Qualman development project

Group 27

Krister Kjellström, kristerk@kth.se
Elena Malioutina, elenamalioutina@hotmail.com
Rahwa Tedros, rahwa@hotmail.com
Pia-Marie Jörgensen, pia.jorgensen@bredband.net

Document Design Document

System Qualman

Version 1.0

Intended readership Client engineers, system architects and system developers.

Date 2007-03-19

References

#	*Document*
1	Qualman Requirements Document, RD_v1.1_group27.doc
2	*Kent Beck, Apple Computer, Inc.Ward Cunningham, Wyatt Software Services, Inc.* "A laboratory for teaching object-oriented thinking, OOPSLA'89 Conference Proceedings October 1-6, 1989, New Orleans, Louisiana ; the special issue of SIGPLAN Notices Volume 24, Number 10, October 1989.

Version history

Version	*Comment (reason for / summary of changes)*	*Date*	*Author(s)*
1.0	First version.	2007-03-19	Elena Malioutina, Rahwa Tedros, Krister Kjellström, Pia-Marie Jörgensen

Contents

1. Introduction

Pharmaceutical companies perform testing of equipment, processes, methods and products to verify that they perform according to specification and do not constitute a risk to patients using the medicines produced. These testing procedures are referred to as qualifications The Qualman system is an administrative system designed to keep track of the information and documentation produced during qualifications and to provide a repository from which reports detailing information requested by authorities can be produced.

This document describes the design of the Qualman software and is intended to give developers sufficient information to develop the system.

A system overview is given in section 2 with a general description of the system and its overall architecture. The system is designed as a three-tier client-server system. The client is a standard web browser, and the presentation and business logic are handled in a server-side web application. A relational database provides a common repository for the system.

General design considerations can be found in section 3 and the graphical user interface in section 4. ASP.NET (Active Server Pages .NET) web forms with code-behind pages provide users with an interface to enter and retrieve data in the system. Design details are listed in section 5 and section 6 concerns test cases to test if the functional requirements of the system are fulfilled.

2. System Overview

2.1 General Description

The Qualman system can be used for managing information about qualifications performed by a pharmaceutical company on products, processes, methods and equipment (a background to the uses of the system can be found in the Requirements Document, *Ref 1*). Information about products, processes, methods and equipment can be entered separately in advance and chosen for a particular qualification from list-of-values to avoid having to enter and store the same information several times.

In addition to storing unique qualification numbers and information relating to these, the system provides possibilities to search for and extract information in a format readable by standard office applications such as MS Office Excel. The system can also store links to electronic documentation regarding qualifications, which can be accessed from within the system.

The system has storage possibilities for references to IDs in other systems that may be used in quality assurance by pharmaceutical companies, such as change control, project tracking, equipment maintenance and standard operating instruction systems. It does however not at present provide any direct interfaces to such systems, although customised data transfer procedures could be implemented using relational database and middleware techniques.

The system provides several levels of access, e.g. admin, change and read access. Authentication can be implemented as single sign-on verifying Windows domain account and domain group information of the logged on user in Active Directory. Server and database to use are chosen when logging on.

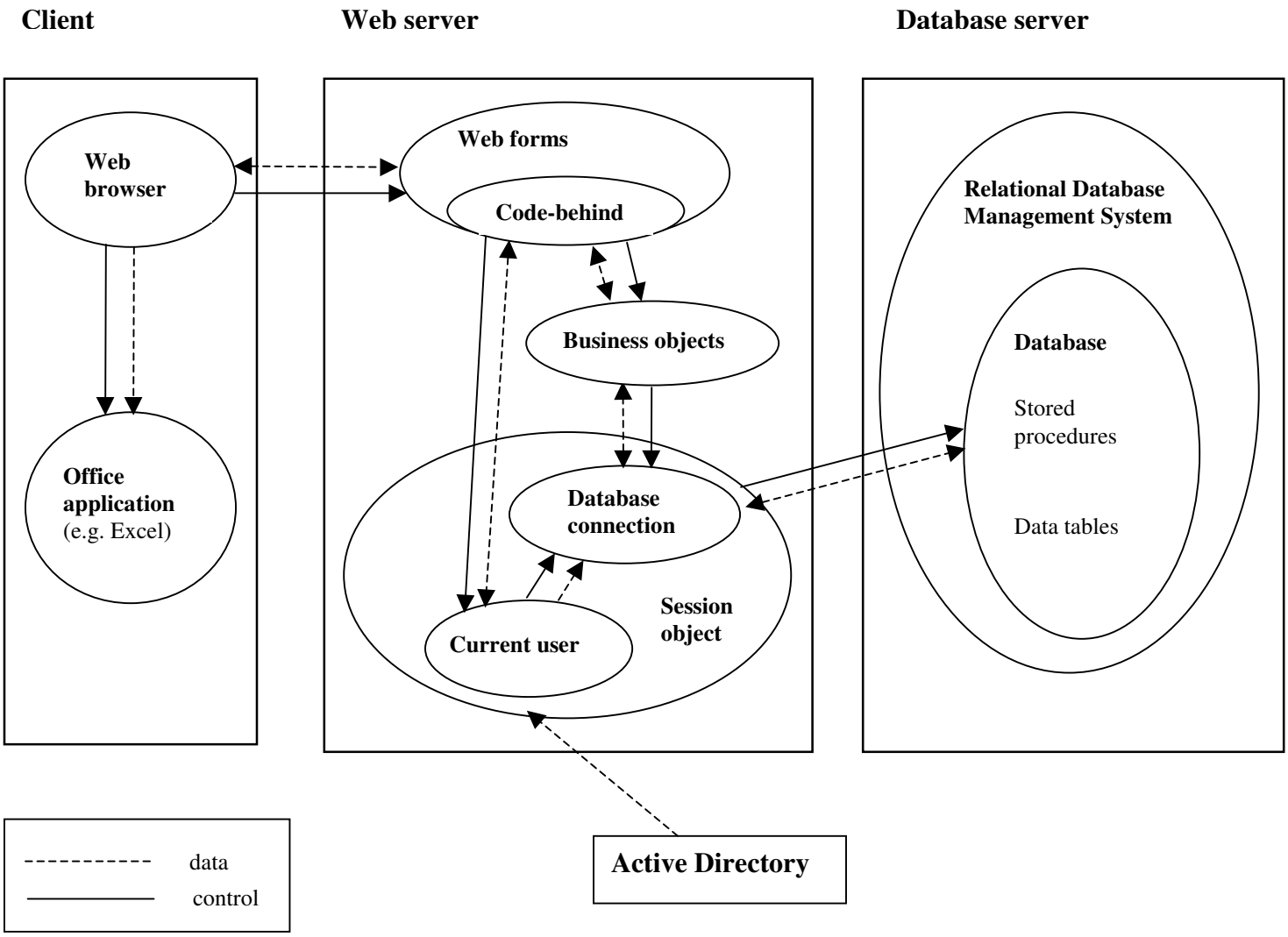
A three-tier client-server architecture will be used for the implementation of the system. The middle tier will be a

web server based on the need for easy access for a large number of users to the system, and the ease of maintenance and validation using a centralised application. In a pharmaceutical environment each installation of computer software needs to be validated separately, which means a substantial amount of work when changes or upgrades are performed.

Database stored procedures will be used for data access and storage. Running code that performs database operations on the database server reduces network traffic, which improves performance. As more data is stored in the system, the time to perform SQL statements and database I/O operations quickly becomes an issue and is often the most important factor in performance of database applications. Stored procedures also give better control of security and more possibilities to prevent injection. Built-in database transaction handling in the Relational Database Management System can also be used.

2.2 Overall Architecture

The Qualman system is a three-tier client server system consisting of a client, a web server and database server as depicted in the figure below.



The client uses a standard web browser to show the pages presented to it by the web server. Users interact with the system by navigating the web pages. Client office applications such as Excel are invoked on the client to present information retrieved by searches in the system or to show electronic documentation, which is linked to the system.

The web server part of the system consists of web forms with corresponding "code-behind" objects that respond to user input and communicates with business logic objects. Database connection and user objects stored for the session authenticate the user, keep track of the user access level and handle database communication. The database connection object invokes database stored procedures to respond to requests from users to store or retrieve information from the database.

The database server runs a relational database management system. The system database consists of tables storing the data and a number of stored procedures used for data access and storage.

2.3 Detailed Architecture

The detailed architecture is illustrated using examples with Class Responsibility Collaborator (CRC) cards (described by e.g. Beck and Cunningham, 1989, *Ref 2*) and sequence diagrams defined in the Unified Modelling Language (UML).

2.3.1 CRC cards

Web browser	
Responsibilities	**Collaborators**
Present user interface Forward user input to web server Invoke office applications to present data on specific formats	Web forms Office applications

Office applications	
Responsibilities	**Collaborators**
Present data of the relevant format	Web browser

User	
Responsibilities	**Collaborators**
Authenticates user. Hold information about current user Hold information about user access level. Holds information about user name.	Code-behind Database connection

DbConnection	
Responsibilities	**Collaborators**
Hold information about the current database server and database and the current database connection.	User Business objects Database

Database	
Responsibilities	**Collaborators**
Store data and procedures Execute stored procedures Return data on request	Database connection

Business objects	
Responsibilities	**Collaborators**
Store data and procedures Execute stored procedures Return data on request	Code-behind Database connection

Code-behind	
Responsibilities	**Collaborators**
React to control events in web forms Provide business logic Store and retrieve information about current user and database (login code-behind) Perform operations and retrieve data requested by web forms in collaboration with the database connection object.	Web forms User Business objects

Web forms	
Responsibilities	**Collaborators**
Provide user interface Respond to user input	Web browser Code-behind

Examples of web forms and corresponding code-behind

Login	
Responsibilities	**Collaborators**
Verify identity of user Determine access level of user and store user information Store information about database connection Initiate Qualman application	Web browser User Database connection Qualman

Qualman	
Responsibilities	**Collaborators**
Welcome user to system Present menu for application choices Respond to user input	Other web forms User

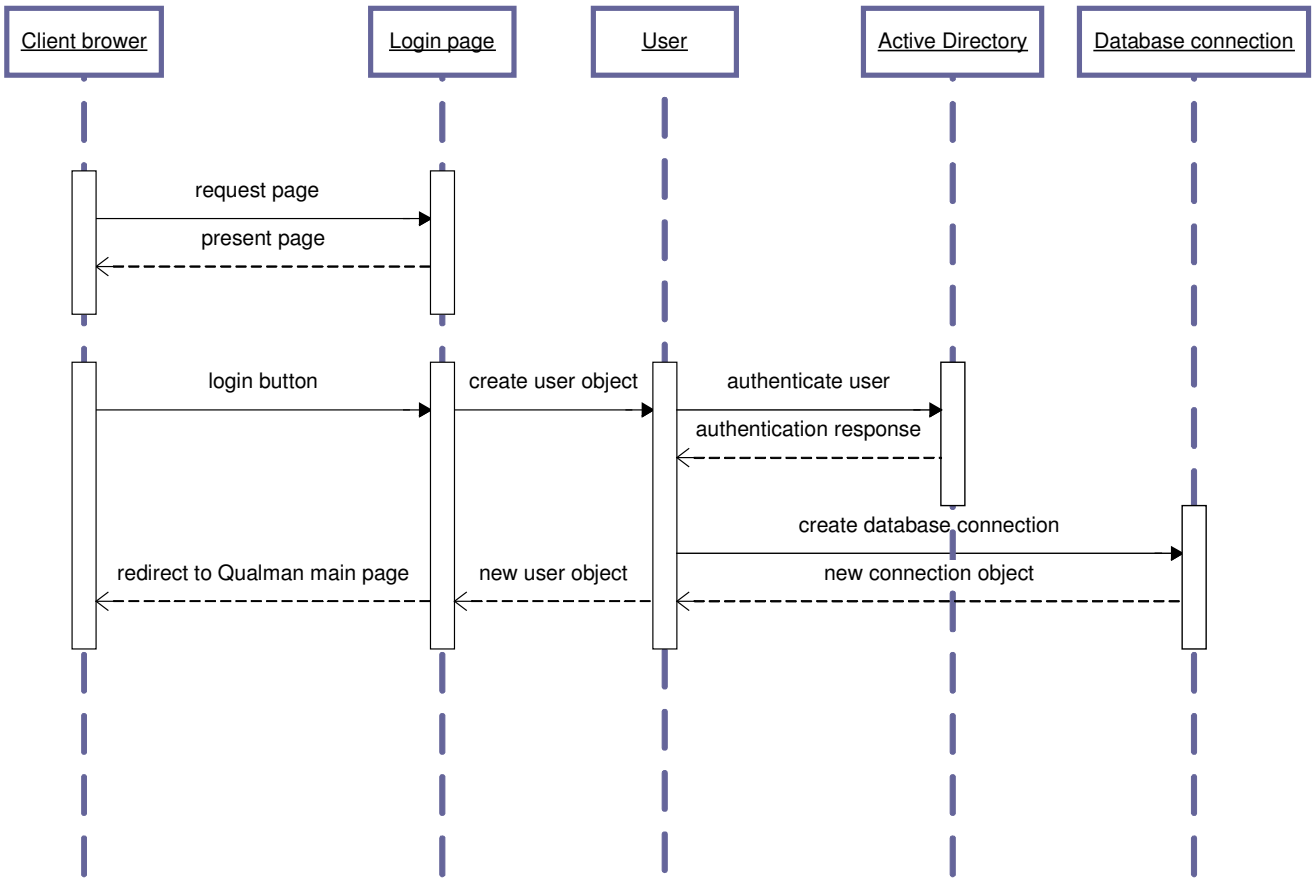
SearchQual	
Responsibilities	**Collaborators**
Collect user search parameters Respond to user initiation of search Display search results. Handle user requested export of results in an Excel readable format Present menu for application choices	User Database connection

For further examples see section 4 for the complete graphical user interface and section 5 for a detailed description of the design.

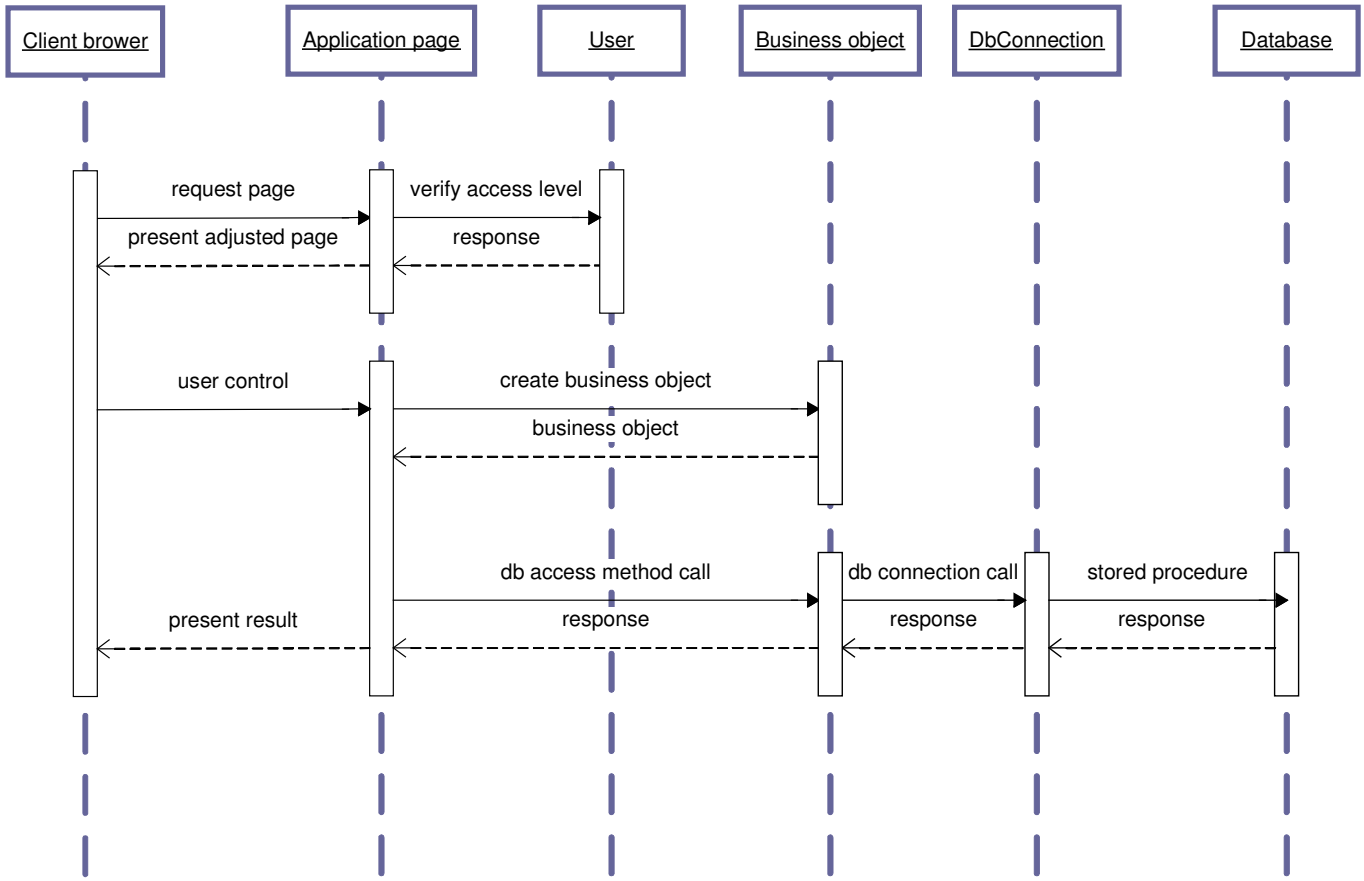
2.3.2 Sequence diagrams

Examples of communication between components are given below. For more detailed interaction diagrams see section 5.4.

Login procedure



Application database communication



3. Design Considerations

3.1 Assumptions and Dependencies

This software will be implemented using Microsoft ASP.NET. It should be used in a Windows OS environment and it will require .NET 2.0 or higher to be installed on the client computers. The web server used should support the use of ASP.NET and ADO.NET web forms. Active Directory will be used for user authentication.

End users should be trained in the use of the system and should also have basic education in the validation process the system is based on.

3.2 General Constraints

Factors affecting the performance of the system is database license availability, network capacity, web server capacity and database server capacity. Stored procedures are used to make it possible to run SQL procedures entirely on the database server to reduce network traffic. The system is designed as a server-side web application to reduce the need for validation of separate installations of the software required in the regulatory environment (i.e. pharmaceutical industry) where the software will be used. The use of a web application also adds scalability and facilitates adding new clients to the system.

4. Graphical User Interface

The graphical user interface provides the user with a number of web forms to access and store data in the Qualman system. A meny is presented to the left to provide navigation options for the current form. There are forms to retreive new qualification numbers, to edit data of existing ones and search options to list and export qualification information for a number of given search parameters to Excel. Where relevant, a search function for qualification numbers is provided at the top of the form and the current qualfication number and title of the qualification are displayed.

The user interface is presented in detail in appendix 2. For further details see section 5 for a detailed description of the presentation layer.

5. Design Details

5.1 Class Responsibility Collaborator (CRC) Cards

5.1.1 Presentation layer

Login	
Responsibilities	**Collaborators**
Display a login interface. Accept username, password, database and server. Authenticate user. Create User session object. Establish a database connection	User DbConnection

Qualman (main page)	
Responsibilities	**Collaborators**
Display a welcome message.	MainMenu

MainMenu	
Responsibilities	**Collaborators**
Display appropriate menu, depending on the user. Accept a choice from user.	User (Session) EditSubMenu EditSubQual LOVMenu CloseQual RemActivity QualDoc

NewQual	
Responsibilities	**Collaborators**
Display an interface for adding a new qualification number. Accept needed qualification information from user. Respond to save-button being pressed; display a message containing the new qualification number. Respond to cancel-button being pressed. Display conformation message. Create Qualification Object	User (Session) Qualification

AddSubQual	
Responsibilities	**Collaborators**
Display an interface for adding a new sub-qualification number to an existing qualification. Accept needed qualification information from user. Display a message containing the next available sub-qualification number Respond to save-button being pressed. Respond to cancel-button being pressed. Display conformation message. Create Qualification Object	User (Session) Qualification

EditSubQual	
Responsibilities	**Collaborators**
Display interface for editing information for a sub-qualification. Respond to save-button being pressed. Respond to cancel-button being pressed. Display conformation message. Create Qualification Object	User (Session) Qualification

EditSubMenu	
Responsibilities	**Collaborators**
Display an edit sub-qualification menu. Accept Choice from user.	User (Session) EditEquip EditMtrlCode EditMethod EditProcessStep EditProcessStep

QualEquip	
Responsibilities	**Collaborators**
Display interface for adding equipment. Respond to save-button. Display confirmation message.	User (Session) Equipment

QualMethod	
Responsibilities	**Collaborators**
Display interface for adding method. Accept information from user. Respond to save-button. Display confirmation message.	User (Session) Method

QualMtrlCode	
Responsibilities	**Collaborators**
Display interface for adding material code. Accept information from user. Respond to save-button. Display confirmation message.	User (Session) MtrlCode

QualProcessStep	
Responsibilities	**Collaborators**
Display interface for adding process step. Accept information from user. Respond to save-button. Display confirmation message.	User (Session) ProcessStep

QualDoc	
Responsibilities	**Collaborators**
Display interface for adding document. Accept information from user. Respond to save-button. Display conformation message. Create Document Object.	User (Session) Document

CloseQual	
Responsibilities	**Collaborators**
Display interface for closing a qualification method. Accept information from user. Respond to close-button. Display conformation message. Create Qualification Object.	User (Session) Qualification

LOV	
Responsibilities	**Collaborators**
Present the LOVMenu and Explain how it is used. Accept a choice from user.	User (session) LOVMenu

LOVMenu	
Responsibilities	**Collaborators**
Display a menu for list of value. Accept a choice from user.	User (Session) LOV

RemActivity	
Responsibilities	**Collaborators**
Display interface for changing remaining activities. Accept changes from user. Display conformation message. Create Activity Object.	User (Session) Activity MainMenu

EditEquip	
Responsibilities	**Collaborators**
Display interface for editing equipment information for a sub-qualification. Respond to save-button being pressed. Respond to cancel-button being pressed. Display conformation message. Create Equipment Object.	User (Session) Equipment Qualification EditSubMenu

EditMethod	
Responsibilities	**Collaborators**
Display interface for editing method information for a sub-qualification. Respond to save-button being pressed. Respond to cancel-button being pressed. Display conformation message. Create Method Object	User (Session) Method Qualification EditSubMenu

EditMtrlCode	
Responsibilities	**Collaborators**
Display interface for editing materialcode information for a sub-qualification. Respond to save-button being pressed. Respond to cancel-button being pressed. Display conformation message. Create MtrlCode Object	User (Session) MtrlCode EditSubMenu

EditProcessStep	
Responsibilities	**Collaborators**
Display interface for editing process step information for a sub-qualification. Respond to save-button being pressed. Respond to cancel-button being pressed. Create ProcessStep Object Create Qualification Object	User (Session) ProcessStep Qualification EditSubMenu

SearchQual (UC11)	
Responsibilities	**Collaborators**
Display interface for editing process step information for a sub-qualification. Respond to search-button being pressed. Respond to cancel-button being pressed.	User (Session) Qualification

EditQualUser	
Responsibilities	**Collaborators**
Display interface for editing user information. Respond to save-button being pressed. Respond to cancel-button being pressed. Display conformation message. Create User Object.	User (Session)

EditProduct	
Responsibilities	**Collaborators**
Display interface for editing product information for equipment. Respond to save-button being pressed. Respond to cancel-button being pressed. Display conformation message. Create Equipment Object.	User (Session) Equipment

EditDepartment	
Responsibilities	**Collaborators**
Display interface for editing department information for a sub-qualification. Respond to save-button being pressed. Respond to cancel-button being pressed. Display conformation message. Create Qualification Object.	User (Session) Qualification

EditRoom	
Responsibilities	**Collaborators**
Display interface for editing room information for a sub-qualification. Respond to save-button being pressed. Respond to cancel-button being pressed. Display conformation message. Create Qualification Object.	User (Session) Qualification

EditInstructionCategory	
Responsibilities	**Collaborators**
Display interface for editing standard operating procedure information for a qualification. Respond to save-button being pressed. Respond to cancel-button being pressed. Display conformation message. Create Qualification Object.	User (Session) Qualification

EditEquipType	
Responsibilities	**Collaborators**
Display interface for editing equipment type information for equipment. Respond to save-button being pressed. Respond to cancel-button being pressed. Display conformation message. Create Equipment Object.	User (Session) Equipment

EditSeries	
Responsibilities	**Collaborators**
Display interface for editing series information for a sub-qualification. Respond to save-button being pressed. Respond to cancel-button being pressed. Display conformation message. Create Qualification Object.	User (Session) Qualification

EditQualType	
Responsibilities	**Collaborators**
Display interface for editing qualification type information for a sub-qualification. Respond to save-button being pressed. Respond to cancel-button being pressed. Display conformation message. Create Qualification Object.	User (Session) Qualification

EditSubQualType	
Responsibilities	**Collaborators**
Display interface for editing sub-qualification type information for a sub-qualification. Respond to save-button being pressed. Respond to cancel-button being pressed. Display conformation message. Create Qualification Object.	User (Session) Qualification

EditPartSubQualType	
Responsibilities	**Collaborators**
Display interface for editing part sub-qualification type information for a sub-qualification. Respond to save-button being pressed. Respond to cancel-button being pressed. Display conformation message. Create Qualification Object.	User (Session) Qualification

EditTypeList	
Responsibilities	**Collaborators**
Display interface for editing type list (system configuration) information. Respond to save-button being pressed. Respond to cancel-button being pressed. Display confirmation message.	ListOfValues

5.1.2 Business layer

User	
Responsibilities	**Collaborators**
Authenticates user. Retrieves user access level. Informs web forms of user access level Keeps track of: • User access level. • User name.	DbConnection Active Directory Login Web forms code-behind

Qualification	
Responsibilities	**Collaborators**
Can create new a qualification number in the system. Can edit a qualification. Keeps track of: • Qualification title. • Qualification number. • Qualification status. • Qualification subtype. • Qualification process steps. • Qualification material codes. • Qualification equipment • The qualifications remaining activities. • Qualification change control number . • Qualification documents. • Qualifications sub-qualifications and parents.	Document Equipment DbConnection Method ProcessStep NewQual AddSubQual EditSubQual QualEquip QualMethod QualProcessStep QualDoc CloseQual RemActivity EditQualUser SearchQual

Document	
Responsibilities	**Collaborators**
Can edit a document entry. Keeps track of: • Document ID. • Document title. • Document authors. • Document approval date. • Document archive location. • Document archive ID. • Document type.	Qualification DbConnection QualDoc

Equipment	
Responsibilities	**Collaborators**
Can edit an equipment entry. Keeps track of: • Equipment ID and version. • Equipment name. • The room the equipment can be found in. • The department the equipment belongs to. • Equipment instructions. • Equipment type.	DbConnection Instruction EditEquip QualEquip

DbConnection	
Responsibilities	**Collaborators**
Uses database stored procedures to add, edit, retrieve and delete items in the database, with the exception of items that are not allowed to be deleted, e.g. qualification numbers.	Database User Qualification Document Equipment Activity ListOfValues ProcessStep MaterialCode Method

Activity	
Responsibilities	**Collaborators**
Keeps track of: • The individuals responsible for the activity. • Activity completion date. • A description of the activity	DbConnection Qualification RemActivity CloseQual

ListOfValues	
Responsibilities	**Collaborators**
Provides methods for setting, getting, creating and deleting items used for lists of values in the system, such as rooms, departments and materials.	DbConnection Room EditProduct EditDepartment EditInstructionCategory EditEquipType EditSeries EditQualType EditSubQualType EditPartSubQualType EditTypeList

Room	
Responsibilities	**Collaborators**
Keeps track of • Room name • Room status	ListOfValues EditRoom

Instruction	
Responsibilities	**Collaborators**
Keeps track of: • Instruction category • instruction text	Equipment EditEquip

ProcessStep	
Responsibilities	**Collaborators**
Keeps track of the following components of a process step: • name • comment • requalification interval • productID • qualification condition	Qualification EditProcessStep QualProcessStep

Method	
Responsibilities	**Collaborators**
Keeps track of the following components of a qualification method: • name • comment • department • editionName • number • requalification interval	Qualification EditMethod QualMethod

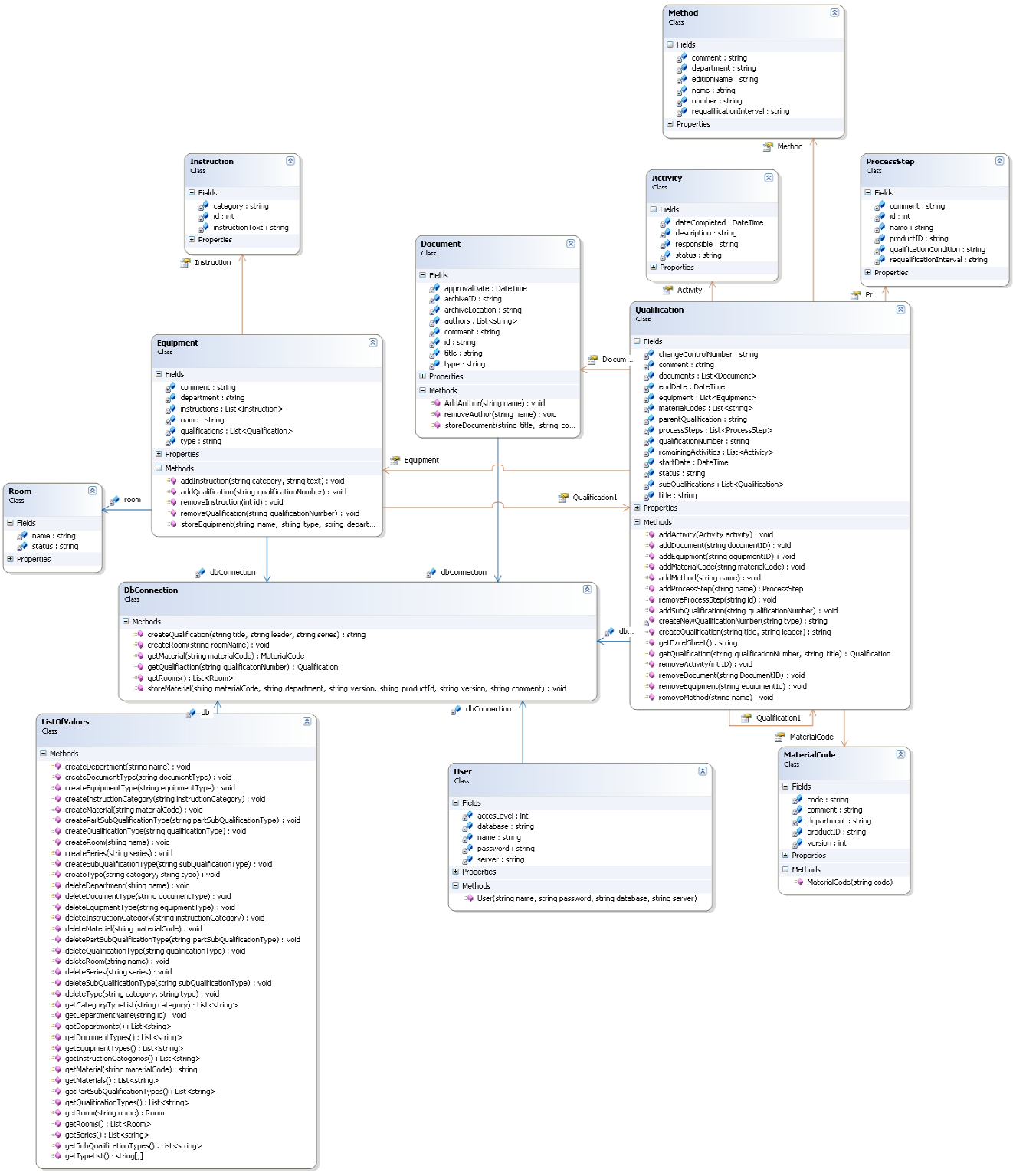
MaterialCode	
Responsibilities	**Collaborators**
Keeps track of the following components of a material code: • The material code • Comment • Department • Version • the product associated with the material • requalification interval • qualification condition	Qualification EditMtrlCode QualMtrlCode

5.1.3 Database layer

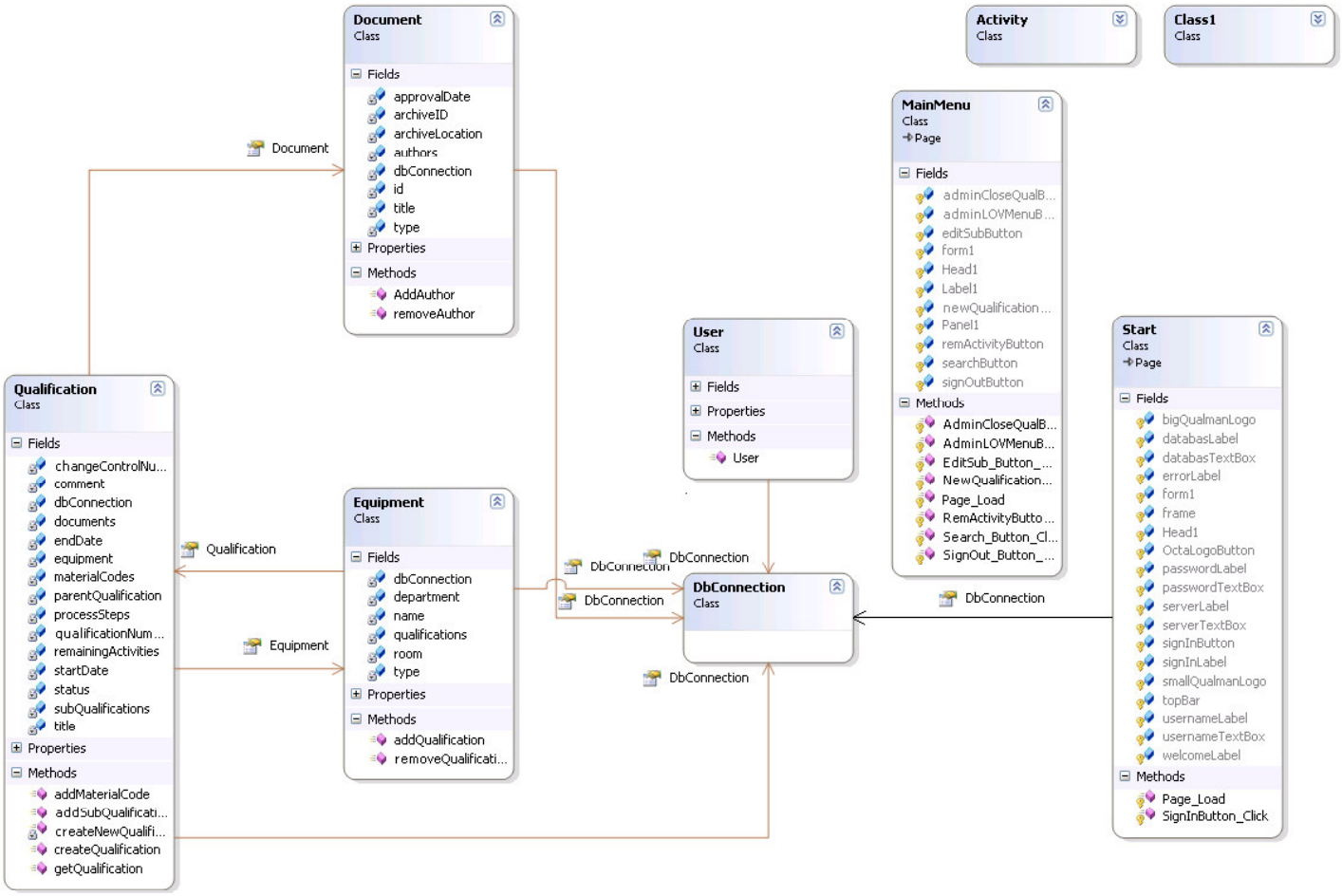
Database	
Responsibilities	**Collaborators**
Store data and procedures Execute stored procedures Return data on request	DbConnection class of business layer

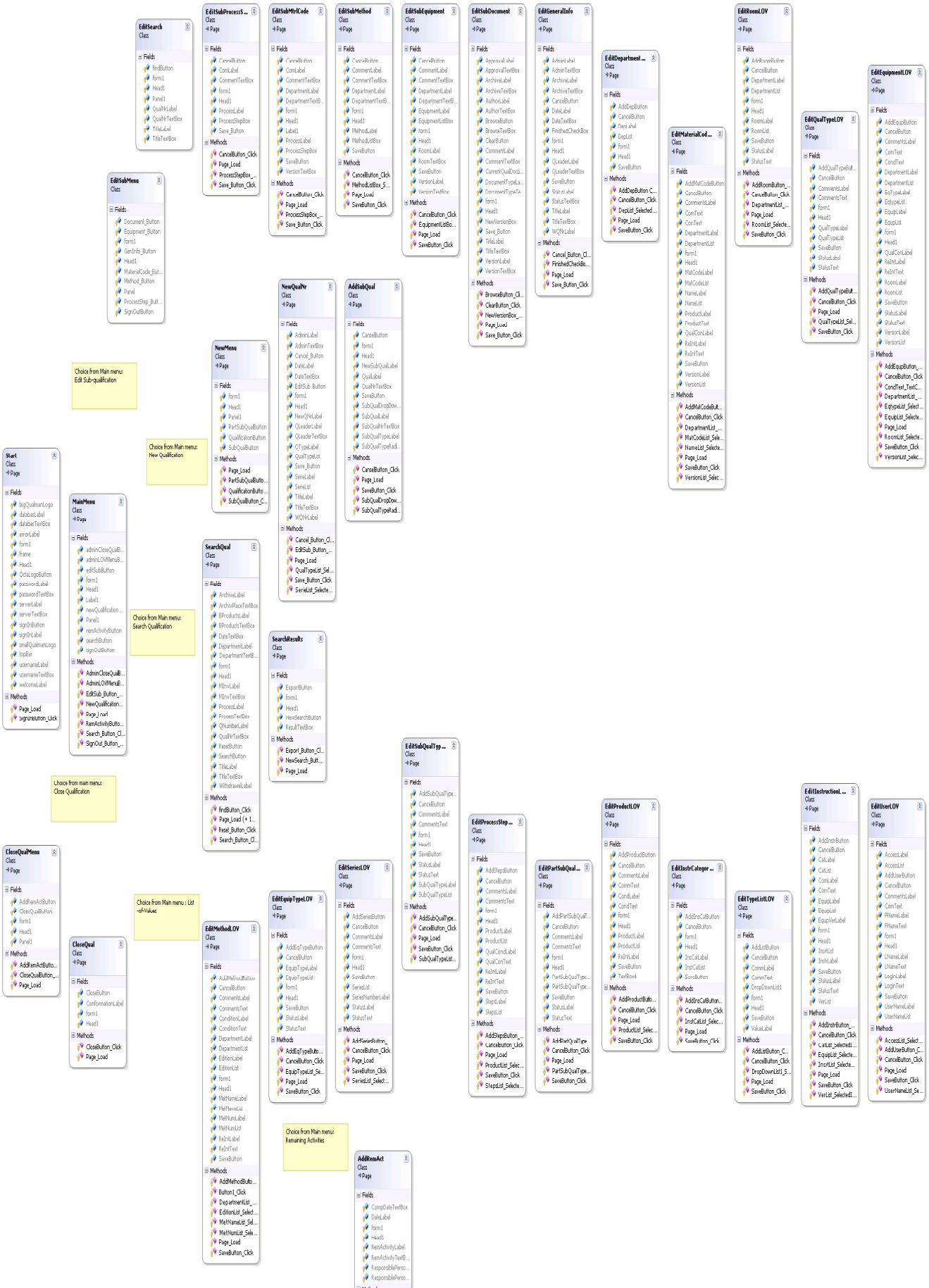
5.2 Class Diagram

Business layer



Presentation layer



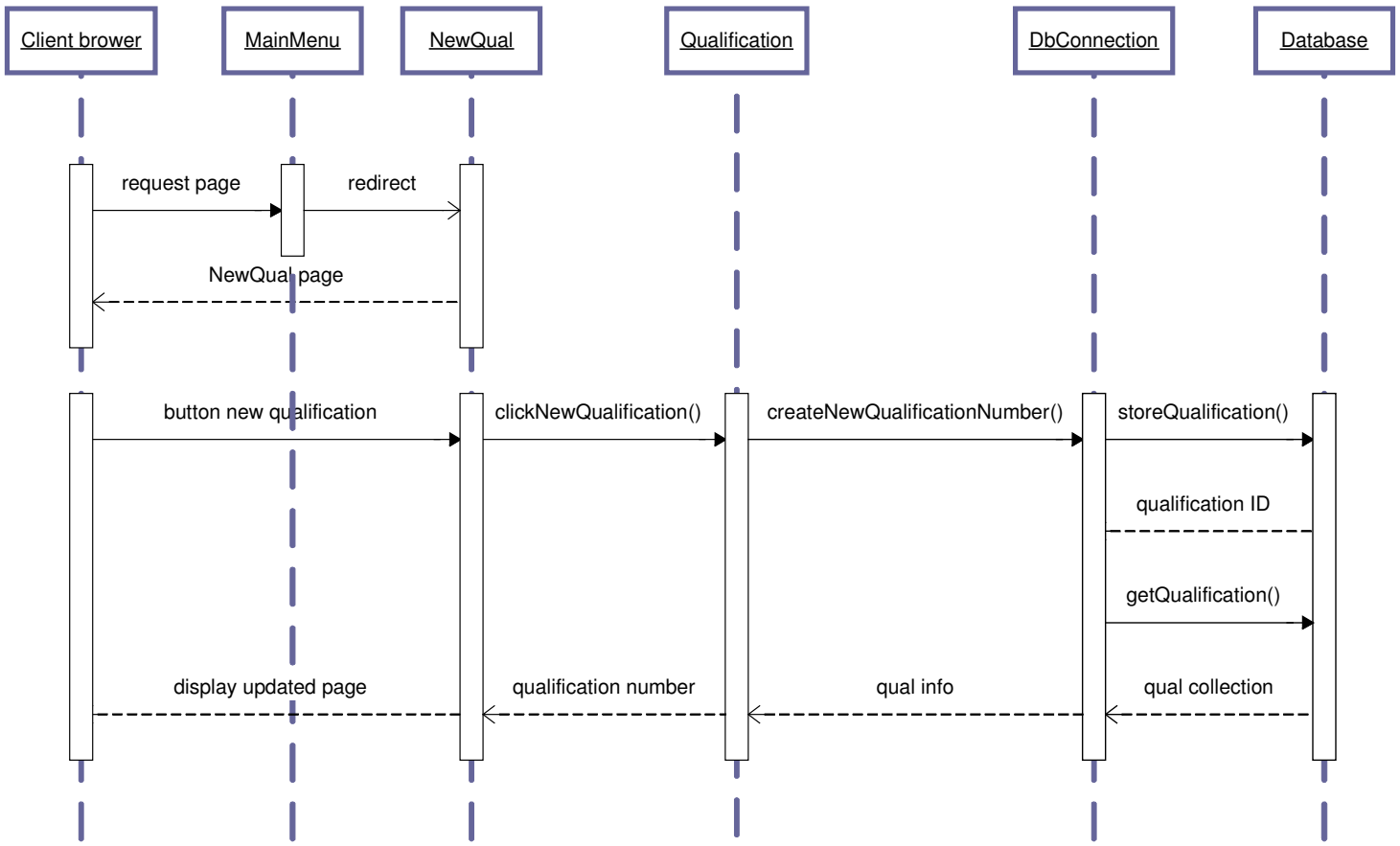

EditSearch
EditSubProcessS...
EditSubMtrlCode
EditSubMethod
EditSubEquipment
EditSubDocument
EditGeneralInfo
EditDepartment...
EditRoomLOV
EditMaterialCod...
EditQualTypeLOV
EditEquipmentLOV
EditSubMenu
NewMenu
NewQualNr
AddSubQual
Choice from Main menu: Edit Sub-qualification
Choice from Main menu: New Qualification
Start
MainMenu
SearchQual
Choice from Main menu: Search Qualification
SearchResults
Choice from main menu: Close Qualification
CloseQualMenu
CloseQual
Choice from Main menu : List -of-Value
EditMethodLOV
EditEquipTypeLOV
EditSeriesLOV
EditSubQualTyp...
EditProcessStep...
EditPartSubQual...
EditProductLOV
EditInstrCategor...
EditTypeListLOV
EditInstructionL...
EditUserLOV
Choice from Main menu: Remaining Activities
AddRemAct

5.3 State Charts

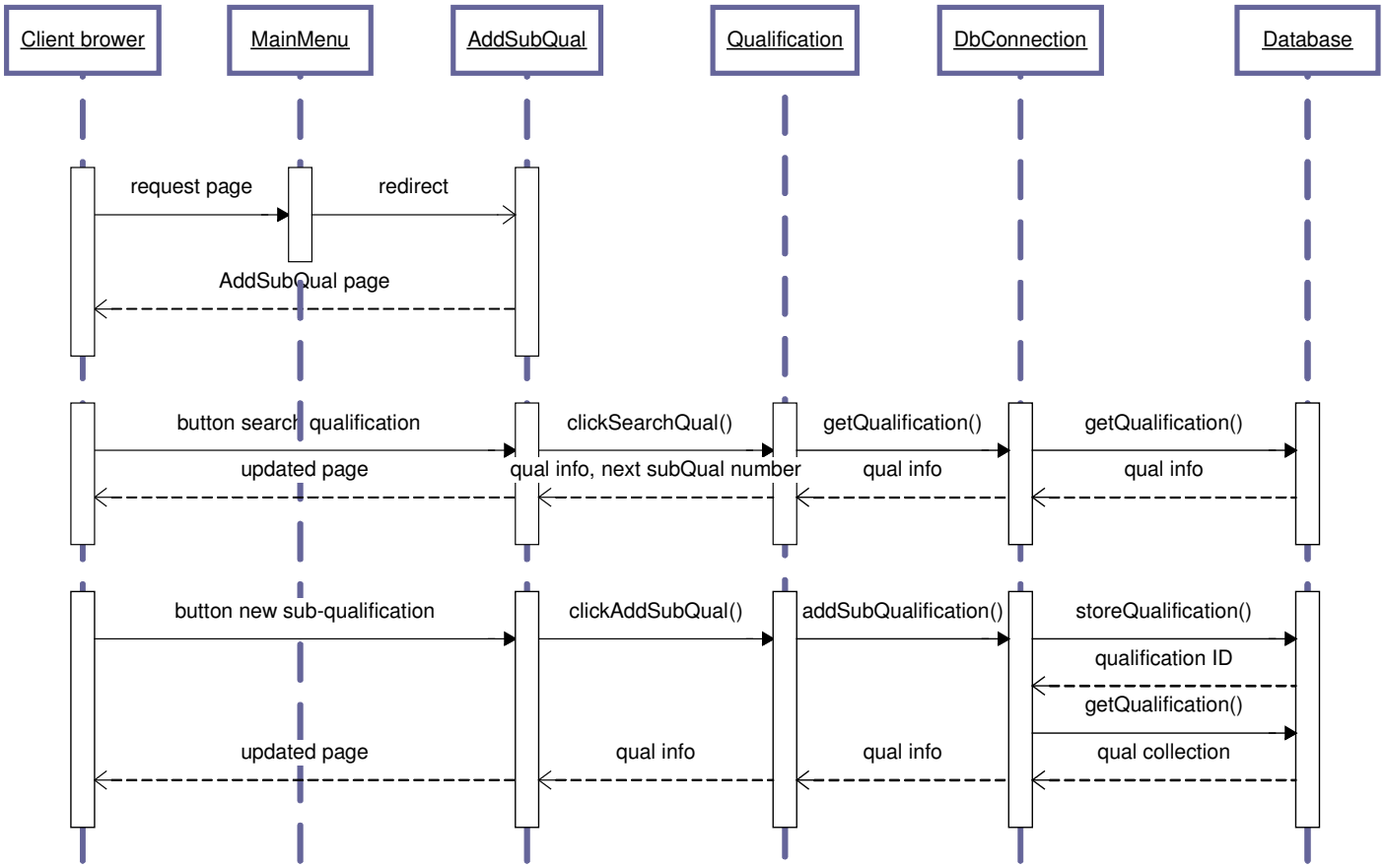
Not applicable.

5.4 Interaction Diagrams

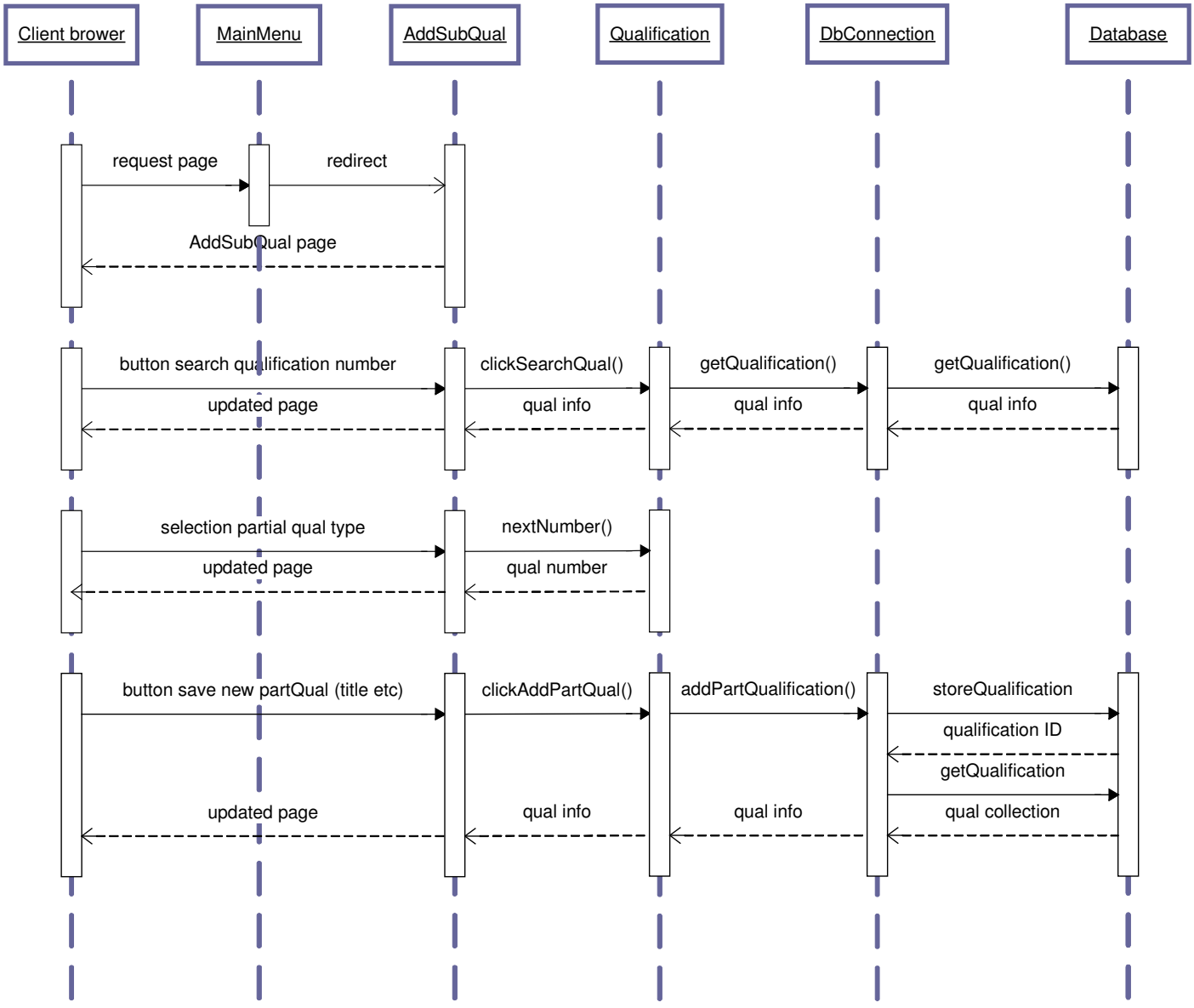
Use Case UC1: Add a new qualification number



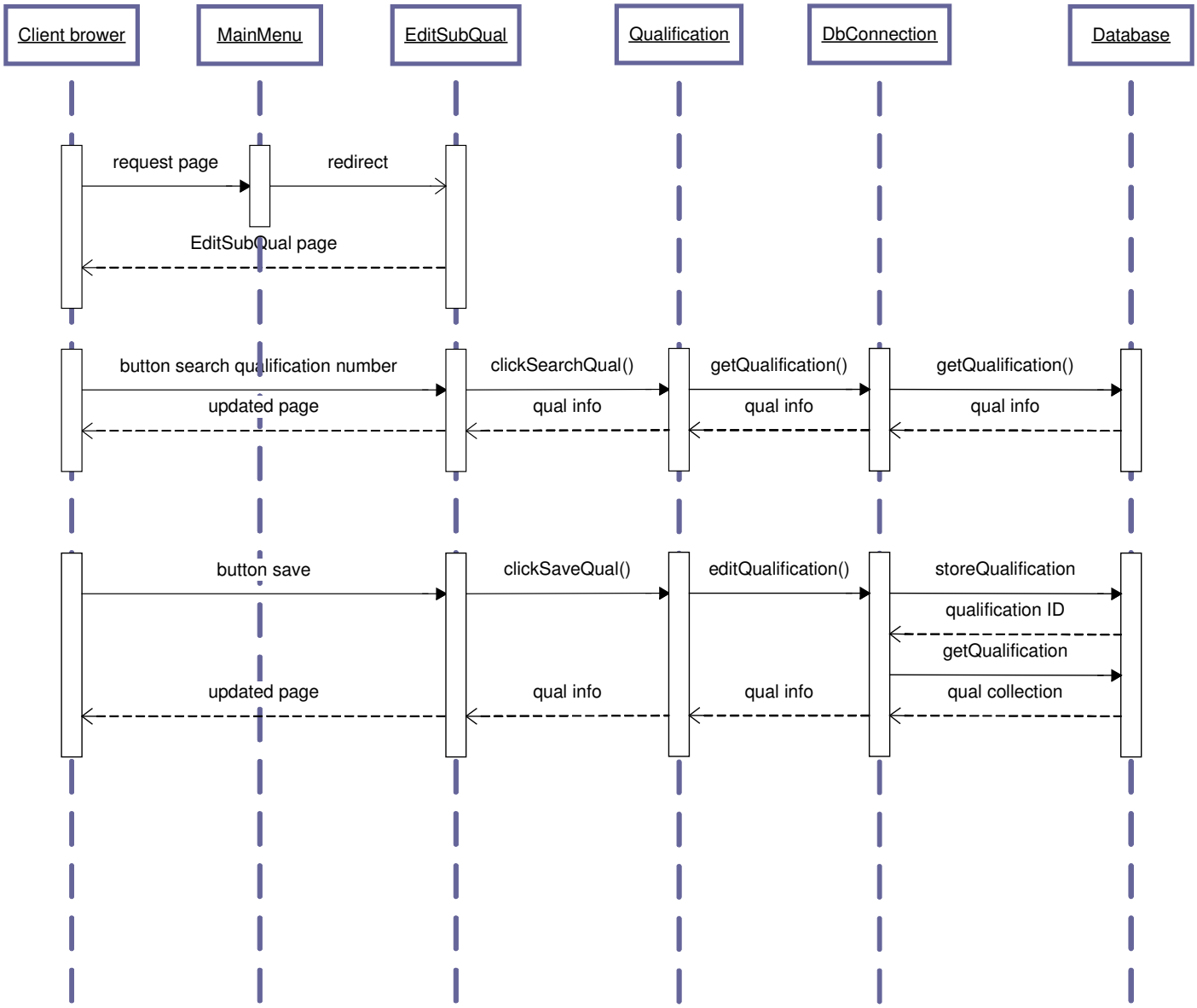
Use Case UC2: Add a sub-qualification to a qualification



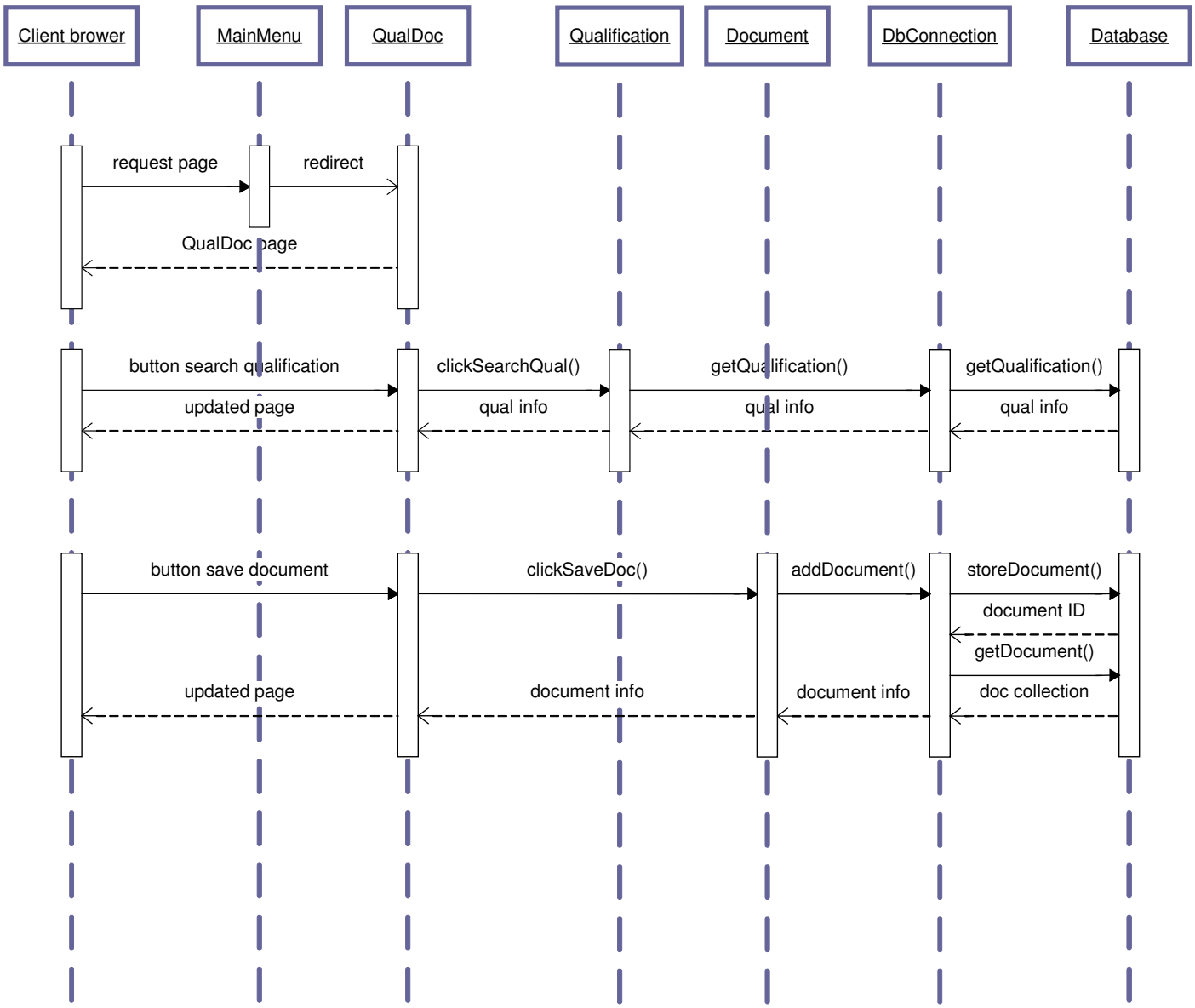
Use Case UC3: Add a partial qualification to a qualification number



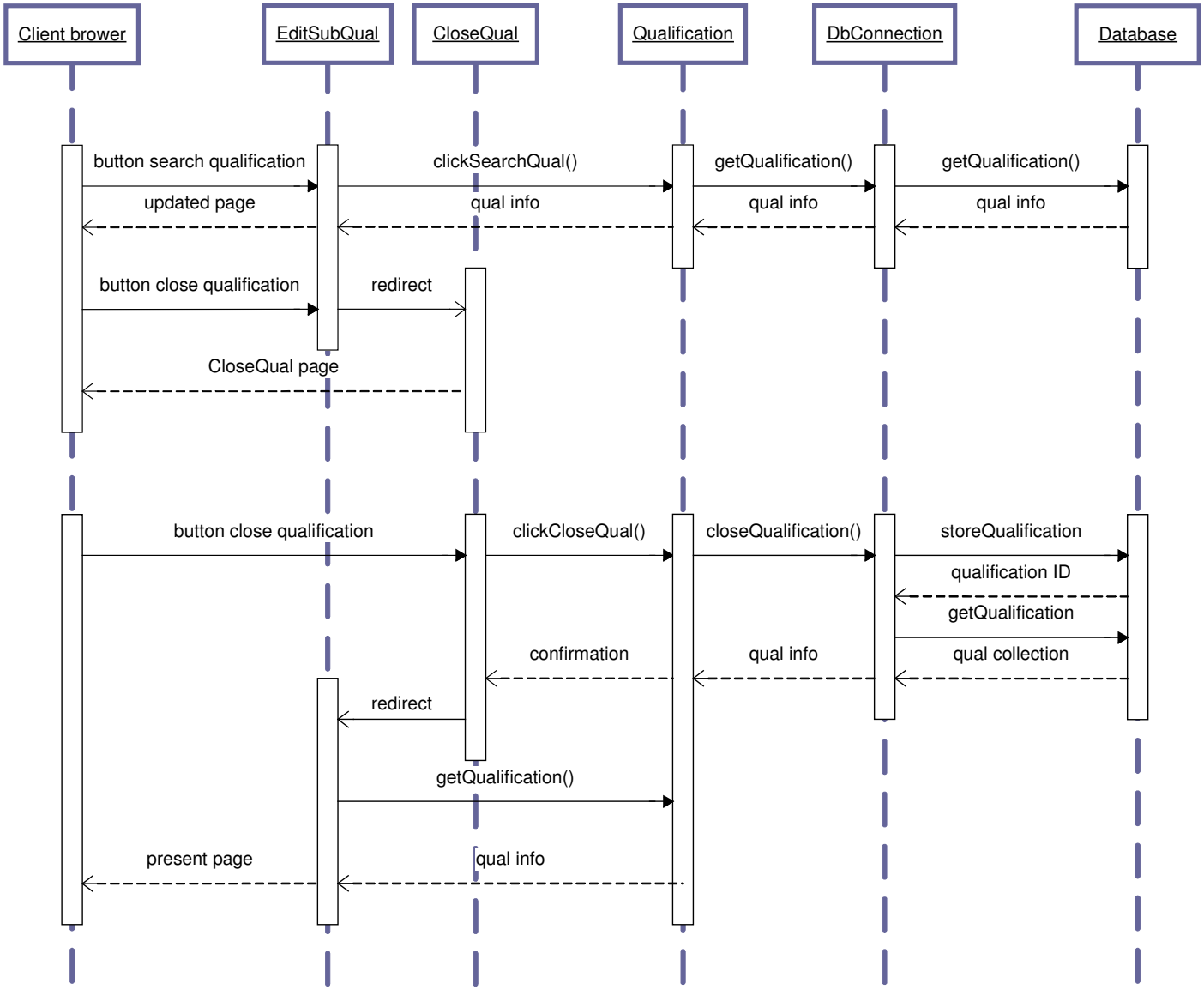
Use Case UC4 : Add or edit qualification information



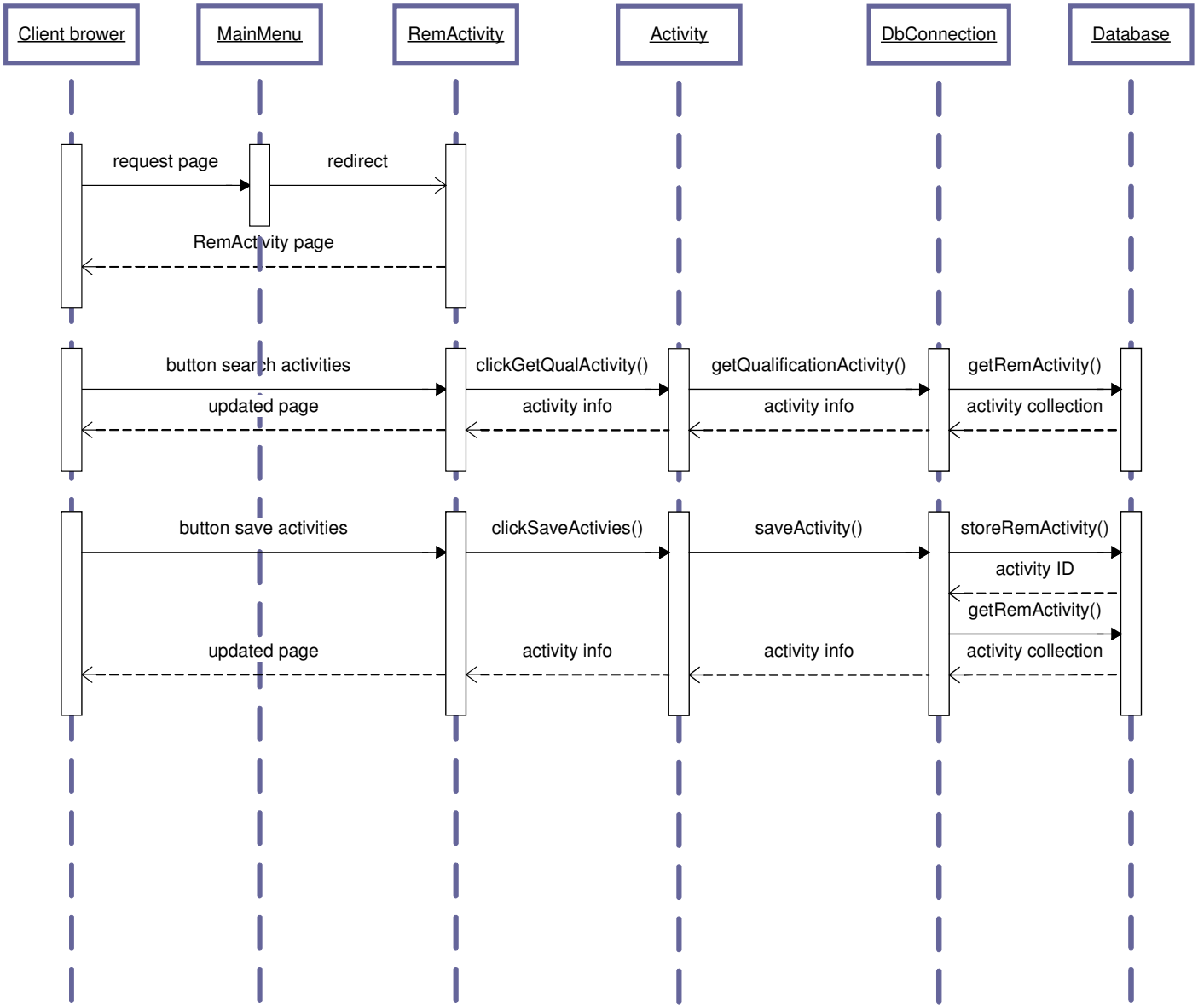
Use Case UC5: Add a new qualification document



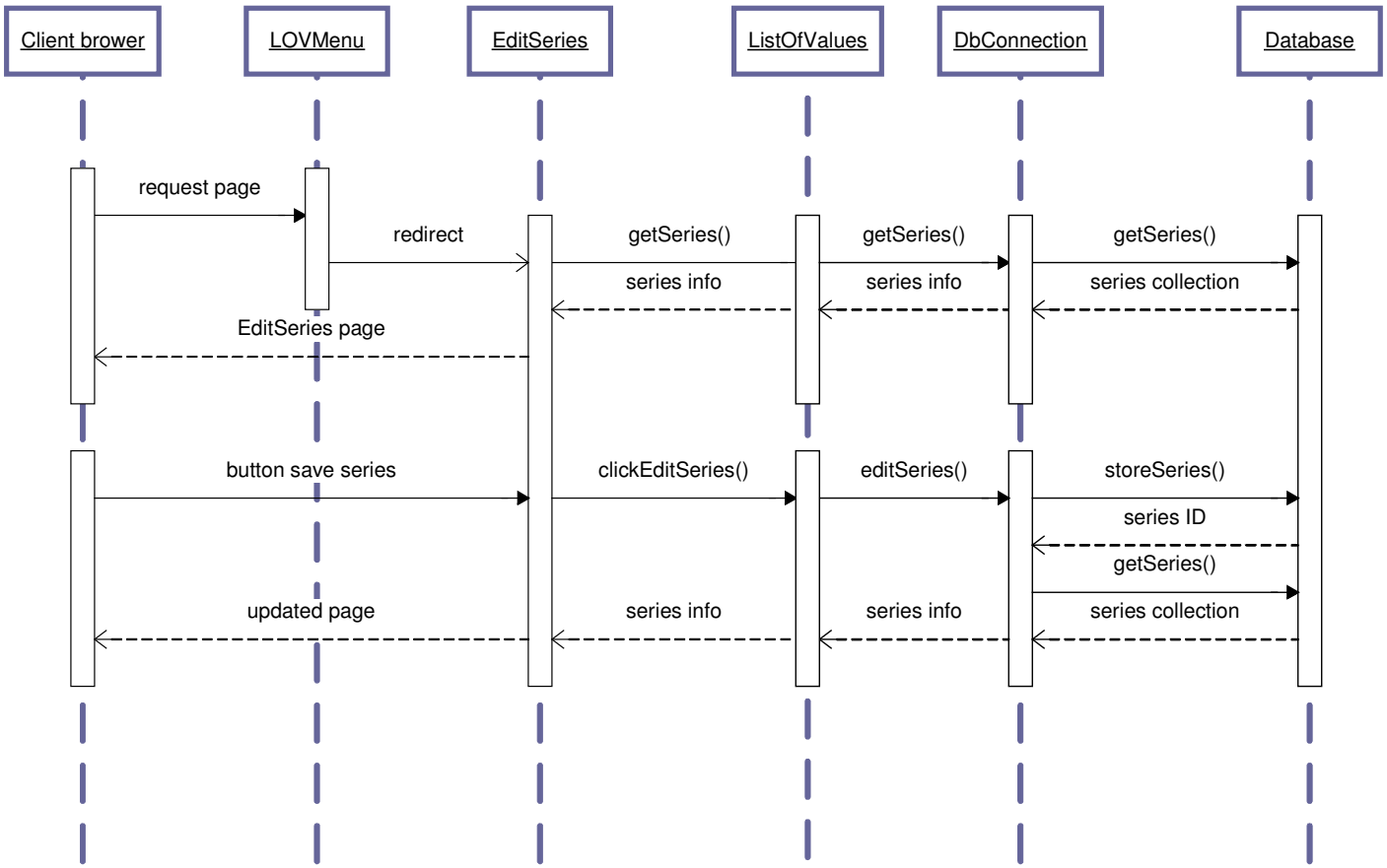
Use Case UC6: Close a sub-qualification



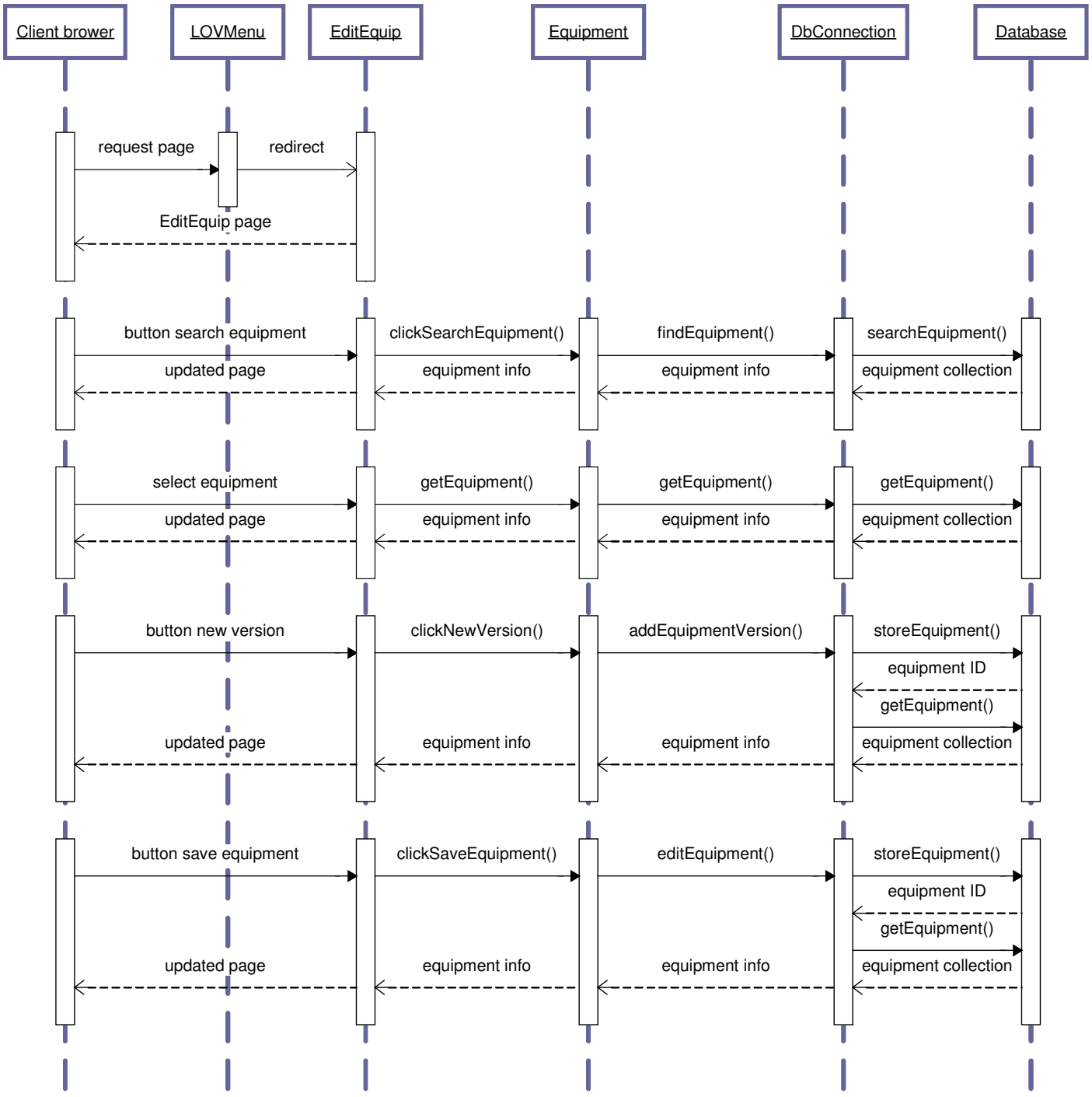
Use Case UC7: Edit remaining activities for a sub-qualification



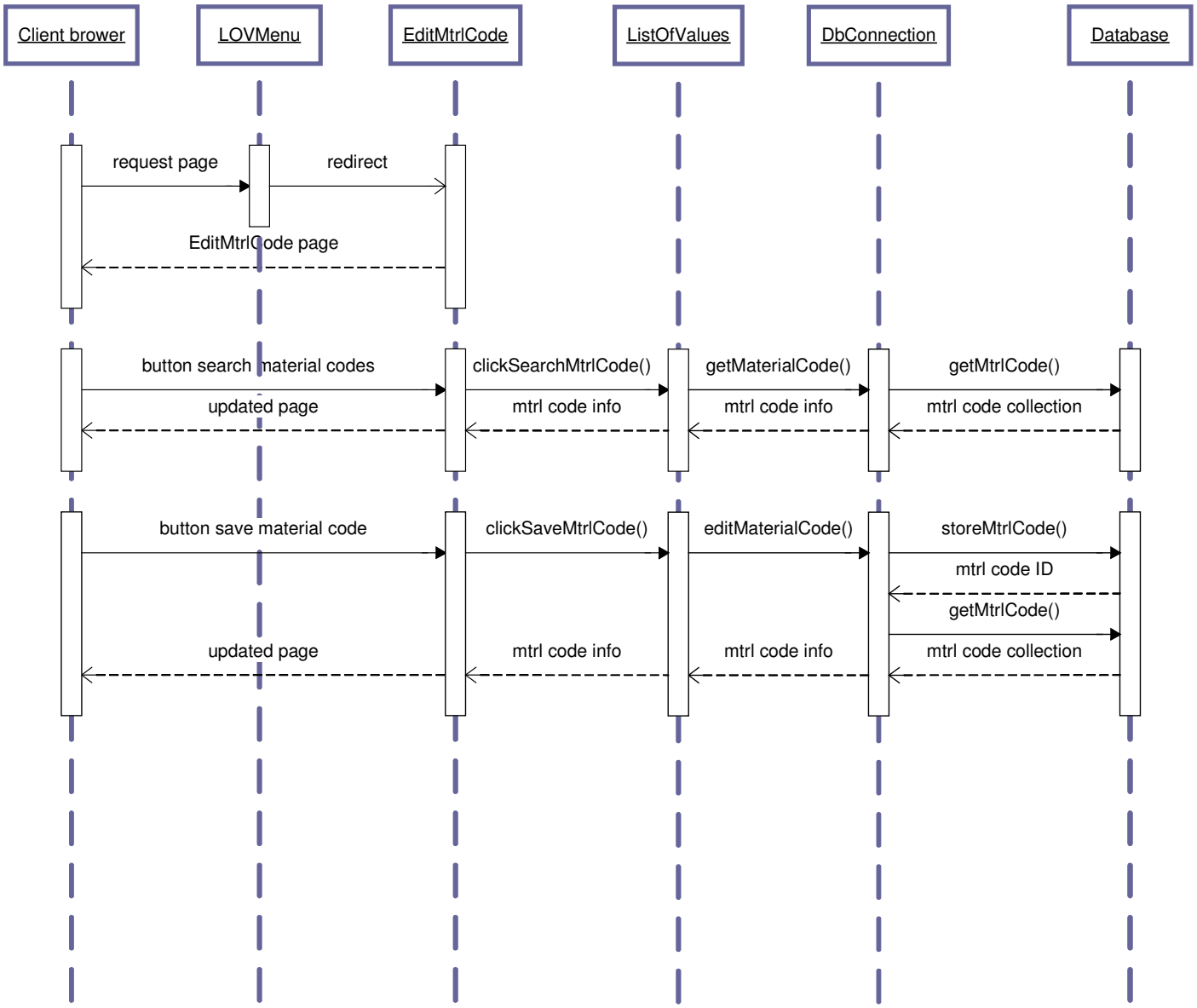
Use Case UC8: Edit List-of-Values information, e.g. qualification series



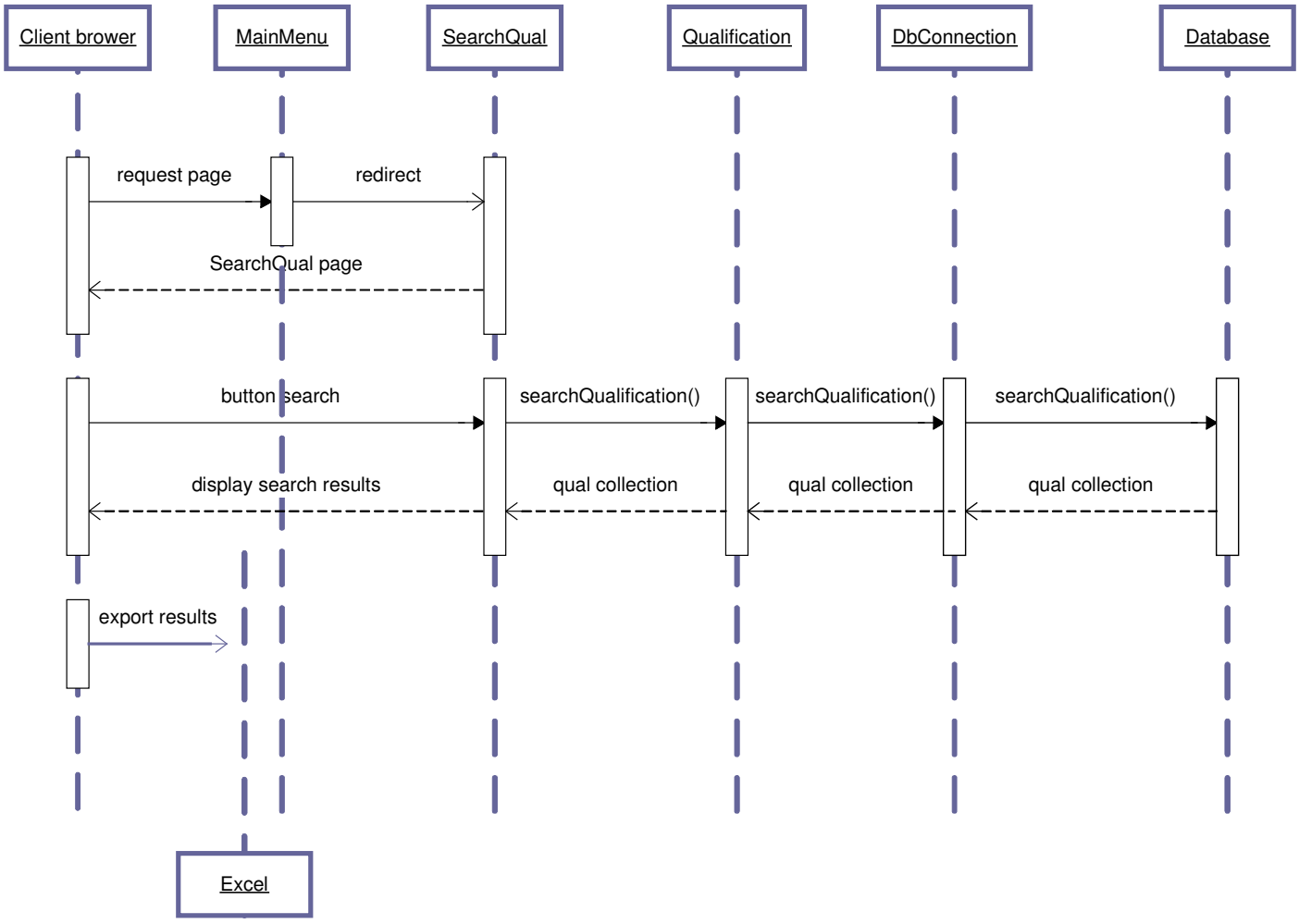
Use Case UC9: Edit equipment information



Use Case UC10: Edit material code information



Use Case UC11: Search for qualifications and export data to Excel



5.5 Detailed Design

5.5.1 Database

A data model, a list of database tables and a list of stored procedures can be found in appendix 1 ”Qualman database”.

5.5.2 Business Layer

For attributes and methods see class diagram in section 5.2. For method descriptions see below.

class Qualification:

```
/// <summary>
/// Adds a sub qualification the the qualification.
/// </summary>
/// <param name="qualificationNumber">The number of the subqualification to be added.</param>
public void addSubQualification(string qualificationNumber)
```

```
/// <summary>
/// Creates a new qualification.
/// </summary>
/// <param name="title">The title of the new qualification</param>
/// <param name="leader">The leader of the new qualification</param>
/// <returns>The qualification number for the new qualification</returns>
public static string createQualification(string title, string leader)
```

```
/// <summary>
/// Gets a qualification for editing.
/// </summary>
/// <param name="qualificationNumber">The qualification number of the qualification you want to get. Can be null.</param>
/// <param name="title">The qualification title of the qualification you want to get. Can be null.</param>
/// <returns>The requested qualification.</returns>
public static Qualification getQualification(string qualificationNumber, string title)
```

```
/// <summary>
/// Generates a new qualification number.
/// </summary>
/// <param name="type">The type of qualification number to be generated</param>
/// <returns>The new qualification number</returns>
private string createNewQualificationNumber(string type)
```

```
/// <summary>
/// Adds a material code to be associated with this qualification.
/// </summary>
/// <param name="materialCode">The material code to be associated with this qualification.</param>
public void addMaterialCode(String materialCode)
```

```
/// <summary>
/// Adds an association of a Document to be associated to this qualification
/// </summary>
/// <param name="documentID">The documentID of the document to be added.</param>
public void addDocument(string documentID)
```

```
/// <summary>
/// Removes the association of a document from the qualification.
/// </summary>
/// <param name="DocumentID">The documentID of the document to be removed.</param>
public void removeDocument(string DocumentID)
```

```
/// <summary>
/// Adds an association of a piece of equipment to the qualification.
/// </summary>
/// <param name="equipmentID">The eqipmentID of the eqipmen to be added.</param>
public void addEquipment(string equipmentID)
```

```
/// <summary>
/// Removes the association of an equipment item from the qualification.
/// </summary>
/// <param name="equipmentId">The id of the Equipment to be removed</param>
public void removeEquipment(string equipmentId)
```

```
/// <summary>
/// Gives the path of an Excel sheet for the qualifiaction.
/// </summary>
/// <returns>The path an Excel sheet for the qualifiaction.</returns>
public string getExcelSheet()
```

```
/// <summary>
/// Adds a activity too the the qualification.
/// </summary>
/// <param name="name">the activity to be added.</param>
public void addActivity(Activity activity)
```

```
/// <summary>
/// Removes an activty from the qualification
/// </summary>
/// <param name="ID">The id if the activity to be deleted.</param>
public void removeActivity(int ID)


/// <summary>
/// Adds a method too the the qualification.
/// </summary>
/// <param name="name">The name of the Method to be added.</param>
public void addMethod(string name)


/// <summary>
/// Removes a method from the the qualification.
/// </summary>
/// <param name="name">The name of the Method to be removed.</param>
public void removeMethod(string name)


/// <summary>
/// Adds a process step too the the qualification.
/// </summary>
/// <param name="name">The name of the process step to be added.</param>


/// <summary>
/// Removes a process step from the the qualification.
/// </summary>
/// <param name="id">The id of the process step to be removed.</param>
```

class Equipment

```
/// <summary>
/// Adds a qualification association with this equipment.
/// </summary>
/// <param name="qualificationNumber">The qualification to be
added.</param>
public void addQualification(string qualificationNumber)

/// <summary>
/// Removes a qualification association with this equipment.
/// </summary>
/// <param name="qualificationNumber">The qualification to be
removed.</param>
public void removeQualification(string qualificationNumber)
```

```
        /// <summary>
        /// Adds an instruction to the eqipment.
        /// </summary>
        /// <param name="category">The category of the instruction</param>
        /// <param name="text">The text in the instruction</param>
        public void addInstruction(string category, string text)

        /// <summary>
        /// Removes an instruction from the equipment.
        /// </summary>
        /// <param name="id">The id of the instruction to be removed</param>
        public void removeInstruction(int id)
```

class Document:

```
        /// <summary>
        /// Adds an author to the document.
        /// </summary>
        /// <param name="name">The name of the author to be added.</param>
        public void AddAuthor(string name)

        /// <summary>
        /// Removes the requested author.
        /// a string as parameter may *not* be ideal...
        /// </summary>
        /// <param name="name"></param>
        public void removeAuthor(string name)
```

class DbConnection

```
        /// <summary>
        /// Retrives the qualification of the associated qualification number.
        /// </summary>
        /// <param name="qualificatonNumber">The qualification number of the qualification to be
retrieved.</param>
        /// <returns>The requested qualification</returns>
        public Qualification getQualifiaction(string qualificatonNumber)


         /// <summary>
        /// Adds a qualification to the database.
        /// </summary>
        /// <param name="title">The title of the qualification to be added.</param>
        /// <param name="leader">The leader of the qualification to be added.</param>
        /// <param name="series">The series of the qualification to be added.</param>
        /// <returns>The qualification number of the added qualification.
        /// Null if failed.</returns>
        public string createQualification(string title, string leader, string series)
```

```
/// <summary>
/// Retreives the material data of the associated material code.
/// </summary>
/// <param name="materialCode">The material code of the material data to be retrieved</param>
/// <returns>The material code data</returns>
public MaterialCode getMaterial(string materialCode)


 /// <summary>
/// Creates a new room in the database.
/// </summary>
/// <param name="roomName">The name of the room to be added</param>
public void createRoom(string roomName)

 /// <summary>
/// Stores a new material code to the database.
/// Any parameter, except the material code, can be null.
/// Those fields which are null, won't be updated.

/// </summary>
/// <param name="materialCode">If the material code doesn't excist in the database,
/// the new material code will be added.</param>
/// <param name="department">The department associated with the material code</param>
/// <param name="productID">The productID associated with the material code</param>
/// <param name="version">The version of the material code</param>
/// <param name="comment">The comment associated with the material code</param>
public void storeMaterial(string materialCode, string department, string version, string productId, string
version, string comment)
```

class ListOfValues:

```
/// <summary>
/// Fetches a material code from the database.
/// </summary>
/// <param name="materialCode">The code of the material to be retrieved</param>
/// <returns>The material</returns>
public string getMaterial(string materialCode)


/// <summary>
/// Adds a material to the database.
/// </summary>
/// <param name="materialCode">The code of the material to be added</param>
public void createMaterial(string materialCode)


/// <summary>
```

```
/// Removes a material from the database.
/// This can only be done if there are no references to the material.
/// </summary>
/// <param name="materialCode">The code of the material to be removed.</param>
public void deleteMaterial(string materialCode)



/// <summary>
/// Adds a room to the database.
/// </summary>
/// <param name="name">The name of the room to be added</param>
public void createRoom(string name)



/// <summary>
/// Deletes a room to the database.
/// </summary>
/// <param name="name">The name of the room to be deleted</param>
public void deleteRoom(string name)



/// <summary>
/// Fetches a room from the datbase.
/// </summary>
/// <param name="name">The name of the room to be fetched</param>
/// <returns>A room</returns>
public Room getRoom(string name)



/// <summary>
/// Fetches a list of all the rooms in the database.
/// </summary>
/// <returns>A list of all the rooms in the database</returns>
public List<Room> getRooms()



/// <summary>
/// Fetches a list of all the materials in the database.
/// </summary>
/// <returns>A list of all the materials in the database</returns>
public List<string> getMaterials()

/// <summary>
/// Adds a department to the database.
```

```
/// </summary>
/// <param name="name">The name of the department to be added</param>
public void createDepartment(string name)



/// <summary>
/// Retrieves a department from the database.
/// </summary>
/// <param name="name">The id of the department to be fetched</param>
public void getDepartmentName(string id)



/// <summary>
/// Deletes a department for the database.
/// This can only be done if there are no references to the material.
/// </summary>
/// <param name="name">The name of the department to be deleted</param>
public void deleteDepartment(string name)



/// <summary>
/// Retrieves a list of all the separtments in the database.
/// </summary>
/// <returns>A list with all the departments in the database</returns>
public List<string> getDepartments()


/// <summary>
/// Adds a new series to the database
/// </summary>
/// <param name="series">The serie to be added to the database</param>
public void createSeries(string series)



/// <summary>
/// Removes a series from the database
/// This can only be done if the series is unused.
/// </summary>
/// <param name="series">The serie to be removed.</param>
public void deleteSeries(string series)



/// <summary>
/// Retrieves a list of all the series in the database.
```

```
/// </summary>
/// <returns>a list of all the series in the database</returns>
public List<String> getSeries()
```

```
/// <summary>
/// Retrieves a list of all the qualification types in the database.
/// </summary>
/// <returns>a list of all the qualification types in the database</returns>
public List<String> getQualificationTypes()
```

```
/// <summary>
/// Adds a new qualification type to the database
/// </summary>
/// <param name="qualificationType">The qualification type to be added to the database</param>
public void createQualificationType(string qualificationType)
```

```
/// <summary>
/// Removes a qualification type from the database
/// This can only be done if the qualification type is unused.
/// </summary>
/// <param name="qualificationType">The qualification type to be removed.</param>
public void deleteQualificationType(string qualificationType)
```

```
/// <summary>
/// Retrieves a list of all the sub-qualification types in the database.
/// </summary>
/// <returns>a list of all the sub-qualification types in the database</returns>
public List<String> getSubQualificationTypes()

/// <summary>
/// Adds a new sub-qualification type to the database
/// </summary>
/// <param name="subQualificationType">The sub-qualification type to be added to the database</param>
public void createSubQualificationType(string subQualificationType)
```

```
/// <summary>
/// Removes a sub-qualification type from the database
/// This can only be done if the sub-qualification type is unused.
/// </summary>
/// <param name="subQualificationType">The sub-qualification type to be removed.</param>
public void deleteSubQualificationType(string subQualificationType)
```

```
        /// <summary>
        /// Retrieves a list of all the partial sub-qualification types in the database.
        /// </summary>
        /// <returns>a list of all the partial sub-qualification types in the database</returns>
        public List<String> getPartSubQualificationTypes()



        /// <summary>
        /// Adds a new partial sub-qualification type to the database
        /// </summary>
        /// <param name="partSubQualificationType">The partial sub-qualification type to be added to the
database</param>
        public void createPartSubQualificationType(string partSubQualificationType)



        /// <summary>
        /// Removes a partial sub-qualification type from the database
        /// This can only be done if the partial sub-qualification type is unused.
        /// </summary>
        /// <param name="partSubQualificationType">The partial sub-qualification type to be removed.</param>
        public void deletePartSubQualificationType(string partSubQualificationType)



        /// <summary>
        /// Retrieves a list of all the equipment types in the database.
        /// </summary>
        /// <returns>a list of all the equipment types in the database</returns>
        public List<String> getEquipmentTypes()


        /// <summary>
        /// Adds a new equipment type to the database
        /// </summary>
        /// <param name="equipmentType">The equipment type to be added to the database</param>
        public void createEquipmentType(string equipmentType)



        /// <summary>
        /// Removes a equipment type from the database
        /// This can only be done if the equipment type is unused.
        /// </summary>
        /// <param name="equipmentType">The equipment type to be removed.</param>
        public void deleteEquipmentType(string equipmentType)
```

```
/// <summary>
/// Retrieves a list of all the instruction categories in the database.
/// </summary>
/// <returns>a list of all the instruction categories in the database</returns>
public List<String> getInstructionCategories()
```

```
/// <summary>
/// Adds a new instruction category to the database
/// </summary>
/// <param name="instructionCategory">The instruction category to be added to the database</param>
public void createInstructionCategory(string instructionCategory)
```

```
/// <summary>
/// Removes a instruction category from the database
/// This can only be done if the instruction category is unused.
/// </summary>
/// <param name="instructionCategory">The instruction category to be removed.</param>
public void deleteInstructionCategory(string instructionCategory)
```

```
/// <summary>
/// Retrieves a list of all the document types in the database.
/// </summary>
/// <returns>a list of all the document types in the database</returns>
public List<String> getDocumentTypes()
```

```
/// <summary>
/// Adds a new document type to the database
/// </summary>
/// <param name="document">The document type to be added to the database</param>
public void createDocumentType(string documentType)
```

```
/// <summary>
/// Removes a document type from the database
/// This can only be done if the document type is unused.
/// </summary>
/// <param name="document">The document type to be removed.</param>
public void deleteDocumentType(string documentType)
```

```
/// <summary>
/// Retrieves the type list from the database.
/// </summary>
/// <returns>a list of all the document types in the database</returns>
public String[,] getTypeList()


/// <summary>
/// Retrieves a type list from the database.
/// </summary>
/// <param name="category">The category of the types retrieved.</param>
/// <returns>a list of all the document types in the database.</returns>
public List<String> getCategoryTypeList(string category)


/// <summary>
/// Adds a new type to the database type list.
/// </summary>
/// <param name="category">The category of the type to be added to the database type list.</param>
/// <param name="type">The type to be added to the database type list.</param>
public void createType(string category, string type)


/// <summary>
/// Removes a type from the database type list.
/// This can only be done if the type is unused.
/// </summary>
/// <param name="category">The category of the type to be removed from the database type list.</param>
/// <param name="type">The type to be added to the database type list.</param>
public void deleteType(string category, string type)
```

5.5.3 Presentation Layer

EditEquipment

- Qualification: Qualification - Equipment: Equipment - LOV: ListOfValues - SaveButton: Button - CancelButton: Button - form1: HtmlForm - DepartmentLabel: Label - DepartmentTextBox: TextBox - RoomLabel: Label - RoomTextBox: TextBox - VersionLabel:Label - VersionTextBox:TextBox - CommentLabel:Label - CommentTextBox:TextBox - EquipmentLabel: Label - EquipList: ListBox

+ AddValues() + Page_Load((object sender, EventArgs e) + EquipList_SelectedIndexChanged(object sender, EventArgs e) + SaveButton_Click(object sender, EventArgs e) + CancelButton_Click(object sender, EventArgs e)
Functional requirements: 2.26, 2.28

EditMethod

- Qualification: Qualification - Method: Method - LOV: ListOfValues - CancelButton: Button - SaveButton:Button - Commentlabel: Label - CommentTextBox:TextBox - MethodList: DropDownList - DepartmentLabel: Label - DepartmentTextBox: TextBox - form1:HtmlForm - MethodLabel:Label - MethodListBox: ListBox
+ AddValues() + Page_Load((object sender, EventArgs e) + MethodListBox_SelectedIndexChanged(object sender, EventArgs e) + SaveButton_Click(object sender, EventArgs e) + CancelButton_Click(object sender, EventArgs e)
Functional requirements: 2.28

EditMtrlCode

- Qualification: Qualification - MtrlCode: MtrlCode - LOV: ListOfValues - MCodeProductLabel: Label - MCodeProductListBox: ListBox - form1: HtmlForm - DepartmentLabel:Label - DepartmentTextBox: TextBox - ComLabel:Label - ComTextBox: TextBox - Save_Button: Button - Cancel_Button: Button

+ AddValues() + Page_Load((object sender, EventArgs e) + SaveButton_Click(object sender, EventArgs e) + CancelButton_Click(object sender, EventArgs e) + MCodeProductList_SelectedIndexChanged(object sender, EventArgs e)
Functional requirements: 2.28

EditProcessStep

- Qualification: Qualification - ProcessStep: ProcessStep - LOV: ListOfValues - ProcessLabel: Label - ProcessStepList: ListBox - ComLabel:Label - ComTextBox: TextBox - SaveButton: Button - CancelButton:Button - form1:HtmlForm
+ AddValues() + Page_Load((object sender, EventArgs e) + SaveButton_Click(object sender, EventArgs e) + CancelButton_Click(object sender, EventArgs e) + ProcessStepList_SelectedIndexChanged(object sender, EventArgs e)
Functional requirements: 2.28

SearchQual

- Qualification: Qualification - QNumberLabel: Label - QNumberTTextBox: TextBox - TitleLabel: Label - TitleTextBox: TextBox - ArchivLabel: Label - ArchiveTextBox: TextBox - WithdrawelLabel: Label - WithdrawelDateBox: TextBox - Department: Label - DepartmentTextBox:TextBox - MInvLabel: Label - MInvTextBox: TextBox - ProcessLabel: Label - ProcessTextBox: TextBox

- BProductsLabel: Label - BProductsTextBox: TextBox - SearchButton: Button - ResetButton: Button
+ Page_Load(object sender, EventArgs e) + getQualification(string QNumber, string Title, string Archive, string Wdate, string Dep, string MInv, Products, string PrSteps) + SearchButton_Click(object sender, EventArgs e) + ResetButton_Load(object sender, EventArgs e)
Functional requirements: 2.33, 2.34, 2.35, 2.36

EditCommentsLOV

- LOV: ListOfValues - AddCommButton: RadioButton - CancelButton:Button - form1: HtmlForm - SaveButton:Button - StatusLabel:Label - StatusText:Text - Text : TextBox - TextLabel: Label - VersionLabel:Label - VersionList:DropDownList
+ AddValues() + Page_Load((object sender, EventArgs e + AddCommButton_CheckedChanged(object sender, EventArgs e) + VersionList_SelectedIndexChanged(object sender, EventArgs e) + CancelButton_Click(object sender, EventArgs e) + SaveButton_Click(object sender, EventArgs e)
Functional requirements: 2.32

EditDepartmentLOV

- LOV: ListOfValues - AddDepButton: RadioButton - CancelButton:Button - form1: HtmlForm - SaveButton:Button - DepLabel:Label - DepList:DropDownList

+ AddValues() + Page_Load((object sender, EventArgs e + AddDepButton_CheckedChanged(object sender, EventArgs e) + DepList_SelectedIndexChanged(object sender, EventArgs e) + CancelButton_Click(object sender, EventArgs e) + SaveButton_Click(object sender, EventArgs e)
Functional requirements: 2.30

EditEquipmentLOV

- LOV: ListOfValues - AddEqupButton: RadioButton - CancelButton:Button - form1: HtmlForm - SaveButton:Button - StatusLabel:Label - StatusText:Text - QualConLabel:Label - ConText :TextBox - CommentsLabel:Label - ComText:TextBox - CondLabel:Label - CondText: TextBox - CondLabel:Label - ReIntText : TextBox - ReIntLabel: Label - EquipLabel:Label - EquipList:DropDownList - EqTypeLabel:Label - EqTypeList: DropDownList - VersionLabel:Label - VersionList: DropDownList - RoomLabel:Label - RoomList: DropDownList - DepartmentLabel:Label - DepartmentList: DropDownList
+ AddValues() + Page_Load((object sender, EventArgs e + AddEquipButton_CheckedChanged(object sender, EventArgs e) + VersionList_SelectedIndexChanged(object sender, EventArgs e) + DepartmentList_SelectedIndexChanged(object sender, EventArgs e) + RoomList_SelectedIndexChanged(object sender, EventArgs e) + EqTypeList_SelectedIndexChanged(object sender, EventArgs e) + EquipList_SelectedIndexChanged(object sender, EventArgs e) + Cancel_Click(object sender, EventArgs e) + SaveButton_Click(object sender, EventArgs e)
Functional requirements: 2.24

EditInstructionLOV

- LOV: ListOfValues - AddInstrButton: RadioButton - CancelButton:Button - form1: HtmlForm - SaveButton:Button - StatusLabel:Label - StatusText:Text - ComLabel:Label - ComText:TextBox - CatLabel:Label - CatList:DropDownList - EquipLabel:Label - EquipList: DropDownList - EquipVerLabel:Label - VerList: DropDownList - InstrLabel:Label - InstrList: DropDownList
+ AddValues() + Page_Load((object sender, EventArgs e + AddInstrButton_CheckedChanged(object sender, EventArgs e) + VerList_SelectedIndexChanged(object sender, EventArgs e) + InstrList_SelectedIndexChanged(object sender, EventArgs e) + EquipList_SelectedIndexChanged(object sender, EventArgs e) + CatList_SelectedIndexChanged(object sender, EventArgs e) + Cancel_Click(object sender, EventArgs e) + SaveButton_Click(object sender, EventArgs e)
Functional requirements: 2.24

EditEquipTypeLOV

- LOV: ListOfValues - AddEqTypeButton: RadioButton - CancelButton:Button - form1: HtmlForm - SaveButton:Button - StatusLabel:Label - StatusText:Text - EquipTypeLabel:Label - EquipTypeList: DropDownList
+ AddValues() + Page_Load((object sender, EventArgs e + AddEqTypeButton_CheckedChanged(object sender, EventArgs e) + EquipTypeList_SelectedIndexChanged(object sender, EventArgs e) + Cancel_Click(object sender, EventArgs e) + SaveButton_Click(object sender, EventArgs e)
Functional requirements: 2.24

EditInstrCategoryLOV

- LOV: ListOfValues - AddInsCatButton: RadioButton - CancelButton:Button - form1: HtmlForm - SaveButton:Button - InsCatLabel:Label - InstCatList: DropDownList
+ AddValues() + Page_Load((object sender, EventArgs e + AddInsCatButton_CheckedChanged(object sender, EventArgs e) + InstCatList_SelectedIndexChanged(object sender, EventArgs e) + Cancel_Click(object sender, EventArgs e) + SaveButton_Click(object sender, EventArgs e)
Functional requirements: 2.24

EditMaterialCodeLOV

- LOV: ListOfValues - AddMatCodeButton: RadioButton - CancelButton:Button - form1: HtmlForm - SaveButton:Button - ProductLabel:Label - ProductText:Text - QualConLabel:Label - ConText: TextBOx - CommentsLabel - ComText:TextBox - MatCodeLabel:Label - MatCodeList:DropDownList - NameLabel:Label - NameList: DropDownList - VersionLabel:Label - VersionList: DropDownList - DepartmentLabel:Label - DepartmentList: DropDownList
+ AddValues() + Page_Load((object sender, EventArgs e + AddMatCodeButton_CheckedChanged(object sender, EventArgs e) + VersionList_SelectedIndexChanged(object sender, EventArgs e) + DepartmentList_SelectedIndexChanged(object sender, EventArgs e) + NameList_SelectedIndexChanged(object sender, EventArgs e) + MatCodeList_SelectedIndexChanged(object sender, EventArgs e) + Cancel_Click(object sender, EventArgs e) + SaveButton_Click(object sender, EventArgs e)

Functional requirements:

EditMethodLOV

- LOV: ListOfValues - AddMethodButton: RadioButton - CancelButton:Button - form1: HtmlForm - SaveButton:Button - CommentsLabel - CommentsText:TextBox - ConditionLabel:Label - ConditionText: TextBox - ReIntText : TextBox - ReIntLabel: Label - MetNumLabel:Label - MetNumList:DropDownList - EditionLabel:Label - EditionList: DropDownList - MetNameLabel:Label - MetNameList: DropDownList - DepartmentLabel:Label - DepartmentList: DropDownList
+ AddValues() + Page_Load((object sender, EventArgs e + AddMethodButton_CheckedChanged(object sender, EventArgs e) + MetNameList_SelectedIndexChanged(object sender, EventArgs e) + DepartmentList_SelectedIndexChanged(object sender, EventArgs e) + MetNumList_SelectedIndexChanged(object sender, EventArgs e) + EditionList_SelectedIndexChanged(object sender, EventArgs e) + Cancel_Click(object sender, EventArgs e) + SaveButton_Click(object sender, EventArgs e)
Functional requirements: 2.27

EditPartSubQualTypeLOV

- LOV: ListOfValues
- AddPartSubQualTypeButton: RadioButton
- CancelButton:Button
- form1: HtmlForm
- SaveButton:Button
- StatusLabel:Label
- StatusText:Text
- CommentsLabel
- CommentsText:TextBox
- PartSubQualTypeLabel:Label
- PartSubQualTypeList:DropDownList

+ AddValues()
+ Page_Load((object sender, EventArgs e
+ AddPartSubQualTypeButton_CheckedChanged(object sender, EventArgs e)
+ ParSubQualTypeList_SelectedIndexChanged(object sender, EventArgs e)
+ Cancel_Click(object sender, EventArgs e)
+ SaveButton_Click(object sender, EventArgs e)

Functional requirements: 2.10

EditProcessStepLOV

- LOV: ListOfValues
- AddStepsButton: RadioButton
- CancelButton:Button
- form1: HtmlForm
- SaveButton:Button
- QualCondLabel:Label
- QualConText:TextBox
- CommentsLabel
- CommentsText:TextBox
- CondText: TextBox
- ReIntText : TextBox
- ReIntLabel: Label
- StepsLabel:Label
- StepsList:DropDownList
- ProductLabel:Label
- ProductList: DropDownList

+ AddValues()

+ Page_Load((object sender, EventArgs e + AddStepsButton_CheckedChanged(object sender, EventArgs e) + ProductList_SelectedIndexChanged(object sender, EventArgs e) + StepsList_SelectedIndexChanged(object sender, EventArgs e) + Cancel_Click(object sender, EventArgs e) + SaveButton_Click(object sender, EventArgs e)
Functional requirements: 2.23

EditProductLOV

- LOV: ListOfValues - AddProductButton: RadioButton - CancelButton:Button - form1: HtmlForm - SaveButton:Button - CommentsLabel - CommText:TextBox - CondText: TextBox - CondLabel: Label - ReIntText : TextBox - ReIntLabel: Label - ProductLabel:Label - ProductList:DropDownList
+ AddValues() + Page_Load((object sender, EventArgs e + AddProductButton_CheckedChanged(object sender, EventArgs e) + ProductList_SelectedIndexChanged(object sender, EventArgs e) + Cancel_Click(object sender, EventArgs e) + SaveButton_Click(object sender, EventArgs e)
Functional requirements: 2.22, 2.25

EditQualTypeLOV

- LOV: ListOfValues - AddQualTypeButton: RadioButton - CancelButton:Button - form1: HtmlForm - SaveButton:Button - StatusLabel:Label - StatusText:Text - CommentsLabel - CommentsText:TextBox - QualTypeLabel:Label - QualTypeList:DropDownList

+ AddValues() + Page_Load((object sender, EventArgs e + AddQualTypeButton_CheckedChanged(object sender, EventArgs e) + QualTypeList_SelectedIndexChanged(object sender, EventArgs e) + Cancel_Click(object sender, EventArgs e) + SaveButton_Click(object sender, EventArgs e)
Functional requirements: 2.2

EditRoomLOV

- LOV: ListOfValues - AddRoomButton: RadioButton - CancelButton:Button - form1: HtmlForm - SaveButton:Button - StatusLabel:Label - StatusText:Text - RoomLabel:Label - RoomList:DropDownList - DepartmentLabel:Label - DepartmentList:DropDownList
+ AddValues() + Page_Load((object sender, EventArgs e + AddRoomButton_CheckedChanged(object sender, EventArgs e) + RoomList_SelectedIndexChanged(object sender, EventArgs e) + DepartmentList_SelectedIndexChanged(object sender, EventArgs e) + Cancel_Click(object sender, EventArgs e) + SaveButton_Click(object sender, EventArgs e)
Functional requirements: 2.30

EditSeriesLOV

- LOV: ListOfValues - AddSeriesButton: RadioButton - CancelButton:Button - form1: HtmlForm - SaveButton:Button - StatusLabel:Label - StatusText:Text - CommentsLabel:Label - CommentsText; TextBox - SeriesLabel:Label - SeriesList:DropDownList
+ AddValues() + Page_Load((object sender, EventArgs e + AddSeriesButton_CheckedChanged(object sender, EventArgs e)

+ SeriesList_SelectedIndexChanged(object sender, EventArgs e) + Cancel_Click(object sender, EventArgs e) + SaveButton_Click(object sender, EventArgs e)
Functional requirements: 2.31

EditSubQualTypeLOV

- LOV: ListOfValues - AddSubQualTypeButton: RadioButton - CancelButton:Button - form1: HtmlForm - SaveButton:Button - StatusLabel:Label - StatusText:Text - CommentsLabel - CommentsText:TextBox - SubQualTypeLabel:Label - SubQualTypeList:DropDownList
+ AddValues() + Page_Load((object sender, EventArgs e + AddSubQualTypeButton_CheckedChanged(object sender, EventArgs e) + SubQualTypeList_SelectedIndexChanged(object sender, EventArgs e) + Cancel_Click(object sender, EventArgs e) + SaveButton_Click(object sender, EventArgs e)
Functional requirements: 2.7

EditTypeListLOV

- LOV: ListOfValues - AddListButton: RadioButton - CancelButton:Button - form1: HtmlForm - SaveButton:Button - CommentsLabel - CommentsText:TextBox - ValueLabel:Label - ValueList:DropDownList
+ AddValues() + Page_Load((object sender, EventArgs e + AddListButton_CheckedChanged(object sender, EventArgs e) + ValueList_SelectedIndexChanged(object sender, EventArgs e) + Cancel_Click(object sender, EventArgs e) + SaveButton_Click(object sender, EventArgs e)
Functional requirements:2.38

EditUserLOV

- LOV: ListOfValues - AddUserButton: RadioButton - CancelButton:Button - form1: HtmlForm - SaveButton:Button - FNameLabel:Label - FNameText:Text - LNameLabel:Label - LNameText:TextBox - Loginlabel:Label - LoginText:TextBox - CommentsLabel - CommentsText:TextBox - UserNameLabel:Label - UserNameList:DropDownList - AccessLabel:Label - AccessList:DropDownList
+ AddValues() + Page_Load((object sender, EventArgs e + AddUserButton_CheckedChanged(object sender, EventArgs e) + UserNameList_SelectedIndexChanged(object sender, EventArgs e) + AccessList_SelectedIndexChanged(object sender, EventArgs e) + Cancel_Click(object sender, EventArgs e) + SaveButton_Click(object sender, EventArgs e)
Functional requirements:2.39, 2.39, 2.40

Login.aspx.cs

```
/// <summary> Authenticate user, create a user session. </summary>
            /// <param name="usernameTextBox">username</param>
            /// <param name="passwordTextBox">password</param>
            /// <param name="serverTextBox">the server</param>
            /// <param name="databaseTextBox">the database </param>///
            protected void SignInButton_Click(object sender, EventArgs e)
```

MainMenu.aspx.cs

```
/// <summary> Signs out the user</summary>
        protected void SignOut_Button_Click(object sender, EventArgs e)

/// <summary> Redirects user to Search page </summary>
        protected void Search_Button_Click(object sender, EventArgs e)

/// <summary> Redirects user to EditSubMenu page </summary>
        protected void EditSub_Button_Click(object sender, EventArgs e)

/// <summary> Redirects user to NewQualification page </summary>
        protected void NewQualification_Button_Click(object sender, EventArgs e)
```

```
/// <summary> Checks user rights </summary>
        protected void checkUser()

/// <summary> Sets admin menu visible </summary>
        protected void showAdminMenu()

/// <summary> Redirects user to LOVMenu </summary>
        protected void AdminLOVMenuButton_Click(object sender, EventArgs e)

/// <summary> Redirects user to CloseQual page </summary>
        protected void AdminCloseQualButton_Click(object sender, EventArgs e)

/// <summary> Redirects user to RemActivity page </summary>
        protected void RemActivityButton_Click(object sender, EventArgs e)
```

NewQual.aspx.cs

```
/// <summary> Runs when page loads, calls checkUser() </summary>
        protected void Page_Load(object sender, EventArgs e)

/// <summary> Checks user rights </summary>
        protected void checkUser()

/// <summary> Redirects user to edit page </summary>
        protected void EditSub_Button_Click(object sender, EventArgs e)

/// <summary> Calls method addNewQual in Qualification </summary>
        protected void Save_Button_Click(object sender, EventArgs e)

/// <summary> Cancel and go back to MainMenu </summary>
        protected void Cancel_Button_Click(object sender, EventArgs e)
```

AddSubQual.aspx.cs

```
/// <summary> calls method addSubQualification in Qualification </summary>
        protected void SaveButton_Click(object sender, EventArgs e)

        /// <summary> Cansel and redirect back to MainMenu </summary>
        protected void CanselButton_Click(object sender, EventArgs e)
```

EditSubQual.aspx.cs

```
/// <summary> Gets the choosen Qualification number from EditSearch</summary>
        /// <param name=QualNr>the qualification to edit</param>
        protected void Page_Load(object sender, EventArgs e, String QualNr)

/// <summary>Cancels and redirect to MainMenu </summary>
        protected void Cancel_Button_Click(object sender, EventArgs e)

/// <summary> calls relevant method in Qualification </summary>
        protected void Save_Button_Click(object sender, EventArgs e)
```

EditSearch.aspx.cs

```
/// <summary>calls relevant method in Qualification</summary>
       protected void findButton_Click(object sender, EventArgs e)
```

EditSubMenu.aspx.cs

```
        /// <summary>shows the general informaiton page for the chosen sub-
qualification</summary>
      protected void GenInfo_Button_Click(object sender, EventArgs e)

        /// <summary>shows the EditDocument page for the chosen sub-
qualification</summary>
        protected void Document_Button_Click(object sender, EventArgs e)

        /// <summary> shows the EditRoomPage for the chosen subqualification
</summary>
        protected void Room_Button_Click(object sender, EventArgs e)

        /// <summary> shows the EditDepartment page for the chosen sub-
qualification</summary>
        protected void Department_Button_Click(object sender, EventArgs e)

        /// <summary> shows the EditEquipment page for the chosen sub-
qualification</summary>
         protected void Equipment_Button_Click(object sender, EventArgs e)

        /// <summary> shows the EditMatrCode page for the chosen sub-
qualification</summary>
        protected void MaterialCode_Button_Click(object sender, EventArgs e)

        /// <summary>shows the EditProcessStep page for the chosen sub-
qualification</summary>
        protected void ProcessStep_Button_Click(object sender, EventArgs e)

        /// <summary> shows the EditProduct page for the chosen sub-
qualification</summary>
        protected void Product_Button_Click(object sender, EventArgs e)

        /// <summary> signs out the user </summary>
        protected void SignOutButton_Click(object sender, EventArgs e)
```

EditSubEquipment.aspx.cs

```
/// <summary> Gets the choosen Qualification number from EditSearch</summary>
      /// <param name=QualNr>the qualification to edit</param>
          protected void Page_Load(object sender, EventArgs e)

/// <summary>calls relevant method in Qualification</summary>
           protected void SaveButton_Click(object sender, EventArgs e)

/// <summary>redirects user to MainMenu </summary>
           protected void CancelButton_Click(object sender, EventArgs e)
```

EditSubMethod.aspx.cs

```
/// <summary> Gets the choosen Qualification number from EditSearch</summary>
      /// <param name=QualNr>the qualification to edit</param>
protected void Page_Load(object sender, EventArgs e)

        /// <summary>calls relevant method in DbConnection</summary>
        protected void SaveButton_Click(object sender, EventArgs e)
```

```
        /// <summary>redirect user to MainMenu</summary>
        protected void CancelButton_Click(object sender, EventArgs e)
```

EditSubMtrCode.aspx.cs

```
/// <summary> Gets the choosen Qualification number from EditSearch</summary>
        /// <param name=QualNr>the qualification to edit</param>
        protected void Page_Load(object sender, EventArgs e)

        /// <summary>calls relevant method in Qualification</summary>
        protected void Save_Button_Click(object sender, EventArgs e)

        /// <summary> redirects user to MainMenu</summary>
        protected void CancelButton_Click(object sender, EventArgs e)
```

EditSubProcessStep.aspx.cs

```
/// <summary> Gets the choosen Qualification number from EditSearch</summary>
        /// <param name=QualNr>the qualification to edit</param>
        protected void Page_Load(object sender, EventArgs e)

        /// <summary>calls relevant method in Qualification</summary>
        protected void Save_Button_Click(object sender, EventArgs e)

        /// <summary> redirects user to MainMenu</summary>
        protected void CancelButton_Click(object sender, EventArgs e)
```

EditSubDoc.aspx.cs

```
/// <summary> Gets the choosen Qualification number from EditSearch</summary>
        /// <param name=QualNr>the qualification to edit</param>
        protected void Page_Load(object sender, EventArgs e)

        /// <summary> calls relevant method in Qualification</summary>
        protected void Save_Button_Click(object sender, EventArgs e)

        /// <summary>lets user search thru the files</summary>
        protected void BrowseButton_Click(object sender, EventArgs e)

        /// <summary> clears the fields</summary>
        protected void ResetButton_Click(object sender, EventArgs e)
```

CloseQualMenu.aspx.cs

```
/// <summary> redirects user to CloseQual</summary>
        protected void CloseQualButton_Click(object sender, EventArgs e)

        /// <summary> redirects user to AddRemAct</summary>
        protected void AddRemActButton_Click(object sender, EventArgs e)
```

CloseQual.aspx.cs

```
/// <summary>calls relevant method in Qualification</summary>
        protected void CloseButton_Click(object sender, EventArgs e)
```

AddSubRemAct.aspx.cs

```
    /// <summary>calls relevent method in Qualification</summary>
        protected void AddButton_Click(object sender, EventArgs e)
```

LovMenu.aspx.cs

```
        /// <summary> redisrects user to EditDepartmentLOV</summary>
        protected void EditDepartmentButton_Click(object sender, EventArgs e)

        /// <summary> redisrects user to EditEquipmentLOV</summary>
        protected void EditEqupButton_Click(object sender, EventArgs e)

        /// <summary> redisrects user to EditEquipTypeLOV</summary>
        protected void EditEqupTypeButton_Click(object sender, EventArgs e)

        /// <summary> redisrects user to EditInstrCategoryLOV</summary>
        protected void InstrCategoryButton_Click(object sender, EventArgs e)

        /// <summary> redisrects user to EditInstructionLOV</summary>
        protected void instructionButton_Click(object sender, EventArgs e)

        /// <summary> redisrects user to EditUserLOV</summary>
        protected void userButton_Click(object sender, EventArgs e)

        /// <summary> redisrects user to EditMaterialCodeLOV</summary>
        protected void mtrlCodeButton_Click(object sender, EventArgs e)

        /// <summary> redisrects user to EditMethodLOV</summary>
        protected void methodButton_Click(object sender, EventArgs e)

        /// <summary> redisrects user to EditPartSubQualTypeLOV</summary>
        protected void partSubQualButton_Click(object sender, EventArgs e)

        /// <summary> redisrects user to EditProcessStepLOV</summary>
        protected void processSteppButton_Click(object sender, EventArgs e)

        /// <summary> redisrects user to EditProductLOV</summary>
        protected void productButton_Click(object sender, EventArgs e)

        /// <summary> redisrects user to EditQualTypeLOV</summary>
        protected void QualTypeButton_Click(object sender, EventArgs e)

        /// <summary> redisrects user to EditRoomLOV</summary>
        protected void roomButton_Click(object sender, EventArgs e)

        /// <summary> redisrects user to EditSeriesLOV</summary>
        protected void seriesButton_Click(object sender, EventArgs e)

        /// <summary> redisrects user to EditSubQualTypeLOV</summary>
        protected void subQualTypeButton_Click(object sender, EventArgs e)

        /// <summary> redisrects user to EditTypeListLOV</summary>
        protected void typeListButton_Click(object sender, EventArgs e)
```

RemActivity.aspx.cs

```
/// <summary>calls listRemAct</summary>
        protected void Page_Load(object sender, EventArgs e)
```

```
/// <summary>calls relevent method in DbConnection, list all remaining
activiteis</summary>
       protected void listRemAct()
```

EditEquipment.aspx.cs

```
/// <summary>checks if new, calls relevant method in ListOfValues</summary>
 protected void SaveButton_Click(object sender, EventArgs e)

/// <summary>redirects user to LOVMenu</summary>
protected void CancelButton_Click(object sender, EventArgs e)
```

EditMethod.aspx.cs

```
/// <summary>checks if new, calls relevant method in ListOfValues </summary>
 protected void SaveButton_Click(object sender, EventArgs e)

/// <summary>redirects user to LOVMenu</summary>
protected void CancelButton_Click(object sender, EventArgs e)
```

EditMtrlCode.aspx.cs

```
/// <summary>checks if new, calls relevant method in ListOfValues </summary>
 protected void SaveButton_Click(object sender, EventArgs e)

/// <summary>redirects user to LOVMenu</summary>
protected void CancelButton_Click(object sender, EventArgs e)
```

EditProcessStep.aspx.cs

```
/// <summary>checks if new, calls relevant method in ListOfValues </summary>
 protected void SaveButton_Click(object sender, EventArgs e)

/// <summary>redirects user to LOVMenu</summary>
protected void CancelButton_Click(object sender, EventArgs e)
```

SearchQual

```
/// <summary>checks if new, calls relevant method in Qualification</summary>
 protected void SearchButton_Click(object sender, EventArgs e)

/// <summary>clears fields</summary>
protected void ResetButton_Click(object sender, EventArgs e)
```

EditDepartmentLOV

```
/// <summary>checks if new, calls relevant method in ListOfValues </summary>
 protected void SaveButton_Click(object sender, EventArgs e)

/// <summary>redirects user to LOVMenu</summary>
protected void CancelButton_Click(object sender, EventArgs e)
```

EditEquipmentLOV

```
/// <summary>checks if new, calls relevant method in ListOfValues </summary>
 protected void SaveButton_Click(object sender, EventArgs e)

/// <summary>redirects user to LOVMenu</summary>
protected void CancelButton_Click(object sender, EventArgs e)
```

EditInstructionLOV.aspx.cs

```
/// <summary>checks if new, calls relevant method in ListOfValues </summary>
 protected void SaveButton_Click(object sender, EventArgs e)

/// <summary>redirects user to LOVMenu</summary>
protected void CancelButton_Click(object sender, EventArgs e)
```

EditEquipTypeLOV.aspx.cs

```
/// <summary>checks if new, calls relevant method in ListOfValues </summary>
 protected void SaveButton_Click(object sender, EventArgs e)

/// <summary>redirects user to LOVMenu</summary>
protected void CancelButton_Click(object sender, EventArgs e)
```

EditInstrCategoryLOV.apsx.cs

```
/// <summary>checks if new, calls relevant method in ListOfValues </summary>
 protected void SaveButton_Click(object sender, EventArgs e)

/// <summary>redirects user to LOVMenu</summary>
protected void CancelButton_Click(object sender, EventArgs e)
```

EditMaterialCodeLOV.aspx.cs

```
/// <summary>checks if new, calls relevant method in ListOfValues </summary>
 protected void SaveButton_Click(object sender, EventArgs e)

/// <summary>redirects user to LOVMenu</summary>
protected void CancelButton_Click(object sender, EventArgs e)
```

EditMethodLOV.aspx.cs

```
/// <summary>checks if new, calls relevant method in ListOfValues </summary>
 protected void SaveButton_Click(object sender, EventArgs e)

/// <summary>redirects user to LOVMenu</summary>
protected void CancelButton_Click(object sender, EventArgs e)
```

EditProcessStepLOV.aspx.cs

```
/// <summary>checks if new, calls relevant method in ListOfValues </summary>
 protected void SaveButton_Click(object sender, EventArgs e)

/// <summary>redirects user to LOVMenu</summary>
protected void CancelButton_Click(object sender, EventArgs e)
```

EditProductLOV.aspx.cs

```
/// <summary>checks if new, calls relevant method in ListOfValues </summary>
 protected void SaveButton_Click(object sender, EventArgs e)

/// <summary>redirects user to LOVMenu</summary>
protected void CancelButton_Click(object sender, EventArgs e)
```

EditQualTypeLOV.aspx.cs

```
/// <summary>checks if new, calls relevant method in ListOfValues </summary>
 protected void SaveButton_Click(object sender, EventArgs e)
```

```
/// <summary>redirects user to LOVMenu</summary>
protected void CancelButton_Click(object sender, EventArgs e)
```

EditRoomLOV.aspx.cs

```
/// <summary>checks if new, calls relevant method in ListOfValues </summary>
 protected void SaveButton_Click(object sender, EventArgs e)

/// <summary>redirects user to LOVMenu</summary>
protected void CancelButton_Click(object sender, EventArgs e)
```

EditSeriesLOV.aspx.cs

```
/// <summary>checks if new, calls relevant method in ListOfValues </summary>
 protected void SaveButton_Click(object sender, EventArgs e)

/// <summary>redirects user to LOVMenu</summary>
protected void CancelButton_Click(object sender, EventArgs e)
```

EditSubQualTypeLOV.aspx.cs

```
/// <summary>checks if new, calls relevant method in ListOfValues </summary>
 protected void SaveButton_Click(object sender, EventArgs e)

/// <summary>redirects user to LOVMenu</summary>
protected void CancelButton_Click(object sender, EventArgs e)
```

EditTypeListLOV.aspx.cs

```
/// <summary>checks if new, calls relevant method in ListOfValues </summary>
 protected void SaveButton_Click(object sender, EventArgs e)

/// <summary>redirects user to LOVMenu</summary>
protected void CancelButton_Click(object sender, EventArgs e)
```

EditUserLOV.aspx.cs

```
/// <summary>checks if new, calls relevant method in User</summary>
 protected void SaveButton_Click(object sender, EventArgs e)

/// <summary>redirects user to LOVMenu</summary>
protected void CancelButton_Click(object sender, EventArgs e)
```

5.5.4 Requirements traceability matrix

#	Requirement
	Design item
1.1	It shall be possible to run the system in a Windows domain environment.
	See section 3. System will be implemented using Microsoft ASP.NET.
1.2	It shall be possible to run the system on Windows operating systems.
	See 1.1
1.3	It shall be possible to use MS SQL Server 2005 for the database layer.
	Relational database (see app. 1) can be implemented using MS SQL Server 2005
1.4	It shall be possible to access the system from the MS Sharepoint Server intranet.
	See section 3. The system will be developed as an ASP.NET web application and will be possible to run from web part in MS Sharepoint.
1.5	It shall be possible to migrate and store data from the present system in the new system.
	Columns from tables in the existing relational database have been taken into account when designing the new data model (see appendix 1), which can be verified visually by comparing the data models and by creating SQL migration scripts. Some parts may need some additional manual work and data may also need some washing prior to or during migration.
1.6	It shall be possible to configure database access information.
	ASP.Net project web.config XML file
1.7	It shall be possible to use different database instances (e.g. training and production instance).
	GUI (app. 2), Login.aspx, class DbConnection
1.8	It shall be possible to implement single sign-on from a Windows domain account.
	Class User
1.9	It shall be possible to use Windows domain groups to restrict access to the database.
	Class User, MS Windows Active Directory

1.10	A users manual and relevant system documentation shall be delivered with the system.
	A manual and system specification will be delivered with the system.
1.11	It should be possible to import information from and export information to other information systems based on relational databases.
	SQL Server 2005 will be used to implement the database (see app. 1). Standard SQL can be used to export data for transfer to other systems or to import data from other relational database systems. The migration of data from the old system (see 1.5) would be an example of this.
2.1	It shall be possible to obtain a new, unique qualification numbers on the format X-YYYY from the system for a series X (e.g. 30, 80, 110) given by the user, where the sequence YYYY is incremented by one for each new qualification.
	Database Stored Procedure storeQualification (app. 1)
2.2	It shall be possible to choose if the qualification is to refer to a validation (V) or a qualification (Q) when obtaining a new number from the system.
	GUI (app. 2)
2.3	It shall be possible to store sub-qualification numbers, along with qualification title, name of the qualification leader, project number, start date, approval date and end date in the system.
	Database Stored Procedure storeQualification (app. 1), GUI (app. 2), class Qualification
2.4	It shall be possible to store the status (e.g. NEW, APPROVED, CLOSED) of a qualification in the system.
	Database Stored Procedure storeQualification (app. 1), GUI (app. 2), class Qualification
2.5	It shall be possible to add sub-qualifications for a given qualification number.
	GUI (app.2), class Qualification
2.6	Sub-qualifications shall be identified by the qualification number (X-YYYY) followed by V for validations and Q for qualifications, followed by a sequence.
	Database Stored Procedure storeQualification (app. 1), class Qualification
2.7	It shall be possible to store the type of the sub-qualification (e.g. IQ, OQ).
	Database Stored Procedure storeQualification (app. 1), GUI (app. 2), class Qualification

2.8	A validation plan (main sub-qualification) shall be stored as X-YYYY-V01.
	class Qualification
2.9	It shall be possible to divide sub-qualification in different parts, designated as qualification number X-YYYY-[V| Q]x-Dy, where x and y are sequences for their respective categories V|Q (validation and qualification respectively) and D (Swedish “del”).
	Database Stored Procedure storeQualification (app. 1), GUI (app 2), class Qualification
2.10	It shall be possible to add e.g. an addendum or a re-qualifications in the same manner as parts (see requirement 2.9), replacing the letter “D” with and appropriate letter (e.g. “T” for Swedish “tillägg” R for requalification and “C” for “commissioning”).
	Database Stored Procedure storeQualification (app. 1), GUI (app. 2), class Qualification
2.11	It shall be possible to link information about documents to a sub-qualification, including document ID, title, author, approval date, archive location and ID in the archiving system.
	Database Stored Procedure storeDocument (app. 1), GUI (app. 2), classes Qualification, Document
2.12	It shall be possible to store the type of the document (e.g. URS, Protocol, Report).
	Database Stored Procedure storeDocument (app. 1), class Document
2.13	It shall be possible to store information about links to electronic documentation in the system.
	Database Stored Procedure storeDocument (app. 1), GUI (app. 2), class Document
2.14	It should be possible to store and access electronic documentation from within the system.
	GUI (app. 2)
2.15	It shall not be possible to change a qualification or sub-qualification number.
	Database Stored Procedure storeQualification (app. 1)
2.16	It shall be possible to edit information about a qualification.
	Database Stored Procedure storeQualification (app. 1), GUI (app. 2), class Qualification
2.17	It shall be possible to edit information about documentation.
	Database Stored Procedure storeDocument (app. 1), GUI (app. 2), class Document

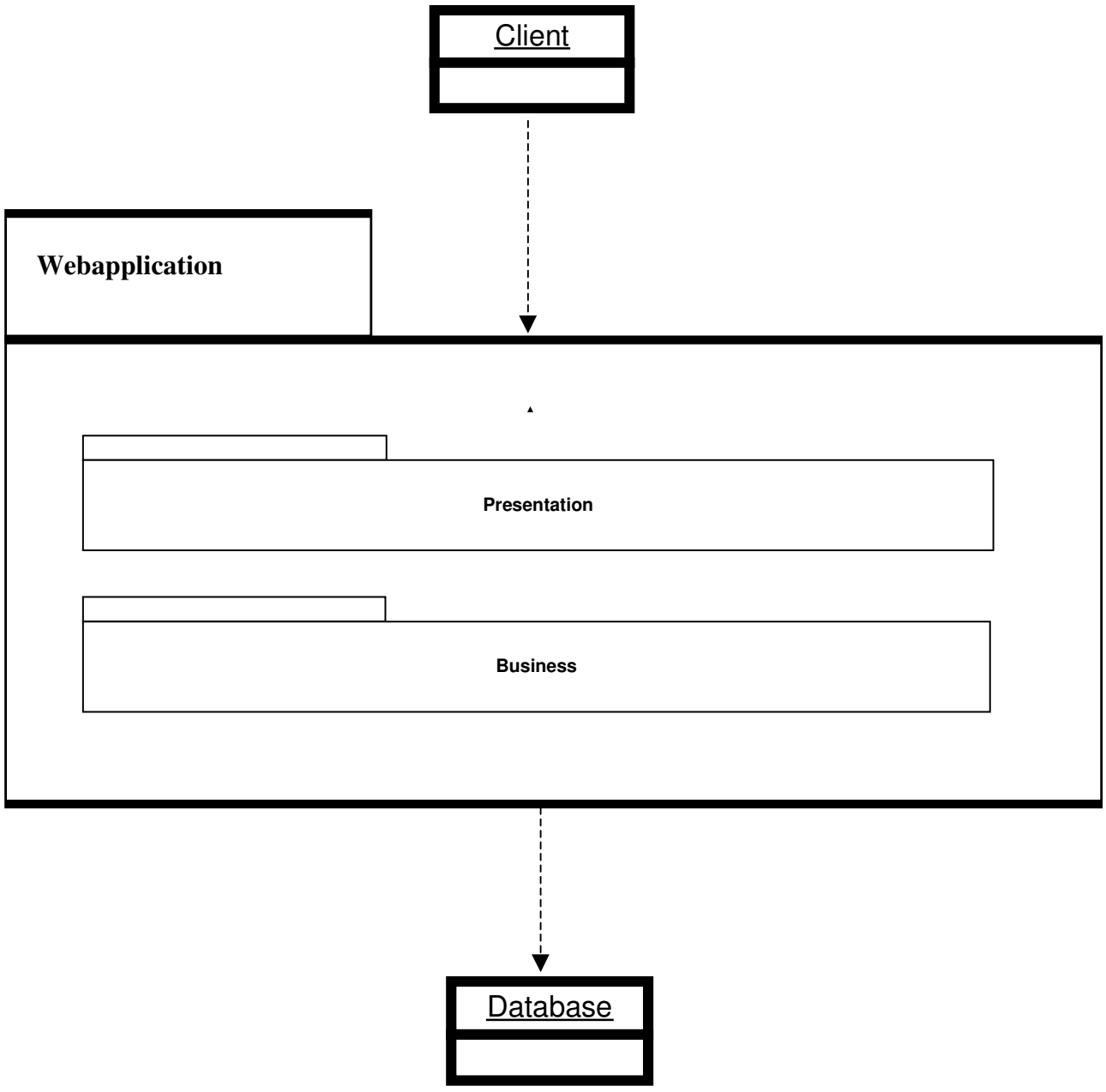
2.18	It shall not be possible to delete qualification records, document records or items being referenced by other items from the system.
	Database SP absence of remove-method for Qualification and Document. Database Stored Procedures (app. 1): removeType, removeProduct , removeEquipType, removeProcessStep, removeRoom, removeDepartment, removeCategory, removeMtrlCode, removeMethod, removeSeries, removeQualType, removeSubQualType, removePartQualType, removeDocType
2.19	It shall be possible to specify a status for qualification and document records in the system, e.g. CURRENT, OBSOLETE.
	Database Stored Procedure storeQualification (app. 1), GUI (app. 2), classes Qualification and Document
2.20	It shall be possible to copy information for a sub-qualification to create a new sub-qualification number.
	GUI (app. 2), class Qualification
2.21	It shall be possible to store and edit information about remaining activities for a sub-qualification (e.g. name of the activity, person responsible, status and date completed).
	Database Stored Procedure storeRemActivity (app. 1), GUI (app.2), class Activity
2.22	It shall be possible to store and edit product names, re-qualification interval and qualification status in the system.
	Database Stored Procedure storeProduct (app. 1), GUI (app.2), class ListOfValues
2.23	It shall be possible to store and edit process steps for different products in the system.
	Database Stored Procedure storeProcessStep (app. 1), GUI (app.2), EditProcessStepLOV, class ProcessStep
2.24	It shall be possible to store and edit information about equipment names, re-qualification intervals and qualification status in the system.
	Database Stored Procedure storeEquipment (app. 1), GUI (app.2), EditEquipmentLOV, EditInsructionLOV, EditInstrCategoryLOV, EditEquipTypeLOV, class Equipment
2.25	It shall be possible to specify which product specific equipment is used for.
	Database Stored Procedure storeEquipProduct (app. 1), GUI (app.2), EditProductLOV, class Equipment

2.26	It shall be possible to store history for equipment use for different products with regards to the qualifications performed.
	Database Stored Procedure storeEquipment (app. 1), GUI (app.2), EditEquip, class Equipment
2.27	It shall be possible to store method names, re-qualification interval and qualification status in the system.
	Database Stored Procedure storeMethod (app. 1), GUI (app.2), EditMethodLOV, class Method
2.28	It shall be possible to link information about which products, process steps, equipment, material codes and methods are affected by a qualification.
	Database Stored Procedures storeQualMethod, storeQualProcess, storeQualMtrl, storeQualEquip (app. 1), GUI (app.2), EditEquip, EditMethod, EditMtrlCode, EditProcessStep, class Qualification
2.29	It shall be possible to specify which change control numbers are relevant for a particular qualification.
	Database Stored Procedure storeQualCC (app. 1), GUI (app.2), class Qualification
2.30	It shall be possible to store and edit information about rooms and departments.
	Database Stored Procedures storeRoom, storeDepartment (app. 1), GUI (app.2), EditDepartmentLOV, EditRoomLOV, classes Room and ListOfValues
2.31	It shall be possible to store and edit information about qualification series (e.g. 30 for production, 45 for laboratories etc).
	Database Stored Procedure storeSeries (app. 1), GUI (app.2), EditSeriesLOV, class ListOfValues
2.32	It shall be possible to add free-text comments for items stored in the database.
	Most Database Stored Procedures (app. 1), GUI (app.2)
2.33	It shall be possible to search the system for qualifications using parameters such as qualification numbers, qualification title, archive location, start, approval and end dates, product, material code, process step, equipment, method, room and department.
	Database Stored Procedure searchQualification (app. 1), GUI (app.2), class Qualification, SearchQual
2.34	When searching for qualifications, it shall be possible to use wildcards (* or %).
	Database Stored Procedure searchQualification (app. 1)
2.35	It shall be possible to export search results in an Excel readable format.
	GUI (app. 2), SearchQual

2.36	It should be possible to invoke Excel with the exported results from within the system.
	GUI (app. 2), SearchQual
2.37	It shall be possible to add customised pre-defined reports to the system.
	SearchQual.aspx, (this requirement may have to be postponed due to time restraints).
2.38	Wherever possible, list-of-values for users to choose from shall be used (to ensure data consistency).
	GUI (app. 2), EditTypeListLOV
2.39	It should be possible to add users to the system or to perform authentication against domain groups.
	GUI (app. 2), Login.aspx, class User
2.40	It should be possible to store user access levels within the system.
	Database stored procedure storeUser (app. 1)
2.41	It shall be possible to choose the database instance to use at start-up.
	GUI (app. 2), Login.aspx, class User
2.42	A user should be automatically logged out of the system after 30 min of inactivity.
	Session, web.config
3.1	It shall be possible to have ten concurrent users in the system.
	Limited by SQL server licences. Web server should be able to handle at least ten clients.
3.2	A user shall not have to wait more than 5s when storing data.
	Optimisation of database stored procedures (app. 1) for storing. 5s may not be possible to fulfil under all circumstances, as server and network capacity also have an impact on the response times.
3.3	A search result shall be displayed within 2 min.
	Optimisation of database stored procedure searchQualification (app. 1)
3.4	A search result should be displayed within 30s.
	Optimisation of database stored procedure searchQualification (app. 1) 30s may not be possible to fulfil under all circumstances, as server and network capacity also have an impact on the response times.

3.5	It shall not be possible for two users to update the same record at the same time.
	Database stored procedures (app. 1)
3.6	Error messages shall be displayed when application and network errors occur.
	Web forms, business layer classes
3.7	On application or network failure, unsaved changes to the database shall be rolled back.
	Database standard rollback functionality rolls back all session information that has not been committed to the database. Transaction handling will be used in database stored procedures (app. 1).
3.8	If the user closes the application without saving, any unsaved changes to the database shall be rolled back.
	See 3.7
3.9	It shall be possible to set different access levels for different users.
	Access levels admin, create, change and read will be implemented using Active Directory domain groups.
3.10	Only users with administrator privileges shall be allowed to set access levels.
	N/A when using Windows domain group functionality. Active Directory is managed externally.
3.11	Users shall be able to operate the system adequately after a four-hour training course (less for read-only access).
	GUI (app. 2)

5.6. Package Diagram



6 Functional Test Cases

Functional test cases can be found in appendix 3.