*Florida International University*

*School of Computing and Information Sciences*

CIS 4911 Senior Capstone Project

Test Case Management System

Early Draft Requirements Document (RD)

September 2014

**Group Members**

Justin Phillips  
Karina Harfouche

**Mentors**Tariq King  
Dionny Santiago

**Instructors**Masoud Sadjadi  
Juan Caraballo

**Requirements Document (RD)**

**The bullets represent the sections before chapter 1 “Introduction”, numbers on the left represent the corresponding chapters and sections.**

· Copyright and trademark notices, restrictions on copying or distributing the documentation, information for contacting the issuing organization (reader’s comments), warranties, contractual obligations or disclaimers, and general warnings and cautions.

·

**Abstract**

Ultimate Software’s testing team manages large sets of test cases using Microsoft Test Manager (MTM) in conjunction with Microsoft’s Team Foundation Server (TFS), several shortcomings have been found in MTM . The current software testers’ experience when creating, editing and managing the tasks related to testing in general could improve in order to significantly and directly enhance the time it takes to fully test software as well as the overall assertiveness and ease of spotting blocking issues that ultimately affect the customer’s satisfaction within a given software product. Therefore a test case management and automation application has been solicited from Ultimate Software which will provide a friendly, interoperable, experience with added features such as metrics to the test case management suite of Ultimate Software.

The present requirements document consists of an introduction and overview of the system along with the detailed requirements elicited. The requirements provide the functional and nonfunctional capabilities which are realized by use cases, UML, and other artifacts as specified by Ultimate Software. The requirements analysis describes the relationships and dependencies of each feature and functionality covers all aspects of the current solution the members of this project. The depiction of clear definitions and UML diagrams of Ultimate Software’s requirements are the core of this document and guide the engineering process towards the implementation of a correct and complete software solution. The human, hardware, software, and other resources estimated on the planning section of this deliverable are in proportion to the system’s functional and nonfunctional requirements.

**<this last sentence is hanging, we should review again once the doc is complete and make a more concise abstract which follows the flow of the overall document>**

Table of Contents

**1. Introduction**

The current chapter presents a general overview of the Test Case and Automation Management software. Initially, the problem definition of the project is depicted in order to provide a context that leads to the explanation, purpose and scope of the proposed system. Prominent definitions, acronyms and abbreviations used in this document are defined in section 1.3. In conclusion, the final section will encapsulate a complete overview and main points discussed in this document.

**1.1. Problem Definition.**

Software testing is a core decision stage within a functional software’s development and operation life cycle, testing determines the correctness, completeness, and overall compliance with a client’s requested features and requirements. Companies such as Ultimate Software depend heavily on efficient and practical management of software testing in order to confidently validate their software products with their customer requirements. While Ultimate Software’s testing team manages large sets of test cases using Microsoft Test Manager (MTM) in conjunction with Microsoft’s Team Foundation Server (TFS), several shortcomings have been found in MTM . The current software testers’ experience when creating, editing and managing the tasks related to testing in general could improve in order to significantly and directly enhance the time it takes to fully test software as well as the overall assertiveness and ease of spotting blocking issues that ultimately affect the customer’s satisfaction within a given software product.

The Test Case and Automation application will help overcome some of the difficulty with software testing. The application seeks to aggregate test data from multiple systems or platforms, various code repositories, and a notification dashboard to create a more effective and efficient test environment.

Testers require the ability to very quickly and easily create a new test project, link test cases to their associated code repository files, and notify the tester to changes in these files. The application should be able to allow for the scheduling and running of automation. The application will be work across OS and be able to support multiple test case management systems systems, in addition to code repositories.

**1.2. Scope of system.**

The proposed Test Management system’s requirements and development will be solely based and personalized on the needs and specifications of the Solution Testing Team at Ultimate Software. As the requirements and use case sections of this document specifies, the scope of the system includes the general tasks involved with the creation, edit, update and deletion of test case documents coupled with providing a mechanism for persistent storage of the information required in addition to interoperability TSF,MTM, and expandability for other management systems. The system shall be responsible for synchronizing its own model with 3rd party management systems and maintaining overall integrity of the data wherein. The system shall also include metrics for test case failure rates, failure types, and tie these metrics into suites and code files. These metrics should be displayed in a graphical, easy to read format. The automation aspect of the system will be pursued only if permissible by the time constraints and resources for this project. In the case that time constraints impede the full development of the application’s ability to automate test cases, analysis and proper planning will still be made accordingly in order to develop an application that allows the addition of an automation component.

**1.3. Terminology - Definitions, acronyms, and abbreviations.**

**COCOMO:** Constructive Cost Model

**MTM:** Microsoft Test Manager

**SCM:** Source Code Management

**SW**: Software

**TFS:** Team Foundation Server

**WA:** Work Activity

**1.4. Overview of document – brief explanation of what to expect in chapters 2 through 6**

This document incorporates the early definitions of the project necessary to complete an accurate requirements analysis for the development of the Test Case Management System. The primary objective of this document is to present the project’s overview in this chapter as well as an in depth analysis of the system requirements. The second chapter of this document provides an overview of the current system. Within chapter 3, the software project’s development and management organization are described in detail. Chapter 4 describes the system requirement analysis and supporting artifacts. Chapter 5 contains the glossary of project related terms. To conclude the appendix provides many of the initial project artifacts which support the system requirements analysis.

**2. Current System**

The current system, within the scope of this project, is centered around MTM utilized in conjunction with TFS. These applications and services are used to create test suites, cases, and steps in addition to providing autonomous build, test, and deployment services; as well as reporting services. Also, MTM supports manual testing via reporting and rich media capturing capabilities. The system provides for the ability to then query reports and tests.

The most major limitation of the current system is the functionality of manipulating test elements (suites, cases, and steps), such as copying these elements or categorizing whither they are high-level or implementation elements. Additionally, the current system does not support displaying metrics on failure types or interoperability with test elements from other management systems.

The usability limitations exist within the structural limitations of test cases, drill-down navigation, and the complexity/limited feature set of the user interface. The client would like the ability to create two levels of test cases, one that is high level and one at an implementation details level; which the current system does not support. MTM provides a drill-down navigation of test suites and cases that allow for attachments to be added. However, viewing these attachments using the drill-down navigation creates for an arduous process. Lastly, in many areas test case creation is too complex, with too busy of a user interface while also not providing necessary features such as copying of cases and steps.

Another limitation of the current system is that MTM cannot associate test suites, cases, or steps with the code repository. Thus, there does not exist a method for which the tester can be notified about file changes associated with a given test element.

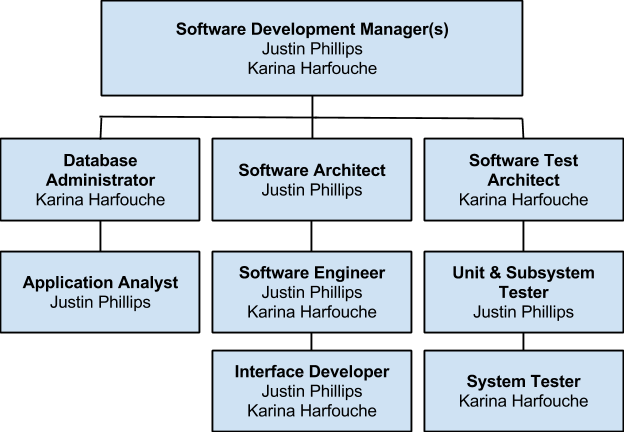
Finally, MTM does not provide a mechanism in which the results of automated test may be displayed. This provides for a fragmented testing system which requires multiple applications to determine the state of the tests and displaying test associations.

**3. Project Plan (This deliverable only)**

The project organization section focuses on the visualization and analysis of the general software organization as well as the software support and service organization. A hierarchical chart will be presented in order to illustrate the proposed structure for the Test Case Management System’s working personnel

**3.1. Project organization**

This is the software development organization for the Test Case Management System:



**3.2. Work breakdown – identification of milestones and deliverables (refer to project schedule in Appendix A and the diary in appendix B).**

The following table identifies the major tasks, milestones and deliverables along with each respective duration dates:

|  |  |  |  |
| --- | --- | --- | --- |
| **Task Name** | **Start Date** | **End Date** | **Duration (days)** |
| **Requirement Elicitation** | **09/04/14** | **09/15/14** | **8** |
| **Meet with client** | **09/04/14** | **09/04/14** | **1** |
| **Determine Requirements** | **09/04/14** | **09/15/14** | **8** |
| **System Analysis** | **09/05/14** | **09/15/14** | **7** |
| **Determine System Feasibility** | **09/05/14** | **09/10/14** | **4** |
| **Compare with Alternatives** | **09/10/14** | **09/15/14** | **4** |
| **Present Deliverable** | **09/15/14** | **09/15/14** | **1** |
| **Software Project Planning** | **09/26/14** | **09/30/14** | **3** |
| **Scheduling** | **09/26/14** | **09/26/14** | **1** |
| **Resource Allocation** | **09/27/14** | **09/27/14** | **1** |
| **Risk Plan** | **09/27/14** | **09/30/14** | **3** |
| **Software Project Organization** | **09/29/14** | **10/06/14** | **6** |
| **Personnel Structurization** | **09/29/14** | **09/29/14** | **1** |
| **Obtaining Tools** | **10/01/14** | **10/01/14** | **1** |
| **Define Metrics** | **10/02/14** | **10/03/14** | **2** |
| **Acceptance of Project Organization** | **10/04/14** | **10/06/14** | **2** |
| **System Design** | **10/06/14** | **10/10/14** | **5** |
| **Create Use Cases** | **10/06/14** | **10/06/14** | **1** |
| **Subsystem Decomposition** | **10/07/14** | **10/07/14** | **1** |
| **Determine Security and Privacy** | **10/08/14** | **10/08/14** | **1** |
| **Create DB Schema** | **10/09/14** | **10/09/14** | **1** |
| **Hardware & Software Mapping** | **10/10/14** | **10/10/14** | **1** |
| **Object Design** | **10/12/14** | **10/24/14** | **11** |
| **Create Class Diagram** | **10/12/14** | **10/12/14** | **1** |
| **Create OCL statements** | **10/13/14** | **10/13/14** | **1** |
| **Create Sequence Diagram** | **10/24/14** | **10/24/14** | **1** |
| **System Implementation** | **10/10/14** | **11/13/14** | **25** |
| **Create/Configure Database** | **10/10/14** | **10/14/14** | **3** |
| **Implement User Interface** | **10/14/14** | **10/24/14** | **9** |
| **Implement System Logic** | **10/25/14** | **11/13/14** | **15** |
| **Present Deliverable** | **10/28/14** | **10/29/14** | **2** |
| **Testing** | **11/13/14** | **12/02/14** | **14** |
| **Create & Implement Test Cases** | **11/13/14** | **11/17/14** | **3** |
| **Run test suites** | **11/13/14** | **11/19/14** | **5** |
| **Evaluate & Fix Defects** | **11/19/14** | **12/02/14** | **10** |
| **Present Final Deliverable** | **12/15/14** | **12/16/14** | **2** |
| **Present Deliverable to Client** | **12/15/14** | **12/15/14** | **1** |
| **System Release** | **12/16/14** | **12/16/14** | **1** |

**3.3. Cost Estimate – cost to develop the software system.**

**4. Proposed System Requirements**

The new system shall provide the client with the ability to create test suites, cases, and steps efficiently. The system shall interface with TFS and be configurable for both TFS communication and code repository linking, in addition to other 3rd party management systems. Furthermore, the system shall allow test elements to be associated with a code repository and be defined as either high level or implementation elements. The system shall also support adding attachments to test elements. Test elements will be copyable and also expose a set of metrics about test case failure rates. This entire system shall be secured behind a single sign-on login which is linked to domain accounts. Each feature within the proposed system shall provide a easy to use, simple interface to the system actors.

**4.1. Functional Requirements**

The implementation of the Test Management System will include ten main functionalities that the system shall perform:

1. The system shall allow for testers within the configured Windows domain to have single sign-on access using the account the tester is signed into a Windows machine with.
2. The system shall provide a mechanism to configured the test management backend system.
3. The system shall provide an interface for creation of test suites, cases, and steps with fluid, intuitive authoring experience.
4. The system shall provide testers with the ability to add attachments and associate them with test suites, cases, and steps. The attachments should then be viewable with ease and with as few clicks as possible.
5. The system shall provide a mechanism to associate test suites, cases, and steps with a code repository or automation.
6. The system shall provide an interface to configure the interaction with a code repository.
7. The system shall display test suites, cases, and steps in a clean explorer or tree view throughout the user navigation within the system.
8. The system shall provide a means to display data categorically, organically, and in a logical view.
9. The system shall provide a simple search interface for searching the test suites, cases, and steps.
10. The system shall provide a logout mechanism so that the current user may be logged out and a user other than the user logged into the current Windows machine may login to the system.
11. The system shall be secured behind the login mechanism. Guests will only be able to visit the login feature.
12. The system shall provide an access table with a list of approved users from within the domain user group, this access table shall be configurable by pre-defined users on an individual basis.
13. The system shall provide functionality for extending the test element model to other 3rd party management systems.
14. The system shall provide a set of metrics which track test case failures, failure types, and source code attached to failures.
15. The system shall provide a visual representation of the failure metrics.

**4.2. Analysis of System Requirements**

This chapter covers the analysis of the system requirements and presents the findings in UML diagrams. The UML diagrams referenced in this chapter are provided in Appendix B through Appendix D.

**4.2.1. Scenarios**

**4.2.2. Use case model**

The use case model is described and detailed in diagrams within the Appendix B. The actors presented are the user, which is a stakeholder in an application’s test or development phases. The code repository actor represents a repository system such as GIT or SVN. The login server represents a Windows Domain server which processes login details. Finally, the 3rd Party Management System actor is a test case management system which stores test elements associated with a project within the system.

**4.2.3. Static model**

The initial UML diagrams and models are depicted in Appendix C. These diagrams include a minimal class diagram, figure 6.1 and a deployment diagram, figure 6.2. The minimal class diagram realizes the scope of the source code to be developed for the system, while the deployment diagram details the scope of the overall system and components.

**4.2.4. Dynamic model**

The initial dynamic UML diagrams are depicted in Appendix D. These artifacts capture the more complex components of the proposed system. They will be refined during the design stage and published in the design document.

**5. Glossary - define terms used in document, especially domain specific terms.**

**6. Appendix**

**6.1. Appendix A - Complete use cases**

**USE CASE ID:** TS001 – Test Case & Automation Manager – Log In

**Level:** High Level

**Details:**

**Actors** - Tester, Administrator, Guest

**Preconditions**

1. Actor is logged in a Windows machine providing a Windows account to the Test Manager
2. Actor has accessed the web application’s login page

**Description**

1. The use case begins when the actor enters and submits login username and password. (Trigger)
2. The system responds by validating the username and password.
3. The use case ends when the account’s main page is succesfully loaded and displayed.

**Post-Conditions**

1. The Actor gets access to the Test Case & Automation Management page.

**Exceptions**

System is down.

**Decision Support:**

Frequency – The most frequent type of actor is the teste who will log into the system daily from Monday to Friday.

Critically – High, allows all actors to log into the system.

Risk – Medium

**Constraints:**

* Single Sign On is required, the user must be able to sign in with a windows account only.
* Username has to start with a letter character.
* Username characters may contain lowercase & uppercase letters, numbers and underscores.

**Usability:**

* No previous training time required

**Reliability:**

* Mean time to failure – 1% failures for every twenty four hours of operation is acceptable.
* Availability – Down time for login back-up 10 minutes in a twenty four hour period.

**Performance:**

* On average, the acotr should be able to login to the system in less than 30 seconds.

**Supportability:**

* System should deliver clear request for the second check point.

**Modification History:**

Owner: TC Senior Project  
Initiation Date: 9/16/2014

Date last modified: 9/16/2014



**USE CASE ID:** TS002 – Test Case & Automation Manager – Create Test Suite

**Level:** High Level

**Details:**

**Actors** - Tester

**Preconditions**

1. Tester is logged in the test manager
2. Actor has accessed the Test Case page

**Description**

1. The use case begins when the tester clicks on add new Test Suite. (Trigger)
2. The system responds by loading a pop-up form requiring the Test Suite information.
3. The tester fills the form with the Test Suite name, description and other relevant information
4. The use case ends when the tester clicks OK to add the suite with the provided information

**Post-Conditions**

1. The Actor gets access to the new suite on the hierarchical menu list.
2. The Actor may now add test cases to this suite.

**Exceptions**

A test suite with the provided name has already been created

**Decision Support:**

Frequency – Daily from Monday to Friday.

Critically – High, allows tester to add suites.

Risk – Medium

**Constraints:**

* Test Suite file name should be created with acceptable ASCII characters.
* Tester creating test suite should have privilege granted.
* Test Suite name should be unique.

**Usability:**

* No previous training time required

**Reliability:**

* Mean time to failure – 1% failures for every twenty four hours is acceptable.
* Availability – Always.

**Performance:**

* On average, the system should be able to create a new Suite in less than 20 seconds after tester’s request.

**Supportability:**

* System should deliver clear request.

**Related Use Cases:**

**Modification History:**

Owner: TC Senior Project  
Initiation Date: 9/16/2014

Date last modified: 9/16/2014



**USE CASE ID:** TS003 – Test Case & Automation Manager – Create Test Case

**Level:** High Level

**Details:**

**Actors** - Tester

**Preconditions**

1. Tester is logged in the test manager
2. Actor has accessed the Test Case page
3. Actor has chosen the Test Suite where he wants to create a new test case

**Description**

1. The use case begins when the tester clicks on add new Test Case. (Trigger)
2. The system responds by loading a pop-up form requiring the Test Case information.
3. The tester fills the form with the Test Case name, description and other relevant information
4. The use case ends when the tester clicks OK to add the case with the provided information

**Post-Conditions**

1. The Actor gets access to the blank document screen where test case description and steps will be added
2. The new test case will be added on the hierarchical menu list under its correspondent test suite.
3. The Actor may now access and edit this test case.

**Exceptions**

A test case with the provided name has already been created under this test suite

**Decision Support:**

Frequency – Daily from Monday to Friday.

Critically – High, allows tester to add suites.

Risk – Medium

**Constraints:**

* Test Case name should be created with acceptable ASCII characters.
* Tester creating test case should have privilege granted.
* Test Case name should be unique.

**Usability:**

* No previous training time required

**Reliability:**

* Mean time to failure – 1% failures for every twenty four hours is acceptable.
* Availability – Always.

**Performance:**

* On average, the system should be able to create a new case in less than 20 seconds after tester’s request.

**Supportability:**

* System should deliver clear request.

**Modification History:**

Owner: TC Senior Project  
Initiation Date: 9/16/2014

Date last modified: 9/16/2014



**USE CASE ID:** TS004 – Test Case & Automation Manager – Create Test Steps

**Level:** High Level

**Details:**

**Actors** - Tester

**Preconditions**

1. Tester is logged in the test manager

2. Tester has created a test case

3. Tester has chosen the test case to be added steps

**Description**

1. The use case begins when the tester clicks on edit test case

2. The system responds by loading the corresponding test case’s steps document

3. The user types new steps in the document

4. The use case ends when the tester clicks Save and changes have been saved on the test case

**Post-Conditions**

1. All actors are able to see the newly modified steps

2. A previous version of the test case steps has been backed up

**Exceptions**

N/A

**Decision Support:**

Frequency – Daily from Monday to Friday.

Critically – High, allows tester to add steps.

Risk – Medium

**Constraints:**

* Tester creating steps should have privileges granted.

**Usability:**

* No previous training time required

**Reliability:**

* Mean time to failure – 1% failures for every twenty four hours is acceptable.
* Availability – Always.

**Performance:**

* On average, the system should be able save modified steps in less than 20 seconds after tester’s request.

**Supportability:**

* System should deliver clear request.

**Modification History:**

Owner: TC Senior Project

Initiation Date: 9/16/2014

Date last modified: 9/16/2014



**USE CASE ID:** TS005 – Test Case & Automation Manager –Duplicate Test Steps

**Level:** High Level

**Details:**

**Actors** - Tester

**Preconditions**

1. Tester is logged in the test manager

2. Tester has created a test case

3. Tester has chosen the test case to be added steps

**Description**

1. The use case begins when the tester clicks on edit test case
2. The system responds by loading the corresponding test case’s steps document
3. The user clicks on “Add Existing Steps” option
4. The tester enters on search a test case or opens the test case menu
5. The tester selects the test case where the desired steps exists
6. The test case end when the tester selects the steps and clicks on “Add to current test case”

**Post-Conditions**

1. All actors are able to see the newly modified steps duplicated on the test case chosen

2. A previous version of the test case steps has been backed up

**Exceptions**

N/A

**Decision Support:**

Frequency – Daily from Monday to Friday.

Critically – High, allows tester to add steps.

Risk – Medium

**Constraints:**

* Tester duplicating test steps should have privileges granted.
* Tester should be able to duplicate steps from existing use cases in the current suite

**Usability:**

· No previous training time required

**Reliability:**

· Mean time to failure – 1% failures for every twenty four hours is acceptable.

· Availability – Always.

**Performance:**

· On average, the system should be able save modified steps in less than 20 seconds after tester’s request.

**Supportability:**

· System should deliver clear request.

**Modification History:**

Owner: TC Senior Project

Initiation Date: 9/16/2014

Date last modified: 9/16/2014



**USE CASE ID:** TS006 – Test Case & Automation Manager –Link Test Suite to a Code Repository

**Level:** High Level

**Details:**

**Actors** - Tester

**Preconditions**

1. Tester is logged in the test manager

2. Tester Opened the Test Suites Menu & Selected a Suite

**Description**

1. The use case begins when the tester clicks on Link Test Suite

2. The system responds by loading a list of existing code repositories

3. The user selects a code repository

4. The use case ends when the tester clicks add repository

**Post-Conditions**

1. Changes have been saved on the test suite and the linked repository can be seen under the “Test Suite’s Repository’s List”

2. This repository can be unlinked now from this test suite

3. Tester may now click on this repository and be directly transferred to the code if available for viewing.

**Exceptions**

N/A

**Decision Support:**

Frequency – Daily from Monday to Friday.

Critically – Medium, allows tester to link repositories.

Risk – Medium

**Constraints:**

* Tester should have privileges to access and link code repositories
* Multiple repositories may be linked to a Test Suite

**Usability:**

· No previous training time required

**Reliability:**

· Mean time to failure – 1% failures for every twenty four hours is acceptable.

· Availability – Always.

**Performance:**

· On average, the system should be able save modified links in less than 20 seconds after tester’s request.

**Supportability:**

· System should deliver clear request.

**Modification History:**

Owner: TC Senior Project

Initiation Date: 9/16/2014

Date last modified: 9/16/2014



**USE CASE ID:** TS008 – Test Case & Automation Manager –Add and Link Attachments

**Level:** High Level

**Details:**

**Actors** - Tester

**Preconditions**

1. Tester is logged in the test manager

2. Tester Opened the Test Suites Menu & Selected a Suite & a Test Case

**Description**

1. The use case begins when the tester clicks on “Attach”

2. The system responds by loading an upload window

3. The user selects path to the file that needs to be attached

4. The use case ends when the tester clicks OK.

**Post-Conditions**

1. Changes have been saved on the test case and the attachment has now been added to this test case

**Exceptions**

N/A

**Decision Support:**

Frequency – Daily from Monday to Friday.

Critically – Medium, allows tester to add attachments.

Risk – Medium

**Constraints:**

* Tester should have privileges to add and link attachments
* Tester should be able to add and link any attachments from the local pc

**Usability:**

· No previous training time required

**Reliability:**

· Mean time to failure – 1% failures for every twenty four hours is acceptable.

· Availability – Always.

**Performance:**

· On average, the system should be able save attachments, including compressed videos and folders.

**Supportability:**

· System should deliver clear request.

**Modification History:**

Owner: TC Senior Project

Initiation Date: 9/16/2014

Date last modified: 9/16/2014



**USE CASE ID:** TS009 – Test Case & Automation Manager –View and download Attachments

**Level:** High Level

**Details:**

**Actors** - Tester

**Preconditions**

1. Tester is logged in the test manager

2. Tester Opened the Test Suites Menu & Selected a Suite & a Test Case

**Description**

1. The use case begins when the tester clicks on “Attachments”

2. The system responds by loading the available attachments for that test case

3. The user selects the attachments to be viewed

4. The use case ends when attachment is loaded and the tester clicks on “download”

**Post-Conditions**

1. Attachment has been downloaded to local computer

**Exceptions**

N/A

**Decision Support:**

Frequency – Daily from Monday to Friday.

Critically – Medium, allows tester to download/view attachments.

Risk – Medium

**Constraints:**

* Tester should have privileges to view & download Attachments

**Usability:**

· No previous training time required

**Reliability:**

· Mean time to failure – 1% failures for every twenty four hours is acceptable.

· Availability – Always.

**Performance:**

· On average, the system should be download attachments, including compressed videos and folders.

**Supportability:**

· System should deliver clear request.

**Modification History:**

Owner: TC Senior Project

Initiation Date: 9/16/2014

Date last modified: 9/16/2014



**USE CASE ID:** TS010 – Test Case & Automation Manager –Associate Test to Code Repository

**Level:** High Level

**Details:**

**Actors** - Tester

**Preconditions**

1. Tester is logged in the test manager

2. Tester Opened the Test Suites Menu & Selected a Suite

**Description**

1. The use case begins when the tester clicks on “Associate Suite”

2. The system responds by acquiring from the server all the test repository files available

3. The user selects the repository to be associated

4. The use case ends when tester clicks on “OK”

**Post-Conditions**

1. The code repository chosen is now available as an associated code to the specific test Suite

**Exceptions**

N/A

**Decision Support:**

Frequency – Daily from Monday to Friday.

Critically – Medium

Risk – Medium

**Constraints:**

* Tester should have privileges to associate Test Suite

**Usability:**

· No previous training time required

**Reliability:**

· Mean time to failure – 1% failures for every twenty four hours is acceptable.

· Availability – Always.

**Performance:**

· On average, the system should be able to associate multiple repositories to a single Test Suite

**Supportability:**

· System should deliver clear request.

**Modification History:**

Owner: TC Senior Project

Initiation Date: 9/16/2014

Date last modified: 9/16/2014

**USE CASE ID:** TS011 – Test Case & Automation Manager –Associate Test to Automation

**Level:** High Level

**Details:**

**Actors** - Tester

**Preconditions**

1. Tester is logged in the test manager

2. Tester Opened the Test Suites Menu & Selected a Suite

**Description**

1. The use case begins when the tester clicks on “Associate Suite”

2. The system responds by acquiring from the server all the test repository files available

3. The user selects the repository to be associated

4. The use case ends when tester clicks on “OK”

**Post-Conditions**

1. The code repository chosen is now available as an associated code to the specific test Suite

**Exceptions**

N/A

**Decision Support:**

Frequency – Daily from Monday to Friday.

Critically – Medium

Risk – Medium

**Constraints:**

* Tester should have privileges to associate Test Suite

**Usability:**

· No previous training time required

**Reliability:**

· Mean time to failure – 1% failures for every twenty four hours is acceptable.

· Availability – Always.

**Performance:**

· On average, the system should be able to associate multiple repositories to a single Test Suite

**Supportability:**

· System should deliver clear request.

**Modification History:**

Owner: TC Senior Project

Initiation Date: 9/16/2014

Date last modified: 9/16/2014

**USE CASE ID:** TS012 – Test Case & Automation Manager – View Code Repository

**Level:** High Level

**Details:**

**Actors** - Tester

**Preconditions**

1. Tester is logged in the test manager

2. Tester Opened the Test Suites Menu & Selected a Suite

**Description**

1. The use case begins when the tester clicks on “View Associations”

2. The system responds by acquiring from all available links to repositories and loads options.

3. The user selects the repository for viewing.

4. The use case ends when tester clicks on “View Selection”

**Post-Conditions**

1. The code repository chosen is now visible for the user.

**Exceptions**

N/A

**Decision Support:**

Frequency – Daily from Monday to Friday.

Critically – High

Risk – Medium

**Constraints:**

* Tester should have privileges to view repository

**Usability:**

· No previous training time required

**Reliability:**

· Mean time to failure – 1% failures for every twenty four hours is acceptable.

· Availability – Always.

**Performance:**

· On average, the system should be able to load multiple repositories to select viewing

**Supportability:**

· System should deliver clear request.

**Modification History:**

Owner: TC Senior Project

Initiation Date: 9/16/2014

Date last modified: 9/16/2014

**USE CASE ID:** TS013 – Test Case & Automation Manager – Search

**Level:** High Level

**Details:**

**Actors** - Tester

**Preconditions**

1. Tester is logged in the test manager

2. Tester Opened the Test Suites Menu & Selected a Suite

**Description**

1. The use case begins when the tester click on the search tab

2. The tester types a title or keyword to search for a test case

3. The use case ends when tester clicks on “Search”

**Post-Conditions**

1. The system returns all search hits contained in the Test Suite

**Exceptions**

N/A

**Decision Support:**

Frequency – Daily from Monday to Friday.

Critically – Medium

Risk – Medium

**Constraints:**

* Search is not case sensitive
* Should search matching test cases and suites

**Usability:**

· No previous training time required

**Reliability:**

· Mean time to failure – 1% failures for every twenty four hours is acceptable.

· Availability – Always.

**Performance:**

· The system should be able to return all test cases matching the search

**Supportability:**

· System should deliver clear request.

**Modification History:**

Owner: TC Senior Project

Initiation Date: 9/16/2014

Date last modified: 9/16/2014

**USE CASE ID:** TS014 – Test Case & Automation Manager – Log Out

**Level:** High Level

**Details:**

**Actors** - Tester, Administrator, Guest

**Preconditions**

1. Actor is logged in the Test Manager

**Description**

1. The use case begins and ends when the actor clicks on the logout button

**Post-Conditions**

1. The Actor gets access to the Test Case & Automation Management page.

**Exceptions**

System is down.

**Decision Support:**

Frequency – The most frequent type of actor is the teste who will log into the system daily from Monday to Friday.

Critically – High, allows all actors to log out the system.

Risk – Medium

**Constraints:**

* System should automatically save the current state for the next login

**Usability:**

* No previous training time required

**Reliability:**

* Mean time to failure – 1% failures for every twenty four hours of operation is acceptable.
* Availability – Down time for login back-up 10 minutes in a twenty four hour period.

**Performance:**

* On average, the actor should be able to logut to the system in less than 20 seconds.

**Supportability:**

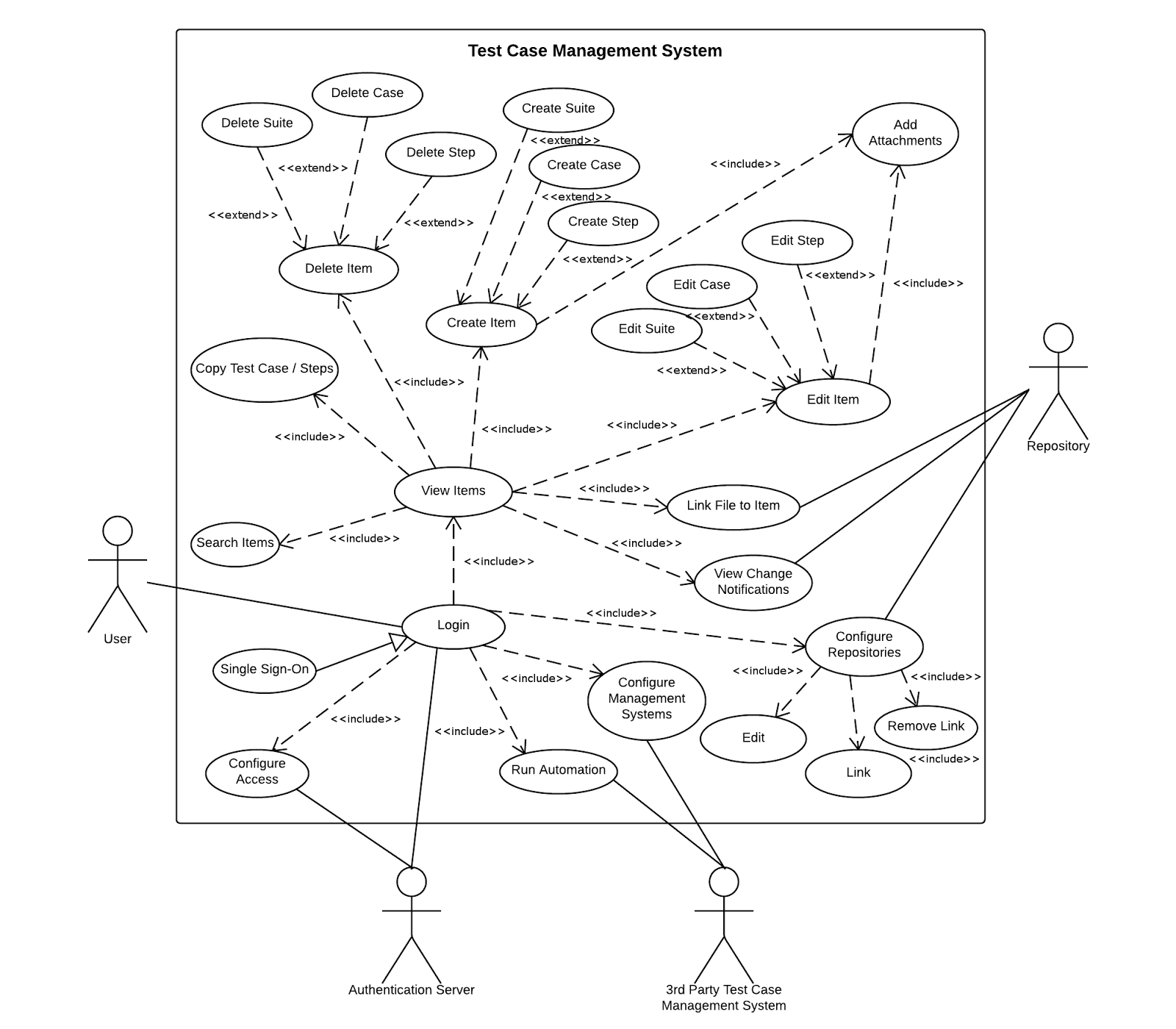
* System should deliver clear request for the second check point.

**Modification History:**

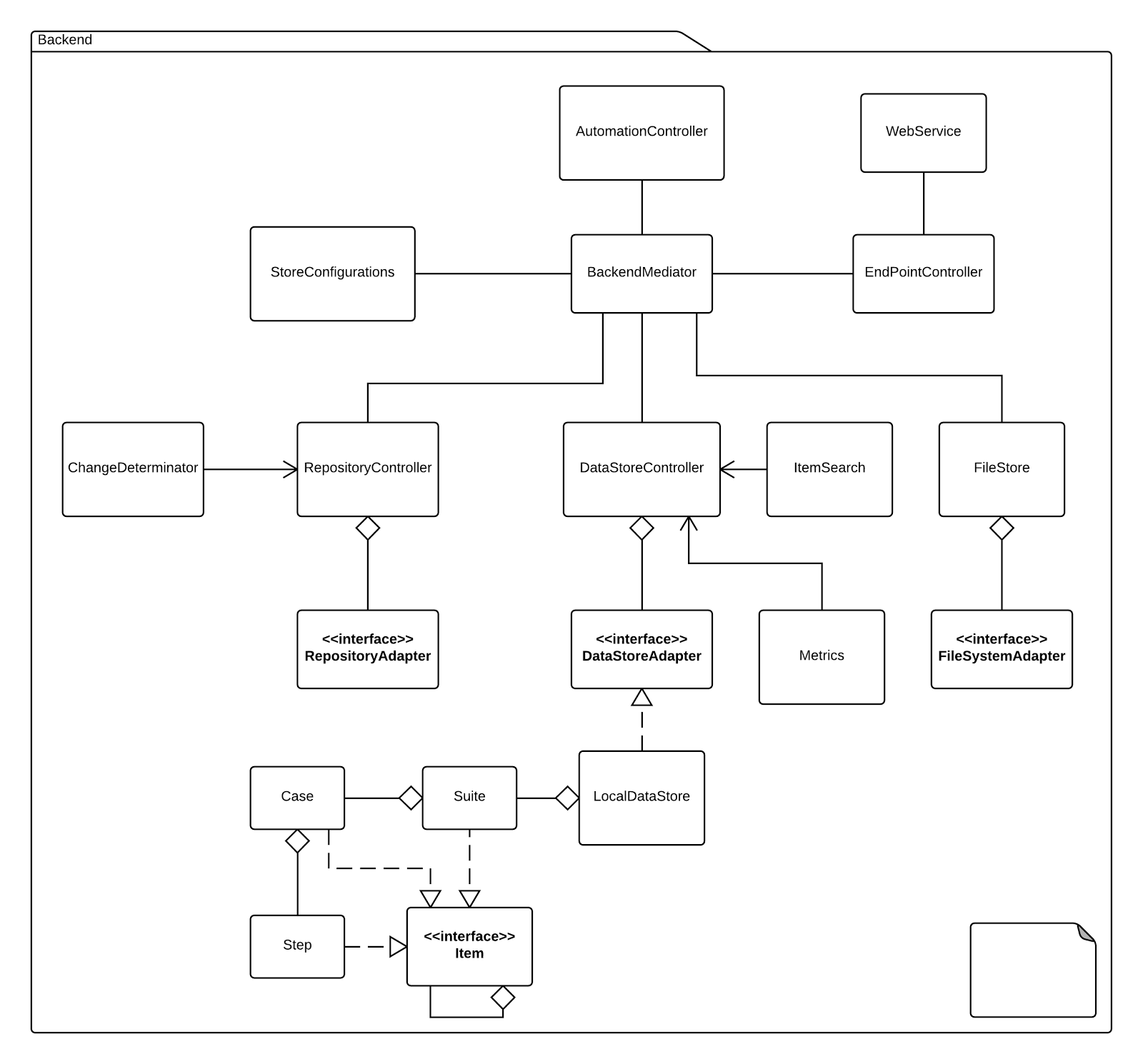
Owner: TC Senior Project  
Initiation Date: 9/16/2014

Date last modified: 9/16/2014

**6.2. Appendix B - Use case diagram using UML**

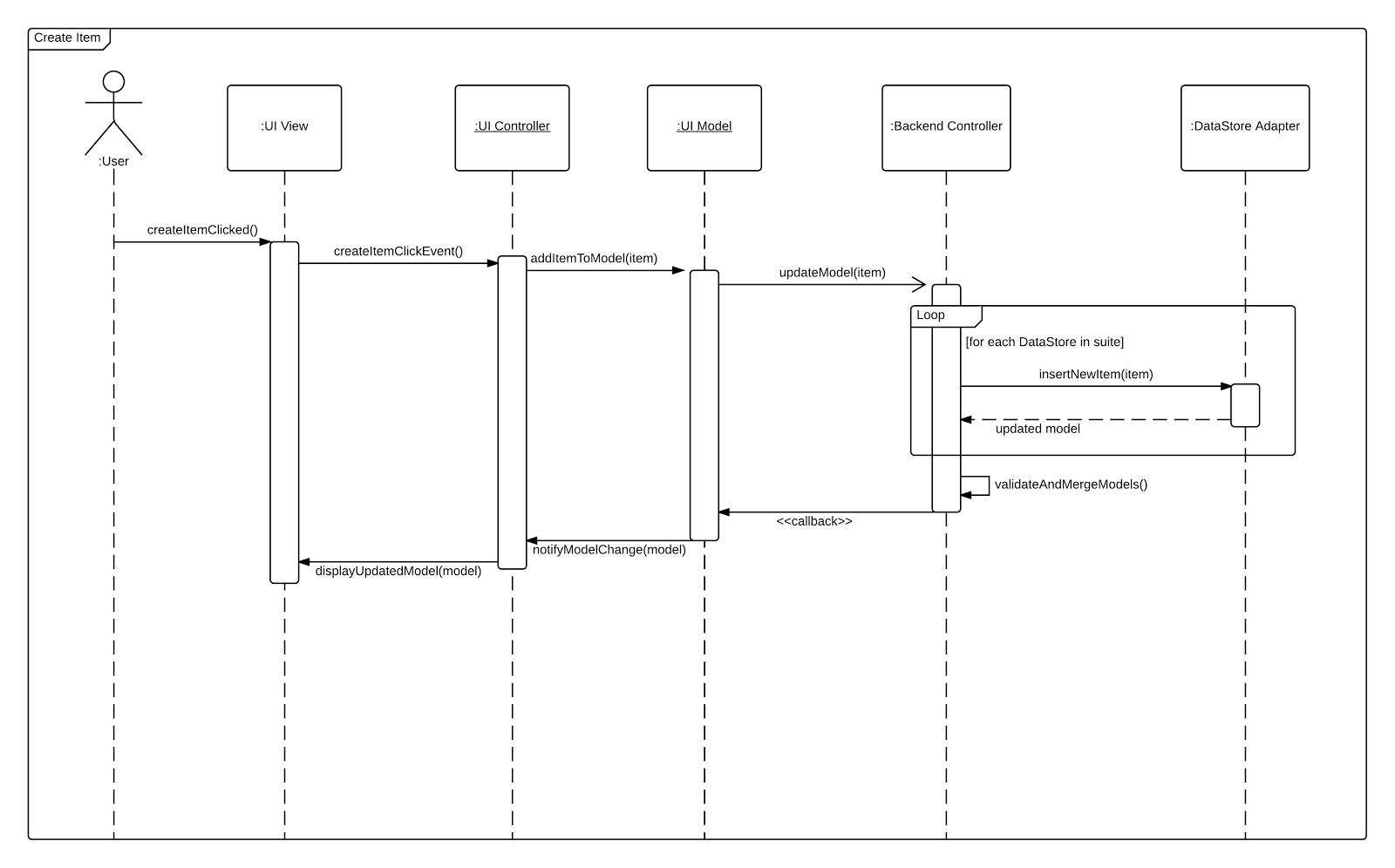


**6.3. Appendix C - Static UML diagram**

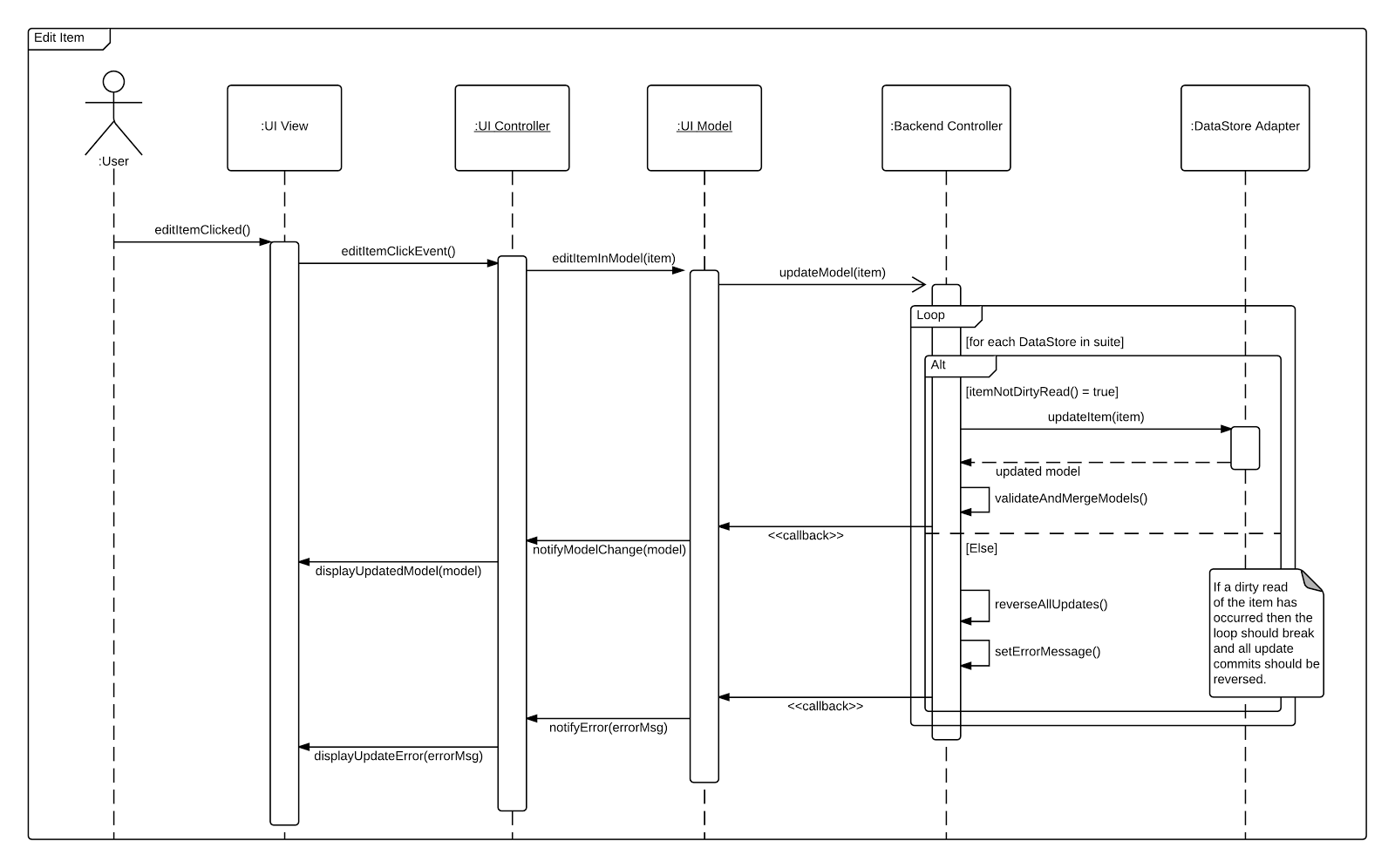


**6.4. Appendix D - Dynamic UML diagrams**

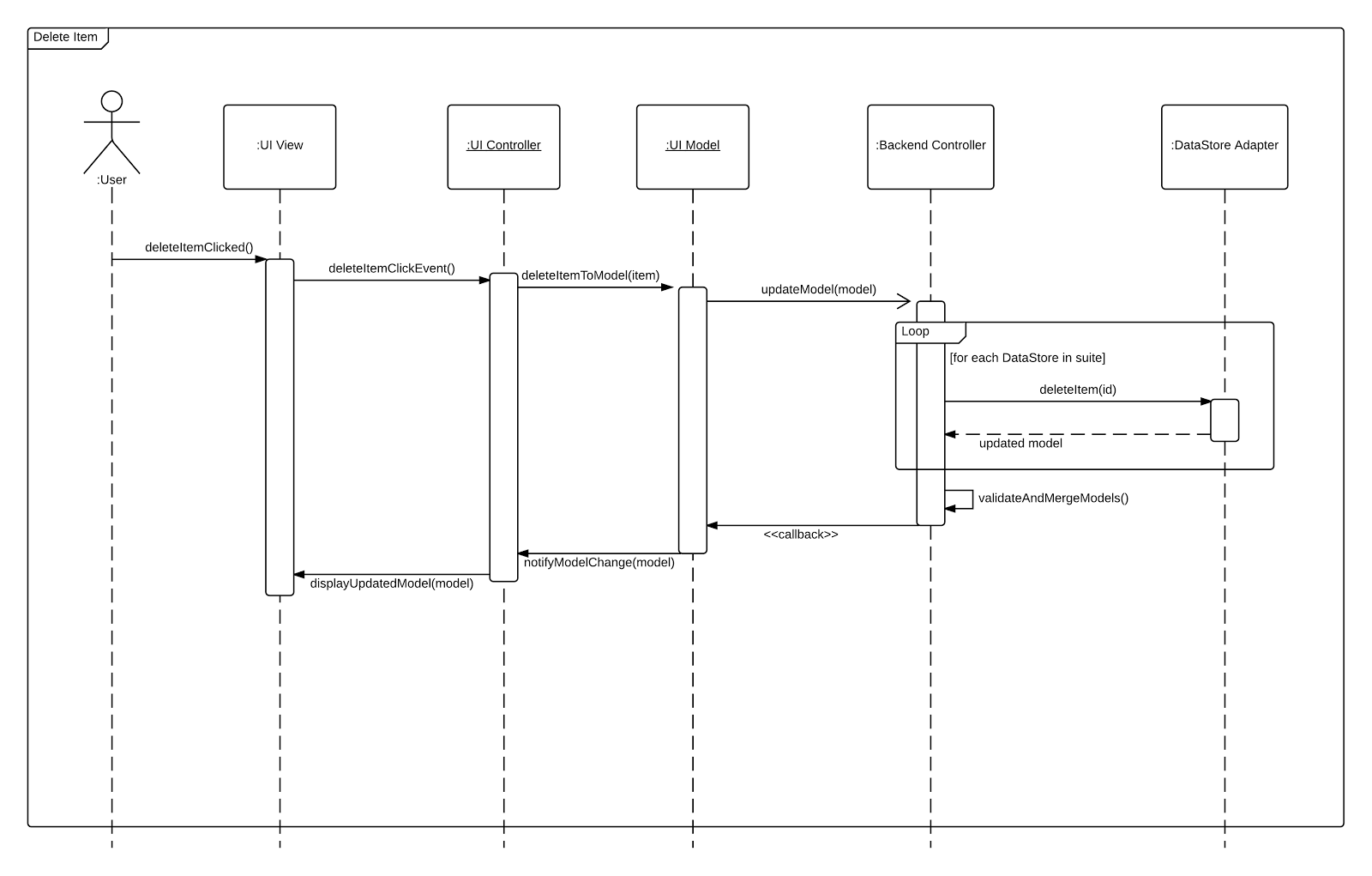
Create Item Use Case Sequence Diagram



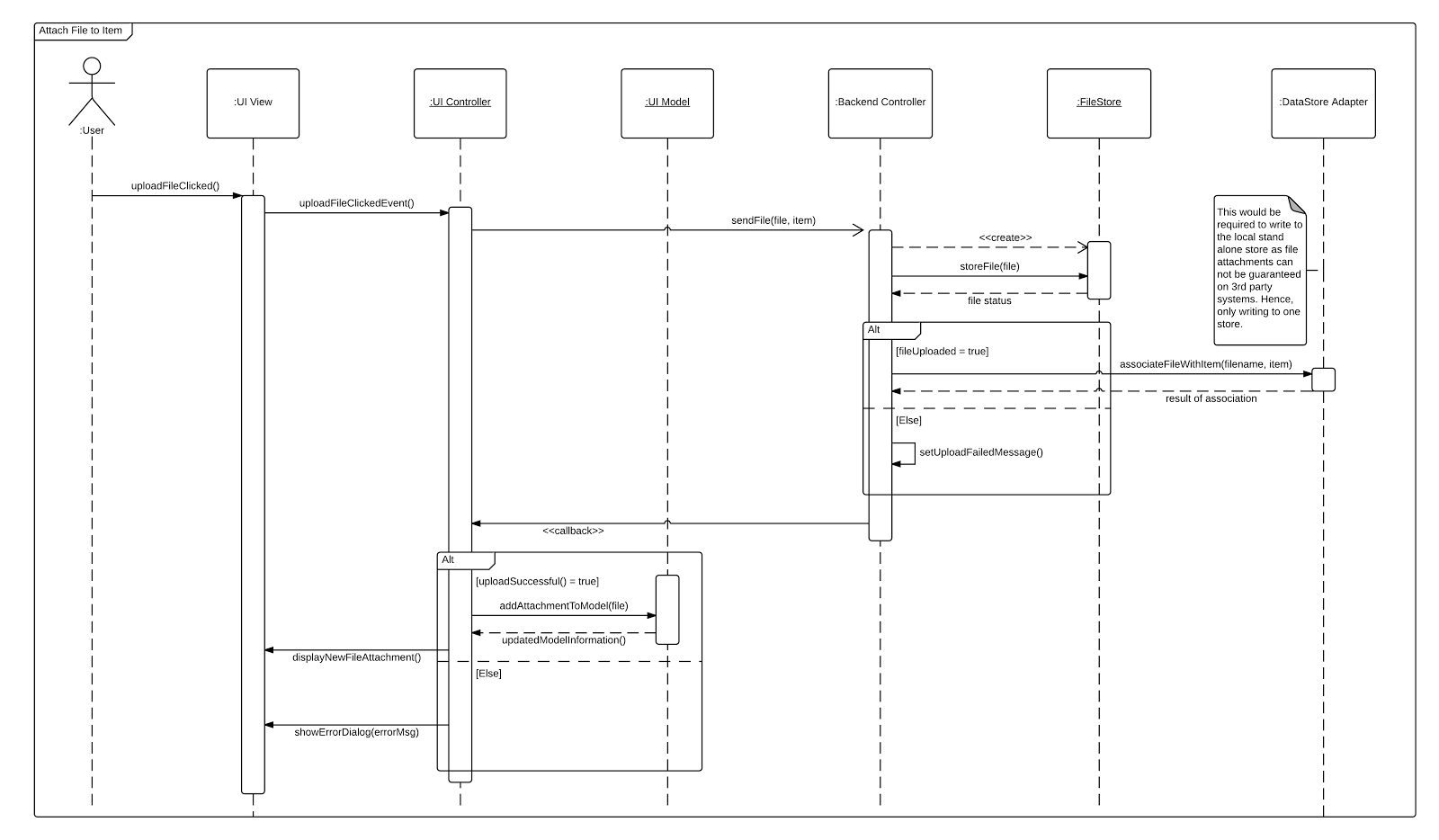
Edit Item Use Case Sequence Diagram



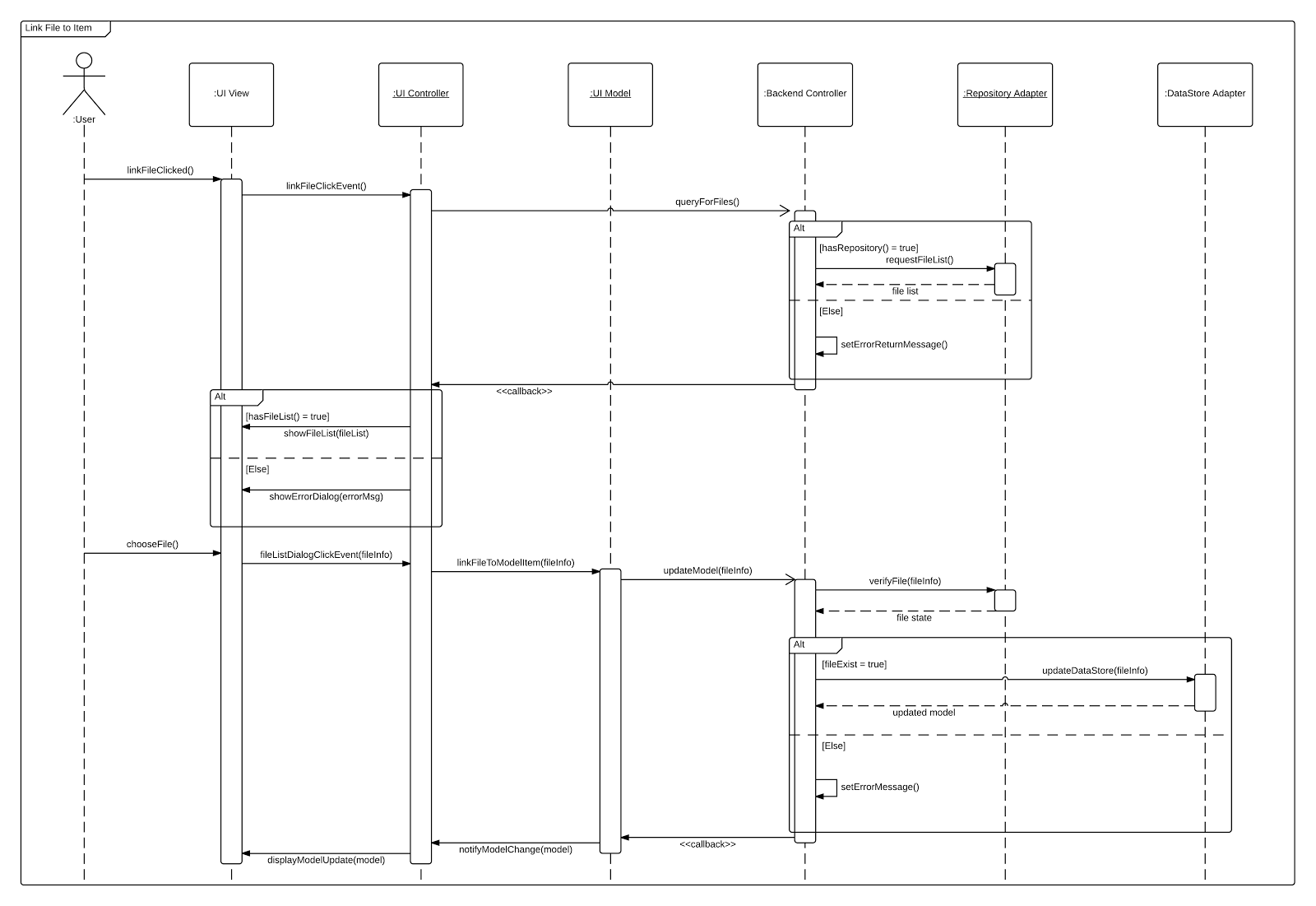
Delete Item Use Case Sequence Diagram



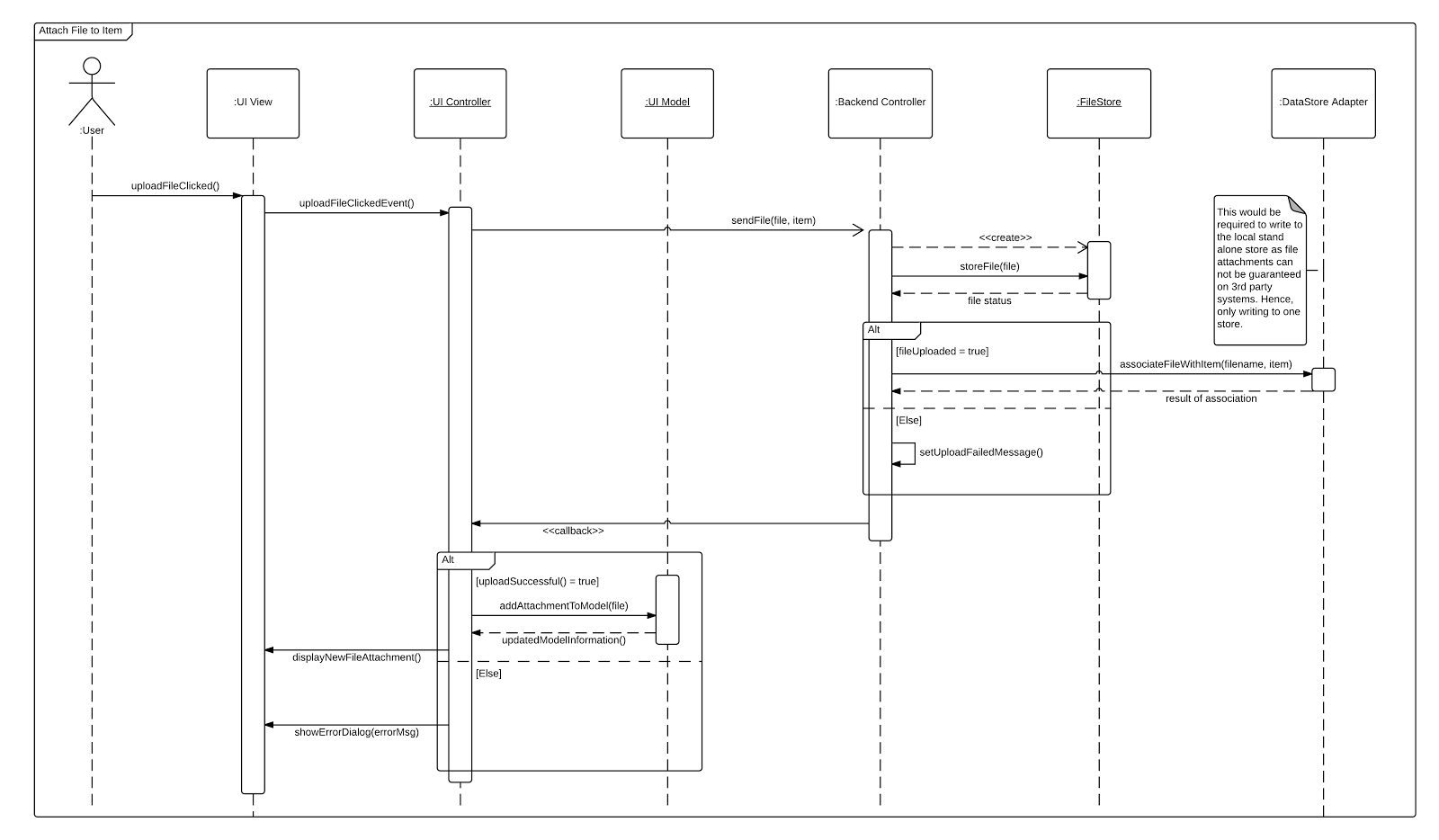
Copy Item Use Case Sequence Diagram



Link File to Item Use Case Sequence Diagram



Attach File to Item Use Case Sequence Diagram



**6.5. Appendix E - User Interface designs.**

**6.6. Appendix F - Diary of meeting and tasks.**

**Diary Entry 1:**

**Date:** Thursday, September 5, 2014

**Location:** ECS Lab 212

**Start time**: 3:00 pm

**End time:** 4:35 pm

**In Attendance:** Dionny Santiago, Tariq King, Justin Phillips, Karina Harfouche

**Late:** N/A

**7. References**

***Please email me the UML diagram in one file before the presentation.***