SOFTWARE REQUIREMENTS SPECIFICATION

Guideal

Date of Issue:
08.01.2016

Group Name: Malum

Project Group Members:
Semih Aktaş 1880913
Emre Külah 1881358
Görkem Özer 1881747
Yusuf Mucahit Çetinkaya 1881705

Version Control History:

Version	Primary Author(s)	Description of Version	Date Completed
1.0	Semih Aktaş Emre Külah Görkem Özer Yusuf Çetinkaya	Software Requirements Specifications	08.01.2016

Table of Contents

Table of Figures

1. INTRODUCTION

This section gives a scope description and overview of everything included in this SRS document. Also, the purpose for this document is described and a list of abbreviations and definitions are provided.

1.1 Purpose

The purpose of this document is to give a detailed decsription of “Guideal” learning management system. It will explain the purpose and features of the system that will be developed. Also, it will focus on system constraint, functions that the software will have, interfaces and how it will interact with other external applications. This document is primarily prepared to inform clients about the “Guideal” system and provide a reference related to the system for the development team.

1.2 Scope

“Guideal is a web-based system that will allow instutions to manage their personal/departmental data, evaluate success status of units with respect to the standarts and get reports about personal/departmental improvement or recession according to obtained data. Users will be able to observe their statistics and get recommendations about which study/working areas they should focus on, according to their success analyses and objective criteria (standarts). Basically it will serve all learning management system functionalities; such as subscribing a training, giving tasks, submitting files to tasks, being graded etc. Additionally, user will be able to track her/his own progress over time and get recommendation for her/his future plans.

1.3 Product Overview

This section will give an overview of the whole system. The system will be explained in its context to show how the system interacts with other systems and introduce the basic functionality of it. It will also describe what type of stakeholders that will use the system and what functionality is available for each type. At last, the limitations for the system will be presented.

1.3.1 Product Perspective

This system will of consist of one part: web portal. Plug-ins can add new parts into the system such as android devices, scanner etc. The web portal will cover all the functionality of

the system. All creating, removing, updating and displaying facilities of entities are included in the web portal. Basic analysis features will come with core product.

The core part is basically facilitate a platform users to register, login and subscribe courses. Users can belong to user groups and departments. Roles are distributed to users according to privilages. These role mechanism makes the system more secure and robust. System will not be busy to do redundant and dangerous operations. System owner/director will give privilages to usergroups and/or users. Users will get privilages from her/his usergroup and also her/his exceptional privilages. As a result, the authorization mechanism is designed to make the system more dynamic.

Training times are seperated into terms and there is only one active term at a moment. There are subscribers of trainings in different relation levels: *Trainer, Trainee, Moderator, Spectator.* Trainings have task sets to store tasks in it during the term. Tasks have task units in it to make more accurate analyses and to track user progress. Measurement sets stand for implying standarts (such as ABET). Each task unit has relation with one or more measurement criteria which is required to detect the percentage of completed part of criteria.

“Guideal” will also provide an API for developers to use and develop new plug-ins for the system. This will make the system more powerful and meet special needs of organisations. Plug-ins can be installed later. API will support wide-range of technologies to develop and integrate new plug-ins to the system.

Since it is a data-centric product it needs to store data with as small as possible units to make more accurate analysis. Solid data should be kept for recommendation. These recommendations are based on previous experiences. Therefore, complete data storage is really significant for this purpose. Since this brings more work to users, the system should ease users’ job. Hence, user-interface will be clear and very user-friendly. User will not click more than two buttons to do or display something.

1.3.1.1 System Interfaces

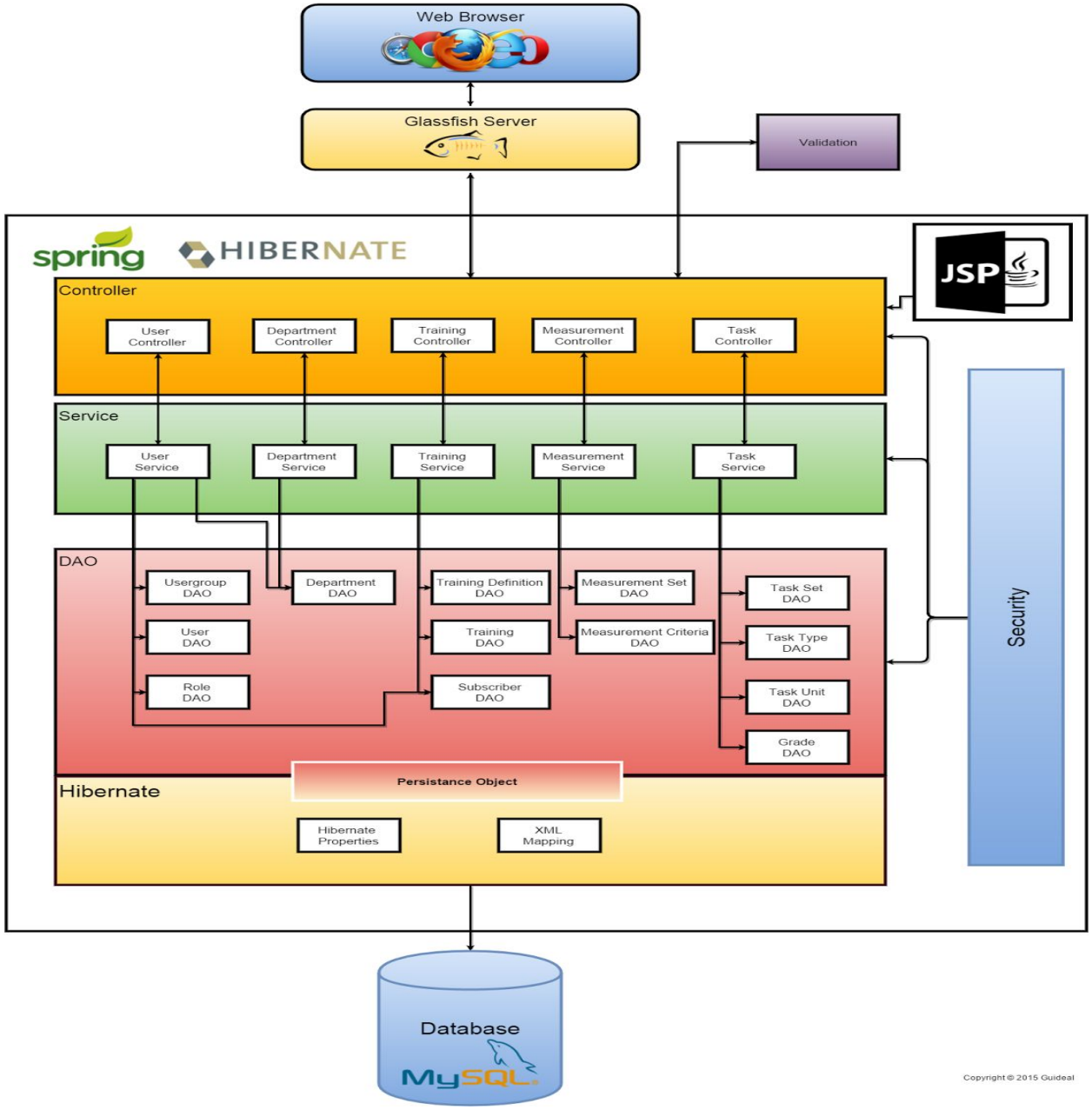


Figure-1: System Architecture

As you can see in figure 1 , system architecture is based on layered system architecture.

1.3.1.2 User Interfaces

In this part, the user-interface of the system will be introduced. Mainly, create, remove, update and display functions of system will be shown.

When user enter web page, Login Page(Figure-2) is shown. If the user is registered and login, then home page will be open.

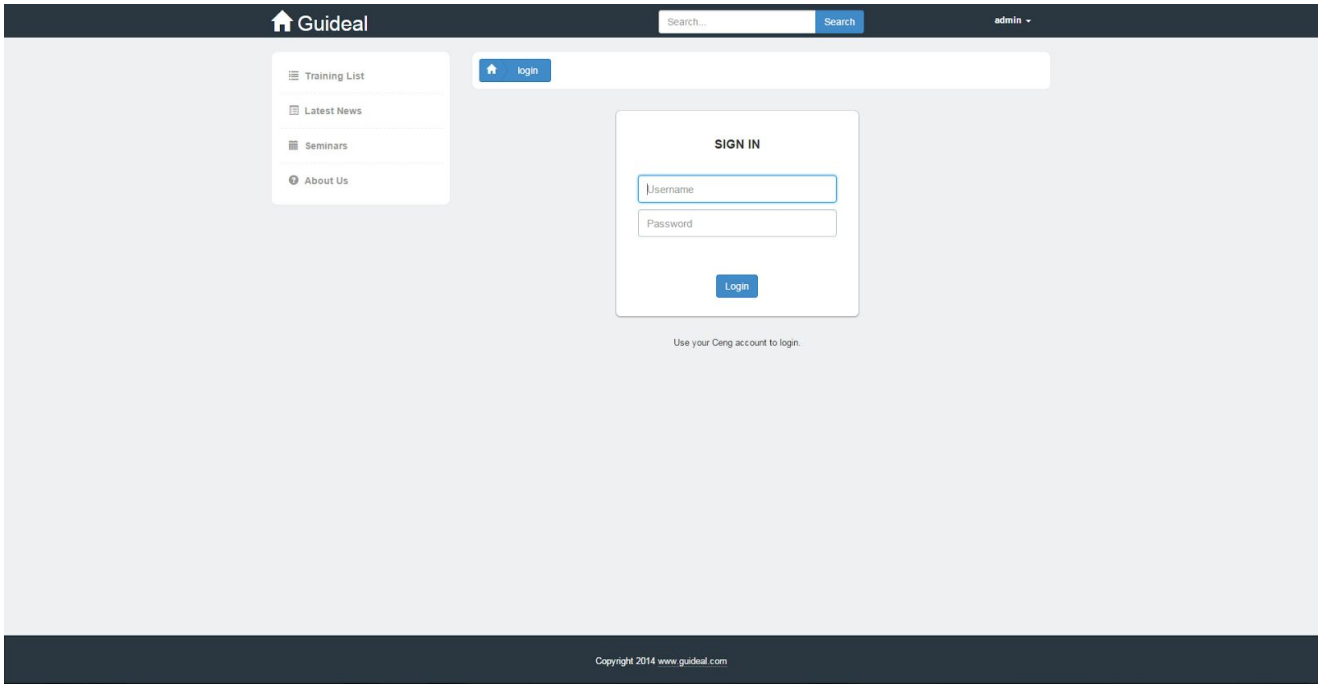


Figure-2: Login Page

When user logs out succesfully the system returns to the login page with a success message as in Figure-3.

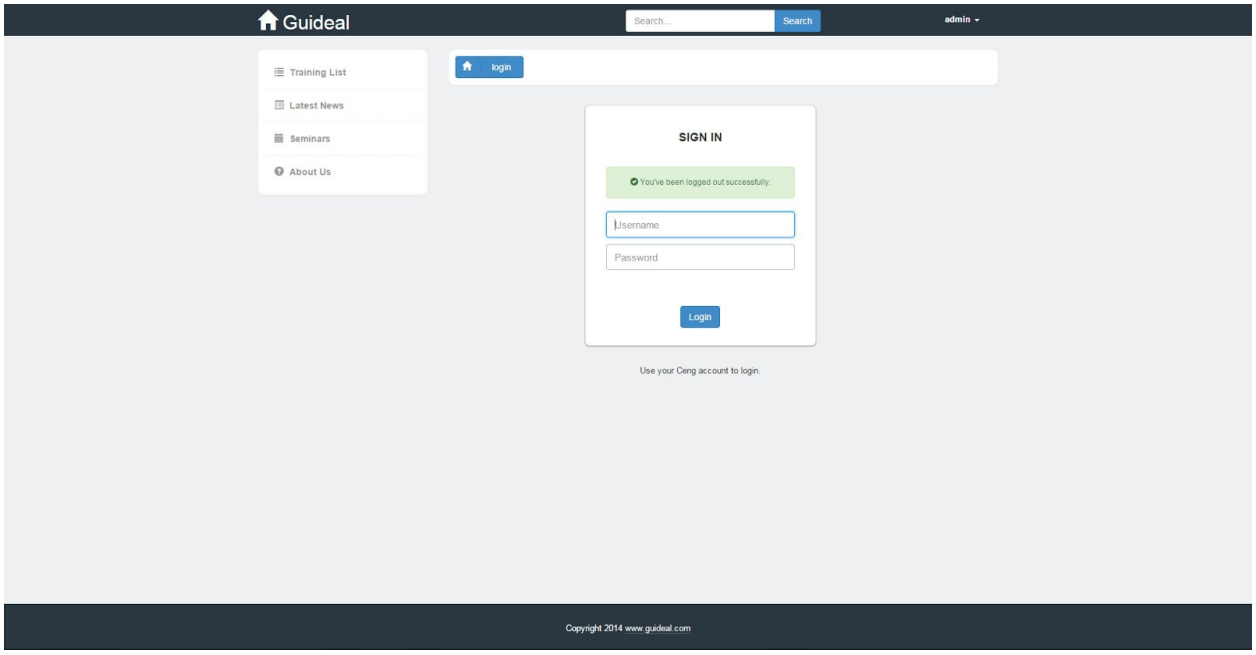


Figure-3: Login Page after Logout

All the functionalities are inserted into menu at the left side of the pages. In all list pages shortcuts are added into the rows. Also checkbox can be used to do multi operations. In addition, all lists include filter functionality.

1.3.1.2.1 User Pages

The first pages in the system are about user operations. In the registeration page (Figure-4) is used to register to the system. Update page (Figure-5) can be used to update the informations of the users. The list of the users are shown in the user list page (Figure-6). The profile page (Figure-7) is contain all the information about the users.

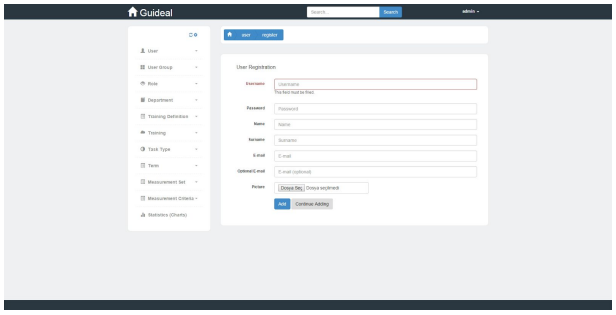


Figure-4: Registration Page

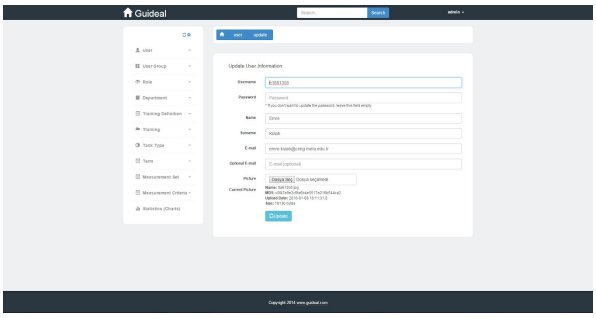


Figure-5: Update User Page

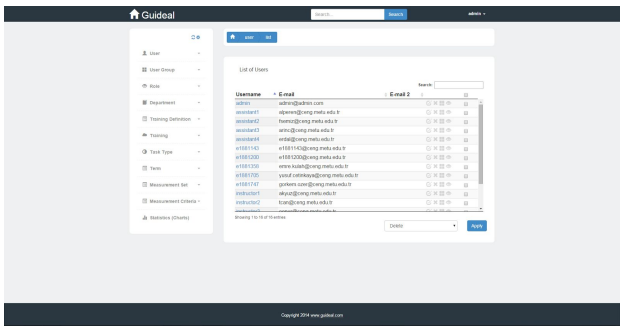


Figure-6: User List Page

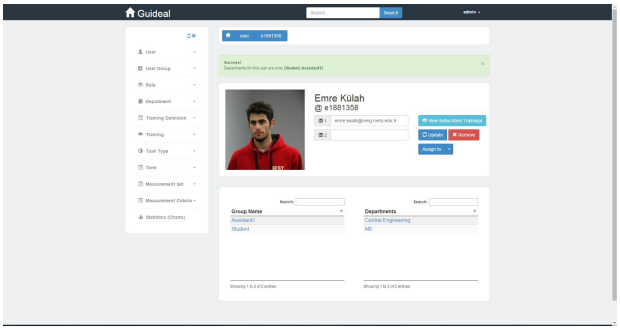


Figure-7: User Profile Page

There are three pages to assign groups, departments and roles to users. They are reachable from user profiles and also directly from user list.

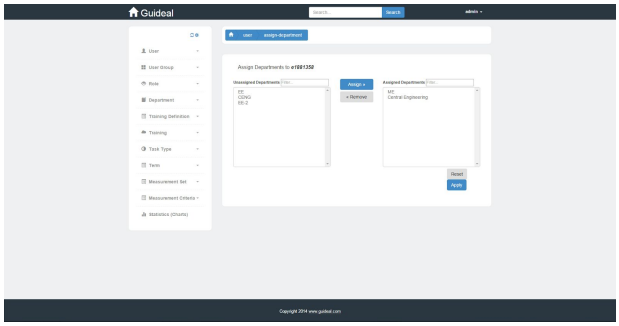


Figure-8: Assign Department to User Page

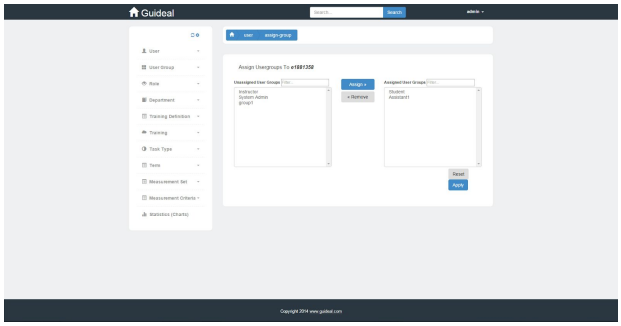


Figure-9: Assign User Group to User Page

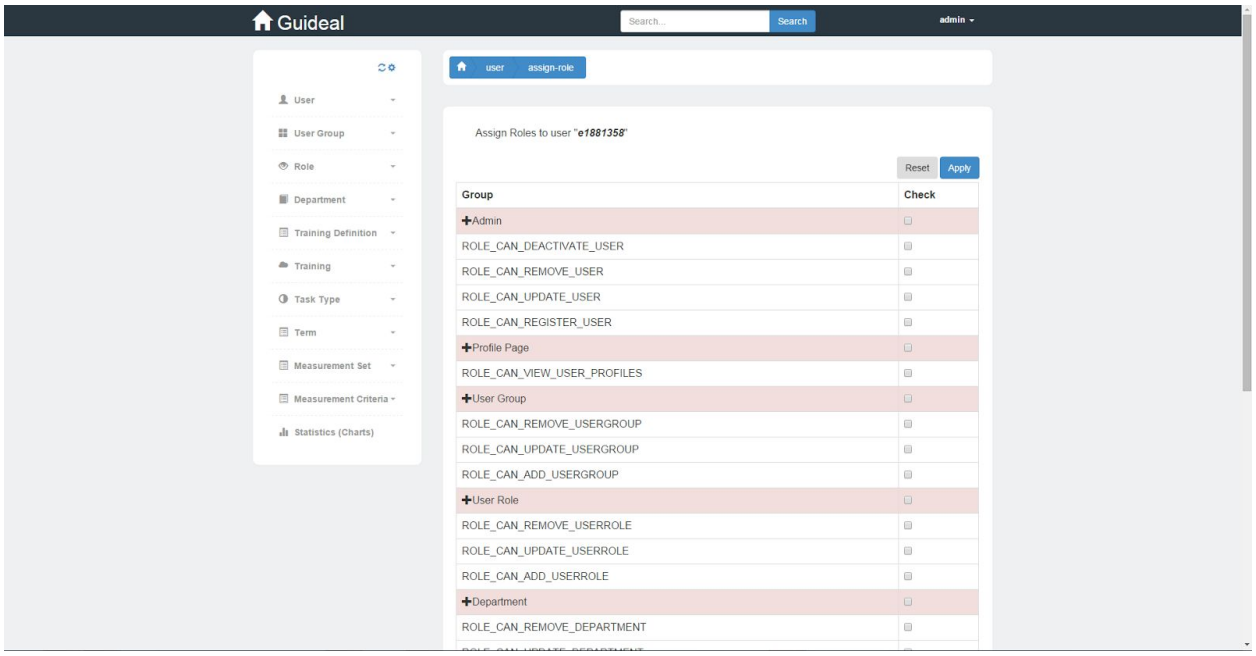


Figure-10: Assign Role to User Page

1.3.1.2.2 User Group Pages

The system includes basic CRUD operation pages of user groups. User groups are defined groups to assign users grouped roles.

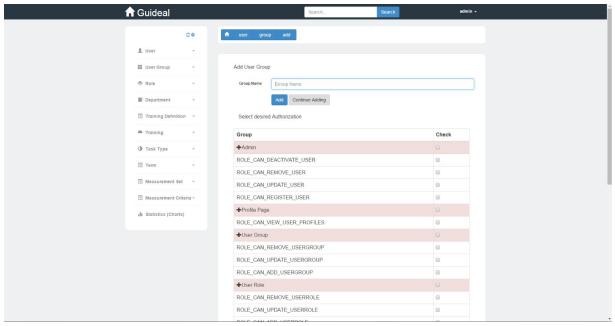


Figure-11: Add User Group Page

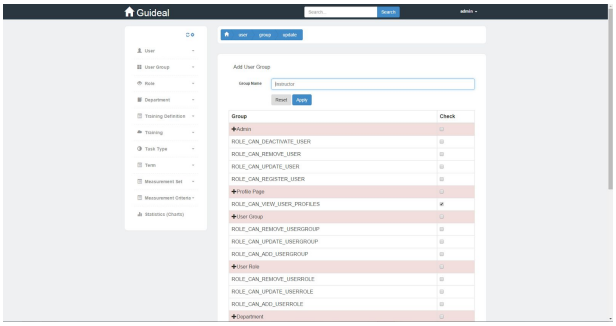


Figure-12: Update User Group Page

When adding new user group admin will decide which roles he will add to the system.

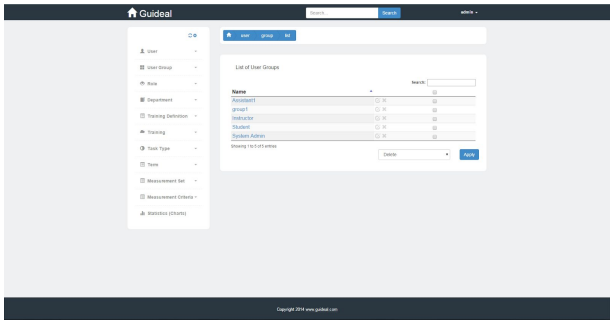


Figure-13: User Group List Page

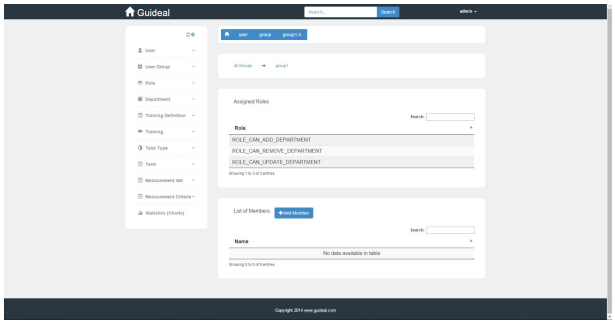


Figure-14: User Group Profile Page

The user group profile page shows which roles the user group has and the list of members which are assigned to this group.

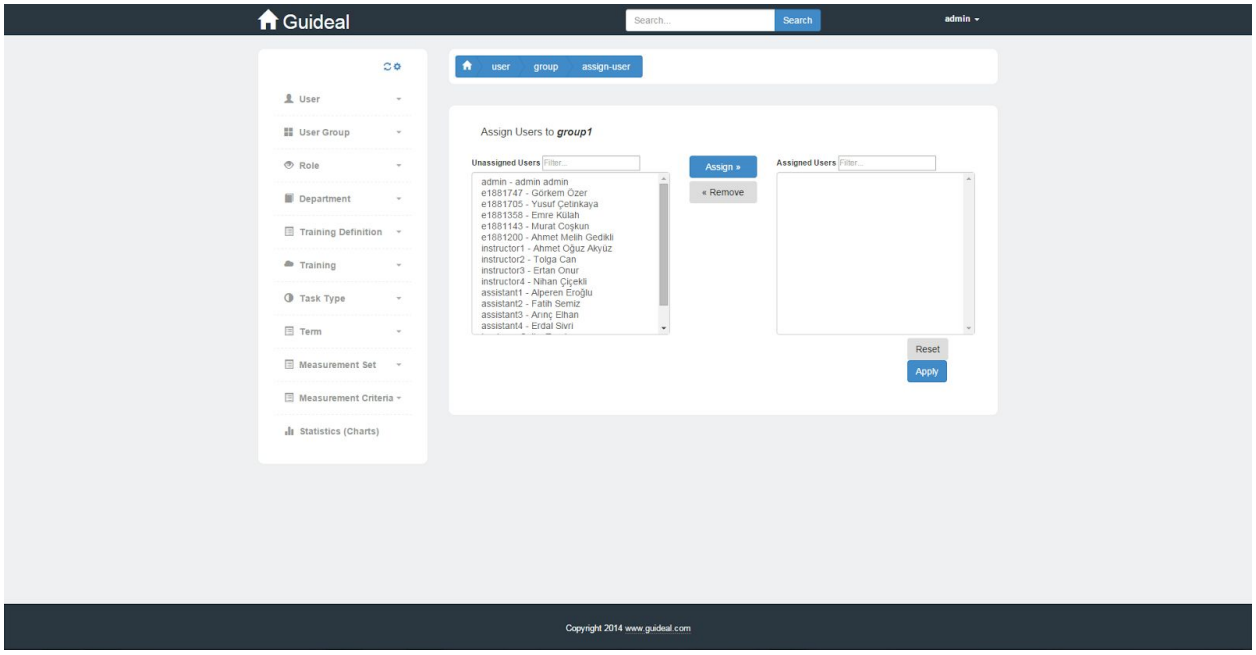


Figure-15: Assign User to User Group Page

1.3.1.2.3 Role List Page

This page shows the list of the roles which can be assigned to the users and user groups.

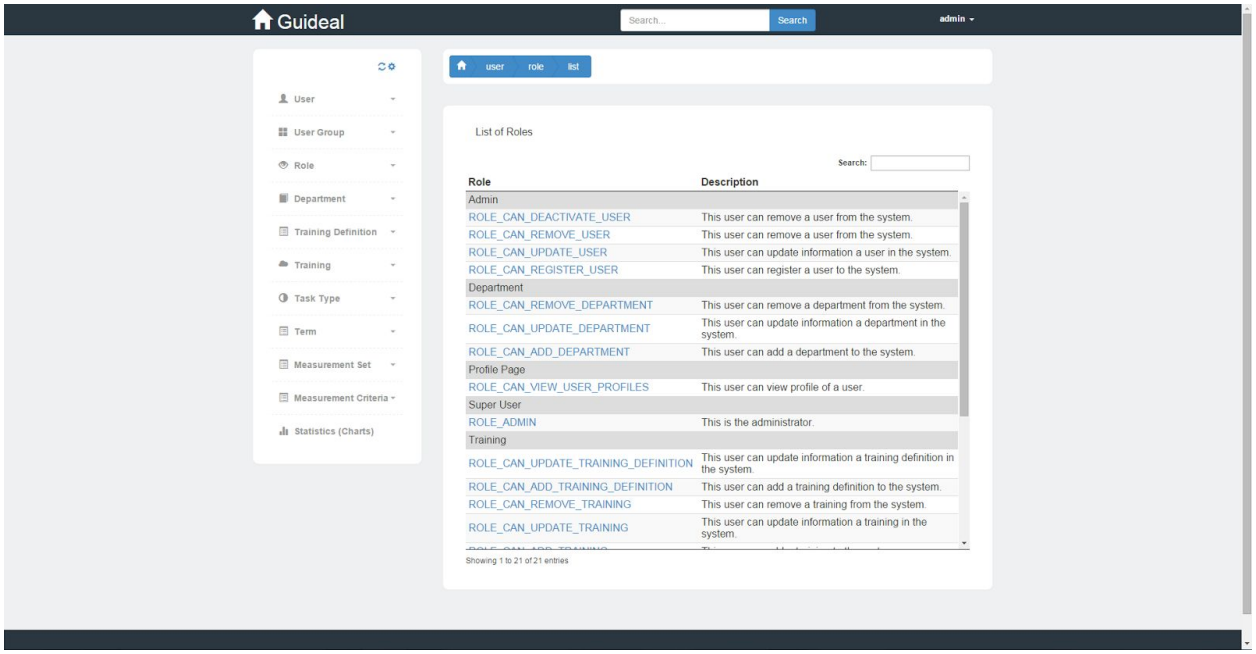


Figure-16: Role List Page

1.3.1.2.4 Department Pages

The system includes basic CRUD operation pages of departments. Departments are the most extensive groups of the institutions as used in the real life.

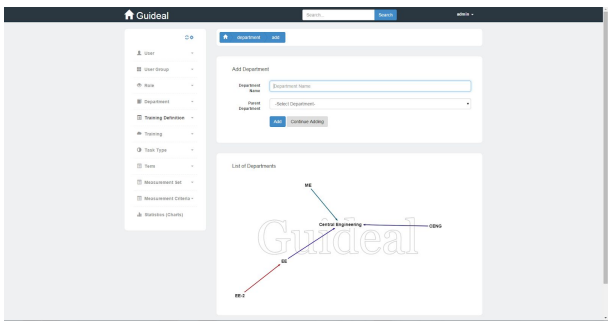


Figure-17: Add Department Page

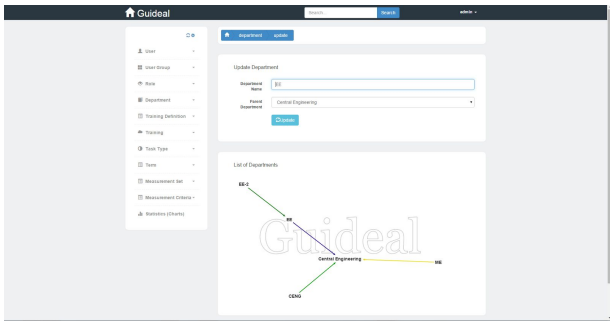


Figure-18: Update Department Page

In the department pages, graphs are used to show the structures. In the add department page (Figure-17) and update department page (Figure-18) User can add parent department by selecting from graph.

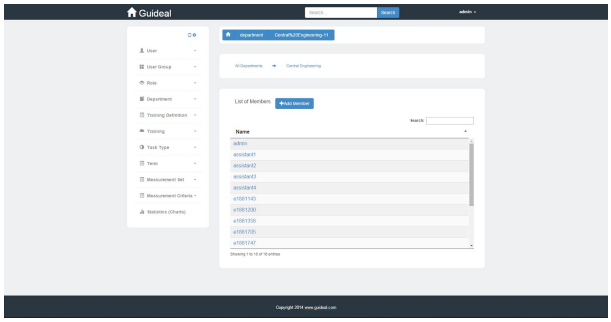


Figure-19: Department List Page

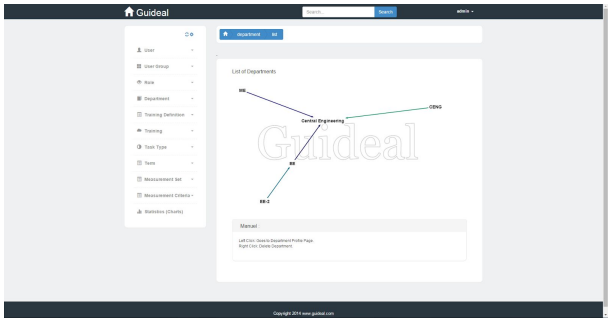


Figure-20: Department Profile Page

1.3.1.2.4 Training Definition Pages

The system includes basic CRUD operation pages of training definitions. Training definitions include basic information about trainings. Wysiwyg editor is used to create more explanatory definitions.

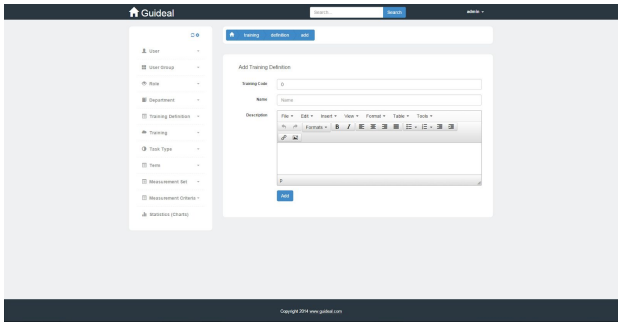


Figure-21: Add Training Definition Page

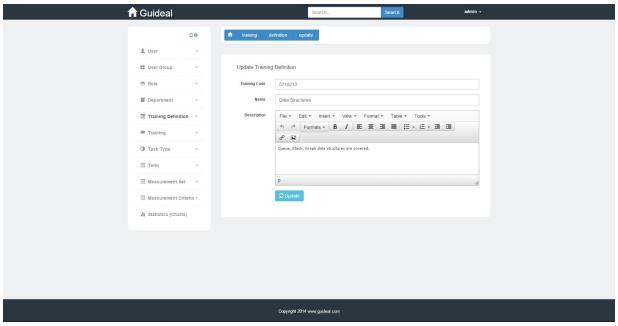


Figure-22: Update Training Definition Page

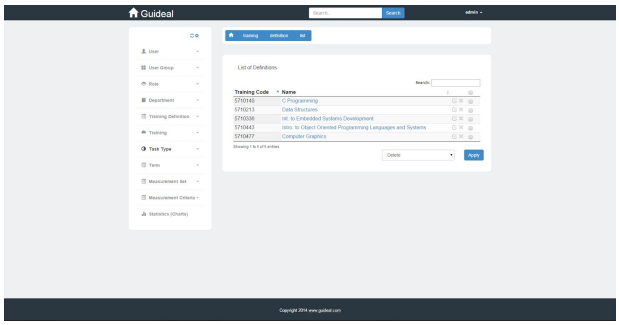


Figure-23: Training Definition List Page

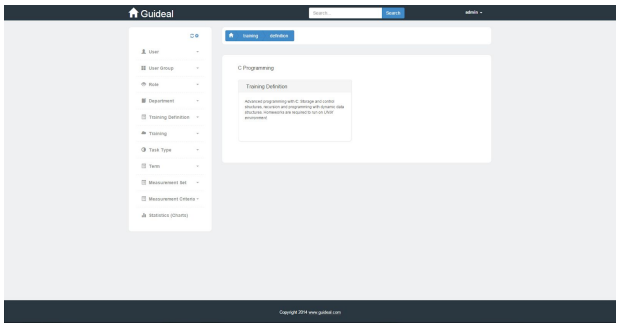


Figure-24: Training Definition Profile Page

1.3.1.2.5 Training Pages

These pages are consist of CRUD operations and more complex operations of trainings. The CRUD pages are designed as the rest of the system.

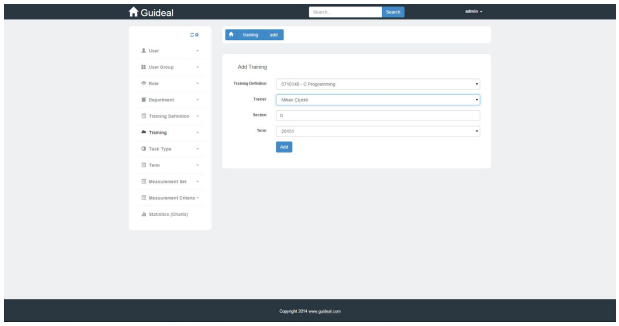


Figure-25: Add Training Page

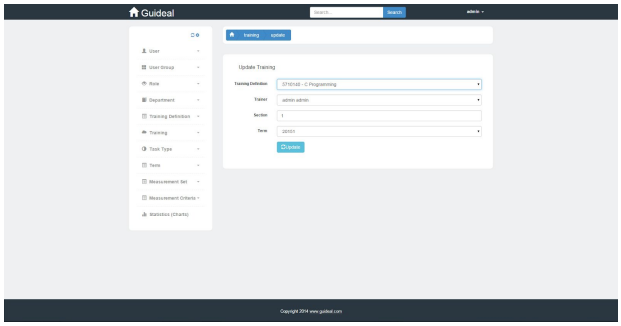


Figure-26: Update Training Page

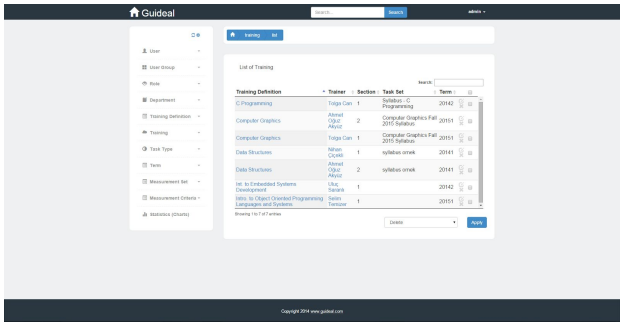


Figure-27: Training List Page

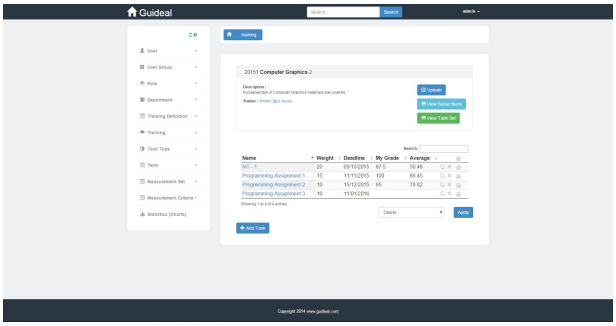


Figure-28: Training Profile Page

The training profile page (Figure-28) includes the tasks list assigned to the training. The task set and subscriber operations are managed from the training pages. If there is no task set user can create from the profile. If there is user can add tasks. In addition, listing and editing subscribers are added as button into training profile page.

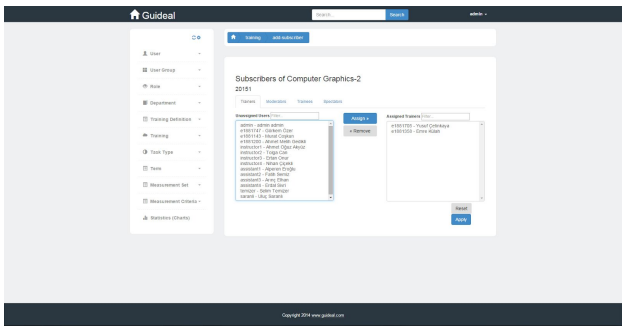


Figure-29: Add Subscriber Page

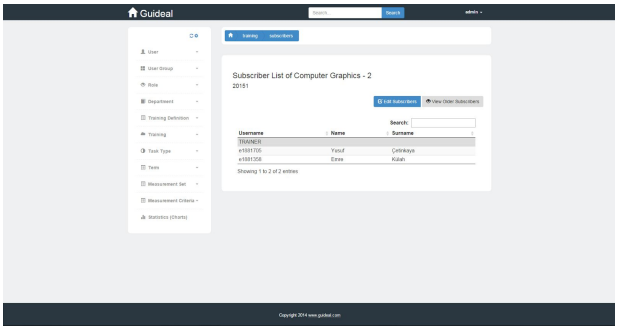


Figure-30: Subscriber List Page

Add Subscriber page (Figure-29) includes 4 tabs: trainer, moderator, trainee and spectators. User can add and remove members from these tabs. Subscriber list page (Figure-30) shows the list of members as grouped.

1.3.1.2.6 Task Type Pages

The system includes basic CRUD operation pages of task types.

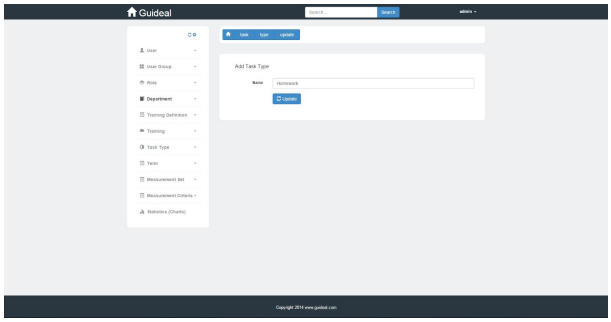


Figure-31: Add Task Type Page

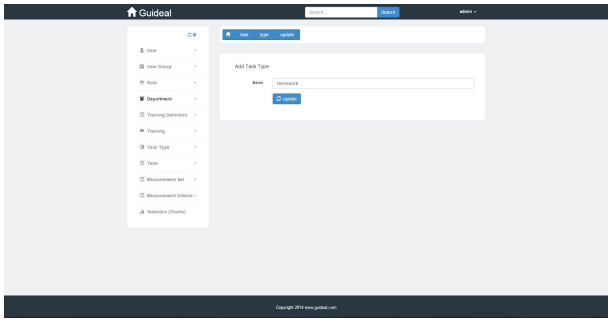


Figure-32: Update Task Type Page

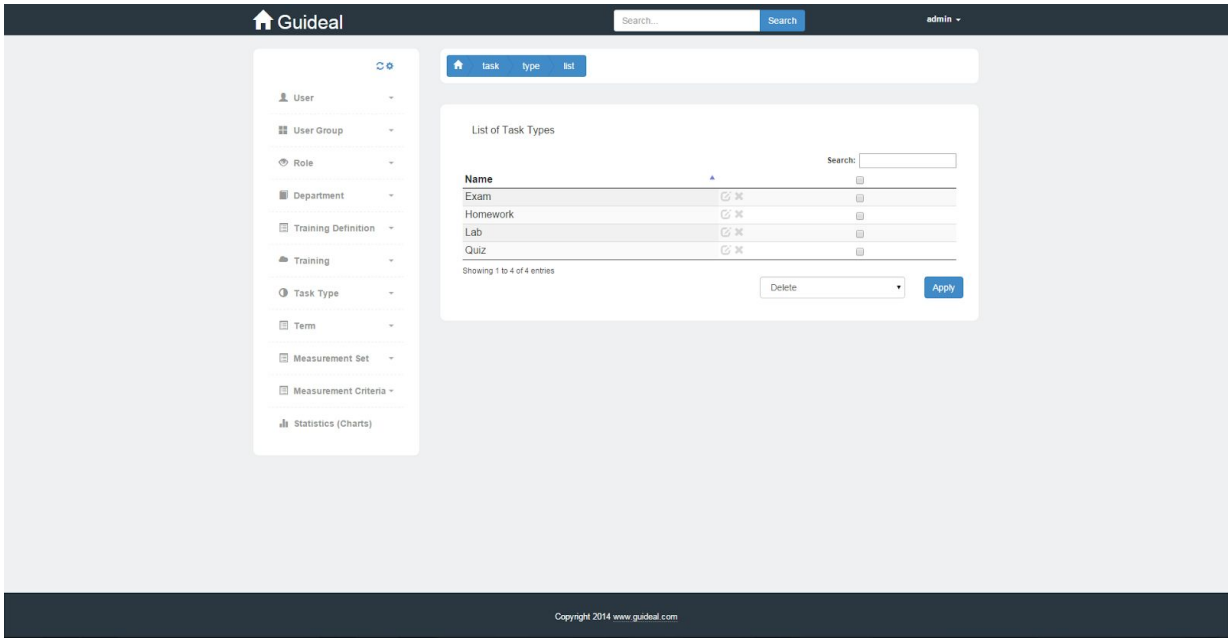


Figure-33: Task Type List Page

1.3.1.2.7 Term Pages

The system includes basic CRUD operation pages of terms.

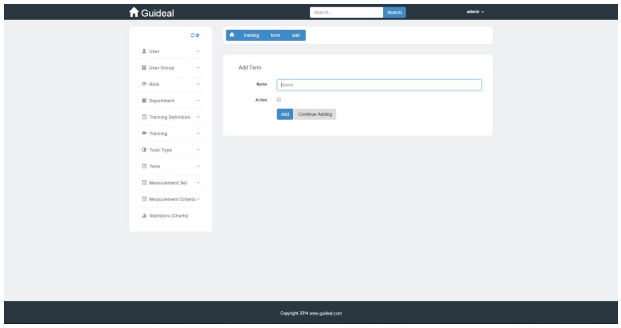


Figure-34: Add Term Page

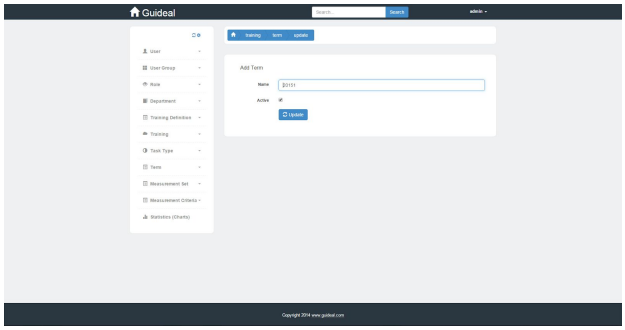


Figure-35: Update Term Page

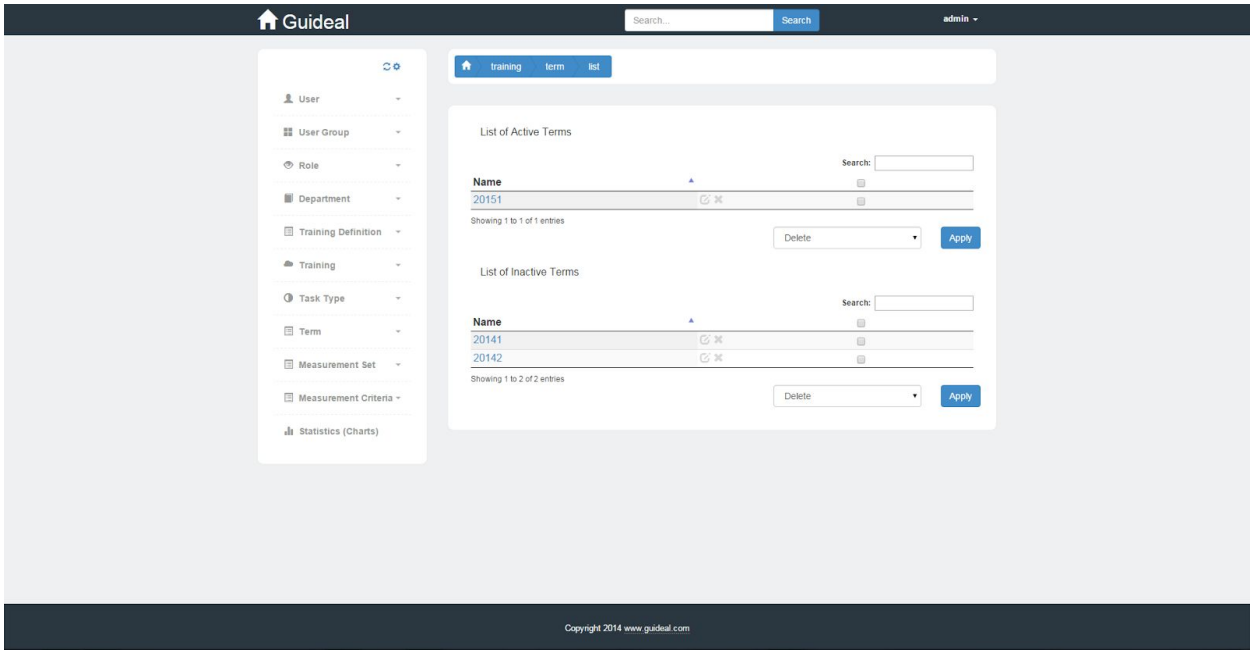


Figure-36: Term List Page

Term list page (Figure-36) splitted into 2 parts. The first one shows the active terms and second shows inactive ones.

1.2.1.2.8 Measurement Set Pages

The system includes basic CRUD operation pages of terms.

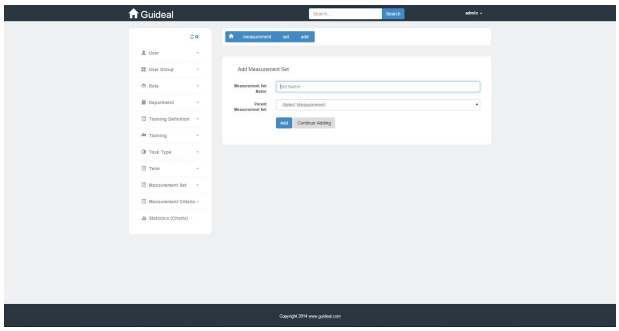


Figure-37: Add Measurement Set Page

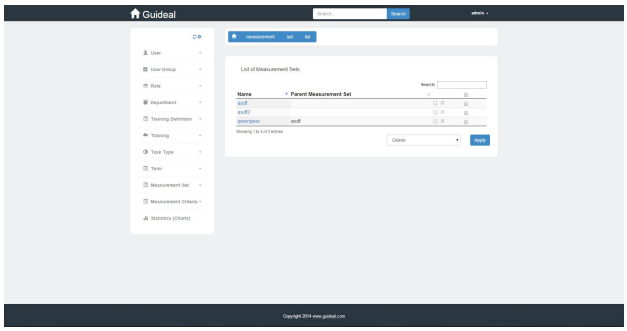


Figure-38: Measurement Set List Page

1.2.1.2.8 Measurement Criteria Pages

The system includes basic CRUD operation pages of terms.

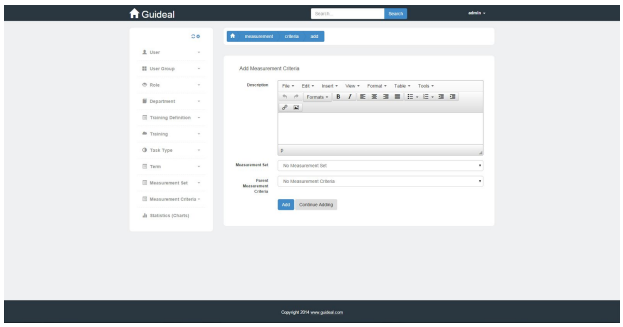


Figure-39: Add Measurement Criteria Page

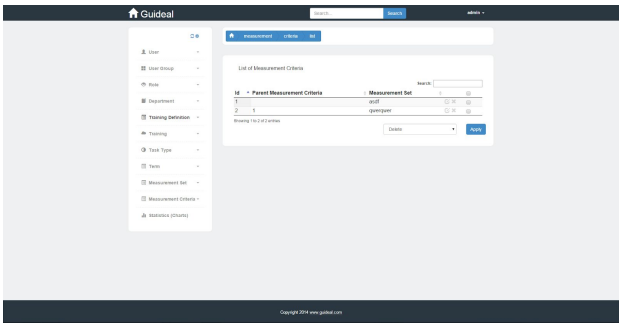


Figure-40: Measurement Criteria List Page

1.2.1.2.9 Grade Page

The system includes grade pages for tasks of trainings.

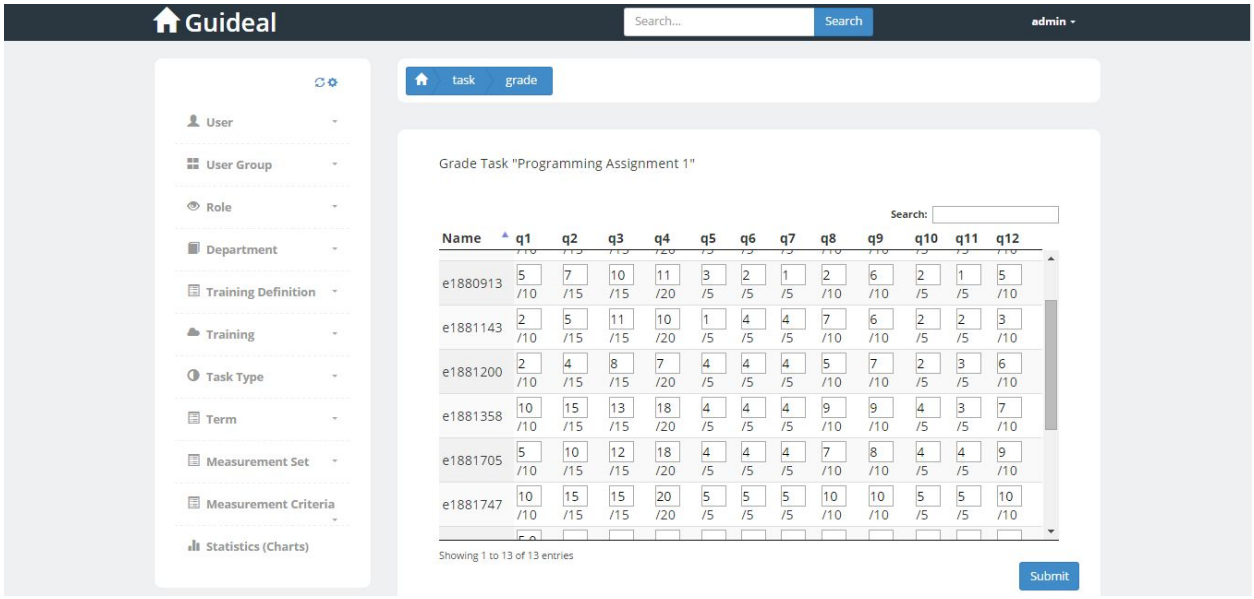


Figure-41: Grade Page

1.3.1.3 Hardware Interface

The main part of the system does not include any hardware devices and interfaces. However, some of the plug-ins will be runnable on the android devices which means there will be android application which will has similar design with web portal.

1.3.2 Product Functions

The users will be able to register to the system on their own or by an admin. Users who have privilage for department assignment will assign these users to departments. This scenario is also same for usergroups and roles. Time is seperated into terms to follow trainings easier. There is only one active term at a time.

There are training definitions which includes detailed description and the aim of the training. A user can give a training of one of these training definitions in a term. Users can subscribe trainings with 4 different roles; *Trainer, Trainee, Moderator, Spectator.* Trainer is obviously the trainer of the training. S/he has all privilages for that training. Trainee is the subscriber who is graded according to some tasks. Moderator is the assistant of the trainer for that training. S/he can moderate the training, enter the grades and arrange tasks. Spectators are the users that want to follow the course without being evaluated.

During the evaluation process predefined measurement criterias are associated with the task units. According to these relations system detects the percentage of success for each criteria according to received grade from the task units.

API support stands for developers to extend the systems functionality for special needs. Developers can choose any technology for using this API.

1.3.3 User Characteristics

There is no seperation between users. Users diffrentiate from each other for privilages according to given roles. Roles are dynamic and predetermined according to system parts such as; CAN_REGISTER_USER, CAN_ASSIGN_DEPARTMENT, CAN_ASSIGN_SUBSCRIBER etc.

New roles can be added this system. By helping this, users can access whatever admin wants.

Logically system has trainers, trainees, moderators and spectators. However, a user can be trainee in a training while being a trainer in another training. Users have usergroups to have common privilages for groups and display in their profile.

Users will subscribe a training and have to complete given tasks. Grading is done by training’s trainers and moderators. Users will track their status according to completed tasks and their success on them. This status can be seen by the related privilage owners also. There will be any kind of analysis on the system. Trainers will also be measured as well as trainees.

1.3.4 Limitations

The web application will be developed in Spring framework with Java language. Database operations are done via Hibernate and direct SQL, HQL queries.

The GUI will be designed with Bootstrap framework.

The internet access is required for web application to operate remote.

1.4 Definitions

Term	Definitions
Guideal	A system that provide a platform to institutions to track trainings and users' success status to recommend new courses & fields to them.
Role	Privilage which is given to users, required to do things.
User group	Group that users, which have common roles, attend.
Training	A training that includes subscribers, task sets.
Subscriber	User that subscribe a training. Subscribers can be in type: Trainer, Trainee, Moderator, Spectator.
Trainer	Trainer of a training.
Trainee	Subscriber whose tasks (assignments) are graded and measured.
Moderator	Moderator of the training, who helps trainer in task create, update, delete operations.
Spectator	Subscriber that follows training content but his/her tasks are not graded.
Task Set	A set which gathers the tasks in it. A task set can be followed by more than one training.
Task	A task which is expected to done by trainee of a training which follows related task

	set.(e.g homework-1)
Task Unit	Smallest part that can be measured. Located in a task.(e.g question-1)
Measurement Set	A set consist of measurement criteria. It is basically the name of standart.(e.g ABET)
Measurement Criteria	The criteria that is wanted to be measured for trainees.(e.g pid-12)
Term	The part of a time that is system is actively used.

2.REFERENCES

This document is prepared according to the ISO/IEC/IEEE 29148 (2011) Systems and Software Engineering – Lifecycle Processes – Requirements Engineering.

3. SPESIFIC REQUIREMENTS

3.1 External Interfaces

3.1.1 User Interface

In this section, details of the section 1.3.1.2 which have already been explained above will be introduced.

3.1.2 Hardware Interface

In this section, details of the section 1.3.1.3 which have already been explained above will be introduced.

3.2 Functions

In this section, the fundamental actions that have to take place in the software in accepting and processing the inputs and in processing and generating the outputs are defined.

3.2.1 Login

Use Case Name	**UC001_login**
XRef	
Actor	**Users who have has login role**
Preconditions	**The url should be known by users**
Trigger	**This use case is initiated when user presses “Login” button**
Basic Flow	**1. System provides “Username” and “Password” fields to user.** **2. The user fills these fields** **3. The user presses the “Login” button.** **4. System sends those information to Database and checks.** **5. System redirects user to the home page**
Exception Flow	**If the database cannot match username or password with its records or if user don’t have login role, then system asks the user to enter correct information**
Post conditions	**User logged in and is redirected to home page**

3.2.1.1 Functional Requirement

User should enter his/her username to the “username” field.

3.2.1.2 Functional Requirement

User should enter his/her password to the “password” field.

3.2.1.3 Functional Requirement

After validating username and password, system shall check the username and password information with database and if user is registered, then checks the roles of the user.

3.2.1.4 Functional Requirement

System redirects the user to home page when the user presses “Login” button.

3.2.1.5 Functional Requirement

System wants to correct username and password when the user presses “Login” button if there is any error such validation, database control.

3.2.2 View Profile

Use Case Name	**UC002_View Profile**
Actor	**Users who have has “ROLE_CAN_VIEW_USER_PROFILES”**
Preconditions	**The url should be known by users**
Trigger	**This use case is initiated when user presses one of the username from user list**
Basic Flow	**1. System should redirect user to desired user page** **2. System should take user information from database and show them** **1. Name & Surname** **2. Username** **3. Email** **4. Optional Email** **5. Group names that user is registered** **6. Departments that user is registered**
Post conditions	**The user is redirected to desired user profile page and see related information about desired user**

3.2.2.1 Functional Requirement

System should check user roles. User should have “ROLE_CAN_VIEW_USER_PROFILE” role.

3.2.2.2 Functional Requirement

System should take user information from database.
That information:

1. Name & Surname
2. Username
3. Email
4. Optional Email
5. Group names that user is registered
6. Departments that user is registered

3.2.2.3 Functional Requirement

System should redirect user to desired user profile page and show related information.

3.2.3 Logout

Use Case Name	**UC003_Logout**
Trigger	**User presses “logout” button**
Basic Flow	**1. User presses “logout” button** **2. System should redirect user to “home“ page**
Exception Flow	
Post conditions	**The user is redirected to “home” page and user logged out.**

3.2.3.1 Functional Requirement

System should provide logout button

3.2.3.2 Functional Requirement

When user presses logout button, system shall logged out him and redirect user to “home Page”.

3.2.4 Add User

Use Case Name	**UC101_Add User**
Actor	**Users who have has “ROLE_CAN_REGISTER_USER”**
Preconditions	**The url should be known by users**
Trigger	**This use case is initiated when user presses “Add” or “Continue Adding” button**
Basic Flow	**1. System provides “Username”, “Password”, “Name”, “Surname”, “E-mail”, “Optional- Email” and “Picture” fields to user.** **2. The user fills these fields.** **3. The user presses the “Add” or “Continue Adding” button.**

	4. System sends those information to Database and saves. **5. System redirects user to the user profile page or register page.**
Exception Flow	**If the username already registered then system sends exception** **If "E-mail", "name", "surname", "username" or "password" one of these fields is empty then system sends exception**
Post conditions	**Desired user is registered and user is redirected to register page or user profile**

3.2.4.1 Functional Requirement

System should check user roles. User should have "ROLE_CAN_REGISTER_USER" role.

3.2.4.2 Functional Requirement

User should fill "Username", "Password", "Name", "Surname", "E-mail", "Optional- Email" and "Picture" fields.

3.2.4.3 Functional Requirement

System should validate filled fields.

Username should be unique in Database.

Password should contain at least 8 characters.

Name and surname field must not be empty.

Email regex format : _*@_* Email field must not be empty.

Optional E-mail format : _*@_* It can be null.

3.2.4.4 Functional Requirement

System should save that user in database.

3.2.4.5 Functional Requirement

System should redirect user to user register page if user presses "Continue Adding" button.

3.2.4.6 Functional Requirement

System should redirect user to user page which was added if user presses "Add" button.

3.2.1.7 Functional Requirement

System should show exception message if there is any error.

3.2.5 Add Department

Use Case Name	**UC102_Add Department**
Actor	**Users who have has "ROLE_CAN_ADD_DEPARTMENT"**
Preconditions	**The url should be known by users**
Trigger	**This use case is initiated when user presses "Add" or "Continue Adding" button**
Basic Flow	**1. System provides "Department Name" and "Parent Department" fields to user.** **2. The user fills these fields.** **3. The user presses the "Add" or "Continue Adding" button.** **4. System sends these information to Database and saves.** **5. System redirects user to the department profile page or add department page.**
Exception Flow	**If department name is empty then system sends exception**
Post conditions	**Desired department is registered and user is redirected to add department page or department profile**

3.2.5.1 Functional Requirement

System should check user roles. User should have "ROLE_CAN_ADD_DEPARTMENT" role.

3.2.5.2 Functional Requirement

System should get department list from database for "Parent Department" field.

3.2.5.3 Functional Requirement

System should validate department name.
It must not be empty
It will be string.

3.2.5.4 Functional Requirement

System should save that department in database.

3.2.5.5 Functional Requirement

System should redirect user to add department page if user presses "Continue Adding" button.

3.2.5.6 Functional Requirement

System should redirect user to department profile which was added if user presses “Add” button.

3.2.5.7 Functional Requirement

System should show exception message if there is any error.

3.2.6 Add Usergroup

Use Case Name	**UC103_Add Usergroup**
Preconditions	**The url should be known by users**
Trigger	**This use case is initiated when user presses “Add” or “Continue Adding” button**
Basic Flow	**1. System provides “Group Name” field and role list to user.** **2. The user fills “Group Name” field and selects roles from role list** **3. The user presses the “Add” or “Continue Adding” button.** **4. System sends these information to Database and saves.** **5. System redirects user to the usergroup profile page or add usergroup page.**
Exception Flow	**If usergroup name is empty then system sends exception message**
Post conditions	**Desired usergorup is registered and user is redirected to add usergroup page or usergroup profile**

3.2.6.1 Functional Requirement

System should check user roles. User should have “ROLE_CAN_ADD_USERGROUP” role.

3.2.6.2 Functional Requirement

System should get role list from database.

3.2.6.3 Functional Requirement

System should validate usergroup name.
It must not be empty. It will be string.

3.2.6.4 Functional Requirement

System should save that usergroup in database.

3.2.6.5 Functional Requirement

System should redirect user to add usergroup page if user presses “Continue Adding” button.

3.2.6.6 Functional Requirement

System should redirect user to usergroup profile which was added if user presses “Add” button.

3.2.6.7 Functional Requirement

System should show exception message if there is any error.

3.2.7 Add Training Definition

Use Case Name	**UC104_Add Training Definition**
Actor	**Users who have has “ROLE_CAN_ADD_TRAINING_DEFINITION”**
Preconditions	**The url should be known by users**
Trigger	**This use case is initiated when user presses “Add” or “Continue Adding” button**
Basic Flow	**1. System provides “Training Code”, “Name” and “Description” fields to user.** **2. The user fills these fields.** **3. The user presses the “Add” or “Continue Adding” button.** **4. System sends these information to Database and saves.** **5. System redirects user to the training definition profile page or add training definition page.**
Exception Flow	**If training code is not integer or name field is empty then system sends exception**
Post conditions	**Desired training definition is registered and user is redirected to add training definition page or training definition profile**

3.2.7.1 Functional Requirement

System should check user roles. User should have “ROLE_CAN_ADD_TRAINING_DEFINITION” role.

3.2.7.2 Functional Requirement

System should provide “Training Code”, “Name” and “Description” fields.

3.2.7.3 Functional Requirement

System should validate Training Code field should be integer.
Name field should be string.
Description field should has text editor.

3.2.7.4 Functional Requirement

System should save that training definition in database.

3.2.7.5 Functional Requirement

System should redirect user to add training definition page if user presses “Continue Adding” button.

3.2.7.6 Functional Requirement

System should redirect user to training definition profile which was added if user presses “Add” button.

3.2.7.7 Functional Requirement

System should show exception message if there is any error.

3.2.8 Add Term

Use Case Name	**UC105_Add Term**
Actor	**Users who have has “ROLE_CAN_ADD_TERM”**
Preconditions	**The url should be known by users**
Trigger	**This use case is initiated when user presses “Add” button**
Basic Flow	**1. System provides “Name” field and “Active” selectable field to user.** **2. The user fills these fields.** **3. The user presses the “Add” or “Continue Adding” button.** **4. System sends those information to Database and saves.** **5. System redirects user to add term page.**
Exception Flow	**If name field is empty then system sends exception**
Post conditions	**Desired term is registered and user is redirected to add term page**

3.2.8.1 Functional Requirement

System should check user roles. User should have “ROLE_CAN_ADD_TERM” role.

3.2.8.2 Functional Requirement

System should provide “Name” and “active” fields.

3.2.8.3 Functional Requirement

Name field should be string.

Active field should be selectable field.

3.2.8.4 Functional Requirement

System should save that term in database.

If active field is filled, then system updates the term ,which was already active, to inactive. (In database, there must be just one active term)

3.2.8.5 Functional Requirement

System should redirect user to add term page when user presses “Add” button.

3.2.8.6 Functional Requirement

System should show exception message if there is any error.

3.2.9 Add Training

Use Case Name	**UC106_Add Training**
Actor	**Users who have has “ROLE_CAN_ADD_TRAINING”**
Preconditions	**The url should be known by users**
Trigger	**This use case is initiated when user presses “Add” button**
Basic Flow	**1. System provides “Training Definition”, “Trainer”, “Section” and “Term” fields to user.** **2. The user fills these fields.** **3. The user presses the “Add” or “Continue Adding” button.** **4. System sends these information to Database and saves.** **5. System redirects user to training profile page.**
Exception Flow	**If section field is not integer then system sends exception**
Post conditions	**Desired training is registered and user is redirected to training profile**

3.2.9.1 Functional Requirement

System should check user roles. User should have “ROLE_CAN_ADD_TRAINING” role.

3.2.9.2 Functional Requirement

System should provide “Training Definition”, “Trainer”, “Section” “Term” fields.

For “Training Definition” field system should take training definitions from database

For “Trainer” field system should take users from database

For “Term” field system should take terms from database

3.2.9.3 Functional Requirement

System should validate “Section” field. It must be integer.

3.2.9.4 Functional Requirement

System should save that training in database.

3.2.9.5 Functional Requirement

System should redirect user to training profile which was added if user presses “Add” button.

3.2.9.6 Functional Requirement

System should show exception message if there is any error.

3.2.10 Add Task Type

Use Case Name	**UC107_Add Task Type**
Actor	**Users who have has “ROLE_CAN_ADD_TASK_TYPE”**
Preconditions	**The url should be known by users**
Trigger	**This use case is initiated when user presses “Add” button**
Basic Flow	**1. System provides “Name” field to user.** **2. The user fills that field.** **3. The user presses the “Add” button.** **4. System sends these information to Database and saves.** **5. System redirects user to add task type page.**
Exception Flow	**If name field is empty then system sends exception**
Post conditions	**Desired task type is registered and user is redirected to add task type page**

3.2.10.1 Functional Requirement

System should check user roles. User should have “ROLE_CAN_ADD_TASK_TYPE” role.

3.2.10.2 Functional Requirement

User should fill “Name” field.

3.2.10.3 Functional Requirement

System should validate filled field.
Name should not be empty and it will be string

3.2.10.4 Functional Requirement

System should save that task type in database.

3.2.10.5 Functional Requirement

System should redirect user to add task type page, if user presses “Add” button.

3.2.10.6 Functional Requirement

System should show exception message if there is any error.

3.2.11 Add Measurement Set

Use Case Name	**UC108_Add Measurement Set**
Actor	**Users who have has “ROLE_CAN_ADD_MEASUREMENT_SET”**
Preconditions	**The url should be known by users**
Trigger	**This use case is initiated when user presses “Add” or “Continue Adding” button**
Basic Flow	**1. System provides “Measurement Set Name” and “Parent Measurement Set” fields to user.** **2. The user fills that field.** **3. The user presses the “Add” or “Continue Adding” button.** **4. System sends these information to Database and saves.** **5. System redirects user to add measurement set page or measurement set profile which was added.**
Exception Flow	**If measurement Set Name field is empty then system sends exception**
Post conditions	**Desired measurement set is registered and user is redirected to add measurement set page or measurement set profile which was added**

3.2.11.1 Functional Requirement

System should check user roles. User should have “ROLE_CAN_ADD_MEASUREMENT_SET” role.

3.2.11.2 Functional Requirement

System should provide “Name” and “Parent Measurement Set” fields

For “Parent Measurement Set” field, system should get measurement sets from database

3.2.11.3 Functional Requirement

System should validate filled field. Name should not be empty and it will be string

3.2.11.4 Functional Requirement

System should save that measurement set in database.

3.2.11.5 Functional Requirement

System should redirect user to add measurement set profile page which was added, if user presses “Add” button.

3.2.11.6 Functional Requirement

System should redirect user to add measurement set page, if user presses “Continue Adding” button.

3.2.11.7 Functional Requirement

System should show exception message if there is any error.

3.2.12 Add Measurement Criteria

Use Case Name	**UC109_Add Measurement Criteria**
Actor	**Users who have has “ROLE_CAN_ADD_MEASUREMENT_CRITERIA”**
Preconditions	**The url should be known by users**
Trigger	**This use case is initiated when user presses “Add” or “Continue Adding” button**
Basic Flow	**1. System provides “Description”, “Measurement Set” and “Parent Measurement Criteria” fields to user.** **2. The user fills that field.** **3. The user presses the “Add” or “Continue Adding” button.** **4. System sends these information to Database and saves.** **5. System redirects user to add measurement criteria page or measurement criteria profile which was added.**

Exception Flow	
Post conditions	**Desired measurement criteria is registered and user is redirected to add measurement criteria page or measurement criteria profile which was added**

3.2.12.1 Functional Requirement

System should check user roles. User should have "ROLE_CAN_ADD_MEASUREMENT_CRITERIA" role.

3.2.12.2 Functional Requirement

System should provide "Description", "Measurement Set" and "Parent Measurement Criteria" fields

"Description" field must has text editor.

For "Measurement Set" field, system should get measurement sets from database

For "Parent Measurement Criteria" field, system should get measurement criteria from database

3.2.12.3 Functional Requirement

System should validate filled field.

3.2.12.4 Functional Requirement

System should save those measurement criteria in database.

3.2.12.5 Functional Requirement

System should redirect user to add measurement criteria profile page which was added, if user presses "Add" button.

3.2.12.6 Functional Requirement

System should redirect user to add measurement criteria page, if user presses "Continue Adding" button.

3.2.12.7 Functional Requirement

System should show exception message if there is any error.

3.2.13 List Users

Use Case Name	**UC201_List Users**
Actor	**Users who have is logged in**

Preconditions	**The url should be known by users**
Trigger	**This use case is initiated when user enters user list page**
Basic Flow	**1. System gets registered users list from database** **2. System shows registered users in table which column names are “Username”, “Email”, “ Email 2”**
Exception Flow	
Post conditions	**User reach user list**

3.2.13.1 Functional Requirement

System should check user is logged in or not.

3.2.13.2 Functional Requirement

System gets user list from database.

3.2.13.3 Functional Requirement

System shows users in table which has columns namely “Username”, “E-Mail” and “E-Mail2”

3.2.14 List Usergroups

Use Case Name	**UC202_List Usergroups**
Actor	**Users who have is logged in**
Preconditions	**The url should be known by users**
Trigger	**This use case is initiated when user enters usergroup list page**
Basic Flow	**1. System gets registered usergroups list from database** **2. System shows registered usergroups in table which column name is “Name”**
Exception Flow	
Post conditions	**User reach usergroup list**

3.2.14.1 Functional Requirement

System should check user is logged in or not.

3.2.14.2 Functional Requirement

System gets usergroup list from database.

3.2.14.3 Functional Requirement

System shows usergroups in table which has column namely “Name”

3.2.15 List Roles

Use Case Name	**UC203_List Roles**
Actor	**Users who have is logged in**
Preconditions	**The url should be known by users**
Trigger	**This use case is initiated when user enters role list page**
Basic Flow	**1. System gets registered role list from database** **2. System shows registered roles in table which column names are “Role” and “Description”**
Exception Flow	
Post conditions	**User reach role list**

3.2.15.1 Functional Requirement

System should check user is logged in or not.

3.2.15.2 Functional Requirement

System gets role list from database.

3.2.15.3 Functional Requirement

System shows roles in table which has columns namely “role and “Description”

3.2.16 List Departments

Use Case Name	**UC204_List Departments**
Actor	**Users who have is logged in**
Preconditions	**The url should be known by users**
Trigger	**This use case is initiated when user enters department list page**
Basic Flow	**1. System gets registered department list from database** **2. System shows registered departments names** **3. System should provide search property.**
Exception Flow	
Post conditions	**User reach department list**

3.2.16.1 Functional Requirement

System should check user is logged in or not.

3.2.16.2 Functional Requirement

System gets department list from database.

3.2.16.3 Functional Requirement

System shows departments names.

3.2.16.4 Functional Requirement

System should provide search tool. It should filter departments according to given string.

3.2.17 List Training Definitions

Use Case Name	**UC205_List Training Definitions**
Actor	**Users who have is logged in**
Preconditions	**The url should be known by users**
Trigger	**This use case is initiated when user enters training definition list page**

Basic Flow	**1. System gets registered training definition list from database** **2. System shows registered training definitions in table which has columns namely “Training Code” and “Name”**
Exception Flow	
Post conditions	**User reach training definition list**

3.2.17.1 Functional Requirement

System should check user is logged in or not.

3.2.17.2 Functional Requirement

System gets training definition list from database.

3.2.17.3 Functional Requirement

System shows training definition in table which has columns namely “Training Code” and “Name”.

3.2.18 List Trainings

Use Case Name	**UC206_List Trainings**
Actor	**Users who have is logged in**
Preconditions	**The url should be known by users**
Trigger	**This use case is initiated when user enters trainings list page**
Basic Flow	**1. System gets registered training list from database** **2. System shows registered trainings in table which has columns namely “Training Definition”, “Trainer”, “Section”, “Task Set” and “Term”**
Exception Flow	
Post conditions	**User reach training list**

3.2.18.1 Functional Requirement

System should check user is logged in or not.

3.2.18.2 Functional Requirement

System gets training list from database.

3.2.18.3 Functional Requirement

System shows registered trainings in table which has columns namely “Training Definition”, “Trainer”, “Section”, “Task Set” and “Term.

3.2.19 List Task Types

Use Case Name	**UC207_List Task Types**
Actor	**Users who have is logged in**
Preconditions	**The url should be known by users**
Trigger	**This use case is initiated when user enters task type list page**
Basic Flow	**1. System gets registered task type list from database** **2. System shows registered task types in table which has column namely “Name”**
Exception Flow	
Post conditions	**User reach task type list**

3.2.19.1 Functional Requirement

System should check user is logged in or not.

3.2.19.2 Functional Requirement

System gets task type list from database.

3.2.19.3 Functional Requirement

System shows registered task types in table which has column namely “Name”

3.2.20 List Terms

Use Case Name	**UC208_List Terms**
Actor	**Users who have is logged in**
Preconditions	**The url should be known by users**
Trigger	**This use case is initiated when user enters term list page**
Basic Flow	**1. System gets registered term list from database** **2. System shows registered terms in table which has column namely "Name". Registered terms shall be split according to active and inactive terms**
Exception Flow	
Post conditions	**User reach term list**

3.2.20.1 Functional Requirement

System should check user is logged in or not.

3.2.20.2 Functional Requirement

System gets term list from database.

3.2.20.3 Functional Requirement

System shall be split terms according to active or inactive and then shows them in table which has column namely "Name".

3.2.21 List Measurement Sets

Use Case Name	**UC209_List Measurement Sets**
Actor	**Users who have is logged in**
Preconditions	**The url should be known by users**
Trigger	**This use case is initiated when user enters measurement set list page**
Basic Flow	**1. System gets registered measurement set list from database**

	2. System shows registered measurement set in table which has column namely "Name" and "Parent Measurement Set".
Exception Flow	
Post conditions	**User reach measurement set list**

3.2.21.1 Functional Requirement

System should check user is logged in or not.

3.2.21.2 Functional Requirement

System gets measurement set list from database.

3.2.21.3 Functional Requirement

System shows measurement sets in table which has column namely "Name" and "Parent Measurement Set".

3.2.22 List Measurement Criteria

Use Case Name	**UC210_List Measurement Criteria**
Actor	**Users who have is logged in**
Preconditions	**The url should be known by users**
Trigger	**This use case is initiated when user enters measurement criteria list page**
Basic Flow	**1. System gets registered measurement criteria list from database** **2. System shows registered measurement criteria in table which has column namely "Id", "Parent Measurement Criteria" and "Parent Measurement Set".**
Exception Flow	
Post conditions	**User reach measurement criteria list**

3.2.22.1 Functional Requirement

System should check user is logged in or not.

3.2.22.2 Functional Requirement

System gets measurement criteria list from database.

3.2.22.3 Functional Requirement

System shows measurement criteria in table which has column namely “Id”, “Parent Measurement Criteria” and “Parent Measurement Set”.

3.2.23 Assign User to Department

Use Case Name	**UC301_Assign User to Department**
Actor	**Users who have “ROLE_ADMIN” role**
Preconditions	**The url should be known by users**
Trigger	**This use case is initiated when user enters “Add Member” page from department profile**
Basic Flow	**1. System gets registered users list from database** **2. System splits them to already registered and not registered to the desired department** **3. System provide “Assign”, “Remove” buttons** **4. The user arrange users by helping “Assign” and “Remove button”** **5. The user press “Apply” button to save to database**
Exception Flow	
Post conditions	**The new users of department will be arranged and saved to database**

3.2.23.1 Functional Requirement

System should check user have “ROLE_ADMIN” role

3.2.23.2 Functional Requirement

System gets user list from database and split them to registered and not registered to desired department

3.2.23.3 Functional Requirement

System provides “Assign”, “Remove” and “Apply” buttons.
“Assign” button is provided for assign users to desired department.
“Remove” button is provided for remove users from desired department.
“Apply” button is provided for save to database

3.2.23.4 Functional Requirement

System shall save the final picture of department.
Related database table: DepartmentUser

3.2.24 Assign User to Usergroup

Use Case Name	**UC302_Assign User To Usergroup**
Actor	**Users who have “ROLE_ADMIN” role**
Preconditions	**The url should be known by users**
Trigger	**This use case is initiated when user enters “Add Member” page from usergorup profile**
Basic Flow	**1. System gets registered users list from database** **2. System splits them to already registered and not registered to the desired usergorup** **3. System provide “Assign”, “Remove” buttons** **4. The user arrange users by helping “Assign” and “Remove button”** **5. The user press “Apply” button to save to database**
Exception Flow	
Post conditions	**The new users of usergorup will be arranged and saved to database**

3.2.24.1 Functional Requirement

System should check user have “ROLE_ADMIN” role

3.2.24.2 Functional Requirement

System gets user list from database and split them to registered and not registered to desired usergroup

3.2.24.3 Functional Requirement

System provides “Assign”, “Remove” and “Apply” buttons.
“Assign” button is provided for assign users to desired usergroup.

“Remove” button is provided for remove users from desired usergroup.
“Apply” button is provided for save to database

3.2.24.4 Functional Requirement

System shall save the final picture of usergroup.
Related database table: UsergroupUser

3.2.25 Assign User to Training

Use Case Name	**UC303_Assign User To Training**
Actor	**Users who have “ROLE_ADMIN” role**
Preconditions	**The url should be known by users**
Trigger	**This use case is initiated when user enters “Edit Subscriber” page from Training profile**
Basic Flow	**1. System gets registered users list from database** **2. System splits them to Subscribe Type and already registered and not registered to the desired Training.** **3. System provide “Assign”, “Remove” buttons** **4. The user arrange users by helping “Assign” and “Remove button”** **5. The user press “Apply” button to save to database**
Exception Flow	
Post conditions	**The new users of Training will be arranged and saved to database**

3.2.25.1 Functional Requirement

System should check user have “ROLE_ADMIN” role

3.2.25.2 Functional Requirement

System gets user list from database and split them to Subscriber Type and registered and not registered to desired Training
Subscribe Types: Trainers, Moderators, Trainee, Spectator

3.2.25.3 Functional Requirement

System provides “Assign”, “Remove” and “Apply” buttons.
“Assign” button is provided for assign users to desired Training.
“Remove” button is provided for remove users from desired Training.
“Apply” button is provided for save to database

3.2.25.4 Functional Requirement

System shall save the final picture of Training.
Related database table: Training, Subscribers
If Trainers of the training is changed then the field of trainer from Training table will be updated.
If the other type of subscribers is edited then the Subscriber table will be updated.

3.2.26 Assign Grade To Subscriber

Use Case Name	**UC304_Assign Grade To Subscriber**
Actor	**Users who have “ROLE_ADMIN” role or moderator of training which is subscriber is registered.**
Preconditions	**The url should be known by users**
Trigger	**This use case is initiated when user enters “Add Grade” page from Task profile**
Basic Flow	**1. System gets subscribers ,which are registered training that related to the task, list from database** **2. System provides input boxes for each subscriber for each task.** **3. The user enters grades in that boxes** **4. The user press “Apply” button to save to database**
Exception Flow	**Input boxes must be filled with integer, otherwise system will sends exception message**
Post conditions	**The grades of given subscriber are saved to database in “Grade” table**

3.2.26.1 Functional Requirement

System should check user have ROLE_ADMIN" role or moderator of training which is subscriber is registered.

3.2.26.2 Functional Requirement

System gets subscribers, which are registered training that related to the task, list from database.

3.2.26.3 Functional Requirement

For each subscriber, system provides input boxes and system provide "Apply" button.
"Apply" button is provided for save to database
In that boxes must be filled with integer values.

3.2.26.4 Functional Requirement

System shall save the grades to database in "Grade" table

3.2.26.5 Functional Requirement

System should provide exception message.

3.2.27 Assign Task To Training

Use Case Name	**UC305_Assign Task to Training**
Actor	**Users who have "ROLE_ADMIN" role or moderator of training.**
Preconditions	**The url should be known by users**
Trigger	**This use case is initiated when user enters "Add Task" page from training profile**
Basic Flow	**1. System provides form that includes "Name", "Weight (%)" and Task Type fields.** **2. The user fills those fields.** **3. The user press "Add" button to save to database**
Exception Flow	**If the "Weight (%)" field is not filled with integer, then system sends exception message**
Post conditions	**The task will be saved to database and user is redirected to related training.**

3.2.27.1 Functional Requirement

System provides form that includes “Name”, “Weight (%)” and Task Type fields

3.2.27.2 Functional Requirement

System provides “Add” button.
“Add” button is provided for save to database

3.2.27.3 Functional Requirement

System shall save the grades to database in “Task” table

3.2.27.4 Functional Requirement

System should validate “Weight (%)” field. That field must be filled with integer.

3.2.27.5 Functional Requirement

System should redirect user to related training profile page.

3.2.27.6 Functional Requirement

System should provide exception message.

3.2.28 Assign Taskunit To Task

Use Case Name	**UC306_Assign Taskunit To Task**
Actor	**Users who have “ROLE_ADMIN” role or moderator of training.**
Preconditions	**The url should be known by users**
Trigger	**This use case is initiated when user enters “Add TaskUnit” page from task profile**
Basic Flow	**1. System provides form that includes “Name” and “point”** **2. The user fills those fields.** **3. The user press “Add” button to save to database**
Exception Flow	**If the “point” field is not filled with integer, then system sends exception message**
Post conditions	**The taskunit will be saved to database and user is redirected to related task profile page.**

3.2.28.1 Functional Requirement

System provides form that includes “Name” and “Point”

3.2.28.2 Functional Requirement

System provides “Add” button.
“Add” button is provided for save to database

3.2.28.3 Functional Requirement

System shall save the taskunit to database in “TaskUnit” table

3.2.28.4 Functional Requirement

System should validate “Point” field. That field must be filled with integer.

3.2.28.5 Functional Requirement

System should redirect user to related task profile page.

3.2.28.6 Functional Requirement

System should provide exception message.

3.2.29 Assign Role To Usergroup

Use Case Name	**UC307_Assign Role To Usergroup**
Actor	**Users who have “ROLE_ADMIN” role**
Preconditions	**The url should be known by users**
Trigger	**This use case is initiated when user enters “Add Role” page from usergorup profile**
Basic Flow	**1. System provides form that includes all roles and provide select box for each role.** **2. System provides “Save” button** **3. The user selects roles** **4. The user press “Save” button to save to database**
Exception Flow	
Post conditions	**The roles will be saved to database in UsergroupRole table and user is redirected to related usergroup profile page**

3.2.29.1 Functional Requirement

System gets all roles from database.

3.2.29.2 Functional Requirement

System provides form that includes all roles and provide select box for each role.
System provides “Save” button.
“Save” button is provided for save to database

3.2.29.3 Functional Requirement

System shall save the roles to database in “UsergroupRole” table.

3.2.29.4 Functional Requirement

System should redirect user to related usergroup profile page.

3.2.30 Assign Role To User

Use Case Name	**UC308_Assign Role To User**
Actor	**Users who have “ROLE_ADMIN” role**
Preconditions	**The url should be known by users**
Trigger	**This use case is initiated when user enters “Add Role” page from user profile**
Basic Flow	**1. System provides form that includes all roles and provide select box for each role.** **2. System provides “Save” button** **3. The user selects roles** **4. The user press “Save” button to save to database**
Exception Flow	

Post conditions	**The roles will be saved to database in UserRole table and user is redirected to related user profile page**

3.2.30.1 Functional Requirement

System gets all roles from database.

3.2.30.2 Functional Requirement

System provides form that includes all roles and provide select box for each role.
System provides “Save” button.
“Save” button is provided for save to database

3.2.30.3 Functional Requirement

System shall save the roles to database in “UserRole” table.

3.2.30.4 Functional Requirement

System should redirect user to related user profile page.

3.2.31 Show User Statistics

Use Case Name	**UC401_Show User Statistics**
Actor	**Users who have “ROLE_ADMIN” role**
Preconditions	**The url should be known by users**
Trigger	**This use case is initiated when user enters show user statistics page from user profile page**
Basic Flow	**1. System gets user subscribed trainings** **2. System gets grades which are related to user subscribed trainings** **3. System make analysis to categorize user statistics under given Measurement Criterias** **4. System make analysis to categorize user statistics semester by semester** **5. System shows statistics charts such “Radar”, “Line”, “Bar” etc.**

Exception Flow	
Post conditions	**The user statistics are present according to year by year and given measurement criteria**

3.2.31.1 Functional Requirement

System should check user have “ROLE_ADMIN” role

3.2.31.2 Functional Requirement

System gets user subscribed trainings

3.2.31.3 Functional Requirement

System gets user grades to make analyse

3.2.31.4 Functional Requirement

System makes analyses get results

3.2.31.5 Functional Requirement

System presents results in charts such “Radar”, “Line”, “Bar”.

3.2.32 Show Term Statistics

Use Case Name	**UC402_Show Term Statistics**
Actor	**Users who have “ROLE_ADMIN” role**
Preconditions	**The url should be known by users**
Trigger	**This use case is initiated when user enters show term statistics page from term profile page**
Basic Flow	**1. System gets trainings which are registered to the term** **2. System gets grades which are related to the term**

	3. System make analysis to categorize user statistics under given Measurement Criteria **4. System make analysis to categorize term statistics under given measurement criteria** **5. System shows statistics charts such “Radar”, “Line”, “Bar” etc.**
Exception Flow	
Post conditions	**The term statistics are present according to given measurement criteria**

3.2.32.1 Functional Requirement

System should check user have “ROLE_ADMIN” role

3.2.32.2 Functional Requirement

System gets trainings which are registered to the term

3.2.32.3 Functional Requirement

System gets trainings, which are registered that term, grades to make analyse

3.2.32.4 Functional Requirement

System makes analyses get results

3.2.32.5 Functional Requirement

System presents results in charts such “Radar”, “Line”, “Bar”.

3.2.33 Show Training Statistics

Use Case Name	**UC403_Show Training Statistics**
Actor	**Users who have “ROLE_ADMIN” role**
Preconditions	**The url should be known by users**
Trigger	**This use case is initiated when user enters show training statistics page from training profile page**
Basic Flow	**1. System gets grades which are related to the training**

	2. System make analysis to categorize user statistics under given Measurement Criteria and section **3. System make analysis to categorize term statistics under given measurement criteria, section** **4. System shows statistics charts such “Radar”, “Line”, “Bar” etc.**
Exception Flow	
Post conditions	**The training statistics are present according to given measurement criteria and section**

3.2.33.1 Functional Requirement

System should check user have “ROLE_ADMIN” role

3.2.33.2 Functional Requirement

System gets grades which are related to the training to make analyse

3.2.33.3 Functional Requirement

System makes analyses get results and categorize these result under section by section and measurement criteria

3.2.33.4 Functional Requirement

System presents results in charts such “Radar”, “Line”, “Bar”.

3.2.34 Show MeasurementSet Statistics

Use Case Name	**UC404_Show MeasurementSet statistics**
Actor	**Users who have “ROLE_ADMIN” role**
Preconditions	**The url should be known by users**
Trigger	**This use case is initiated when user enters show MeasurementSet statistics page from MeasurementSet profile page**

Basic Flow	**1. System gets related measurement criterias with that measurementset** **2. System gets grades which are related to those measurement criteria** **3. System make analysis to categorize user statistics year by year** **4. System shows statistics charts such “Radar”, “Line”, “Bar” etc.**
Exception Flow	
Post conditions	**The MeasurementSet statistics are present according year by year**

3.2.34.1 Functional Requirement

System should check user have “ROLE_ADMIN” role

3.2.34.2 Functional Requirement

System gets measurement criteria which are related to the measurementset

3.2.34.3 Functional Requirement

System gets grades which are related to those measurement criteria

3.2.34.4 Functional Requirement

System makes analyses get results and categorize them to year by year

3.2.34.5 Functional Requirement

System presents results in charts such “Radar”, “Line”, “Bar”.

3.2.35 Show TaskType Statistics

Use Case Name	**UC405_Show TaskType Statistics**
Actor	**Users who have “ROLE_ADMIN” role**
Preconditions	**The url should be known by users**

Trigger	**This use case is initiated when user enters show tasktype statistics page from tasktype profile page**
Basic Flow	**1. System gets related tasks with that tasktype** **2. System gets grades which are related to those task** **3. System make analysis to categorize user statistics year by year** **4. System shows statistics charts such “Radar”, “Line”, “Bar” etc.**
Exception Flow	
Post conditions	**The tasktype statistics are present according year by year**

3.2.35.1 Functional Requirement

System should check user have “ROLE_ADMIN” role

3.2.35.2 Functional Requirement

System gets tasks which are related to the tasktype

3.2.35.3 Functional Requirement

System gets grades which are related to those task

3.2.35.4 Functional Requirement

System makes analyses get results and categorize them to year by year

3.2.35.5 Functional Requirement

System presents results in charts such “Radar”, “Line”, “Bar”.

3.2.36 Show Trainer Statistics

Use Case Name	**UC406_Show Trainer Statistics**
Actor	**Users who have “ROLE_ADMIN” role**

Preconditions	**The url should be known by users**
Trigger	**This use case is initiated when user enters show trainer statistics page from user profile page**
Basic Flow	**1. System gets trainings which are related with that trainer.** **2. System gets subscriber list of that trainings** **3. System gets grades of these subscribers** **4. System make analyse these grades and categorize these grades training by training, year by year, measurement set by measurement set** **5. System present these result by helping charts such "Bar", "Line", "Radar"**
Exception Flow	
Post conditions	**The trainer statistics are present according training by training, year by year, measurement set by measurement set**

3.2.36.1 Functional Requirement

System should check user have "ROLE_ADMIN" role

3.2.36.2 Functional Requirement

System gets trainings which are related with that trainer.

3.2.36.3 Functional Requirement

System gets subscriber list of that trainings

3.2.36.4 Functional Requirement

System gets grades of these subscribers

3.2.36.5 Functional Requirement

System make analyse these grades and categorize these grades training by training, year by year, measurement set by measurement set

3.2.36.5 Functional Requirement

System present these result by helping charts such "Bar", "Line", "Radar"

3.2.37 Update User

Use Case Name	**UC501_UpdateUser**

Actor	**Users who have has “ROLE_CAN_UPDATE_USER” role**
Preconditions	**The url should be known by users**
Trigger	**This use case is initiated when user presses “Save” button**
Basic Flow	**1. System provides “Username”, “Password”, “Name”, “Surname”, “E-mail”, “Optional- Email” and “Picture” fields to user.** **2. The user fills these fields.** **3. The user press the “Save” button.** **4. System sends those information to Database and updates the user.** **5. System redirects user to the user profile page.**
Exception Flow	**If the username already registered then system sends exception** **If “E-mail”, “name”, “surname”, “username” or “password” one of these fields is empty then system sends exception**
Post conditions	**Desired user’s information are updated and the user redirected to user profile**

3.2.37.1 Functional Requirement

System should check user roles. User should have “ROLE_CAN_UPDATE_USER” role.

3.2.37.2 Functional Requirement

User should fill “Username”, “Password”, “Name”, “Surname”, “E-mail”, “Optional- Email” and “Picture” fields.

3.2.37.3 Functional Requirement

System should validate filled fields.
Username should be unique in Database.
Password should contain at least 8 characters.
Name and surname field must not be empty.
Email regex format : _*@_* Email field must not be empty.
Optional E-mail format : _*@_* It can be null.

3.2.37.4 Functional Requirement

System should update that user into database.

3.2.37.5 Functional Requirement

System should redirect user to user page which was added if user presses “Save” button.

3.2.37.6 Functional Requirement

System should show exception message if there is any error.

3.2.38 Update Department

Use Case Name	**UC502_UpdateDepartment**
Actor	**Users who have has "ROLE_CAN_UPDATE_DEPARTMENT" role**
Preconditions	**The url should be known by users**
Trigger	**This use case is initiated when user presses "Save" button**
Basic Flow	**1. System provides "Department Name" and "Parent Department" fields to user.** **2. The user fills these fields.** **3. The user presses the "Save" button.** **4. System sends those information to Database and updates the department.** **5. System redirects user to the department profile page.**
Exception Flow	**If department name is empty then system sends exception**
Post conditions	**Desired department is updated and user is redirected to department profile**

3.2.38.1 Functional Requirement

System should check user roles. User should have "ROLE_CAN_UPDATE_DEPARTMENT" role.

3.2.38.2 Functional Requirement

System should get department list from database for "Parent Department" field.

3.2.38.3 Functional Requirement

System should validate department name.
It must not be empty
It will be string.

3.2.38.4 Functional Requirement

System should update that department into database.

3.2.38.5 Functional Requirement

System should redirect user to department profile which was added if user presses "Save" button.

3.2.38.6 Functional Requirement

System should show exception message if there is any error

3.2.39 Update Usergroup

Use Case Name	**UC503_UpdateUsergroup**
Actor	**Users who have has “ROLE_CAN_UPDATE_USERGROUP”**
Preconditions	**The url should be known by users**
Trigger	**This use case is initiated when user presses “Save” button**
Basic Flow	**1. System provides “Group Name” field and role list to user.** **2. The user fills “Group Name” field and selects roles from role list** **3. The user presses the “Save” button.** **4. System sends these information to Database and updates the usergroup.** **5. System redirects user to the usergroup profile page.**
Exception Flow	**If usergroup name is empty then system sends exception message**
Post conditions	**Desired usergorup is updated and user is redirected to usergroup profile**

3.2.39.1 Functional Requirement

System should check user roles. User should have “ROLE_CAN_ADD_USERGROUP” role.

3.2.39.2 Functional Requirement

System should get role list from database.

3.2.39.3 Functional Requirement

System should validate usergroup name.
It must not be empty. It will be string.

3.2.39.4 Functional Requirement

System should update that usergroup into database.

3.2.39.5 Functional Requirement

System should redirect user to usergroup profile which was updated if user presses “Save” button.

3.2.39.6 Functional Requirement

System should show exception message if there is any error.

3.2.40 Update Training

Use Case Name	**UC504_UpdateTraining**

Actor	**Users who have has “ROLE_CAN_UPDATE_TRAINING”**
Preconditions	**The url should be known by users**
Trigger	**This use case is initiated when user presses “Save” button**
Basic Flow	**1. System provides “Training Definition”, “Trainer”, “Section” and “Term” fields to user.** **2. The user fills these fields.** **3. The user presses the “Save” button.** **4. System sends these information to Database and updates.** **5. System redirects user to training profile page.**
Exception Flow	**If section field is not integer then system sends exception**
Post conditions	**Desired training is updated and user is redirected to training profile**

3.2.40.1 Functional Requirement

System should check user roles. User should have “ROLE_CAN_UPDATE_TRAINING” role.

3.2.40.2 Functional Requirement

System should provide “Training Definition”, “Trainer”, “Section” “Term” fields.
For “Training Definition” field system should take training definitions from database
For “Trainer” field system should take users from database
For “Term” field system should take terms from database

3.2.40.3 Functional Requirement

System should validate “Section” field. It must be integer.

3.2.40.4 Functional Requirement

System should save that training in database.

3.2.40.5 Functional Requirement

System should redirect user to training profile which was updated if user presses “Save” button.

3.2.40.6 Functional Requirement

System should show exception message if there is any error.

3.2.41 Update Training Definition

Use Case Name	**UC505_UpdateTrainingDefination**
Actor	**Users who have has “ROLE_CAN_UPDATE_TRAINING_DEFINITION”**
Preconditions	**The url should be known by users**
Trigger	**This use case is initiated when user presses “Save” button**
Basic Flow	**1. System provides “Training Code”, “Name” and “Description” fields to user.** **2. The user fills these fields.** **3. The user presses the “Save” button.** **4. System sends these information to Database and updates.** **5. System redirects user to the training definition profile page.**
Exception Flow	**If training code is not integer or name field is empty then system sends exception**
Post conditions	**Desired training definition is updated and user is redirected to training definition profile**

3.2.41.1 Functional Requirement

System should check user roles. User should have “ROLE_CAN_UPDATE_TRAINING_DEFINITION” role.

3.2.41.2 Functional Requirement

System should provide “Training Code”, “Name” and “Description” fields.

3.2.41.3 Functional Requirement

System should validate Training Code field should be integer.
Name field should be string.
Description field should has text editor.

3.2.41.4 Functional Requirement

System should update that training definition into database.

3.2.41.5 Functional Requirement

System should redirect user to training definition profile which was updated if user presses “Update” button.

3.2.41.6 Functional Requirement

System should show exception message if there is any error.

3.2.42 Update Term

Use Case Name	**UC506_UpdateTerm**

Actor	**Users who have has “ROLE_CAN_UPDATE_TERM”**
Preconditions	**The url should be known by users**
Trigger	**This use case is initiated when user presses “Save” button**
Basic Flow	**1. System provides “Name” field and “Active” selectable field to user.** **2. The user fills these fields.** **3. The user presses the “Save” button.** **4. System sends those information to Database and updates.** **5. System redirects user to term list page.**
Exception Flow	**If name field is empty then system sends exception**
Post conditions	**Desired term is updated and user is redirected to term list page**

3.2.42.1 Functional Requirement

System should check user roles. User should have “ROLE_CAN_UPDATE_TERM” role.

3.2.42.2 Functional Requirement

System should provide “Name” and “active” fields.

3.2.42.3 Functional Requirement

Name field should be string.

Active field should be selectable field.

3.2.42.4 Functional Requirement

System should save that term in database.

If active field is filled, then system updates the term, which was already active, to inactive. (In database, there must be just one active term)

3.2.42.5 Functional Requirement

System should redirect user to term list page when user presses “Save” button.

3.2.42.6 Functional Requirement

System should show exception message if there is any error.

3.2.43 Update Task Type

Use Case Name	**UC507_UpdateTaskType**

Actor	**Users who have has “ROLE_CAN_UPDATE_TASK_TYPE”**
Preconditions	**The url should be known by users**
Trigger	**This use case is initiated when user presses “Save” button**
Basic Flow	**1. System provides “Name” field to user.** **2. The user fills that field.** **3. The user presses the “Save” button.** **4. System sends these information to Database and updates.** **5. System redirects user to task type list page.**
Exception Flow	**If name field is empty then system sends exception**
Post conditions	**Desired task type is updated and user is redirected to task type list page**

3.2.43.1 Functional Requirement

System should check user roles. User should have “ROLE_CAN_UPDATE_TASK_TYPE” role.

3.2.43.2 Functional Requirement

User should fill “Name” field.

3.2.43.3 Functional Requirement

System should validate filled field.
Name should not be empty and it will be string

3.2.43.4 Functional Requirement

System should update that task type into database.

3.2.43.5 Functional Requirement

System should redirect user to task type list page, if user presses “save” button.

3.2.43.6 Functional Requirement

System should show exception message if there is any error.

3.2.44 Update Task

Use Case Name	**UC508_UpdateTask**
Actor	**Users who have has “ROLE_CAN_UPDATE_TASK”**
Preconditions	**The url should be known by users**
Trigger	**This use case is initiated when user presses “Save” button**
Basic Flow	**1. System provides “Name”, “Weight” and “TaskType” fields to user.** **2. The user fills that field.** **3. The user presses the “Save” button.** **4. System sends these information to Database and updates.** **5. System redirects user to task list page.**
Exception Flow	**If “Name” field is empty or “Weight” field is not integer then system sends exception**
Post conditions	**Desired task is updated and user is redirected task list page.**

3.2.44.1 Functional Requirement

System should check user roles. User should have “ROLE_CAN_UPDATE_TASK” role.

3.2.44.2 Functional Requirement

System provides “Name”, “Weight” and “TaskType” fields to user

3.2.44.3 Functional Requirement

System should validate filled field.

Name should not be empty and it will be string

Weight field must be filled with integer value

3.2.44.4 Functional Requirement

System should updates that task into database.

3.2.44.5 Functional Requirement

System should redirect user to task list page, if user presses “Save” button.

3.2.44.6 Functional Requirement

System should show exception message if there is any error.

3.2.45 Update Measurement Criteria

Use Case Name	**UC509_UpdateMeasurementCriteria**
Actor	**Users who have has "ROLE_CAN_UPDATE_MEASUREMENT_CRITERIA"**
Preconditions	**The url should be known by users**
Trigger	**This use case is initiated when user presses "Save" button**
Basic Flow	**1. System provides "Description", "Measurement Set" and "Parent Measurement Criteria" fields to user.** **2. The user fills that field.** **3. The user presses the "Save" button.** **4. System sends these information to Database and updates.** **5. System redirects user to measurement criteria profile which was updated.**
Exception Flow	
Post conditions	**Desired measurement criteria is registered and user is redirected to add measurement criteria page or measurement criteria profile which was added**

3.2.45.1 Functional Requirement

System should check user roles. User should have "ROLE_CAN_UPDATE_MEASUREMENT_CRITERIA" role.

3.2.45.2 Functional Requirement

System should provide "Description", "Measurement Set" and "Parent Measurement Criteria" fields. "Description" field must has text editor.

For "Measurement Set" field, system should get measurement sets from database

For "Parent Measurement Criteria" field, system should get measurement criteria from database

3.2.45.3 Functional Requirement

System should validate filled field.

3.2.45.4 Functional Requirement

System should update that measurement criteria into database.

3.2.45.5 Functional Requirement

System should redirect user to measurement criteria profile page which was updated, if user presses "Save" button.

3.2.45.6 Functional Requirement

System should show exception message if there is any error.

3.2.46 Add Measurement Set

Use Case Name	**UC510_UpdateMeasurementSet**
Actor	**Users who have has “ROLE_CAN_UPDATE_MEASUREMENT_SET”**
Preconditions	**The url should be known by users**
Trigger	**This use case is initiated when user presses “Save” button**
Basic Flow	**1. System provides “Measurement Set Name” and “Parent Measurement Set” fields to user.** **2. The user fills that field.** **3. The user presses the “Save” button.** **4. System sends these information to Database and updates.** **5. System redirects user to measurement set profile which was updated.**
Exception Flow	**If measurement Set Name field is empty then system sends exception**
Post conditions	**Desired measurement set is updated and user is redirected to measurement set profile which was updated**

3.2.46.1 Functional Requirement

System should check user roles. User should have ROLE_CAN_UPDATE_MEASUREMENT_SET” role.

3.2.46.2 Functional Requirement

System should provide “Name” and “Parent Measurement Set” fields

For “Parent Measurement Set” field, system should get measurement sets from database

3.2.46.3 Functional Requirement

System should validate filled field.
Name should not be empty and it will be string

3.2.46.4 Functional Requirement

System should update that measurement set into database.

3.2.46.5 Functional Requirement

System should redirect user to measurement set profile page which was updated, if user presses “Save” button.

3.2.46.6 Functional Requirement

System should redirect user to add measurement set page, if user presses “Continue Adding” button.

3.2.46.7 Functional Requirement

System should show exception message if there is any error.

3.2.47 Remove User

Use Case Name	UC601_RemoveUser
Actor	Users who have has “ROLE_CAN_REMOVE_USER” role
Preconditions	The url should be known by users
Trigger	This use case is initiated when user presses “Remove” button
Basic Flow	1. System provides “Remove” button. 2. System redirects user to the user list page
Exception Flow	
Post conditions	The desired user is removed and user is redirected to user list page

3.2.47.1 Functional Requirement

System should check user roles. User should have “ROLE_CAN_REMOVE_USER” role.

3.2.47.2 Functional Requirement

System should remove user from database.

3.2.47.3 Functional Requirement

System should redirect user to user list page

3.2.48 Remove Department

Use Case Name	UC602_RemoveDepartment
Actor	Users who have has “ROLE_CAN_REMOVE_DEPARTMENT” role
Preconditions	The url should be known by users
Trigger	This use case is initiated when user presses “Remove” button
Basic Flow	1. System provides “Remove” button. 2. System redirects user to the department list page

Exception Flow	
Post conditions	**The desired department is removed and user is redirected to department list page**

3.2.48.1 Functional Requirement

System should check user roles. User should have “ROLE_CAN_REMOVE_DEPARTMENT” role.

3.2.48.2 Functional Requirement

System should remove department from database.

3.2.48.3 Functional Requirement

System should redirect user to department list page

3.2.49 Remove Role

Use Case Name	**UC603_RemoveRole**
Actor	**Users who have has “ROLE_CAN_REMOVE_ROLE” role**
Preconditions	**The url should be known by users**
Trigger	**This use case is initiated when user presses “Remove” button**
Basic Flow	**1. System provides “Remove” button.** **2. System redirects user to the role list page**
Exception Flow	
Post conditions	**The desired role is removed and user is redirected to role list page**

3.2.49.1 Functional Requirement

System should check user roles. User should have “ROLE_CAN_REMOVE_ROLE” role.

3.2.49.2 Functional Requirement

System should remove role from database.

3.2.49.3 Functional Requirement

System should redirect user to role list page

3.2.50 Remove Role

Use Case Name	**UC604_RemoveUsergroup**
Actor	**Users who have has “ROLE_CAN_REMOVE_USERGROUP” role**
Preconditions	**The url should be known by users**
Trigger	**This use case is initiated when user presses “Remove” button**
Basic Flow	**1. System provides “Remove” button.** **2. System redirects user to the usergroup list page**
Exception Flow	
Post conditions	**The desired usergroup is removed and user is redirected to usergroup list page**

3.2.50.1 Functional Requirement

System should check user roles. User should have “ROLE_CAN_REMOVE_USERGROUP” role.

3.2.50.2 Functional Requirement

System should remove usergroup from database.

3.2.50.3 Functional Requirement

System should redirect user to usergroup list page

3.2.51 Remove Training

Use Case Name	**UC605_RemoveTraining**
Actor	**Users who have has “ROLE_CAN_REMOVE_TRAINING” role**
Preconditions	**The url should be known by users**
Trigger	**This use case is initiated when user presses “Remove” button**
Basic Flow	**1. System provides “Remove” button.**

	2. System redirects user to the training list page
Exception Flow	
Post conditions	**The desired training is removed and user is redirected to training list page**

3.2.51.1 Functional Requirement

System should check user roles. User should have “ROLE_CAN_REMOVE_TRAINING” role.

3.2.51.2 Functional Requirement

System should remove training from database.

3.2.51.3 Functional Requirement

System should redirect user to training list page

3.2.52 Remove Training Definition

Use Case Name	**UC606_RemoveTrainingDefinition**
Actor	**Users who have has “ROLE_CAN_REMOVE_TRAINING_DEFINITION” role**
Preconditions	**The url should be known by users**
Trigger	**This use case is initiated when user presses “Remove” button**
Basic Flow	**1. System provides “Remove” button.** **2. System redirects user to the training definition list page**
Exception Flow	
Post conditions	**The desired training definition is removed and user is redirected to training definition list page**

3.2.52.1 Functional Requirement

System should check user roles. User should have “ROLE_CAN_REMOVE_TRAINING_DEFINITION” role.

3.2.52.2 Functional Requirement

System should remove training definition from database.

3.2.52.3 Functional Requirement

System should redirect user to training definition list page

3.2.53 Remove Subscriber

Use Case Name	**UC607_RemoveSubscriber**
Actor	**Users who have has “ROLE_CAN_REMOVE_SUBSCRIBER” role**
Preconditions	**The url should be known by users**
Trigger	**This use case is initiated when user presses “Remove” button**
Basic Flow	**1. System provides “Remove” button.** **2. System redirects user to the subscriber list page**
Exception Flow	
Post conditions	**The desired subscriber is removed and user is redirected to subscriber list page**

3.2.53.1 Functional Requirement

System should check user roles. User should have “ROLE_CAN_REMOVE_SUBSCRIBER” role.

3.2.53.2 Functional Requirement

System should remove subscriber from database.

3.2.53.3 Functional Requirement

System should redirect user to subscriber list page

3.2.54 Remove Term

Use Case Name	**UC608_RemoveTerm**
Actor	**Users who have has “ROLE_CAN_REMOVE_TERM” role**
Preconditions	**The url should be known by users**

Trigger	**This use case is initiated when user presses “Remove” button**
Basic Flow	**1. System provides “Remove” button.** **2. System redirects user to the term list page**
Exception Flow	
Post conditions	**The desired term is removed and user is redirected to term list page**

3.2.54.1 Functional Requirement

System should check user roles. User should have “ROLE_CAN_REMOVE_TERM” role.

3.2.54.2 Functional Requirement

System should remove term from database.

3.2.54.3 Functional Requirement

System should redirect user to term list page

3.2.55 Remove TaskSet

Use Case Name	**UC609_RemoveTaskset**
Actor	**Users who have has “ROLE_CAN_REMOVE_TASKSET” role**
Preconditions	**The url should be known by users**
Trigger	**This use case is initiated when user presses “Remove” button**
Basic Flow	**1. System provides “Remove” button.** **2. System redirects user to the taskset list page**
Exception Flow	
Post conditions	**The desired taskset is removed and user is redirected to taskset list page**

3.2.55.1 Functional Requirement

System should check user roles. User should have “ROLE_CAN_REMOVE_TASKSET” role.

3.2.55.2 Functional Requirement

System should remove taskset from database.

3.2.55.3 Functional Requirement

System should redirect user to taskset list page

3.2.56 Remove Grade

Use Case Name	**UC610_RemoveGrade**
Actor	**Users who have has “ROLE_CAN_REMOVE_GRADE” role**
Preconditions	**The url should be known by users**
Trigger	**This use case is initiated when user presses “Remove” button**
Basic Flow	**1. System provides “Remove” button.** **2. System redirects user to the grade list page**
Exception Flow	
Post conditions	**The desired grade is removed and user is redirected to grade list page**

3.2.56.1 Functional Requirement

System should check user roles. User should have “ROLE_CAN_REMOVE_GRADE” role.

3.2.56.2 Functional Requirement

System should remove grade from database.

3.2.56.3 Functional Requirement

System should redirect user to grade list page

3.2.57 Remove TaskType

Use Case Name	**UC611_RemoveTaskType**
Actor	**Users who have has “ROLE_CAN_REMOVE_TASKTYPE” role**

Preconditions	**The url should be known by users**
Trigger	**This use case is initiated when user presses “Remove” button**
Basic Flow	**1. System provides “Remove” button.** **2. System redirects user to the tasktype list page**
Exception Flow	
Post conditions	**The desired tasktype is removed and user is redirected to tasktype list page**

3.2.57.1 Functional Requirement

System should check user roles. User should have “ROLE_CAN_REMOVE_TASKTYPE” role.

3.2.57.2 Functional Requirement

System should remove tasktype from database.

3.2.57.3 Functional Requirement

System should redirect user to tasktype list page

3.2.58 Remove Task

Use Case Name	**UC612_RemoveTask**
Actor	**Users who have has “ROLE_CAN_REMOVE_TASK” role**
Preconditions	**The url should be known by users**
Trigger	**This use case is initiated when user presses “Remove” button**
Basic Flow	**1. System provides “Remove” button.** **2. System redirects user to the task list page**
Exception Flow	
Post conditions	**The desired task is removed and user is redirected to task list page**

3.2.58.1 Functional Requirement

System should check user roles. User should have “ROLE_CAN_REMOVE_TASK” role.

3.2.58.2 Functional Requirement

System should remove task from database.

3.2.58.3 Functional Requirement

System should redirect user to task list page

3.2.59 Remove TaskUnit

Use Case Name	**UC613_RemoveTaskUnit**
Actor	**Users who have has “ROLE_CAN_REMOVE_TASKUNIT” role**
Preconditions	**The url should be known by users**
Trigger	**This use case is initiated when user presses “Remove” button**
Basic Flow	**1. System provides “Remove” button.** **2. System redirects user to the taskunit list page**
Exception Flow	
Post conditions	**The desired taskunit is removed and user is redirected to taskunit list page**

3.2.59.1 Functional Requirement

System should check user roles. User should have “ROLE_CAN_REMOVE_TASKUNIT” role.

3.2.59.2 Functional Requirement

System should remove taskunit from database.

3.2.59.3 Functional Requirement

System should redirect user to taskunit list page

3.2.60 Remove Measurement Criteria

Use Case Name	**UC614_RemoveMeasurementCriteria**
Actor	**Users who have has “ROLE_CAN_REMOVE_MEASUREMENT_CRITERIA” role**
Preconditions	**The url should be known by users**
Trigger	**This use case is initiated when user presses “Remove” button**
Basic Flow	**1. System provides “Remove” button.** **2. System redirects user to the measurement criteria list page**
Exception Flow	
Post conditions	**The desired measurement criteria is removed and user is redirected to measurement criteria list page**

3.2.60.1 Functional Requirement

System should check user roles. User should have “ROLE_CAN_REMOVE_MEASUREMENT_CRITERIA” role.

3.2.60.2 Functional Requirement

System should remove measurement criteria from database.

3.2.60.3 Functional Requirement

System should redirect user to measurement criteria list page

3.2.61 Remove Measurement Set

Use Case Name	**UC615_RemoveMeasurementSet**
Actor	**Users who have has “ROLE_CAN_REMOVE_MEASUREMENT_SET” role**
Preconditions	**The url should be known by users**
Trigger	**This use case is initiated when user presses “Remove” button**
Basic Flow	**1. System provides “Remove” button.** **2. System redirects user to the measurement set list page**
Exception Flow	

Post conditions	**The desired measurement criteria is removed and user is redirected to measurement set list page**

3.2.61 .1 Functional Requirement

System should check user roles. User should have “ROLE_CAN_REMOVE_MEASUREMENT_SET” role.

3.2.61 .2 Functional Requirement

System should remove measurement set from database.

3.2.61 .3 Functional Requirement

System should redirect user to measurement set list page

3.3 Usability Requirements

Since the target group is from any country, system will have multiple language support. Users will be able to make their own translation.

System is designed for wide range of sectors. It should be easy to adapt to software companies, survey companies, universities, colleges, high scols etc. It will have much more abstraction on entities such as users, trainings etc.

3.4 Performance Requirements

System should be scalable to increase the maximum number of online users. Loading of a single page shouldn’t exceed 500ms for user satisfaction.

Database of the system should support 20 term with 50.000 users with 1.500 training. Server capacity should be sufficient enough to handle this much user at a time.

3.5 Logical Database Requirements

The Entity-Relation Diagram of database is detailed in Appendix H.

The package diagram is detailed in Appendix K.

The class diagrams are detailed in Appendix L, Appendix M, Appendix N and Appendix O, Appendix P.

3.6 Design Constraints

All detailed information is shown in Appendix A, B, C, D, E, F, G, H, I, K, L, M, O, P.

3.7 Software System Attributes

3.7.1 Availability

System should be robust to serve 98% of a time. Users will be able to use it as long as they have internet connection and proper web browser.

3.7.2 Security

The system should protect users personal information. A session will be created by members through entering his/her id and password and any other user shouldn't be able to access to other members' private data.

The web portal should be protected against SQL-Injection and other attack techniques.

Besides, system should be robust against DDOS attacks to serve at any time.

3.7.3 Portability

The web part of the system is independent from the type of OS and web browser installed on device. Through the use of proper design techniques the application will be compatible with both desktop and mobile devices.

3.7.4 Usability

The web portal GUI should be clear and understandable for users. It will not contain any ambiguous meaning. All of the functions should have help boxes which contain meaning of the function and manuals.

Error and success messages will provide a feedback to users to improve their usage skills and knowledge.

The embedded systems should be simple and they should have manuals.

3.8 Supporting Information

There is no supporting information.

4. VERIFICATION

The verification standards are not considered in this SRS document.

5. APPENDICIES

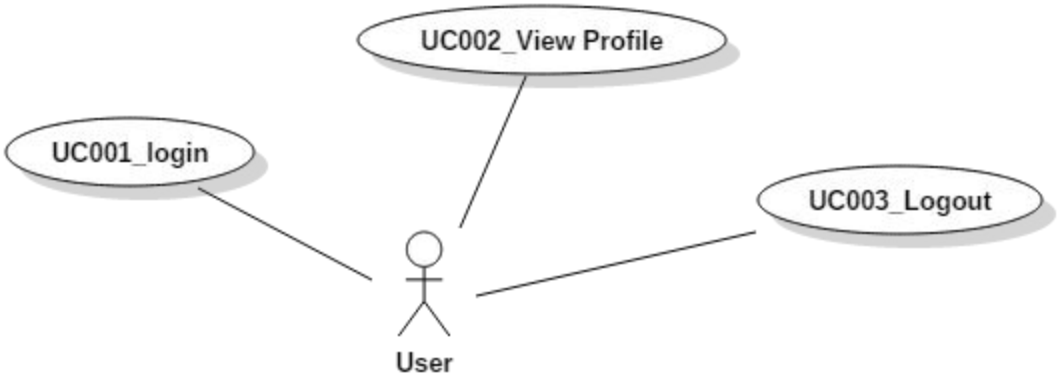
5.1Assumption and Dependencies

Project team members will remain the same project until the project is completed.
The users are expected to have ability to understand manuals.

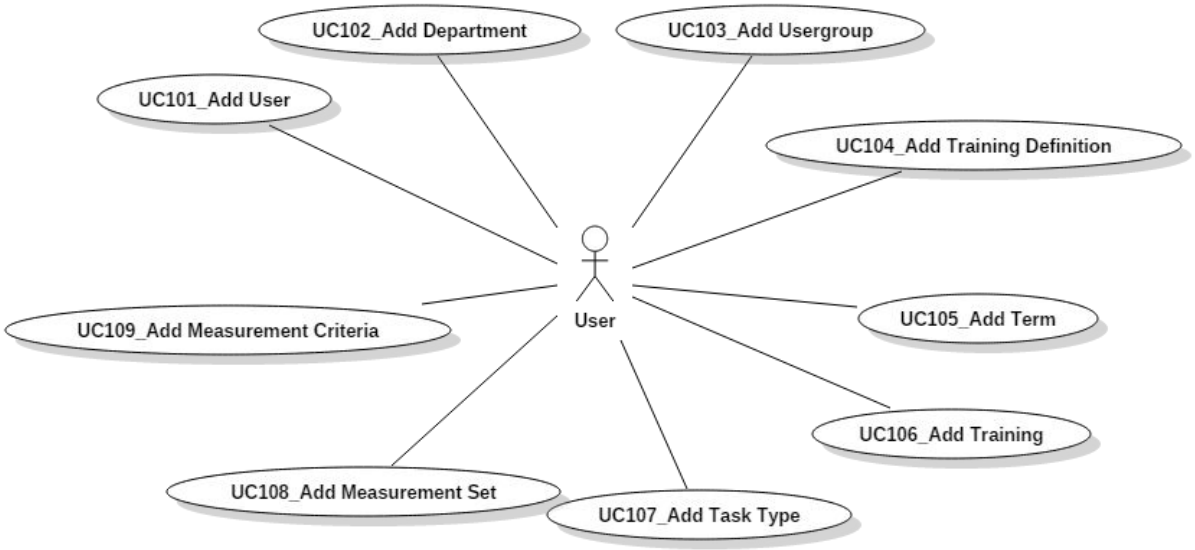
5.2 Acronyms and Abbrevations

Term	Definition
SQL	Structured Query Language
GUI	Graphical User Interface
CRUD	Create, Read, Update, Delete operations
WYSIWYG	What you see is what you get editor

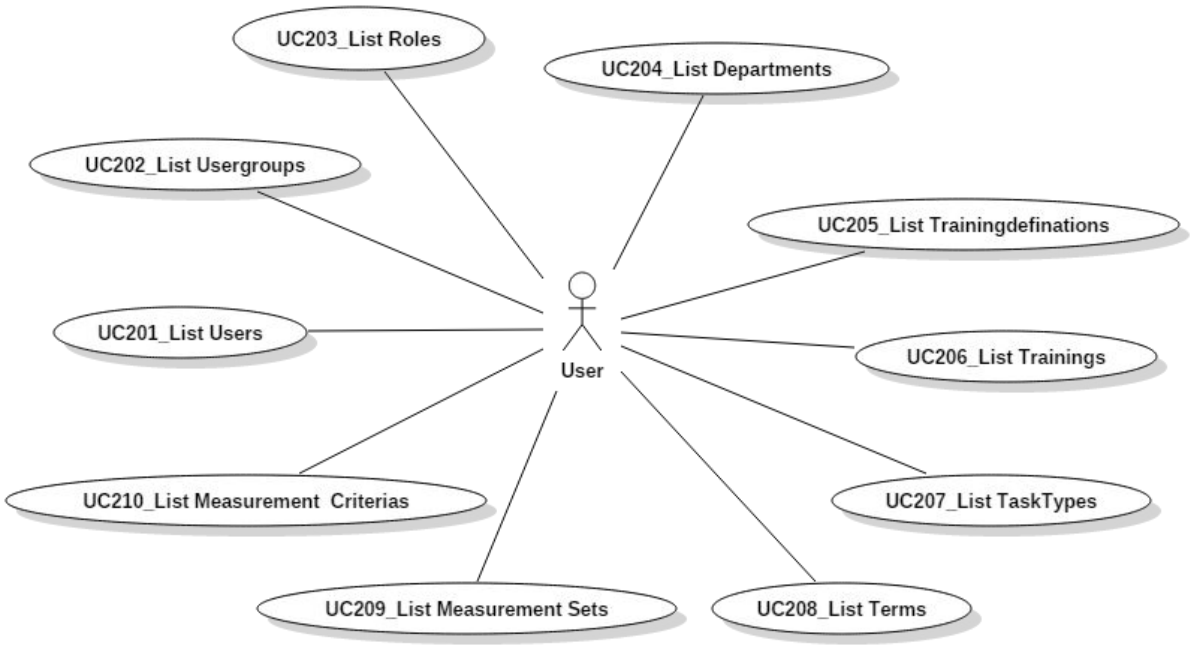
APPENDIX A – Use Case Diagram User Basic



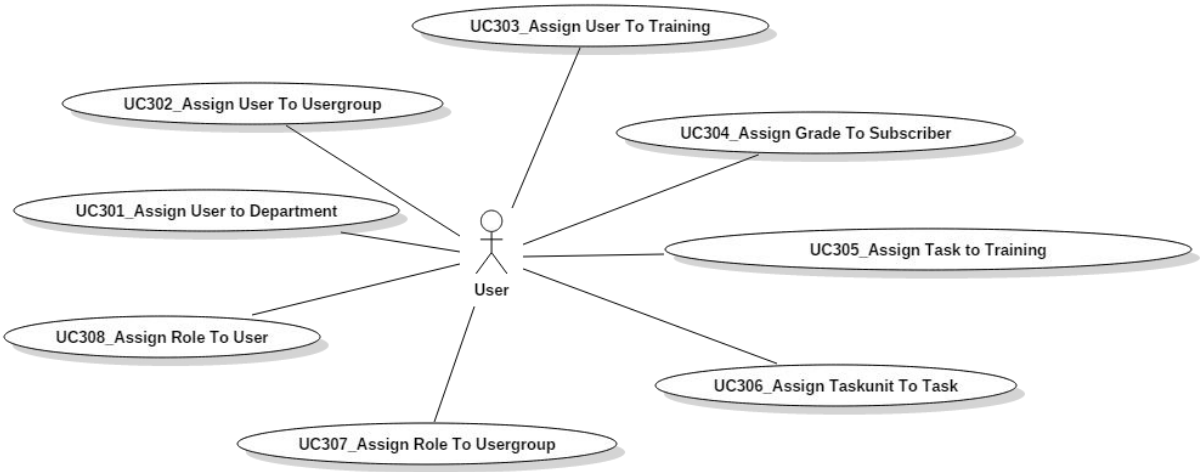
APPENDIX B – Use Case Diagram Adding Model



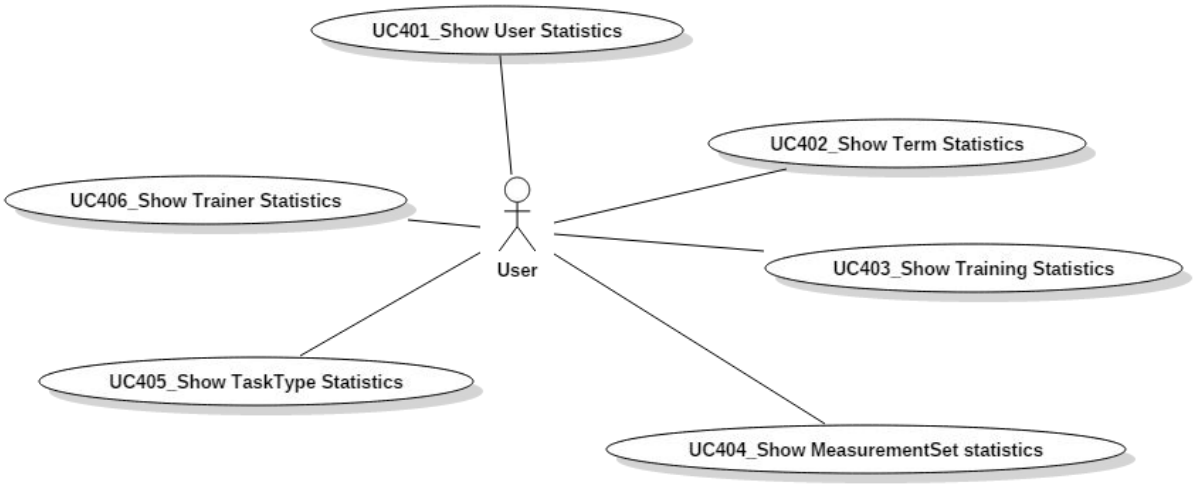
APPENDIX C – Use Case Diagram List



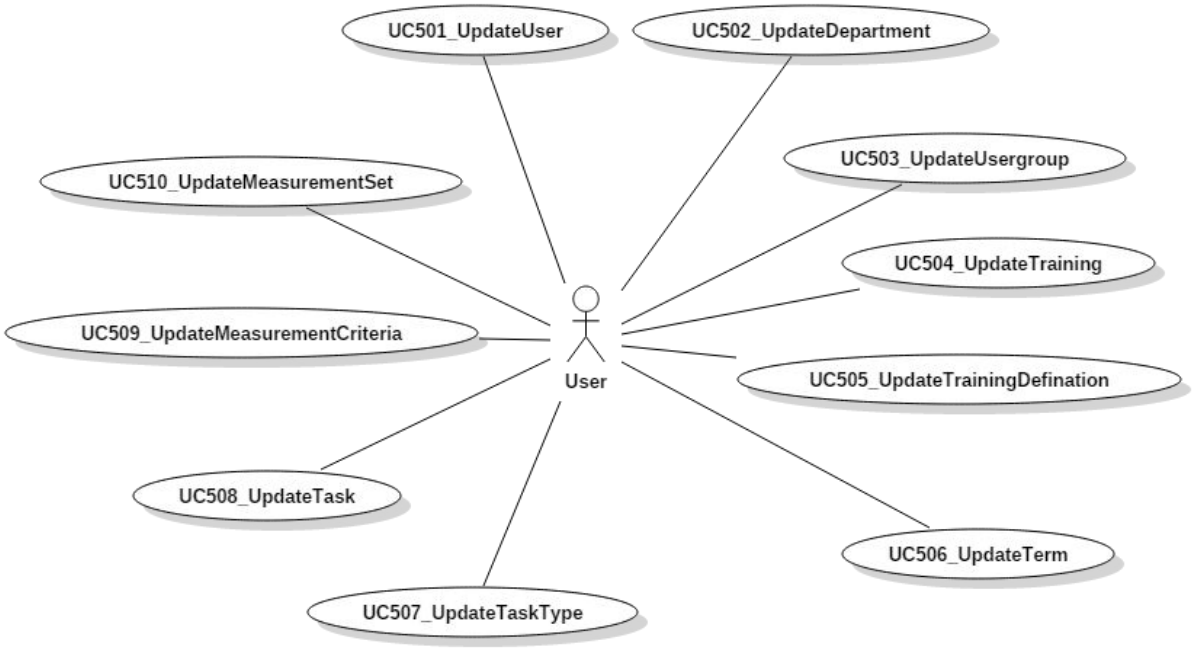
APPENDIX D – Use Case Diagram Assign



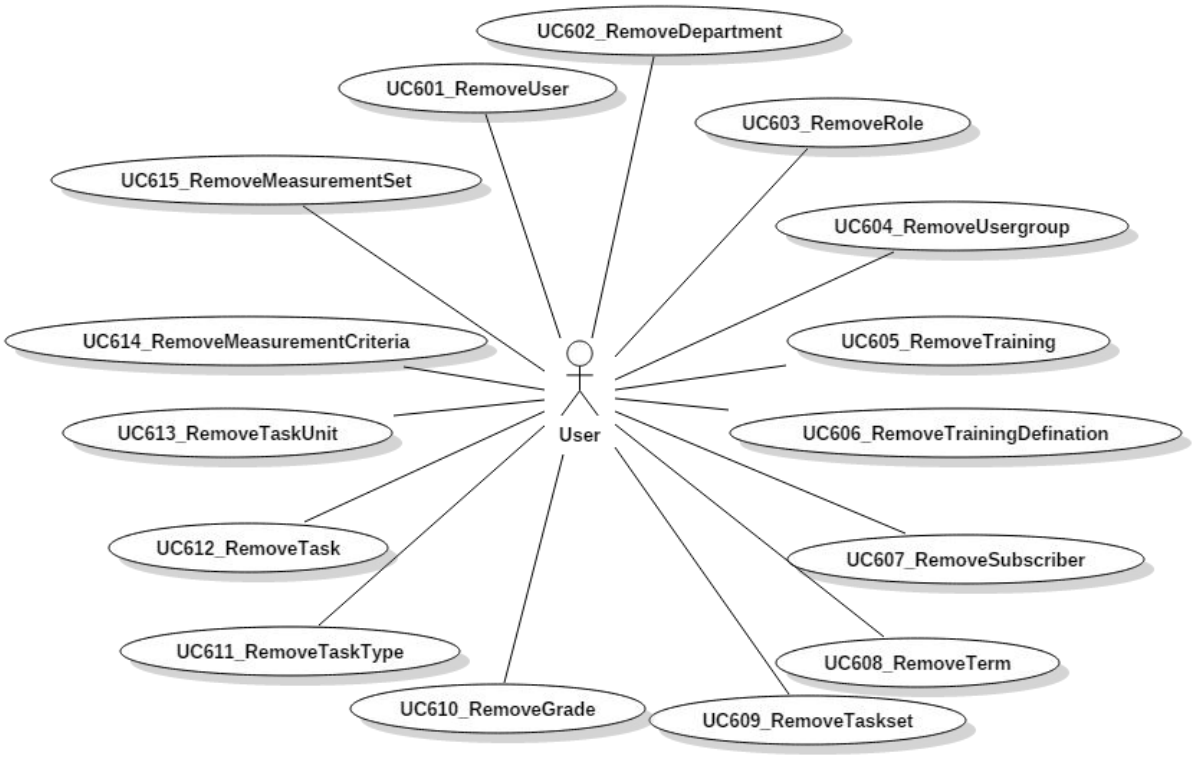
APPENDIX E – Use Case Diagram Statistics



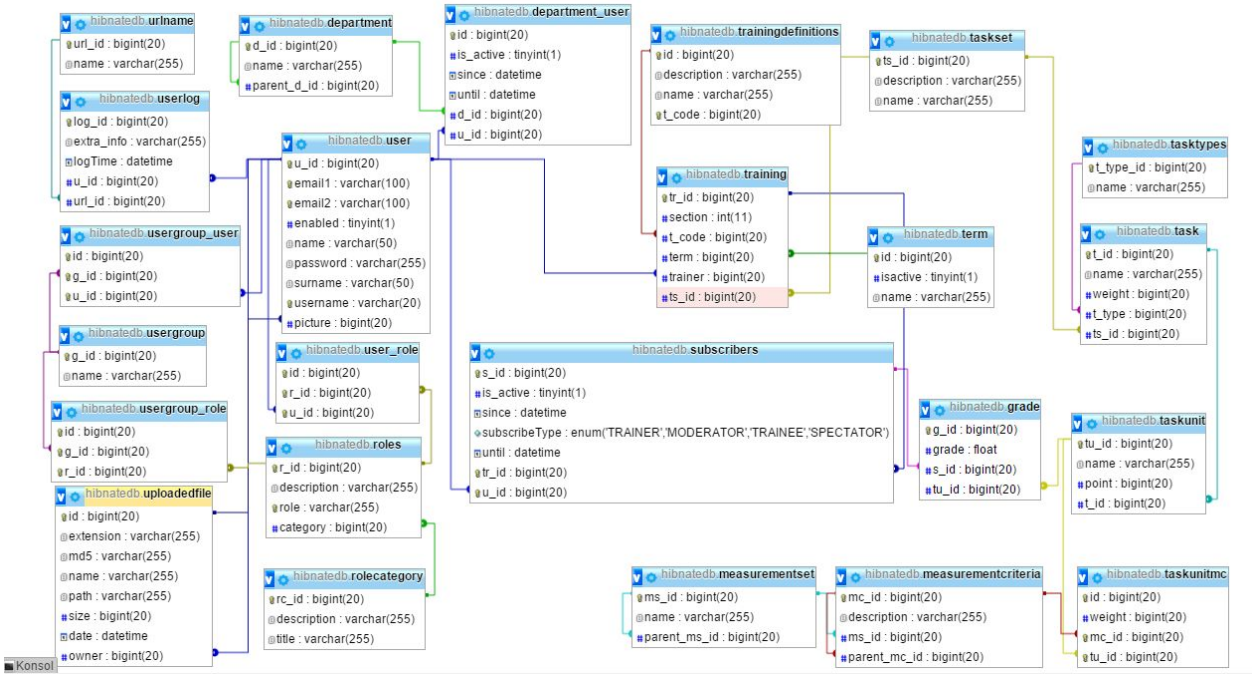
APPENDIX F – Use Case Diagram Update



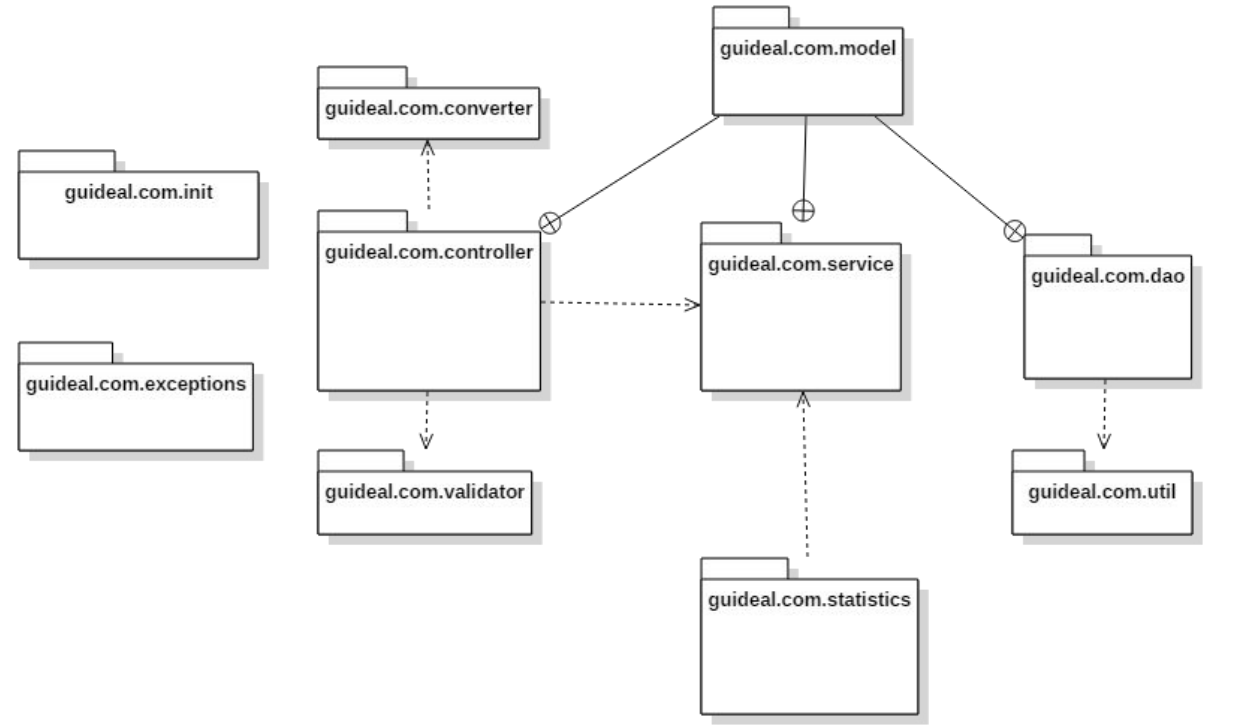
APPENDIX G – Use Case Diagram Remove



APPENDIX H – Database ER Diagram

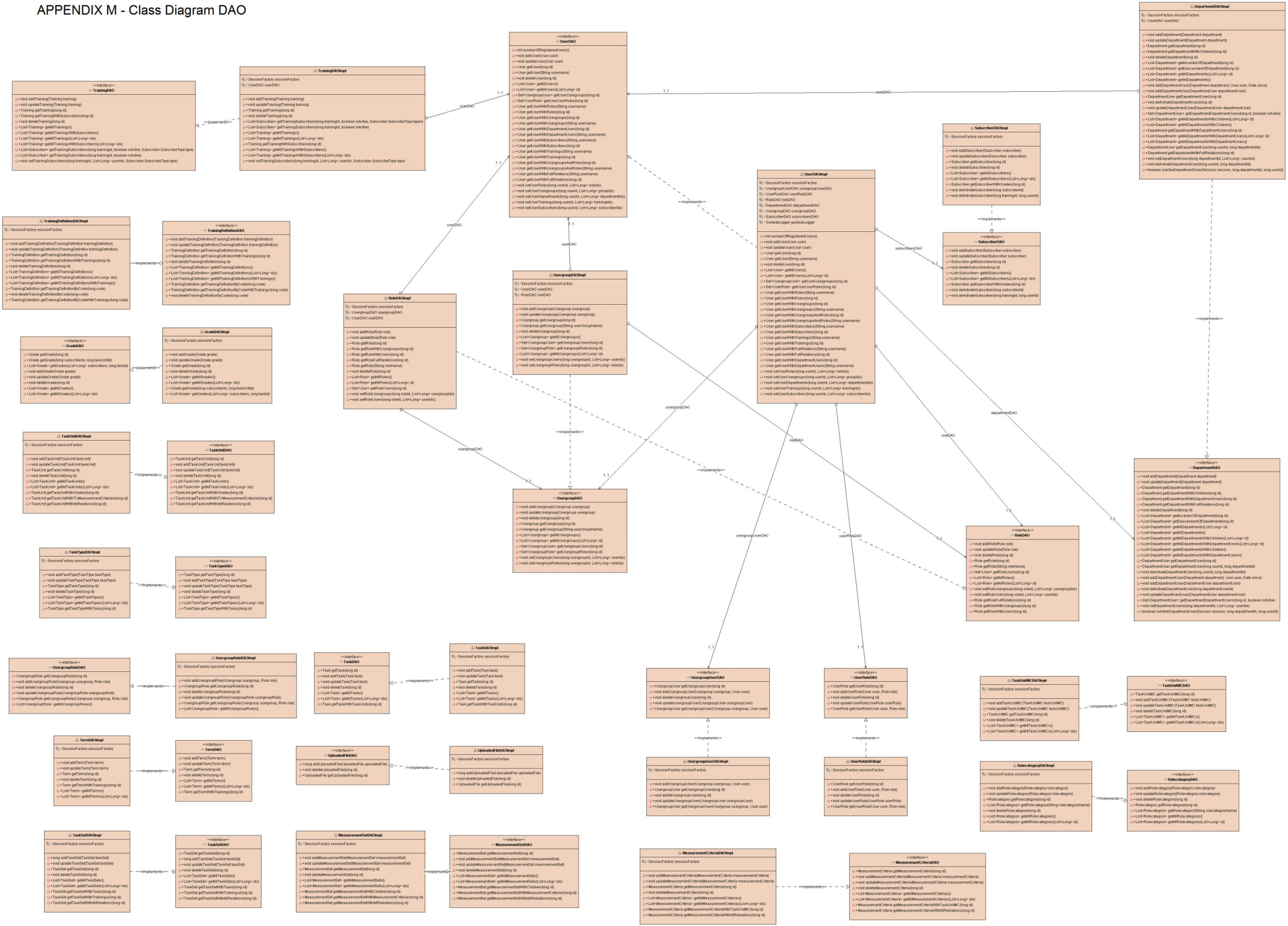


APPENDIX K - Package Diagram

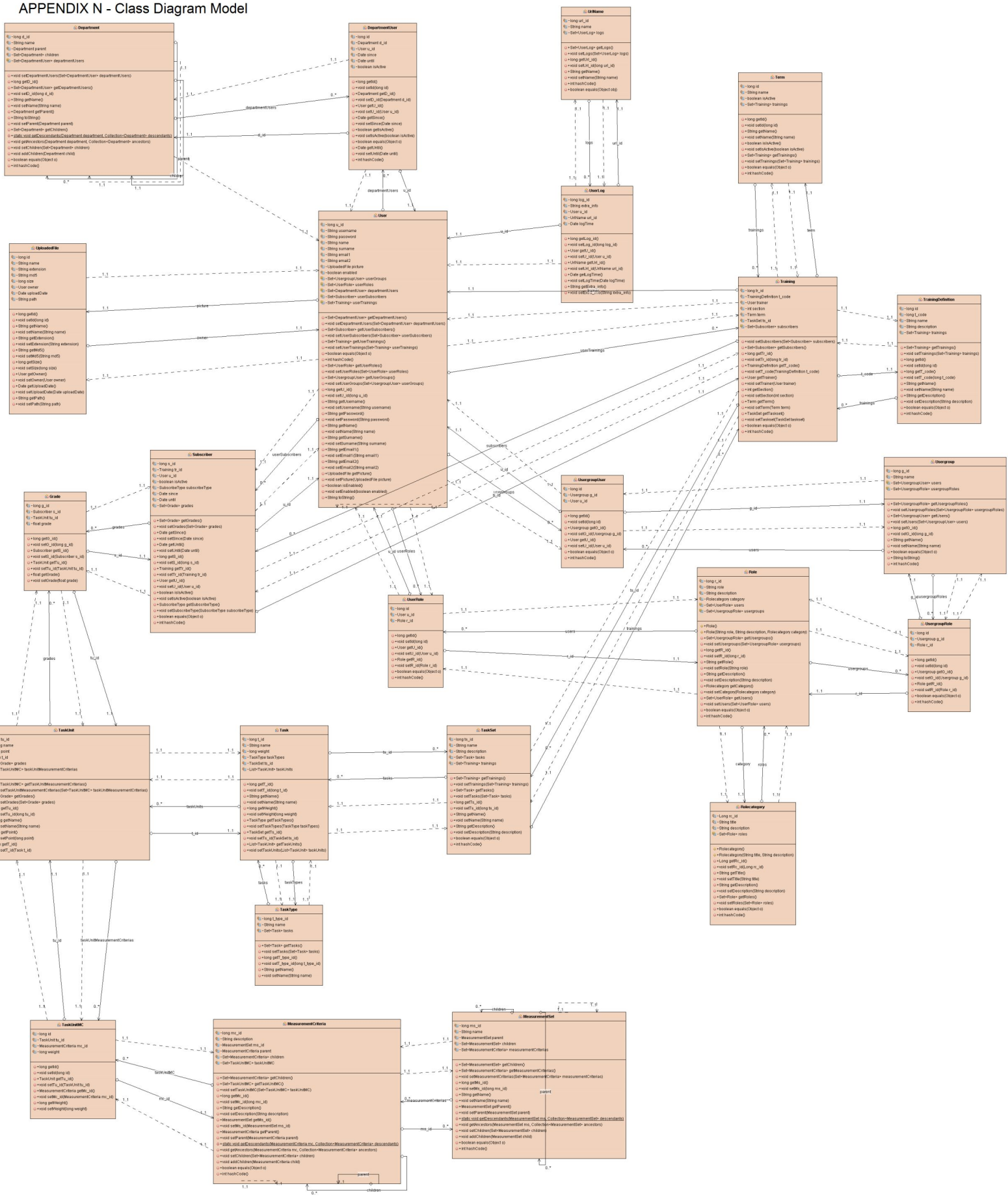


APPENDIX L - Class Diagram Controller

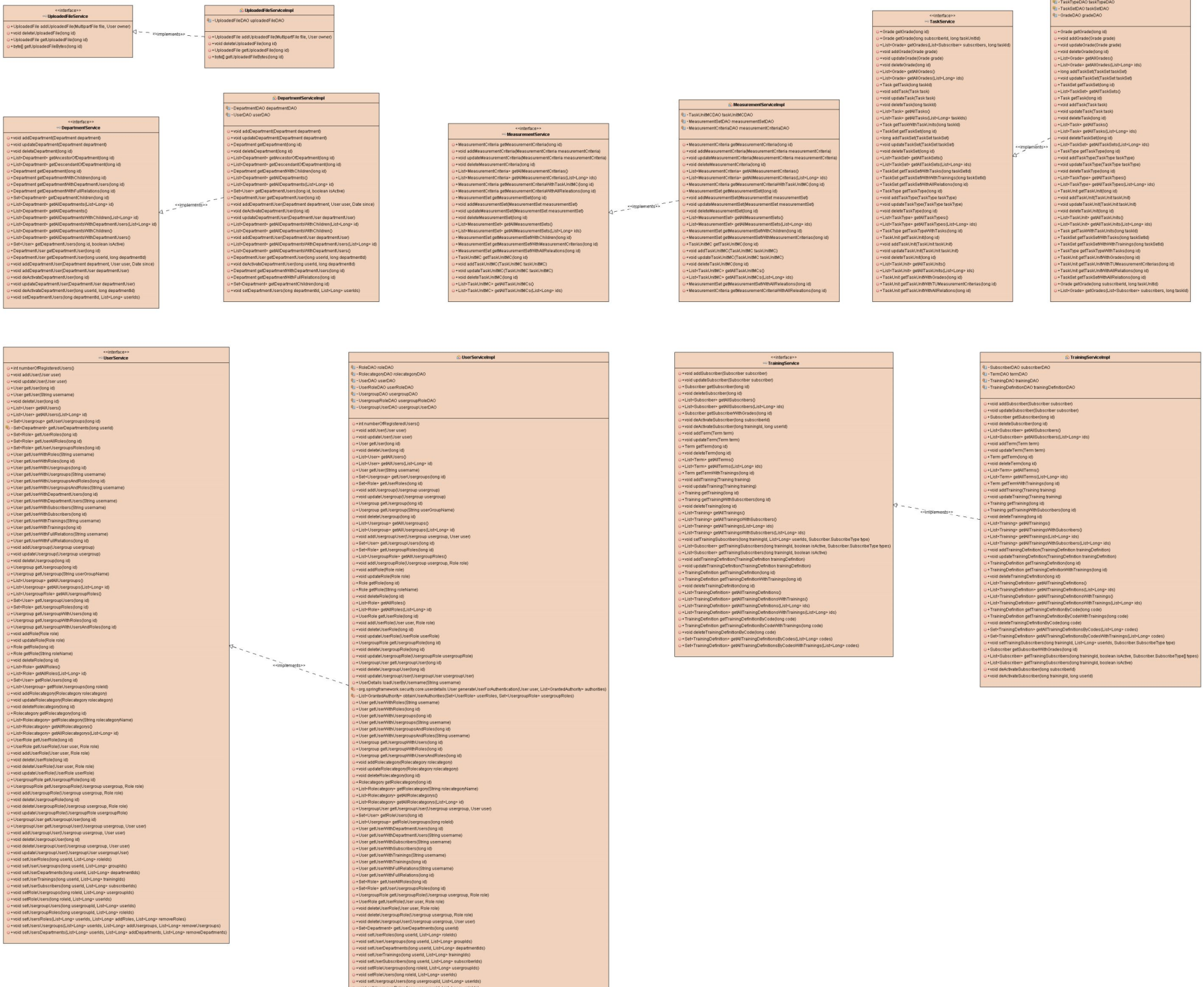
APPENDIX M - Class Diagram DAO



APPENDIX N - Class Diagram Model



APPENDIX L - Class Diagram Service



APPENDIX L - Class Diagram Util