that in the Gantt chart, and responsibility breakdown of work-packages the numbering for the work-packages of this section will be used, and work-package names will not be rephrased.

Work Package Name	Numbering
Project: DProject	01-00-00
Sub-Project: Prototype Production	01-01-00
Work-package: Creation of Database Tables (Limited for prototype, including: User Account Tables, Project Tables; Company Tables, Task Tables)	01-01-01
Work-package: Database Interface Classes	01-01-02
Work-package: Implementation of Visual Classes (Limited for prototype, including: Upper Menu, Right Menu, Header, Footer)	01-01-03
Work-package: Implementation of Procedural Classes (Limited for prototype, including: Session class(limited), Initializer class (limited), SqlConnection class, Project class(limited), User class (limited), Task class(limited))	01-01-04
Work-package: Implementation of JSP architecture (Limited for prototype, including: login screen, projects screen, users screen, tasks screen)	01-01-05
Sub-project: Implementation for Development Snapshot	01-02-00
Work-package: Database Implementation	01-02-01
Work-package: Project Management Module	01-02-02
Work-package: Task Management Module	01-02-03
Work-package: Meeting Management Module	01-02-04
Work-package: Notifications Module	01-02-05
Work-package: Implementation of Visual Classes	01-02-06
Work-package: Implementation of JSP pages	01-02-07
Sub-project: Implementation for Final Product	01-03-00
Work-package: Database Implementation Completion	01-03-01
Work-package: Calendar Module	01-03-02
Work-package: Report Generation Module	01-03-03
Work-package: Statistics Generation Module	01-03-04
Work-package: Forum Module	01-03-05
Sub-project: Documentation	01-04-00
Work-package: User's manual	01-04-01
Work-package: Installation manual	01-04-02
Work-package: Software Specification & Release Notes	01-04-03
Work-package: Help pages	01-04-04
Sub-project: Testing & Debugging	01-05-00
Work-package: Determination of Test-cases for Security	01-05-01
Work-package: Determination of Test-cases for Database Integrity	01-05-02

Work-package: Determination of Test-cases for Architectural Integrity	01-05-03
Work-package: Application of Security Tests	01-05-04
Work-package: Application of Database Integrity Tests	01-05-05
Work-package: Application of Architectural Integrity Tests	01-05-06
Work-package: Debugging	01-05-07

Tentative Responsibility Breakdown For Work-Packages

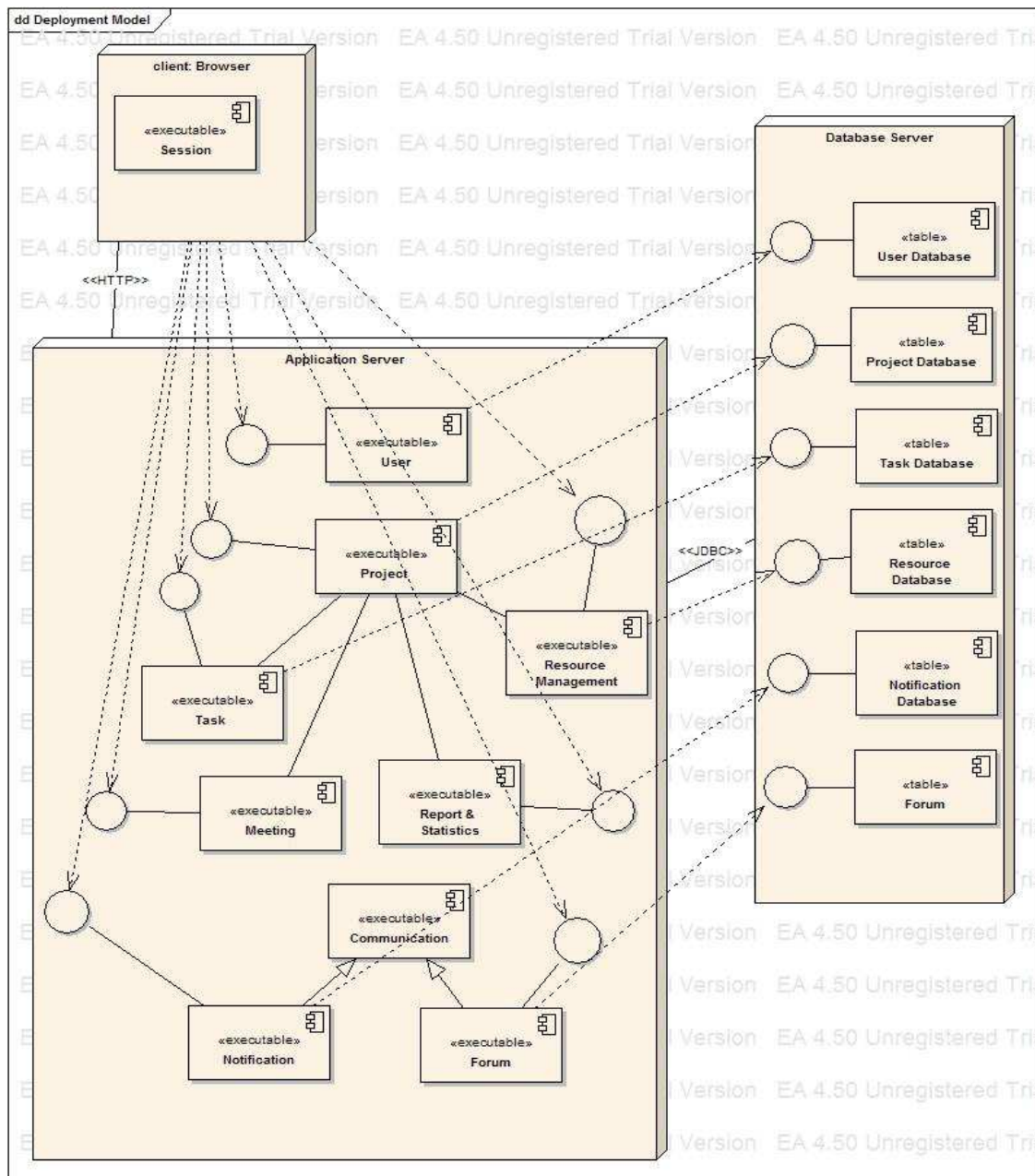
Here is the tentative responsibility breakdown of the work-packages between members of our company, D&D software. Each member's name and the numbers (using the numbering of the section Work-Breakdown Structure) of the work-packages he is responsible of are listed.

D&D Software Member Name	Work-Package Numbers	
Tuncay Namli	01-01-01	01-01-02
	01-01-03	01-02-01
	01-02-06	01-02-07
	01-03-04	01-03-05
	01-04-03	01-05-03
	01-05-06	01-05-07
Dogan Yazar	01-01-01	01-01-02
	01-01-03	01-02-01
	01-02-04	01-02-05
	01-03-01	01-03-05
	01-04-02	01-05-02
	01-05-05	01-05-07
Firat Alpergin	01-01-04	01-01-05
	01-02-02	01-02-03
	01-02-04	01-02-05
	01-03-03	01-04-04
	01-05-03	01-05-06
	01-05-07	
Mehmet Remzi Doğar	01-01-04	01-01-05
	01-02-02	01-02-03
	01-02-06	01-02-07
	01-03-02	01-04-04
	01-05-01	01-05-04
	01-05-07	

2. ARCHITECTURAL & MODULAR DESIGN

2.1 DEPLOYMENT VIEW

The deployment diagram of our system is as follows:



DProject consists of 3 main nodes, namely the browser, the application server, and the database server. Each node consists of a number of components that declares interfaces and the components communicate among themselves by means of the methods in the interfaces.

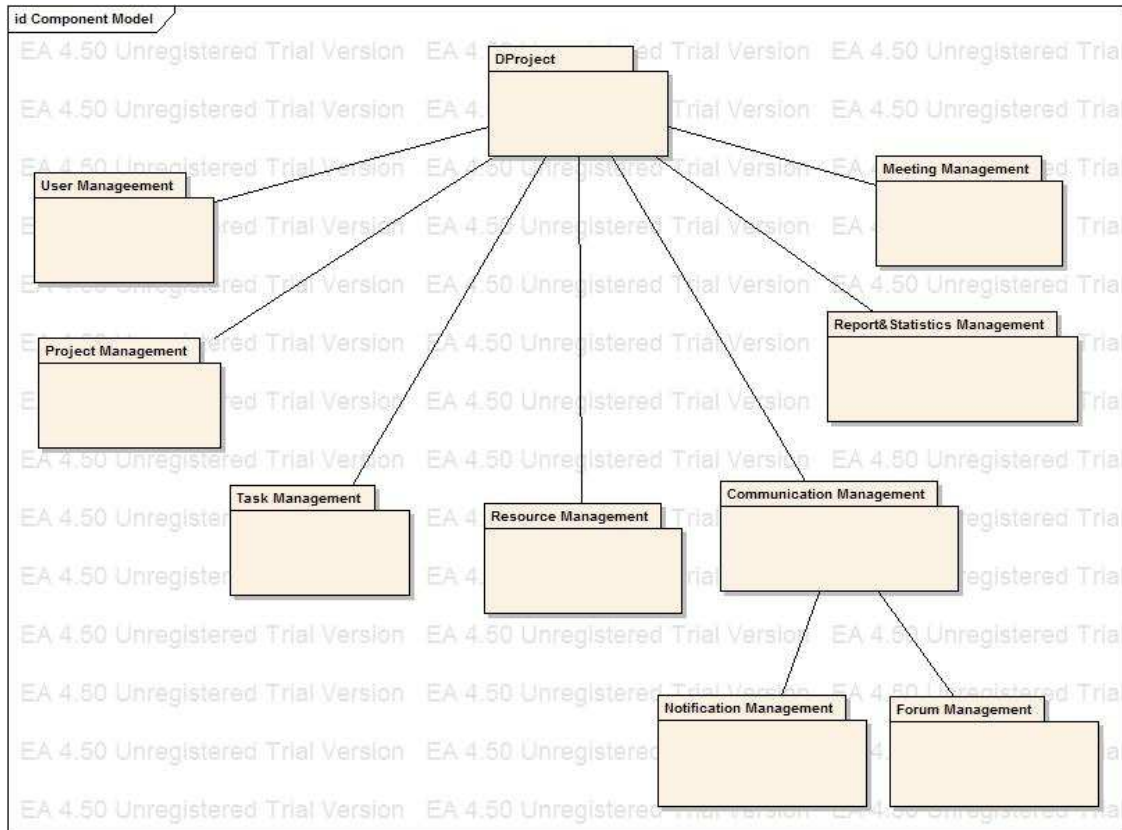
Browser works on the user side. It communicates with the application server through HTTP protocol. It consists of the Session component, which connects to the other components working on the application server through the interfaces they declare.

In the application server, the main components that constitute the functionality of DProject sit. First, there is the User component, which is responsible for handling the operations related to the users in the system. The project component is responsible for the Project operations in the system and it further consists of 4 sub-components, which are Task, Meeting, Report & Statistics, and Resource Management. These components consist of all the classes and methods that implement the task management, meeting management, report & statistics management, and the resource management features of DProject. Another component is the Communication component which further generalizes into 2 components: the Notification component and the Forum component. These two components are responsible for the 2 main communication features of DProject, which are notification and forum. Each component in the application server declares their own interfaces so that the Session component sitting on the browser can use the services provided by these components through these interfaces.

Finally, there is the database server node. There is a JDBC connection between the database server and the application server. The database server consists of the database portions of DProject. There are specific database components for the specific components in the application server. There is the User Database, Project Database, Task Database, Resource Database, Notification Database, and Forum Database, which consists of the user, project, task, resource, notification, and forum records respectively. All these components again declare interfaces, and the appropriate component in the application server communicates with these components through the interfaces they declare.

2.2 MODULAR VIEW

The modular decomposition of our system is as follows:



The structural decomposition of DProject consists of the following modules. There are separate modules for:

1. User Management
2. Project Management
3. Task Management
4. Resource Management
5. Communication Management
 - 5.1 Notification Management
 - 5.2 Forum Management
6. Report & Statistics Management
7. Meeting Management

The explanation of each module is as follows:

2.2.1 User Management Module

The user management module handles all the operations related to the users of the system. New user accounts can be created and their information is set (and can be modified later). While creating new users, long with their basic information like name, address, password, etc.; access rights of the users are also set. There are 2 different access right types of users:

1 – Global Access Rights: These are specific for a user in the system and shows that whether the user has normal access rights or administrator access rights.

2 – Project-Based Access Rights: These can be different for a user in different projects and specify what operations the user is allowed to perform in the project.

Also, there is a client type user in DProject (clients are the individuals who finance the project), and these individuals have a different kind of access rights, which let them see only the basic things about the project, that show the overall progress of the project.

Another important info about a user is the user's payment type (weekly, monthly, etc.) and the amount of payment. This is required by the Resource Management module and used in the automatic updates of the project budget. Also, users have calendars, and these calendars can either be modified manually by the user (entering the information in the time slots), and automatically by DProject (e.g. when the user is assigned to a task, its deadline is marked automatically on the user's calendar).

2.2.2 Project Management Module

The project management module handles the operations related to the projects in DProject. New projects can be created and their information can be modified in DProject (if the user to create the project has enough access rights). The operations like adding new users to project (current users can be assigned and also new user accounts can be created upon adding the user to the project), adding new tasks to projects, specifying project budget & resources, generating reports & statistics about the project are all handled by the project management module.

2.2.3 Task Management Module

Task management module consists of all the operations related to the tasks in DProject. Tasks are single pieces of work that should be done for a project to be complete. Each project has numbers of tasks and they are managed by the task management module.

For a user to create a new task, he should have enough access rights to do so. The tasks that are already created can be modified later by the users with enough access rights. There is important information about a task that should be kept and is managed by our system. First of all, tasks have priorities, which show the relative importance of the

task in the project, a priority should be attained to a task. Also there are different task types (e.g. development task, accounting task, etc.) and a task can be categorized this way. Dependencies among tasks is another important concept and information about the task dependency should be present so that DProject automatically generates permissions to users when they want to work on a task (e.g. when a user is assigned to a task, which is dependant on a former task that is not completed yet, DProject does not let the user start working on the task before the other one is completed). Tasks can also be related to the milestones of the project and that is necessary for the critical path management features of DProject. Upon creation time, task is assigned to users and reviewers (these assignments can be modified later - with necessary access rights, of course). The assigned users are the ones that should complete the task by working on it. When they complete the task, the work done by them is sent to the reviewers and reviewers can either accept or reject the work done. If rejected, the assigned users are notified and obliged to do the work again. When an assigned user wants to work on a task, he uses the IN/OUT facility. The user presses the IN button upon he starts working and closes the IN item by pressing the OUT button upon he finishes working on the task. Then the time the user spent working on the task is kept in both the task's history, and the user's history. Also, necessary resources can be assigned to tasks. If a specific amount of a particular project resource is assigned to a task, the resource management module of DProject automatically updates the project resource information. Also, files can be attached to a task, which is necessary in the case that the users assigned to the task should read some documents about the task before they start working on it, for example.

2.2.4 Resource Management Module

The resource management module handles all the operations related to the material (including project budget and all the materials used in the project) resources of a project. The information about the resources of a project are kept and processed in DProject. This information includes resource name, unit price, address to buy, etc. These resources can be assigned to tasks, as explained in the task management module. The project budget is also managed by DProject.

The project budget and project resources are updated automatically by DProject as follows:

- Project resources are
 - * incremented as new purchases are made
 - * decremented as resources are assigned to tasks
- Project budget is
 - * decremented as employees are paid
 - * decremented as new purchases are made

Also, the manual update of project resources and budget is always possible, provided that the user has the necessary access rights.

2.2.5 Communication Management Module

The communication management module handles all the communication issues among the users of DProject. It consists of two separate modules, namely the notification management module and the forum management module.

a) Notification Management Module

Notification management module is responsible for handling the notification operations in the system. Notifications are the messages that are local to our system. Each user can see his notifications in his notification inbox and can send notifications to other users. Also, if the user selects that option, an e-mail message can be sent along with the notification.

Also, DProject sends automatic notifications to the users. This occurs when the user is:

- assigned to a task
- assigned as the reviewer of a task
- required to attend a meeting
- required to make preferences about a meeting time
- to be reported on the results of user preferences about a meeting

The first two operations are already explained in the task management module and the last three ones are explained in the meeting management module.

b) Forum Management Module

A forum is provided for each company in DProject. These forums have all the features presented by a typical forum and it can be used by all the members of that company.

2.2.6 Report & Statistics Management Module

The report and statistics management module is related to the generation of all the reports and/or statistics of a project for all users (including the project members and clients).

The clients, as explained earlier, can only see basic statistics about a project, including the overall progress of the project (which is stated by comparing the total person-hours worked up to now with the initially estimated person-hours) and, the probability of missing the pre-defined milestones or the project deadline, etc. These statistics are presented to the client in a simple style, with additional visual features (e.g. pie charts) if needed.

The users can generate the overall statistics of a project. These statistics include all the information about a project, including the amount of progress, number of completed tasks, number of tasks in progress, total person-hours worked, overall completion times of tasks (possibly among with different task groups and/or user teams), etc. These statistics can also be presented by visual features, if desired.

The users can also generate reports using filters. Filters provide different options to the users while generating the reports. For example, a user can generate filters between specific time periods, for specific users, for specific task types, for specific task statuses (e.g. completed, in progress, etc.), for specific milestones, etc. The generated filters can be saved for later user. When the filter is specified, the report is generated by DProject. Also, reports can be imported and exported in different formats (e.g. XML, Excel). Users also have the opportunity of creating Gantt charts.

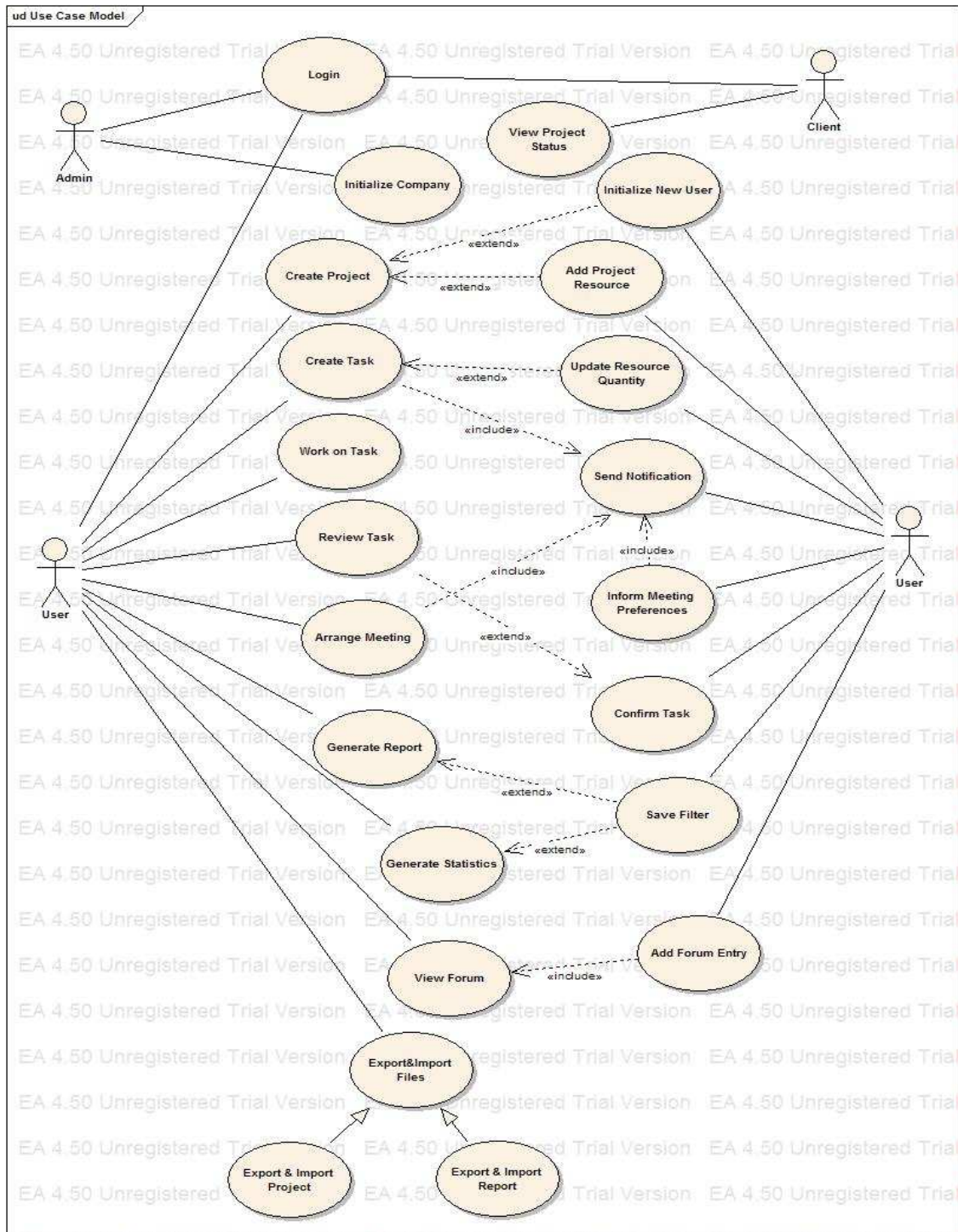
2.2.7 Meeting Management Module

The meeting management module handles all the operations related to the organization of meetings. Basically there are three sides in the arrangement of a meeting: arranger side, attendant side, and DProject. The meeting management operations are handled as follows.

When a user wants to arrange a meeting (arranger side), all he has to do is to provide necessary meeting information (e.g, duration, place, attached files if necessary, etc.), select the potential attendants (attendant side) and provide the different time options. Then these time options are automatically sent to the attendants by DProject (by means of notifications). Then to potential attendants give priorities to these different options, this is all they have to do. Then comes in the role of DProject: depending on the priorities of the potential attendants of the user, DProject fixes the best possible meeting time and notifies the arranger. If none of the time options is suitable for none of the users, then DProject notifies the arranger that no option could be selected (it is the arranger's option then, whether changing the time options, or canceling the meeting). If a meeting time is fixed and confirmed by the arranger, automated notifications are sent to all the attendants of the meeting, specifying the final meeting details.

3. SYSTEM DESIGN

3.1 USE CASE VIEW



<i>Flow of events for the Login use case</i>	
Objective	To log in the system
Precondition	None
Main Flow	1 – The user enters his login id 2- The user enters his password 3 – The entered id and password are checked for validity 4 – The system creates a new session for the user and displays the main screen of the new user
Alternative Flows	At 3, if the entered id or password is invalid, the user is prompted to enter a new id or password
Post Condition	A new session is created for the user

<i>Flow of events for the Initialize Company use case</i>	
Objective	To set up a new company account in the system
Precondition	The user should have administrator access rights
Main Flow	1 – The user enters new company information 2 – The user selects an id and password for the company 3 – Access rights of the user are checked to see if they are enough or not 4– The entered information is checked for validity (i.e. non-existing company name, non-existing company id) 5 – The main screen of the user is displayed
Alternative Flows	At 2, if the user does not have enough access rights, he is not allowed to set up new company account At 3, if there is a conflict, the user is prompted to enter valid information into the conflicting fields
Post Condition	A new company information is saved in the database

<i>Flow of events for the Initialize New User use case</i>	
Objective	To set up a new user account into the system
Precondition	The user setting up the new account should have administrator access rights
Main Flow	1 – The user enters new account information 2 – The user enters an id and password for the new account 3 – Access rights of the user is checked whether they are enough or not 4 – The entered information for the new user is checked for validity (e.g. non-existing id) 5 - The user is assigned to projects, if necessary 6 – The main screen of the user is displayed
Alternative Flows	At 3, if the access rights of the user are not enough, the user is prompted stating that the intended operation can not be carried on At 4, if the information for the new user is invalid, the user is prompted to enter valid information
Post Condition	A new user account information is saved in the database

<i>Flow of events for the Create Project use case</i>	
Objective	To create a new project
Precondition	A company should be already selected and the user should have enough access rights

<i>Flow of events for the Create Project use case</i>	
Main Flow	<p>1 – The user sets up the information for the new project</p> <p>2 – The access rights of the user is checked to see whether they are enough or not</p> <p>3 – The entered information is checked for validity (e.g. non-existing project name)</p> <p>4 – Existing users are assigned to the new project, if necessary</p> <p>5 – Task groups, task types and task priorities are set up for the new project, if necessary</p> <p>6 – Resource information is set up for the new project, if necessary</p> <p>7 – The main screen of the user is displayed</p>
Alternative Flows	<p>At 2, if the user does not have the necessary access rights, he is prompted stating that the operation can not be carried out</p> <p>At 3, if the entered information is not valid, the user is prompted to enter valid information to the invalid fields</p> <p>At 4, if a new user should be assigned to the project, a new user account is created</p>
Post Condition	A new project is created and saved in the database

<i>Flow of events for the Create Task use case</i>	
Objective	To create a new task in a project
Precondition	A project should be selected already and the user should have enough access rights
Main Flow	<p>1 – The user enters information for the new task</p> <p>2 – The user assigns reviewers to the new task</p> <p>3 – The user assigns user to the new task</p> <p>4 – The user assigns resources to the new task, if necessary</p> <p>5 – Files are attached to the new task by the user, if necessary</p> <p>6 – A unique identifier is created and saved for the new task</p>
Alternative Flows	None
Post Condition	A new task is saved in the database

<i>Flow of events for the Work on Task use case</i>	
Objective	To work on a particular task in a project
Precondition	A task should be selected already and the user should be assigned to the selected task
Main Flow	1 – The user opens the IN/OUT item to start working on a task 2 – The user selects preferences for the current IN/OUT item 3 – The user adds comments on the work done, if necessary 4 – The user closes the IN/OUT item when the work is completed 5 – The user sends the task to reviewers, if necessary
Alternative Flows	None
Post Condition	The new progress status of the task is saved and the task history is updated, working history of the user is updated

<i>Flow of events for the Review Task use case</i>	
Objective	To review the work done on task
Precondition	A task should be already selected, the user should be assigned as reviewer to the task and should be notified for review
Main Flow	1 – The user reviews the work done on task 2 – The user either accepts or rejects the work done 3 – The user that sent the task for review is notified on the reaction of the reviewer
Alternative Flows	None
Post Condition	Depending on the reaction of the reviewer, the work done is accepted or the user is obliged to do the work again, the status of the task is updated accordingly

<i>Flow of events for the Confirm Task use case</i>	
Objective	To confirm the task as completed or not

<i>Flow of events for the Confirm Task use case</i>	
Precondition	A task should be already selected, the user should be assigned to the task as reviewer and should be notified for review
Main Flow	1 – The user reviews the work done on task 2 – The user either selects the task as completed or not 3 – The user that sent the task for review is notified depending on the reaction of the reviewer
Alternative Flows	None
Post Condition	Depending on the reaction of the reviewer, the task is marked as completed or not, and the status of the task is updated accordingly

<i>Flow of events for the Generate Report use case</i>	
Objective	To create and view a time report or task report
Precondition	None
Main Flow	1 – The user selects the type of the report to be created 2 – The user selects the filter to generate the report 3 – The user saves the filter, if necessary 4 – The report is generated depending on the filter 5 – The report is displayed
Alternative Flows	At 2, if the user makes invalid selections (e.g. non-existing date), the user is prompted to change the selections At 3, if there is a conflict in saving the filter (e.g. existing filter name), the user is prompted to remove the conflict
Post Condition	The report is generated and the filter is saved, if selected

<i>Flow of events for the Generate Statistics use case</i>	
Objective	To create and view statistics of a project
Precondition	A project should be already selected

<i>Flow of events for the Generate Statistics use case</i>	
Main Flow	1 – The user selects the filter to generate the statistics 2 – The user saves the filter, if necessary 3 – The statistics are generated depending on the filter 4 – The statistics are displayed
Alternative Flows	At 1, if the user makes invalid selections (e.g. non-existing date), the user is prompted to change the selections At 2, if there is a conflict in saving the filter (e.g. existing filter name), the user is prompted to remove the conflict
Post Condition	The statistics are generated and the filter is saved, if selected

<i>Flow of events for the Save Filter use case</i>	
Objective	To save a filter for later use
Precondition	None
Main Flow	1 – The user makes the selections for the different fields of the filter 2 – The user selects a name for the filter 3 – The user saves the filter
Alternative Flows	At 1, if the user makes an invalid selection (e.g. non-existing date), the user is prompted to change the selection At 2, if the user selects an existing date, he is prompted to change the name
Post Condition	A filter is saved in the system

<i>Flow of events for the Arrange Meeting use case</i>	
Objective	To arrange a meeting
Precondition	The user should have necessary access rights to arrange a meeting

<i>Flow of events for the Arrange Meeting use case</i>	
Main Flow	1 – The user selects potential dates for the meeting 2 – The user selects the potential attendants of the meeting 3 – The user notifies the potential attendants on the potential dates 4 – Depending on the selections of the potential attendants, the user fixes the details of the meeting 5 – The user notifies the user stating the meeting details and attendants
Alternative Flows	None
Post Condition	A new meeting is created and its details are saved

<i>Flow of events for the Inform Meeting Preference use case</i>	
Objective	To inform the arranger about the selections about a meeting
Precondition	The user should have been notified by the arranger
Main Flow	1 – The user views the potential dates sent by the arranger 2 – The user notifies the arranger stating the dates suitable for him
Alternative Flows	None
Post Condition	The user preferences are sent to the arranger for further processing

<i>Flow of events for the Export & Import Files use case</i>	
Objective	To export & import files from/to the system
Precondition	A project or a report should be already selected
Main Flow	1 – The user selects whether to import/export a report or a whole project 2 – Depending on the selection of the user, either a report is imported/exported in the specified format, or the whole project is imported/exported as SQL statements
Alternative Flows	At 2, if the file to be imported/exported is invalid, the user is prompted stating that the file is invalid

<i>Flow of events for the Export & Import Files use case</i>	
Post Condition	Depending on the exported/imported file, either a new report file, or a new project is saved/opened

<i>Flow of events for the View Forum use case</i>	
Objective	To view forum threads
Precondition	None
Main Flow	1 – The user selects the forum he wants to view 2 – The user selects the thread to be viewed 3 – The thread that the user selected is displayed
Alternative Flows	None
Post Condition	A forum thread is displayed

<i>Flow of events for the Add Forum Entry use case</i>	
Objective	To add a new forum entry
Precondition	None
Main Flow	1 – The user selects the forum to which he wants to add a new entry 2 – The user selects the thread under which he wants to add a new entry 3 – The user adds the entry to the forum thread 4 – The thread is displayed with the new entry added
Alternative Flows	None
Post Condition	A new entry is added to the forum

<i>Flow of events for the Add Project Resource use case</i>	
Objective	To add a new resource information to a project

<i>Flow of events for the Add Project Resource use case</i>	
Precondition	A project should be selected and the user should have enough access rights
Main Flow	1 – The user enters the information of the new resource 2 – The user enters the quantity of the new resource 3 – The user enters the unit price of the new resource
Alternative Flows	At 1, if one of the fields is conflicting (e.g. existing resource name), the user is prompted to change the conflicting field
Post Condition	New resource type and information is saved

<i>Flow of events for the Update Project Resource use case</i>	
Objective	To update the information & quantity of a resource
Precondition	A project should be selected and the user should have enough access rights to make the update
Main Flow	1 – The user selects the resource to be updated 2 – The user selects the fields of the resource that are to be updated 3 – User updates the fields accordingly
Alternative Flows	At 3, if there is an invalid selection (e.g. resource quantity below zero), the user is prompted to change the selection
Post Condition	The information of the resource is updated and saved

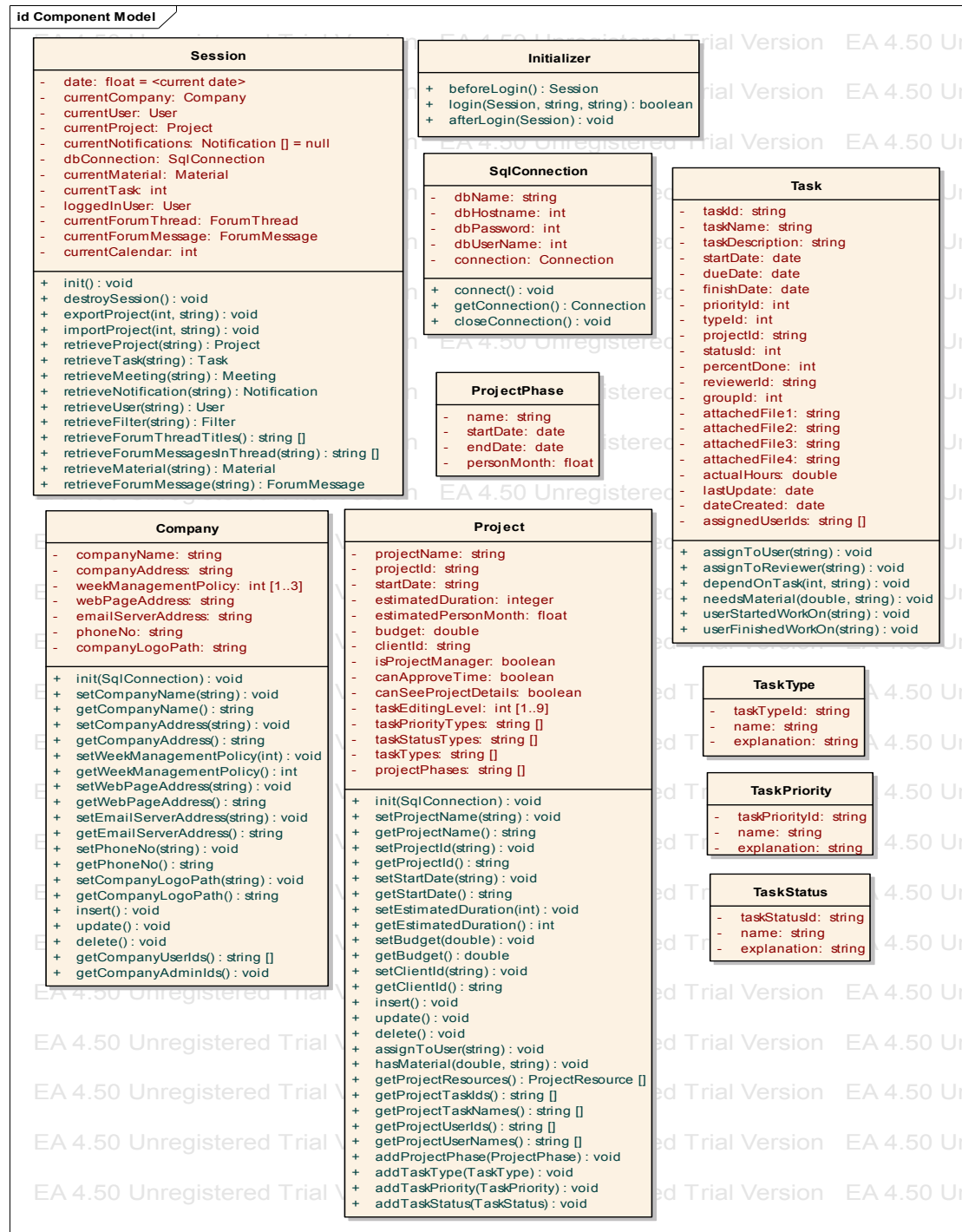
<i>Flow of events for the Send Notification use case</i>	
Objective	To send notifications to other users in the system
Precondition	None

<i>Flow of events for the Send Notification use case</i>	
Main Flow	1 – The user enters the subject of the notification, if desired 2 – The user writes the main body of the notification, if desired 3 – Files are attached to the notification by the user, if desired 4 – The users selects the users to send the notification 5 – The user sends the notification
Alternative Flows	At 3, if the user tries to attach an invalid file (e.g. excess file size, corrupted file), the user is prompted about the error At 4, if the user tries to send the notification to a non-existing user, he is prompted about the error
Post Condition	A notification is sent to other users in the system

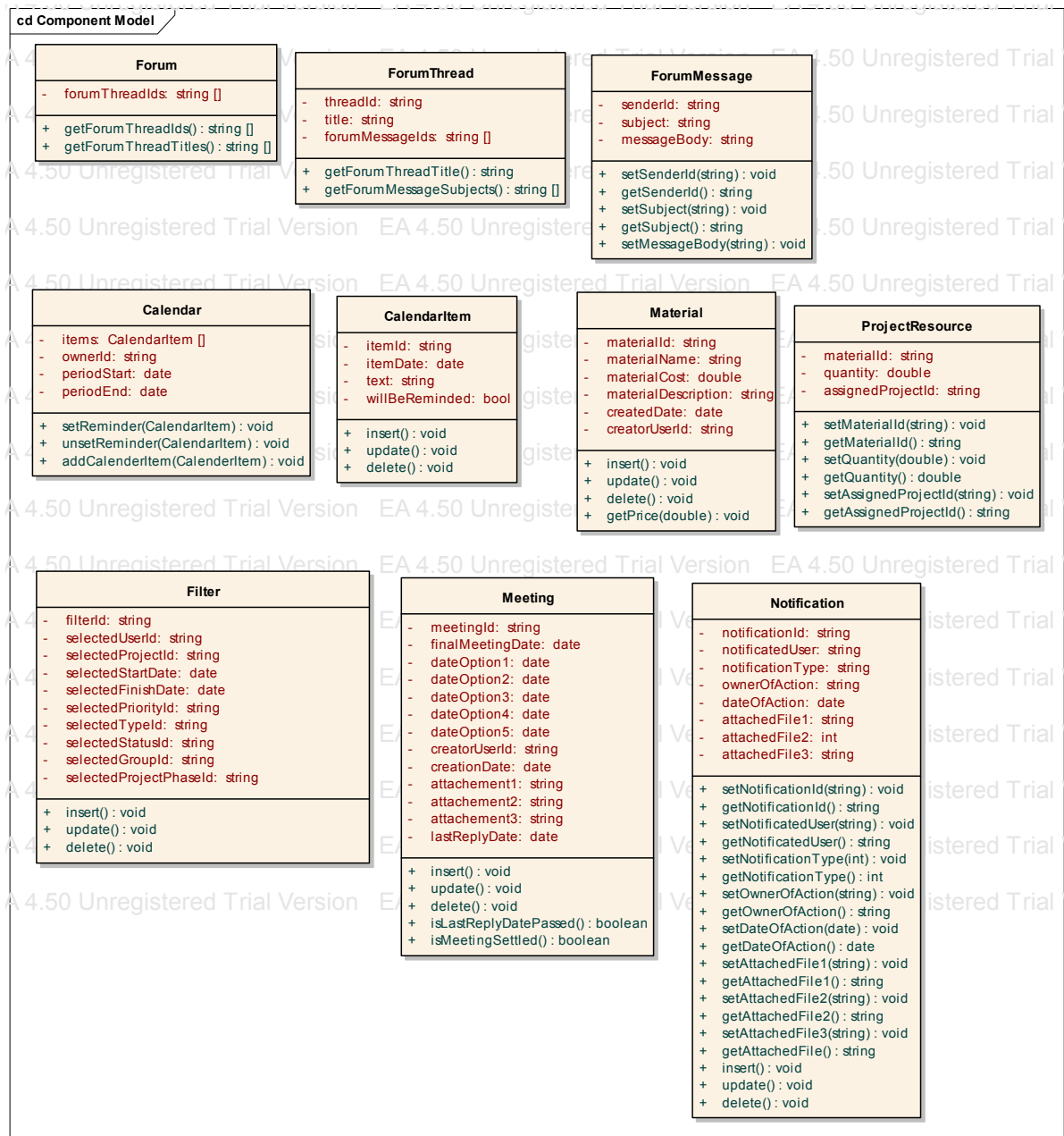
<i>Flow of events for the View Project Status use case</i>	
Objective	For a client to see the overall status of the project
Precondition	The client should be logged in to the system
Main Flow	1 – The client selects the project he wants to view 2 – System generates the statistics 3 – The statistics are displayed to the client
Alternative Flows	None
Post Condition	The statistics are displayed on client's screen

3.2 CLASS VIEW

3.2.1 CLASSES



User
<ul style="list-style-type: none"> - userId: string - password: string - name: string - middleName: string - surname: string - birthDate: string - speciality: string - address: string - sex: int [1..3] - emailAddress: string - photo: string - paymentPolicy: int [1..3] - paymentAmount: double - emailNotificationForNewTaskPreference: int [0..1] = 1 - numOfTasksPerPagePreference: int [1..50] = 10 - numOfMonthPerPagePreference: int = 4 - numOfWeekPerPagePreference: int = 4 - userProjects: Project [] - canAddProject: boolean - userDirectory: int [1..4] - globalAccessRight: int
<ul style="list-style-type: none"> + init(SqlConnection): void + setUserId(string): void + getUserId(): string + setPassword(string): void + getPassword(): string + setName(string): void + getName(): string + setmiddleName(string): void + getMiddleName(): string + setSurname(string): void + getSumame(): string + setBirthDate(string): void + getBirthDate(): string + setSpeciality(string): void + getSpeciality(): string + getAddress(string): string + setSex(int): void + getSex(): int + getEmailAddress(string): string + setPhoto(string): void + getPhoto(): string + setPaymentPolicy(int): void + getPaymentPolicy(): int + setPaymentAmount(double): void + getPaymentAmount(): double + setEmailNotificationForNewTaskPreference(int): void + getEmailNotificationForNewTaskPreference(): int + setNumOfTasksPerPagePreference(int): void + getNumOfTasksPerPagePreference(): int + setNumOfWeekPerPagePreference(int): void + getNumOfWeekPerPagePreference(): int + setNumOfMonthPerPagePreference(int): void + getNumOfMonthPerPagePreference(): int + setUserProjects(Project []): void + getUserProjects(): Project [] + insert(): void + update(): void + delete(): void + setAccessRightsOfUser(string, string, int, boolean, boolean): void + createNewProject(int, string, string, string, double, date, date, date, string, string, string): void + createUser(int, double, int, int, int, int, boolean, int, boolean, int, string, string, string, int, date, string, string, string, string, string): void + ceateNewTask(string, string, string, string, int, string, int, int, string, int, int, date, date, date, string, string, string): void + createMaterial(string, double, string, string): void + createMeeting(date, string, string, string, date, date, date, date, date, date, string): void + buysMaterial(double, double, string): void + setUserPreferencesForMeeting(string, int, int, int, int): void + approveTask(): void + assignToProject(string, boolean, boolean, int): void + createCompany(string, string, int, string, string, string): void + sendNotification(string, string, string, string, string): void



In this section a description of all the classes in DProject is presented. The order is hierarchical:

Firstly, the classes that live all through the lifetime of a DProject session is given:

- Session,
- Initializer,
- and SqlConnection classes.

Then the classes that handle each module's functionalities and their helper classes are given:

- Company class for creation of new companies in DProject;
- Project class for project management module (with helper class ProjectPhase);
- Task class for task management module (with helper classes TaskPriority, TaskType, TaskStatus);
- Meeting class for meeting management module;
- User class for user management module;
- Calendar class for personal time-scheduling module (with helper class CalendarItem)
- Notification class for handling notification facility in DProject
- Forum class for forum module of DProject (with helper classes ForumThread, ForumMessage)
- ProjectResource class for resource management module (with helper class Material)

Session

Type: **public Class**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Component Model
Details: Created on 04.12.2004 12:15:36. Modified on 09.01.2001 22:20:09.

'Session' class represents a unique session started by a user. It is initialized when the user opens a DProject page using the web-browser, and terminated when the user terminates the web session. Session class is always active and in association with all other classes. It holds pointers to the currently active project objects, task objects, user objects, etc. and acts like an interface between the user and the rest of the application.

Session Attributes

Attribute	Type	Notes
date	private <i>float</i>	'date' attribute shows the time that the session is started by the user. It is set to the server system time initially. Initial Value: <current date>;
currentCompany	private <i>Company</i>	'currentCompany' attribute is a pointer to an instance of the Company class, which is the logged-

		in user's company.
currentUser	private <i>User</i>	: 'currentUser' attribute is a pointer to an instance of the User class, which is a user being modified, or shown to the user.
currentProject	private <i>Project</i>	: 'currentProject' attribute is a pointer to an instance of the Project class, which is the one the user is currently working on. It is initially set to null.
currentNotifications	private <i>Notification []</i>	: 'currentNotifications' attribute is an array of instances of the Notification class, which are waiting for the currently working user. It is initially set to null. Initial Value: null;
dbConnection	private <i>SqlConnection</i>	: 'dbConnection' is an instance of the class SqlConnection and is used to connect to the database in this session.
currentMaterial	private <i>Material</i>	: 'currentMaterial' attribute represents the currently active material in the session.
currentTask	private <i>int</i>	: 'currentTask' attribute represents the currently active task in the session.
loggedInUser	private <i>User</i>	: 'loggedInUser' attribute is a pointer to an instance of the User class, which is the logged-in user.
currentForumThread	private <i>ForumThread</i>	: 'currentForumThread' attribute is a pointer to an instance of the ForumThread class, which is the currently active forum thread.
currentForumMessage	private <i>ForumMessage</i>	: 'currentForumMessage' attribute is a pointer to an instance of the ForumMessage class, which is the currently active forum message.
currentCalendar	private <i>int</i>	: 'currentCalendar' attribute is a pointer to an instance of the currentCalendar class, which is the currently active calendar.

Session Methods

Method	Type	Notes
init ()	public: <i>void</i>	'init' initializes the Session object for a user session. It sets the attributes 'date', 'currentCompany', and 'loggedInUser', by calling their 'init' functions. In this way the system initializes itself hierarchal.
destroySession ()	public: <i>void</i>	'destroySession' destroys the current session securely, if the user logs-out.
exportProject (<i>int</i> , <i>string</i>)	public: <i>void</i>	param: format [int - in] param: projectId [string - in] 'exportProject' exports the specified project using the specified format.
importProject (<i>int</i> ,	public: <i>void</i>	param: fileFormat [int - in]

<i>string</i>)		param: fileName [string - in] 'importProject' imports a project using the specified file name and the specified file format.
retrieveProject (<i>string</i>)	public: <i>Project</i>	param: projectId [string - in] 'retrieveProject' returns the Project object that has the specified project id. It queries the database with the specified project id, creates the Project object, and returns it.
retrieveTask (<i>string</i>)	public: <i>Task</i>	param: taskId [string - in] 'retrieveTask' returns the Task object that has the specified task id. It queries the database with the specified task id, creates the Task object, and returns it.
retrieveMeeting (<i>string</i>)	public: <i>Meeting</i>	param: meetingId [string - in] 'retrieveMeeting' returns the Meeting object that has the specified meeting id. It queries the database with the specified meeting id, creates the Meeting object, and returns it.
retrieveNotification (<i>string</i>)	public: <i>Notification</i>	param: notificationId [string - in] 'retrieveNotification' method returns the Notification object that has the specified notification id. It queries the database with the specified notification id, creates the Notification object, and returns it.
retrieveUser (<i>string</i>)	public: <i>User</i>	param: userId [string - in] 'retrieveUser' method returns the User object that has the specified user id. It queries the database with the specified user id, creates the User object, and returns it.
retrieveFilter (<i>string</i>)	public: <i>Filter</i>	param: filterId [string - in] 'retrieveFilter' method returns the Filter object that has the specified filter id. It queries the database with the specified filter id, creates the Filter object, and returns it.
retrieveForumThreadTitles ()	public: <i>string</i> []	'retrieveForumThreadTitles' returns the titles of the threads in the forum. It queries the database to get the titles of ForumThread objects, and returns them in a string array.
	public: <i>string</i> []	param: threadId [string - in]

retrieveForumMessage sInThread (<i>string</i>)		'retrieveForumMessagesInThread' method returns the subject attributes of the forum messages in the specified forum thread with the threadId. It queries the database with this id and returns the subject attributes of the forum messages in a string array.
retrieveMaterial (<i>string</i>)	public: <i>Material</i>	param: materialId [string - in] 'retrieveMaterial' method returns the Material object that has the specified material id. It queries the database with the specified material id, creates the Material object, and returns it.
retrieveForumMessage (<i>string</i>)	public: <i>ForumMessage</i>	param: messageId [string - in] 'retrieveForumMessage' returns the forum message object with the specified messageId. It queries the database with the messageId, creates the ForumMessage class and returns it.

SqlConnection

Type: public **Class**

Status: Proposed. Version 1.0. Phase 1.0.

Package: Component Model

Details: Created on 04.12.2004 20:00:28. Modified on 07.01.2005 02:19:15.

SqlConnection class holds the attributes needed to connect to the database and acts as a wrapper class for connecting to the database. It is used throughout a DProject session, to handle database connections.

SqlConnection Attributes

Attribute	Type	Notes
dbName	private <i>string</i>	: 'dbName' attribute represents the name of the database. It is needed to connect to the database.
dbHostname	private <i>int</i>	: 'dbHostName' represents the database host name. It is needed to connect to the database.
dbPassword	private <i>int</i>	: 'dbPassword' attribute represents the database password, for the database host and user. It is required to get a connection to the database.
dbUserName	private <i>int</i>	: 'dbUserName' attribute represents the database user name. It is required to get a connection to the database.
connection	private <i>Connection</i>	: 'connection' is the actual database connection that is established by the SqlConnection class. Its type

		depends on the programming language that will be used. In our case it will most probably be of type <code>java.sql.Connection</code> .
--	--	--

SqlConnection Methods

Method	Type	Notes
<code>connect ()</code>	<code>public: void</code>	'connect' connects to the database using <code>dbName</code> , <code>dbHostName</code> , <code>dbPassword</code> , and <code>dbUsername</code> attributes of <code>SqlConnection</code> class and initializes the connection variable.
<code>getConnection ()</code>	<code>public: Connection</code>	'getConnection' returns the connection variable which was connected to the database.
<code>closeConnection ()</code>	<code>public: void</code>	'closeConnection' closes the database connection.

Initializer

Type: `public Class`

Status: Proposed. Version 1.0. Phase 1.0.

Package: Component Model

Details: Created on 04.12.2004 20:25:31. Modified on 07.01.2005 02:19:15.

'Initializer' class is a helper class for logging in and initializing the session. It initializes the environment before the login takes place, handles the login process, and initializes the environment after login.

Initializer Methods

Method	Type	Notes
<code>beforeLogin ()</code>	<code>public: Session</code>	'beforeLogin' is called when a user opens DProject page with but before he/she logs in. It handles all the work done before the login takes place; like showing the login screen, setting-up the visual environment, etc.
<code>login (Session, string, string)</code>	<code>public: boolean</code>	param: session [Session - in] param: userPassword [string - in] param: userName [string - in] 'login' takes the password and loginId and checks to see if the id and password is valid and consistent. Returns 'true' if consistent, 'false' if inconsistent or invalid.
<code>afterLogin (Session)</code>	<code>public: void</code>	param: session [Session - in]

		'afterLogin' is called after 'login' method returns as 'true' and it initializes all required attributes of the current session for a logged in user.
--	--	---

Company

Type: *public* **Class**

Status: Proposed. Version 1.0. Phase 1.0.

Package: Component Model

Details: Created on 04.12.2004 12:58:53. Modified on 07.01.2005 02:45:58.

'Company' class represents the company that the current logged-in user is a member of. If the user has the required access rights, he/she can create, modify, and delete companies in DProject.

Company Attributes

Attribute	Type	Notes
companyName	private <i>string</i>	: 'companyName' is the name of this Company instance.
companyAddress	private <i>string</i>	: 'companyAddress' is the actual address of this Company instance.
weekManagementPolicy	private Range:1 to 3: <i>int</i>	: 'weekManagementPolicy' attribute holds integer values corresponding to the preference of the company on how to arrange working days of a week. That integer values have the range 1-3. The relations are 1:Monday-to-Friday, 2:Monday-to-Saturday, 3:Monday-to-Sunday
webPageAddress	private <i>string</i>	: 'webPageAddress' attribute holds the string representing the company's web page address.
emailServerAddress	private <i>string</i>	: 'emailServerAddress' holds the mail-server address of the company that will be used to send e-mails using the company's server.
phoneNo	private <i>string</i>	: 'phoneNo' is the telephone number of this Company instance.
companyLogoPath	private <i>string</i>	: 'companyLogoPath' attribute holds the path to the image file that the company had submitted. This is used to show the company logo when it is a session of this company's users.

Company Methods

Method	Type	Notes
init (<i>SqlConnection</i>)	public: <i>void</i>	param: dbConnection [SqlConnection - in]

insert ()	public: <i>void</i>	'insert' method writes the information in this 'Company' class instance to the database, creating a new entry in the database table.
update ()	public: <i>void</i>	'update' method updates the record of this company in the database, using the current values of the attributes.
delete ()	public: <i>void</i>	'delete' method deletes the record of this company from the database.
getCompanyUserIds ()	public: <i>string []</i>	'getCompanyUserIds' returns the user ids who are members of this company.
getCompanyAdminIds ()	public: <i>void</i>	'getCompanyAdminIds' returns ids of the administrators of this company.

Project

Type: *public Class*

Status: Proposed. Version 1.0. Phase 1.0.

Package: Component Model

Details: Created on 04.12.2004 14:06:28. Modified on 09.01.2001 21:16:41.

'Project' class represents a project in DProject. An instance of this class is created and manipulated when a user is creating, modifying, or inspecting a DProject project.

Project Attributes

Attribute	Type	Notes
projectName	private <i>string</i>	: 'name' is the project's name in real life.
projectId	private <i>string</i>	: 'projectId' is the unique id that a project gets in DProject.
startDate	private <i>string</i>	: 'startDate' is the start date of the project.
estimatedDuration	private <i>integer</i>	: 'estimatedDuration' attribute specifies the project's estimated duration.
estimatedPersonMonth	private <i>float</i>	: 'estimatedPersonMonth' is the value specified by the creator of the project for the estimated value of the work required for the projects in terms of person-months.
budget	private <i>double</i>	: 'budget' attribute holds the project's budget.
clientId	private <i>string</i>	: 'clientId' attribute specifies this project's client.
isProjectManager	private <i>boolean</i>	: 'isProjectManager' specifies if the current user is a manager of this project. 'true' means he/she is a manager, 'false' means he/she is not.
canApproveTime	private	: 'canApproveTime' specifies if the current user can

	<i>boolean</i>	approve users' timesheets in this project. 'true' means he/she can, 'false' means he/she can not.
canSeeProjectDetails	private <i>boolean</i>	: 'canSeeProjectDetails' specifies whether the current user can see all tasks and meetings in the project, or can see only the ones that he/she is assigned to. 'true' means he/she can see all, 'false' means he/she can not.
taskEditingLevel	private Range:1 to 9: <i>int</i>	: 'taskEditingLevel' specifies user2s permissions about task editing. 1 means read-only permission, 2 means limited task editing, 3 means limited task editing and file attachment creation/deletion, 4 means partial task editing, 5 means partial task editing and deleting the tasks that he/she created, 6 means full control task editing, 7 means full control task editing and deleting the tasks that he/she created, 8 means full control task editing and creating tasks and deleting his/her own tasks, 9 means full control task editing and creating deleting all tasks.
taskPriorityTypes	private <i>string []</i>	: 'taskPriorityTypes' is the list of the task priority types specified for this project.
taskStatusTypes	private <i>string []</i>	: 'taskStatusTypes' is the list of the task status types specified for this project.
taskTypes	private <i>string []</i>	: 'taskTypes' is the list of the task types specified for this project.
projectPhases	private <i>string []</i>	: 'projectPhases' is the list of the task project phases specified for this project.

Project Methods

Method	Type	Notes
init (<i>SqlConnection</i>)	public: <i>void</i>	param: dbConnection [<i>SqlConnection</i> - in] 'init' method is called to initialize a Project object. Using the project id database is queried, and fields of the object are filled.
setProjectName (<i>string</i>)	public: <i>void</i>	param: name [<i>string</i> - in]
getProjectName ()	public: <i>string</i>	
setProjectId (<i>string</i>)	public: <i>void</i>	param: id [<i>string</i> - in]
getProjectId ()	public: <i>string</i>	
setStartDate (<i>string</i>)	public: <i>void</i>	param: startDate [<i>string</i> - in]
getStartDate ()	public: <i>string</i>	
setEstimatedDuration	public: <i>void</i>	param: duration [<i>int</i> - in]

<i>(int)</i>		
getEstimatedDuration ()	public: <i>int</i>	
setBudget (<i>double</i>)	public: <i>void</i>	param: budget [<i>double</i> - in]
getBudget ()	public: <i>double</i>	
setClientId (<i>string</i>)	public: <i>void</i>	param: id [<i>string</i> - in]
getClientId ()	public: <i>string</i>	
insert ()	public: <i>void</i>	'insert' method writes the information in this 'Project' class instance to the database, creating a new entry in the database table.
update ()	public: <i>void</i>	'update' method updates the record of this project in the database, using the current values of the attributes.
delete ()	public: <i>void</i>	'delete' method deletes the record of this project from the database.
assignToUser (<i>string</i>)	public: <i>void</i>	param: userId [<i>string</i> - in] 'assignToUser' method add the relation to the database so that the user specified with the userId becomes a member of this project.
hasMaterial (<i>double</i> , <i>string</i>)	public: <i>void</i>	param: quantity [<i>double</i> - in] param: materialId [<i>string</i> - in] 'hasMaterial' method marks the database so that this project has specified quantity of the specified material.
getProjectResources ()	public: <i>ProjectResource []</i>	'getProjectResources' method queries this projects resources from the database and returns them.
getProjectTaskIds ()	public: <i>string []</i>	'getProjectTaskIds' returns the project's task ids.
getProjectTaskNames ()	public: <i>string []</i>	'getProjectTaskNames' returns the project's task names.
getProjectUserIds ()	public: <i>string []</i>	'getProjectUserIds' returns the users' ids who are assigned to the project.
getProjectUserNames ()	public: <i>string []</i>	'getProjectUserNames' returns the users' names assigned to this project.
addProjectPhase (<i>ProjectPhase</i>)	public: <i>void</i>	param: phase [<i>ProjectPhase</i> - in] 'addProjectPhase' method adds the specified <i>ProjectPhase</i> to this project as a phase.
addTaskType (<i>TaskType</i>)	public: <i>void</i>	param: type [<i>TaskType</i> - in] 'addTaskType' method adds the specified <i>TaskType</i> to this project as a type.

addTaskPriority (TaskPriority)	public: void	param: priority [TaskPriority - in] 'addTaskPriority' adds the specified TaskPriority to this project as a phase.
addTaskStatus (TaskStatus)	public: void	param: status [TaskStatus - in] 'addTaskStatus' adds the specified TaskStatus to this project as a phase.

ProjectPhase

Type: public **Class**

Status: Proposed. Version 1.0. Phase 1.0.

Package: Component Model

Details: Created on 09.01.2001 22:00:41. Modified on 09.01.2001 22:14:20.

'ProjectPhase' class represents phases for a specific project, so that a milestone for this phase can be specified and new tasks can be specified as belonging to this category.

ProjectPhase Attributes

Attribute	Type	Notes
name	private : string	'name' is the name of this project phase, as specified by the creator of the project.
startDate	private : date	'startDate' is the date that this project phase must start.
endDate	private : date	'endDate' is the date that this project phase must end.
personMonth	private : float	'personMonth' specifies the effort estimation for this project phase in terms of personMonths.

Task

Type: public **Class**

Status: Proposed. Version 1.0. Phase 1.0.

Package: Component Model

Details: Created on 04.12.2004 18:58:10. Modified on 09.01.2001 21:16:52.

'Task' class represents a task in DProject. It is created during a session when the user creates, modifies, or simply inspects a specific task. It provides the methods for task management.

Task Attributes

Attribute	Type	Notes
taskId	private <i>string</i>	: 'taskId' is the unique id a task gets in a DProject system. This also represents the primary key value of this task in the Tasks table of the database.
taskName	private <i>string</i>	: 'taskName' is the name of the task, supplied by the creator of the task.
taskDescription	private <i>string</i>	: 'taskDescription' is the general information about the task supplied by the creator of the task.
startDate	private <i>date</i>	: 'startDate' is the date that a user can start working on the task.
dueDate	private <i>date</i>	: 'dueDate' is the date by which this task should be complete.
finishDate	private <i>date</i>	: 'finishDate' is the actual date that this task was completed. If finishDate is before dueDate, then that means the task was completed before deadline; if finishDate is after dueDate that means the task was completed late.
priorityId	private <i>int</i>	: 'priorityId' is the priority level of this task. Since different priority levels can be created in DProject, they are represented by an id.
typeId	private <i>int</i>	: 'typeId' is the type of this task. Since different types can be created in DProject, they are represented by an id.
projectId	private <i>string</i>	: 'projectId' is the id of the project that this task belongs to.
statusId	private <i>int</i>	: 'statusId' shows the status of this task. Since different status levels can be created in DProject, they are represented by an id.
percentDone	private <i>int</i>	: 'percentDone' shows what portion of the task is completed, in terms of percentages.
reviewerId	private <i>string</i>	: 'reviewerId' is the id of the user who is the reviewer of this task.
groupId	private <i>int</i>	: 'groupId' is the id of the user group that this task is assigned.
attachedFile1	private <i>string</i>	: 'attachedFile1' represents the path of the file that is attached to this task.
attachedFile2	private <i>string</i>	: 'attachedFile2' represents the path of the file that is attached to this task.
attachedFile3	private <i>string</i>	: 'attachedFile3' represents the path of the file that is attached to this task.
attachedFile4	private <i>string</i>	: 'attachedFile4' represents the path of the file that is attached to this task.
actualHours	private <i>double</i>	: 'actualHours' represents the total hours that has been spent on this task, by all users.
lastUpdate	private	: 'lastUpdate' is the date that this task is modified last.

	<i>date</i>	
dateCreated	private <i>date</i>	: 'dateCreated' is the date that this task was created.
assignedUserIds	private <i>string []</i>	: 'assignedUserIds' is an array of user ids, who are assigned to this task.

Task Methods

Method	Type	Notes
assignToUser (<i>string</i>)	public: <i>void</i>	param: userId [string - in] 'assignToUser' method is used to assign this task to a user specified by the user-id parameter.
assignToReviewer (<i>string</i>)	public: <i>void</i>	param: reviewerId [string - in] 'assignToReviewer' method is used to assign this task to the reviewer specified by the reviewerId parameter.
dependOnTask (<i>int, string</i>)	public: <i>void</i>	param: dependencyType [int - in] param: taskId [string - in] 'dependOnTask' method marks the database so that this task will depend on the task specified by the taskId, by the relation specified by the dependencyType.
needsMaterial (<i>double, string</i>)	public: <i>void</i>	param: quantity [double - in] param: materialId [string - in] 'needsMaterial' assigns the specified quantity of the specified material to this task. The corresponding price will be decreased from the project budget.
userStartedWorkOn (<i>string</i>)	public: <i>void</i>	param: userId [string - in] 'userStartedWorkOn' method marks the database showing that the specified user started working on this task.
userFinishedWorkOn (<i>string</i>)	public: <i>void</i>	param: userId [string - in] 'userFinishedWorkOn' method marks the database showing that the specified user finished working on this task.

TaskPriority

Type: *public* **Class**

Status: Proposed. Version 1.0. Phase 1.0.
Package: Component Model
Details: Created on 09.01.2001 21:52:59. Modified on 09.01.2001 22:15:02.

'TaskPriority' class represents a priority category for the tasks. When creating a project a project manager can specify new task priority types, so that the priorities of new tasks can be specified.

TaskPriority Attributes

Attribute	Type	Notes
taskPriorityId	private <i>string</i>	: 'taskPriorityId' is the unique id given to this taskPriority object in DProject. This is also the primary key value used for this task priority in the database.
name	private <i>string</i>	: 'name' is the name of this task status type as given by the creator of this task status.
explanation	private <i>string</i>	: 'explanation' is the explanation for this task priority as specified by its creator.

TaskStatus

Type: *public* **Class**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Component Model
Details: Created on 09.01.2001 21:42:29. Modified on 09.01.2001 23:56:26.

'TaskStatus' class represents a status category for the tasks. When creating a project a project manager can specify new task statuses, so that task statuses can be changed as the work on task progresses.

TaskStatus Attributes

Attribute	Type	Notes
taskStatusId	private <i>string</i>	: 'taskStatusId' is the unique id given to this taskStatus object in DProject. This is also the primary key value used for this task status in the database.
name	private <i>string</i>	: 'name' is the name of this task status type as given by the creator of this task status.
explanation	private <i>string</i>	: 'explanation' is the explanation for this task status as specified by its creator.

TaskType

Type: *public* **Class**

Status: Proposed. Version 1.0. Phase 1.0.

Package: Component Model

Details: Created on 09.01.2001 21:34:53. Modified on 09.01.2001 21:54:08.

'TaskType' class represents a category for the tasks. When creating a project a project manager can specify new task types, so that new tasks can be put under this category.

TaskType Attributes

Attribute	Type	Notes
taskId	private <i>string</i>	: 'taskId' is the unique id given to this taskType object in DProject. This is also the primary key value used for this task type in the database.
name	private <i>string</i>	: 'name' is the name of this task type as given by the creator of this task type.
explanation	private <i>string</i>	: 'explanation' is the explanation for this task type as specified by its creator.

Meeting

Type: *public* **Class**

Status: Proposed. Version 1.0. Phase 1.0.

Package: Component Model

Details: Created on 04.12.2004 18:58:22. Modified on 09.01.2001 21:16:52.

'Meeting' class represents a scheduled or completed meeting of a project. A user with enough access rights can create, modify, delete meetings; choose other users as attendants, specify date options. Meeting object is created when a user is modifying, creating or inspecting a meeting in DProject.

Meeting Attributes

Attribute	Type	Notes
meetingId	private <i>string</i>	: 'meetingId' is the unique id a meeting gets in DProject. It is also the primary key value used in the database table for this meeting.
finalMeetingDate	private <i>date</i>	: 'finalMeetingDate' is the actual date that the meeting took/will take place.
dateOption1	private <i>date</i>	: 'dateOption1' is one of the date options specified by the arranger of the meeting. The potential attendants of the meetings make choices between these options.

dateOption2	private <i>date</i>	:	'dateOption2' is one of the date options specified by the arranger of the meeting. The potential attendants of the meetings make choices between these options.
dateOption3	private <i>date</i>	:	'dateOption3' is one of the date options specified by the arranger of the meeting. The potential attendants of the meetings make choices between these options.
dateOption4	private <i>date</i>	:	'dateOption4' is one of the date options specified by the arranger of the meeting. The potential attendants of the meetings make choices between these options.
dateOption5	private <i>date</i>	:	'dateOption5' is one of the date options specified by the arranger of the meeting. The potential attendants of the meetings make choices between these options.
creatorUserId	private <i>string</i>	:	'creatorUserId' is the id of the user who arranged the meeting.
creationDate	private <i>date</i>	:	'creationDate' is the actual date that this meeting object was created in the Dproject system by the arranger of the meeting.
attachement1	private <i>string</i>	:	'attachedFile1' represents the path of the file that is attached to this meeting object.
attachement2	private <i>string</i>	:	'attachedFile2' represents the path of the file that is attached to this meeting object.
attachement3	private <i>string</i>	:	'attachedFile3' represents the path of the file that is attached to this meeting object.
lastReplyDate	private <i>date</i>	:	'lastReplyDate' is the deadline for the potential attendants of the meeting to make their choicedes between the time options.

Meeting Methods

Method	Type	Notes
insert ()	public: <i>void</i>	'insert' method inserts the meeting information to the database creating a new entry.
update ()	public: <i>void</i>	'update' method updates the database record of this meeting, using the new attribute values.
delete ()	public: <i>void</i>	'delete' method deletes the database record of this meeting.
isLastReplyDatePassed ()	public: <i>boolean</i>	'isLastReplyDatePassed' returns true if the last reply/decision date for the meeting has passed; false otherwise.
isMeetingSettled ()	public: <i>boolean</i>	'isMeetingSettled' returns true if the date of this meeting had been decided and settled by all attendants; false otherwise.

User

Type: *public* **Class**

Status: Proposed. Version 1.0. Phase 1.0.

Package: Component Model

Details: Created on 04.12.2004 14:06:20. Modified on 07.01.2005 00:52:43.

'User' class represents a user of DProject. In a DProject session, after the user log-in there is always an active user class which represents the currently logged-in user, and holds his/her information. Another use of the User class is to represent a user whose information is being inspected, modified or created by the logged-in user.

User Attributes

Attribute	Type	Notes
userId	private <i>string</i>	: 'userId' attribute holds the id that is used as a unique key to specify a user. This attribute is also used as a login-id.
password	private <i>string</i>	: 'password' attribute holds the user's password.
name	private <i>string</i>	: 'name' attribute holds the user's real life name.
middleName	private <i>string</i>	: 'middleName' attribute holds the user's real life middle name.
surname	private <i>string</i>	: 'surname' attribute holds the user's real life surname.
birthDate	private <i>string</i>	: 'birthDate' attribute holds the user's real life birth date.
speciality	private <i>string</i>	: 'speciality' attribute represents what the user is specialized in as an employee.
address	private <i>string</i>	: 'address' attribute holds the user's real life address.
sex	private Range:1 to 3: <i>int</i>	: 'sex' represents the user's sexual gender. It can only have three values; 1: male, 2: female, 3: other
emailAddress	private <i>string</i>	: 'emailAddress' attribute holds the user's e-mail address.
photo	private <i>string</i>	: 'photo' attribute holds the path to the image file which includes the user's photo if submitted.
paymentPolicy	private Range:1 to 3: <i>int</i>	: 'paymentPolicy' field holds integer values ranging from 1 to 3, representing three different payment policies. These are 1, if the user is paid monthly; 2, if the user is paid weekly; 3, if the user is paid on an

		hourly basis.
paymentAmount	private <i>double</i>	: 'paymentAmount' attribute holds the amount that is paid to the user, for a month (if payment policy is monthly), for a week (if payment policy is weekly), for an hour (if the user is paid for hourly work).
emailNotificationForNewTaskPreference	private Range:0 to 1: <i>int</i>	: If 'emailNotificationForNewTaskPreference' attribute has the value 1, the user is notified via e-mail whenever a task is assigned to him/her; if this attribute has the value 0 he/she is not notified. Initial Value: 1;
numOfTasksPerPagePreference	private Range:1 to 50: <i>int</i>	: 'numOfTasksPerPagePreference' attribute specifies the user's preference so that, when he/she views the tasks of a project, they are shown in groups of this quantity. Initially it is set to 10 so that in a page at most ten tasks are shown. Initial Value: 10;
numOfMonthsPerPagePreference	private <i>int</i>	: 'numOfMonthsPerPage' attribute specifies the user's preference so that, when he/she views a monthly gantt chart, at most this much month will be shown in a page. Initially this will be set to 4, so that in a monthly gantt chart, 4 months at a page will be shown. Initial Value: 4;
numOfWeeksPerPagePreference	private <i>int</i>	: 'numOfWeeksPerPage' attribute specifies the user's preference so that, when he/she views a gantt chart in weekly mode, at most this much week will be shown in a page. Initially this will be set to 4, so that in a weekly gantt chart, 4 weeks at a page will be shown. Initial Value: 4;
userProjects	private <i>Project []</i>	: 'userProjects' array holds instances for all the projects that this user is a member of. This field is set only if this user is the current user of the session.
canAddProject	private <i>boolean</i>	: 'canAddProject' attribute specifies if the user has the permission to create a new project for his/her company.
userDirectory	private Range:1 to 4: <i>int</i>	: 'userDirectory' attribute specifies what a user can see in his/her user directory. 1 means user can see all other users in the same company; 2 means user can see all other users in the same project; 3 means user can see only the administrators; 4 means user can not see anyone so does not have a user directory.
globalAccessRight	private <i>int</i>	: 'globalAccessRight' attribute specifies this user's global permissions in the system. A value of '1' means the user is a client of a project not an

		employee; '2' means the user is an administrator and have all the global rights; '3' means the user is a normal user and his/her permissions are further specified by other attributes.
--	--	---

User Methods

Method	Type	Notes
init (<i>SqlConnection</i>)	public: void	param: dbConnection [SqlConnection - in] After a user login to DProject, 'init' method of the Session class calls this init method of User class, so the the User class attributes are set to correct values. Init class queries the database using the user's login id, and using the information coming from the database, fills in the fields of the User object.
insert ()	public: void	'insert' method writes the information in this 'User' class instance to the database, creating a new entry in the database table.
update ()	public: void	'update' method updates the record of this user in the database, using the current values of the attributes.
delete ()	public: void	'delete' method deletes the record of this user from the database.
setAccessRightsOfUser (<i>string, string, int, boolean, boolean</i>)	public: void	param: userId [string - in] param: projectId [string - in] param: editTaskLevel [int - in] param: canApproveTime [boolean - in] param: isProjectManager [boolean - in] 'setAccessRightsOfUser' sets access rights of the user with the specified id to the specified access rights. It updates the corresponding database tables to do this.
createNewProject (<i>int, string, string, string, double, date, date, date, string, string, string</i>)	public: void	param: projectId [int - in] param: contactEmail [string - in] param: contactPhone [string - in] param: contactName [string - in] param: budget [double - in] param: dueDate [date - in] param: finishDate [date - in] param: startDate [date - in] param: projectDescription [string - in] param: projectName [string - in] param: projectId [string - in] 'createNewProject' method creates a new project

		with the specified attributes. It updates the corresponding database tables to do this.
createUser (int, double, int, int, int, int, boolean, int, boolean, int, string, string, string, int, date, string, string, string, string, string, string)	public: void	param: userDepartmentId [int - in] param: paymentAmount [double - in] param: paymentPolicy [int - in] param: numOfWeeksPerPage [int - in] param: NumOfMonthsPerPage [int - in] param: NumOfTasksPerPage [int - in] param: emailNotificationForNewTask [boolean - in] param: userDirectory [int - in] param: canAddProject [boolean - in] param: globalAccessRight [int - in] param: address [string - in] param: photo [string - in] param: speciality [string - in] param: gender [int - in] param: birthDate [date - in] param: email [string - in] param: phone [string - in] param: lastName [string - in] param: firstName [string - in] param: password [string - in] param: userId [string - in] 'createUser' method creates a new user with the specified attributes. It updates the corresponding database tables to do this.
ceateNewTask (string, string, string, string, int, string, int, int, string, int, int, date, date, date, string, string, string)	public: void	param: atachedFile4 [string - in] param: attachedFile3 [string - in] param: attachedFile2 [string - in] param: attachedFile1 [string - in] param: groupId [int - in] param: reviewerId [string - in] param: percentDone [int - in] param: statusId [int - in] param: projectId [string - in] param: priorityId [int - in] param: typeId [int - in] param: dueDate [date - in] param: finishDate [date - in] param: startDate [date - in] param: taskDescription [string - in] param: taskName [string - in] param: taskId [string - in]

		'createNewTask' method creates a new task with the specified attributes. It updates the corresponding database tables to do this.
createMaterial (<i>string, double, string, string</i>)	public: void	param: materialDescription [string - in] param: materialCost [double - in] param: materiaName [string - in] param: materialId [string - in] 'createMaterial' method is used to create new materials with the specified attributes. It updates the corresponding database tables to do this.
createMeeting (<i>date, string, string, string, date, date, date, date, date, date, string</i>)	public: void	param: lastReplyDate [date - in] param: attachement3 [string - in] param: attachement2 [string - in] param: attachement1 [string - in] param: dateOption5 [date - in] param: dateOption4 [date - in] param: dateOption3 [date - in] param: dateOption2 [date - in] param: dateOption1 [date - in] param: finalMeetingDate [date - in] param: meetingId [string - in] 'createMeeting' method is used to create a new meeting with the specified attributes. It updates the corresponding database tables to do this.
buysMaterial (<i>double, double, string</i>)	public: void	param: unitPrice [double - in] param: quantity [double - in] param: materialId [string - in] 'buysMaterial' method marks the database so that the purchase information is recorded.
setUserPrefencesForMeeting (<i>string, int, int, int, int, int</i>)	public: void	param: meetingId [string - in] param: option5 [int - in] param: option4 [int - in] param: option3 [int - in] param: option2 [int - in] param: option1 [int - in] 'setUserPreferencesForMeeting' method marks the database according to the preferences made by this user for a meeting he/she was assigned to.
approveTask ()	public: void	'approveTask' method is called when the reviewer approves a task of another user. The database tables are marked accordingly.
assignToProject	public: void	param: projectId [string - in]

(string, boolean, boolean, int)		param: isProjectManager [boolean - in] param: isTimeApprover [boolean - in] param: taskEditLevel [int - in] 'assignToProject' method marks the database so that this user is assigned to the specified project with the projectId.
createCompany (string, string, int, string, string, string)	public: void	param: companyName [string - in] param: companyAddress [string - in] param: weekManagementPolicy [int - in] param: webPageAddress [string - in] param: phoneNo [string - in] param: companyLogoPath [string - in] 'createCompany' method is called when this user wants to create a new company. If the user has the required access rights, a new entry for the company is created in the database.
sendNotification (string, string, string, string, string)	public: void	param: notificationId [string - in] param: notifiedUser [string - in] param: attachedFile1 [string - in] param: attachedFile2 [string - in] param: attachedFile3 [string - in] 'sendNotification' method is called, if this user wants to send a notification to another user. The database tables are marked accordingly, so that when the to-be-notificated user logs-in to the system the next time she/he will see the notification.

Calendar

Type: public **Class**

Status: Proposed. Version 1.0. Phase 1.0.

Package: Component Model

Details: Created on 09.01.2005 14:41:20. Modified on 09.01.2001 23:53:22.

'Calendar' class represents a calendar of a specific user in DProject. It will be presented through a calendar-like visual interface. The user will be able to see calendar items (which also constitutes a class in DProject) which are either created automatically by the system or manually by the user. This class will also handle the items that are to be reminded on a specific date.

Calendar Attributes

Attribute	Type	Notes
-----------	------	-------

items	private <i>CalendarItem</i> []	: 'items' are the collection of CalendarItem instances in this calendar. These are the items belonging to the owner of the calendar.
ownerId	private <i>string</i>	: 'ownerId' is the id of the user that owns this calendar.
periodStart	private <i>date</i>	: 'periodStart' is the start date of the period for which the calendar information are held.
periodEnd	private <i>date</i>	: 'periodEnd' is the end date of the period for which the calendar information are held.

Calendar Methods

Method	Type	Notes
setReminder (<i>CalendarItem</i>)	public: <i>void</i>	param: item [<i>CalendarItem</i> - in] 'setReminder' method marks the database table of the specified calendarItem object, so that the user will be reminded when on the date of the calendarItem.
unsetReminder (<i>CalendarItem</i>)	public: <i>void</i>	param: item [<i>CalendarItem</i> - in] 'unsetReminder' method marks the database so that a formerly set reminder item is unset.
addCalendarItem (<i>CalendarItem</i>)	public: <i>void</i>	param: item [<i>CalendarItem</i> - in] This method adds a new calendar Item to calendar

CalendarItem

Type: *public* **Class**

Status: Proposed. Version 1.0. Phase 1.0.

Package: Component Model

Details: Created on 09.01.2005 15:04:14. Modified on 09.01.2001 23:56:26.

'CalendarItem' class represents a single item in the calendar of a user. It may be created automatically by the DProject system in cases of task deadlines, project deadlines and meeting dates. Or it may be created manually by the user for any purpose.

CalendarItem Attributes

Attribute	Type	Notes
itemId	private <i>string</i>	: 'itemId' is the unique id of the CalendarItem object in DProject. It is also the primary key used in the database for this CalendarItem record.
itemDate	private	: 'itemDate' is the date of this calendarItem object.

	<i>date</i>	
text	private <i>string</i>	: 'text' is the text which will be displayed in the calendar, in place of this calendarItem.
willBeReminded	private <i>bool</i>	: 'willBeReminded' specifies if this calendarItem should be automatically reminded to the user when its time comes. The value 'true' means the item will be reminded, 'false' means the item will not be reminded.

CalendarItem Methods

Method	Type	Notes
insert ()	public: <i>void</i>	'insert' method writes the information in this 'CalendarItem' class instance to the database, creating a new entry in the database table.
update ()	public: <i>void</i>	'update' method updates the record of this 'CalendarItem' in the database, using the current values of the attributes.
delete ()	public: <i>void</i>	'delete' method deletes the record of this CalendarItem from the database.

Notification

Type: *public* **Class**

Status: Proposed. Version 1.0. Phase 1.0.

Package: Component Model

Details: Created on 04.12.2004 14:06:47. Modified on 09.01.2001 21:16:52.

'Notification' class represents a means of communication between users of DProject. A notification can be sent to a user by another user, or by the system automatically. 'Notification' class is created, modified, and deleted accordingly.

Notification Attributes

Attribute	Type	Notes
notificationId	private <i>string</i>	: 'notificationId' is the unique id a notification gets in DProject. This is also the primary key used in the database entry for notifications.
notificatedUser	private <i>string</i>	: 'notificatedUser' is the id of the user this notification is aiming at.
notificationType	private <i>string</i>	: 'notificationType' is the type of this notification. It can be a meeting notification, a task notification, or a message notification.

ownerOfAction	private <i>string</i>	:	'ownerOfAction' is the user who creates the reason of this notification being sent. If it is a meeting notification, this is the arranger of the meeting; if it is a task notification, it is the creator of the task; if it is a message notification, it is the sender of the message.
dateOfAction	private <i>date</i>	:	'dateOfAction' is the date of the event that caused this notification to be sent.
attachedFile1	private <i>string</i>	:	'attachedFile1' represents the path of the file that is attached to this notification.
attachedFile2	private <i>int</i>	:	'attachedFile2' represents the path of the file that is attached to this notification.
attachedFile3	private <i>string</i>	:	'attachedFile3' represents the path of the file that is attached to this notification.

Notification Methods

Method	Type	Notes
insert ()	public: <i>void</i>	'insert' method writes the information in this 'Notification' class instance to the database, creating a new entry in the database table.
update ()	public: <i>void</i>	'update' method updates the record of this 'Notification' in the database, using the current values of the attributes.
delete ()	public: <i>void</i>	'delete' method deletes the record of this Notification from the database.

Forum

Type: *public* **Class**

Status: Proposed. Version 1.0. Phase 1.0.

Package: Component Model

Details: Created on 04.12.2004 14:07:04. Modified on 07.01.2005 03:30:57.

'Forum' class represents a forum that presents a means of communication between the users of the same company. A forum object is created in DProject whenever a user wants to inspect the Forum messages, or compose one.

Forum Attributes

Attribute	Type	Notes
forumThreadIds	private <i>string []</i>	'forumThreadIds' is an array of strings, that are the ids of the ForumThread objects, in this specific Forum.

Forum Methods

Method	Type	Notes
getForumThreadIds ()	public: <i>string []</i>	'getForumThreadIds' returns the ForumThread object ids that belong to this Forum.
getForumThreadTitles ()	public: <i>string []</i>	'getForumThreadTitles' returns the titles of the threads of this Forum in a string array. This method is called whenever user wants to see the thread titles in the forum.

ForumMessage

Type: *public* **Class**

Status: Proposed. Version 1.0. Phase 1.0.

Package: Component Model

Details: Created on 06.12.2004 20:35:57. Modified on 07.01.2005 03:29:16.

'ForumMessage' class is the class that represents a single forum message in DProject. The ForumMessage class is created in DProject whenever a user wants to create a forum message, or inspect an existing one.

ForumMessage Attributes

Attribute	Type	Notes
senderId	private : <i>string</i>	'senderId' is the id of the user that sent the message.
subject	private : <i>string</i>	'subject' is the subject of the message supplied by the creator of the message.
messageBody	private : <i>string</i>	'messageBody' is the actual message sent by the user.

ForumMessage Methods

Method	Type	Notes
setMessageBody (<i>string</i>)	public: <i>void</i>	param: messageBody [<i>string</i> - in]

ForumThread

Type: *public* **Class**

Status: Proposed. Version 1.0. Phase 1.0.

Package: Component Model

Details: Created on 06.12.2004 20:35:47. Modified on 07.01.2005 03:25:20.

'ForumThread' is a collection of Forum messages in a forum. It has a specific title, and constitutes a categorization between forum messages. A ForumThread object is created in DProject whenever a user wants to inspect or create a new thread.

ForumThread Attributes

Attribute	Type	Notes
threadId	private : <i>string</i>	'threadId' is the unique id a forum thread object gets in DProject. It is also the primary key value of the database entry of this object.
title	private : <i>string</i>	'title' is the title of the ForumThread object.
forumMessageIds	private : <i>string []</i>	'forumMessageIds' is the array of ids of the forum messages belonging to this thread.

ForumThread Methods

Method	Type	Notes
getForumThreadTitle() ()	public: <i>string</i>	Returns the ForumThread's title.
getForumMessageSubjects() ()	public: <i>string []</i>	'getForumMessageSubjects' returns the subjects of the messages that belongs to this thread.

ProjectResource

Type: *public* **Class**

Status: Proposed. Version 1.0. Phase 1.0.

Package: Component Model

Details: Created on 04.12.2004 14:07:30. Modified on 09.01.2001 21:16:52.

'ProjectResource' is a quantity of some material that is dedicated to a specific project. For a material to be a ProjectResource, it first has to be purchased. During the purchase, the cost of this resource is calculated and decreased from the project budget. A ProjectResource instance is created in DProject, when a user purchases a material for some project.

ProjectResource Attributes

Attribute	Type	Notes
materialId	private : <i>string</i>	'materialId' specifies what type of material is included in this project resource.
quantity	private : <i>double</i>	'quantity' specifies the quantity of the material.
assignedProjectId	private : <i>string</i>	'assignedProjectId' specifies which project this resource belongs to.

ProjectResource Methods

Method	Type	Notes
setMaterialId (<i>string</i>)	public: <i>void</i>	param: materialId [<i>string</i> - in] Sets the materialId field to the specified value.
getMaterialId ()	public: <i>string</i>	returns the material id of this projectResource.
setQuantity (<i>double</i>)	public: <i>void</i>	param: quantity [<i>double</i> - in] Sets the quantity field to the specified value.
getQuantity ()	public: <i>double</i>	Returns the value in the quantity field.
setAssignedProjectId (<i>string</i>)	public: <i>void</i>	param: projectId [<i>string</i> - in] 'setAssignedProjectId' method sets the assignedProjectId field of this resource, which specifies the project that this resource belongs to.
getAssignedProjectId ()	public: <i>string</i>	Returns the value of the assignedProjectId field.

Material

Type: *public Class*

Status: Proposed. Version 1.0. Phase 1.0.

Package: Component Model

Details: Created on 04.12.2004 14:07:22. Modified on 07.01.2005 03:30:57.

'Material' class represents a specific material type that has been created to be used in projects. A material can be created in DProject for use in several projects of a company. For a material to be a projectResource, it first has to be purchased. This purchase is done according to the information held in Material object.

Material Attributes

Attribute	Type	Notes
materialId	private <i>string</i>	: 'materialId' is the unique id a material object gets in DProject. This is also the primary key value used in the database table for this material.
materialName	private <i>string</i>	: 'materialName' is the name of the material specified by the creator of the material.
materialCost	private <i>double</i>	: 'materialCost' is the current unit price of this material in the market. This attribute is used when a purchase of this material is being done.
materialDescription	private <i>string</i>	: 'materialDescription' is information supplied by the creator of the material about the material.

Session 'hasActive' Company

For every session that is started for a user, we will hold an instance of the Company class which represents the company of the logged-in user.

Session 'hasLoggedIn' User

For every session, we will hold an instance of the User class which represents the logged-in user.

Session 'hasActive' User

During a session, if the user wants to create a new user, or wants to modify/delete the records of an existing user, then the user whose records are being modified (or created) will be held as the 'currentUser' in the session. This determines the 'hasActive' relationship.

Session 'hasActive' Project

During a session, if the user wants to create a new project, or wants to modify/delete the records of an existing project, then the project whose records are being modified (or created) will be held as the 'currentProject' in the session. This determines the 'hasActive' relationship.

Session 'hasActive' Task

During a session, if the user wants to create a new task, or wants to modify/delete the records of an existing task, then the task whose records are being modified (or created) will be held as the 'currentTask' in the session. This determines the 'hasActive' relationship.

Session 'hasActive' Meeting

During a session, if the user wants to create a new meeting, or wants to modify/delete the records of an existing meeting, then the meeting whose records are being modified (or created) will be held as the 'currentMeeting' in the session. This determines the 'hasActive' relationship.

Session 'hasActive' ForumThread

During a session, if the user wants to view the contents of a forum thread, then the forum thread whose records are being viewed will be held as the 'currentForumThread' in the session. This determines the 'hasActive' relationship.

Session 'hasActive' ForumMessage

During a session, if the user wants to create a new forum message, or wants to view an existing message, then the message whose records are being created (or viewed) will be held as the 'currentForumMessage' in the session. This determines the 'hasActive' relationship.

Session 'hasActive' Material

During a session, if the user wants to create a new type of material, or wants to modify/delete the records of an existing material, then the material whose records are being modified (or created) will be held as the 'currentMaterial' in the session. This determines the 'hasActive' relationship.

Session 'hasActive' Notification

During a session, if the user wants to create a new notification, or wants to view those notifications (which may be more than one), then, the notifications whose records are being created or being viewed will be held as the 'currentNotifications' in the session. This determines the 'hasActive' relationship.

Initializer 'initializes' Session

For every session that is started for a user, an Initializer class is held to handle the initializations both before and after the login. This determines the 'initializes' relationship.

SqlConnection 'connects' Session

For every session that is started for a user, a SqlConnection class is held to handle the database connections both before and after the login. This determines the 'connects' relationship.

Project 'includes' Task

For every project there are zero or more tasks that belong to the project.

Forum 'consistsOf' ForumThread

In a Forum, there may be zero or more ForumThreads. That is; a forum consists of threads. This also determines the aggregation character of the association.

ForumThread 'consistsOf' ForumMessages

In a ForumThread, there may be one or more ForumMessages. That is; a forum thread consists of messages. This also determines the aggregation character of the association.

ProjectResource 'consistsOf' Material

Every project resource consists of some material. There may be only one material type in a project resource.

Calendar 'is composed of' CalendarItem

Every Calendar consists of zero or more CalendarItem.

Task 'has TaskType in' Project

Every Task has a specified TaskType in a Project. This constitutes a categorization of tasks in a project.

Task 'has TaskPriority in' Project

Every Task has a specified TaskPriority in a Project. This constitutes a categorization of tasks in a project in terms of priority.

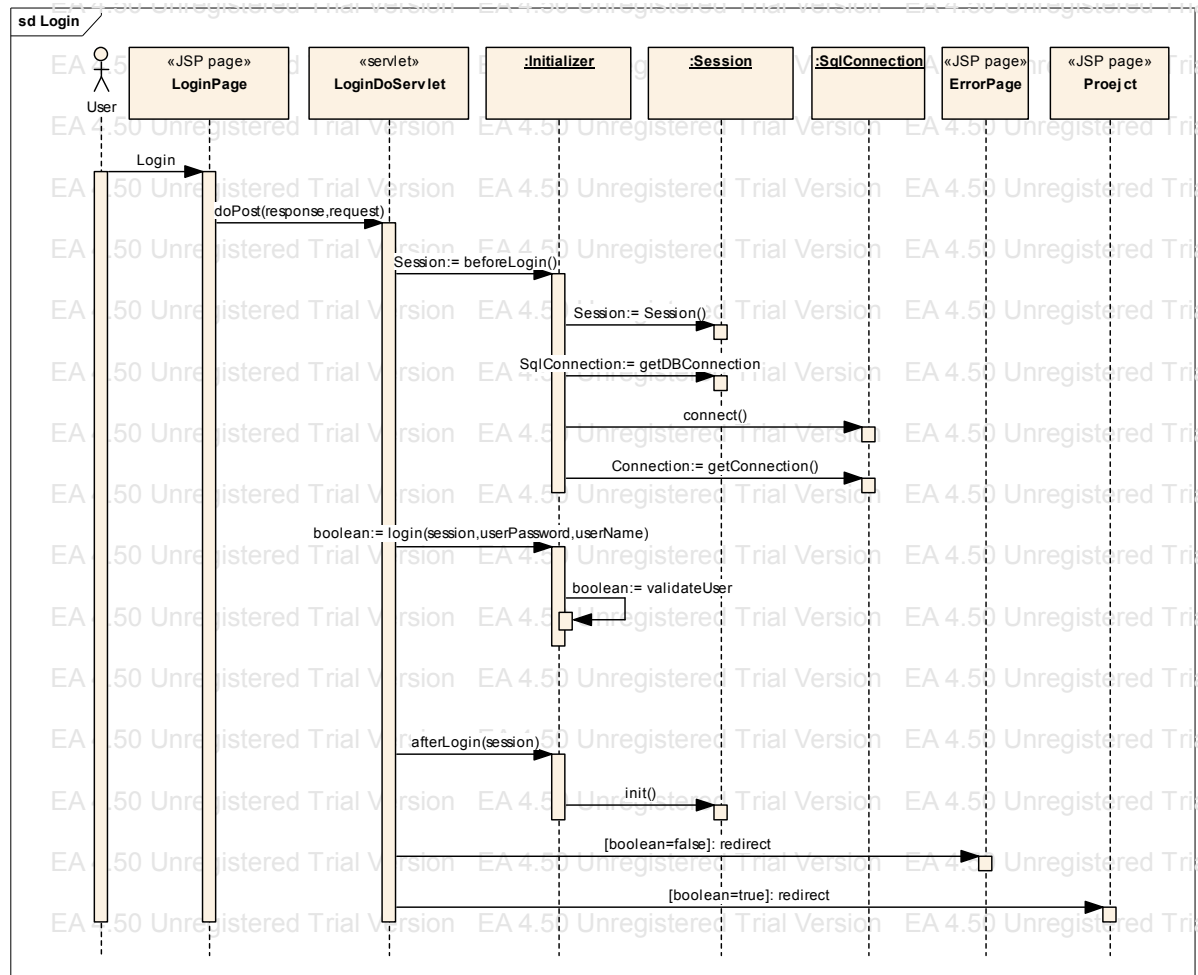
Task 'has TaskStatus in' Project

Every Task has a specified TaskStatus in a Project. This constitutes a categorization of tasks in a project in terms of the progress of the task.

3.3 DYNAMIC VIEW

The sequence diagrams of our system are as follows:

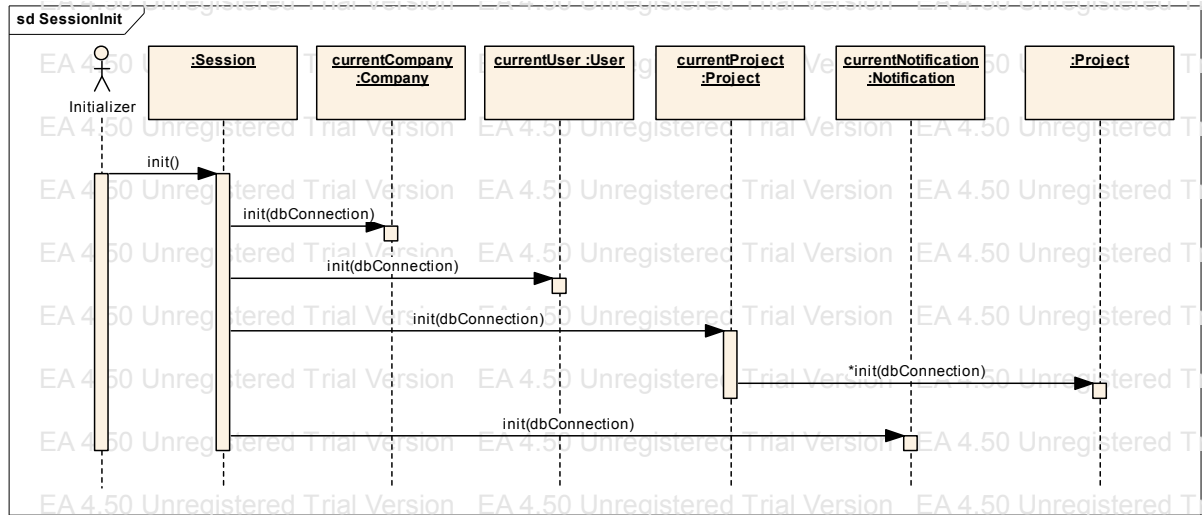
1) Diagram: Login



This diagram specifies the session initialization and login procedure in DProject for every user. User starts with the page 'Login.jsp' and after entering his 'company_name', 'user_id' and 'password', the page calls the doPost method of the 'LoginServlet'. Servlet calls the 'beforeLogin()' operation of the 'Initializer' class. This operation constructs a session and creates connection for database by using the SqlConnection class. Session object is set as 'HtmlSession' attribute and obtained from there for every page so its lifetime is entire session. Then servlet calls the 'Initializer.login()' operation of the 'Initializer' that checks the user login and password and returns whether user is authorized or not in which an error page is shown by the system. After that, authorization servlet calls the 'afterLogin' operation and this makes the necessary initialization for the Session variables. We show this initialization part at

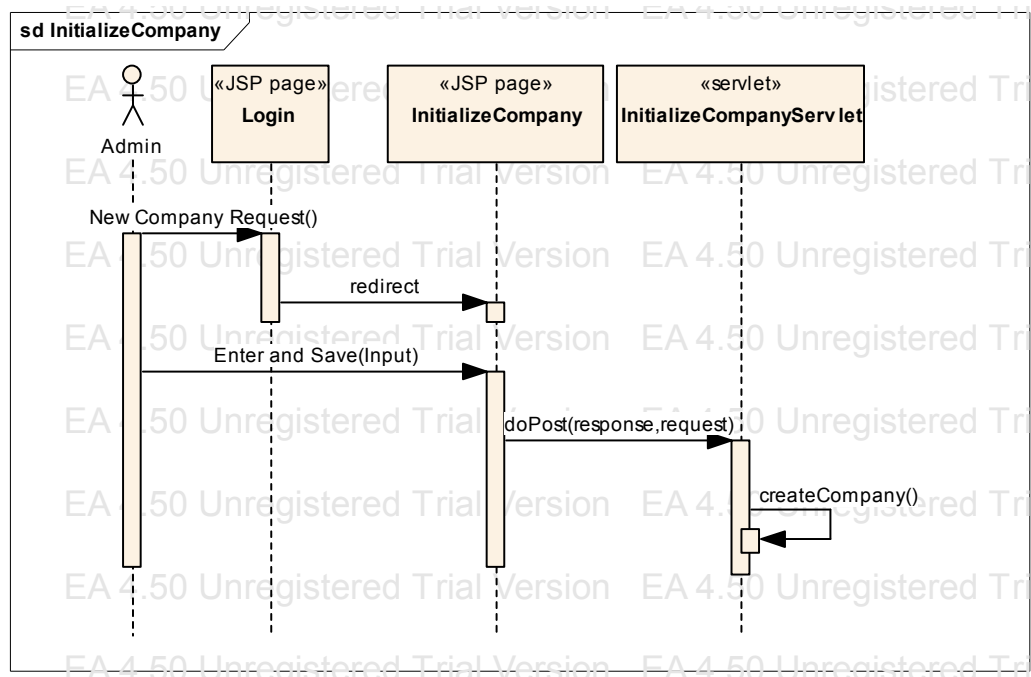
Session Initialization diagram. Finally, the page is redirected to some jsp page which is optional for users.

2) Diagram: SessionInit



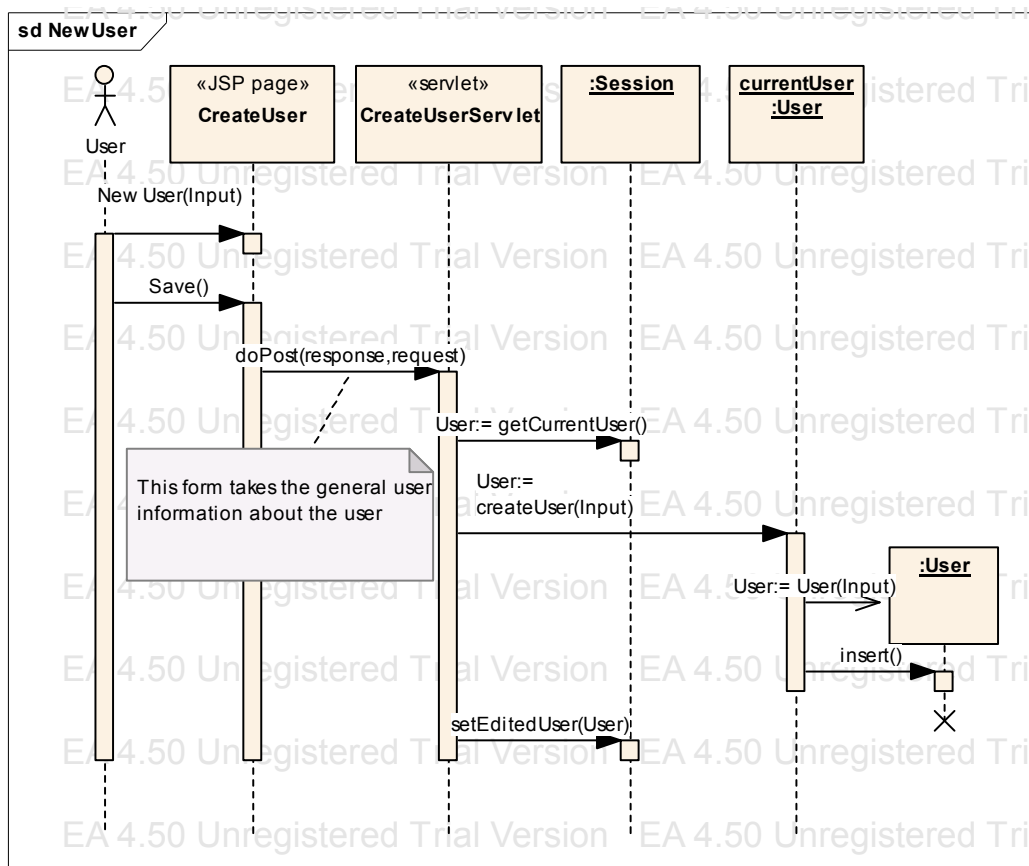
This diagram shows the initialization process of the DProject. 'Session.init()' calls its variables' 'init()' operations so all session variables that will be used in process are initialized.

3) Diagram: InitializeCompany



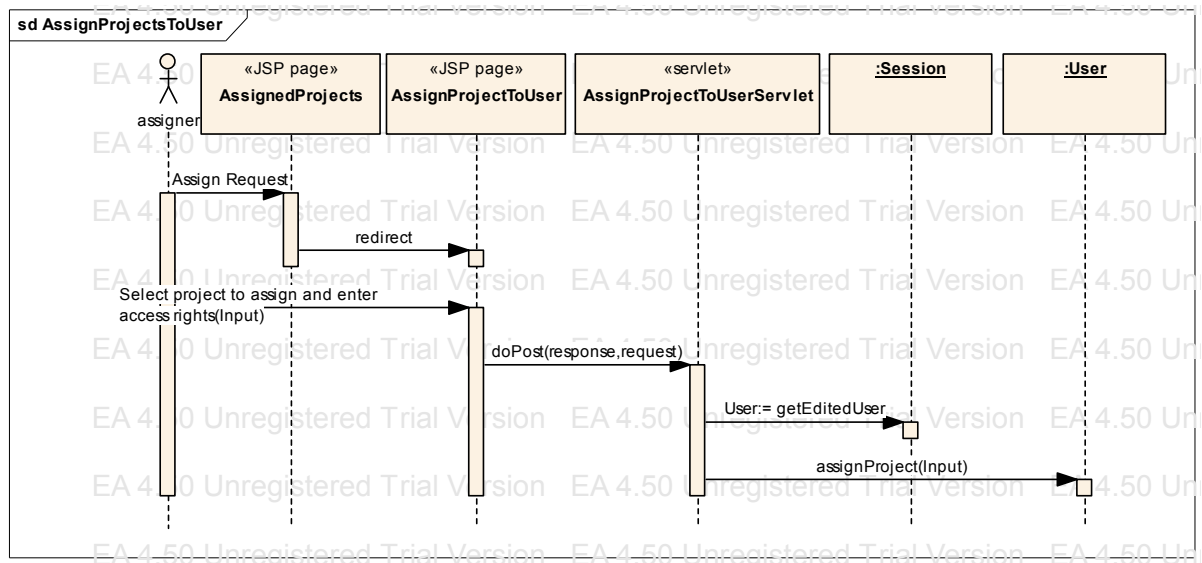
Company initialization starts from the 'Login.jsp' page when user requests to initialize his company for the first time. He starts this action with the given company login and password for first login. When he enters this information, login page is redirected to the company initialization page. User gives the necessary input and save the input. Then input is send to 'InitializeCompanyServlet' servlet which will generate the company database and all tables in it. Then user is redirected to main page.

4) Diagram: NewUser



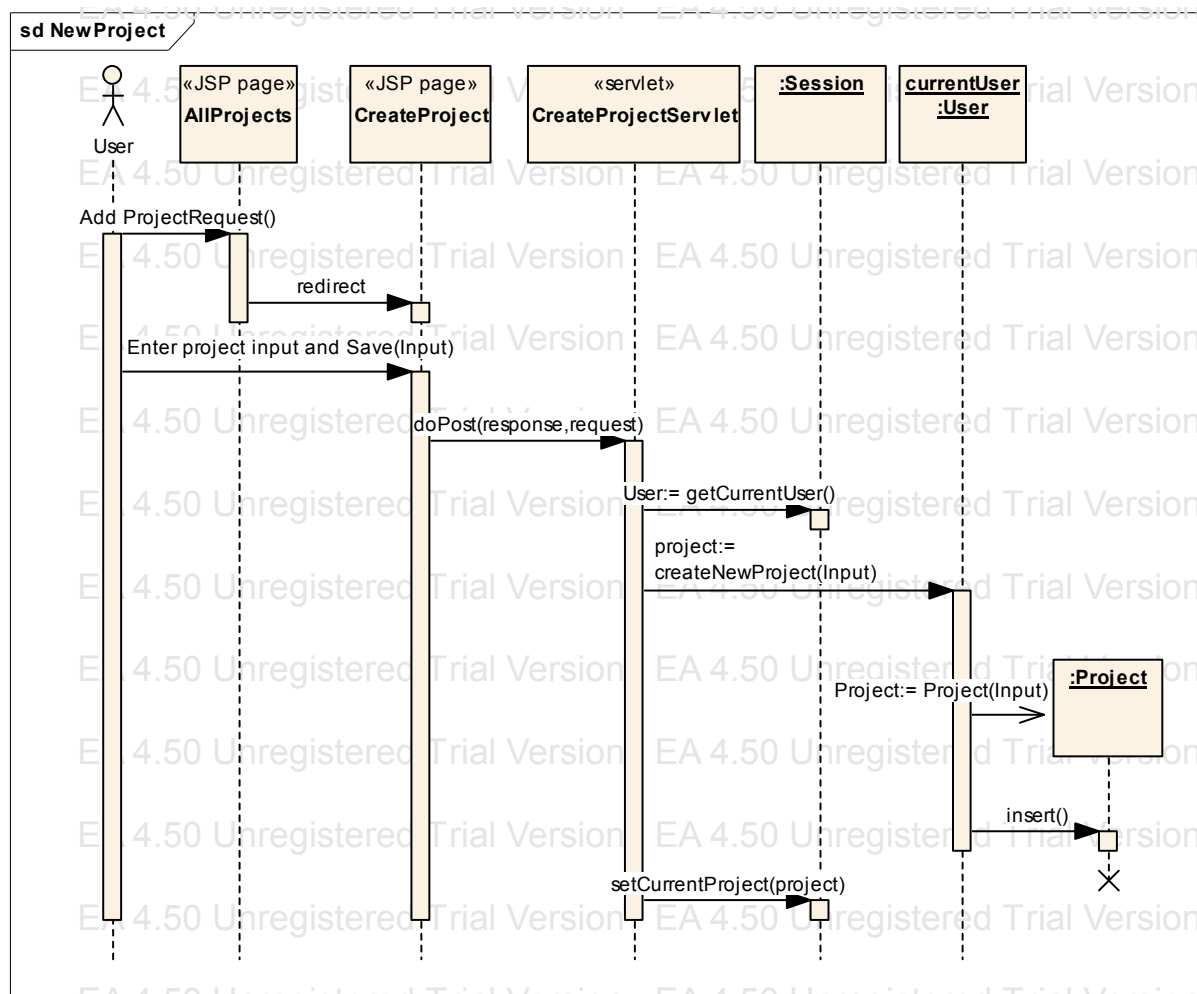
Like above, this diagram shows the complete process of creating a user which also includes the assignation of the created user to some projects. The general information is used for creating the user by the operation 'createUser' of the 'currentUser' object. This operation creates the user object and calls its insert method to insert the user into database and sets the 'editedUser' to this user. In the second phase 'AssignNewUserServlet' takes the 'editeUser' from session and calls the 'assignToProject' operation of this object.

5) Diagram: AssignProjectsToUser



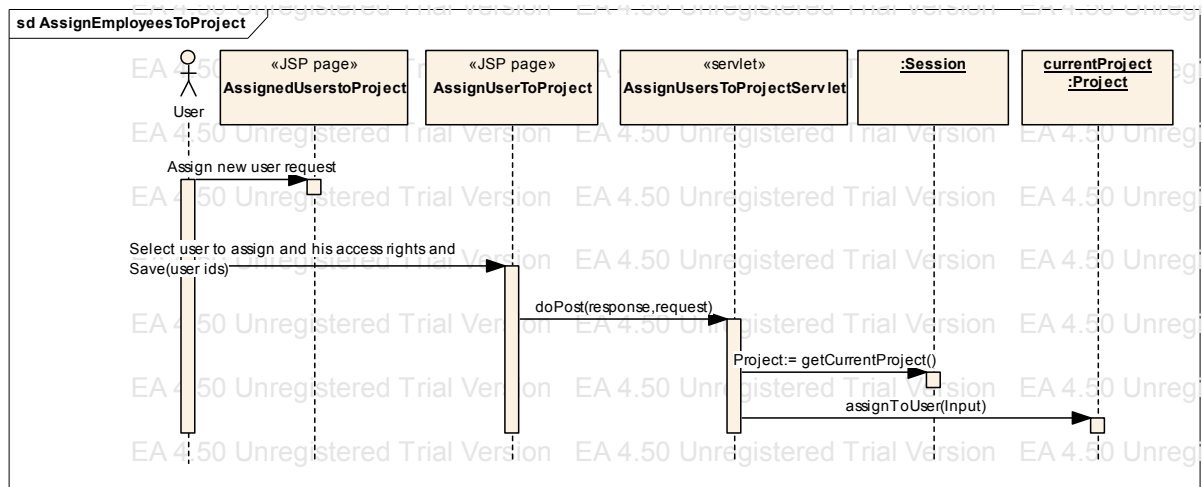
User can see projects that are assigned to some user on 'AssignedProjects.jsp' page. When he assign a new project to user all information is sent to AssignProjectToUsersServlet. Servlet takes the 'editedUser' object from the session and calls 'assignProject()' method of this object.

6) Diagram: Project



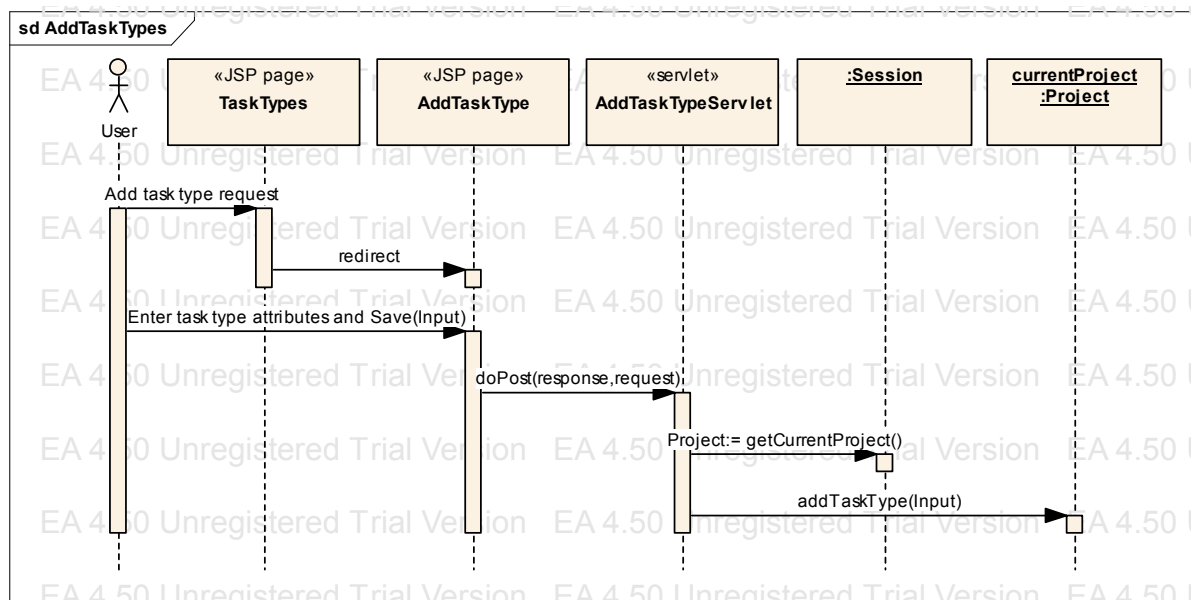
User requests to add project from 'projects.jsp' page which shows all projects in company. Then 'CreateProject.jsp' is redirected. User enters general information about project and pressed save button to create this project. (See the interface and project class for the entered information) Then page sends this input to 'CreateprojectServlet' for necessary actions by the doPost method. Servlet takes the current user from the session object. (Current user is the user who now creates the project) Then servlet calls the 'createNewProject()' method of the currentUser object. This method create a project object by the given input and returns this object. Then servlet sets the 'currentProject' object of 'session' by returned project.

7) Diagram: AssignEmployeesToProject



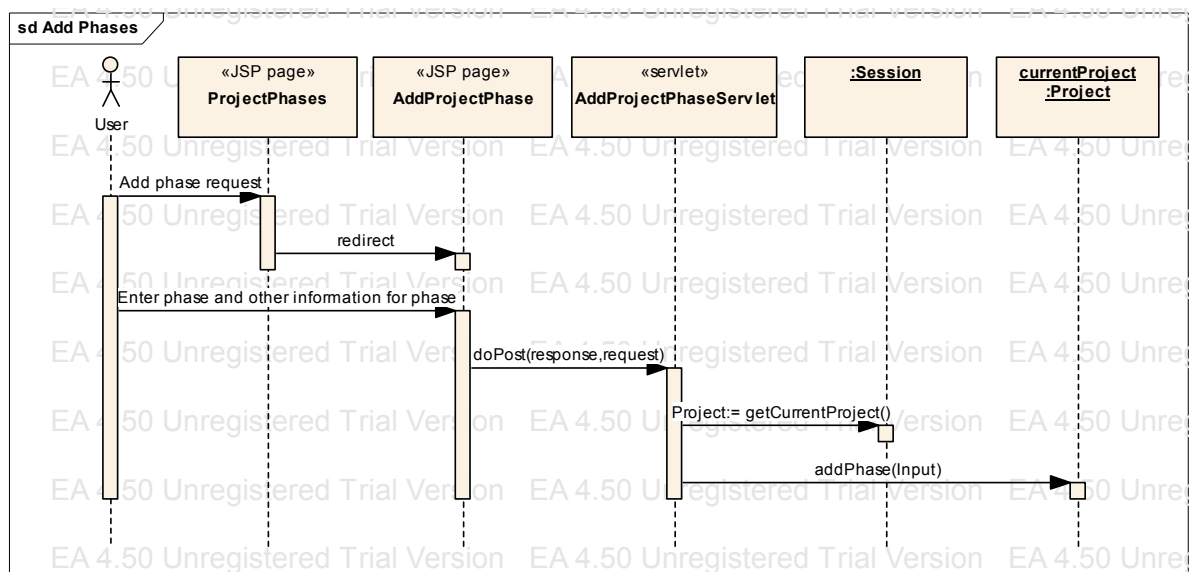
User can be redirected 'AssignedUsers.jsp' from 'Project.jsp' which shows the selected project information. 'AssignedUsers.jsp' shows the currently assigned users for that project. When the user request to assign a new user to project page redirects to 'AssignUserToProject.jsp'. User select a user and his project specific rights and save this information. Then page send this input to AssignUserToProjectServlet by doPost method of the servlet. Servlet takes the 'currentProject' object of 'session' and calls the 'assignToUser()' method of this object to save information in database.

8) Diagram: AddTaskTypes



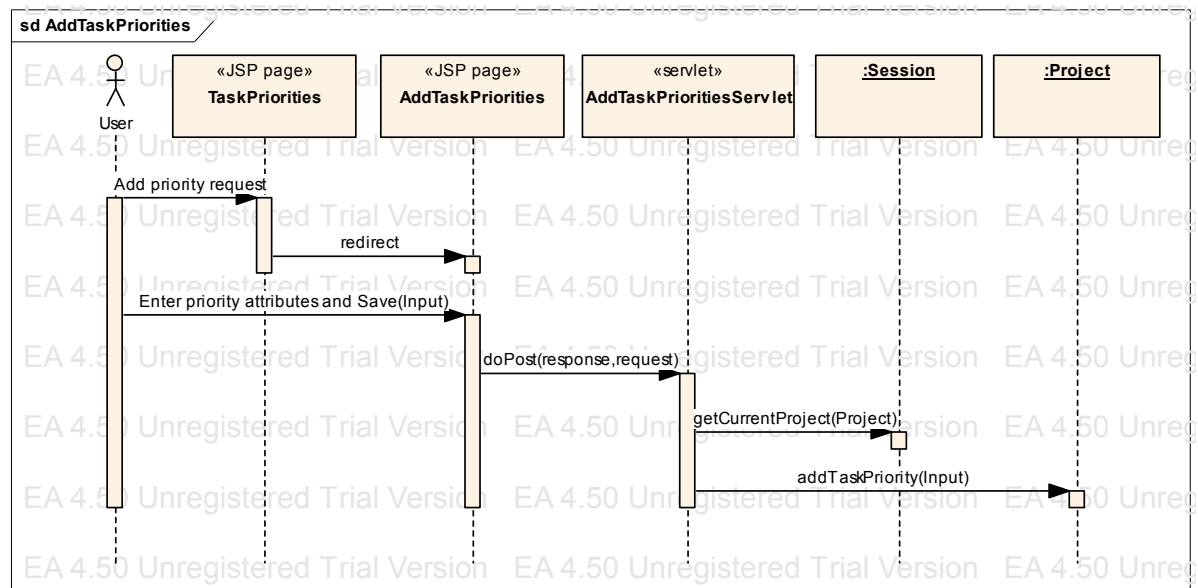
User can be redirected to 'TaskTypes.jsp' from 'Project.jsp' by pressing the tab. 'TaskTypes.jsp' shows the task types that project currently have. When user requests to define new task type for the project page is redirected to 'AddTaskType.jsp'. User enters the information for the new task type and save it. Then page sends this information to 'AddTaskTypeServlet' servlet for the actions. Servlet takes the 'currentProject' object from the 'session' object and calls 'addTaskType()' method of this object which saves the new task type to database.

9) Diagram: Add Phases



User can be redirected to 'TaskPhases.jsp' from 'Project.jsp' by pressing the tab. Since the sequence of actions are very similar to 'Add Task Types' sequence we do not need to explain again.(See Add Task Types sequence)

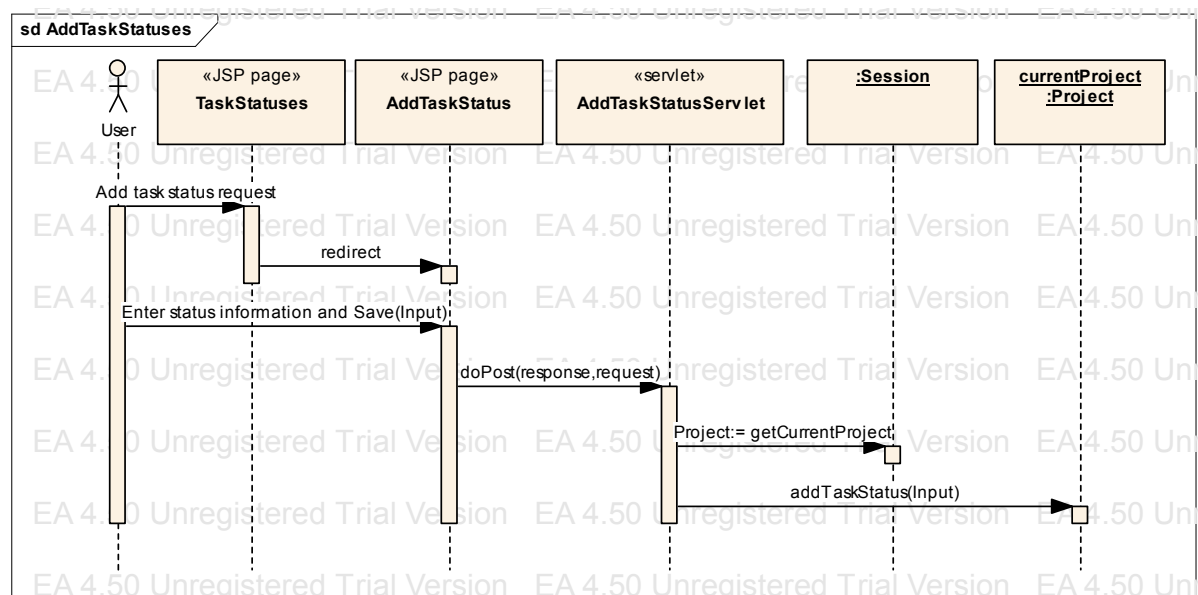
10) Diagram: AddTaskPriorities



User can be redirected to 'Taskpriorities.jsp' from 'Project.jsp' by pressing the tab.

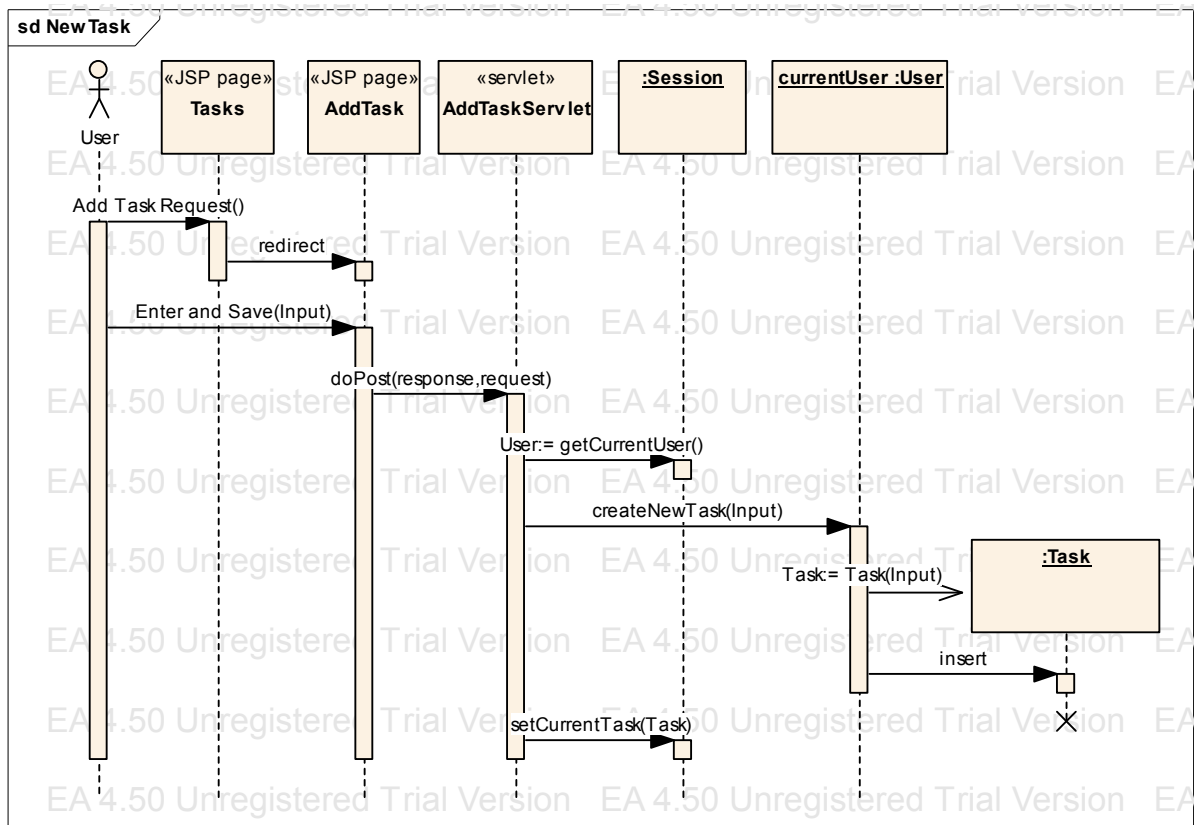
Since the sequence of actions are very similar to 'Add Task Types' sequence we do not need to explain again.(See Add Task Types sequence)

11) Diagram: AddTaskStatuses



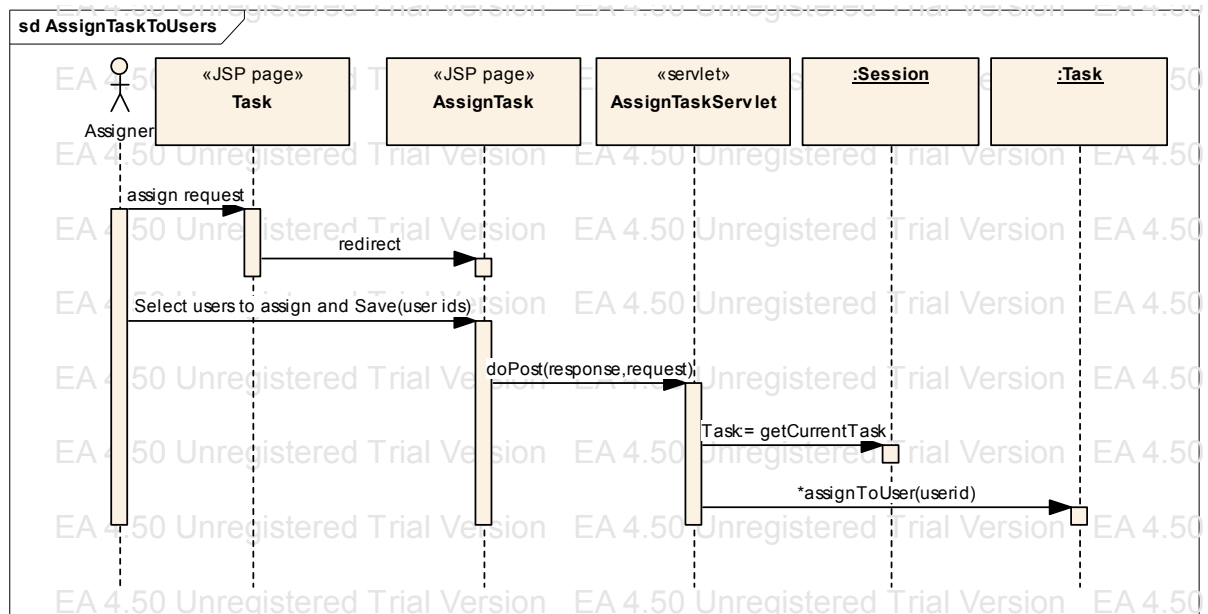
User can be redirected to 'TaskStatuses.jsp' from 'Project.jsp' by pressing the tab. Since the sequence of actions are very similar to 'Add Task Types' sequence we do not need to explain again.(See Add Task Types sequence)

12) Diagram: NewTask



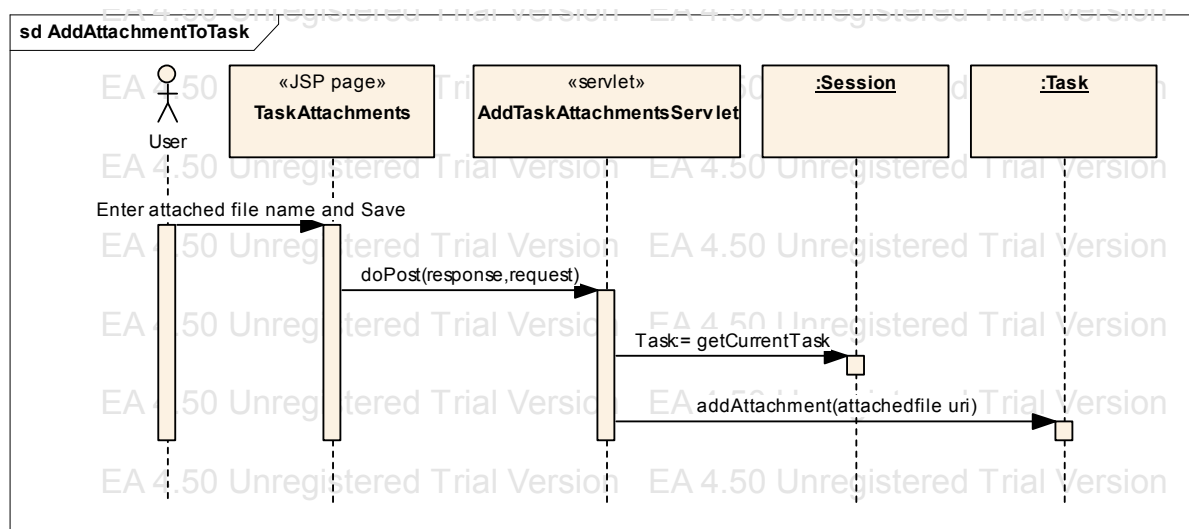
User can request add a new task from the 'Tasks.jsp' which shows the tasks of the opened project. Then page is redirected to 'AddTask.jsp' page to take the input about general information for the task. When user save the information page sends the input to 'AddTaskServlet' servlet by doPost method. Servlet get the 'currentUser' object from the 'session' object. Then servlet calls the 'createNewTask()' method of the 'currentUser' object. This method create a new 'Task' object by the input and calls the insert method of this object to save to the database. Then method returns the created object to servlet. Servlet sets the 'currentTask' object of session to this returned object.

13) Diagram: AssignTaskToUsers



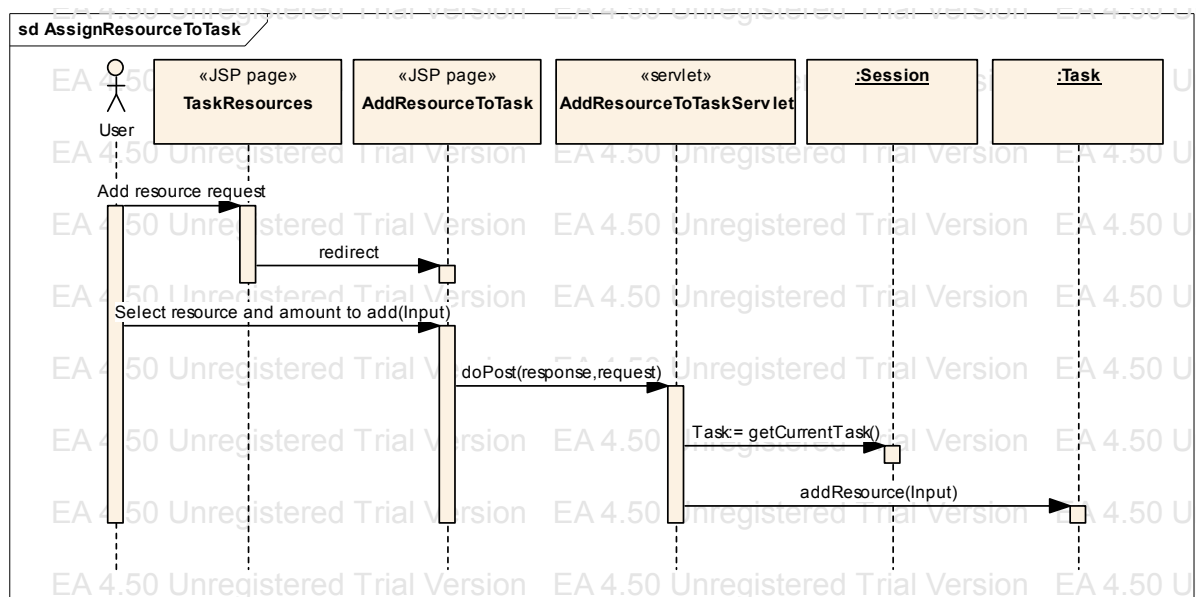
User can request to assign task to users from 'Task.jsp' which shows the selected task information. Then page is redirected to 'AssignTask.jsp'. User can select the users to assign and save the information. Then page sends this input to 'AssignTaskServlet' by doPost method. Servlet take the 'currentTask' object from the 'session'. Then it calls 'assignToUser()' method iterated way for all users so method saves this information to database.

14) Diagram: AddAttachmentToTask



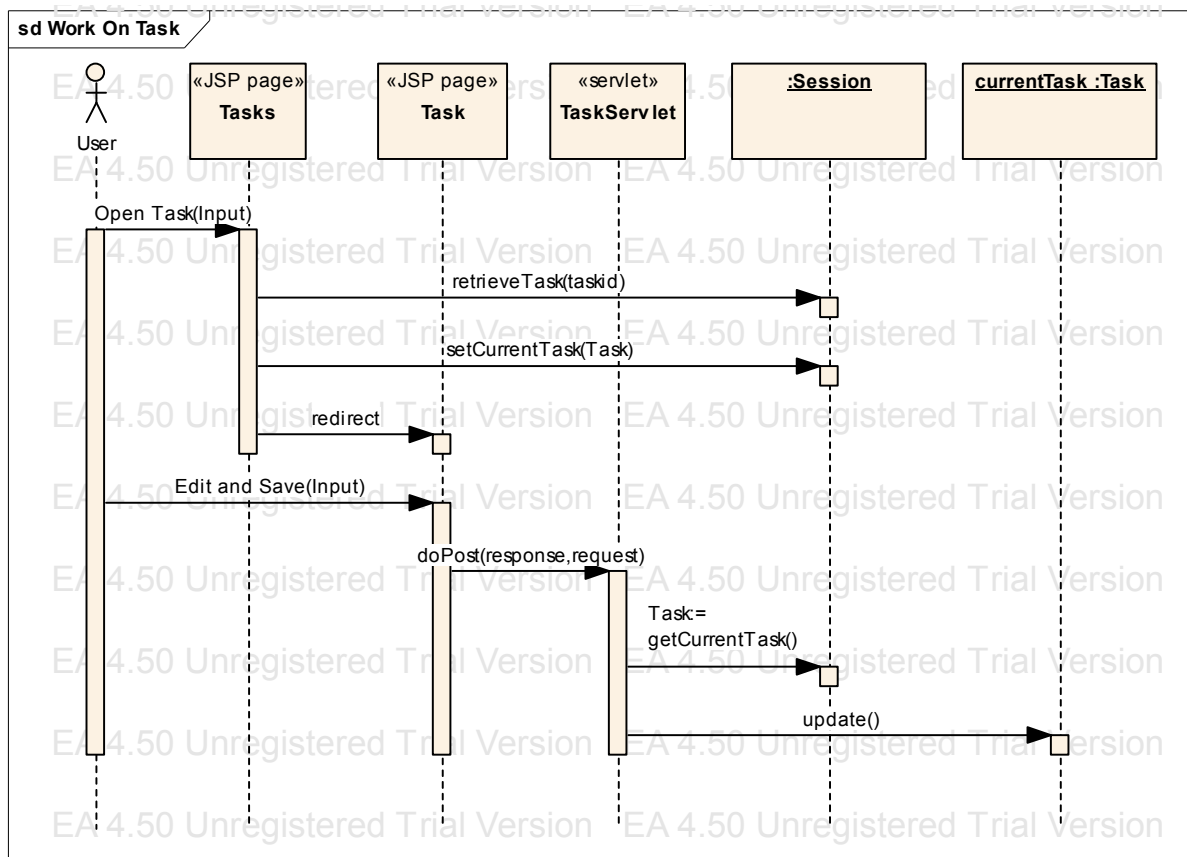
User can see the currently attached files on 'TaskAttachments.jsp'. When he wants to add a new attachment he enters the necessary input and pressed to add button. Then input is send to 'AddTaskAttachmentsServlet' servlet. Servlet takes the 'currentTask' object from session and calls 'addAttachment()' method of the object. Method saves the filename of attachment to database and read the attached file and copy it to the file system of the system.

15) Diagram: AssignResourceToTask



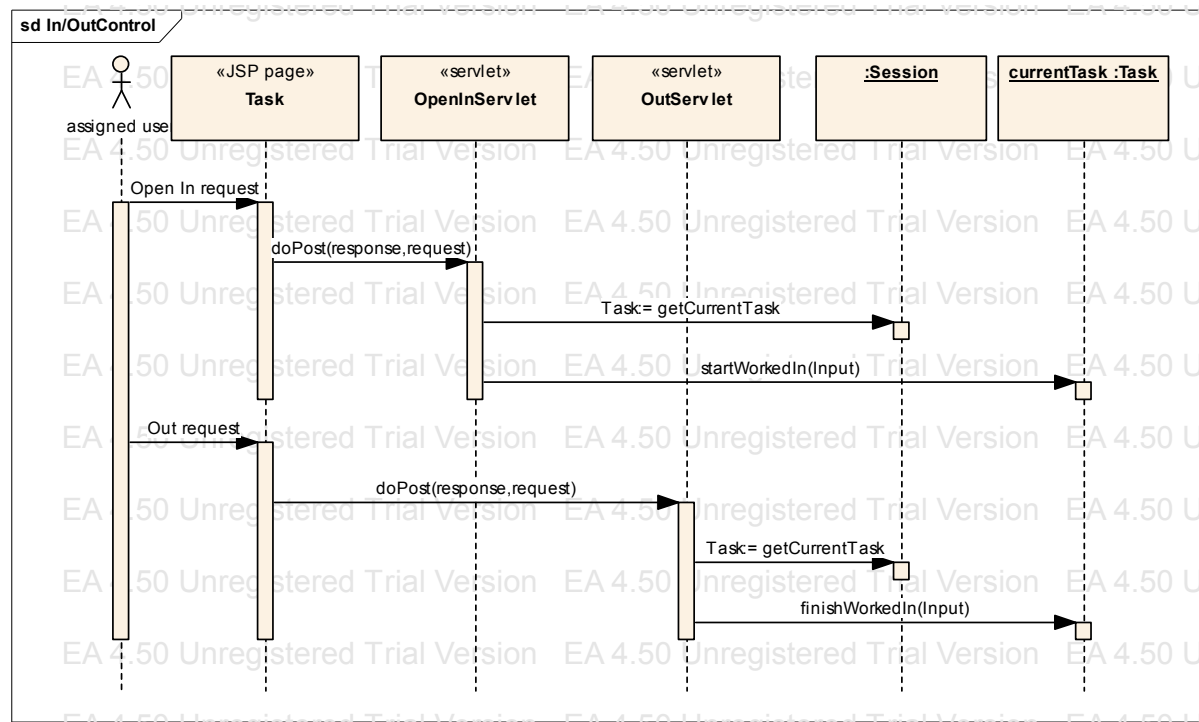
User can request to bind some resource to task from 'Task.jsp' that is used while doing this task. By this request page is redirected to 'AddResourceToTask.jsp'. User select a material which exists in project currently and amount of the material and add to task. Then input is send to 'AddResourceToTaskServlet' servlet. Servlet takes the 'currentTask' object from session and calls addResource() method of the object. Method decrease the amount of material in project resource and add it to task by changing database.

16) Diagram: Work On Task



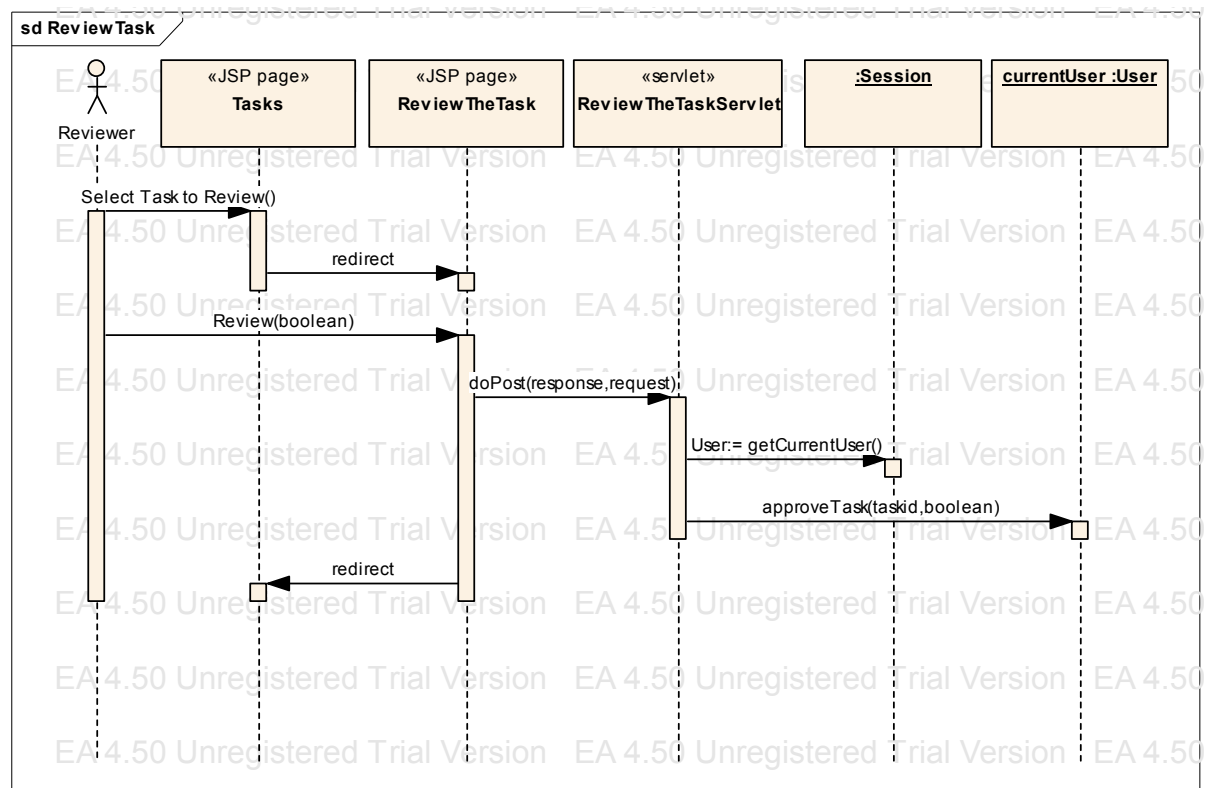
Users can open the task from the 'Tasks.jsp' page which is assigned to them. Then 'Task.jsp' page is shown which shows the information for that task. User can edit some fields which he is allowed to and save the changes. Then all input is sent to 'TaskServlet' servlet. Servlet takes the 'currentTask' object from the session and calls 'update()' method to save the changes to database.

17) Diagram: In/OutControl



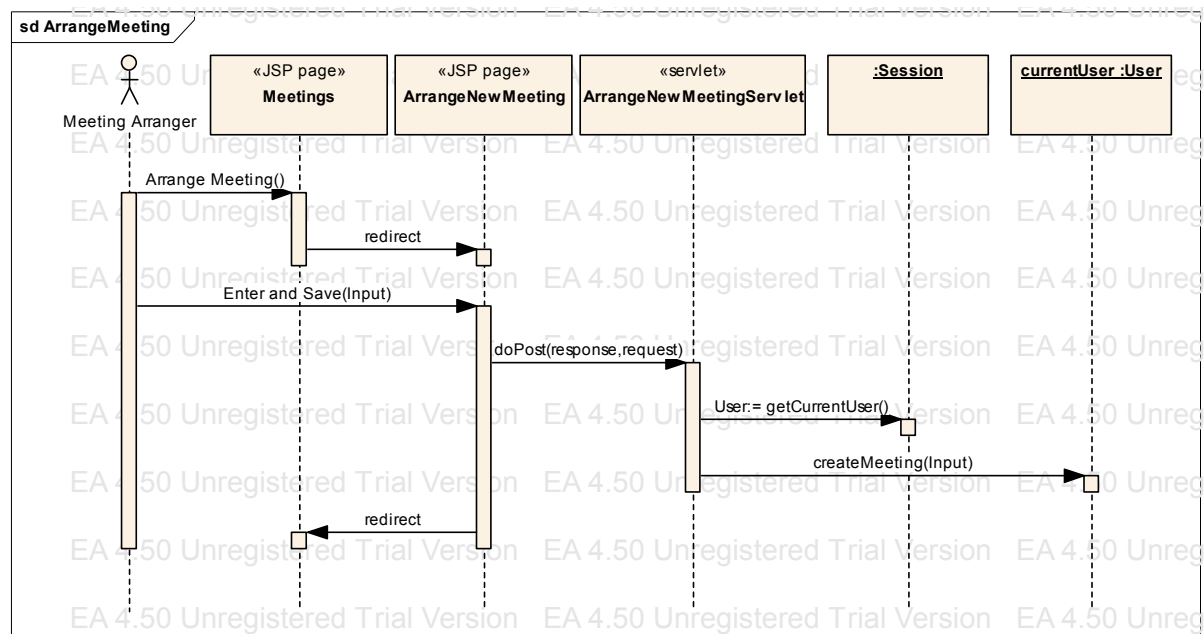
When working on the tasks user must open 'IN' option so the working time is calculated for the user. User can open IN from 'Task.jsp' which sends the input 'OpenInServlet' servlet. It takes the 'currentTask' object from the session and calls startWorkedIn() method to save the start time to database. Then after working he presses the 'OUT' button from 'Task.jsp'. Then page sends tyhe input to 'OutServlet' servlet. Servlet takes the currentTask object and calls finishWorkedIn() method to save the information to database.

18) Diagram: ReviewTask



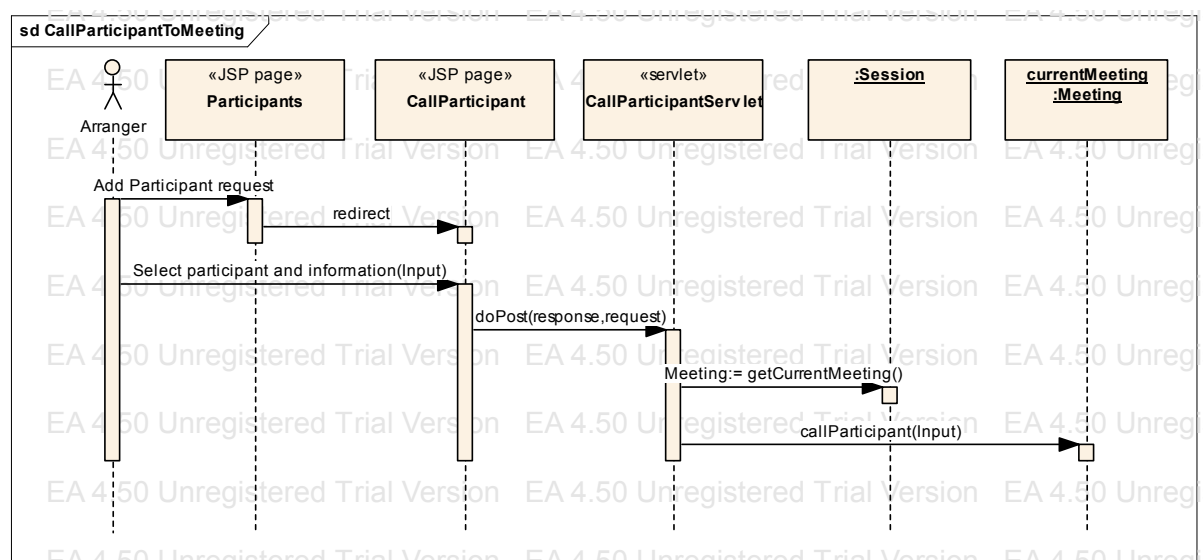
This diagram shows the process of reviewing the works of users on tasks and rejecting or accepting them. Page calls the servlet's doPost method after reviewer selects the task to review. Servlet calls the session's 'retrieveTask()' method which returns a 'Task' object to take the related Task from database. Then it takes the currentUser object from the session and calls its 'approveTask()' method to save the decision of the reviewer into the database.

19) Diagram: ArrangeMeeting



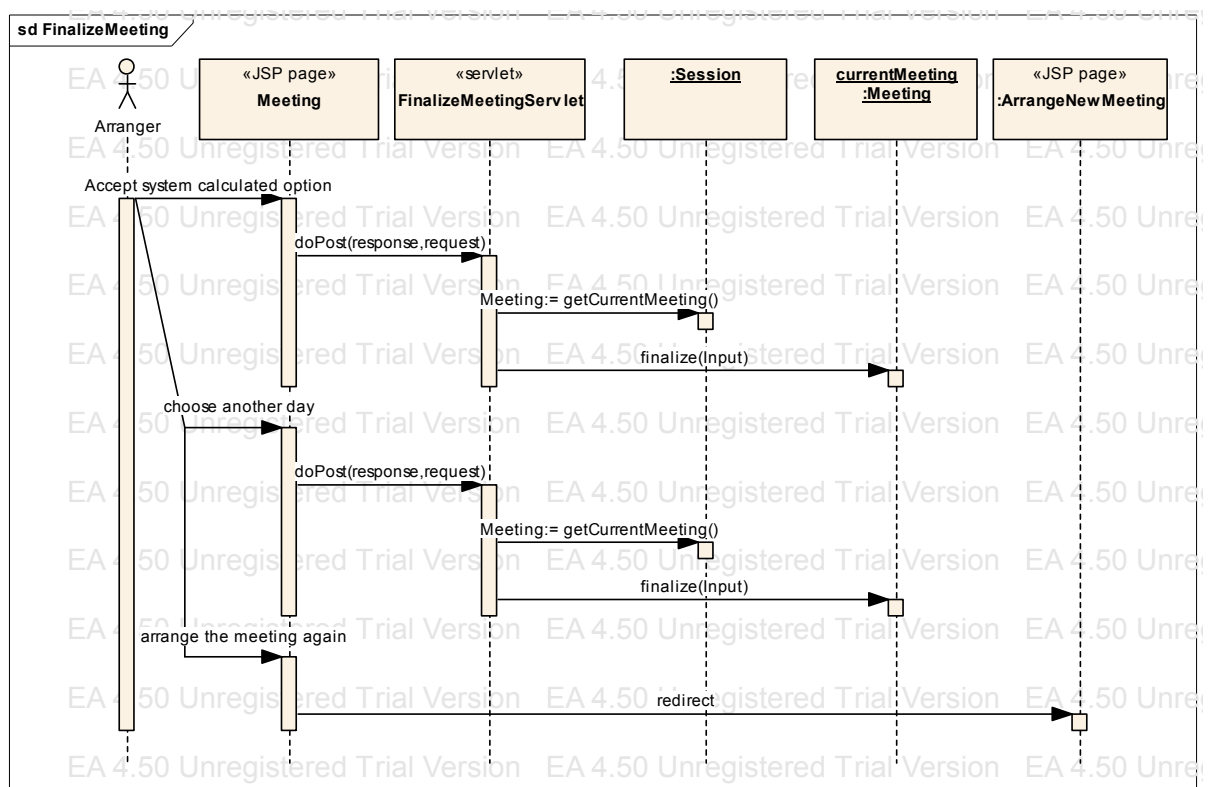
User can see his meetings and their information on 'Meetings.jsp'. When user requests to arrange a new meeting page is redirected to 'ArrangeNewMeeting.jsp' page. User specifies the information for the meeting and save them. Then input is sent to 'ArrangeNewMeetingServlet' servlet which will get the 'currentUser' object from the session. Then servlet calls the 'createMeeting()' method of the 'currentUser' object to create the meeting in database.

20) Diagram: CallParticipantToMeeting



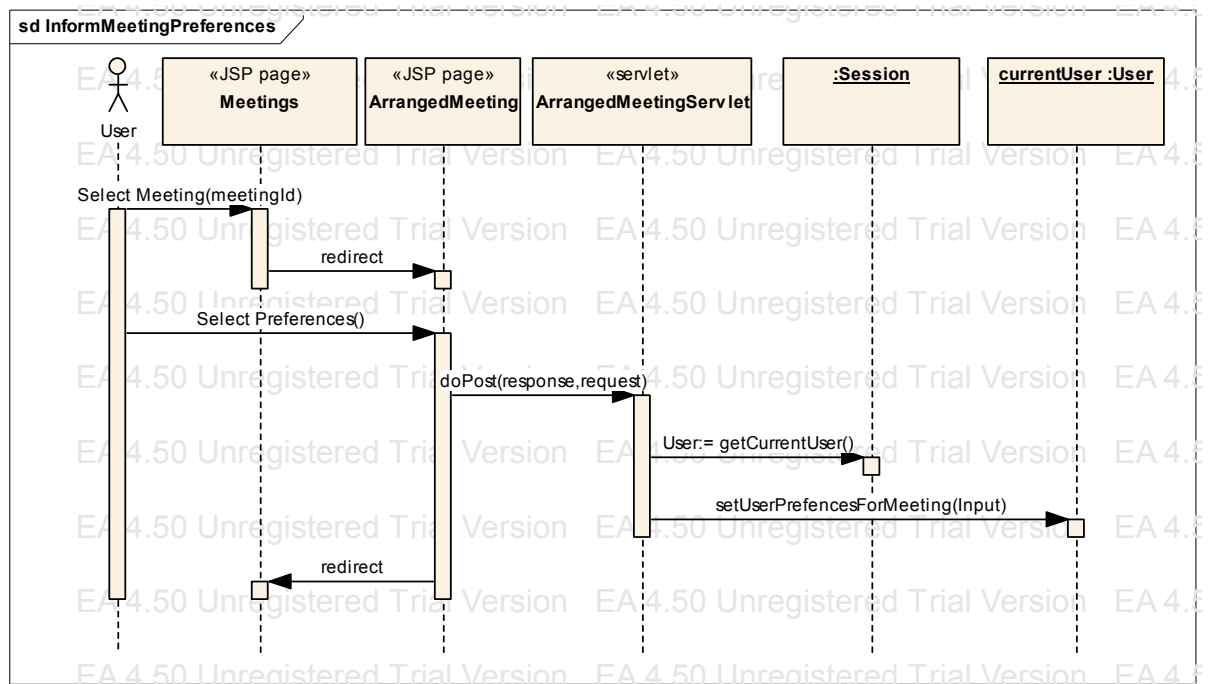
User can go to 'Participants.jsp' page from 'Meeting.jsp' page. This page shows the participants of the meeting. When user requests to call a new participant, user is redirected to 'CallParticipant.jsp' page. User selects the user and other information for the participant and saves the information. Then page sends the input to the 'CallParticipantServlet' servlet by the doPost method. Servlet gets the 'currentMeeting' object from the session and calls callParticipant() method of this object which will save the information to database.

21) Diagram: FinalizeMeeting



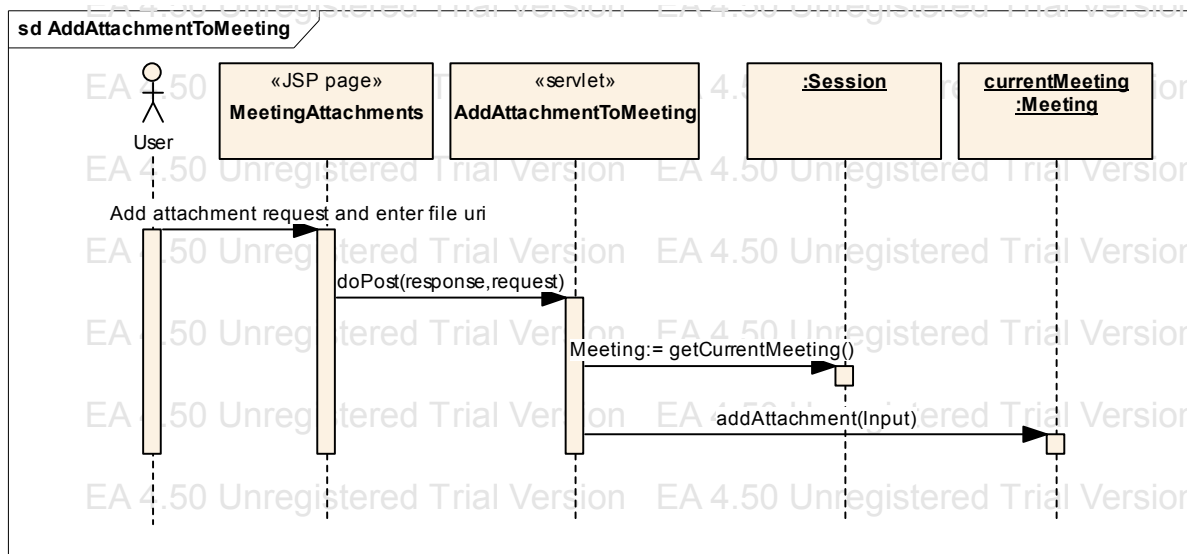
Meeting arranger can view the participant's selections and system selection for meeting date. Then he can accept this date or arrange new day for meeting. Servlet takes the 'currentMeeting' object and calls its finalize() method.

22) Diagram: InformMeetingPreferences



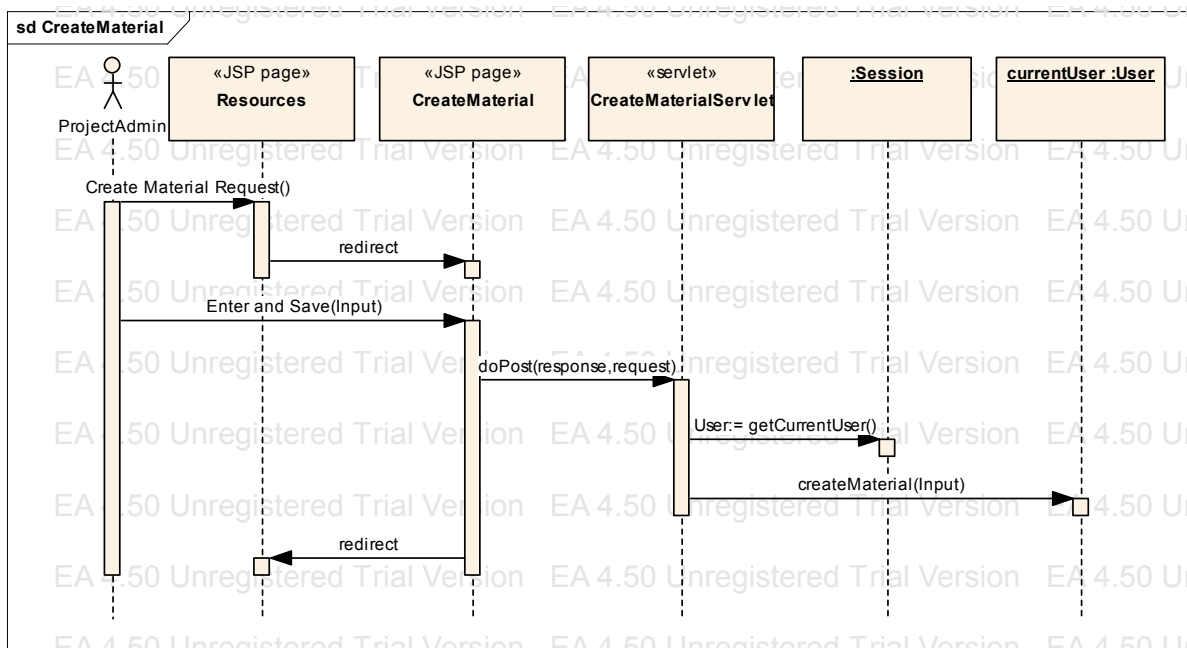
A user can go to the 'ArrangedMeeting.jsp' page from 'Meetings.jsp' if he has necessary right and he is called to meeting as participant. Then user selects his date preferences and save this information. Then page send the input to the 'ArrangedMeetingServlet' by doPost method. Servlet takes 'currentUser' object from the session and calls the 'setUserPrefencesForMeeting()' method to save the information to database.

23) Diagram: AddAttachmentToMeeting



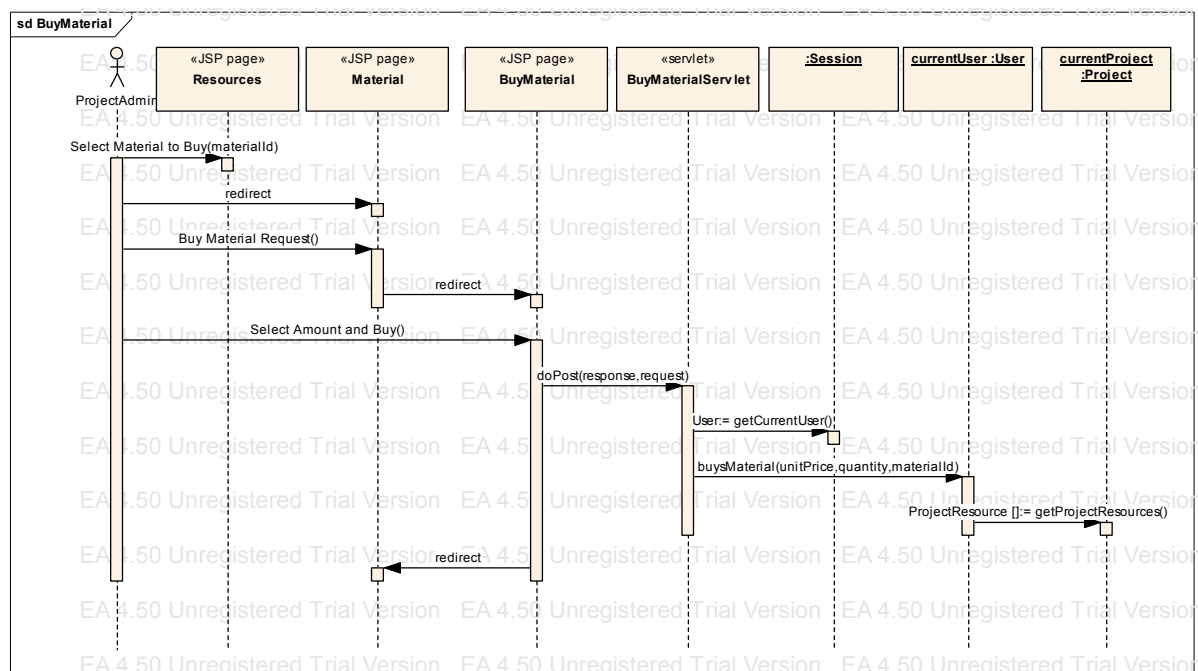
User can view meeting attachments from 'MeetingAttachments.jsp' page. When user requests to add new attachment to meeting he gives the file uri as input. Then input is send to 'AddAttachmentToMeetingServlet' servlet that will get 'currentMeeting' object from session and calls 'addAttachment()' method. This method save the filename to database and copy file to system's file directory.

24) Diagram: CreateMaterial



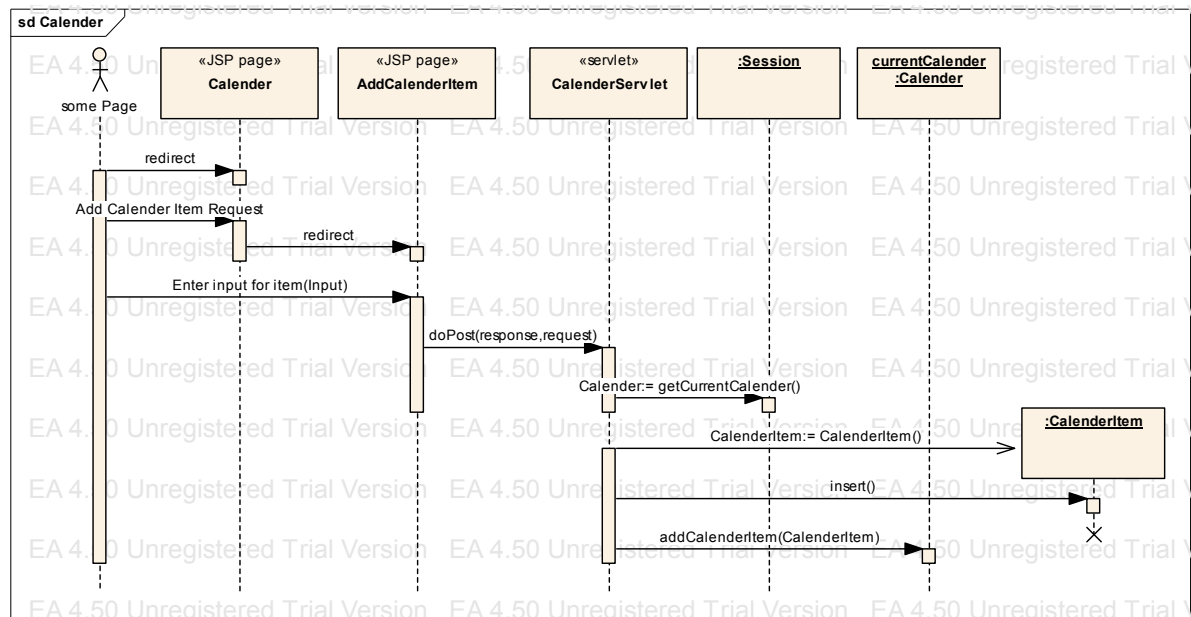
This diagram shows the sequence of the processes for defining a new material for company. From the Resources page user can request to define new material which will redirect him to 'CreateMaterial.jsp'. In this page user specifies the information about the material and submits to create the material which calls the servlet's doPost method. Servlet gets the 'currentUser' object from the session and calls the 'createMaterial()' method of this object. This method creates a 'Material' object and calls its 'insert()' method to create the material in database.

25) Diagram: BuyMaterial



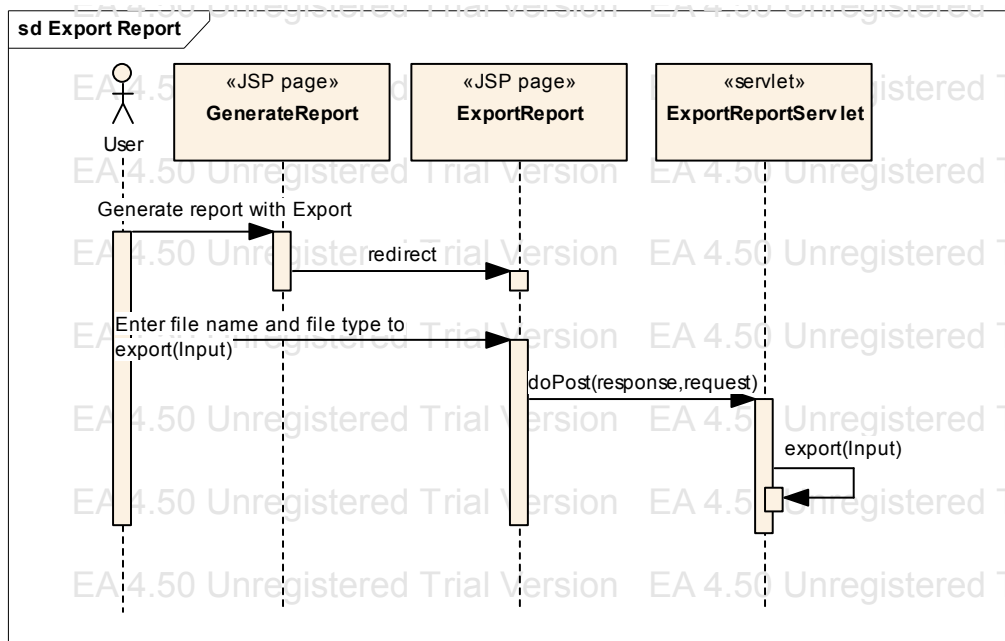
This diagram shows the process of purchasing some quantity of specified material to project. In the Resources page user can see the project resources and defined materials. If he select a material page redirects to 'Material.jsp' which shows the materials properties. Then user request to purchase some quantity of this material type which redirects to page 'BuyMaterial.jsp'. After specifying amount of material that is bought page calls the 'BuyMaterialServlet' servlets doPost method. Servlet gets the 'currentUser' object from the session and calls the 'buysMaterial()' method of this object. This method get the 'ProjectResources' of the 'currentProject' object and call the 'setQuantity()' to set the new quantity to database and to object.

26) Diagram: Calender



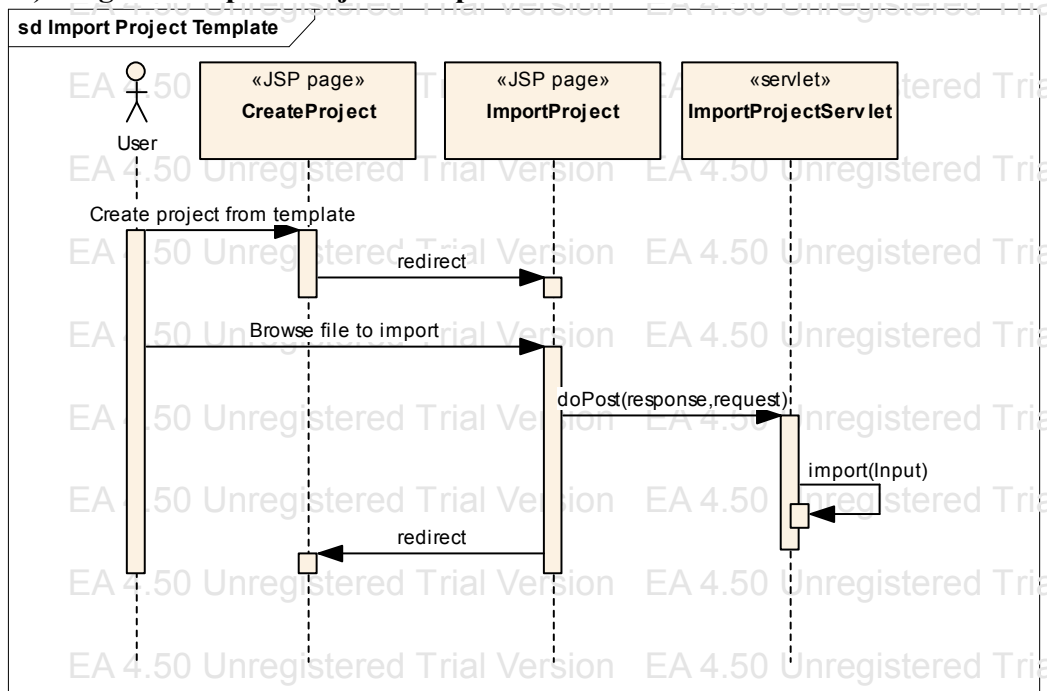
User can view the calendar and his items on the 'Calender.jsp' page. When user requests to add a calendar item to calendar page is redirected to 'AddCalenderItem.jsp' page. User enters the information and input is send to the servlet. Servlet gets the 'currentCalender' object from session and create a CalendarItem object from the input. Then this object is added to 'currentCalender' object by 'addCalendarItem()' method.

27) Diagram: Export Report

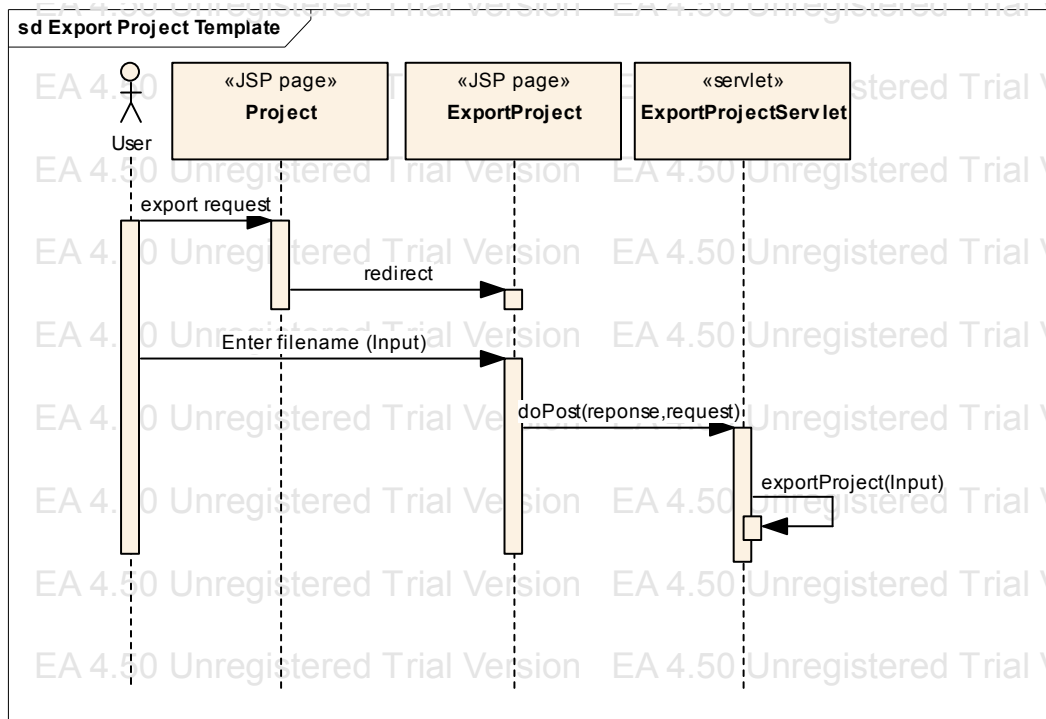


When user is in 'GenerateReport.jsp' he can select generating report with export option. Then ExportReport.jsp page is redirected and after user gives the file uri and export type the input is given to 'ExportReportServlet' servlet. This servlet generates the report in specified format and redirect back.

28) Diagram: Import Project Template

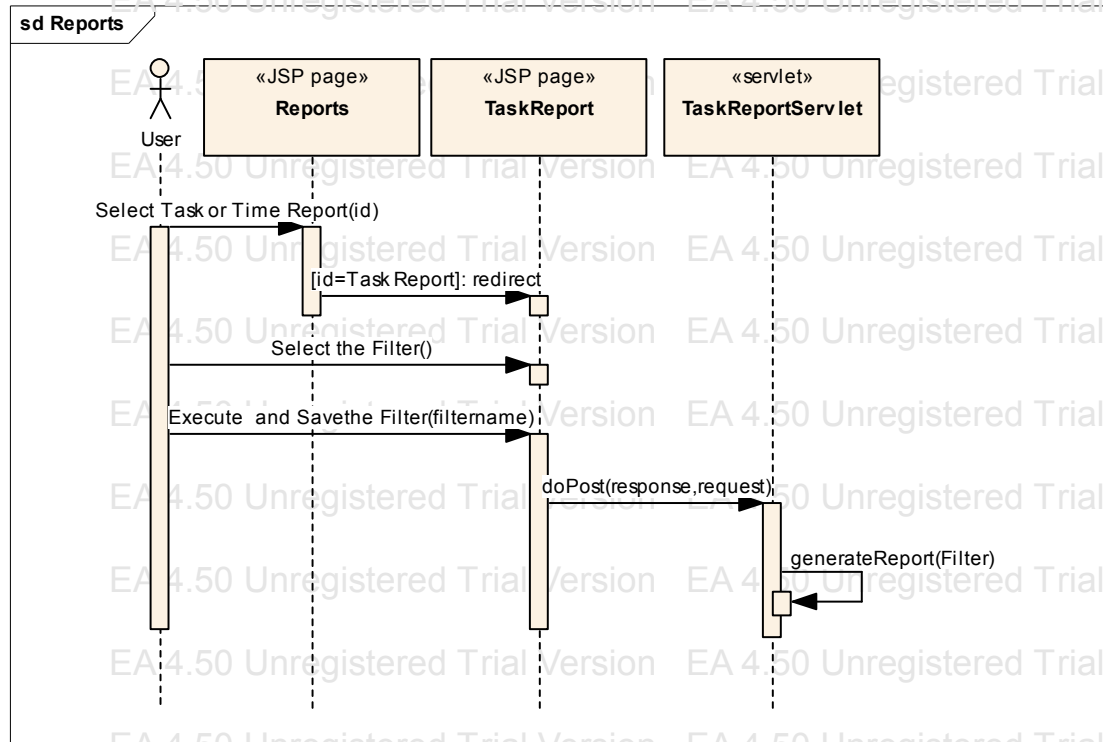


29) Diagram: Export Project Template



While creating project user can select creating project from a template. Then 'ImportProject.jsp' is redirected and after specifying the filename to import all information is passed to 'ImportProjectServlet' servlet. This servlet takes the file and generate necessary things in database. Sequence is same for export project.

30)

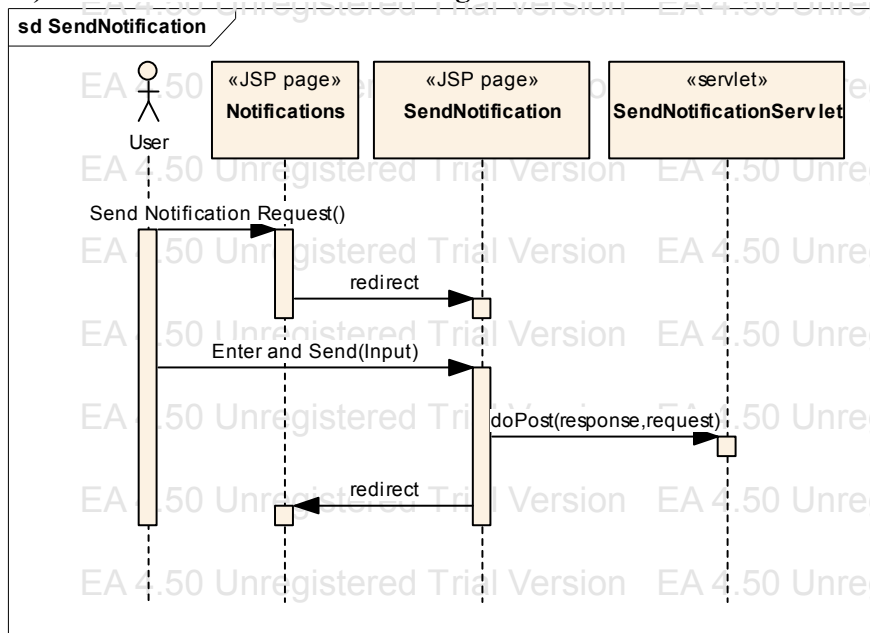
Diagram:**Reports**

This diagram only shows the process of creating a report from selected filter. User selects the filter and submits his request to generate report. Servlet's doPost method is called and this method generates the report from the information in the database. If the user also wants to save this filter by giving a name to it, servlet creates a 'Filter' object and calls its 'insert()' method to create the Filter in the database.

31)

Diagram:

SendNotification



This diagram shows the process of sending a notification to another user. Servlet gets the 'currentUser' object from session and calls the 'sendNotification()' method for creating the notification in the database. This method create the notification and calls its 'insert()' method.

3.4 ACTIVITY VIEW

The activity diagram of our system is presented on a A3 page layout in the Appendix since it does not fit on a A4 paper.

The explanation of the activity diagram is below:

The activities are started by displaying the login screen.

- The user will fill in with his/her id and password, after the user enters his/her id and password these information will be compared with the one that will be retrieved from the database. If the password turns out to be valid then the main screen will be displayed, if the password entered by the user turns out to be invalid then we return to the initial screen.

First of all, some information isn't shown in this activity diagram for the sake of simplicity. The omission is; after main screen is displayed, the user can select display main screen link or logout link any time s/he wants.

In the main screen the user has the following options:

- If the user selects 'Send Notification' link then the notification form will be displayed. And the user will fill in. Then the user will click send button and the notification will be sent to the specified users. (As I mentioned in the beginning, in any of the stages, the user can select the 'Main Screen' link so that the notification process will be canceled and system will return to the Main screen. Also the user can select 'Logout' in any stage so that the system will terminate the session. These possible activities won't be specified in any other option.)
- If the user selects 'Create new filter' link then the new filter creation form will be displayed. And the user will fill in. Then the user will click save button and the filter will be stored in database.
- If the user selects 'Edit Preferences' link then the 'Preferences Screen' will be displayed. And the user will edit his/her preferences. Then the user will click save button and the preferences will be stored in database.
- If the user selects 'Generate Statistics Link' link, then s/he will select the filters to be applied. After that the statistics will be generated and displayed depending on the filters selected by the user.
- If the user selects 'Help' link then the 'Help Screen' will be displayed. And the user will enter the topic that s/he wants to get information about. Then the system will display the information about the topic if there is any record about that topic in the database.
- If the user selects 'Forum' link then the 'Forum Screen' will be displayed. After that user can either read a message or write a new message. If s/he wants

to read a message, s/he will simply select the thread and the message will be displayed. If the user wants to write a new message, s/he will select the thread under which s/he wants to write new message and then will write the body of the message and click the send button after that the message will be stored in database.

- If the user selects 'Reports' link then the 'Reports Screen' will be displayed. After that user has two other options :
 - If the user selects 'Import Report' link, the report will be fetched from user's computer and will be viewed.
 - If the user selects 'Generate Report' link, s/he will specify the type of the report and the filters to be applied and after that system will generate the report based on this selections and display it. In this stage user can select to export the report into his/her computer or go back to the main reports screen.
- If the user selects 'Administration' link then the access rights of the user will be checked. If the user doesn't have the necessary rights, s/he won't be able to do any administration operation and 'Main Screen' will be displayed. Else if the user has admin rights 'Administration Screen' will be displayed. And the user can either select to create a new user account to the system or create a new company account. In either case, admin will enter the necessary information, then select the save button and the records will be saved in database.
- If the user selects 'Arrange Meeting' link then the access rights of the user will be checked. If the user doesn't have the necessary rights, s/he won't be able to arrange any meeting and 'Main Screen' will be displayed. Else if the user has enough rights user will specify the potential dates and the attendants of the meeting. After the potential attendants of the meeting stated their choices, user will fix the details of the meeting depending on these choices. And then user will select the save button and the records will be saved in database.
- If the user selects 'Projects' link then the 'Projects Main Screen' will be displayed. In this stage, user has following options:
 - User can view the details of a project by selecting 'View Project' link. After this selection the system will display the project details and now user has another two options:
 - If the user selects 'Export Project' link, the project will be saved in a file into the user's computer.
 - If the user selects 'Task creation' link, then the access rights of the user will be checked. If the user doesn't have the enough rights, s/he won't be able to create any task and 'Projects Main Screen' will be displayed. Else if the user has enough rights, s/he will enter the necessary information to create a new task (task name, assigned users, etc) and hit the save button. After that new task will be saved in the database.
 - User can select 'Create New Project' link. Of course, first of all the access rights of the user are fetched from the database to see whether

s/he has the necessary access rights to create a project and if not user won't be able to create the project and 'Projects Screen' will be displayed. Else if the user has enough access rights, s/he will specify the creation type (from template or from scratch). To create project from template, the project file is fetched from user's computer and for the other case a blank project is created and the user enters the necessary information about the new project (name, tasks, assigned users, etc). After that system saves the project in database.

- If the user selects 'Tasks' link then the 'Tasks Main Screen' will be displayed. In this stage, user has following options:
 - User can select 'View Tasks' link. In this case, the access rights of the user will be checked. If the user doesn't have the enough rights, s/he won't be able to view any task and 'Tasks Main Screen' will be displayed. Else if the user has enough rights, task will be displayed. Now, user can either choose to send the finished tasks to the reviewer or work on a task. If the user selects to work on a task, it will be checked that whether the task is assigned to the user. If it's not, the user won't be able to open in/out and 'Main Tasks Screen' will be displayed. Else in/out will be opened, the user will work on the task and in/out will be closed.
 - User can select 'Review Tasks' link. In this case, the user will select from the finished tasks which are sent for reviewing and if the reviewer of the task is assigned to be the user, s/he will be able to review the task and either accept or reject the work done. In either case, a notification is sent to the user who did the task and if reviewer rejected the work done, task will be marked as undone and the assigned user will have to do it again.
- If the user selects 'Logout' in any stage then the system terminates the session.

3.5 STATECHART VIEW

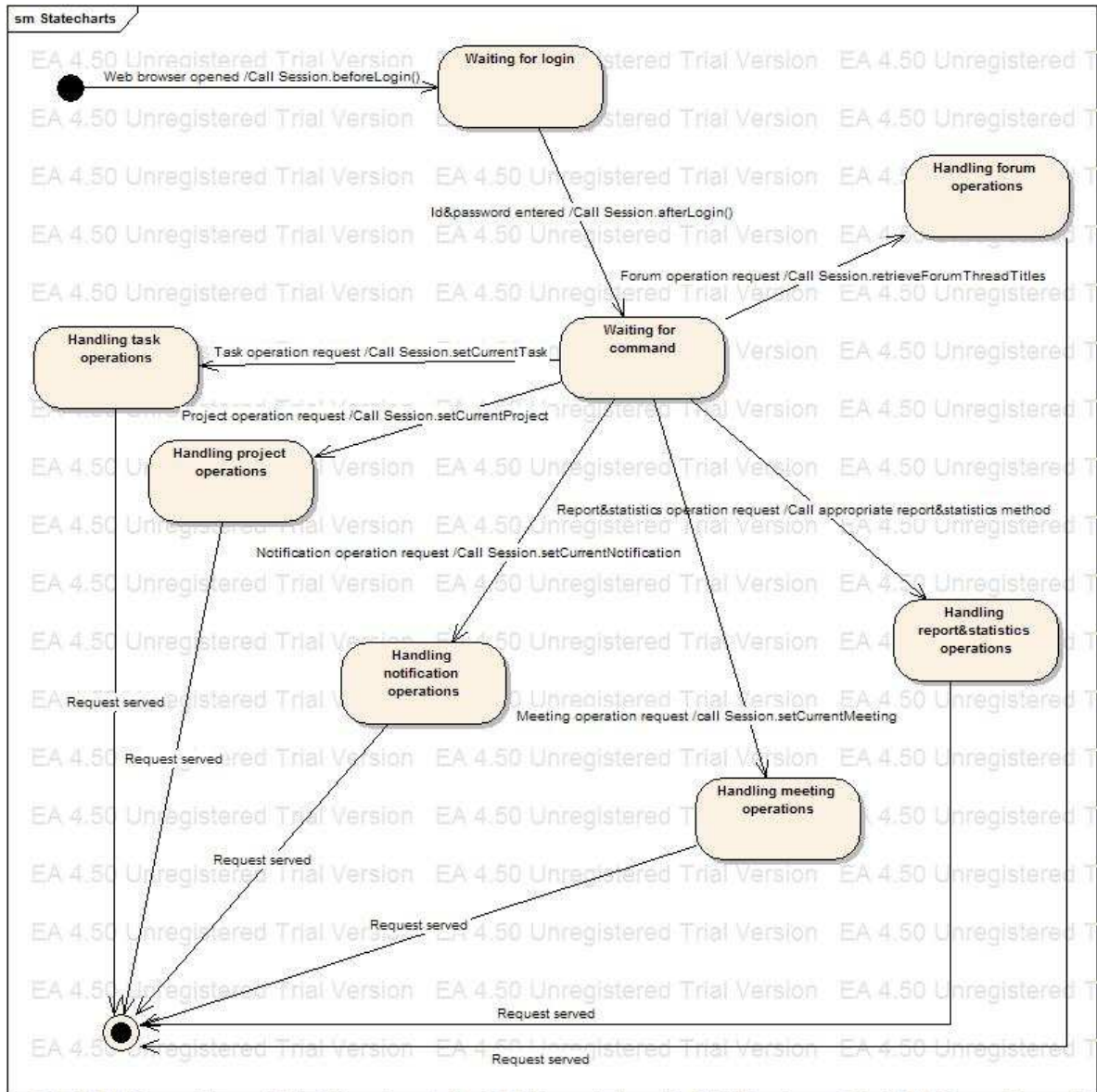


Figure 1: Session class state diagram

When the user opens the address of the project management tool in his browser, a session is created and its beginLogin method is called, triggering a translation from the initial state to the Waiting for login state. When the user enters his id and password correctly, the new state is Waiting for command state, in which requests of the user are being waited to be handled.

There are a number of possible translations from the Waiting for command state. The state changes to:

- Handling task operations state, if the user makes a task operation request
- Handling project operations state, if the user makes a project operation request
- Handling notification operations state, if the user makes a notification operation request
- Handling meeting operations state, if the user makes a meeting operation request
- Handling report&statistics operations state, if the user makes a report&statistics operation request
- Handling forum operations state, if the user makes a forum operation request

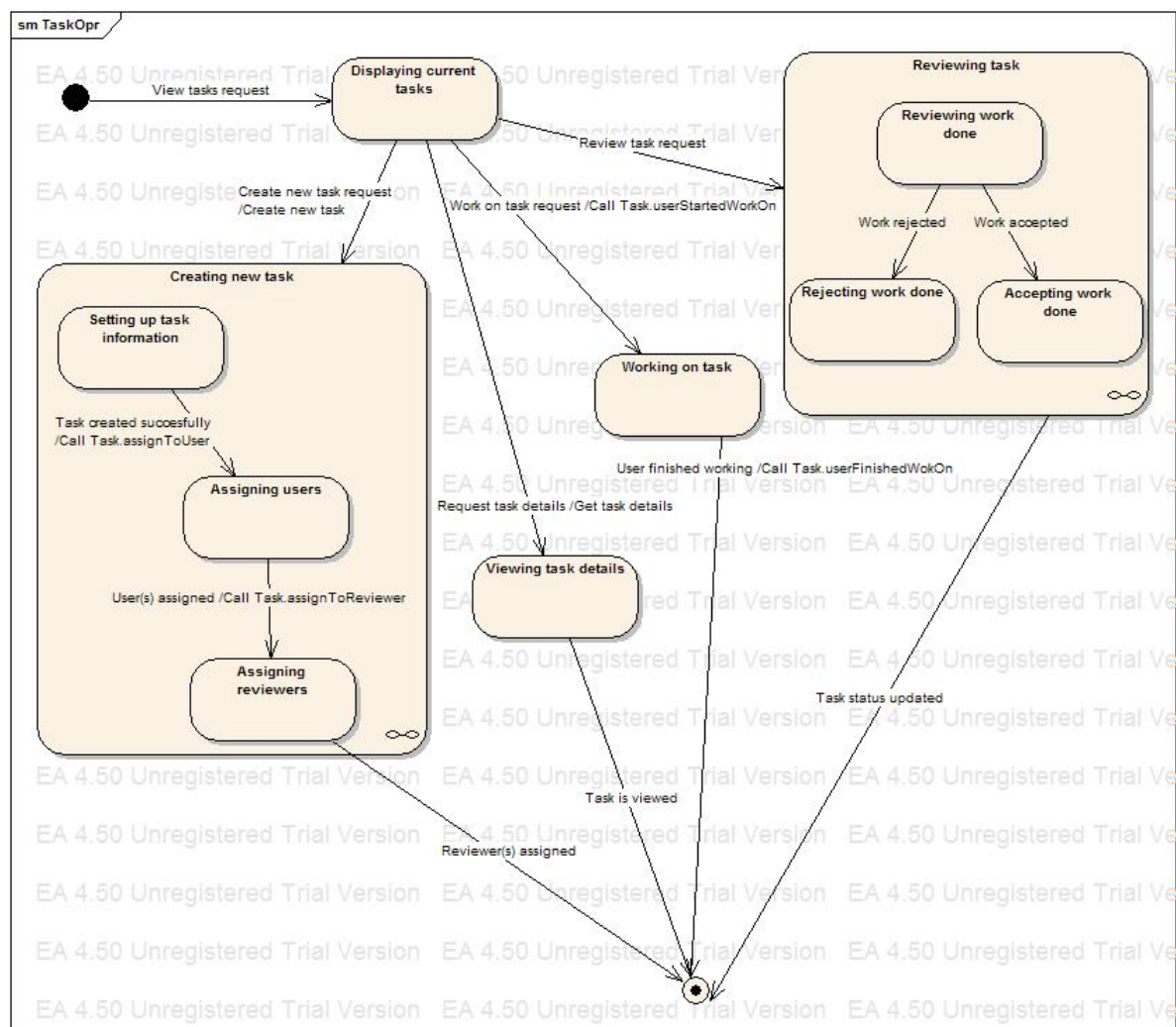


Figure 2: Task class state diagram

When the user makes a view tasks request, a transition occurs from the initial state to the Displaying current tasks state.

When the user wants to create a new task, a transition occurs from the Displaying current tasks state to the Creating new task state. This state has three sub-states, namely the Setting up task information state, Assigning users state, and the Assigning reviewers state. In the Setting up task information sub-state, the necessary information for the creation of a task is entered. When this necessary information is provided, a transition occurs to the Assigning users state, in which the task is assigned to users. When the assignment is done properly, a transition occurs to the Assigning reviewers state, in which reviewers are assigned to the task.

When the user wants to review a task, a transition occurs from the Displaying current tasks state to the Reviewing task state. This state has three sub-states, namely the Reviewing work done state, Rejecting work done state, and the Accepting work done state. The user reviews the task in the Reviewing work done state. Depending on the decision of the reviewer, a transition occurs either to the Rejecting work done state (the work done is rejected), or to the Accepting work done state (the work done is accepted).

When the user wants to work on a task, a work on task request causes a transition from the Displaying current tasks state to the Working on task state.

When the user wants to view the details of a task, requesting the details of the task causes a transition from the Displaying current tasks state to the Viewing task details state.

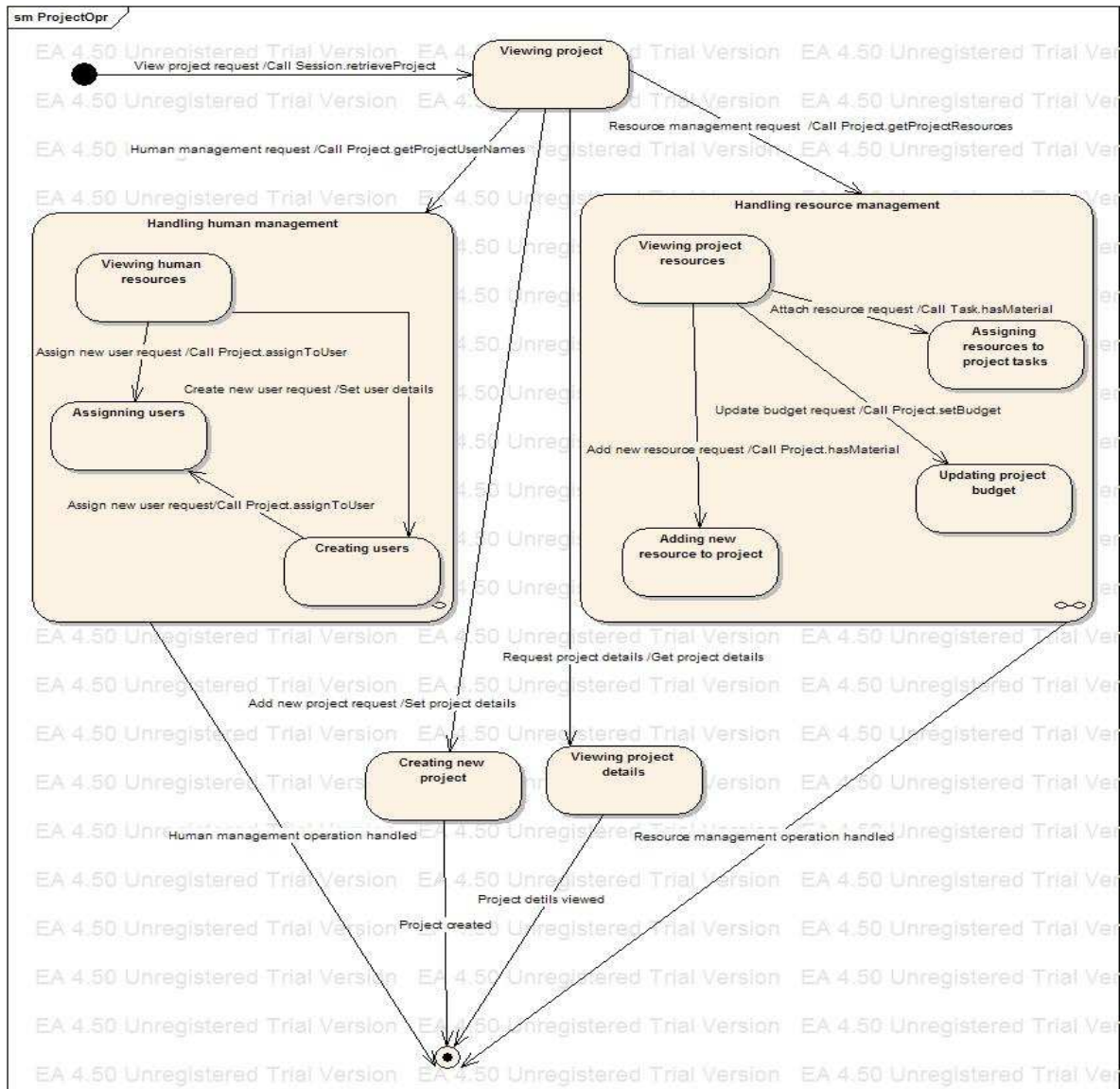


Figure 3: Project class state diagram

When the user makes a view project request, a transition occurs from the initial state to the Viewing project state.

When the user wants to perform human management operations, a transition occurs from the Viewing project state to the Handling human management state. This state has three sub-states, namely the Viewing human resources state, Assigning users state, and the Creating users state. The user views that human resources of the project in the Viewing human resources state. If the user wants to assign a user to the project, a transition occurs to the Assigning user state. If the user wants to create a new user, a transition occurs to the Creating users state. If the user wants to assign the newly created users to the project, a transition occurs to the Assigning users state.

When the user wants to perform Resource management operations, a transition occurs from the Viewing project state to the Handling resource management state. This state has four sub-states, namely the Viewing project resources state, Assigning resources to project tasks state, Updating project budget state, and the Adding new resource to project state. The user can view the project resources in the Viewing project resources state. If the user wants to add a new resource to the project, a transition occurs to the Adding new resource to project state. If the user wants to update the budget of the project, a transition occurs to the Updating project budget state. If the user wants to assign resources to any of the project tasks, a transition occurs to the Assigning resources to project state.

When the user wants to create a new project, a transition occurs from the Viewing project state to the Creating new project state.

When the user wants to view the details of a project, a transition occurs from the Viewing project state to the Viewing project details state.

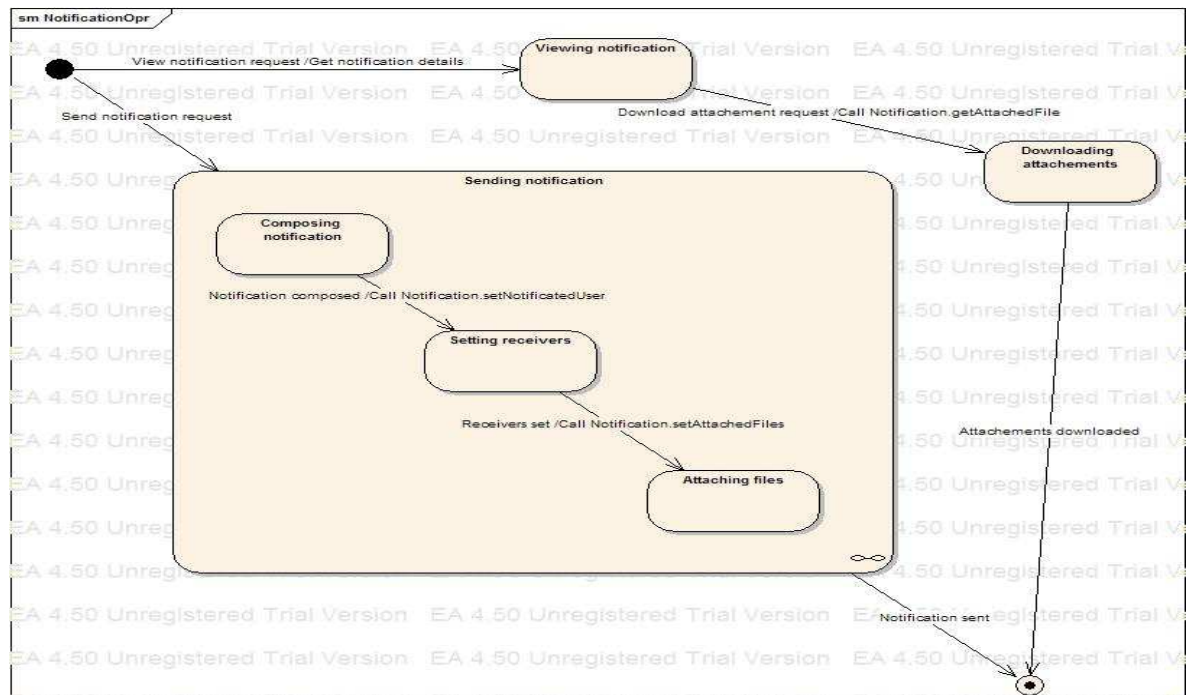


Figure 4: Notification class state diagram

When the user makes a send notification request, the transition occurs from the initial state to the Sending notification state. This state has three sub-states, namely the Composing notification state, Setting receivers state, and the Attaching files state. The user composes the notification in the Composing notification state. When the notification is composed, a transition occurs to the Setting receivers state, in which the receivers of the notification are set. When the receivers are set successfully, a transition occurs to the Attaching files state, in which the files (if exists) of the notification are attached to it. After this state, the notification is ready to be sent.

When the user wants to view his notifications, a transition occurs from the initial state to the viewing notification state. If the user makes a download attachments request at this state, a transition occurs to the Downloading attachments state.

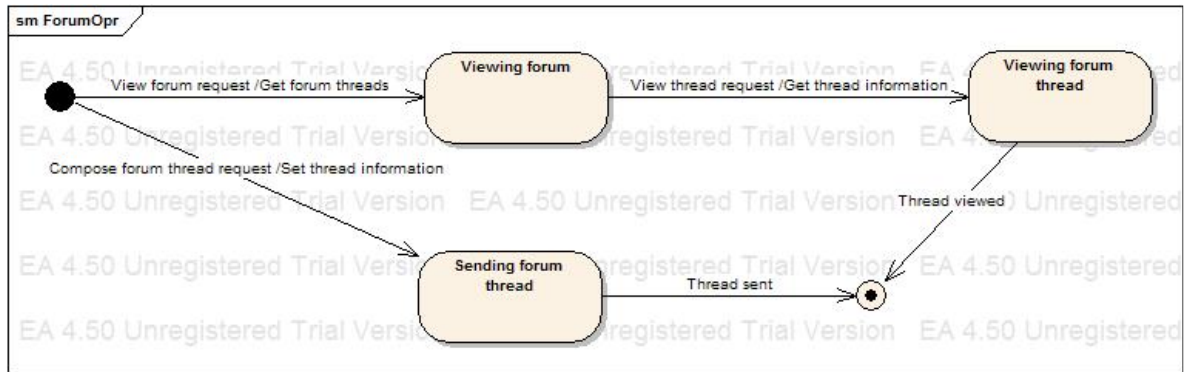


Figure 5: Forum class state diagram

When the user wants to view the forum a view forum request causes a transition from the initial state to the Viewing forum state. If the user wants to view a thread in the forum, a view thread request causes a transition to the Viewing forum thread state.

When the user wants to compose a forum thread, a compose forum thread request causes a transition from the initial state to the Sending forum thread state.

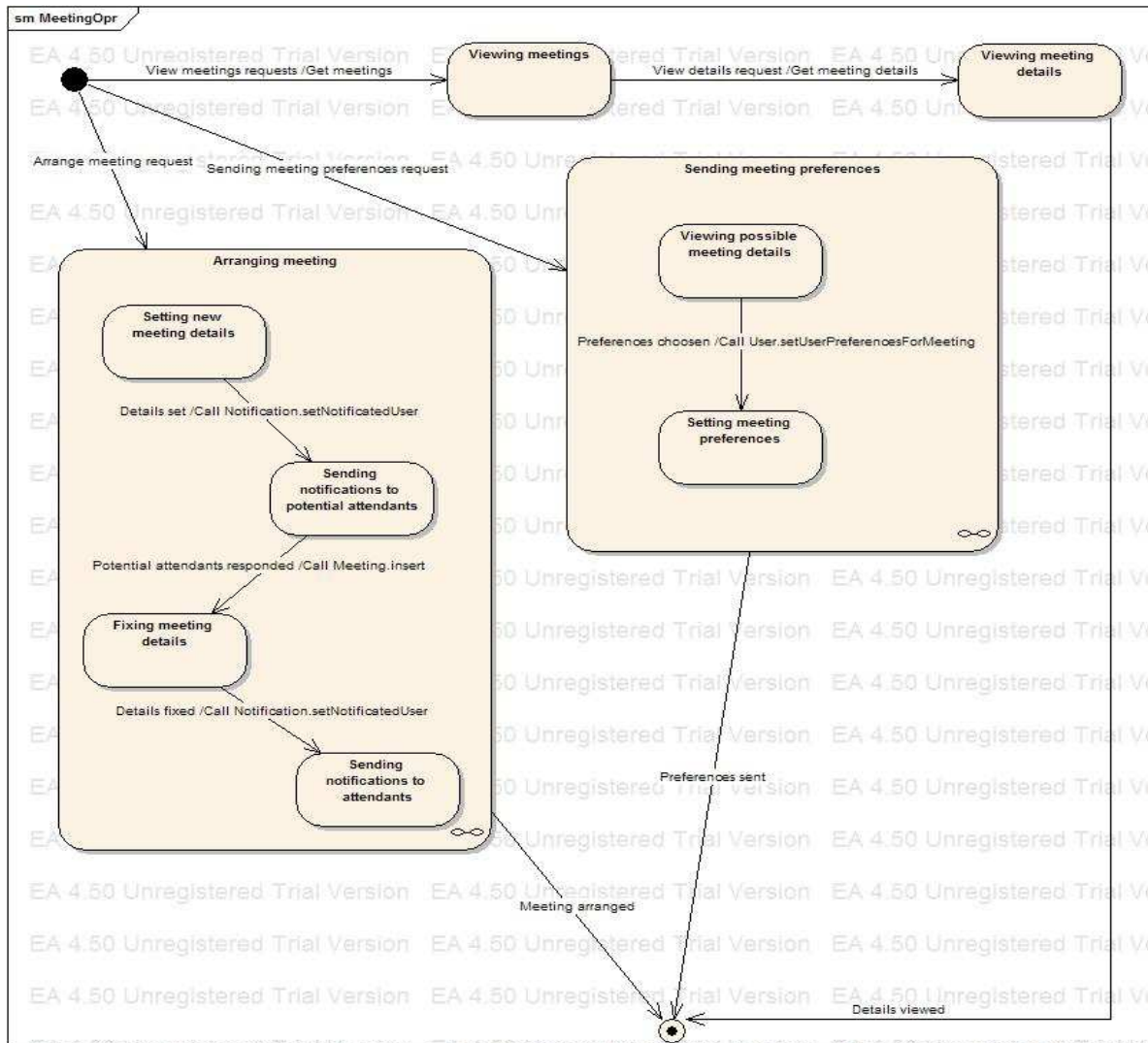


Figure 6: Meeting class state diagram

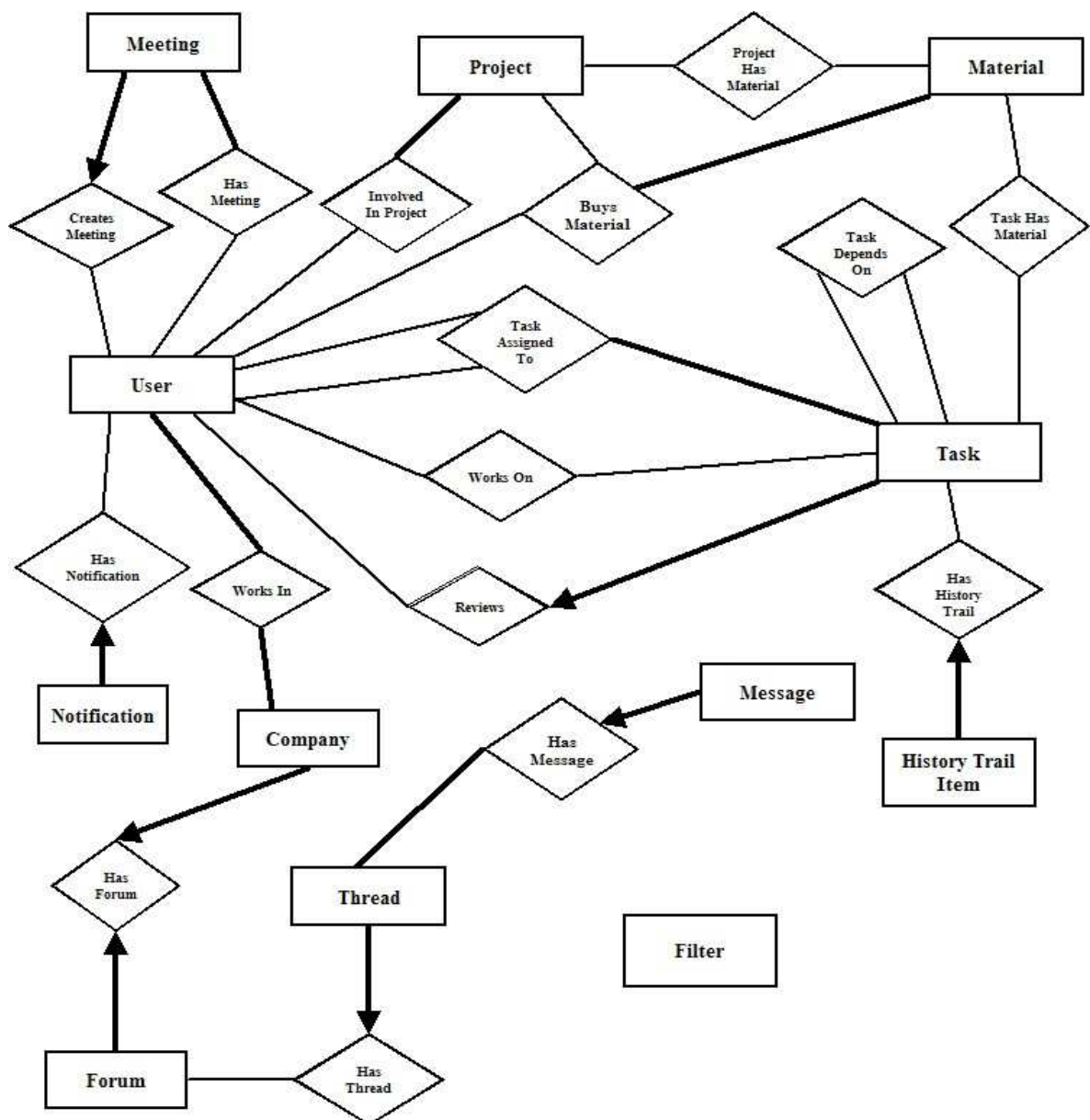
When the user wants to arrange a meeting, a transition occurs from the initial state to the Arranging meeting state. This state has four sub-states, namely the Setting new meeting details state, Sending notifications to potential attendants state, Fixing meeting details state, and Sending notifications to attendants state. In the Setting new meeting details state, the user specifies options for meeting details (e.g. meeting place, date). When these are set, a transition occurs to the Sending notifications to potential attendants state, in which the potential attendants are notified of the meeting options. When the potential attendants notify the arranger of their preferences, a transition occurs to the Fixing meeting details state, in which all the details of the meeting are fixed. Then the exact details of the meeting are sent to the attendants in the Sending notifications to attendants state.

When the user is a potential attendant of a meeting and wants notify the arranger of his choices, a transition occurs to the Setting meeting preferences state. This state has

two sub-states, namely the Viewing possible meeting details state and the Setting meeting preferences state. The user views the possible meeting options in the Viewing possible meeting details state. When he wants to set his preferences for that meeting, a transition occurs to the Setting meeting preferences state.

3.6 DATABASE DESIGN

3.6.1 ER DIAGRAM



3.6.2 DATABASE TABLES

USER

user_id
password
date_created
last_visit_time
first_name
last_name
phone
email
birth_date
gender
speciality
photo
address
global_access_right
can_add_project
user_directory
num_of_tasks_per_page
num_of_months_per_page
num_of_weeks_per_page
payment_policy
payment_amount
user_group_id
PRIMARY KEY(user_id)

COMPANY

company_id
company_name
company_address
contact_info
logo
email_server
webpage
timezone_format
PRIMARY KEY(company_id)

TASK

task_id
task_name
task_description
start_date
due_date

finish_date
 priority_id FOREIGN KEY(priority_table:priority_id)
 type_id FOREIGN KEY(task_type_table:task_type_id)
 project_id FOREIGN KEY(project:project_id)
 status_id FOREIGN KEY(status:status_id)
 percent_done
 reviewer_id FOREIGN KEY(user:user_id)
 group_id FOREIGN KEY(group_table:group_id)
 attached_file1
 attached_file2
 attached_file3
 attached_file4
 actual_hours
 last_update
 date_created
 last_reviewed_percent_done
 PRIMARY KEY(task_id)

CALENDAR_ITEMS

calendar_item_id
 user_id FOREIGN KEY(user:user_id)
 date
 text
 is_to_be_reminded
 PRIMARY KEY(calendar_item_id)

HISTORY_TRAIL_ITEM

history_trail_id
 task_id FOREIGN KEY(task:task_id)
 user_id FOREIGN KEY(user:user_id)
 modification_type
 old_value
 new_value
 PRIMARY KEY(history_trail_id)

PROJECT

project_id
 project_name
 project_description
 project_creator FOREIGN KEY(user:user_id)
 create_date
 start_date
 finish_date
 due_date

budget
contact_name
contact_phone
contact_email
project_type_id FOREIGN KEY(project_type_table:project_type_id)
PRIMARY KEY(project_id)

FILTER
filter_id
selected_user_id
selected_project_id
selected_priority_id
selected_type_id
selected_status_id
selected_group_id
selected_startdate
selected_finishdate
PRIMARY KEY(filter_id)

MEETING
meeting_id
final_meeting_date
date_option1
date_option2
date_option3
date_option4
date_option5
creator_userid FOREIGN KEY(user:user_id)
create_date
attachment1
attachment2
attachment3
last_reply_date
PRIMARY KEY(meeting_id)

NOTIFICATION
notification_id
notificated_user FOREIGN KEY(user:user_id)
notification_type
notification_text
owner_of_action FOREIGN KEY(user:user_id)
date_of_action
attached_file1
attached_file2
attached_file3

PRIMARY KEY(notification_id)

Task_Depends_On

task_id1 FOREIGN KEY(task:task_id)

task_id2 FOREIGN KEY(task:task_id)

dependency_type

PRIMARY KEY(task_id1, task_id2)

User_Has_Meeting

user_id FOREIGN KEY(user:user_id)

meeting_id FOREIGN KEY(meeting:meeting_id)

condition

user_selected_option1

user_selected_option2

user_selected_option3

user_selected_option4

user_selected_option5

PRIMARY KEY(user_id, meeting_id)

MATERIAL

material_id

material_name

material_description

created_date

creator_user_id FOREIGN KEY(user:user_id)

PRIMARY KEY(material_id)

TASK_NEEDS_MATERIAL

task_id FOREIGN KEY(task:task_id)

material_id FOREIGN KEY(material:material_id)

quantity

date

PRIMARY KEY(task_id, material_id, date)

PROJECT_HAS_MATERIAL

project_id FOREIGN KEY(project:project_id)

material_id FOREIGN KEY(material:material_id)

quantity

PRIMARY KEY(project_id, material_id)

User_Buys_Material
 purchaser_id FOREIGN KEY(user:user_id)
 material_id FOREIGN KEY(material:material_id)
 project_id FOREIGN KEY(project:project_id)
 quantity
 unit_price
 supplier_id FOREIGN KEY(supplier:supplier_id)
 date
 PRIMARY KEY(purchaser_id, material_id, date)

SUPPLIER
 supplier_id
 supplier_name
 contact_info
 PRIMARY KEY(supplier_id)

TASK_ASSIGNED_TO
 user_id FOREIGN KEY(user:user_id)
 task_id FOREIGN KEY(task:task_id)
 date
 assigner_id FOREIGN KEY(user:user_id)
 PRIMARY KEY(user_id, task_id)

WORKS_ON
 user_id FOREIGN KEY(user:user_id)
 task_id FOREIGN KEY(task:task_id)
 start_date
 finish_date
 is_approved
 PRIMARY KEY(user_id, task_id, start_date)

HAS_ACCESS_RIGHT
 user_id FOREIGN KEY(user:user_id)
 project_id FOREIGN KEY(project:project_id)
 is_project_manager
 can_approve_time
 can_open_project
 can_arrange_meeting
 level
 PRIMARY KEY(user_id, project_id)

TASK_TYPE_TABLE

task_type_id

task_type

PRIMARY KEY(task_type_id)

PRIORITY_TABLE

priority_id

priority_name

PRIMARY KEY(priority_id)

STATUS_TABLE

status_id

status_name

PRIMARY KEY(status_id)

GROUP_TABLE

group_id

group_name

group_logo

PRIMARY KEY(group_id)

PROJECT_TYPE_TABLE

project_type_id

project_type

PRIMARY KEY(project_type_id)

FORUM_MESSAGES

message_id

thread_id

reply_to

message

subject

creator_name

creator_email

creation_date

PRIMARY KEY(message_id, thread_id)

4. INTERFACES

Here are some tentative visual interfaces from Dproject.

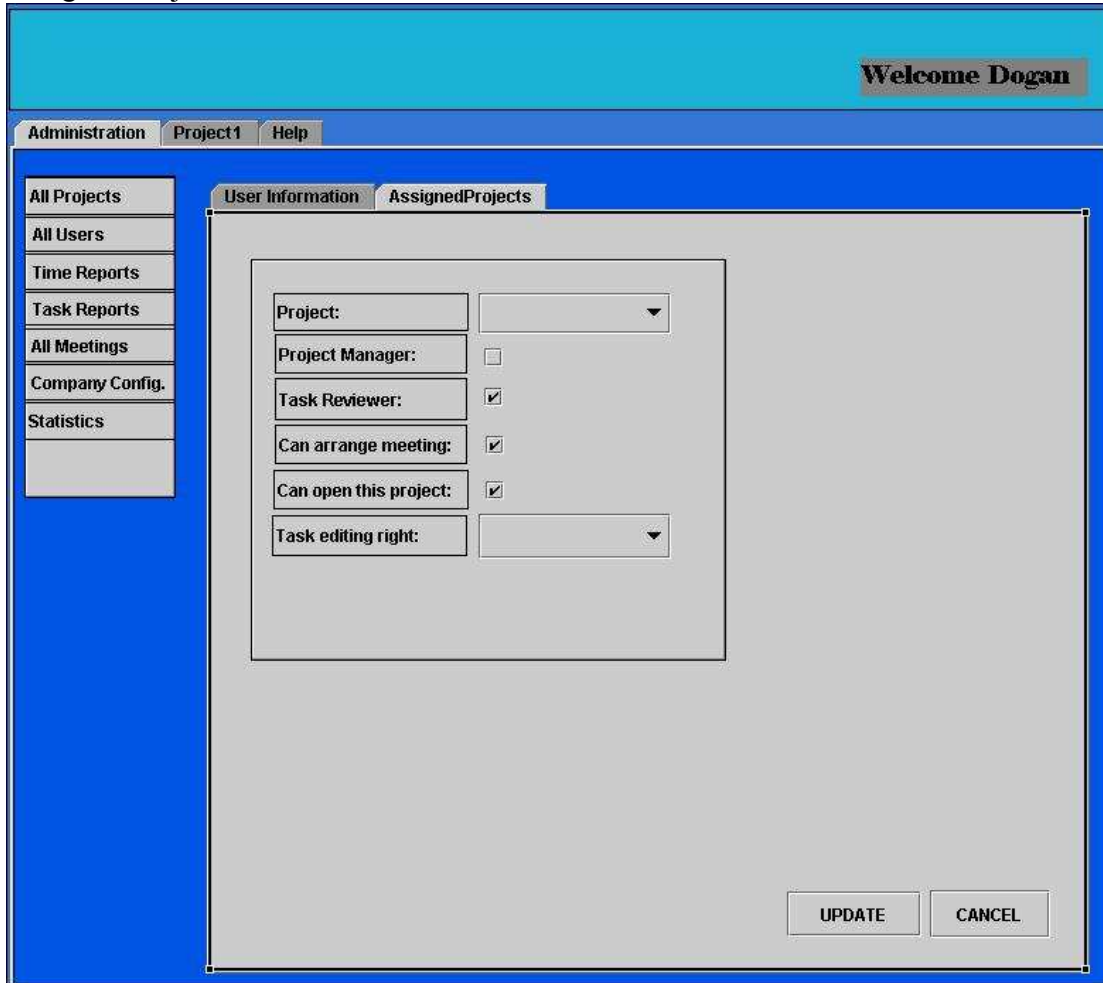
New User Interface

The screenshot displays the 'New User Interface' of the Dproject application. At the top, a blue header bar contains the text 'Welcome Dogan'. Below this, a navigation bar includes tabs for 'Administration', 'Project1', and 'Help'. The main content area is divided into a left sidebar and a central form. The sidebar lists various menu items: 'All Projects', 'All Users', 'Time Reports', 'Task Reports', 'All Meetings', 'Company Config.', and 'Statistics'. The central form is titled 'User Information' and 'AssignedProjects'. It contains several input fields for user details: 'Userid' (mehmet), 'Password' (*****), 'Confirm Password' (*****), 'First Name' (Mehmet Remzi), 'Last Name' (DOGAR), 'E-mail' (mehmet@ddsoft.com), 'Address' (empty), and 'Phone' (03127153245). There are also dropdown menus for 'E-mail notification', 'Tasks Per Page', 'Projects Per Page', 'Project Upon Login', 'Number of Months', and 'Number of Weeks'. A 'Profile' section includes radio buttons for 'Administrator' (selected) and 'Normal User', and a 'User Directory' dropdown. At the bottom right, there are three buttons: 'SAVE', 'SAVE+ASSIGN', and 'CANCEL'.

User Information	
Userid:	mehmet
Password:	*****
Confirm Password:	*****
First Name:	Mehmet Remzi
Last Name:	DOGAR
E-mail:	mehmet@ddsoft.com
Address:	
Phone:	03127153245
Profile:	<input checked="" type="radio"/> Administrator <input type="radio"/> Normal User
User Directory:	

SAVE SAVE+ASSIGN CANCEL

Assigned Projects Interface



The screenshot shows a web application interface for managing projects. At the top, a blue header bar contains a welcome message "Welcome Dogan". Below this is a navigation bar with tabs for "Administration", "Project1", and "Help". On the left side, there is a vertical menu with options: "All Projects", "All Users", "Time Reports", "Task Reports", "All Meetings", "Company Config.", and "Statistics". The main content area has two tabs: "User Information" and "AssignedProjects". The "AssignedProjects" tab is active, displaying a form with the following fields:

Project:	<input type="text"/>
Project Manager:	<input type="checkbox"/>
Task Reviewer:	<input checked="" type="checkbox"/>
Can arrange meeting:	<input checked="" type="checkbox"/>
Can open this project:	<input checked="" type="checkbox"/>
Task editing right:	<input type="text"/>

At the bottom right of the form, there are two buttons: "UPDATE" and "CANCEL".

New Task Interface

The screenshot shows a web application interface for creating a new task. At the top, a blue header bar contains the text "Welcome Dogan" on the right. Below the header is a navigation bar with tabs: "Administration", "Project1", and "Help". The main content area has a left sidebar with a list of menu items: "My Tasks", "Tasks", "Forum", "Statistics", "Gantt Charts", "Time Reports", "Task Reports", "Meetings", and "Notifications". The main area is titled "New Task" and contains several sub-tabs: "New Task", "Assigned Users", "Attachments", and "Task History". The "New Task" tab is active, displaying a form with the following fields:

- Task Creator:** A text box containing "Dogan".
- Task Summary:** A text box containing "Design report".
- Details:** A large text area containing the text "Write a Design Report for the 491 project".
- Task Status:** A dropdown menu.
- Percentage Done:** A dropdown menu.
- Hours Worked on:** A text box containing "0.00".
- Task Group:** A dropdown menu.
- Task Type:** A dropdown menu.
- Task Priority:** A dropdown menu.
- Start Date:** A date picker showing "24.12.2004".
- Due Date:** A date picker showing "9.01.2005".
- Task Reviewer:** A dropdown menu.

At the bottom of the form are three buttons: "SAVE", "SAVE+ASSIGN", and "CANCEL".

Assigned Users Interface

Welcome Dogan

AdministrationProject1Help

My Tasks

Tasks

Forum

Statistics

Gantt Charts

Time Reports

Task Reports

Meetings

Notifications

New TaskAssigned UsersAttachmentsTask History

Project:Project 1

Task:Design Report

User	Assigned
mehmet	<input checked="" type="checkbox"/>
dogan	<input type="checkbox"/>
firat	<input checked="" type="checkbox"/>

SAVE

SAVE+ATTACH

CANCEL

New Meeting Interface

Welcome Dogan

AdministrationProject1Help

My Tasks

Tasks

Forum

Statistics

Gantt Charts

Time Reports

Task Reports

Meetings

Notifications

New MeetingAttachmentsParticipants

Meeting Summary:Design Report Meeting

Details:

We will talk about what we have done and what we will do.

Date Option1:26.12.2004/16.00

Date Option2:26.12.2004/14.00

Date Option3:26.12.2004/9.00

Date Option4:27.12.2004/9.00

Date Option5:27.12.2004/12.00

Last Reply Date:25.12.2004/16.00

Duration:1.00hours

SAVE

SAVE+ATTACH

CANCEL

Meeting Options Interface

Welcome Firat

AdministrationProject1Help

My Tasks
Tasks
Forum
Statistics
Gantt Charts
Time Reports
Task Reports
Meetings
Notifications

Select Meeting OptionsAttachmentsParticipants

Meeting Summary:Design Report Meeting

Details:

We will talk about what we have done and what we will do.

Date Arranged:24.12.2004/12.30

Last Reply Date:25.12.2004/16.00

Arranger of Meeting:Dogan

Date Option1:26.12.2004/16.00

Date Option2:26.12.2004/14.00

Date Option3:26.12.2004/9.00

Date Option4:27.12.2004/9.00

Date Option5:27.12.2004/12.00

My Choice1:

My Choice2:

My Choice3:

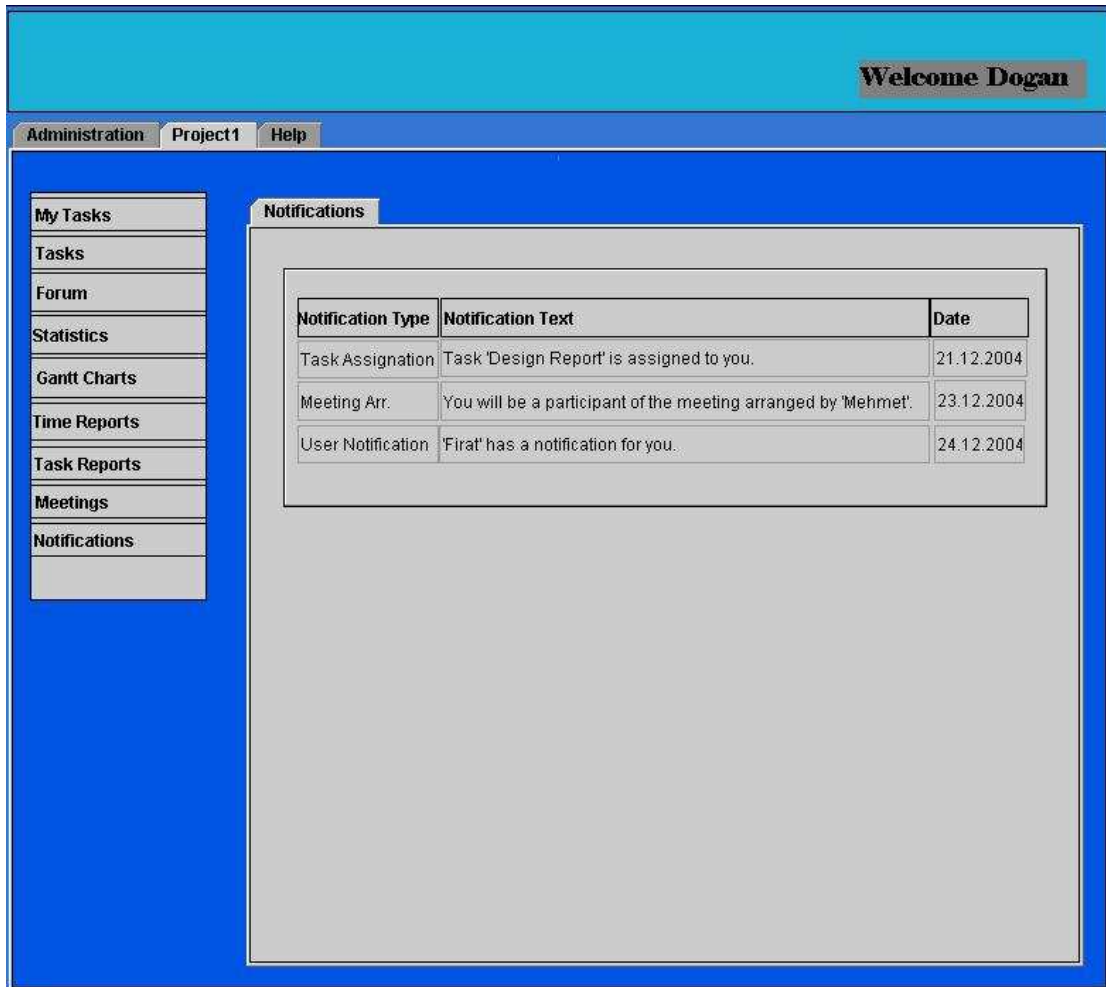
My Choice4:

My Choice5:

SAVE

CANCEL

Notification Interface



5. TESTING ISSUES

DProject is a web-based multi-user system containing of different modules each specialized for performing a different group of operations. Therefore, the testing issues can be divided into 3 main categories:

1) Testing Security Issues

The web-based nature of DProject makes security one of the most important aspects of it. Therefore, a specific testing module is devoted to security issues. Mainly black-box testing will be used to determine the flaws in the encryption and decryption of data. Specialized testing modules can be prepared if different cryptography techniques is decided to be used.

2) Testing Database Integrity

DProject makes heavy use of the database in means of the data flowing from/to the database. This heavy interaction makes database integrity testing worth considering as a specific test module. First thing to be tested is the connection to the database. The interaction between the modules and the database will be used along with the architectural integrity testing. The interaction between each module and the database will be tested by specific cases included in these testing modules, which will be white-box mainly.

3) Testing Architectural Integrity

As specified earlier, DProject shows a high compartmentalization in means of specific modules for different system capabilities. These modules will be in high interaction among themselves and also with the database. The modules for testing architectural integrity are combined with the modules for testing database integrity. Each module will be tested by a different white-box testing case to see the problematic issues, if exists.

6. PROJECT SCHEDULE

Project schedule can be found in the Appendix since it does not fit in an ordinary A4 page.