

## DESIGN DESCRIPTION

1

Project **Online Component Repository**

### DISTRIBUTION

**Steering group:**

Frank Lüders

**Project group:**

Vladimir Djukanovic

Cristian Capozucco

Oskar Palmgren

Aleksandar Matovic

Bastien Delbouys

Mohamed Abdi

**CONTENTS**

<b>1</b>	<b>Introduction.....</b>	<b>4</b>
1.1	Background.....	4
1.2	Definitions .....	4
1.3	Related Documents.....	4
<b>2</b>	<b>Functional Description .....</b>	<b>6</b>
2.1	Use Case Model.....	6
2.1.1	Actors	7
2.1.2	Use Cases	7
2.2	Browse & Search .....	7
2.2.1	Participating Actors	7
2.2.2	Related Use Cases	7
2.2.3	Precondition	7
2.2.4	Main Flow of Events	7
2.2.5	Alternative: No component found	7
2.3	Inspect Classes & Interfaces .....	7
2.3.1	Participating Actors	7
2.3.2	Related Use Cases	7
2.3.3	Precondition	8
2.3.4	Main Flow of Events	8
2.4	Download.....	8
2.4.1	Participating Actors	8
2.4.2	Related Use Cases	8
2.4.3	Precondition	8
2.4.4	Main Flow of Events	8
2.4.5	Alternative: The component is not available	8
2.5	Login.....	9
2.5.1	Participating Actors	9
2.5.2	Related Use Cases	9
2.5.3	Main Flow of Events	9
2.5.4	Alternative: Invalid information	9
2.6	Register.....	9
2.6.1	Participating Actors	9

2.6.2	Main Flow of Events	9
2.6.3	Alternative: Invalid information	9
2.7	EditProfil.....	10
2.7.1	Participating Actors	10
2.7.2	Related Use Cases	10
2.7.3	Precondition	10
2.7.4	Main Flow of Events	10
2.7.5	Alternative: Invalid information	10
2.8	AddComponent.....	10
2.8.1	Participating Actors	10
2.8.2	Related Use Cases	10
2.8.3	Precondition	11
2.8.4	Main Flow of Events	11
2.8.5	Alternative: The upload component form is invalid	11
2.9	RemoveComponent .....	12
2.9.1	Participating Actors	12
2.9.2	Related Use Cases	12
2.9.3	Precondition	12
2.9.4	Main Flow of Events	13
2.9.5	Alternative: Cancel confirmation panel for component removal	13
2.10	EditComponent .....	14
2.10.1	Participating Actors	14
2.10.2	Related Use Cases	14
2.10.3	Precondition	15
2.10.4	Main Flow of Events	15
2.10.5	Alternative: The updated form is invalid	16
<b>3</b>	<b>External Interfaces.....</b>	<b>16</b>
3.1	Graphical User Interface.....	16
<b>4</b>	<b>Software Architecture.....</b>	<b>19</b>
4.1	Overview and Rationale.....	19
4.2	System Decomposition .....	19
4.3	Hardware/Software Mapping.....	20

4.4	Persistent Data .....	20
4.5	Access Control.....	20
4.6	Synchronization and Timing.....	20
4.7	Start-Up and Shut-Down .....	20
4.8	Error Handling .....	20
<b>5</b>	<b>Detailed Software Design.....</b>	<b>20</b>
5.1	</Component Name>.....	20
5.1.1	Static Structure	20
5.1.2	Dynamic Behaviour	20

## 1 Introduction

### 1.1 Background

*Help: Describe the reasons for the product/component/function to be developed. Also, delineate the purpose of this document and specify the intended audience.*

This report describe the design of a software component repository that is a part of the course Component-based Technologies.

The system is a web based system that lets the users browse, search and download component. The system also have administrators that can add, remove and edit information of the components.

The software architecture is a component-based architecture written in C#. Some components must however be written in C++ and Java to access information from COM component and JavaBeans components.

### 1.2 Definitions

*Help: List the special terms used, if any. Make references to other documents with definitions.*

#### Terms

#### Definitions

### 1.3 Related Documents

*Help: If applicable, provide a list of documents related to the product/component/function, This could be documents that this document is based on or that explain a specific requirement more in detail. Only include documents that do exist.*

Document identity	Document title
-------------------	----------------

## 2 Functional Description

### 2.1 Use Case Model

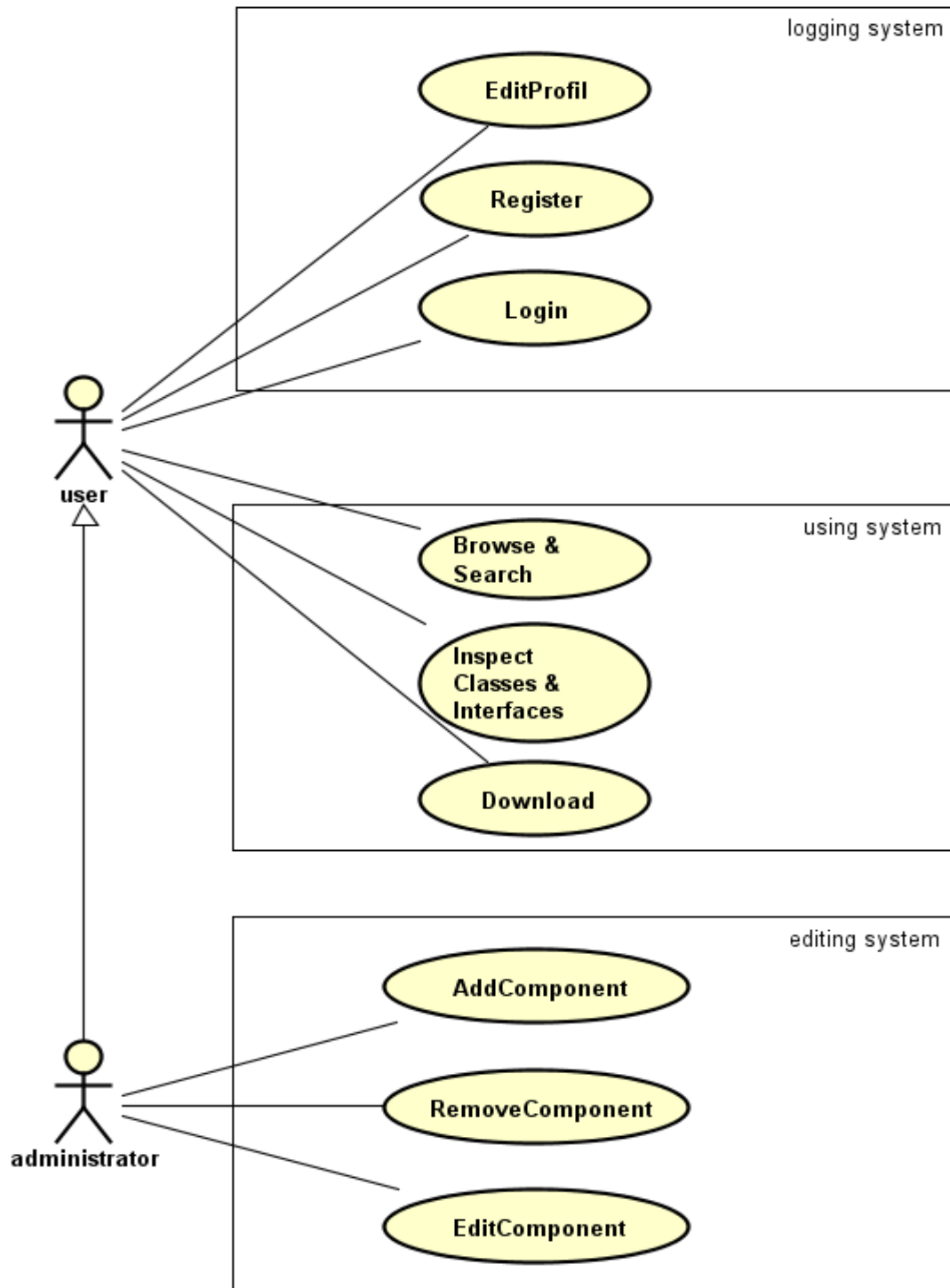


Figure 1 - Use Case Diagram

### 2.1.1 Actors

*Help: Describe the actors in your use case model. Each actor should describe a role an external entity may play when interacting with the system.*

### 2.1.2 Use Cases

Each use case is described in a separate section in the remainder of this chapter.

## 2.2 Browse & Search

### 2.2.1 Participating Actors

User

### 2.2.2 Related Use Cases

Login

### 2.2.3 Precondition

The user has to be logged in.

### 2.2.4 Main Flow of Events

1. The user search for the name of a component
2. The system looks into the component database for any correspondences
3. The system displays the list of components that match the search

### 2.2.5 Alternative: No component found

- At step 2 : If no component match the search

The system displays a message to the user to inform him that no component match the search.

- Return to step 1

## 2.3 Inspect Classes & Interfaces

### 2.3.1 Participating Actors

User

### 2.3.2 Related Use Cases

Login

Browse & Search

### 2.3.3 Precondition

The user has to be logged in.

The user has to have done a successful search.

### 2.3.4 Main Flow of Events

1. The user selects one of the displayed components.
2. The user clicks the “INSPECT” button.
3. The system display component’s Classes and Interfaces.
4. The user has to clicks the “RETURN” button to end the case.

## 2.4 Download

### 2.4.1 Participating Actors

User

### 2.4.2 Related Use Cases

Login

Browse & Search

### 2.4.3 Precondition

The user has to be logged in.

The user has to have done a successful search.

### 2.4.4 Main Flow of Events

1. The user selects one of the displayed components.
2. The user clicks the “DOWNLOAD” button.
3. The system checks the availability of the component.
4. The system sends the component to the user.

### 2.4.5 Alternative: The component is not available

- At step 3 : The component is not available for the download.

The system displays a message telling the user than the component in not available for the download.

- The system return to the previous user’s search.



## 2.5 Login

### 2.5.1 Participating Actors

User

### 2.5.2 Related Use Cases

Register

### 2.5.3 Main Flow of Events

1. The user clicks the “LOGIN” button.
2. The system displays a login form (login, password).
3. The user completes the form.
4. The system checks if the form is valid.
5. The user is now logged in, the system displays the user interface.

### 2.5.4 Alternative: Invalid information

- At step 4 : Form is not valid, the association login-password is not valid.  
The system displays to the user the error.
- The system brings back the user to the home page.

## 2.6 Register

### 2.6.1 Participating Actors

User

### 2.6.2 Main Flow of Events

1. The user clicks the “REGISTER” button.
2. The system displays a form for the user to complete (login, password).
3. The user completes the register form.
4. The system checks the validity of the form.
5. The system brings back the user to the home page.

### 2.6.3 Alternative: Invalid information

- At step 4 : Form is not valid, login already exist, or the password is not valid.  
The system displays the error committed by the user in the form.
- Return to step 2.

## 2.7 EditProfil

### 2.7.1 Participating Actors

User

### 2.7.2 Related Use Cases

Login

### 2.7.3 Precondition

The user has to be logged in.

### 2.7.4 Main Flow of Events

1. The user clicks the “EDIT” button.
2. The system displays a form to the user (old password, new password).
3. The user fills the form.
4. The system check if the form is valid.
5. The system brings back the user to the home page.

### 2.7.5 Alternative: Invalid information

- At step 4 : Form is not valid, the old password is incorrect or the new one is not valid.

The system displays to the user the error.

- Return to step 2.

## 2.8 AddComponent

This use case is responsible for adding components to the repository and displaying a component in the UI if the upload was successful.

### 2.8.1 Participating Actors

The initiating actor is administrator and only the user with admin privileges can perform this use case.

### 2.8.2 Related Use Cases

Based on the belonging subsystem the related use cases is/are:

- Remove Component and
- Edit Component

Based on the preconditions the related use case/s is/are:

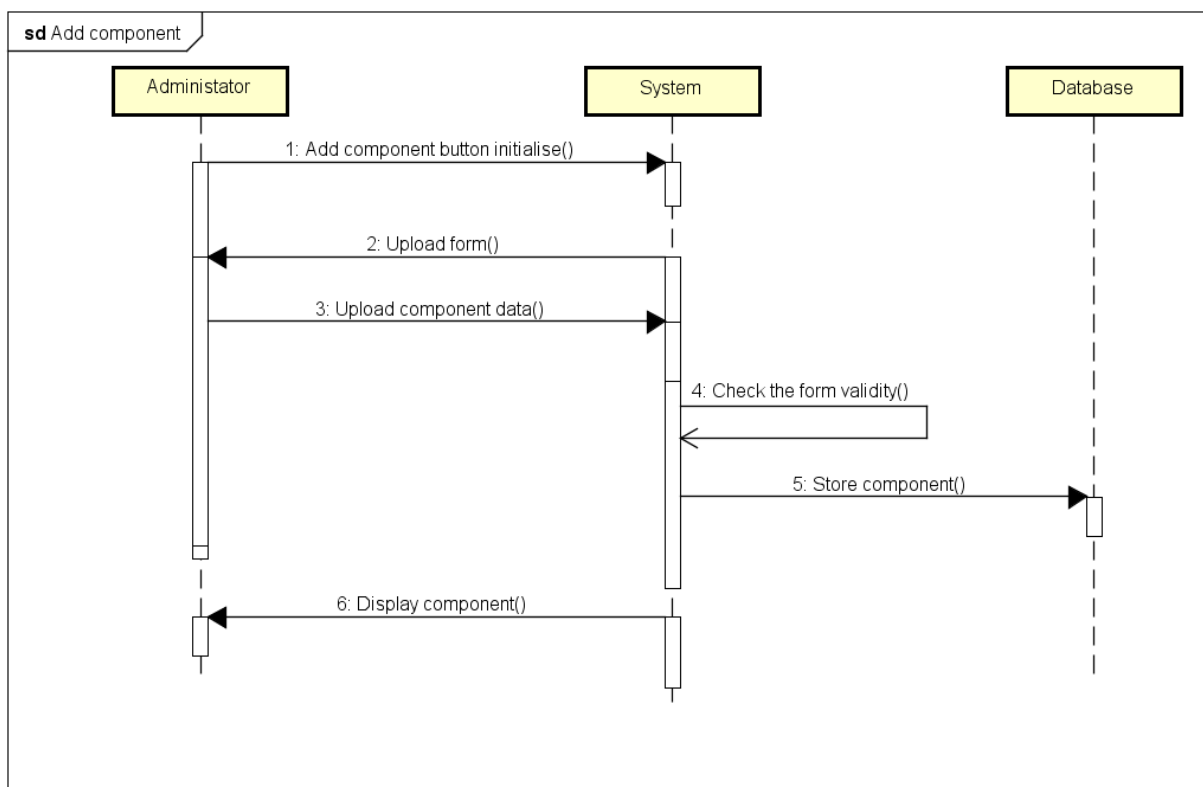
- Login

### 2.8.3 Precondition

User has to be logged in.

### 2.8.4 Main Flow of Events

1. The administrator initiates to add the component (by pressing a button etc.)
2. The system gives the user the form to upload component
3. The administrator fills the form and press the upload button
4. The system checks the validity of form
5. The system stores the component in the database
6. The system displays newly added component

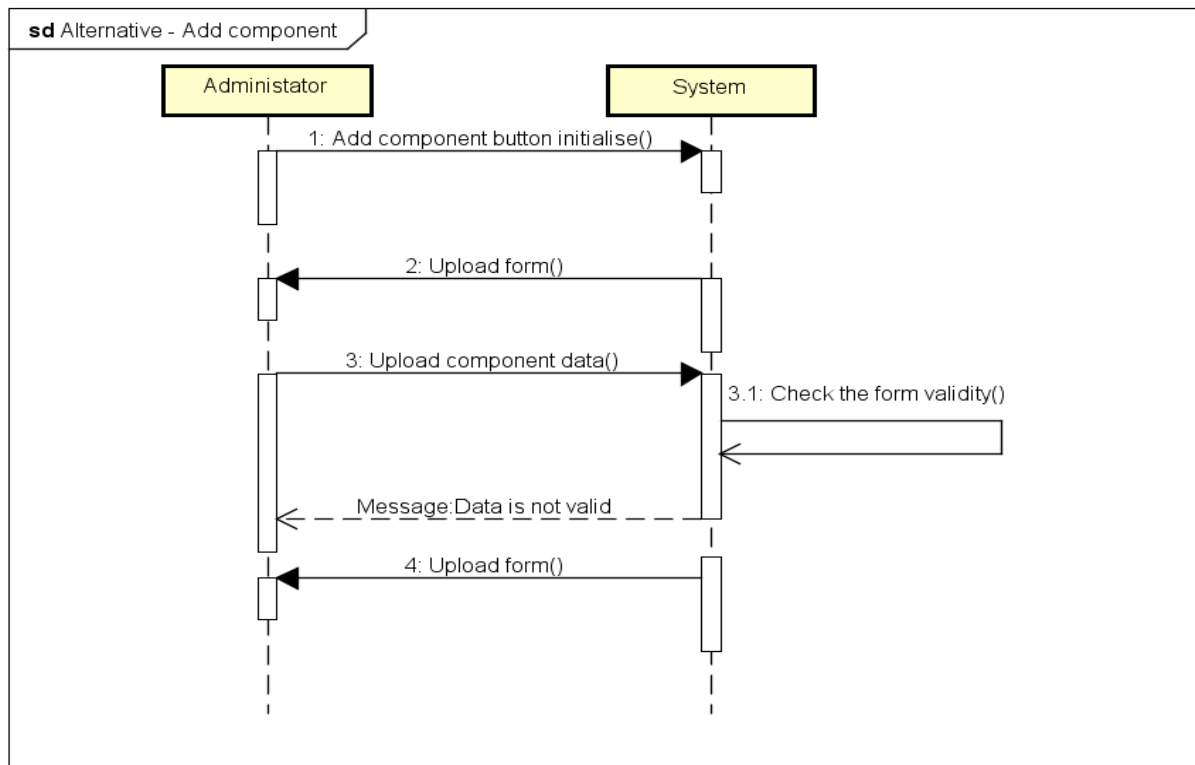


powered by Astah

Figure 2: Add Component sequence diagram

### 2.8.5 Alternative: The upload component form is invalid

- At step 4: Form is not valid, all data not provided or invalid data.  
The system displays the error committed by the user in the form.
- Return to step 3



powered by Astah

Figure 3: Alternative sequence diagram (form invalid)

## 2.9 RemoveComponent

This use case is responsible for removing components from the repository and updating UI if the removal was successful.

### 2.9.1 Participating Actors

The initiating actor is administrator and only the user with admin privileges can perform this use case.

### 2.9.2 Related Use Cases

Based on the belonging subsystem the related use case/s is/are:

- Add Component and
- Edit Component

Based on the preconditions the related use cases is/are:

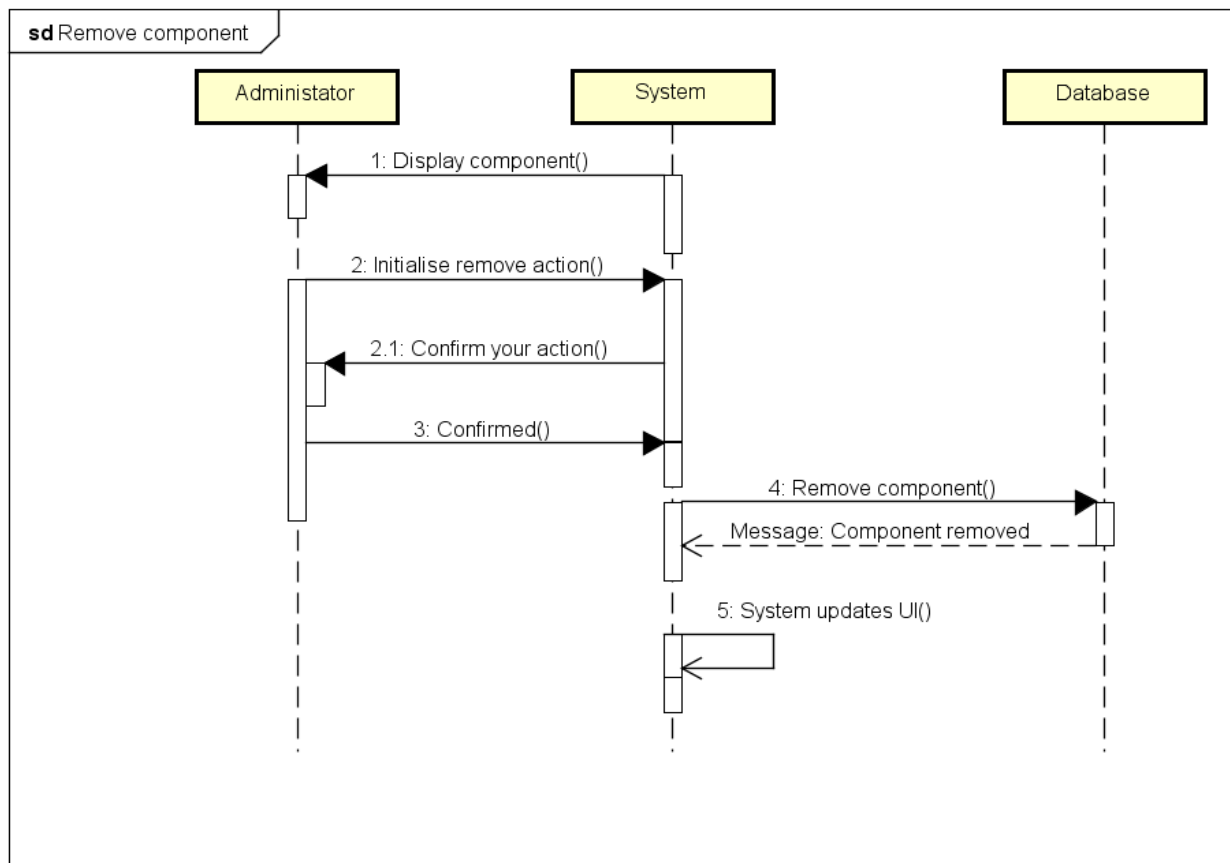
- Login

### 2.9.3 Precondition

User has to be logged in.

#### 2.9.4 Main Flow of Events

1. The administrator selects the component and clicks remove button
2. The system asks for confirmation to remove selected component
3. The administrator clicks to confirm removal
4. The system removes the component from database
5. The system updates UI

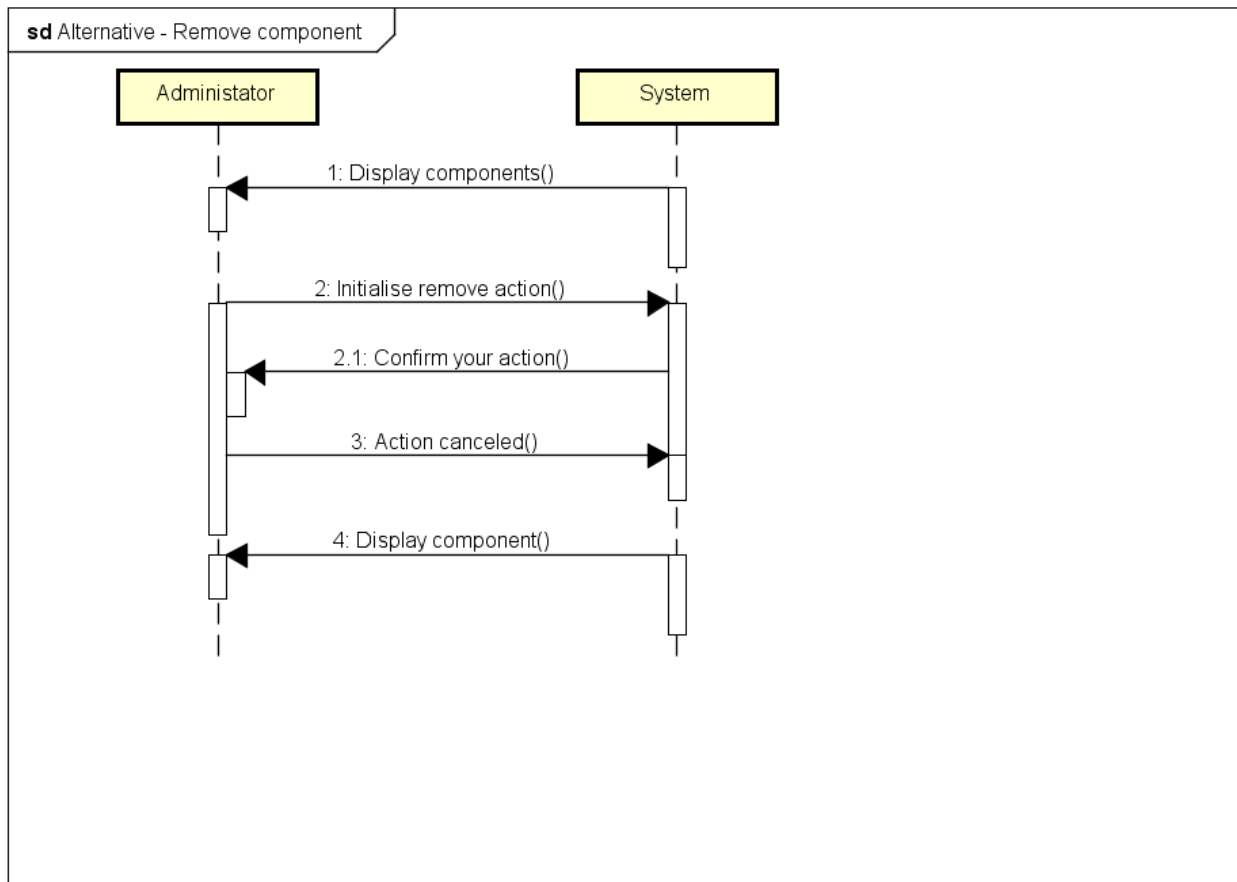


powered by Astah

Figure 4: Remove Component sequence diagram

#### 2.9.5 Alternative: Cancel confirmation panel for component removal

- At step 3: The administrator clicks to cancel component removal  
The system closes the confirmation panel.
- The system returns to home page.



powered by Astah

Figure 5: Alternative sequence diagram (cancel confirmation)

## 2.10 EditComponent

This use case is responsible for editing component information and therefore updating the repository and UI if the editing was successful.

### 2.10.1 Participating Actors

The initiating actor is administrator and only the user with admin privileges can perform this use case.

### 2.10.2 Related Use Cases

Based on the belonging subsystem the related use cases is/are:

- Remove Component and
- Remove Component

Based on the preconditions the related use case/s is/are:

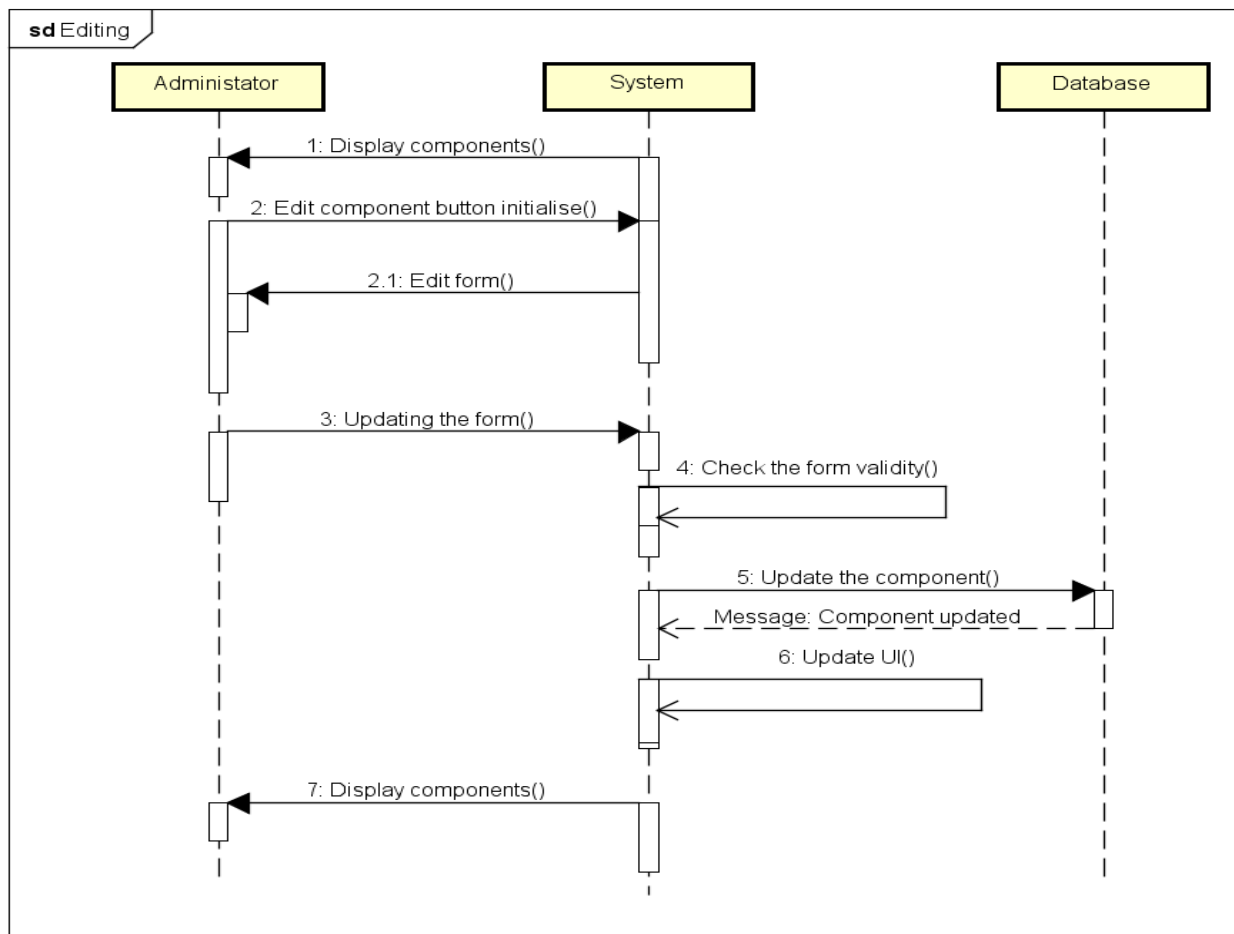
- Login

### 2.10.3 Precondition

User has to be logged in.

### 2.10.4 Main Flow of Events

1. The administrator clicks the edit button for specific component
2. The system gives administrator the form to edit component information
3. The administrator changes the form and clicks to update component information
4. The system checks the validity of the form
5. The system updates the component in the database and updates UI

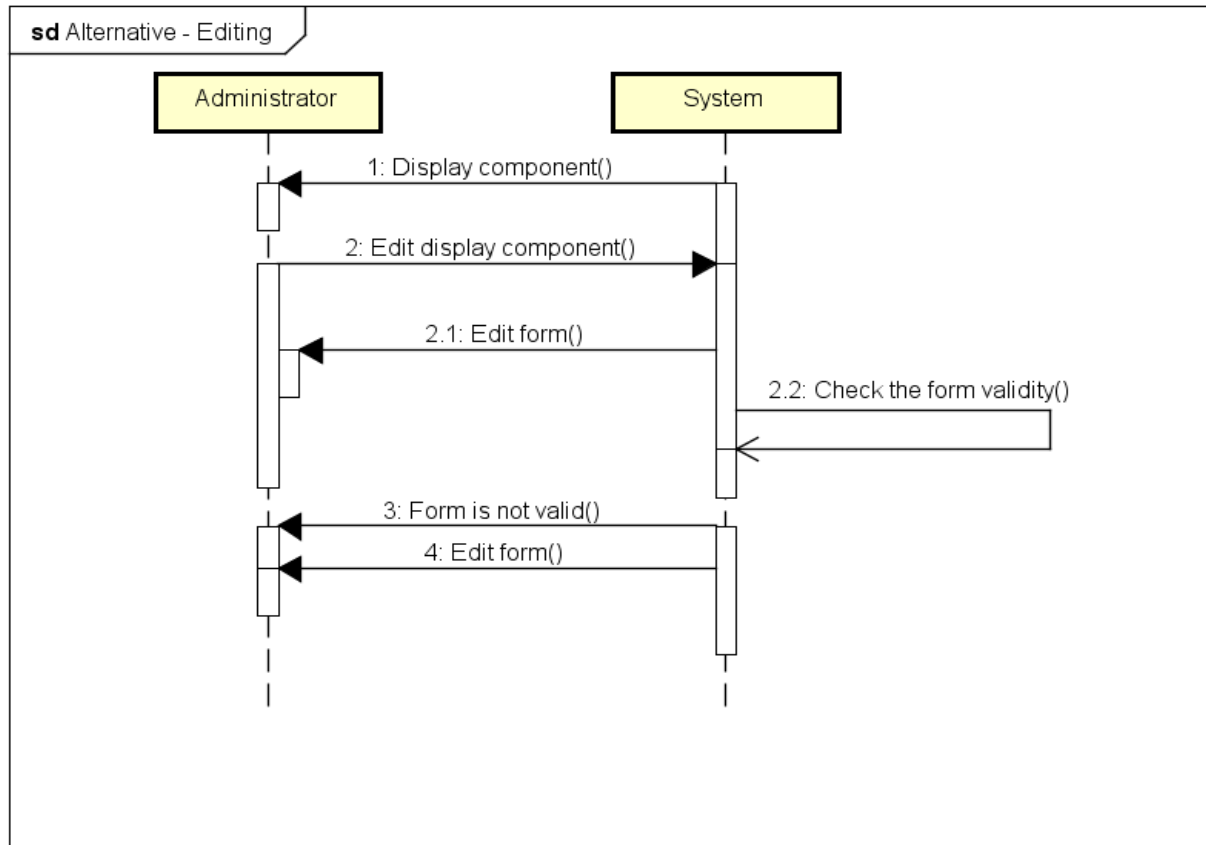


powered by Astah

Figure 6: Edit Component sequence diagram

## 2.10.5 Alternative: The updated form is invalid

- At step 4: Form is not valid, all data not provided or invalid data.  
The system displays the error committed by the user in the form.
- Return to step 3



powered by Astah

Figure 7: Alternative sequence diagram (form invalid)

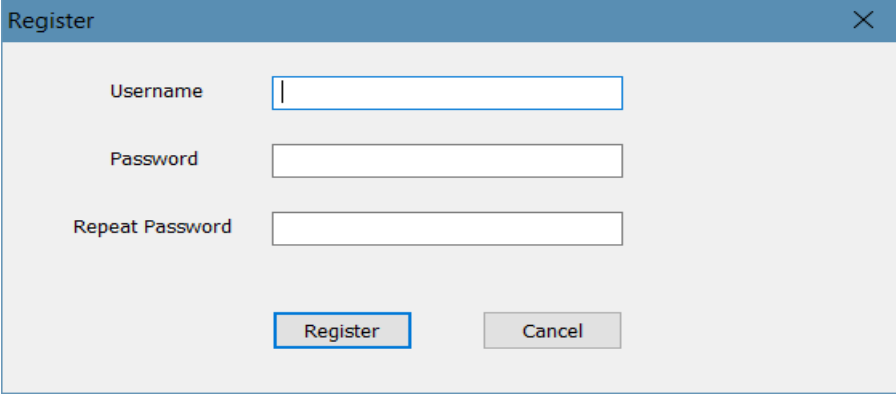
### 3 External Interfaces

*Help* Describe the interface(s) to users and other external entities.

#### 3.1 Graphical User Interface

*Help* Omit if not relevant. Otherwise describe the structure and general layout of the interface using subsections and figures as appropriate.

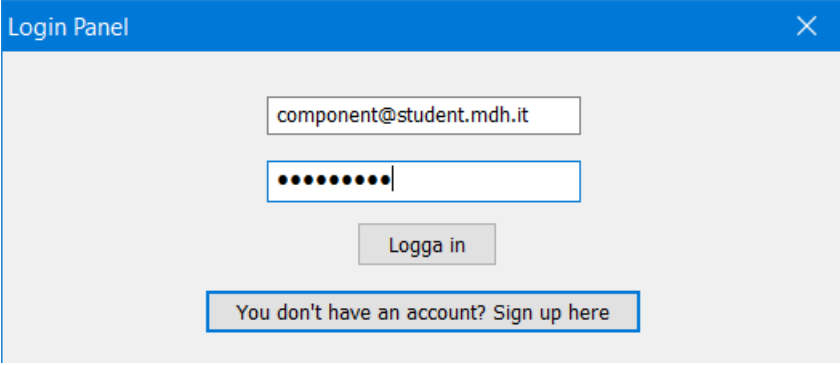




The image shows a 'Register' dialog box with a blue title bar and a close button (X) in the top right corner. The dialog contains three text input fields: 'Username', 'Password', and 'Repeat Password'. Below these fields are two buttons: 'Register' and 'Cancel'.

*Figure 2 - Register form*

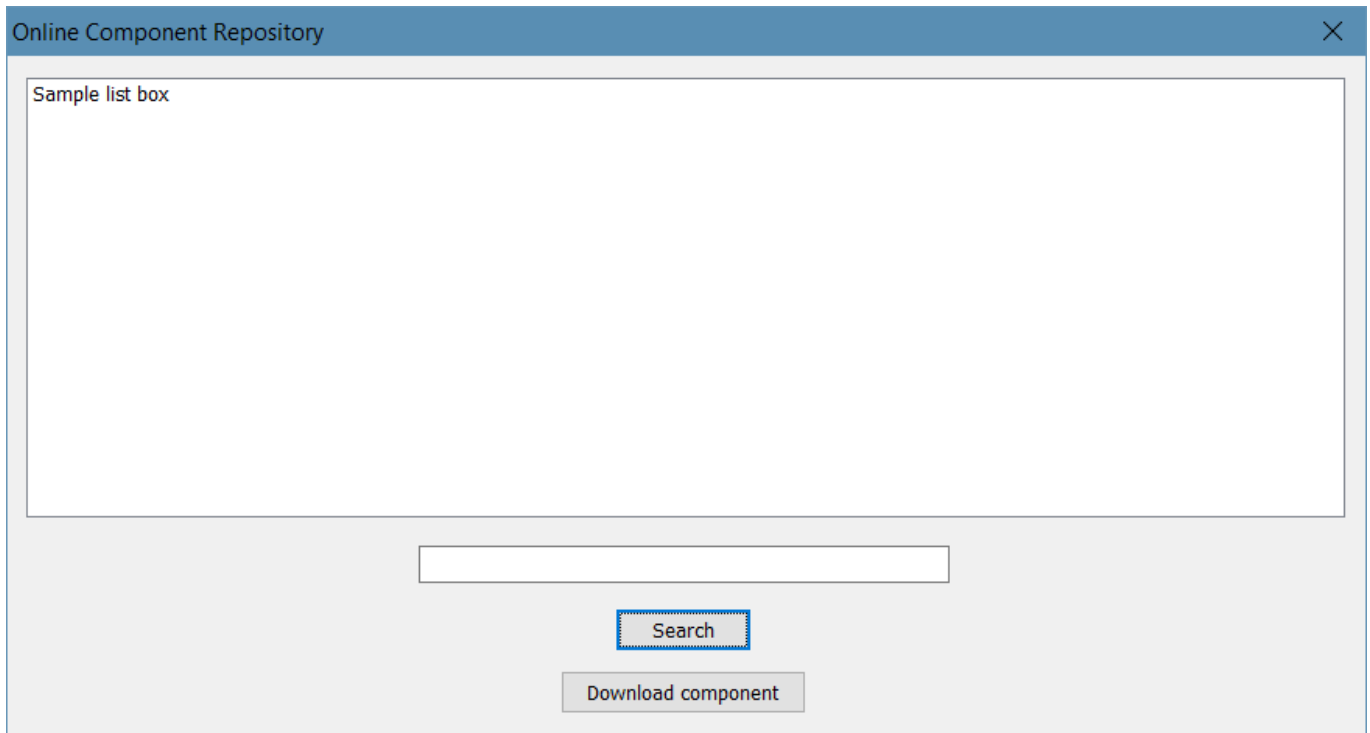
In the *Figure 2* is showed the register form: this dialog will appear every time a new user need to register in the system. There are 3 simple text box: one for the username and two for the password. Of course, the user can then register through the *Register button* or cancel the registration by closing the dialog or by pushing the *Cancel button*.



The image shows a 'Login Panel' dialog box with a blue title bar and a close button (X) in the top right corner. The dialog contains two text input fields: the first contains the email 'component@student.mdh.it' and the second contains a masked password represented by ten dots. Below these fields is a 'Logga in' button. At the bottom of the dialog is a link that says 'You don't have an account? Sign up here'.

*Figure 3 - Login form*

Through this dialog, a user can log in the system by inserting his credentials (username and password) or creating a new account by clicking the second button.



*Figure 4 - Main user interface*

This is the main interface for the user: there is a big listbox in which ALL THE COMPONENTS are showed, than a little textbox in which the user can insert the name of a component and search for it. If a component is selected, the user can press the *Download Component button* in order to download it

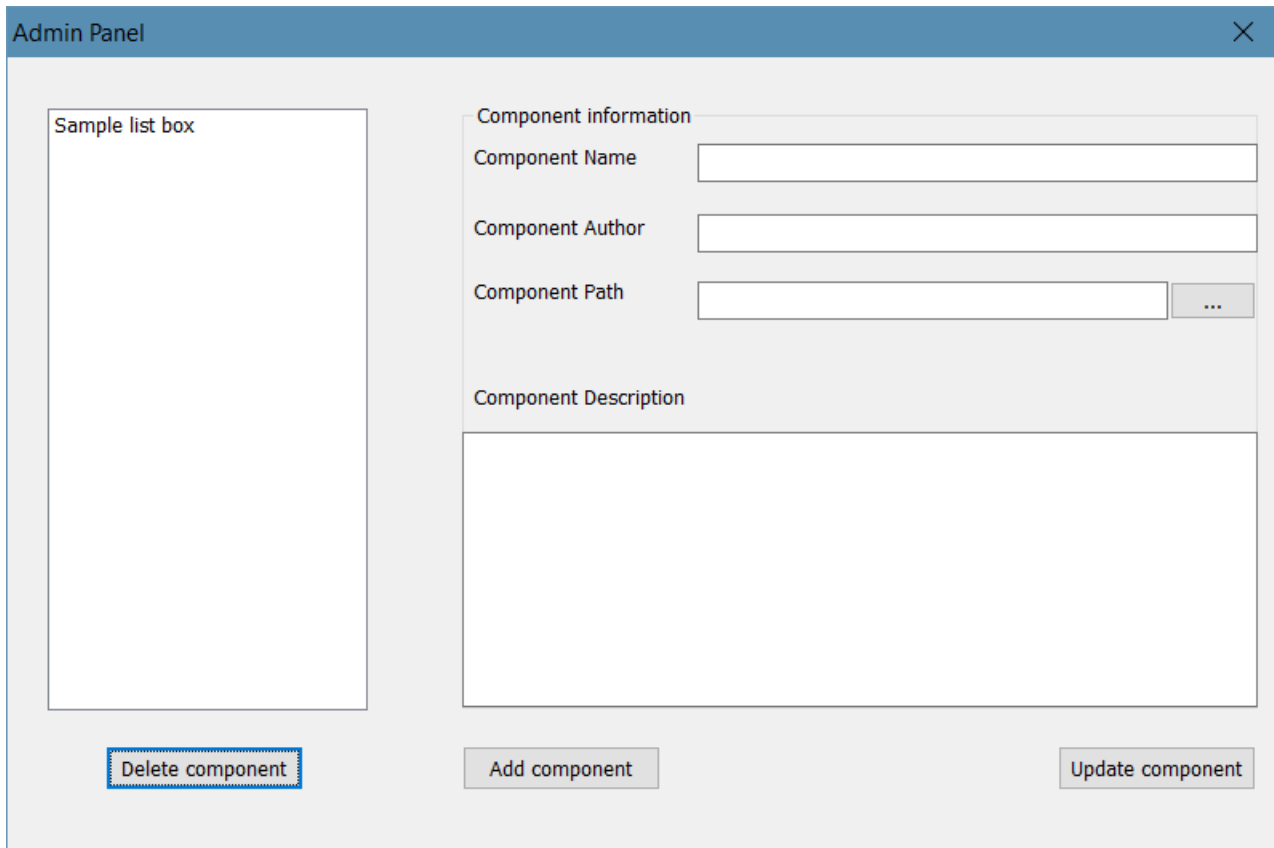
The image shows a software window titled "Admin Panel" with a close button (X) in the top right corner. The window is divided into two main sections. On the left, there is a large, empty rectangular box labeled "Sample list box". Below this box is a button labeled "Delete component". On the right, there is a section titled "Component information" which contains three input fields: "Component Name", "Component Author", and "Component Path". The "Component Path" field has a small "..." button to its right. Below these fields is a larger text area labeled "Component Description". At the bottom of the right section, there are two buttons: "Add component" and "Update component".

Figure 5 - Admin Panel

If a user is an administrator of the system, then he can access this panel. On the left, a *list box* shows every component registered in the system. If the admin selects one of them and clicks the *Delete component* button, then the component is deleted from the system. If a component is selected, then every information regarding it are showed in the textboxes on the right and the admin can update them by pushing the *Update component button*. If no component is selected, then the admin can fill every textbox in order to register a new component by pressing the *Add component* button.

## 4 Software Architecture

### 4.1 Overview and Rationale

*Help* Give a short overview of the architecture and its rationale. Mention any architectural styles or strategies you have used.

### 4.2 System Decomposition

*Help* Present the decomposition into components and the purpose of each.

#### 4.3 Hardware/Software Mapping

*Help*      *Describe how components are assigned to hardware and off-the-shelf software components.*

#### 4.4 Persistent Data

*Help*      *Describe what persistent data is managed by the system and how this is done.*

#### 4.5 Access Control

*Help*      *If relevant.*

#### 4.6 Synchronization and Timing

*Help*      *If relevant.*

#### 4.7 Start-Up and Shut-Down

*Help*      *This is always relevant.*

#### 4.8 Error Handling

*Help*      *This is always relevant.*

### 5 Detailed Software Design

#### 5.1 </Component Name>

*Help*      *Add a section for each component. Add subsections as appropriate in each case.*

##### 5.1.1 Static Structure

*Help*      *Show the classes in the subsystem and relationships between them using a UML class diagram. Describe the purpose of each class briefly.*

##### 5.1.2 Dynamic Behaviour

*Help*      *Describe object interactions using UML sequence diagrams. You may also describe behaviour of nontrivial classes using state diagrams.*

REVISION

Rev. Ind.	Page (P) Chapt.(C)	Description	Date Initials