Mälardalen University
School of Innovation, design and engineering
Västerås, Sweden

Component Technologies - 7.5 hp - CDT401

COMPONENT REPOSITORY

Design Description

Anton Roslund
ard15003@student.mdh.se

Cécile Cayèré
cce18001@student.mdh.se

Milos Ojdanic
moc16001@student.mdh.se

Stanislas Pedebearn
spn18013@student.mdh.se

Vincenzo Stoico
vso18003@student.mdh.se

October 4, 2018

Contents

1 Introduction

1.1 Background

The purpose of this project is to provide a component repository.

There will be two types of users. Administrators should be able to add, modify and delete components. Normal users should be able to see, access and download components. This project goes in continuity of the development of different components codes in different languages and using different frameworks. This components will be in the repository and users will be able to access them and call various operations from them.

Grouping various components into one repository is useful because it allows a user to find the services they need in one place. The final solution for the software will be intended for programmers that need a component for their own project.

1.2 Definitions

Terms	Definitions
Component	A component is an element of a software that can be deployed independently and can be reuse easily. It have interfaces...
Interface	An interface delimits two components exchanging informations among them.
Repository	A repository allows to group various items.
Use Case Model	List of actions or events defining the interactions between a role and a system to achieve a goal.

1.3 Related Documents

Document identity	Document title
Requirements.pdf	Requirements Description

2 Functional Description

2.1 Use Case Model

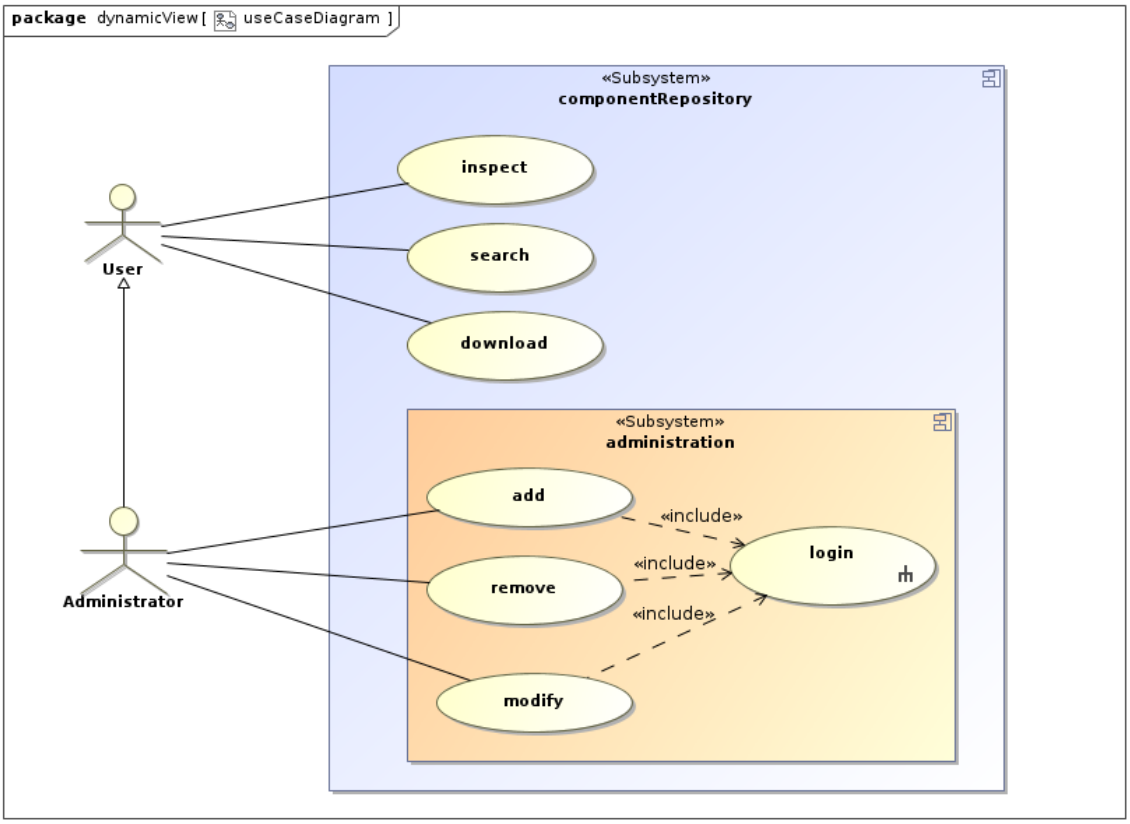

Figure 1: Use Case Diagram

2.1.1 Actors

- User: This actor represents the general concept of user embracing all the entities that can perform an action on the system (e.g. visitor, admin, script)
- Administrator: This actor represent the entity that manages the system. An admin does significant changes to the repository (e.g. deletion, addition).

2.1.2 Use Cases

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

2.2 See List of Components

Initiator : User.
Goal : The system presents the user with the list of all available components

Main Flow of Events :

1. User request the list of components.
2. The system gets list of all available components .
3. The system presents the list of components.

Extensions :

2. There are no components.
   (a) The system displays message that there are no available components.
   (b) Scenario ends.

2.3 Download Component

Initiator : User.
Goal : The user download a requested component.

Main Flow of Events :

1. User request to download a component.
2. The download starts.

Extensions :

1. The component does not exist.
   (a) The system displays error message.
   (b) Scenario ends.

2.4 Modify Component Metadata

Initiator: Administrator.
Goal : The administrator modify a component of the repository.

Main Flow of Events :

1. The administrator requests a modification on a component.
2. The system modify the component.
3. The system sends a notification to the administrator.

Extensions :

1. The component does not exist.
   (a) The system displays error message.
   (b) Scenario ends.

2.5 Remove Component from Repository

Initiator : Administrator.
Goal : The admin remove a component of the repository .

Main Flow of Events :

1. Admin requests to delete the component.
2. The system deletes the component.
3. The system sends a notification.

Extensions :

1. The component does not exist
   (a) The system display error message.
   (b) Scenario ends.

2.6 Add Component to Repository

Initiator : Administrator.
Goal : The administrator add a component to the repository.

Main Flow of Events :

1. The administrator requests a adding of a component on the repository.
2. The system add the component.
3. The system sends a notification to the administrator.

Extensions :

1. Constraints are not met.
   (a) The system display error message.
   (b) Scenario ends.

3 External Interfaces

3.1 Graphical User Interface

4 Software Architecture

4.1 Overview and Rationale

4.2 System Decomposition

4.3 Hardware/Software Mapping

4.4 Persistent Data

4.5 Access Control

4.6 Synchronization and Timing

4.7 Start-Up and Shut-Down

4.8 Error Handling

5 Detailed Software Design

5.1 Subsection for each