ISTANBUL TECHNICAL UNIVERSITY

Q&A Web Page

Test Specification Report

12/21/2013

Volkan İlbeyli	**040100118**
Gökhan Çoban	**040100057**
Faruk Yazıcı	**040100112**
Emre Gökrem	**040100124**
Tuğrul Yatağan	**040100117**

Istanbul Technical University Computer and Informatics Faculty
BLG 411E Software Engineering
CRN: 10827
2013/2014 Fall

1. Introduction

The document contains 3 sections: Introduction, Test Plan and Test Procedure. Introduction provides an overview of the entire document, specifying goals and objectives, statement scope and major constraints of the project. Test plan section describes overall testing strategy and the project management issues that are required to properly execute effective tests. Unit testing strategies, integration testing, high-order testing and validation testing are presented as well as the environmental settings such as tools, scheduling, metrics and record keeping. Test procedure section describes as detailed test procedure including test tactics and test cases for the software. Unit tests for each component are described in detail.

1.1 Goals and objectives

The purpose of the test process is to test the product before deployment and debug when necessary. The system will be tested for validity of the logical operations, general layout, error handling, functionality and behavior to fix the issues before deployment.

1.2 Statement of scope

The components responsible for add, delete and update operations are tested for the validity of their operations. Some components require admin authorization will be tested as a user to see if the access is denied.

1.3 Major constraints

Major constraints include internal component logic being flawless and input/output behavior being problem free.

2. Test Plan

Test plan includes several methods of testing procedures. Each test will be performed completely. Finally, the system testing will be performed to find the bugs of the software.

2.1 Testing strategy

Unit testing, integration testing, validation testing and system testing procedures are applied to the software.

2.1.1 Unit testing

Unit testing is the testing of each module by the developer. Working mechanisms of each module are tested. It's performed by dividing the modules' operation into every possible test case. Test cases are the paths to be followed to perform a single distinct operation. For example, a page may be the module and the buttons inside may be the test cases. Since the working mechanism is examined in unit tests, white box method is used to transparently study the modules.

2.1.2 Integration testing

Integration testing is the testing of the overall project by smaller components. Every module is supposed to have been tested and verified with unit testing beforehand, however the modules have to be tested together as well. Integration testing is used to catch and fix the errors which might occur when the whole system is working, while each module is correct within their scopes.

In integration testing, the driver is the component at the top and stubs are its sub-components. Then, each stub might have stubs too. Top-down (depth-first) integration method is used in the project. It starts with testing the first level stub that comes after the home page is opened. Then all stubs are tested respectively. After the bottom, second path from the home page is followed. Therefore, every component is tested again after each one is added. This way the integration testing shall find out the exact position of the errors that might occur during the integration of the modules.

2.1.3 Validation testing

Validation testing is the testing of the software by providing a range of input-output cases. Usual inputs are already tested with this method. Then, especially the extreme cases of inputs are chosen, which are the values between invalid and valid inputs. This will provide the testing of all conditions which users might experience. Following the validation testing, related warnings and limitations are added to the software.

2.1.4 High-order testing

Recovery, stress, performance and security testing methods will be applied for the project. The purpose of these tests are checking data loss, speed, performance and security issues of the program.

2.2 Testing resources and staffing

There are no specialized testing resources. Since the number of modules are not very high, every team member will test each module individually.

2.3 Test record keeping

Microsoft Team Foundation Server is used for issue tracking and document storing as team collaboration software in between project member. Finding and tracking bugs is one of the main objectives of tests. TSF can tack issues or bugs on its servers. This provides us to find and decide importance bugs from test results. TFS can store the all documents with their all versions about project. TFS is also integrated with Microsoft Visual Studio integrated development environment. So firstly test results puts related documents on TFS and then related issues are created if necessary. These issues are assigned to project members to resolve.
Test results are kept in TFS on related tables with information such as test case tables. Other automated test record tools data are also stored in TFS such as performance analysis, screenshots and screen records. SmartBear TestComplete software is used for automated recording during tests.

2.4 Testing tools and environment

2.4.1 Software

The following software should be installed on the test servers:

- Microsoft Team Foundation Server
- MSSQL Database Server 2012
- Microsoft Windows Server 2012 for installation environment
- Microsoft Windows 8.1 Operating System for development environment
- Microsoft ISS 8.5
- Microsoft Server Performance Adviser for performance analysis
- Microsoft Visual Studio 2012 for development and debugging IDE
- VMware Workstation for virtualization

Different operating systems are required for observing the results and the difference between them. The following operating systems are needed to maintain client side test environment:

- Microsoft Windows 8.1 Operating System
- Microsoft Windows 8 Operating System
- Microsoft Windows 7 Operating System
- Microsoft Windows Vista Operating System
- Microsoft Windows XP Operating System
- Ubuntu 12.10 Operating System
- Apple MAC OS X Operating System

Different browsers are tested on different operating systems for observing the results and the difference between them. The following browsers are needed to perform the client side tests:

- Internet Explorer 9, Internet Explorer 10, Internet Explorer 11
- Google Chrome (Latest three Version)
- Mozilla Firefox (Latest three Version)
- Safari (Latest Version)

2.4.2 Hardware

Testing procedures are mainly done with different computers:

- 1st PC:
    - Intel Core i7-4960HQ Processor (6M Cache, up to 3.80 GHz)
    - 8 GB DDR3 1600MHz Kingston RAM
    - 120 GB Samsung SSD

- 2nd PC:
    - AMD FX X6 6300 AM3+ 3,5GHz 4MB Cache
    - 6 GB DDR3 1333 MHz Corsair RAM
    - 500 GB 7200 rpm WD Hard Drive

- iPhone 5S
- iPhone 4S
- HTC with Android 4.2.2

2.5 Test schedule

TEST	TEST DEFINITION	DATE
Performance and load testing	Database is tested for loading speed with sample data.	17.12.2013-22.12.2013
Stress testing	The database is tested for repeated queries with large amount of data.	20.12.2013-21.12.2013
Power failure test	Data on the database are checked after power interrupts	20.12.2013-21.12.2013

Visualization test	Visual properties of the web pages are tested	18.12.2013-23.12.2013
Installation test	The web site is tested for proper installation	18.12.2013-21.12.2013
Stress Testing	Different user types are used for complex processing	21.12.2013-23.12.2013
Reliability test	Different user types were tested for relations	20.12.2013-23.12.2013
Unit tests	Working mechanisms of each module are tested independently	17.12.2013-20.12.2013
Integration test	After the unit test, all components are tested together for overall running mechanism	20.12.2013-22.12.2013
Validation test	Links, buttons and redirections are tested according to inputs for validation	21.12.2013-23.12.2013
Common user interface test	An user-friendly interfaces are tested	17.12.2013-23.12.2013

3. Test Procedure

This section describes as detailed test procedure including test tactics and test cases for the software.

3.1 Testing procedure

A table for unit test cases are provided below:

Tester:			Date:	
Unit	**#**	**Test Case**	**Expected Result**	**Actual Result**
CRUD	1.	click add button - enter values - update the database - view the new list	proper update of the database with the badge entered by the user in the add operation	
	2.	click delete button - update the database - view the new list	the proper update of the database with the removal of the badge chosen by the user in the delete operation	
	3.	click edit button - enter values - update the database - view the new list	proper update of the database with the badge edited by the user in the edit operation	
	4.	click list	When no action taken, or page is refreshed, the badge list should stay the same	
Login	5.	click submit button - username entered – no password entered -	missing username / password, user should	

		display error message - return login page	be warned to fill the input textboxes	
	6.	click submit button - username entered – no password entered - display error message - return login page	missing username / password, user should be warned to fill the input textboxes	
	7.	click submit button - username entered - password entered - query database - username/password mismatch - display error - return login page	displaying of incorrect user/password combination error	
	8.	click submit button - username entered - password entered - query database - username/password match – successful login - go to profile view page	entering valid information, user should be directed to the profile view page	
Interesting Question List	9.	update the database-view the interesting question list page	retrieving the correct tags and questions from database, performed by user in an interesting question list operation. Interesting Question List consists of question titles according to favorite tags of current user	
Question View	10.	click on question title - retrieve question from database - view the question page	retrieving the correct information about question, performed by user in a view operation. Question informations are question title, question text, tags, question's answers and question's comments. Also answer question, add comment and vote up/down buttons should be displayed in question view page	
Question Delete	11.	attempt for deletion of question by a user	unauthorized operation error and user should be directed question list page after displaying the unauthorized access	
	12.	attempt for deletion of question by an admin	updating the database with the removal of the question, performed by the admin in the delete operation. When the list of questions is viewed	

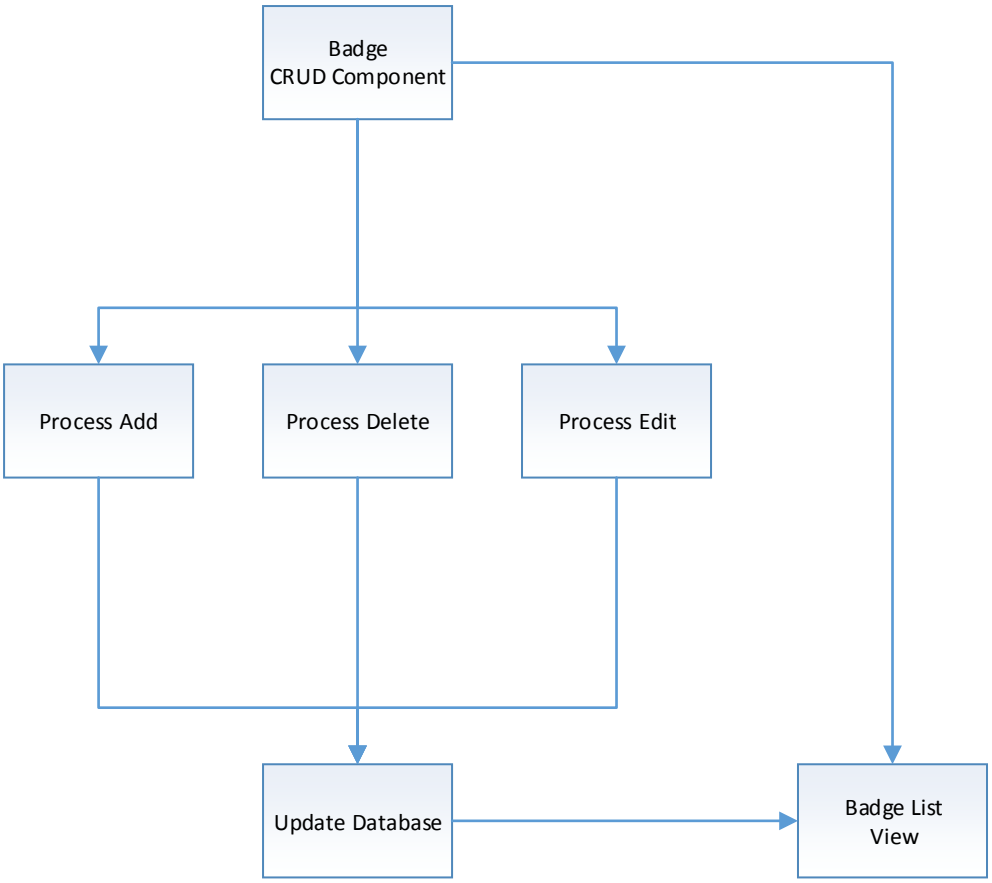
			again, the question should be removed	
User Delete	13.	the attempt for deletion of a user by a user	unauthorized operation error and user should be directed user list page after displaying the unauthorized access	
	14.	attempt for deletion of a user by an admin	updating the database with the removal of the user, performed by the admin in the delete operation. When the list of users is viewed again, the user should be removed	
Question List	15.	open questions-list-click see answers-retrieve from database-view answers	retrieval of answers from the database and listing of them on the page	
	16.	open questions-list-click see comments-retrieve from database-view comments	retrieval of the comments from the database and listing of them on the page	

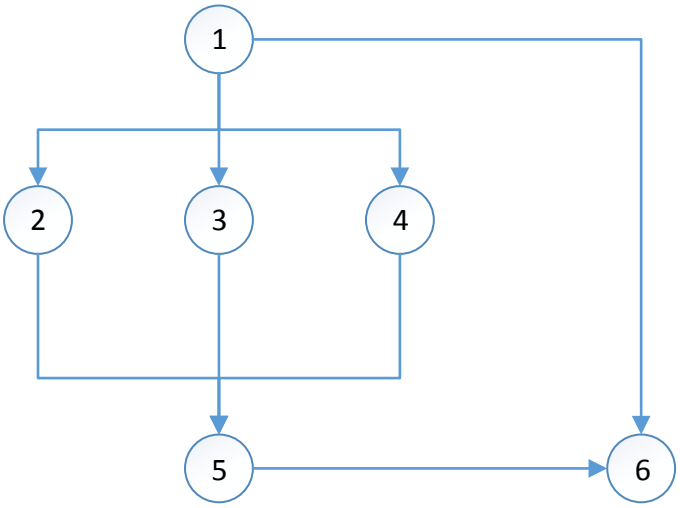
3.1.1 Unit test cases

Below are the detailed list of unit test cases.

3.1.1.1 Unit test cases for all of the CRUD Components

Since all the entities are string-based, there are no critical values for input for these components. Therefore black-box test will be skipped and white-box method will be used for the unit test cases. The flow chart and flow graph of the badge CRUD component is as follows:





Node that since all the CRUD components work the same way, these figures may be generalized and thought as interchangeable between components.

The cyclomatic complexity of the flow graph obtained from the flow chart of the Badge CRUD component is as follows:

$$V(G) = E - N + 2$$

E: Edges. There are 8 edges in this flow graph.
N: Nodes. There are 6 nodes in this flow graph.

$$V(G) = 4$$

There are 4 paths, hence 4 possible test cases in this component:

Path 1: 1-2-5-6
Path 2: 1-3-5-6
Path 3: 1-4-5-6
Path 4: 1-6

3.1.1.1.1 Stubs and/or drivers for Badge CRUD component

Since add, delete and edit operations are very simple and trivial operations, there is no need for stubs. Directly querying database upon clicking those buttons would do just fine. The driver is the badge list page.

3.1.1.1.2 Test cases Badge CRUD component

3.1.1.1.2.1 Test case 1 (add)

Following the path 1, test case 1 takes the actions click add button - enter values - update the database - view the new list, respectively.

3.1.1.1.2.2 Test case 2 (delete)

Following the path 2, test case 2 takes the actions click delete button - update the database - view the new list, respectively.

3.1.1.1.2.3 Test case 3 (edit)

Following the path 3, test case 3 takes the actions click edit button - enter values - update the database - view the new list, respectively.

3.1.1.1.2.4 Test case 4 (no action)

Following the path 4, test case 4 takes no actions and simply the list is displayed.

3.1.1.1.3 Purpose of tests for Badge CRUD component

3.1.1.1.3.1 Purpose of the test case 1

Test case 1 simply studies the tries to verify the correctness of the Add operation.

3.1.1.1.3.2 Purpose of the test case 2

Test case 2 simply studies the tries to verify the correctness of the Delete operation.

3.1.1.1.3.3 Purpose of the test case 3

Test case 3 simply studies the tries to verify the correctness of the Edit operation.

3.1.1.1.3.4 Purpose of the test case 4

Test case 4 simply studies the tries to verify the correctness of the case where no operation performed. So when no button is clicked and the list is viewed again, same values should be listed.

3.1.1.1.4 Expected results for Badge CRUD component

3.1.1.1.4.1 Expected results of the test case 1

Expected results of the test case 1 is the proper update of the database with the badge entered by the user in the add operation. When the badge list is viewed afterwards, the new badge should be listed.

3.1.1.1.4.2 Expected results of the test case 2

Expected results of the test case 2 is the proper update of the database with the removal of the badge chosen by the user in the delete operation. When the badge list is viewed afterwards, the deleted badge should not be listed.

3.1.1.1.4.3 Expected results of the test case 3

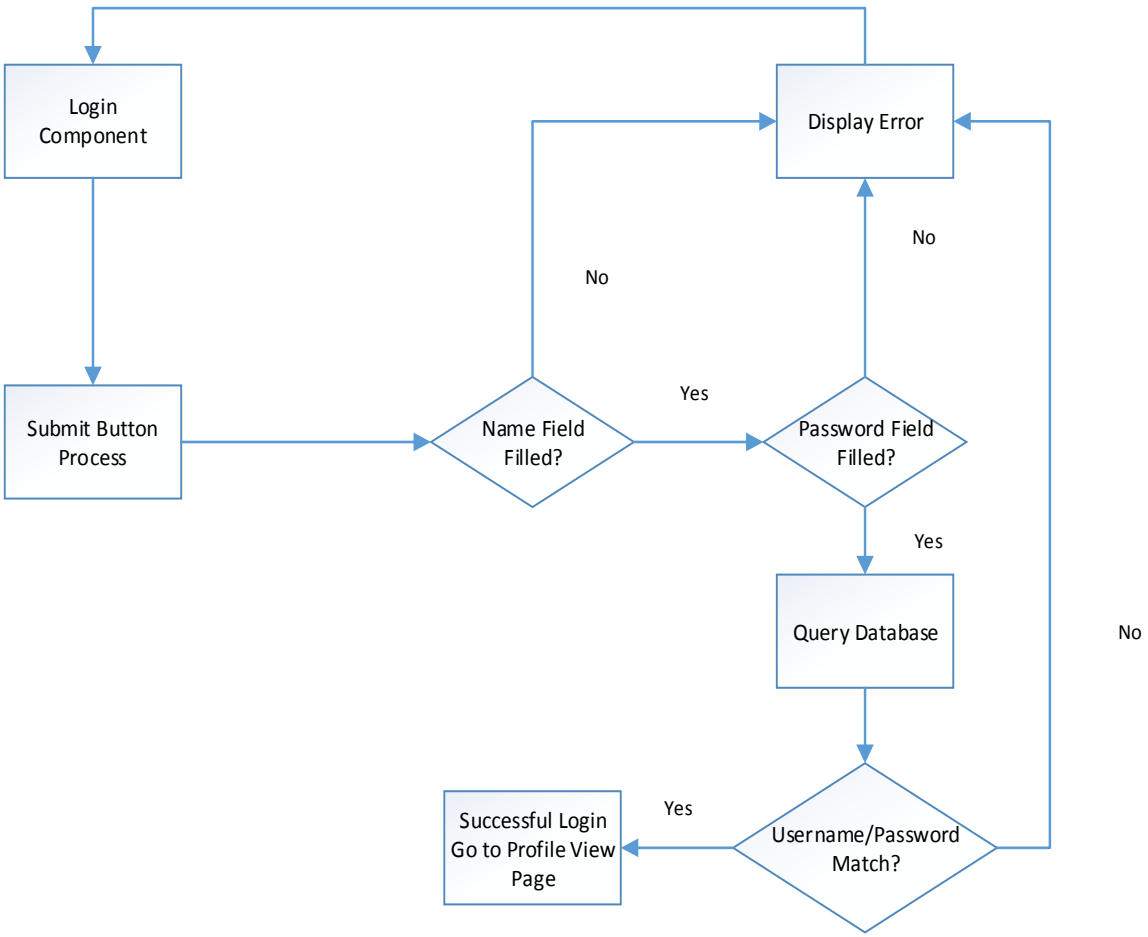
Expected results of the test case 3 is the proper update of the database with the badge edited by the user in the edit operation. When the badge list is viewed afterwards, the updated badge should be listed with the updated information.

3.1.1.1.4.4 Expected results of the test case 4

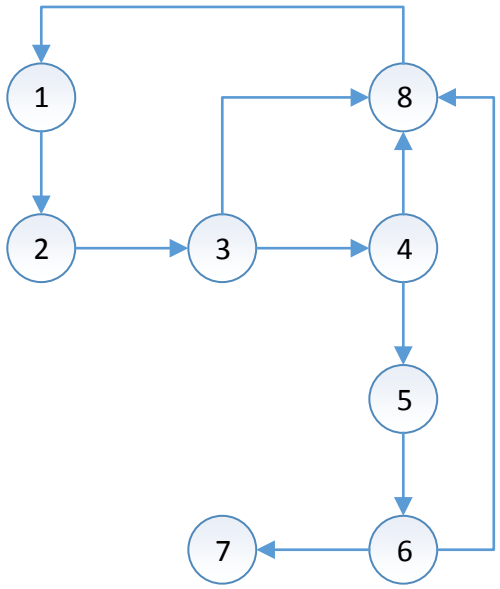
Expected results of the test case 4 are the no-change state. When no action taken, or page is refreshed, the badge list should stay the same.

3.1.1.2 Unit test cases for Login component

Since all the entities are string-based, there are no critical values for input for this component. Therefore black-box test will be skipped and white-box method will be used for the unit test cases. The flow chart and flow graph of the login component is as follows:



The flow graph:



The cyclomatic complexity of the flow graph obtained from the flow chart of the login component is as follows:

$$V(G) = E - N + 2$$

E: Edges. There are 10 edges in this flow graph.
N: Nodes. There are 8 nodes in this flow graph.

$$V(G) = 4$$

There are 4 paths, hence 4 possible test cases in this component:

Path 1: 1-2-3-8-1
Path 2: 1-2-3-4-8-1
Path 3: 1-2-3-4-5-6-8-1
Path 4: 1-2-3-4-5-6-7

3.1.1.2.1 Stubs and/or drivers for Login component

The driver will be the homepage from where the Login component will be called. The stubs will be the following pre-filled input textboxes:

- Empty username
- Empty password
- Incorrect username/password
- Correct username/password

3.1.1.2.2 Test cases Login component

3.1.1.2.2.1 Test case 1 (empty username)

Following the path 1, test case 1 takes the actions click submit button - no username entered - display error message - return login page, respectively.

3.1.1.2.2.2 Test case 2 (empty password)

Following the path 2, test case 2 takes the actions click submit button - username entered – no password entered - display error message - return login page, respectively.

3.1.1.2.2.3 Test case 3 (incorrect username/password)

Following the path 3, test case 3 takes the actions click submit button - username entered - password entered - query database - username/password mismatch - display error - return login page, respectively.

3.1.1.2.2.4 Test case 4 (correct username/password)

Following the path 4, test case 4 takes the actions click submit button - username entered - password entered - query database - username/password match – successful login - go to profile view page, respectively.

3.1.1.2.3 Purpose of tests for Login component

3.1.1.2.3.1 Purpose of the test case 1 & 2

Test case 1 & 2 studies response to the empty input field cases when the submit button is clicked.

3.1.1.2.3.2 Purpose of the test case 3

Test case 3 studies the response of the system to the incorrect login information.

3.1.1.2.3.3 Purpose of the test case 4

Test case 4 studies the successful login process.

3.1.1.2.4 Expected results for Login component

3.1.1.2.4.1 Expected results of the test case 1 & 2

Expected results of the test case 1 & 2 is the proper displaying of empty input field errors. Upon submitting with missing username / password, user should be warned to fill the input textboxes.

3.1.1.2.4.2 Expected results of the test case 3

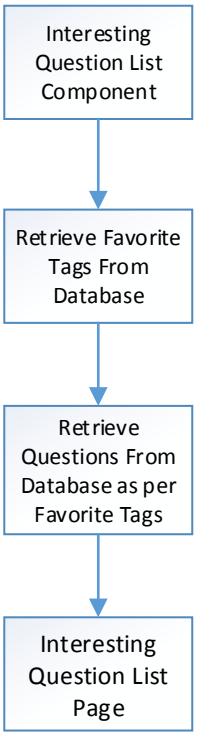
Expected results of the test case 3 is the proper displaying of incorrect user/password combination error.

3.1.1.2.4.3 Expected results of the test case 4

Expected results of the test case 4 is a successful login session. Upon entering valid information, user should be directed to the profile view page.

3.1.1.3 Unit test cases for Interesting Question List Component

White-box method is used for the unit test cases. The aim of using white-box method is the verification of whether the operations are performed right or not. The flow chart and flow graph of the interesting question component is as follows:



Flow chart of Interesting Question List Component

Flow Graph of Interesting Question List Component

The cyclomatic complexity of the flow graph obtained from the flow chart of the interesting question list component is as follows:

$$V(G) = E - N + 2$$

E: Edges. There is 1 edge in this flow graph.
N: Nodes. There are 2 nodes in this flow graph.

$$V(G): 1$$

There is only 1 path in this component:

Path 1: 1-2

3.1.1.3.1 Stubs and/or drivers for Interesting Question List Component

Interesting question list component has not any stubs. This component very simple. User just can see questions list. Driver of the interesting question list component is the interesting question list page.

3.1.1.3.2 The test case Interesting Question List Component

Following the path 1, test case 1 respectively involves update the database-view the interesting question list page.

3.1.1.3.3 Purpose of tests for Interesting Question List Component

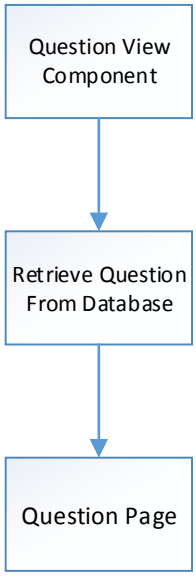
Test case 1 simply studies the tries to verify the correctness of the interesting question list operation by user.

3.1.1.3.4 Expected results for Interesting Question List Component

Expected results of the test case 1 is the retrieving the correct tags and questions from database, performed by user in an interesting question list operation. Interesting Question List consists of question titles according to favorite tags of current user.

3.1.1.4 Unit test cases for Question View Component

The white-box method is used for the unit test cases for question view page. The aim of using white-box method is the verification of whether the operations are performed right or not. The flow chart and flow graph of the question view component is as follows:



Flow chart of Question View Component

Flow Graph of Question View Component

The cyclomatic complexity of the flow graph obtained from the flow chart of the question view component is as follows:

$$V(G) = E - N + 2$$

E: Edges. There are 1 edges in this flow graph.
N: Nodes. There are 2 nodes in this flow graph.

$$V(G) = 1$$

There is only 1 path for this component.

Path 1: 1-2

3.1.1.4.1 Stubs and/or drivers for Question View Component

Since this is a very simple module, there are no stubs. The driver is the question list page where one can simply click the title of the question to view it.

3.1.1.4.2 Test cases for Question View Component

Following the path 1, test case 1 respectively involves click on question title - retrieve question from database - view the question page.

3.1.1.4.3 Purpose of tests for Question View Component

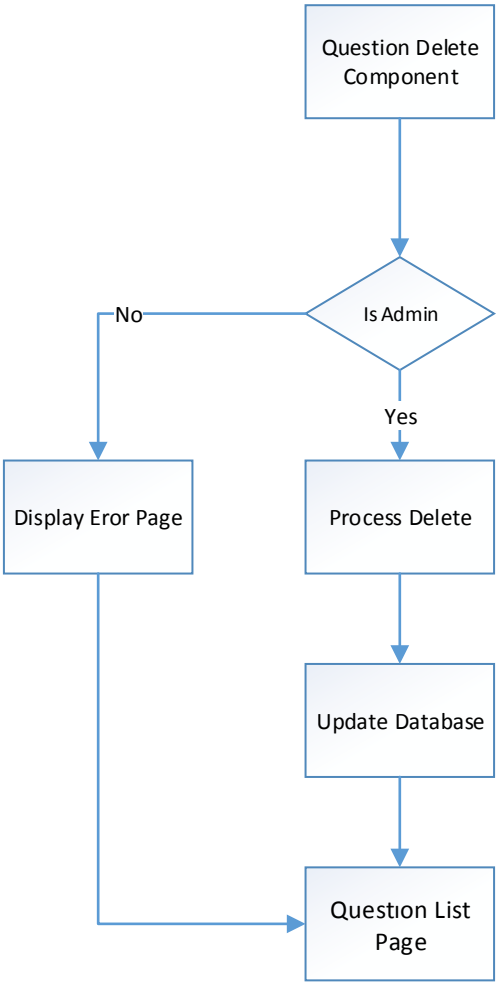
Test case 1 simply studies the tries to verify the correctness of the question view operation by user.

3.1.1.4.4 Expected results for Question View Component

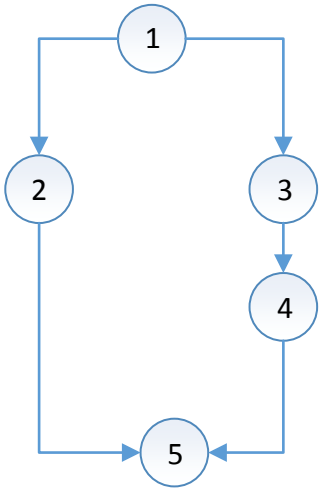
Expected results of the test case 1 is the retrieving the correct information about question, performed by user in a view operation. Question informations are question title, question text, tags, question's answers and question's comments. Also answer question, add comment and vote up/down buttons should be displayed in question view page.

3.1.1.5 Unit test cases for Question Delete Component

White-box method is used for the unit test cases. The aim of using white-box method is the verification of whether the operations are performed right or not. The flow chart and flow graph of the question delete component is as follows:



Flow chart of Question Delete Component



Flow Graph of Question Delete Component

The cyclomatic complexity of the flow graph obtained from the flow chart of the question delete component is as follows:

$$V(G) = E - N + 2$$

E: Edges. There are 5 edges in this flow graph.
N: Nodes. There are 5 nodes in this flow graph.

$$V(G): 2$$

There are two paths:

Path 1: 1-2-5
Path 2: 1-3-4-5

3.1.1.5.1 Stubs and/or drivers for Question Delete Component

Since this is a very simple module, there are no stubs. The driver is the question list page where one can simply click the delete button of the question to delete it.

3.1.1.5.2 Test cases Question Delete Component

3.1.1.5.2.1 Test case 1 (Unauthorized access)

Following the path 1, test case 1 is the attempt for deletion of question by a user.

3.1.1.5.2.2 Test case 2 (Authorized access)

Following the path 2, test case 2 is the attempt for deletion of question by an admin.

3.1.1.5.3 Purpose of tests for Question Delete Component

3.1.1.5.3.1 Purpose of the test case 1

Test case 1 simply studies the tries to verify the correct process of the question delete operation by ordinary user. This operation should not be permitted to users with regular user privileges.

3.1.1.5.3.2 Purpose of the test case 2

Test case 2 simply studies the tries to verify the correctness of the question delete operation by admin.

3.1.1.5.4 Expected results for Question Delete Component

3.1.1.5.4.1 Expected results of the test case 1

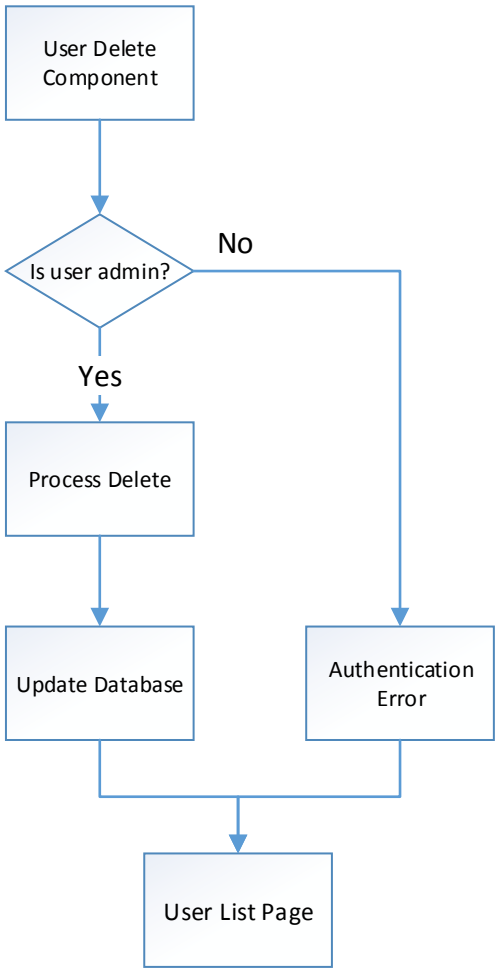
Expected results of the test case 1 is unauthorized operation error and user should be directed question list page after displaying the unauthorized access.

3.1.1.5.4.2 Expected results of the test case 2

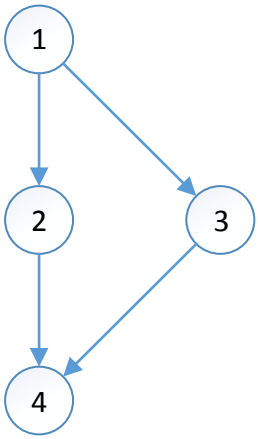
Expected results of the test case 2 is the updating the database with the removal of the question, performed by the admin in the delete operation. When the list of questions is viewed again, the question should be removed.

3.1.1.6 Unit test cases for User Delete Component

The white-box method is used for the unit test cases for user delete page. The aim of using white-box method is the verification of whether the operations are performed right or not. The flow chart and flow graph of the user delete component is as follows:



Flow chart of User Delete Component



Flow Graph of User Delete Component

The cyclomatic complexity of the flow graph obtained from the flow chart of the user delete component is as follows:

$$V(G) = E - N + 2$$

E: Edges. There are 4 edges in this flow graph.
N: Nodes. There are 4 nodes in this flow graph.

$$V(G) = 2$$

There are 2 paths, hence 2 possible test cases in this component:

Path 1: 1-3-4
Path 2: 1-2-4

3.1.1.6.1 Stubs and/or drivers for User Delete Component

Since this is a very simple module, there are no stubs. The driver is the user list page where one can simply click the delete button of the user to delete it.

3.1.1.6.2 Test cases User Delete Component

3.1.1.6.2.1 Test case 1 (Unauthorized access)

Following the path 1, test case 1 is the attempt for deletion of a user by a user.

3.1.1.6.2.2 Test case 2 (Authorized access)

Following the path 2, test case 2 is the attempt for deletion of a user by an admin.

3.1.1.6.3 Purpose of tests for User Delete Component

3.1.1.6.3.1 Purpose of the test case 1

Test case 1 simply studies the tries to verify the correct process of the user delete operation by ordinary user. This operation should not be permitted to users with regular user privileges.

3.1.1.6.3.2 Purpose of the test case 2

Test case 2 simply studies the tries to verify the correctness of the user delete operation by admin.

3.1.1.6.4 Expected results for User Delete Component

3.1.1.6.4.1 Expected results of the test case 1

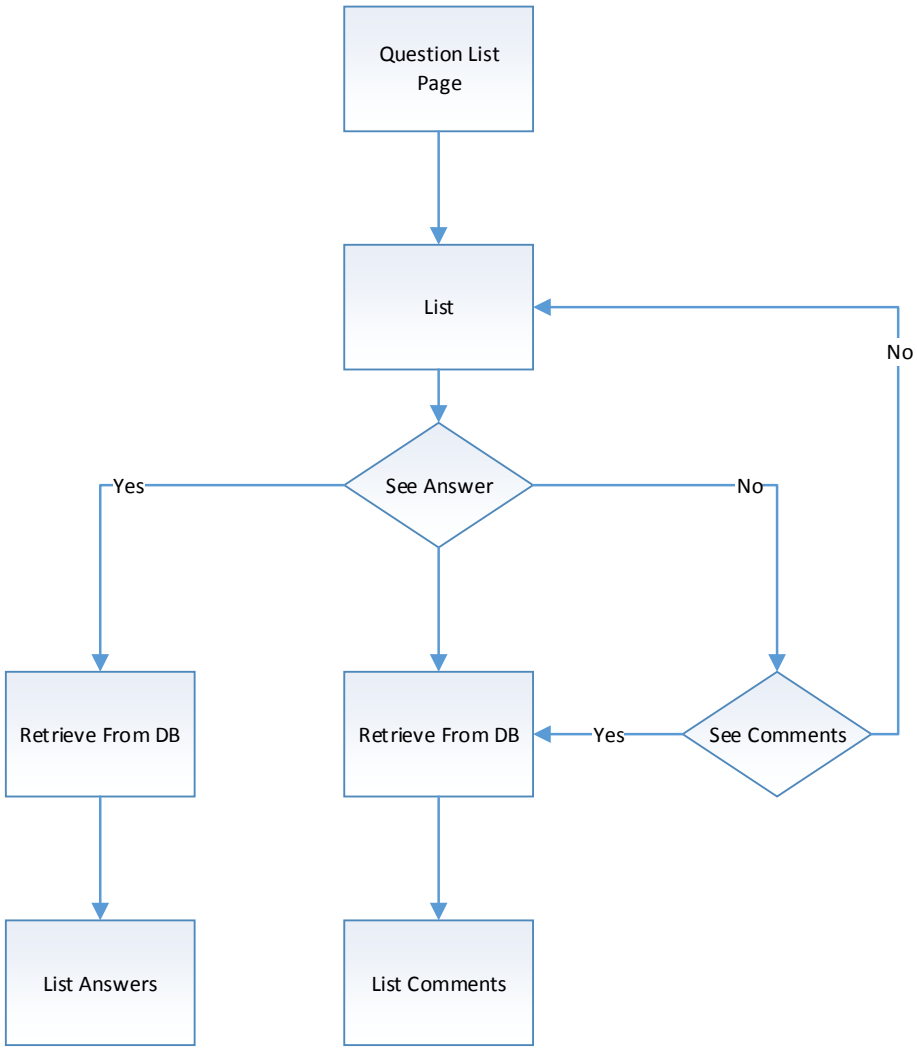
Expected results of the test case 1 is unauthorized operation error and user should be directed user list page after displaying the unauthorized access.

3.1.1.6.4.2 Expected results of the test case 2

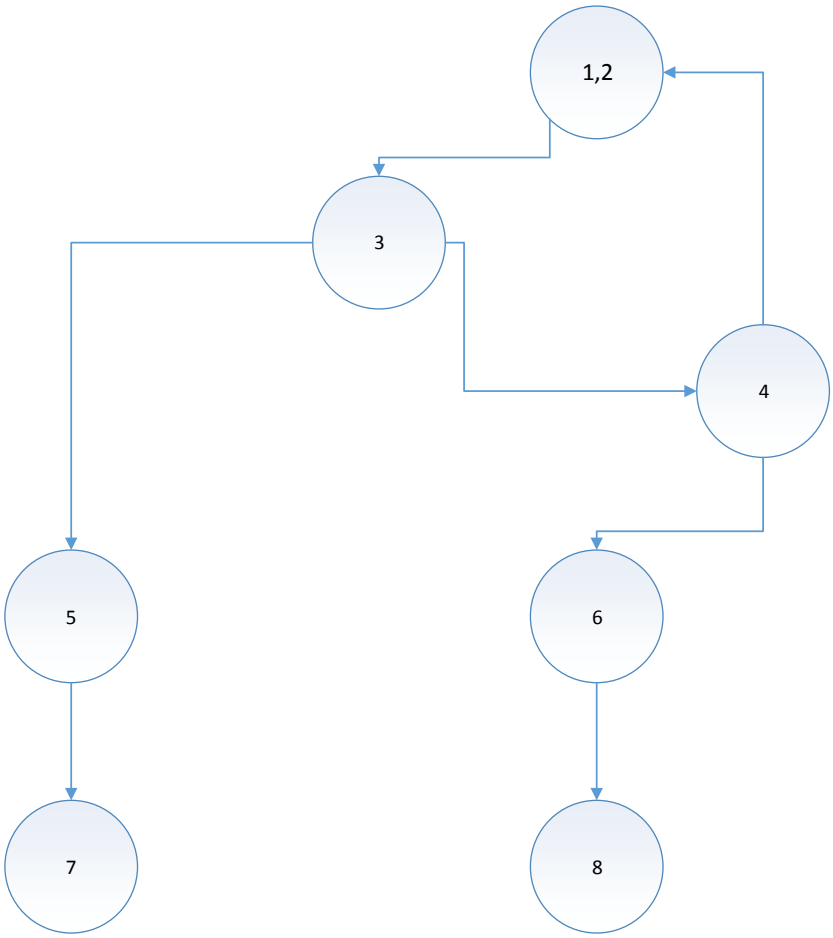
Expected results of the test case 2 is the updating the database with the removal of the user, performed by the admin in the delete operation. When the list of users is viewed again, the user should be removed.

3.1.1.7 Unit test cases for Question List component

White-box method is used for the unit test cases. The aim of using white-box method is the verification of whether the operations are performed right or not. The flow chart and flow graph of the Questions List component is as follows:



Flow chart of Questions List Page



Flow Graph of Questions List Page

The cyclomatic complexity of the flow graph obtained from the flow chart of the questions list component is as follows:

$$V(G) = E - N + 2$$

E: Edges. There are 7 edges in this flow graph.
N: Nodes. There are 7 nodes in this flow graph.

$$V(G): 2$$

There are 2 paths, hence 2 possible test cases in this component:

Path 1: 1-2-3-5-7
Path 2: 1-2-3-4-6-8

3.1.1.7.1 Stubs and/or drivers for Questions List component

Stubs of the questions list component are list, see answers and see comments.
Driver is the homepage.

3.1.1.7.2 Test cases for Questions List component

3.1.1.7.2.1 Test case 1

Following the path 1, test case 1 respectively involves open questions-list-click see answers-retrieve from database-view answers.

3.1.1.7.2.2 Test case 2

Following the path 2, test case 2 respectively involves open questions-list-click see comments-retrieve from database-view comments.

3.1.1.7.3 Purpose of tests for Questions List component

3.1.1.7.3.1 Purpose of the test case 1

Test case 1 simply studies the tries to verify the correctness of the List and See Answers operations.

3.1.1.7.3.2 Purpose of the test case 2

Test case 2 simply studies the tries to verify the correctness of the List and See Comments operations.

3.1.1.7.4 Expected results for Questions List component

3.1.1.7.4.1 Expected results of the test case 1

Expected results of the test case 1 is, after listing the questions, retrieval of answers from the database and listing of them on the page.

3.1.1.7.4.2 Expected results of the test case 2

Expected results of the test case 2 is, after listing the questions, retrieval of the comments from the database and listing of them on the page.

3.1.2 Integration testing

Integration testing is used for finding out the error step by step while combining the components together. Every atomic module is checked separately in this test after integration, enabling finding the exact position of the error.

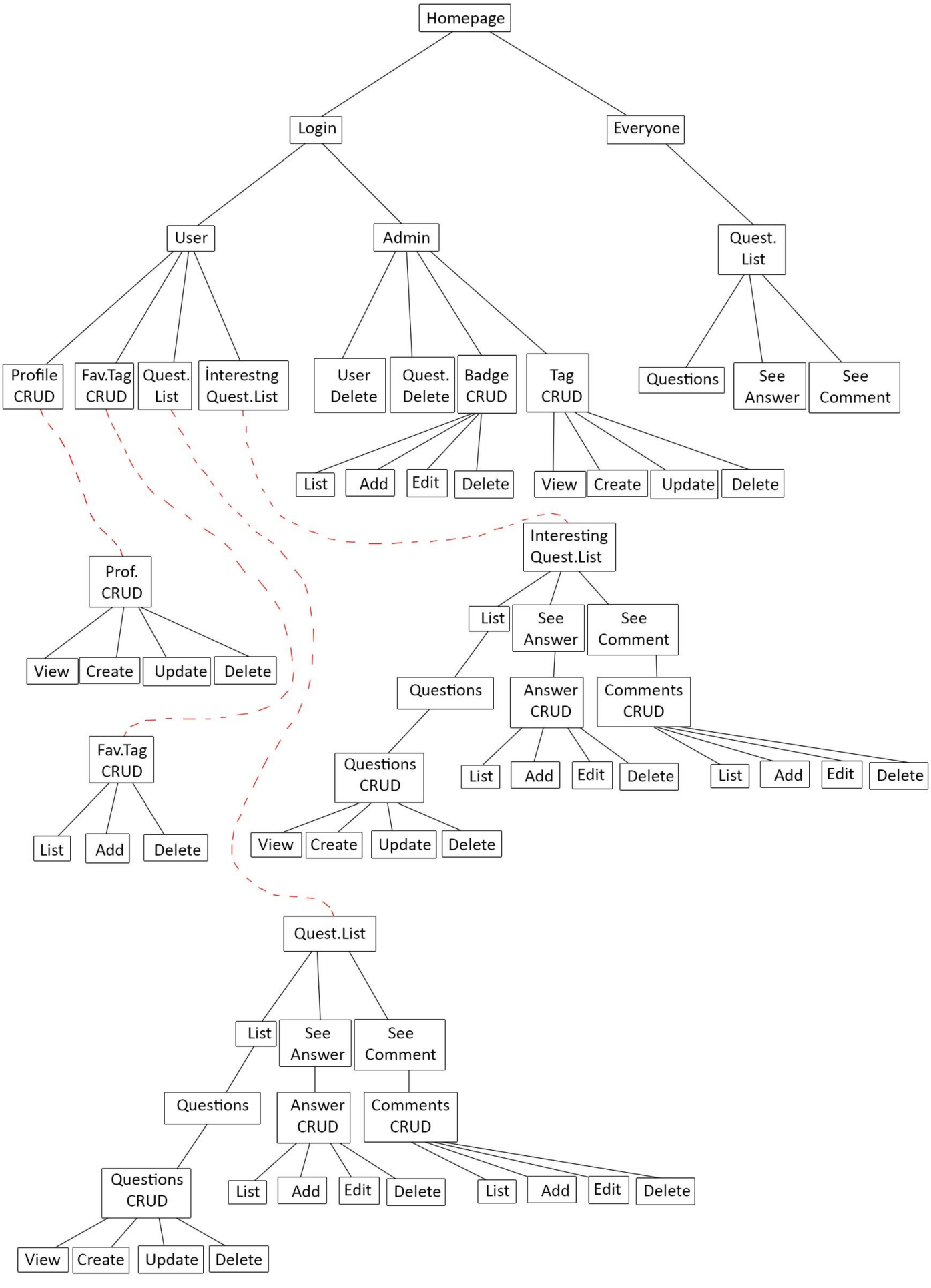
3.1.2.1 Testing procedure for integration

Top-down (depth-first) integration method is used.

3.1.2.2 Stubs and drivers required

- Main driver is the Home Page. Its stubs are Login Page, and everyone's Question List Page.
- There are two groups of stubs of Login.
  - Profile CRUD Page, Favorite Tag CRUD Page, Question List Page and Interesting Question List Page are the stubs of user-level.
  - User Delete, Question Delete, Badge CRUD and Tag CRUD are the stubs of admin-level.
- Stubs of Question List are Question View (list), Answer CRUD (after see answer) and Comments CRUD (after see comments).
- Then, each CRUD page specified here has view, add, edit, delete stubs.

Below is the scheme of integration testing procedure:


Homepage
Login
Everyone
User
Admin
Quest. List
Profile CRUD
Fav.Tag CRUD
Quest. List
İnterestng Quest.List
User Delete
Quest. Delete
Badge CRUD
Tag CRUD
Questions
See Answer
See Comment
List
Add
Edit
Delete
View
Create
Update
Delete
Interesting Quest.List
Prof. CRUD
List
See Answer
See Comment
Questions
Answer CRUD
Comments CRUD
Fav.Tag CRUD
Questions CRUD
List
Add
Edit
Delete
List
Add
Edit
Delete
List
Add
Delete
View
Create
Update
Delete
Quest.List
List
See Answer
See Comment
Questions
Answer CRUD
Comments CRUD
Questions CRUD
List
Add
Edit
Delete
List
Add
Edit
Delete
View
Create
Update
Delete

3.1.2.3 Test cases and their purpose

Since there is no hierarchical structure in this web page, each path is tested from top to bottom separately.

3.1.2.4 Expected results

For each test, the expected result should be error-free performance of each module after a module is added.

3.1.3 Validation testing

Validation testing is applied for checking the outputs with different types of inputs. In this project, different actions at the buttons/text fields are the inputs and related changes are the outputs.

3.1.3.1 Testing procedure for validation

Black-box method is used for the validation tests. The program is tested with four different base conditions. Testing is applied to the add, edit, delete and list operations, since they are the main components that exhibit an input-output relation.

- Checking the input-output when usual values are entered in an operation
- Checking the input-output when no change at all is done in an operation
- Checking the input-output when same values are entered in an operation
- Checking the input-output when extreme strings are entered

3.1.3.2 Expected results

Expected results are as follows:

Basis: usual values			
Test case	Expected result	Notes	Fix action
Add: Add a tag	Tag should be added/Database should be updated	Tag is added/Database is updated	
Edit: Edit a tag	Tag should be edited/Database should be updated	Tag is edited/Database is updated	
Delete: Delete a tag	Tag should be deleted/ Database should be updated	Tag is deleted/Database is updated	
List: List tags	Tags should be listed/No change in database	Tags are listed/Database is not changed	

Basis: no change

Test case	Expected result	Notes	Fix action
Add: Add an empty tag/user/question/answer/comment	It should not be added/Database should not be updated	Empty entry is added/Database is updated	A warning pop-up is added
Edit: Edit a tag/user/question/answer/comment with no change	Database should not be updated	Database is updated	A warning pop-up is added
Delete: no actions			

Basis: same values			
Test case	Expected result	Notes	Fix action
Add: Add a tag/user that is already in the list/database	Tag/user should not be added/Database should not be updated	Same tag/user is added/Database is updated	A warning pop-up is added
Edit: Edit a tag/user and change it to a tag that is in the list/database	Tag/user should not be changed/Database should not be updated	Tag/user is changed/Database is updated	A warning pop-up is added
Delete: no actions			

Basis: extreme strings			
Test case	Expected result	Notes	Fix action
Add: Add a very long tag/username	Tag/username should be added/Database should be updated	Tag/username is added/Database is updated	A warning pop-up is added (for warning, not a fix)
Edit: Edit a tag/username with a very long string	Tag/username should be edited/Database should be updated	Tag/username is edited/Database is updated	A warning pop-up is added (for warning, not a fix)
Delete: no actions			

3.1.3.3 Pass/fail criterion for all validation tests

Since all the tests are examining the buttons, they are related to the database actions. So the pass criteria are the successful update of the database/web page and the related warning message if any required. The fail criteria are the update of the database with redundant values or no update at all. Also not giving the proper warning is another fail criterion.

3.1.4 High-order testing (a.k.a. System Testing)

3.1.4.1 Recovery testing

The program is to be tested with a shutting down of the browser while a process is going on. The case is as follows.

- During a user profile/tag/question/answer/comment creation, web page will be closed. Browsers generally give warning message while leaving a page with an unfinished form operations. However, skipping the warning causes the loss of data entered. No other warning is written.

3.1.4.2 Security testing

Only encryption procedure in the project is the user passwords. All user information and database accesses are provide via these passwords. An unauthorized access page is designed for unauthorized accesses to the admin panel. So, usual users are not able to view admin panels. Possible ways to make an unauthorized access will be examined

3.1.4.3 Stress testing

The program will be tested under a very slow internet connection. Under this circumstance, the database will be examined in terms of how much of the data arrives. Also it will be tested when a huge amount of data is requested to be written to database.

3.1.4.4 Performance testing

The memory usage and running times will be tested. And any possible interaction with other applications will be tested.

3.1.4.5 Pass/fail criterion for all validation tests

Fail criteria of the tests are loss of data, not being able to view the requested data, or very slowly viewing the data.

Pass criteria are the retrieval/update/save of the data in a short time. Also users' quick familiarity with the program is another pass criterion.

3.2 Testing resources and staffing

There are no specialized testing resources. Since the number of modules are not very high, every team member will test each module individually.

3.3 Test record keeping and test log

Microsoft Team Foundation Server is used for issue tracking and document storing as team collaboration software in between project member. Finding and tracking bugs is one of the main objectives of tests. TSF can tack issues or bugs on its servers. This provides us to find and decide importance bugs from test results. TFS can store the all documents with their all versions about project. TFS is also integrated with Microsoft Visual Studio integrated development environment. So firstly test results puts related documents on TFS and then related issues are created if necessary. These issues are assigned to project members to resolve.
Test results are kept in TFS on related tables with information such as test case tables. Other automated test record tools data are also stored in TFS such as performance analysis, screenshots and screen records. SmartBear TestComplete software is used for automated recording during tests.