*Florida International University*

*School of Computing and Information Sciences*

CIS 4911 Senior Capstone Project

Test Case Management System

**Deliverable 3 - Design Document (DD)**

September 2014

**Group Members**

Justin Phillips  
Karina Harfouche

**Mentors**Tariq King  
Dionny Santiago

**Instructors**Masoud Sadjadi  
Juan Caraballo

**Abstract**

This design document covers design related topics beginning with the problem statement, overview of the document, and the design methodology employed during the design phase. Following these sections is the system design chapter where the system is overviewed, decomposed, the persistent data model is depicted, security and system accessibility is covered. The next chapter covers detail design, including many of the static / dynamic artifacts generated over the course of this project. Finally the appendix closes out the document with detailed artifacts generated from the system design; these include UML diagrams, code interfaces, and meeting minutes.

**1.      Introduction**

**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. Additionally, MTM cannot connect to 3rd party systems so Ultimate test engineers must used multiple user interfaces in order to maintain their test plans. 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 with a given software product.

The Test Case and Automation application will help overcome some of the difficulty with software testing and the reviewing of test results. A software tester that runs dozens of test cases a day relies heavily on endless amounts of logs scattered across different applications to accurately keep track of failures and its regular tendencies. Common causes of legitimate or false failures are often overlooked due to the large amount of details required to analyze test results and conclude its resolution with the correct form of action on a timely matter.

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; in addition to viewing the results is a quick, easy to comprehend format.

**1.2.   Design Methodology**

The overall design methodology employed for this project is agile, where design, development, and verification happen in an iterative cycle. The initial design iteration encompassed overall system design and deciding on the technology stack to utilize.  The requirements were analyzed and many factors were considered with choosing the overall system design and technology stack such as existing developer knowledge of technologies, requirements of 3rd party systems, and  long term goals of the system.

**1.3.   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**

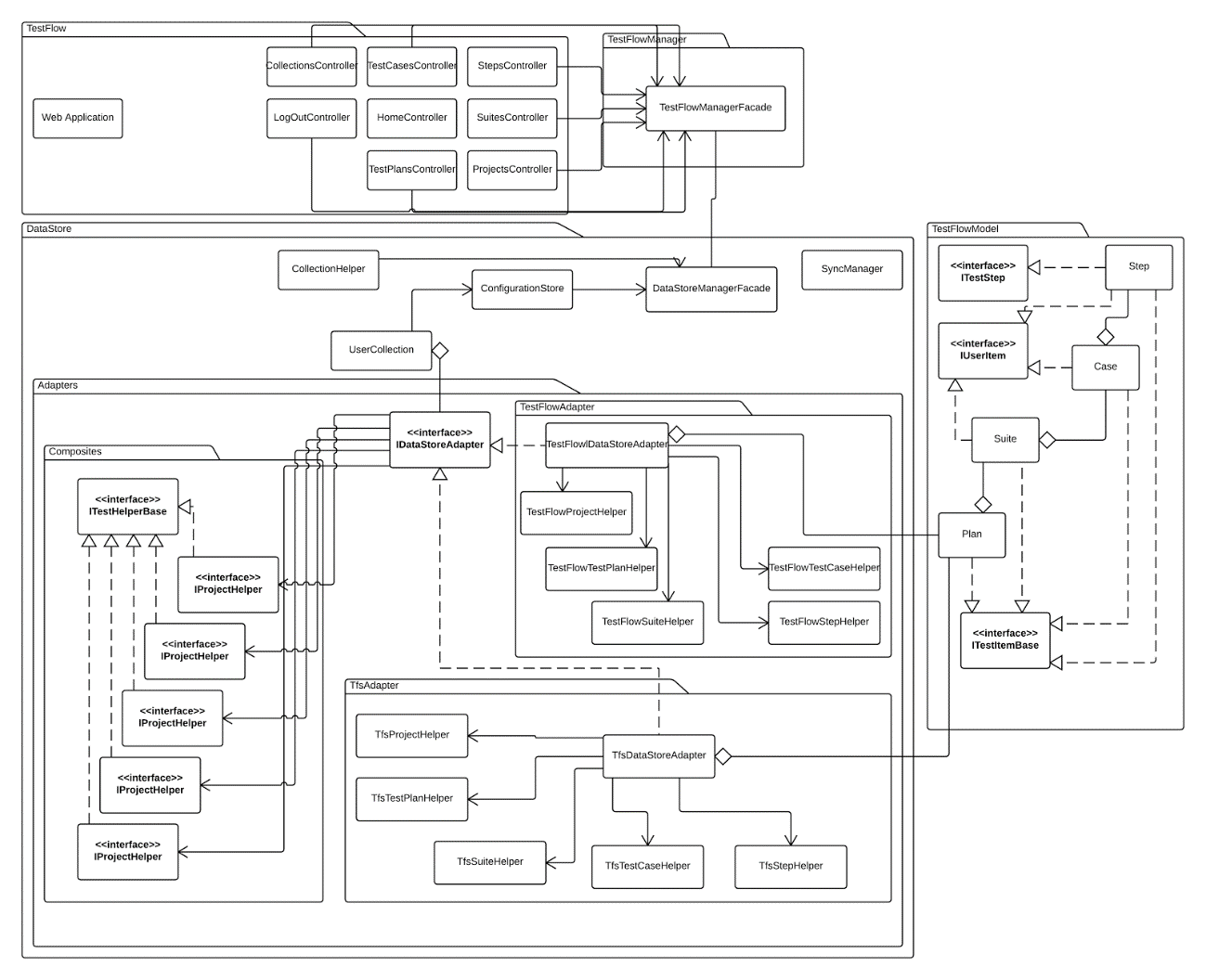
**2.      System Design**

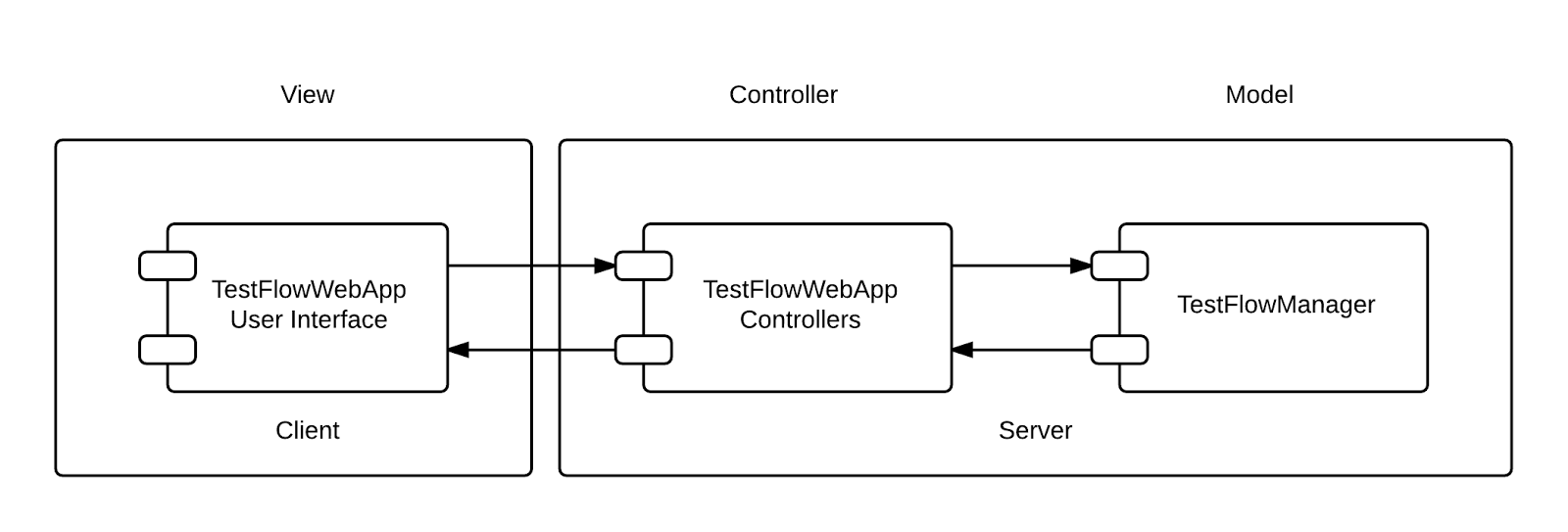
This chapter covers the overall system design and decomposition in great detail starting with a overview of the entire system. Subsystem decomposition describes each major subsystem and hardware and software mapping provides a picture of the systems implementation. These sections are followed by persistent data management, security, and finally detailed design where details specific to each object in the system are presented.

**2.1.   Overview**

The system’s overall architectural pattern is composed of a client / server model where  a backend system provides connectivity to data and operations on the data while a client provides

access to the data to an end user. Below the package diagram shows the structure of the system’s design, starting with the test flow web application which hosts a varity of controllers and view followed by the point of access to the libries developed for operating on the test plan data. This system is exposed by TestFlowManager, which is a facade patterned object for managing the test flow system. The DataStore represents a series of adapters which provide connectivity to databases and 3rd party test management systems. This is where the test plans are provided persistant storage. Finally there is the TestFlowModel which realizes the testplan model data in code.



**Package Diagram of system, readable version in UML directory.** 

**Architectural Component Diagram**

The above component diagram depicts the system’s two architectures. The first being MVC where the HTML user interface of the web application servers as the view while the TestFlowWebApp’s controllers serve to decouple the model from the user interface. The model is provided by the TestFlowManager. The second architecture employed is client / server, where the user interface is located on a client browser and the rest of the system is provided on the server.

**2.2.   Subsystem Decomposition**

There are four major components of the system, the DataStore, TestFlowManager, TestFlowWebApp, and Metrics.  The DataStore and TestFlowManager are libraries which form the connectivity and logic behind managing the data both on the system’s database and 3rd party management systems.

The DataStore is the central component to satisfying the projects requirements regarding the distribution of test between the system and other test management systems. The DataStore is composed of adapters which facilitate communication via a common interface with test management systems, these adapters are accessed through the DataStore facade pattern based manager object.

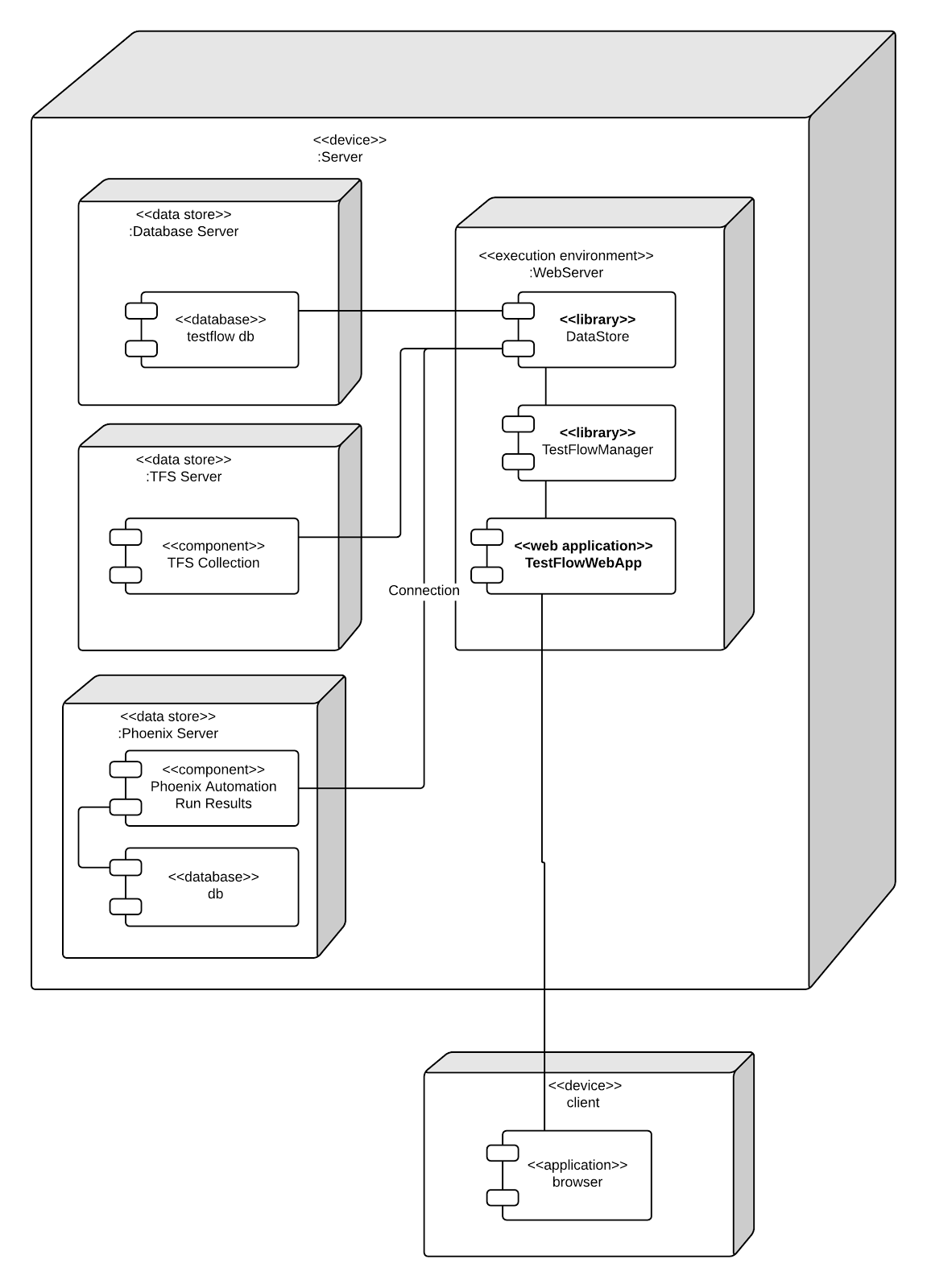
The TestFlowManager component is a facade pattern which allows for the operations exposed by the various libraries to be utilized in one object. The component controls all access to the DataStore and any future components.

The TestFlowWebApp component is the point of access to the system, it is MVC patterned website. It utilizes services which allow for access to the TestFlowManager.

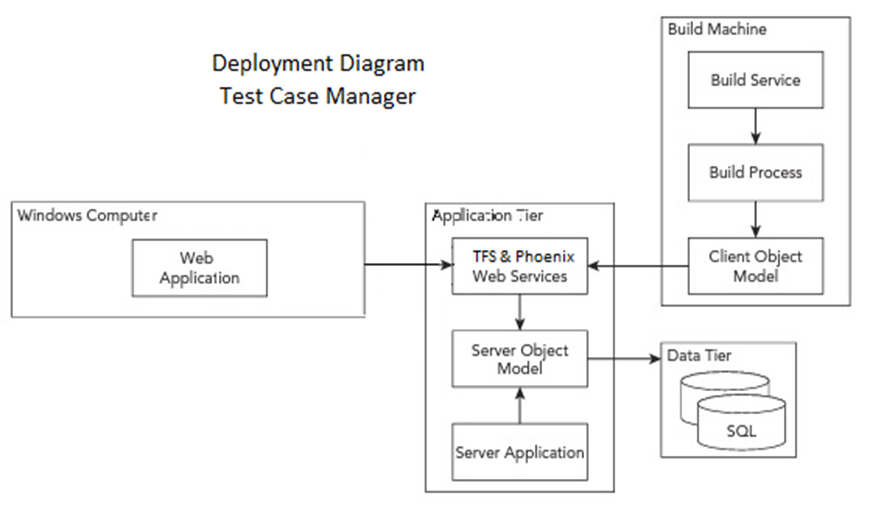
The Metrics component accesses the mock Phoenix server in order to run automation utilizing the data on the Phoenix database. The results are then transferred via web services to the TestFlowManager metrics subsystem.

**2.3.   Hardware and Software Mapping**

A Microsoft Windows server houses a web server as an execution environment for the system’s components which connect and store data on two data stores, a database system and a Team Foundation Server. A client connects to the server execution environment via a web browser. Finally, the system requires a Phoenix automation service to connect with.

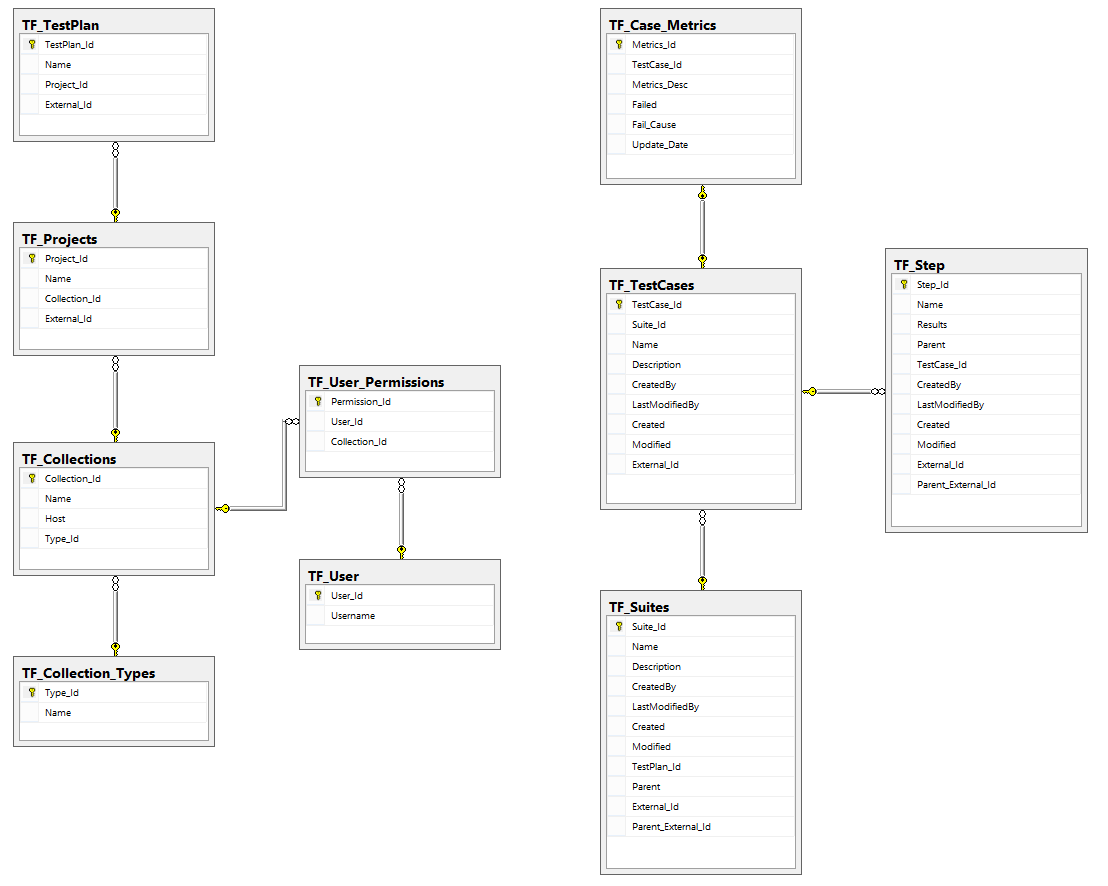


In the deployment diagram featured above, the entire system resides on a central server, although the TFS server, Phoenix server, and DB server could be hosted elsewhere. The web server components must be hosted on a single web server.



**2.4.   Persistent Data Management**

The persistent data is managed via a database server and stored in a database. This data includes user permission information, external storage mapping using IDs, and test plan data.



**Figure 1.2 ERD of database**

**2.5.   Security/Privacy**

Users are authenticated via Windows user identity, anonymous authentication will not be allowed. The Windows identity will be stored in the database and any projects in TFS which the user has permission to access are mapped to the user via the TF\_User\_Permissions table.

**3.      Detailed Design**

This chapter presents the Test Case Manager’s object design. First, an overview of the entire system is depicted followed by the details of the object design process.  A minimal class diagram shows the complete system which will provide the appropriate context to introduce object interactions represented on the sequence diagrams.  The detailed class description and each class’ purpose and functions have also been included in order to effectively portray and explain the overall object design.

**3.1.   Overview**

**DataStore**

The DataStore component is the core of the system, it provides interoperability via a suite of adapters made of up composite interfaces to external management systems and the test flow database. The adapter is an interface which provides common functionality and usage across different test management platforms. There is also a synchronization manager which ensures the test plan stays synchronized across platforms.

**TestFlowManager**

The TestFlowManager is a simple facade component which provides accessibility to the other test flow libraries as to create a single point of access to testflow.

TestFlowModel

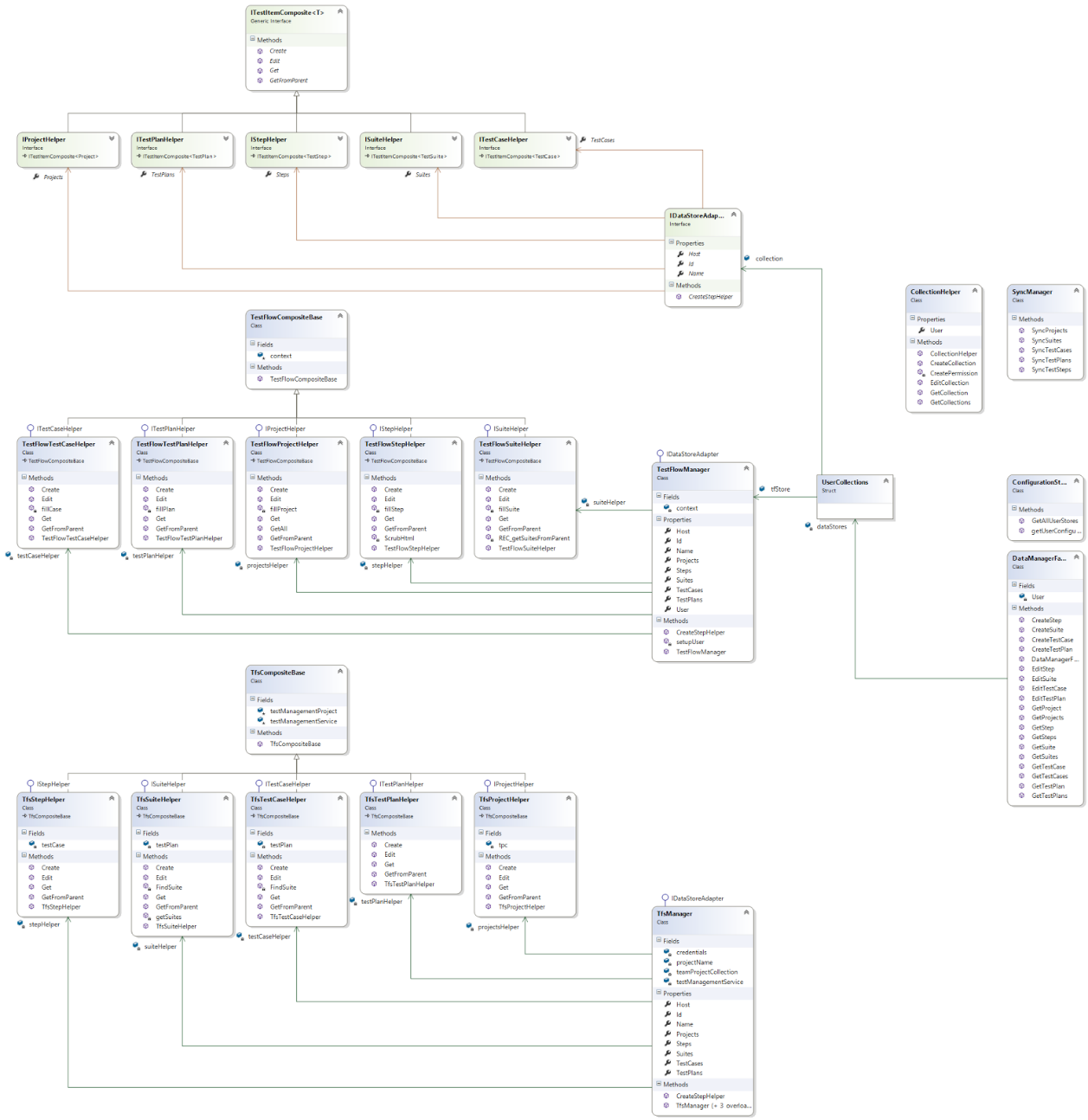
The TestFlowModel component realizes the data as objects in code from the database, these objects are used to create, view, and edit the test elements.

**TestFlowWebApp**

The WebApp is the entry point to using testflow, it provides the user interface and operation endpoints to users. The WebApp is two MVC architectures layered on one another, the first provides most of the features on the server side, with it’s view providing a client side MVC application.

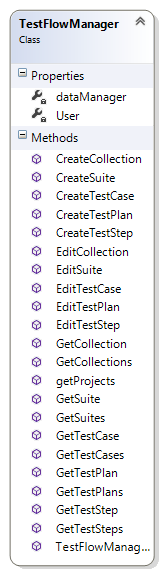
**3.2.   Static model**

DataStore



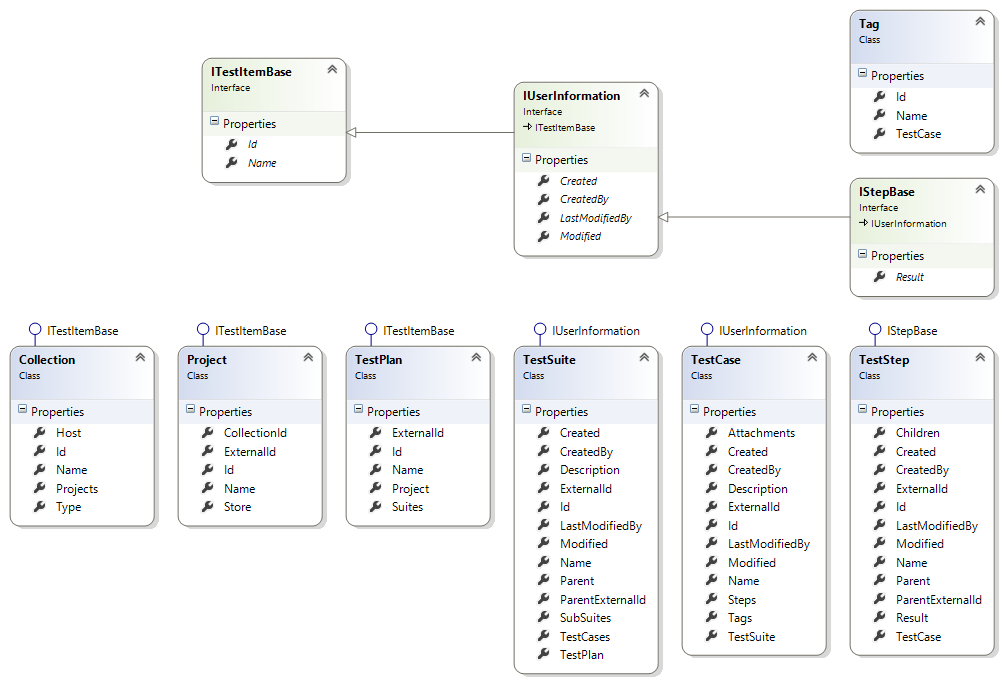
**DataStore Class Diagram: Note that a more readable version can be found in the GIT files under UML.**

TestFlowManager



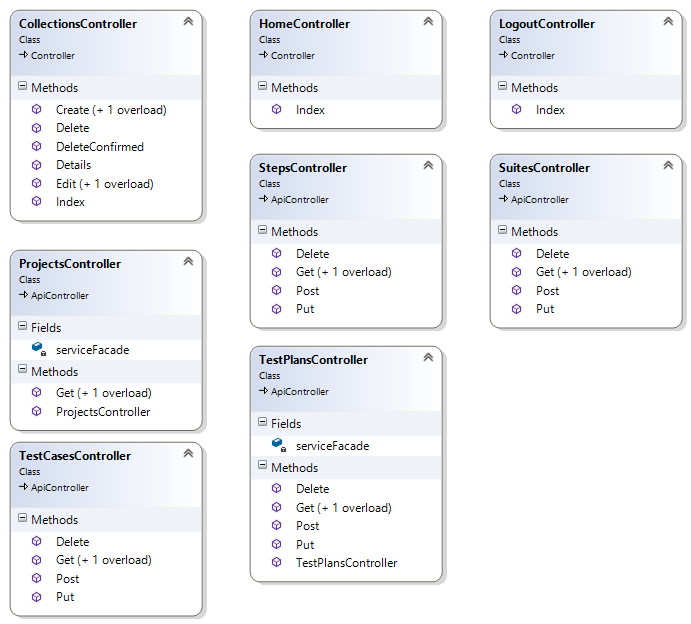
**TestFlowManager Class Diagram**

TestFlowModel



**TestFlowModel Class Diagram**

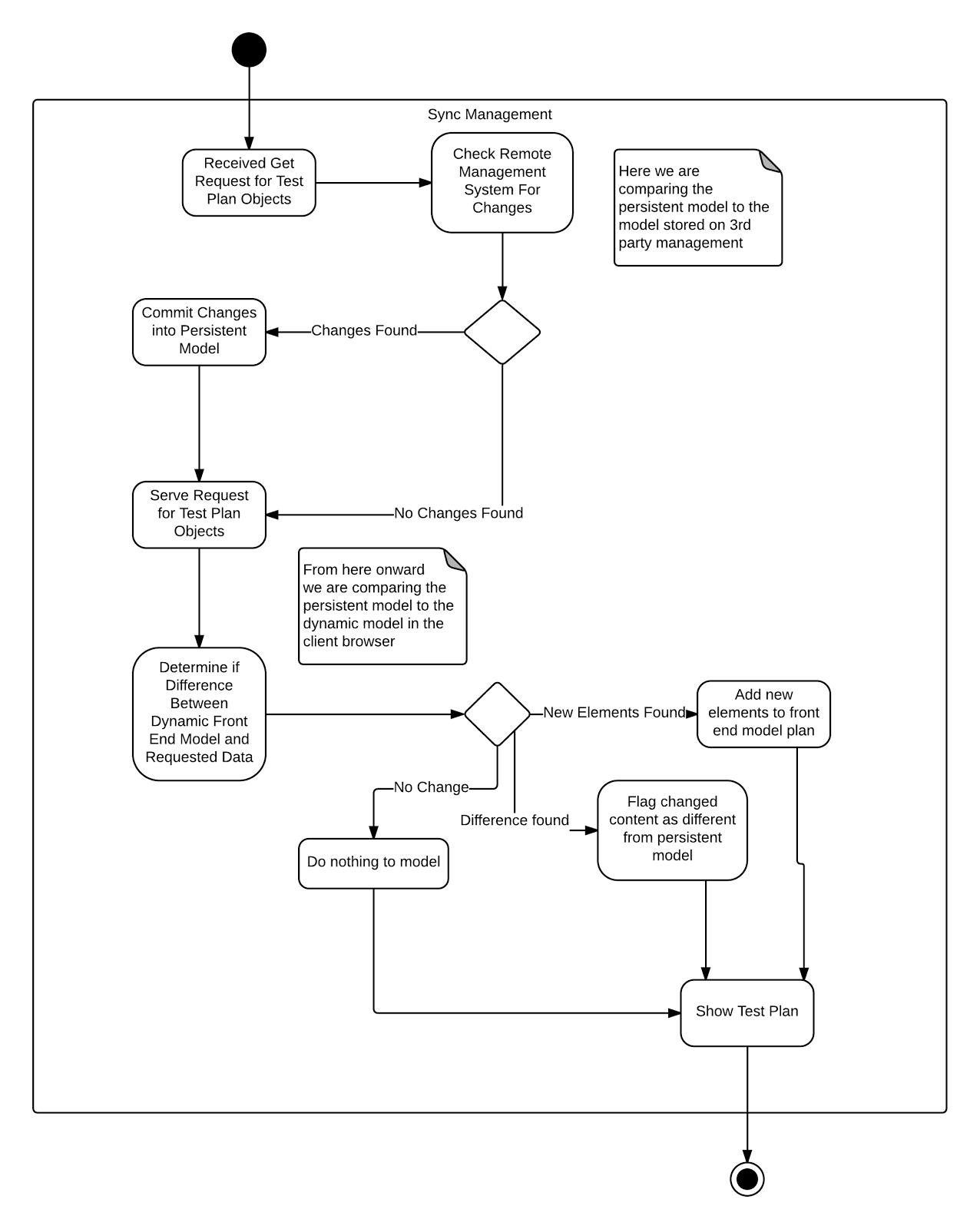
TestFlowWebApp



**TestFlowWebApp Class Diagram: Note that the majority of the TestFlowWebApp is implemented in JavaScript which does not have formal diagrams.**

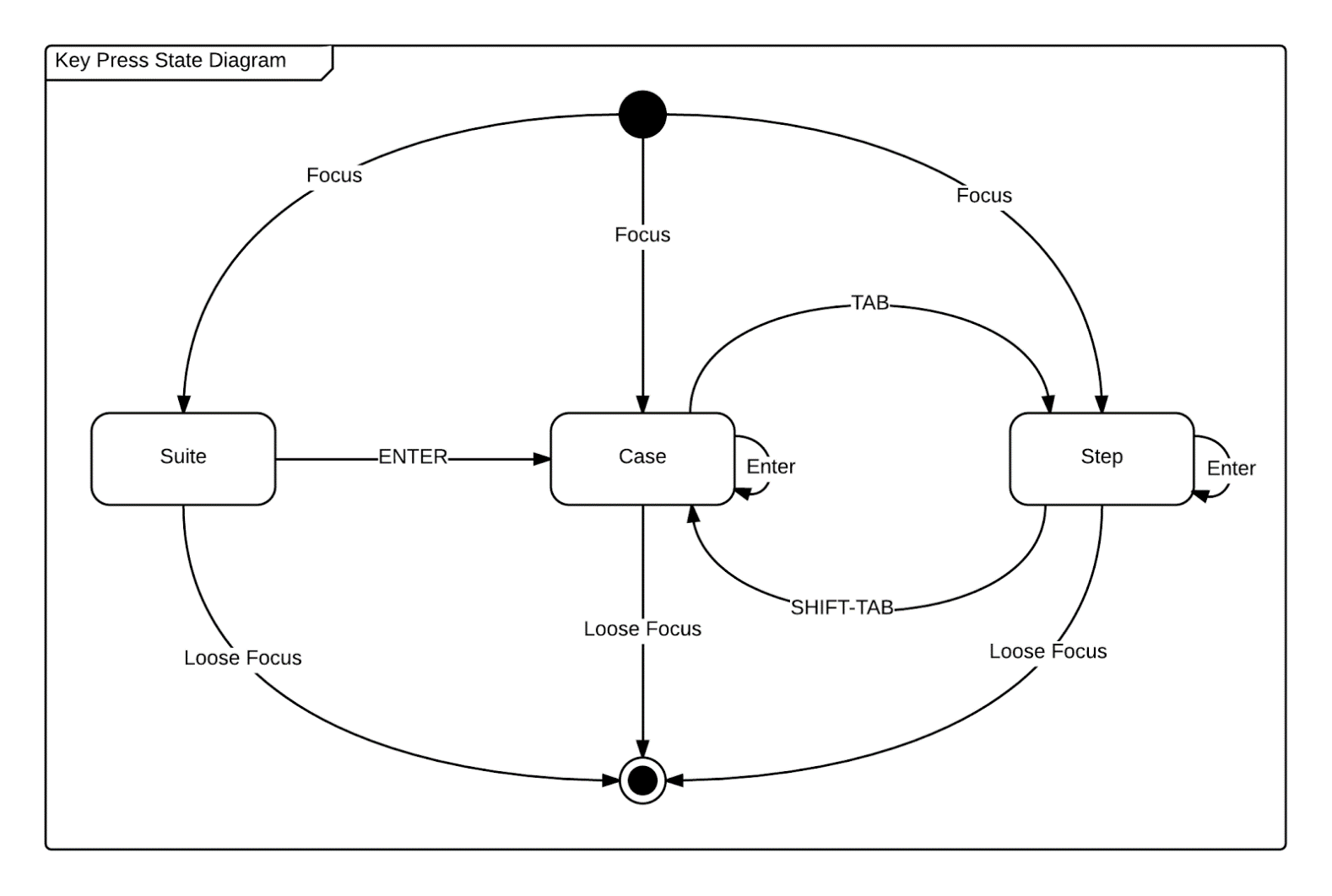
**3.3.   Dynamic model**

*DataStore*



The above activity diagram demonstrates the flow of information and changes through the system, essentially how the data is synchronized across platforms. It begins with a request for test plan data and ends with the data being shown after synchronizing with external sources and any changes in the user interface model.

*TestFlowWebApp*



The above state diagram represents focus and keypress action states within the user interface. Ultimate software required rapid type changing of elements via key presses. When an element has focus in the user interface this state diagram is used to determine what action is performed when certain keys are pressed.

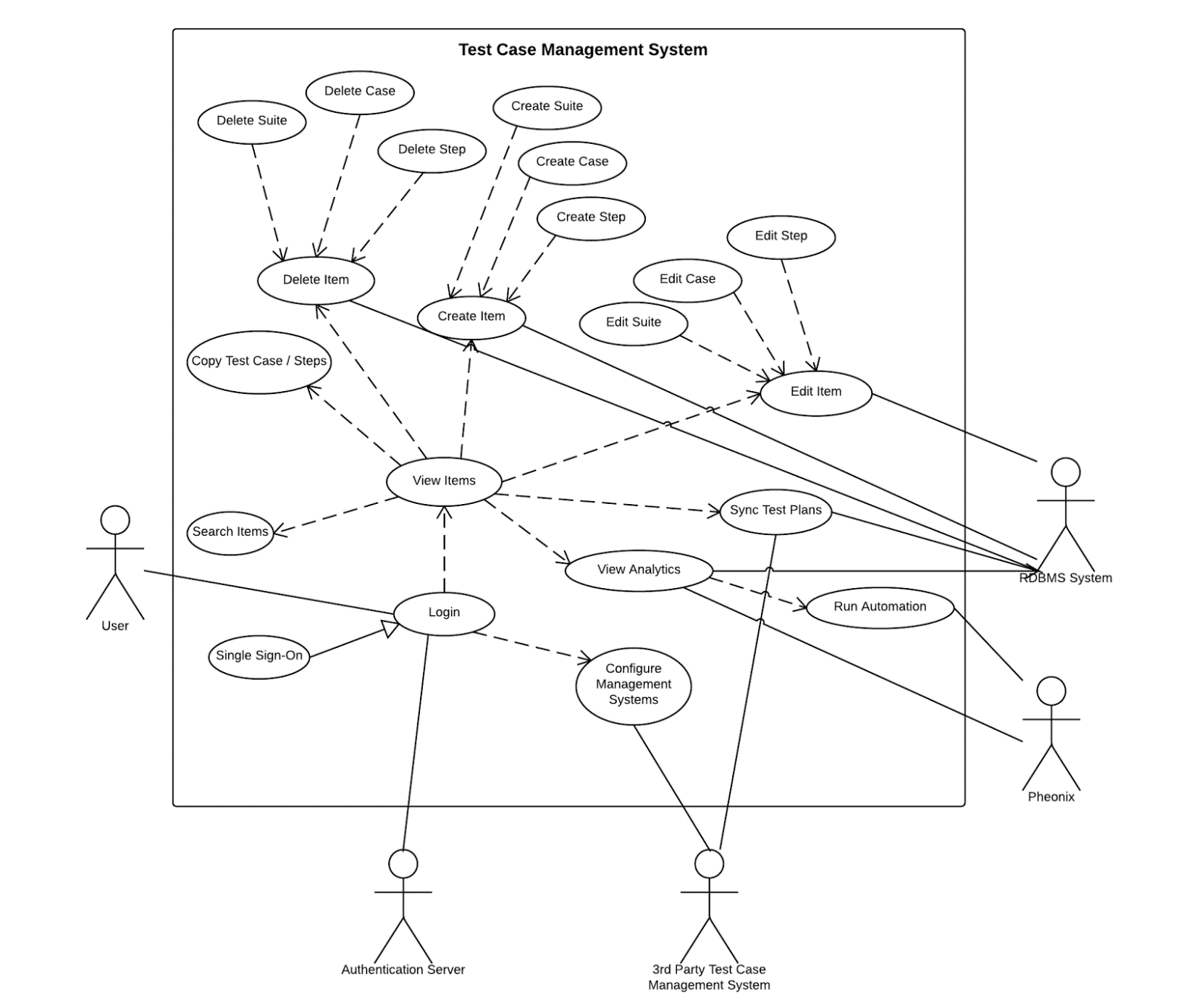
**3.4.   Code Specification**

The code stored in appendix C represents the interfaces from each of the components. The interfaces within DataStore represent the IDataAdaper’s potential realization. This adapter is formed from a series of helper interfaces that, as composites, realize the IDataAdapter’s functionality in manipulating test plans. The interfaces within the TestFlowModel reduce code redundancy and increase the consistency of the elements.

**4.      Glossary**

**5.      Appendix**

**5.1.   Appendix A**



**Use Case Diagram of implemented use cases.**

**5.2.   Appendix B**

**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 tester 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

https://lh3.googleusercontent.com/1K2lx-c3FrYjITA0S01EAySDtsEOLDe04EYL9m-VeJG0rdEt6E2aXAZLF-q6ZrmQ8An_3yDfG6ubapJsoEXkUxVeFfCL09UQ5rM4ImIAkNUZw8riF979868wSO-7qAuNOQ

**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 the plus sign from the suites left panel. (Trigger)
2. The user then inputs a Suite Case title.
3. The use case ends when the tester hits enter 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

https://lh5.googleusercontent.com/oDQTl5Bl5JUBea6pqBrNEnhP49l1vF0nbBUiBDyPFg22Kp4EgZSN16BqnN6g48Hr7DKtolrcVTCH8vQJuUmTtjWRmSP3TH5GBTpXVQLt6Hc9fPXJWtuHb9S-X1uF6JFSsw

**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 sign beside a Test Suite on the main panel to add a test case to this suite. (Trigger)
2. The system responds by loading an editable tab to name a test case.
3. The tester fills the tab with a test case title
4. The use case ends when the tester hits enter to add the case with the provided information

**Post-Conditions**

1. The new test case will be added on the hierarchical menu list under its correspondent test suite.
2. 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

https://lh6.googleusercontent.com/Bwbr6B_iPbQitTmZTqQhG0wqG7av7Ynn-oqbjtLzHyk6-AY6xToA8MAuqkdC8V_est1SRjNWnMeSmmkjoNRozcUto0loKw4wNJwY0ieCTvb_J_1UYXwoLG9UV-Fa2JzAsQ

**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 pull down test case

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

3.      The user clicks on th plus sign button and types new step

4.      The use case ends when the tester hits enter 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

https://lh5.googleusercontent.com/0WzVy8QUdYpx6FYyQIT4hl4VqLxhkxpnuuZqHslnPZGNtbPQctRzg5extK3KzlO4KjQFqVsUWGNvWzx3Vj3ZlV7CrBR3k1-4AlIIU-IjTX5J-O_QhmI8giVfe7DcMbgmFA

**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

https://lh6.googleusercontent.com/Gi9-kI50MGCh2RZkQCNTbCiRCN2OHK6WGWwY0Xi7eED7tUqGAsCajNdIcg9LiifpSEY_1gT8WuAyhXF2cHAgpDyLkz5bUXctOrG3_hHq8LTI4BAXcocGVqbRR0ZQPhuHbw

**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

https://lh3.googleusercontent.com/SHU8CDnfHkuS4_kssgCUnq7lGneI22DLV04gEE09t4mNrW2QbDvOOZ8O2hUUBAemhxR_9__U4gv0EhPRTY2wULmEVqihlxNIl5EnA9PS0UzRwp7Q4j4k9xj7pz-FvdY8Ig

**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 tester clicks on “Attachments” at the Details section of the current test case

2.      The system responds by loading an upload window

3.      The user selects one or multiple files 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

https://lh5.googleusercontent.com/87Y66Kz1WVHt3IAVQvjuhuglqz5HbKi0puNf3G0OPTTgpL7orilvtdj2cJz6F0R_gpwJTN6y3ngkMm0nRWig8lDNrU6A4SHusL4LgbPgyK3nQhU5yZN5qHqEDTTwNWJ5EA

**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

https://lh5.googleusercontent.com/87Y66Kz1WVHt3IAVQvjuhuglqz5HbKi0puNf3G0OPTTgpL7orilvtdj2cJz6F0R_gpwJTN6y3ngkMm0nRWig8lDNrU6A4SHusL4LgbPgyK3nQhU5yZN5qHqEDTTwNWJ5EA

**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

**5.3.   Appendix C**

**DataStore Interfaces**

**ITestItemComposite**

/// <summary>

   /// This composite pattern allows for the creation, editing, retrieving, and maping of test items

   /// </summary>

   /// <typeparam name="T"></typeparam>

   public interface ITestItemComposite<T>

   {

       int Create(T item);

       bool Edit(T item);

       T Get(int id);

       List<T> GetFromParent(int parentId);

   }

**IProjectHelper**

public interface IProjectHelper : ITestItemComposite<Project>

   {

   }

**ITestPlanHelper**

public interface ITestPlanHelper : ITestItemComposite<TestPlan>

   {

   }

**ISuiteHelper**

public interface ISuiteHelper : ITestItemComposite<TestSuite>

   {

   }

**ITestCaseHelper**

public interface ITestCaseHelper : ITestItemComposite<TestCase>

   {

   }

**IStepHelper**

public interface IStepHelper : ITestItemComposite<TestStep>

   {

   }

**IDataStoreAdapter**

/// <summary>

   /// Adapter interface providing common functionality for any data store connectivity.

   /// </summary>

   public interface IDataStoreAdapter

   {

       Uri Host { get; set; }

       int Id { get; set; }

       string Name { get; set; }

       // helper objects for each test item type

       IProjectHelper Projects { get; }

       ITestPlanHelper TestPlans { get; }

       ISuiteHelper Suites { get; }

       ITestCaseHelper TestCases { get; }

       IStepHelper Steps { get; }

       // create instances of helpers after manager is created.

       void CreateStepHelper(int testCaseId);

   }

**TestFlowModel**

**ITestItemBase**

public interface ITestItemBase

{

int Id { get;set; }

string Name { get;set; }

}

**IUserInformation**

public interface IUserInformation : ITestItemBase

   {

       int LastModifiedBy { get; set; }

       int CreatedBy { get; set; }

       DateTime Created { get; set; }

       DateTime Modified { get; set; }

   }

**IStepBase**

public interface IStepBase : IUserInformation

{

string Result { get;set; }

}

**TestFlowManager**

**ITestFlowManager**

public interface ITestFlowManager

{

   private DataManagerFacade dataManager;

   private IPrincipal User;

   public TestFlowManager(IPrincipal user, bool collectionManagement);

   public TestFlowManager(IPrincipal user, int projectId = -1);

   public TestFlowManager(IPrincipal user, int projectId, int testPlanId);

   public TestFlowManager(IPrincipal user, int projectId, int testPlanId, int testCaseId);

   // Project Stuff

   // \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

public List<Project> getProjects();

   // \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

   // end Project Stuff

   // TestPlan Stuff

   // \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

public TestPlan GetTestPlan(int testPlanId);

   public List<TestPlan> GetTestPlans(int projectId);

   public void EditTestPlan(TestPlan testPlan);

   public void CreateTestPlan(TestPlan testPlan);

   // \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

   // end TestPlan Stuff

   // Suite Stuff

   // \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

   public TestSuite GetSuite(int suiteId);

   public List<TestSuite> GetSuites(int testPlanId);

   public bool EditSuite(TestSuite suite);

   public int CreateSuite(TestSuite suite);

   // \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

   // end Suite Stuff

   // TestCase Stuff

   // \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

   public TestCase GetTestCase(int testCaseId);

   public List<TestCase> GetTestCases(int suiteId);

   public bool EditTestCase(TestCase testCase);

   public int CreateTestCase(TestCase testCase);

   // \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

   // end TestCase Stuff

   // Step Stuff

   // \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

   public TestStep GetTestStep(int testStepId);

   public List<TestStep> GetTestSteps(int projectId);

   public bool EditTestStep(TestStep step);

   public int CreateTestStep(TestStep step);

   // \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

   // end Step Stuff

   // collection management metods

   // Collections stuff

   public void EditCollection(Collection collection);

   public Collection GetCollection(int id);

   public List<Collection> GetCollections();

   public void CreateCollection(Collection collection, int type);

}

**5.4.   Appendix D**

**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

|  |  |
| --- | --- |
| Meeting: ***Regularly Schedule Meeting*** | Date: *09/17/2014* |
| Meeting Minutes Taken By: *Justin Phillips and Karina Harfouche* |  |
| Attendance: *Tariq King, Dionny Santiago* |  |

|  |
| --- |
| **Minutes:** |
| **Item 1: Project Complexity**  Time Allotted: 15 minutes  Decision: The model aspect of the system shall be fully implemented by the team and an adapter interface shall be used to leverage 3rd party management systems and migration.  Responsible Individuals: N/A |
| **Item 2: Functional Requirements and System Actors**  Time Allotted: 15 minutes  Decision: Review of the functional requirements of the system based on the use case diagram. The client asked that the use case diagram show added complexity of the system, such as the login begin required and that all use cases be visible, perhaps through generalizations or other relationship notation.  Responsible Individuals: N/A |
| **Item 6: Finalize Functional Requirements**  Time Allotted: 5 minutes  Decision: Based on the use case diagram feedback the requirements were finalized.  Responsible Individuals: Justin / Karina |
| **Item 6: Present Project Timeline**  Time Allotted: 10 minutes  Decision: It was decided the timeline was acceptable, though optimistic. It may need to be refactored as we end the requirements phase. Rushing the requirements phase was highly discouraged.  Responsible Individuals: Justin / Karina |

|  |  |
| --- | --- |
| Meeting: ***Regularly Schedule Meeting*** | Date: *09/22/2014* |
| Meeting Minutes Taken By: *Justin Phillips and Karina Harfouche* |  |
| Attendance: *Tariq King, Dionny Santiago* |  |

|  |
| --- |
| **Minutes:** |
| **Item 1: Feedback on Use Cases**  Time Allotted: 15 minutes  Decision: Still pending review from the client  Responsible Individuals: Karina Harfouche |
| **Item 2: Feedback on use case diagram and sequence diagrams**  Time Allotted: 15 minutes  Decision: Mentor suggested more relationships on the use cases and further complexity in the sequence diagrams.  Responsible Individuals: Justin Phillips |
| **Item 3: Logistics**  Time Allotted: 5 minutes  Decision: Server capacity, hosting, etc...  Responsible Individuals: Justin / Karina |

**Meeting Minutes**

|  |  |
| --- | --- |
| Meeting: ***Regularly Schedule Meeting*** | Date: *09/29/2014* |
| Meeting Minutes Taken By: *Justin Phillips and Karina Harfouche* |  |
| Attendance: *Tariq King, Dionny Santiago* |  |

|  |
| --- |
| **Minutes:** |
| **Item 1: Feedback on use case diagram and sequence diagrams**  Time Allotted: 15 minutes  Decision: Final diagrams were moved to done.  Responsible Individuals: N/A |
| **Item 2: Discuss Potential Features**  Time Allotted: 15 minutes  Decision: Mentor recommended a stand alone model to be used along side TFS which Justin will implement for two way flow of data. Karina recommended metrics to the client which they accepted.  Responsible Individuals: Justin & Karina |
| **Item 3: Project Summary**  Time Allotted: 5 minutes  Decision: Summarized where we are at and what we have done so far.  Responsible Individuals: Justin / Karina |

**Meeting Minutes**

|  |  |
| --- | --- |
| Meeting: ***Regularly Schedule Meeting*** | Date: *10/08/2014* |
| Meeting Minutes Taken By: *Justin Phillips and Karina Harfouche* |  |
| Attendance: *Tariq King, Dionny Santiago* |  |

|  |
| --- |
| **Minutes:** |
| **Item 1: Show Demo**  Time Allotted: 15 minutes  Decision: Feedback from client was very positive, client may schedule company meeting for us to do the demo in.  Responsible Individuals: N/A |

**Meeting Minutes**

|  |  |
| --- | --- |
| Meeting: ***Regularly Schedule Meeting*** | Date: *10/15/2014* |
| Meeting Minutes Taken By: *Justin Phillips and Karina Harfouche* |  |
| Attendance: *Tariq King, Dionny Santiago* |  |

|  |
| --- |
| **Minutes:** |
| **Item 1: Review of system architecture**  Time Allotted: 15 minutes  Decision: Great feedback from Dionny and some  architectural pattern suggestions.  Responsible Individuals: N/A |

**Meeting Minutes**

|  |  |
| --- | --- |
| Meeting: ***Regularly Schedule Meeting*** | Date: *10/22/2014* |
| Meeting Minutes Taken By: *Justin Phillips and Karina Harfouche* |  |
| Attendance: *Tariq King, Dionny Santiago* |  |

|  |
| --- |
| **Minutes:** |
| **Item 1: Review of minimal class diagram**  Time Allotted: 15 minutes  Decision: Feedback was that diagram was clean and a good start.  Responsible Individuals: N/A |

**Meeting Minutes**

|  |  |
| --- | --- |
| Meeting: ***Regularly Schedule Meeting*** | Date: *10/29/2014* |
| Meeting Minutes Taken By: *Justin Phillips and Karina Harfouche* |  |
| Attendance: *Tariq King, Dionny Santiago* |  |

|  |
| --- |
| **Minutes:** |
| **Item 1: Discussion of what is required for the company demo**  Time Allotted: 15 minutes  Decision: Feedback from client was very positive, client may schedule company meeting for us to do the demo in.  Responsible Individuals: N/A |

**Meeting Minutes**

|  |  |
| --- | --- |
| Meeting: ***Regularly Schedule Meeting*** | Date: *11/05/2014* |
| Meeting Minutes Taken By: *Justin Phillips and Karina Harfouche* |  |
| Attendance: *Tariq King, Dionny Santiago* |  |

|  |
| --- |
| **Minutes:** |
| **Item 1: New company demo schedule**  Time Allotted: 15 minutes  Decision: New deadline for our presentation at Ultimate’s company wide QA meeting.  Responsible Individuals: N/A  **Item 1: Two features to be implemented before demo**  Time Allotted: 15 minutes  Decision: File upload (Karina) and drag-and-drop / new keybind system (Justin)  Responsible Individuals: Justin and Karina |

**Meeting Minutes**

|  |  |
| --- | --- |
| Meeting: ***Impromptu Weekend Meeting*** | Date: *11/8/2014* |
| Meeting Minutes Taken By: *Justin Phillips and Karina Harfouche* |  |
| Attendance: *Tariq King, Dionny Santiago* |  |

|  |
| --- |
| **Minutes:** |
| **Item 1: Show Demo**  Time Allotted: 15 minutes  Final demo for QA meeting, great response from Tariq and Dionny.  Responsible Individuals: N/A |

**Meeting Minutes**

|  |  |
| --- | --- |
| Meeting: ***Ultimate Company Wide QA Meeting*** | Date: *11/11/2014* |
| Meeting Minutes Taken By: *Justin Phillips and Karina Harfouche* |  |
| Attendance: *Tariq King, Dionny Santiago* |  |

|  |
| --- |
| **Minutes:** |
| **Item 1: QA Meeting Presentation**  Time Allotted: 15 minutes  Decision: Great feedback from mentor about our presentation.  Responsible Individuals: N/A |

**Meeting Minutes**

|  |  |
| --- | --- |
| Meeting: ***Regularly Schedule Meeting*** | Date: *11/19/2014* |
| Meeting Minutes Taken By: *Justin Phillips and Karina Harfouche* |  |
| Attendance: *Tariq King, Dionny Santiago* |  |

|  |
| --- |
| **Minutes:** |
| **Item 1: Moving forward**  Time Allotted: 15 minutes  Decision: Received some suggestions on path forward.  Responsible Individuals: N/A |