

**CZ2006**

**Software Engineering**

**System Requirement Specification**

**twoCube™**

|  |  |
| --- | --- |
| **Name:** | **Matric No.:** |
| **Chen Yuhui, Tiffany** | **U1123037H** |
| **Cheok Jia De** | **U1121561B** |
| **June Quak Ren Feng** | **U1121476E** |
| **Khok Hong Jing** | **U1122104L** |
| **Lim Guan** | **U1122655H** |
| **Peh Weileng** | **U1123164B** |
| **Sri Hartati** | **U1123011H** |
| **Wesley Djingga** | **U1121491E** |
| **Xu Ai** | **U1120855F** |

**Date of Submission:**

Contents Page

1. Project Team Information 1

2. Refined Software Requirement Specification 2

2.1 Product Description 2

2.1.1. Product Vision 2

2.1.2. Business Requirements 2

2.1.3. Stakeholders and Users 2

2.1.4. Project Scope 3

2.1.5. Assumptions 3

2.1.6. Constraints 3

2.2 Functional Requirements 4

2.3 Data Requirements 6

2.4 Non-functional requirements 10

2.4.1 Compatibility 10

2.4.2 User interface 10

2.4.3 Security 10

2.4.4 Performance 10

2.4.5 Backup and Recovery 10

2.4.6 Reliability 10

2.4.7 System Maintenance 10

2.5 Interface Requirements 11

2.5.1 User Interfaces 11

2.5.2 Hardware Interfaces 11

2.5.3 Software Interfaces 11

2.6 Use Case Model 12

2.6.1 Use Case Diagram 12

2.6.2 Use Case Description 13

2.7 Glossary 29

2.8 References 29

2.9 Revision History 29

3. Use Case/Activity Diagrams 30

4. Analytical Model – Class Diagrams 31

5. Design Model – Sequence Diagrams 32

5.1 Account Management 32

5.1.1 Create member account 32

5.1.2 Login 33

5.1.3 View member account 34

5.1.4 Update member account 34

5.1.5 Update password 35

5.2 Survey Management 36

5.2.1 Create Survey 36

5.2.2 Create Survey Questions 36

5.2.3 Create Survey Option 37

5.2.4 View Survey 38

5.2.5 Update Survey 39

5.2.6 Close Survey 39

5.3 Survey Report Management 40

5.3.1 Generate Survey Report 40

5.3.2 View Survey Report 41

5.3.3 Export Survey Report 41

5.4 Respondents Actions 42

5.4.1 Submit Survey 42

5.5 Ground Data Logging 43

5.5.1 Data Logging 43

6. Testing – Blackbox, Whitebox and Regression Testing using JUnit or other tools 44

7. Discussion 45

8. Explain how you derived your analytical design and models 45

9. Explain how you performed your UI design 45

10. Describe difficulties encountered and solutions applied 45

11. WBS 46

1. Project Team Information

|  |  |  |  |
| --- | --- | --- | --- |
| **Team Name:** | twoCube™ | | |
| **Project Title:** | NTUSurvey | | |
| **Team Website URL:** | http://twocube1.elasticbeanstalk.com | | |
| **Project Description:** | http://twocube1.elasticbeanstalk.com | | |
|  | | | |
|  | **Name** | **Email Address** | **Contact No.** |
| **Team Leader** | Cheok Jia De | [jdcheok1@e.ntu.edu.sg](mailto:jdcheok1@e.ntu.edu.sg) | 92390354 |
| **Member 1** | Wesley Djingga | [wdjingga1@e.ntu.edu.sg](mailto:wdjingga1@e.ntu.edu.sg) | 92234537 |
| **Member 2** | June Quak Ren Feng | [jquak1@e.ntu.edu.sg](mailto:jquak1@e.ntu.edu.sg) | 97368902 |
| **Member 3** | Peh Weileng | [wlpeh001@e.ntu.edu.sg](mailto:wlpeh001@e.ntu.edu.sg) | 94593932 |
| **Member 4** | Sri Hartati | [hsri001@e.ntu.edu.sg](mailto:hsri001@e.ntu.edu.sg) | 81127957 |
| **Member 5** | Lim Guan | [c110050@e.ntu.edu.sg](mailto:c110050@e.ntu.edu.sg) | 92230282 |
| **Member 6** | Xu Ai | [xuai0001@e.ntu.edu.sg](mailto:xuai0001@e.ntu.edu.sg) | 93362832 |
| **Member 7** | Chen Yuhui, Tiffany | [chen0791@e.ntu.edu.sg](mailto:chen0791@e.ntu.edu.sg) | 92266801 |
| **Member 8** | Khok Hong Jing | [hjkhok1@e.ntu.edu.sg](mailto:hjkhok1@e.ntu.edu.sg) | 96964903 |

1. Refined Software Requirement Specification
2. Product Description
   * 1. Product Vision

twoCube**™** offers a one-stop solution to all survey needs. Getting feedback and responses from the people are the vital activities every organization will conduct. twoCube**™** survey allows users to create surveys, distribute them and view statistics report on the responses.

twoCube**™** is a new player in the survey industry. It strives to assist individuals and organizations by creating a platform to gather and analyze data. It also strives to achieve to make its web survey as simple and convenient for all the users.

Not only measuring customer satisfaction, a survey aids organizations in learning customers’ preferences and uncover new ideas. In addition, individuals are also able to determine a business idea’s feasibility through surveys. Online surveys offer the advantage of convenience and time saving as well. Over the years, online survey websites has come to play in the process of business improvement.

twoCube**™ offers** 3 simple steps:

* Creating a survey
* Collecting data
* Generating reports
  + 1. Business Requirements

The first version of the survey system must be available within three months of the development project launch.

The survey system aims to reduce the manpower in distributing and collecting survey responses thus saving time and efforts.

* + 1. Stakeholders and Users

The targeted users are users (members) and survey respondents.

Users are those who designs and creates the survey whom twoCube**™** named them as members as they have to be registered with twoCube**™** in order to enjoy the features provided.

Survey respondents are people who respond to the survey by answering the survey questions.

Survey respondents can answer the survey questions using personal computers, laptops, and smart phones with web browsers that can easily be accessible at anytime, anywhere.

As for the members, they are able to do the following features with similar equipment as the respondents. They are able to create member account to register with twoCube™, login to the website, manage members account, create surveys, generate survey report, and exporting them to a separate file.

* + 1. Project Scope

The scope is to develop a survey system which provide a platform for users to publish survey question online and have respondents to respond to the question.

The users are able to design survey and customize each question and publish the survey by distributing the link for respondent’s responses. At any point in time, users can view the survey’s responses and also generate reports.

By launching the system, it increases the efficiency of survey data collection and reduces manual workload which can be very tedious.

Smartphone users can also download our android app, <twocube mobile> to access all of the features available on our site as well.

* + 1. Assumptions

*Describe the assumptions that can affect the requirements specified in this SRS.*

* Once creates, a survey cannot be deleted.
* There is no limitation on how many survey questions a survey can consist of.
  + 1. Constraints

*Describe the constraints that can affect the requirements specified in this SRS.*

* To set up a full working system within ten weeks.
  1. Functional Requirements

**Overview**

**For Member:**

* Create Member account
* Login
* Update Member account
* View Member account
* Update password
* Create survey
* Create survey questions
* Create survey options
* View survey
* Generate survey report
* View survey report
* Export survey report
* Data logging

**For respondent:**

* Submit survey
* **Create Member account**

Allow users to input their details

System must save these details into database

* **Login**

Allow users to input username and password

Password will be compared with the password in database

System save username session

* **Update Member account**

System display user’s information

Users can change other details

System will save changes by updating the database

* **View Member account**

Allow users to view the details of their account

System displays details of users account like username, email, etc.

* **Update password**

Allow users to change the password of their account.

System will save the changes by updating the database

* **Create survey**

Allow users to input the title of each question

Allow users to select survey questions type, options, question’s status (whether is compulsory question)

Allow users to publish the survey questions

System will save the survey periodically automatically (optional)

System will save new survey into the database

* **Create survey questions**

Allow users to create different types of surveys questions

The questions can be multiple choice, text, scale and etc.

Allow users to make the question a compulsory question

* **Create survey options**

Allow users to create survey options based on the questions

Allow users to choose the range of scale for scale questions

Allow users to create multiple options for multiple choice questions

* **View survey**

Allow users to view the created survey(s)

* **Generate survey report**

Allow users to collect survey respondent data

System will generate survey report

* **View survey report**

Allow users to view survey report which consist of statistics of question options and how long the users take to answer a question.

System will display generated report

* **Export survey report**

System will export selected survey’s responses in Spreadsheet format

* **Data logging**

System will record respondent’s IP address to ensure one person only does the survey once.

System will also record the time taken to answer each question.

* **Submit survey**

Respondent will answer each survey question and submit it.

* 1. Data Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| **Member** | | | |
| **Field Name** | **Data Type** | **Description** | **Possible Values** |
| Id | int4(10) | Primary Key |  |
| memberFirstName | varchar(255) | Member’s first name |  |
| memberLastName | varchar(255) | Member’s last name |  |
| userName | varchar(255) | Member’s login name |  |
| memberAge | Int4(2) | Member’s age |  |
| dateOfBirthday | timestamp | Member’s birth date |  |
| memberLocation | varchar(255) | Member’s location |  |
| memberEmail | varchar(255 | Member’s email |  |
| memberQuestion | varchar(255 | Secret question for forgetting Member ID or password |  |
| memberAnswer | varchar(255 | Secret answer for forgetting Member ID or password |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Survey** | | | |
| **Field Name** | **Data Type** | **Description** | **Possible Values** |
| Id | int4(10) | Primary Key |  |
| surveyTitle | varchar(255) | Title of the survey |  |
| surveyDescription | varchar(255) | Description of survey |  |
| surveyStatus | bool | Is survey open or closed | True : open  False : closed |
| surveyCreated | timestamp | Date the survey was created |  |
| surveyStartDate | timestamp | Date the survey opens |  |
| surveyEndDate | timestamp | Date the survey ends |  |
| Member\_Id | int4(10) | Foreign Key |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **SurveyQuestion** | | | |
| **Field Name** | **Data Type** | **Description** | **Possible Values** |
| Id | int4(10) | Primary Key |  |
| surveyQuestionType | int4(10)) | Type of survey question | 0 : Radio button  1 : Checkbox  2 : Scale Slider  3 : Numerical Input  4 : Date Input  5 : Scale Radio Button  6 : Text  7 : Textarea  8 : Signature  9 : Image  10 : Dropdown List |
| surveyQuestionIsCompulsory | bool | Is answering the question compulsory or not? | True : Question compulsory  False : Not compulsory |
| surveyQuestionTitle | varchar(255) | Title of survey question |  |
| survey\_Id | int4(10)) | Foreign Key |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **SurveyQuestionOption** | | | |
| **Field Name** | **Data Type** | **Description** | **Possible Values** |
| Id | int4(10) | Primary Key |  |
| surveyQuestionOptionType | int4(10)) | Type of survey question option. Follow the question type or a text input | 0 : According to Qn type  1 : Text input |
| surveyQuestionOptionTitle | varchar(255) | Title of option | True : Question compulsory  False : Not compulsory |
| surveyQuestionOptionTitleType | int4(10)) | Text, Link or Img | 0 : Text  1 : Link  2 : Image |
| surveyQuestionOptionRange | int4(10)) | Range from 0 to this number | Integer larger than 0 |
| surveyQuestionOptionMinText | varchar(255) | If survey question is scale type, the text at the min |  |
| surveyQuestionOptionMaxText | varchar(255) | If survey question is scale type, the text at the max |  |
| surveyQuestion\_Id | int4(10)) | Foreign Key |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **SurveyQuestionResponse** | | | |
| **Field Name** | **Data Type** | **Description** | **Possible Values** |
| Id | int4(10) | Primary Key |  |
| responseIsAnwered | bool |  | True : Answered  False : Unanswered |
| responseType | int4(10) | Check boxes, radio buttons, sliders will be integer type. Date, and text will be String. Question options that allow text response will be both. In this case Integer will be the option they chose, String will be the answer. | 0 : Integer  1 : String  2 : Both |
| responseIntegerValue | int4(10) | Value if response is an integer type |  |
| responseStringValue | varchar(255) | Value if response is a String type |  |
| respondent\_Id | int4(10 | Foreign Key |  |
| surveyQuestion\_Id | int4(10) | Foreign Key |  |

* 1. Non-functional requirements
     1. Compatibility

The website must be compatible with all HTML5 compliant browsers.

The mobile application must be compatible with all android 2.1+ devices.

* + 1. User interface

The user interface must be as familiar as possible to users who have used other web applications and Windows desktop applications. E.g., we will follow the UI guidelines for naming menus, buttons, and dialog boxes whenever possible.

* + 1. Security

Access shall be controlled with usernames and passwords

Access to the database will be done via web services.

* + 1. Performance

The system must be up and running with a 99% uptime.

It must support at least 100 users replying to surveys concurrently without any noticeable lag.

* + 1. Backup and Recovery

There must be a backup server and database to prevent service interruption or loss of data when the main server and database are down.

Downtime must not last more than 30sec when switching from main server to the backup server in case of a breakdown.

* + 1. Reliability

The whole survey system must achieve a 99% success rate. I.e. downtime should not be more than 1% of its total operating time.

System review shall take place monthly. Any lack in performance or reliability shall be addressed and improved on after each review.

* + 1. System Maintenance

Maintenance of the system shall be conducted weekly. Maintenance shall be conducted during off-peak hours e.g. between 12am - 6am.

* 1. Interface Requirements
     1. User Interfaces

*Describe the screen design requirements. Include mock-ups and any report format.*

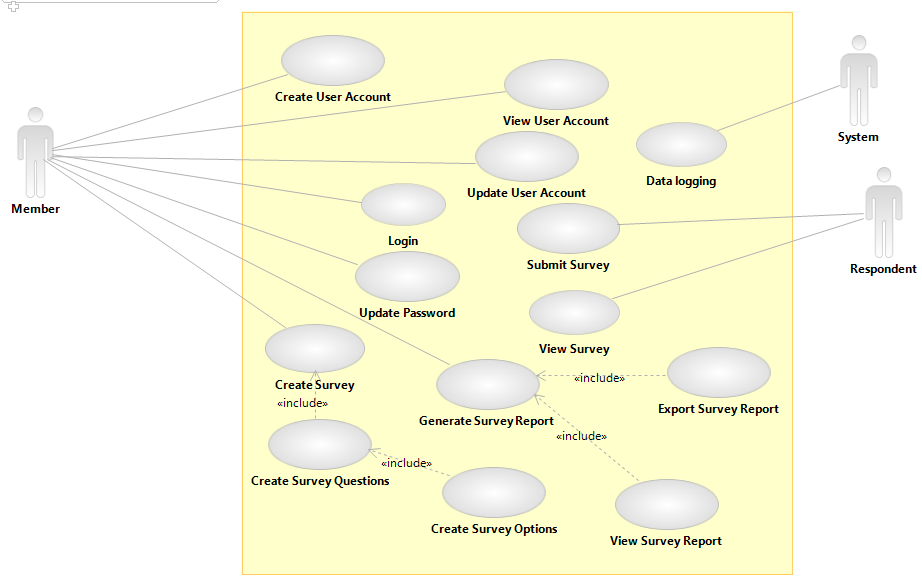
* + 1. Hardware Interfaces

*Describe how the software application interfaces with hardware that exists outside the scope of the system.*

* + 1. Software Interfaces

*Describe how the software application interfaces with software systems that exist outside the scope of the system.*

* 1. Use Case Model
     1. Use Case Diagram



* + 1. Use Case Description

**Use Case List**

|  |  |
| --- | --- |
| Primary Actor | Use Cases |
| Member | Create Member Account |
| Member | Login |
| Member | View Member Account |
| Member | Update Member Account |
| Member | Update Password |
| Member | Create Survey |
| Member | Create Survey Questions |
| Member | Create Survey Options |
| Respondent | View Survey |
| Member | Generate Survey Report |
| Member | View Survey Report |
| Member | Export Survey Report |
| Respondent | Submit Survey |
| System | Data logging |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | UC01 | | |
| Use Case Name: | Create Member Account | | |
| Created By: | Wesley Djingga | Last Updated By: | June Quak |
| Date Created: | 2 September 2012 | Date Last Updated: | 3 October 2012 |

|  |  |
| --- | --- |
| Actors: | Member |
| Description: | The create account use case allows the Member to login and become the registered Member. |
| Trigger: | Member initiates the create account button. |
| Preconditions: | None |
| Postconditions: | 1. Upon success, the Member entered data is stored in the Member’s account. Confirmation is sent to the email address. 2. If the Member fails enters invalid data or cancels the account creation request, no account will be created. |
| Normal Flow: | * 1. The Member enters the required Member account information and request the system to save the entered Member account information.   2. The system validates the entered Member account information.   3. The entered information is then stored in the Member’s account.   4. The system notifies the Member that the account has been created. |
| Alternative Flows: |  |
| Exceptions: | Member cancellation  UC01.0.E.1 Use case ends and account is not created.  Invalid information  UC01.0.E.2 System will prompt Member to re-enter the information.  UC01.0.E.2.1 System will store the information once the entered data is valid.  Invalid information includes:   * Missing information * Username already exist * Email address already exist * Not well informed email address |
| Includes: |  |
| Priority: |  |
| Frequency of Use: |  |
| Business Rules: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | UC02 | | |
| Use Case Name: | Login | | |
| Created By: | June Quak | Last Updated By: | Wesley Djingga |
| Date Created: | 2 September 2012 | Date Last Updated: | 24 September 2012 |

|  |  |
| --- | --- |
| Actors: | Member |
| Description: | This use case documents the process of users to log in to their account to manage the surveys they have created. |
| Trigger: | Member initiates the login feature. |
| Preconditions: | 1. Member must exist. 2. Member must have a valid Member id and password |
| Postconditions: | 1. Member logged on to the system successfully. |
| Normal Flow: | * 1. Member enters the Member id and password when prompt.   2. System verifies Member id and password.   3. System displays Member home page. |
| Alternative Flows: | * + 1. Member logged in through “Facebook authenticator”     2. System verify if any Facebook account is logged in     3. Facebook check for permission     4. Facebook check for authorization code     5. System displays Member home page |
| Exceptions: | Required fields (id and password) not entered.  UC02.0.E.2 System prompts Member to enter the required fields.  Users enter invalid Member id and password.  UC02.0.E.2 System prompts Member to reenter id and password. |
| Includes: |  |
| Priority: |  |
| Frequency of Use: |  |
| Business Rules: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | UC03 | | |
| Use Case Name: | View Member Account | | |
| Created By: | Peh Wei Leng | Last Updated By: | Peh Wei Leng |
| Date Created: | 2 September 2012 | Date Last Updated: | 2 September 2012 |

|  |  |
| --- | --- |
| Actors: | Member |
| Description: | This use case allows users to view their own account. |
| Trigger: | Member initiates the view Member account feature. |
| Preconditions: | 1. Member must be logged on to the system. |
| Postconditions: | 1. Users are able to view the details of their account. |
| Normal Flow: | * 1. Member logged on the system.   2. Member clicked on the view Member account feature.   3. System displays the details of the Member account. |
| Alternative Flows: |  |
| Exceptions: | Users enter invalid Member id and password.  UC03.0.E.1 System prompts Member to reenter id and password. |
| Includes: |  |
| Priority: |  |
| Frequency of Use: |  |
| Business Rules: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | UC04 | | |
| Use Case Name: | Update Member Account | | |
| Created By: | Wesley Djingga | Last Updated By: | Wesley Djingga |
| Date Created: | 2 September 2012 | Date Last Updated: | 2 September 2012 |

|  |  |
| --- | --- |
| Actors: | Member |
| Description: | The manage Member account use case allows Member to update the Member account information that is stored in the Member’s account. |
| Trigger: | The Member initiates the edit button in the Member account information page. |
| Preconditions: | The Member must have login before editing the account’s information. |
| Postconditions: | 1. Upon success, the Member entered data is updated in the Member’s account. 2. If the Member fails enters invalid data or cancels the account creation request, there will be no change to the Member’s account. |
| Normal Flow: | * 1. The Member update the required Member account information and request the system to save the updated Member account information.   2. The system validates the updated Member account information.   3. The updated information is then stored in the Member’s account.   4. The system notifies the Member that the account has been updated. |
| Alternative Flows: |  |
| Exceptions: | Member cancellation  UC04.0.E.1 Member case ends and account is not updated.  Invalid information  UC04.0.E.2 System will prompt Member to re-enter the information.  UC04.0.E.2.1 System will update the information once the entered data is valid.  Invalid information includes:   * Missing information * Username already exist * Not well informed email address |
| Includes: |  |
| Priority: |  |
| Frequency of Use: |  |
| Business Rules: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | UC05 | | |
| Use Case Name: | Update Password | | |
| Created By: | Peh Wei Leng | Last Updated By: | Peh Wei Leng |
| Date Created: | 19 September 2012 | Date Last Updated: | 19 September 2012 |

|  |  |
| --- | --- |
| Actors: | Member |
| Description: | The update password use case allows Member to update the password of their account. |
| Trigger: | The Member initiates the change password button in the Member account information page. |
| Preconditions: | The Member must have login before editing the account’s information. |
| Postconditions: | 1. Upon success, the password is updated in the Member’s account. 2. If the Member enter mismatched password or cancels the update password request, there will be no change to the Member’s account. |
| Normal Flow: | * 1. The Member first enter the old password, then the new one and password confirmation to update the password of Member account and request the system to save the updated one.   2. The system validates if the old password matches with the initial password.   3. The updated information is then stored in the Member’s account.   4. The system notifies the Member that the password has been changed. |
| Alternative Flows: |  |
| Exceptions: | Member cancellation  UC05.0.E.1 Member case ends and password is not updated.  Invalid information  UC05.0.E.2 System will prompt Member to re-enter the information.  UC05.0.E.2.1 System will update the information once the entered data is valid.  Invalid information includes:   * Mismatched password |
| Includes: |  |
| Priority: |  |
| Frequency of Use: |  |
| Business Rules: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | UC06 | | |
| Use Case Name: | Create Survey | | |
| Created By: | June Quak | Last Updated By: | June Quak |
| Date Created: | 2 September 2012 | Date Last Updated: | 2 September 2012 |

|  |  |
| --- | --- |
| Actors: | Member |
| Description: | This use case documents the process of users creating new survey. Each survey must consist of at least one question. |
| Trigger: | Member initiates the create survey feature. |
| Preconditions: | 1. Member must be logged on to the system. |
| Postconditions: | 1. Survey created and posted up for respondents to start their survey. |
| Normal Flow: | * 1. Member initiates the “Create Survey” button.   2. System displays the form for Member to enter survey title and description.   3. Member enters the corresponding fields as needed.   4. System validates the fields Member has entered.   5. System proceeds to UC06 Create Survey Questions to proceed on to create questions for the survey. |
| Alternative Flows: |  |
| Exceptions: | Member exits page before saving.  UC06.0.E.3 Use case ends and survey not created.  Required fields not entered.  UC06.0.E.4 System prompts Member to enter the required fields.  UC06.0.E.4.1 Member reinitiates the create button after entering the required fields. |
| Includes: | UC07 Create Survey Questions, UC08 Create Survey Options |
| Priority: |  |
| Frequency of Use: |  |
| Business Rules: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | UC07 | | |
| Use Case Name: | Create Survey Questions | | |
| Created By: | June Quak | Last Updated By: | June Quak |
| Date Created: | 3 September 2012 | Date Last Updated: | 3 September 2012 |

|  |  |
| --- | --- |
| Actors: | Member |
| Description: | This use case documents the process of a Member creating survey questions. |
| Trigger: | Member initiate the create survey question button. |
| Preconditions: | Survey must exist. |
| Postconditions: | 1. Survey question created. |
| Normal Flow: | * 1. Member initiates the create survey question button.   2. System redirects Member to the create survey question page.   3. Member enters the question.   4. System validates question entered.   5. System proceeds to UC07 Create Survey Options to create options for the question. |
| Alternative Flows: |  |
| Exceptions: | Required fields not entered.  UC07.0.E.1 System prompts Member to enter required fields before proceeding. |
| Includes: | UC08 Create Survey Options |
| Priority: |  |
| Frequency of Use: |  |
| Business Rules: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | UC08 | | |
| Use Case Name: | Create Survey Options | | |
| Created By: | Wesley Djingga | Last Updated By: | Wesley Djingga |
| Date Created: | 3 September 2012 | Date Last Updated: | 3 September 2012 |

|  |  |
| --- | --- |
| Actors: | Member |
| Description: | The create survey options use case will allow Member to select and add the type of options for the respondent to choose. |
| Trigger: | Member initiates the add option button |
| Preconditions: | 1. Survey and survey question is created. |
| Postconditions: | 1. Option is added after the question. |
| Normal Flow: | * 1. Member has to create survey. System proceeds to UC06.   2. Member has to add a question. System proceeds to UC07.   3. System show several types of options and wait for Member selection.   4. Member fills in the option description for the respondent to choose.   5. Option is added.   6. Member can either add another option (repeat from 8.3) or add a question (repeat from 8.2) |
| Alternative Flows: |  |
| Exceptions: | Member does not add any option for the question.  UC08.0.E.1 System prompts the Member to add at least 1 option of any type. |
| Includes: |  |
| Priority: | UC06 Create Survey, UC07 Create Survey Question. |
| Frequency of Use: |  |
| Business Rules: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | UC09 | | |
| Use Case Name: | View Survey | | |
| Created By: | Peh Wei Leng | Last Updated By: | Peh Wei Leng |
| Date Created: | 2 September 2012 | Date Last Updated: | 2 September 2012 |

|  |  |
| --- | --- |
| Actors: | Member |
| Description: | This use case allows users to view the survey they have created. |
| Trigger: | Member initiates the view survey feature. |
| Preconditions: | 1. Member must be logged on to the system. 2. An existing survey created by Member must exist. |
| Postconditions: | 1. Users are able to view the survey that they have created. |
| Normal Flow: | * 1. Member initiates the “View Survey” button.   2. System displays details of the survey questions. |
| Alternative Flows: |  |
| Exceptions: | Users enter invalid Member id and password.  UC09.0.E.1 System prompts Member to reenter id and password.  There is no existing survey.  UC09.0.E.2 System prompts Member to first create a survey. |
| Includes: |  |
| Priority: |  |
| Frequency of Use: |  |
| Business Rules: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | UC10 | | |
| Use Case Name: | Close Survey | | |
| Created By: | Peh Wei Leng | Last Updated By: | Peh Wei Leng |
| Date Created: | 2 September 2012 | Date Last Updated: | 2 September 2012 |

|  |  |
| --- | --- |
| Actors: | Member |
| Description: | This use case document the process of Member closing an existing survey they have created. |
| Trigger: | Member initiates the close survey feature. |
| Preconditions: | 1. Member must be logged on to the system. 2. Survey created by Member must exist. |
| Postconditions: | 1. Status of the survey is changed from “open” to “closed”. 2. Survey is closed successfully, and is not visible to anyone else. |
| Normal Flow: | * 1. Member initiates the “Close Survey” button on the navigation.   2. System displays all the details of the survey questions.   3. Member changes the status of the survey from “open” to “closed”.   4. System prompts Member to confirm the process.   5. System displays a message showing the updated status of survey. |
| Alternative Flows: |  |
| Exceptions: | Member exits page before saving.  UC010.0.E.1 Use case ends and survey status not changed. |
| Includes: |  |
| Priority: |  |
| Frequency of Use: |  |
| Business Rules: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | UC11 | | |
| Use Case Name: | Generate Survey Report | | |
| Created By: | Peh Wei Leng | Last Updated By: | Peh Wei Leng |
| Date Created: | 2 September 2012 | Date Last Updated: | 2 September 2012 |

|  |  |
| --- | --- |
| Actors: | Member |
| Description: | This use case allows Member to generate the statistical report of the data collected. |
| Trigger: | Member initiates the generate report feature. |
| Preconditions: | 1. Survey must exist. |
| Postconditions: | 1. Report generated successfully. |
| Normal Flow: | * 1. Member initiates the “Generate Report” button.   2. System generates the statistical report of the survey questions.   3. Member views the generated report and initiates the “Save Report” button.   4. System saved the generated report. |
| Alternative Flows: |  |
| Exceptions: | Member exits page without initiating the “Save Report” button.  UC11.0.E.1 Use case ends and report not generated. |
| Includes: |  |
| Priority: |  |
| Frequency of Use: |  |
| Business Rules: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | UC12 | | |
| Use Case Name: | View Survey Report | | |
| Created By: | June Quak | Last Updated By: | June Quak |
| Date Created: | 3 September 2012 | Date Last Updated: | 3 September 2012 |

|  |  |
| --- | --- |
| Actors: | Member |
| Description: | This use case allows users to view the generated report. |
| Trigger: | Member initiates the view survey report feature. |
| Preconditions: | 1. Report has to be generated. |
| Postconditions: | 1. Users are able to view the survey report that they have generated. |
| Normal Flow: | * 1. Member initiates the “View Survey Report” button.   2. System displays details of the survey report – for example charts. |
| Alternative Flows: |  |
| Exceptions: |  |
| Includes: | UC11 Generate Survey Report |
| Priority: |  |
| Frequency of Use: |  |
| Business Rules: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | UC13 | | |
| Use Case Name: | Export Survey Results | | |
| Created By: | Wesley Djingga | Last Updated By: | Wesley Djingga |
| Date Created: | 2 September 2012 | Date Last Updated: | 2 September 2012 |

|  |  |
| --- | --- |
| Actors: | Member |
| Description: | The export results use case will allow the Member to generate report of the result in Excel, Word, or other types of file that is selected. |
| Trigger: | Member initiates the export button. |
| Preconditions: | 1. The Member has to be logged in. 2. The Member has to have a survey created before. Status has to be “closed” when exporting. |
| Postconditions: | 1. The survey result is generated in a specified format. |
| Normal Flow: | * 1. The Member will choose which file format the survey will be exported as.   2. The Member will choose what type of data format the survey will show.   3. Member will also filter what kind of information (specific questions or answers) will be exported.   4. The system will generate the specified report in the chosen format.   5. The system will prompt the Member to save the exported survey report. |
| Alternative Flows: |  |
| Exceptions: | Member cancellation  UC13.0.E.1 Member case ends and account is not updated.  Member exits page before closing  UC13.0.E.2 The process will be cancelled, thus the report will not be generated. |
| Includes: | UC11 Generate Survey Report |
| Priority: |  |
| Frequency of Use: |  |
| Business Rules: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | UC14 | | |
| Use Case Name: | Submit Survey | | |
| Created By: | June Quak | Last Updated By: | June Quak |
| Date Created: | 2 September 2012 | Date Last Updated: | 2 September 2012 |

|  |  |
| --- | --- |
| Actors: | Respondent |
| Description: | This use case documents the process of respondents submitting the survey upon survey completion. |
| Trigger: | Respondent initiates the submit button. |
| Preconditions: | 1. Respondent must have completed all required-to-do questions. |
| Postconditions: | 1. Survey submission completed. |
| Normal Flow: | * 1. Respondent initiate the submit button.   2. System verifies respondent has completed all the required questions.   3. System redirect respondents to thank you page. |
| Alternative Flows: |  |
| Exceptions: | Respondents did not answer all required-to-do questions.  UC14.0.E.2 System prompts respondents to complete all required-to-do questions before submission.  UC14.0.E.2.1 Respondents resubmit the survey upon completion. |
| Includes: |  |
| Priority: |  |
| Frequency of Use: |  |
| Business Rules: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | UC15 | | |
| Use Case Name: | Data Logging | | |
| Created By: | June Quak | Last Updated By: | June Quak |
| Date Created: | 9 September 2012 | Date Last Updated: | 9 September 2012 |

|  |  |
| --- | --- |
| Actors: | System |
| Description: | This use case documents the process of system data logging what Member does. |
| Trigger: | Respondent initiates to do the survey. |
| Preconditions: | 1. One respondent must only do survey once. |
| Postconditions: | 1. Respondent’s IP address logged. 2. Time taken for respondents to complete the survey is logged. |
| Normal Flow: | * 1. Respondents initiate to do the survey.   2. System validate if IP address has already been logged.   3. Respondents start completing the survey.   4. System times the speed of respondents completing the survey. |
| Alternative Flows: |  |
| Exceptions: | IP address already logged.  UC15.0.E.2 System prompts respondents that they have already did the survey once.  UC15.E.2.1 System closes the window and thank respondent for their participation. |
| Includes: |  |
| Priority: |  |
| Frequency of Use: |  |
| Business Rules: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

* 1. Glossary

Define all terms and acronyms required to interpret the SRS properly.  This is the (problem) domain dictionary.

* 1. References

Provide a list of all documents and other sources of information referenced in the SRS and utilized in developing the SRS. Include for each the document number, title, date and author.

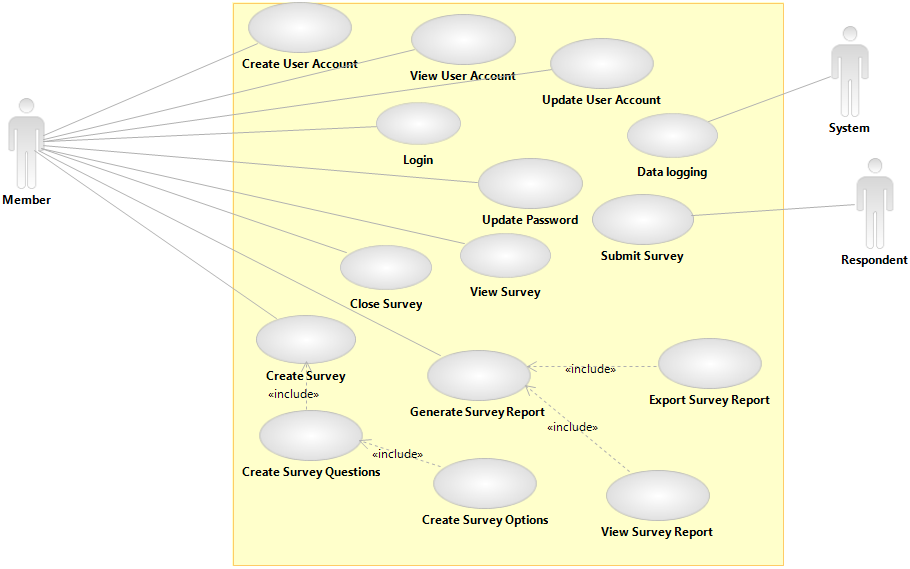
|  |  |  |  |
| --- | --- | --- | --- |
| **Document No.** | **Document Title** | **Date** | **Author** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

* 1. Revision History

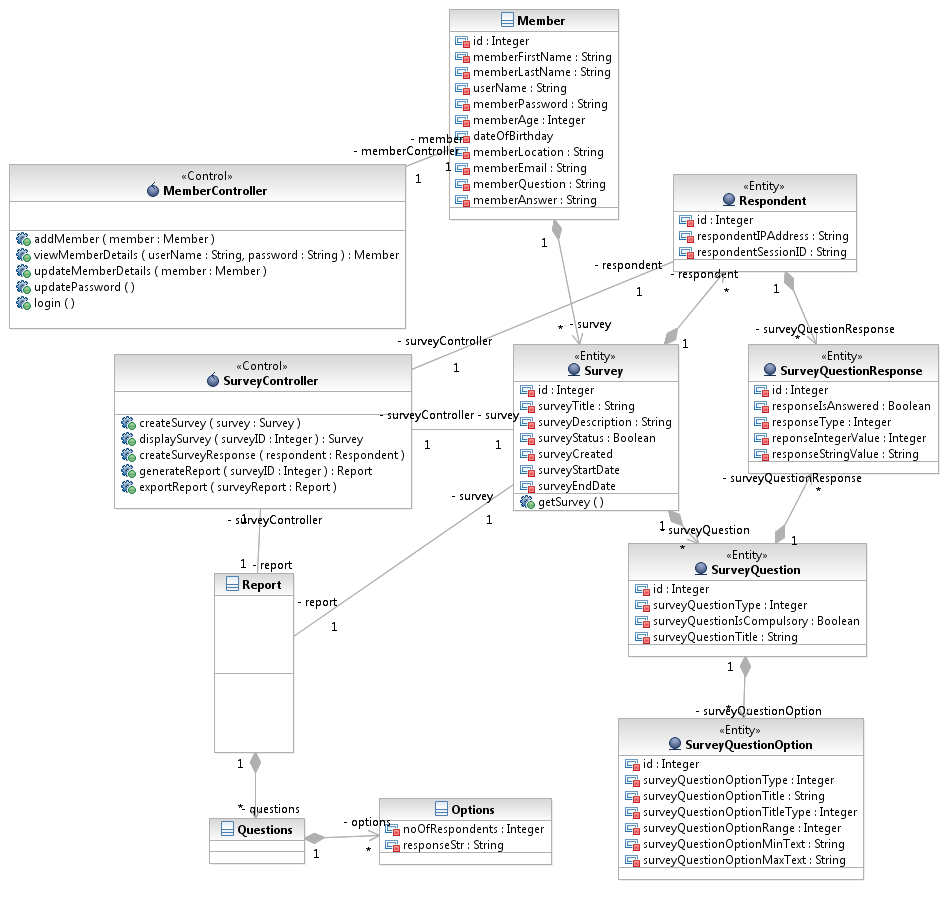
Identify changes to the SRS.

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Name** | **Description** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

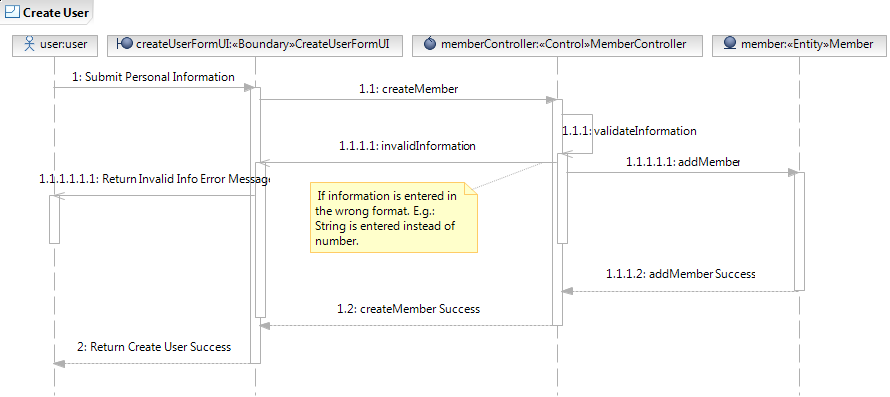
1. Use Case/Activity Diagrams

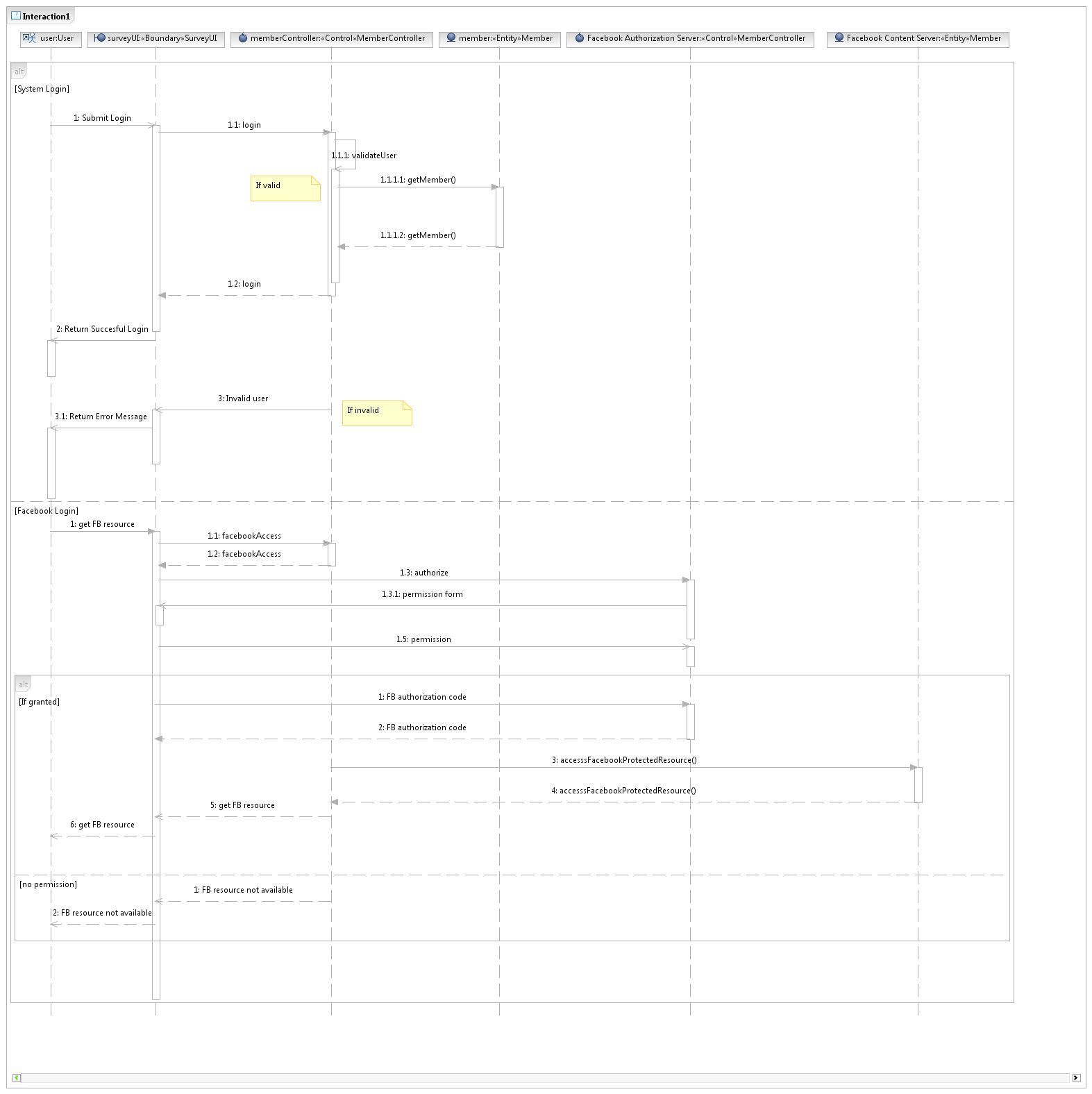


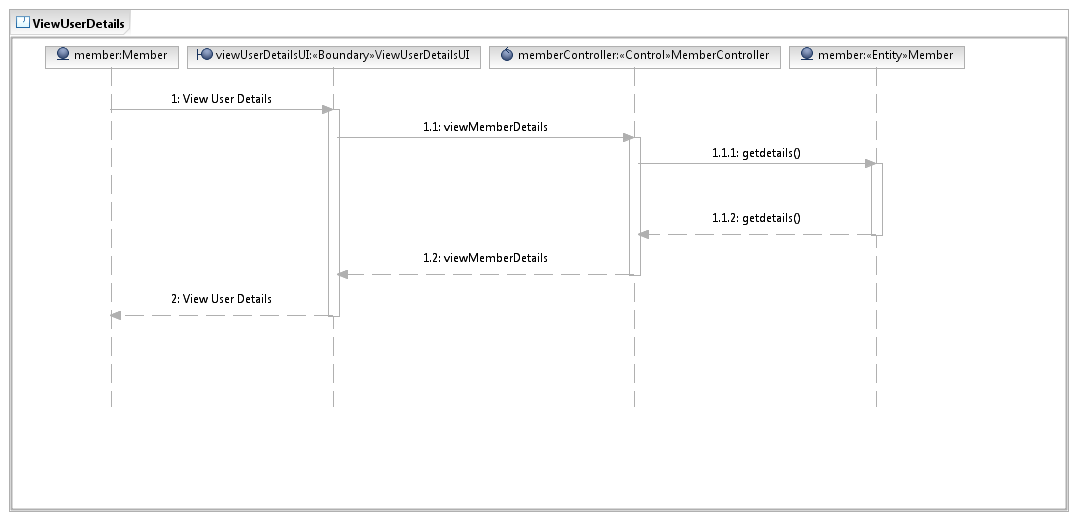
1. Analytical Model – Class Diagrams



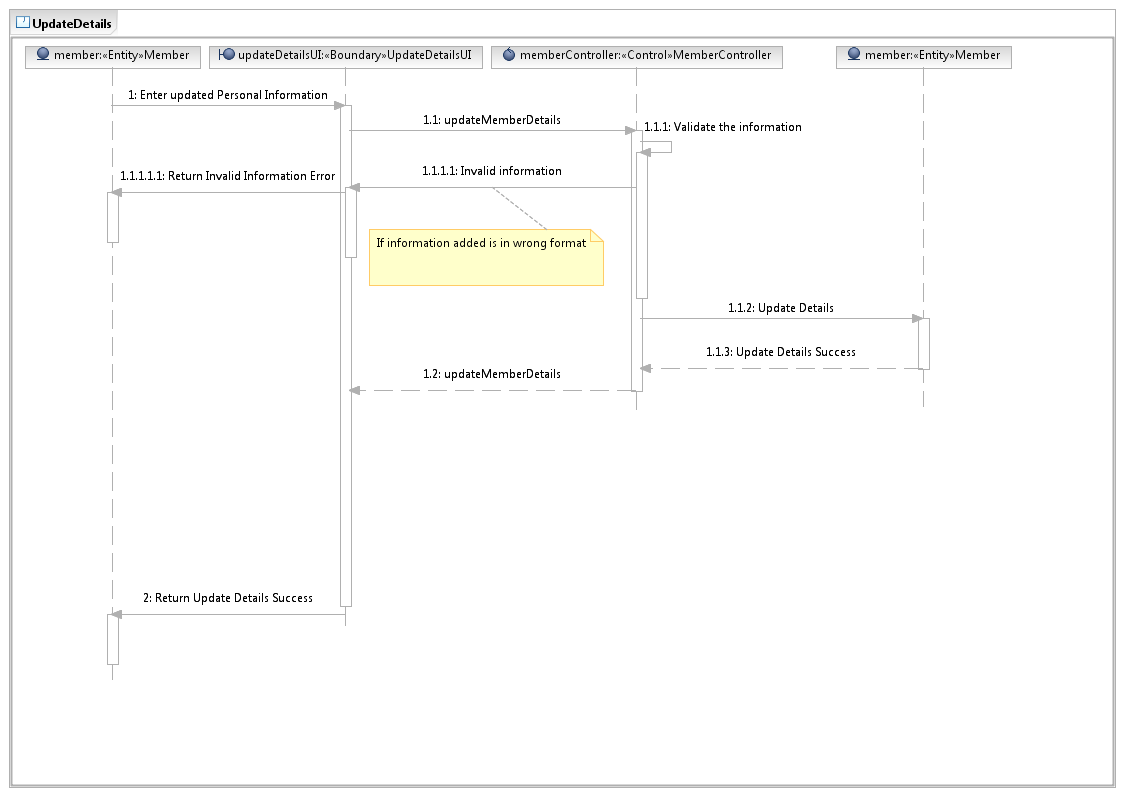
1. Design Model – Sequence Diagrams
   1. Account Management
      1. Create member account



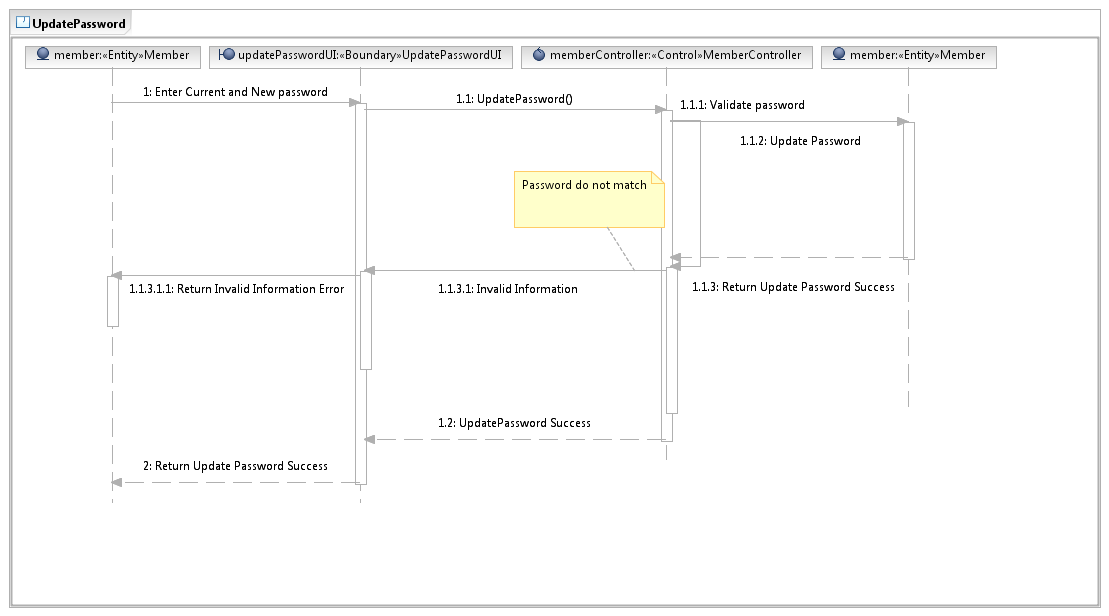
* + 1. Login
    2. View member account



* + 1. Update member account



* + 1. Update password



* 1. Survey Management
     1. Create Survey



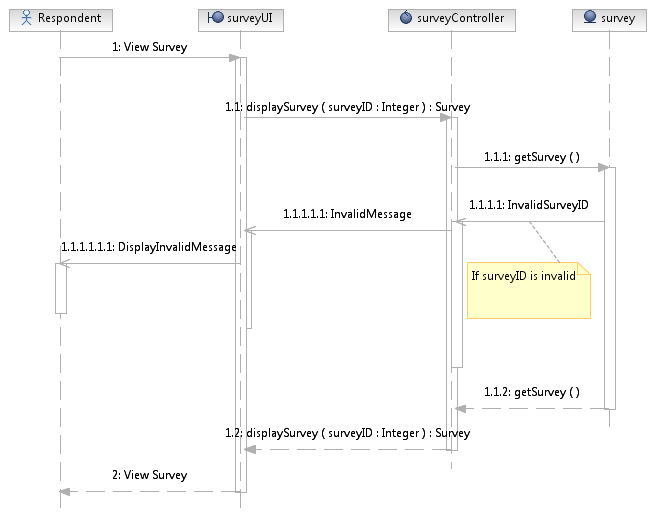
* + 1. Create Survey Questions



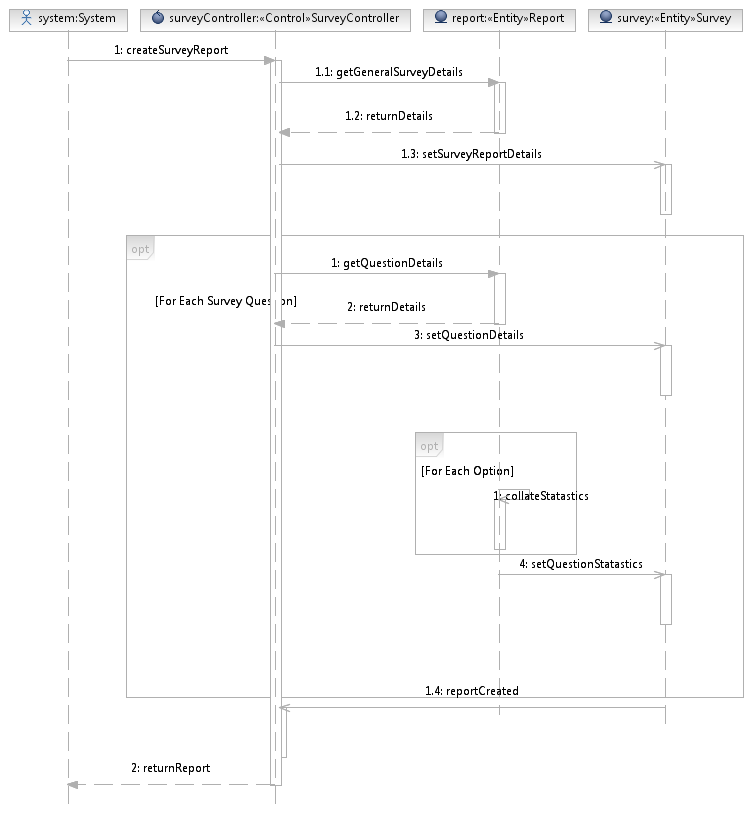
* + 1. Create Survey Option



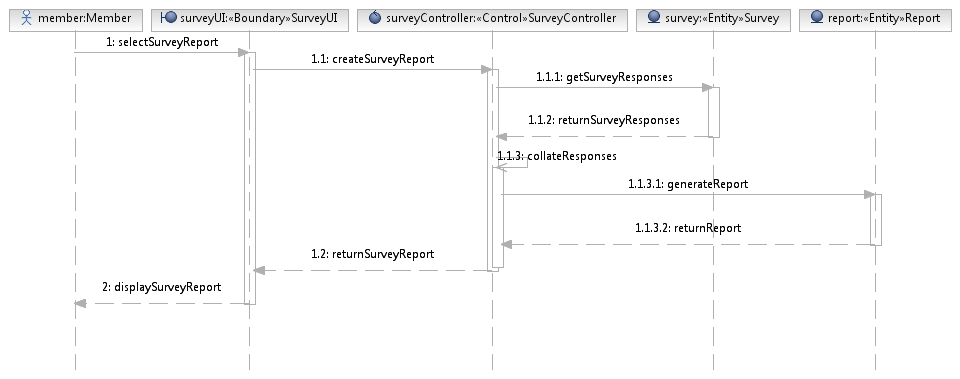
* + 1. View Survey



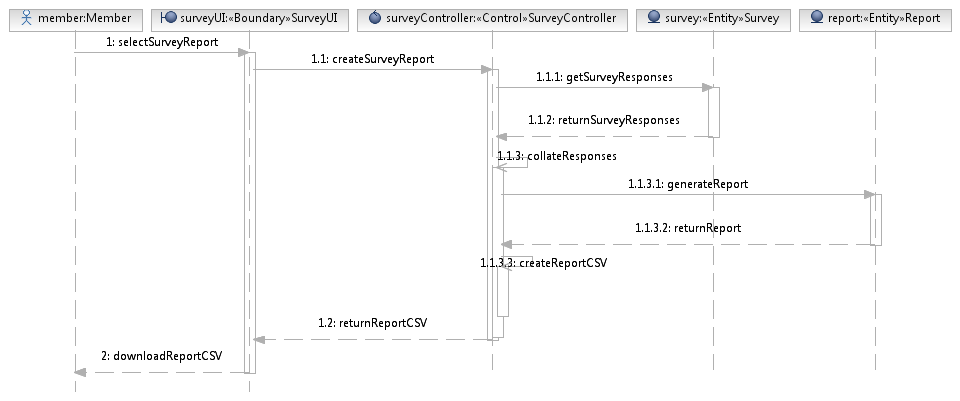
* 1. Survey Report Management
     1. Generate Survey Report



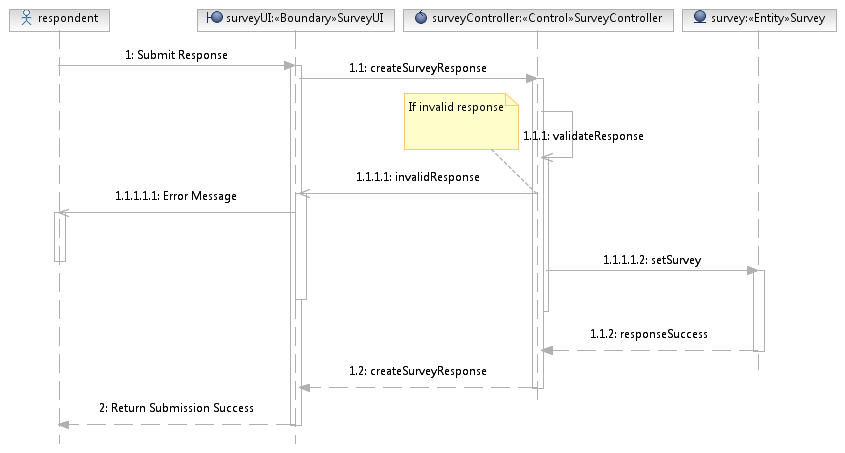
* + 1. View Survey Report



* + 1. Export Survey Report



* 1. Respondents Actions
     1. Submit Survey



* 1. Ground Data Logging
     1. Data Logging

1. Testing
   * + 1. Unit testing

namespace twoCube.Entities

{

using NUnit.Framework;

[TestFixture]

public class SurveyTest

{

[Test]

public void getSurveyById()

{

using (var session = FluentNHibernateConfiguration.InitFactory.sessionFactory.OpenSession())

{

JavaScriptSerializer js = new JavaScriptSerializer();

var survey = Entities.Survey.GetById(session, 1);

survey.respondentList = null;

foreach(var question in survey.surveyQuestionList)

{

question.surveyQuestionResponseList = null;

}

Assert.AreEqual(1, survey.Id);

Assert.AreEqual(10, survey.surveyQuestionList.Count);

Assert.AreEqual("Sample Survey", survey.surveyTitle);

Context.Response.Write(js.Serialize(survey));

}

}

[Test]

public void getSurveyList()

{

using (var session = FluentNHibernateConfiguration.InitFactory.sessionFactory.OpenSession())

{

JavaScriptSerializer js = new JavaScriptSerializer();

var member = Entities.Member.GetByHash(session, "FF4AA02FCD977DF9B1B3F54D9AEDAFB8");

List<Survey> surveyList = member.memberSurveyList.ToList();

List<SurveyResponse> surveyResponseList = new List<SurveyResponse>();

foreach (var survey in surveyList)

{

surveyResponseList.Add(new SurveyResponse {Id = survey.Id, surveyName = survey.surveyTitle, surveyStatus = survey.surveyStatus });

}

Assert.AreEqual(3, surveyResponseList.Count);

Assert.AreEqual(1, surveyResponseList.Get(0).Id);

Assert.AreEqual(98, surveyResponseList.Get(1).Id);

Assert.AreEqual(246, surveyResponseList.Get(2).Id);

Assert.AreEqual("Sample Survey", surveyResponseList.Get(0).surveyTitle);

Assert.AreEqual("Test Survey", surveyResponseList.Get(1).surveyTitle);

Assert.AreEqual("This is a test", surveyResponseList.Get(2).surveyTitle);

Context.Response.Write(js.Serialize(surveyResponseList));

}

}

/\*

formString:{"id":0,"name":"Untitled Form","description":"This is your form description. Click here to edit.","redirect":"","success\_message":"Success! Your submission has been saved!","password":"","frame\_height":"","unique\_ip":0,"captcha":0}

elementString{"elements":[{"title":"Text","guidelines":"","size":"medium","is\_required":"0","is\_unique":"0","is\_private":"0","type":"text","object":"","position":0,"id":0,"is\_db\_live":"0","default\_value":"","constraint":"","options":[{"option":"First option","is\_default":0,"is\_db\_live":"0","id":"0"},{"option":"Second option","is\_default":0,"is\_db\_live":"0","id":"0"},{"option":"Third option","is\_default":0,"is\_db\_live":"0","id":"0"}]},{"title":"Number","guidelines":"","size":"medium","is\_required":"0","is\_unique":"0","is\_private":"0","type":"number","object":"","position":1,"id":1,"is\_db\_live":"0","default\_value":"","constraint":"","options":[{"option":"First option","is\_default":0,"is\_db\_live":"0","id":"0"},{"option":"Second option","is\_default":0,"is\_db\_live":"0","id":"0"},{"option":"Third option","is\_default":0,"is\_db\_live":"0","id":"0"}]},{"title":"Paragraph","guidelines":"","size":"medium","is\_required":"0","is\_unique":"0","is\_private":"0","type":"textarea","object":"","position":2,"id":2,"is\_db\_live":"0","default\_value":"","constraint":"","options":[{"option":"First option","is\_default":0,"is\_db\_live":"0","id":"0"},{"option":"Second option","is\_default":0,"is\_db\_live":"0","id":"0"},{"option":"Third option","is\_default":0,"is\_db\_live":"0","id":"0"}]},{"title":"Checkboxes","guidelines":"","size":"medium","is\_required":"0","is\_unique":"0","is\_private":"0","type":"checkbox","object":"","position":3,"id":3,"is\_db\_live":"0","default\_value":"","constraint":"","options":[{"option":"First option","is\_default":0,"is\_db\_live":"0","id":"0"},{"option":"Second option","is\_default":0,"is\_db\_live":"0","id":"0"},{"option":"Third option","is\_default":0,"is\_db\_live":"0","id":"0"}]},{"title":"Multiple Choice","guidelines":"","size":"medium","is\_required":"0","is\_unique":"0","is\_private":"0","type":"radio","object":"","position":4,"id":104,"is\_db\_live":"0","default\_value":"","constraint":"","options":[{"option":"First option","is\_default":0,"is\_db\_live":"0","id":"0"},{"option":"Second option","is\_default":0,"is\_db\_live":"0","id":"0"},{"option":"Third option","is\_default":0,"is\_db\_live":"0","id":"0"}]},{"title":"Drop Down","guidelines":"","size":"medium","is\_required":"0","is\_unique":"0","is\_private":"0","type":"select","object":"","position":5,"id":105,"is\_db\_live":"0","default\_value":"","constraint":"","options":[{"option":"First option","is\_default":0,"is\_db\_live":"0","id":"0"},{"option":"Second option","is\_default":0,"is\_db\_live":"0","id":"0"},{"option":"Third option","is\_default":0,"is\_db\_live":"0","id":"0"}]},{"title":"Slider","guidelines":"","size":"large","is\_required":"0","is\_unique":"0","is\_private":"0","type":"slider","object":"","position":6,"id":106,"is\_db\_live":"0","default\_value":"","constraint":"100","options":[{"option":"First option","is\_default":0,"is\_db\_live":"0","id":"0"},{"option":"Second option","is\_default":0,"is\_db\_live":"0","id":"0"},{"option":"Third option","is\_default":0,"is\_db\_live":"0","id":"0"}],"size\_min":"10","size\_max":"100"},{"title":"Date","guidelines":"","size":"medium","is\_required":"0","is\_unique":"0","is\_private":"0","type":"date","object":"","position":7,"id":107,"is\_db\_live":"0","default\_value":"","constraint":"","options":[{"option":"First option","is\_default":0,"is\_db\_live":"0","id":"0"},{"option":"Second option","is\_default":0,"is\_db\_live":"0","id":"0"},{"option":"Third option","is\_default":0,"is\_db\_live":"0","id":"0"}]},{"title":"Image Radio","guidelines":"","size":"medium","is\_required":"0","is\_unique":"0","is\_private":"0","type":"img\_radio","object":"","position":8,"id":108,"is\_db\_live":"0","default\_value":"","constraint":"images/default\_img.jpg","options":[{"option":"First option","is\_default":0,"is\_db\_live":"0","id":"0"},{"option":"Second option","is\_default":0,"is\_db\_live":"0","id":"0"},{"option":"Third option","is\_default":0,"is\_db\_live":"0","id":"0"}],"img\_radio\_Unsatisfactory":"images/default\_img.jpg","img\_radio\_Neutral":"images/default\_img.jpg","img\_radio\_Satisfactory":"images/default\_img.jpg","img\_radio\_vSatisfactory":"images/default\_img.jpg"},{"title":"Scaler","guidelines":"","size":"medium","is\_required":"0","is\_unique":"0","is\_private":"0","type":"scaler","object":"","position":9,"id":109,"is\_db\_live":"0","default\_value":"","constraint":"Best","options":[{"option":"First option","is\_default":0,"is\_db\_live":"0","id":"0"},{"option":"Second option","is\_default":0,"is\_db\_live":"0","id":"0"},{"option":"Third option","is\_default":0,"is\_db\_live":"0","id":"0"}],"scaleAmount":"7","mintext":"Worst","maxtext":"Best"},{"title":"Satisfactory","guidelines":"","size":"medium","is\_required":"0","is\_unique":"0","is\_private":"0","type":"satisfactory","object":"","position":10,"id":110,"is\_db\_live":"0","default\_value":"","constraint":"Very Satisfactory","options":[{"option":"First option","is\_default":0,"is\_db\_live":"0","id":"0"},{"option":"Second option","is\_default":0,"is\_db\_live":"0","id":"0"},{"option":"Third option","is\_default":0,"is\_db\_live":"0","id":"0"}],"satisfactory\_Unsatisfactory":"Unsatisfactory","satisfactory\_Neutral":"Neutral","satisfactory\_Satisfactory":"Satisfactory","satisfactory\_vSatisfactory":"Very Satisfactory"},{"title":"Image Checkbox","guidelines":"","size":"medium","is\_required":"0","is\_unique":"0","is\_private":"0","type":"img\_checkbox","object":"","position":11,"id":111,"is\_db\_live":"0","default\_value":"","constraint":"images/default\_img.jpg","options":[{"option":"First option","is\_default":0,"is\_db\_live":"0","id":"0"},{"option":"Second option","is\_default":0,"is\_db\_live":"0","id":"0"},{"option":"Third option","is\_default":0,"is\_db\_live":"0","id":"0"}],"satisfactory\_Unsatisfactory":"images/default\_img.jpg","satisfactory\_Neutral":"images/default\_img.jpg","satisfactory\_Satisfactory":"images/default\_img.jpg","satisfactory\_vSatisfactory":"images/default\_img.jpg"}]}

memberHash:FF4AA02FCD977DF9B1B3F54D9AEDAFB8

\*/

[Test]

public void submitSurvey()

{

using (var session = FluentNHibernateConfiguration.InitFactory.sessionFactory.OpenSession())

{

using (var transaction = session.BeginTransaction())

{

JObject jsonObject = JObject.Parse(elements);

var survey = new Entities.Survey { surveyTitle = jsonObject.SelectToken("name").ToString(), surveyDescription = jsonObject.SelectToken("description").ToString() };

JToken jToken;

jsonObject = JObject.Parse(elements);

var questionList = jsonObject.SelectToken("elements").ToList();

foreach (var question in questionList)

{

switch (question.SelectToken("type").ToString())

{

case "number":

{

var surveyQuestion = new Entities.SurveyQuestion { surveyQuestionTitle = question.SelectToken("title").ToString(), surveyQuestionType = 3, surveyQuestionIsCompulsory = question.SelectToken("title").ToString() == "1" };

surveyQuestion.surveyQuestionOptionList.Add(new Entities.SurveyQuestionOption { surveyQuestionOptionTitle = "" });

survey.surveyQuestionList.Add(surveyQuestion);

break;

}

case "radio":

{

var surveyQuestion = new Entities.SurveyQuestion { surveyQuestionTitle = question.SelectToken("title").ToString(), surveyQuestionType = 0, surveyQuestionIsCompulsory = question.SelectToken("title").ToString() == "1" };

foreach (var option in question.SelectToken("options").ToList())

{

surveyQuestion.surveyQuestionOptionList.Add(new Entities.SurveyQuestionOption { surveyQuestionOptionTitle = option.SelectToken("option").ToString() });

}

survey.surveyQuestionList.Add(surveyQuestion);

break;

}

case "checkbox":

{

var surveyQuestion = new Entities.SurveyQuestion { surveyQuestionTitle = question.SelectToken("title").ToString(), surveyQuestionType = 1, surveyQuestionIsCompulsory = question.SelectToken("title").ToString() == "1" };

foreach (var option in question.SelectToken("options").ToList())

{

surveyQuestion.surveyQuestionOptionList.Add(new Entities.SurveyQuestionOption { surveyQuestionOptionTitle = option.SelectToken("option").ToString() });

}

survey.surveyQuestionList.Add(surveyQuestion);

break;

}

case "date":

{

var surveyQuestion = new Entities.SurveyQuestion { surveyQuestionTitle = question.SelectToken("title").ToString(), surveyQuestionType = 4, surveyQuestionIsCompulsory = question.SelectToken("title").ToString() == "1" };

surveyQuestion.surveyQuestionOptionList.Add(new Entities.SurveyQuestionOption { surveyQuestionOptionTitle = "" });

survey.surveyQuestionList.Add(surveyQuestion);

break;

}

case "slider":

{

var surveyQuestion = new Entities.SurveyQuestion { surveyQuestionTitle = question.SelectToken("title").ToString(), surveyQuestionType = 2, surveyQuestionIsCompulsory = question.SelectToken("title").ToString() == "1" };

surveyQuestion.surveyQuestionOptionList.Add(new Entities.SurveyQuestionOption { surveyQuestionOptionTitle = "", surveyQuestionOptionMaxText = question.SelectToken("size\_max") != null ? question.SelectToken("size\_max").ToString() : "Max", surveyQuestionOptionMinText = question.SelectToken("size\_min") !=null? question.SelectToken("size\_min").ToString() : "Min" });

survey.surveyQuestionList.Add(surveyQuestion);

break;

}

case "scaler":

{

var surveyQuestion = new Entities.SurveyQuestion { surveyQuestionTitle = question.SelectToken("title").ToString(), surveyQuestionType = 5, surveyQuestionIsCompulsory = question.SelectToken("title").ToString() == "1" };

int size = Int32.Parse(question.SelectToken("scaleAmount").ToString());

for (int i = 0; i < size; i++)

{

if (i == 0) surveyQuestion.surveyQuestionOptionList.Add(new Entities.SurveyQuestionOption { surveyQuestionOptionTitle = question.SelectToken("mintext")!=null?question.SelectToken("mintext").ToString():"Worst" });

else if (i == (size - 1)) surveyQuestion.surveyQuestionOptionList.Add(new Entities.SurveyQuestionOption { surveyQuestionOptionTitle = question.SelectToken("maxtext")!=null?question.SelectToken("maxtext").ToString():"Best" });

else surveyQuestion.surveyQuestionOptionList.Add(new Entities.SurveyQuestionOption { surveyQuestionOptionTitle = "" });

}

survey.surveyQuestionList.Add(surveyQuestion);

break;

}

case "text":

{

var surveyQuestion = new Entities.SurveyQuestion { surveyQuestionTitle = question.SelectToken("title").ToString(), surveyQuestionType = 6, surveyQuestionIsCompulsory = question.SelectToken("title").ToString() == "1" };

surveyQuestion.surveyQuestionOptionList.Add(new Entities.SurveyQuestionOption { surveyQuestionOptionTitle = "" });

survey.surveyQuestionList.Add(surveyQuestion);

break;

}

case "textarea":

{

var surveyQuestion = new Entities.SurveyQuestion { surveyQuestionTitle = question.SelectToken("title").ToString(), surveyQuestionType = 7, surveyQuestionIsCompulsory = question.SelectToken("title").ToString() == "1" };

surveyQuestion.surveyQuestionOptionList.Add(new Entities.SurveyQuestionOption { surveyQuestionOptionTitle = "" });

survey.surveyQuestionList.Add(surveyQuestion);

break;

}

case "signature":

{

break;

}

case "photo":

{

break;

}

case "section":

{

break;

}

case "img\_checkbox":

{

var surveyQuestion = new Entities.SurveyQuestion { surveyQuestionTitle = question.SelectToken("title").ToString(), surveyQuestionType = 1, surveyQuestionIsCompulsory = question.SelectToken("title").ToString() == "1" };

if (question.SelectToken("satisfactory\_vUnsatisfactory")!=null)

surveyQuestion.surveyQuestionOptionList.Add(new Entities.SurveyQuestionOption { surveyQuestionOptionTitle = question.SelectToken("satisfactory\_vUnsatisfactory").ToString(), surveyQuestionOptionTitleType = 2 });

if (question.SelectToken("satisfactory\_Unsatisfactory") != null)

surveyQuestion.surveyQuestionOptionList.Add(new Entities.SurveyQuestionOption { surveyQuestionOptionTitle = question.SelectToken("satisfactory\_Unsatisfactory").ToString(), surveyQuestionOptionTitleType = 2 });

if (question.SelectToken("satisfactory\_Neutral") != null)

surveyQuestion.surveyQuestionOptionList.Add(new Entities.SurveyQuestionOption { surveyQuestionOptionTitle = question.SelectToken("satisfactory\_Neutral").ToString(), surveyQuestionOptionTitleType = 2 });

if (question.SelectToken("satisfactory\_Satisfactory")!=null)

surveyQuestion.surveyQuestionOptionList.Add(new Entities.SurveyQuestionOption { surveyQuestionOptionTitle = question.SelectToken("satisfactory\_Satisfactory").ToString(), surveyQuestionOptionTitleType = 2 });

if (question.SelectToken("satisfactory\_vSatisfactory")!=null)

surveyQuestion.surveyQuestionOptionList.Add(new Entities.SurveyQuestionOption { surveyQuestionOptionTitle = question.SelectToken("satisfactory\_vSatisfactory").ToString(), surveyQuestionOptionTitleType = 2 });

survey.surveyQuestionList.Add(surveyQuestion);

break;

}

case "satisfactory":

{

var surveyQuestion = new Entities.SurveyQuestion { surveyQuestionTitle = question.SelectToken("title").ToString(), surveyQuestionType = 5, surveyQuestionIsCompulsory = question.SelectToken("title").ToString() == "1" };

if (question.SelectToken("satisfactory\_vUnsatisfactory") != null)

surveyQuestion.surveyQuestionOptionList.Add(new Entities.SurveyQuestionOption { surveyQuestionOptionTitle = question.SelectToken("satisfactory\_vUnsatisfactory").ToString(), surveyQuestionOptionTitleType = question.SelectToken("satisfactory\_vUnsatisfactory").ToString().Contains("://") ? 2 : 1 });

if (question.SelectToken("satisfactory\_Unsatisfactory") != null)

surveyQuestion.surveyQuestionOptionList.Add(new Entities.SurveyQuestionOption { surveyQuestionOptionTitle = question.SelectToken("satisfactory\_Unsatisfactory").ToString(), surveyQuestionOptionTitleType = question.SelectToken("satisfactory\_vUnsatisfactory").ToString().Contains("://") ? 2 : 1 });

if (question.SelectToken("satisfactory\_Neutral") != null)

surveyQuestion.surveyQuestionOptionList.Add(new Entities.SurveyQuestionOption { surveyQuestionOptionTitle = question.SelectToken("satisfactory\_Neutral").ToString(), surveyQuestionOptionTitleType = question.SelectToken("satisfactory\_vUnsatisfactory").ToString().Contains("://") ? 2 : 1 });

if (question.SelectToken("satisfactory\_Satisfactory") != null)

surveyQuestion.surveyQuestionOptionList.Add(new Entities.SurveyQuestionOption { surveyQuestionOptionTitle = question.SelectToken("satisfactory\_Satisfactory").ToString(), surveyQuestionOptionTitleType = question.SelectToken("satisfactory\_vUnsatisfactory").ToString().Contains("://") ? 2 : 1 });

if (question.SelectToken("satisfactory\_vSatisfactory") != null)

surveyQuestion.surveyQuestionOptionList.Add(new Entities.SurveyQuestionOption { surveyQuestionOptionTitle = question.SelectToken("satisfactory\_vSatisfactory").ToString(), surveyQuestionOptionTitleType = question.SelectToken("satisfactory\_vUnsatisfactory").ToString().Contains("://") ? 2 : 1 });

survey.surveyQuestionList.Add(surveyQuestion);

break;

}

case "img\_radio":

{

var surveyQuestion = new Entities.SurveyQuestion { surveyQuestionTitle = question.SelectToken("title").ToString(), surveyQuestionType = 0, surveyQuestionIsCompulsory = question.SelectToken("title").ToString() == "1" };

if (question.SelectToken("img\_radio\_vUnsatisfactory") != null)

surveyQuestion.surveyQuestionOptionList.Add(new Entities.SurveyQuestionOption { surveyQuestionOptionTitle = question.SelectToken("img\_radio\_vUnsatisfactory").ToString(), surveyQuestionOptionTitleType = 2 });

if (question.SelectToken("img\_radio\_Unsatisfactory") != null)

surveyQuestion.surveyQuestionOptionList.Add(new Entities.SurveyQuestionOption { surveyQuestionOptionTitle = question.SelectToken("img\_radio\_Unsatisfactory").ToString(), surveyQuestionOptionTitleType = 2 });

if (question.SelectToken("img\_radio\_Neutral") != null)

surveyQuestion.surveyQuestionOptionList.Add(new Entities.SurveyQuestionOption { surveyQuestionOptionTitle = question.SelectToken("img\_radio\_Neutral").ToString(), surveyQuestionOptionTitleType = 2 });

if (question.SelectToken("img\_radio\_Satisfactory") != null)

surveyQuestion.surveyQuestionOptionList.Add(new Entities.SurveyQuestionOption { surveyQuestionOptionTitle = question.SelectToken("img\_radio\_Satisfactory").ToString(), surveyQuestionOptionTitleType = 2 });

if (question.SelectToken("img\_radio\_vSatisfactory") != null)

surveyQuestion.surveyQuestionOptionList.Add(new Entities.SurveyQuestionOption { surveyQuestionOptionTitle = question.SelectToken("img\_radio\_vSatisfactory").ToString(), surveyQuestionOptionTitleType = 2 });

survey.surveyQuestionList.Add(surveyQuestion);

break;

}

case "select":

{

var surveyQuestion = new Entities.SurveyQuestion { surveyQuestionTitle = question.SelectToken("title").ToString(), surveyQuestionType = 10, surveyQuestionIsCompulsory = question.SelectToken("title").ToString() == "1" };

foreach (var option in question.SelectToken("options").ToList())

{

surveyQuestion.surveyQuestionOptionList.Add(new Entities.SurveyQuestionOption { surveyQuestionOptionTitle = option.SelectToken("option").ToString() });

}

survey.surveyQuestionList.Add(surveyQuestion);

break;

}

}

}

//var member = Member.GetById(session,1);

var member = Member.GetByHash(session, memberHash);

member.AddSurvey(survey);

session.SaveOrUpdate(member);

transaction.Commit();

HttpContext.Current.Response.Redirect("./../../viewsurveylist.htm");

Assert.AreEqual(12, survey.surveyQuestionList.Count);

Assert.AreEqual(6, survey.surveyQuestionList.Get(0).surveyQuestionType);

Assert.AreEqual(3, survey.surveyQuestionList.Get(1).surveyQuestionType);

Assert.AreEqual(7, survey.surveyQuestionList.Get(2).surveyQuestionType);

Assert.AreEqual(1, survey.surveyQuestionList.Get(3).surveyQuestionType);

Assert.AreEqual(10, survey.surveyQuestionList.Get(4).surveyQuestionType);

Assert.AreEqual(2, survey.surveyQuestionList.Get(5).surveyQuestionType);

Assert.AreEqual(4, survey.surveyQuestionList.Get(6).surveyQuestionType);

Assert.AreEqual(0, survey.surveyQuestionList.Get(7).surveyQuestionType);

Assert.AreEqual(5, survey.surveyQuestionList.Get(8).surveyQuestionType);

Assert.AreEqual(5, survey.surveyQuestionList.Get(9).surveyQuestionType);

Assert.AreEqual(1, survey.surveyQuestionList.Get(10).surveyQuestionType);

Assert.AreEqual(5, survey.surveyQuestionList.Get(11).surveyQuestionType);

}

}

}

}

[TestFixture]

public class ResultsTest

{

[Test]

public void getSurvey()

{

using (var session = FluentNHibernateConfiguration.InitFactory.sessionFactory.OpenSession())

{

using (var transaction = session.BeginTransaction())

{

JavaScriptSerializer js = new JavaScriptSerializer();

var member = Member.GetByHash(session,memberHash);

if (member == null)

return;

var survey = member.memberSurveyList.ToList<Survey>().Find(item => item.Id == id);

//var survey = Entities.Survey.GetById(session, id);

var result = new SurveyResults();

result.surveyTitle = survey.surveyTitle;

result.surveyDescription = survey.surveyDescription;

result.noOfRespondents = survey.respondentList.Count;

int i = 0;

int totalTime = 0;

foreach (var respondent in survey.respondentList)

{

totalTime += respondent.respondentTime;

//add country code to list sequential search

bool add = true;

var countryCode = new KeyValueResponse { key = respondent.respondentCountryCode, value=1 };

foreach (var tempCode in result.countryCode)

{

if(tempCode.key.Equals(countryCode.key))

{

tempCode.value++;

add = false;

}

}

if (add) result.countryCode.Add(countryCode);

//add browser type

add = true;

var browserBrowser = new KeyValueResponse { key = respondent.respondentBrowser, value = 1 };

foreach (var tempBrowser in result.browserBrowser)

{

if (tempBrowser.key.Equals(browserBrowser.key))

{

tempBrowser.value++;

add = false;

}

}

if (add) result.browserBrowser.Add(browserBrowser);

//add operating system

add = true;

var browserOS = new KeyValueResponse { key = respondent.respondentOS, value = 1 };

foreach (var tempOS in result.browserOS)

{

if (tempOS.key.Equals(browserOS.key))

{

tempOS.value++;

add = false;

}

}

if (add) result.browserOS.Add(browserOS);

}

if (result.noOfRespondents > 0)

result.avgTime = (totalTime / result.noOfRespondents) / 1000;

else result.avgTime = 0;

foreach (var question in survey.surveyQuestionList)

{

var resultQn = new Questions();

i++;

resultQn.questionTitle = question.surveyQuestionTitle;

resultQn.questionType = question.surveyQuestionType;

switch (question.surveyQuestionType)

{

//multiple choices and checkboxes

case 0:

case 1:

case 5:

case 10:

int count = 0;

foreach (var option in question.surveyQuestionOptionList)

{

var questionOption = new Options();

questionOption.optionTitle = option.surveyQuestionOptionTitle;

foreach (var response in question.surveyQuestionResponseList)

{

if (response.responseIntegerValue == count)

{

questionOption.noOfRespondents++;

}

}

count++;

resultQn.optionList.Add(questionOption);

}

result.questionList.Add(resultQn);

break;

case 2: //slider

case 3: //numerical input

{

foreach (var response in question.surveyQuestionResponseList)

{

var questionOption = new Options { responseStr = response.responseIntegerValue.ToString() , noOfRespondents=1};

bool add = true;

foreach(var tempOption in resultQn.optionList)

{

if (tempOption.responseStr.Equals(questionOption.responseStr))

{

tempOption.noOfRespondents++;

add = false;

}

}

if(add)

resultQn.optionList.Add(questionOption);

}

result.questionList.Add(resultQn);

break;

}

case 4: //date

case 6: //text

case 7: //textarea

foreach (var response in question.surveyQuestionResponseList)

{

var questionOption = new Options {responseStr = response.responseStringValue, noOfRespondents=1};

bool add = true;

foreach(var tempOption in resultQn.optionList)

{

if (tempOption.responseStr.Equals(questionOption.responseStr))

{

tempOption.noOfRespondents++;

add = false;

}

}

if(add)

resultQn.optionList.Add(questionOption);

}

result.questionList.Add(resultQn);

break;

}

}

Assert.AreEqual("Test Survey", result.surveyTitle);

Assert.AreEqual(2, result.noOfRespondents);

Assert.AreEqual(2, result.browserBrowser.Count);

Assert.AreEqual(2, result.browserOS.Count);

Assert.AreEqual(2, result.countryCode.Count);

Assert.AreEqual("Test Survey First Question", result.questionList.Get(0).questionTitle);

Assert.AreEqual("Test Survey Second Question", result.questionList.Get(1).questionTitle);

Assert.AreEqual("Test Survey Third Question", result.questionList.Get(2).questionTitle);

Assert.AreEqual("Test Survey Fourth Question", result.questionList.Get(3).questionTitle);

//print to webservice

Context.Response.Write(js.Serialize(result));

}

[Test]

public void createCSV()

{

System.Web.HttpResponse csvresponse = System.Web.HttpContext.Current.Response;

csvresponse.Clear();

csvresponse.AddHeader("content-disposition", "attachment; filename=surveyResults.csv");

csvresponse.ContentType = "text/csv";

csvresponse.Write("Question No: ,Question: ,Option: ,Responses: ,No of Responses: ,");

csvresponse.Write(Environment.NewLine);

using (var session = FluentNHibernateConfiguration.InitFactory.sessionFactory.OpenSession())

{

using (var transaction = session.BeginTransaction())

{

var result = new SurveyResults();

var member = Member.GetByHash(session, memberHash);

if (member == null)

return;

var survey = member.memberSurveyList.ToList<Survey>().Find(item => item.Id == id);

int i = 0;

foreach (var question in survey.surveyQuestionList)

{

var resultQn = new Questions();

resultQn.questionTitle = question.surveyQuestionTitle;

resultQn.questionType = question.surveyQuestionType;

switch (question.surveyQuestionType)

{

case 0:

case 1:

case 5:

case 10:

{

int optionNo = 0;

foreach (var option in question.surveyQuestionOptionList)

{

string fileRow = "";

string cell = "";

cell += "Question " + (i + 1).ToString() + ",";

if (question.surveyQuestionTitle.Contains(","))

{

cell += "\"" + question.surveyQuestionTitle + "\"" + ",";

}

else

{

cell += question.surveyQuestionTitle + ",";

}

if (option.surveyQuestionOptionTitle.Contains(","))

{

cell += "\"" + option.surveyQuestionOptionTitle + "\"" + ",";

}

else

{

cell += option.surveyQuestionOptionTitle + ",";

}

cell += "N.A" + ",";

int choices = 0;

foreach (var response in question.surveyQuestionResponseList)

{

if (response.responseIntegerValue == optionNo)

{

choices++;

}

}

optionNo++;

cell += choices + ",";

fileRow += cell + ",";

csvresponse.Write(fileRow);

csvresponse.Write(Environment.NewLine);

}

i++;

break;

}

case 2:

case 3:

{

foreach (var response in question.surveyQuestionResponseList)

{

var questionOption = new Options { responseStr = response.responseIntegerValue.ToString() , noOfRespondents=1};

bool add = true;

foreach(var tempOption in resultQn.optionList)

{

if (questionOption.responseStr == null)

{

tempOption.noOfRespondents++;

add = false;

questionOption.noOfRespondents = tempOption.noOfRespondents;

}

else if (tempOption.responseStr.Equals(questionOption.responseStr))

{

tempOption.noOfRespondents++;

add = false;

questionOption.noOfRespondents = tempOption.noOfRespondents;

}

}

if (add)

{

questionOption.optionTitle = response.responseIntegerValue.ToString();

resultQn.optionList.Add(questionOption);

}

}

foreach (var qnOption in resultQn.optionList)

{

string fileRow = "";

string cell = "";

cell += "Question " + (i + 1).ToString() + ",";

if (question.surveyQuestionTitle.Contains(","))

{

cell += "\"" + question.surveyQuestionTitle + "\"" + ",";

}

else

{

cell += question.surveyQuestionTitle + ",";

}

cell += "N.A" + ",";

if (qnOption.optionTitle == null)

{

cell += qnOption.noOfRespondents.ToString() + " Responpondents has not responded." + ",";

cell += qnOption.noOfRespondents.ToString() + ",";

fileRow += cell + ",";

csvresponse.Write(fileRow);

csvresponse.Write(Environment.NewLine);

}

else

{

cell += qnOption.optionTitle + ",";

cell += qnOption.noOfRespondents.ToString() + ",";

fileRow += cell + ",";

csvresponse.Write(fileRow);

csvresponse.Write(Environment.NewLine);

}

}

i++;

break;

}

case 4:

case 6:

case 7:

{

foreach (var response in question.surveyQuestionResponseList)

{

var questionOption = new Options { responseStr = response.responseStringValue, noOfRespondents = 1 };

bool add = true;

foreach (var tempOption in resultQn.optionList)

{

if (tempOption.responseStr.Equals(questionOption.responseStr))

{

tempOption.noOfRespondents++;

add = false;

questionOption.noOfRespondents = tempOption.noOfRespondents;

}

}

if (add)

{

questionOption.optionTitle = response.responseStringValue;

resultQn.optionList.Add(questionOption);

}

}

foreach (var qnOption in resultQn.optionList)

{

string fileRow = "";

string cell = "";

cell += "Question " + (i + 1).ToString() + ",";

if (question.surveyQuestionTitle.Contains(","))

{

cell += "\"" + question.surveyQuestionTitle + "\"" + ",";

}

else

{

cell += question.surveyQuestionTitle + ",";

}

cell += "N.A" + ",";

if (qnOption.optionTitle.Contains(","))

{

cell += "\"" + qnOption.optionTitle + "\"" + ",";

}

else

{

cell += qnOption.optionTitle + ",";

}

cell += qnOption.noOfRespondents.ToString() + ",";

fileRow += cell + ",";

csvresponse.Write(fileRow);

csvresponse.Write(Environment.NewLine);

}

i++;

break;

}

}

}

}

}

csvresponse.End();

}

}

[TestFixture]

public class MemberTest

{

/\*

jsonString { "firstName": "first", "lastName": "last", "userName": "username01", "email": "email@domain.com", "password": "password", "cpassword": "password" }

\*/

[Test]

public void AddUser()

{

using (var session = FluentNHibernateConfiguration.InitFactory.sessionFactory.OpenSession())

{

using (var transaction = session.BeginTransaction())

{

JObject jsonObject = JObject.Parse(jsonString);

var user = new Entities.Member

{

memberFirstName = jsonObject.SelectToken("firstName").ToString(),

memberLastName = jsonObject.SelectToken("lastName").ToString(),

userName = jsonObject.SelectToken("userName").ToString(),

memberPassword = jsonObject.SelectToken("password").ToString(),

memberEmail = jsonObject.SelectToken("email").ToString()

};

JavaScriptSerializer js = new JavaScriptSerializer();

DateTime dt = new DateTime();

user.memberHash = util.UtilityMethods.CalculateMD5Hash(user.userName + dt.ToShortTimeString());

Context.Response.Write(js.Serialize(new Response3 { LogIn = 1, twocubeSSO = user.memberHash }));

session.Save(user);

transaction.Commit();

Assert.AreEqual("first", user.memberFirstName);

Assert.AreEqual(“last”, user.memberLastName);

Assert.AreEqual(“username01”, user.userName);

Assert.AreEqual("email@domain.com", user.memberEmail);

Assert.AreEqual("password", user.memberPassword);

}

}

}

[Test]

public void listOfUsers()

{

using (var session = FluentNHibernateConfiguration.InitFactory.sessionFactory.OpenSession())

{

using (var transaction = session.BeginTransaction())

{

JavaScriptSerializer js = new JavaScriptSerializer();

JObject jsonObject = JObject.Parse(jsonString);

var member = Member.GetByUserName(session, “username1”);

Assert.AreEqual("username1", member.userName);

var member = Member.GetByUserName(session, “usernamedoesnotexist”);

Assert.AreEqual(null, member.userName);

if (member == null)

{

Context.Response.Write(js.Serialize(new Response {userExists = 0}));

}

else {

Context.Response.Write(js.Serialize(new Response { userExists = 1 }));

}

}

}

}

[Test]

public void checkEmailExist()

{

using (var session = FluentNHibernateConfiguration.InitFactory.sessionFactory.OpenSession())

{

using (var transaction = session.BeginTransaction())

{

JavaScriptSerializer js = new JavaScriptSerializer();

JObject jsonObject = JObject.Parse(jsonString);

var member = Member.GetByEmail(session, “user@domain.com”);

Assert.AreEqual("user@domain.com", member.memberEmail);

var member = Member.GetByUserName(session, “emaildoesnotexist”);

Assert.AreEqual(null, member.memberEmail);

if (member == null)

{

Context.Response.Write(js.Serialize(new Response1 { emailExists = 0 }));

}

else

{

Context.Response.Write(js.Serialize(new Response1 { emailExists = 1 }));

}

}

}

}

[Test]

public void ViewUserDetails()

{

using (var session = FluentNHibernateConfiguration.InitFactory.sessionFactory.OpenSession())

{

using (var transaction = session.BeginTransaction())

{

JObject jsonObject = JObject.Parse(jsonString);

JavaScriptSerializer js = new JavaScriptSerializer();

Context.Response.Write(js.Serialize(Entities.Member.GetByHash(session, “FF4AA02FCD977DF9B1B3F54D9AEDAFB8”)));

Assert.AreEqual("user@domain.com", member.memberEmail);

}

}

}

[Test]

public void UpdateUserDetails()

{

using (var session = FluentNHibernateConfiguration.InitFactory.sessionFactory.OpenSession())

{

using (var transaction = session.BeginTransaction())

{

JObject jsonObject = JObject.Parse(jsonString);

JavaScriptSerializer js = new JavaScriptSerializer();

var user = Entities.Member.GetByLogin(session, “username1”, “password”);

user.memberFirstName = “changedtofirst”;

user.memberLastName = “changedtolast”;

user.memberEmail = “changed@two.com”;

session.SaveOrUpdate(user);

transaction.Commit();

user = Entities.Member.GetByLogin(session, “username1”, “password”);

Assert.AreEqual("changedtofirst", user.memberFirstName);

Assert.AreEqual("changedtolast",user.memberLastName);

Assert.AreEqual("changed@two.com”, user.memberEmail);

}

}

}

[Test]

public void UpdateUserPassword()

{

using (var session = FluentNHibernateConfiguration.InitFactory.sessionFactory.OpenSession())

{

using (var transaction = session.BeginTransaction())

{

JObject jsonObject = JObject.Parse(jsonString);

JavaScriptSerializer js = new JavaScriptSerializer();

var user = Entities.Member.GetByLogin(session, jsonObject.SelectToken("username").ToString(), jsonObject.SelectToken("password").ToString());

user.memberPassword = “changedpassword”;

session.SaveOrUpdate(user);

transaction.Commit();

user = Entities.Member.GetByLogin(session, “username1”, “password”);

Assert.AreEqual("changedpassword", user.memberPassword);

}

}

}

[Test]

public void validatecpw()

{

using (var session = FluentNHibernateConfiguration.InitFactory.sessionFactory.OpenSession())

{

using (var transaction = session.BeginTransaction())

{

JavaScriptSerializer js = new JavaScriptSerializer();

JObject jsonObject = JObject.Parse(jsonString);

var member = Entities.Member.GetByLogin(session, “username1”, “password”);

Assert.AreEqual("user@domain.com", member.memberEmail);

var member = Entities.Member.GetByLogin(session, “username1”, “password1”);

Assert.AreEqual(null, member.memberEmail);

if (member == null)

{

Context.Response.Write(js.Serialize(new pwResponse { pwExists = 0 }));

}

else

{

Context.Response.Write(js.Serialize(new pwResponse { pwExists = 1 }));

}

}

}

}

[Test]

public void Login()

{

using (var session = FluentNHibernateConfiguration.InitFactory.sessionFactory.OpenSession())

{

using (var transaction = session.BeginTransaction())

{

JObject jsonObject = JObject.Parse(jsonString);

JavaScriptSerializer js = new JavaScriptSerializer();

var member = Entities.Member.GetByLogin(session, “username1”, “password”);

Assert.AreEqual("user@domain.com", member.memberEmail);

var member = Entities.Member.GetByLogin(session, “username1”, “password1”);

Assert.AreEqual(null, member.memberEmail);

if (member == null)

{

Context.Response.Write(js.Serialize(new Response3 { LogIn = 0 }));

}

else

{

DateTime dt = new DateTime();

member.memberHash = util.UtilityMethods.CalculateMD5Hash(member.userName+dt.ToShortTimeString());

Context.Response.Write(js.Serialize(new Response3 { LogIn = 1, twocubeSSO = member.memberHash }));

session.SaveOrUpdate(member);

transaction.Commit();

}

}

}

}

[Test]

public void FBLogin()

{

using (var session = FluentNHibernateConfiguration.InitFactory.sessionFactory.OpenSession())

{

using (var transaction = session.BeginTransaction())

{

//JObject jsonObject = JObject.Parse(jsonString);

JavaScriptSerializer js = new JavaScriptSerializer();

var member = Member.GetByFBID(session, FBID);

var member = Entities.Member.GetByFBID(session, “1584353”);

Assert.AreEqual("user@domain.com", member.memberEmail);

var member = Entities.Member.GetByFBID(session, “word15843593”);

Assert.AreEqual(null, member.memberEmail);

if (member == null)

{

var user = new Entities.Member

{

memberFirstName = firstName,

memberLastName = lastName,

userName = FBID,

memberPassword = FBID,

memberEmail = email,

memberFBID = FBID

};

user.memberHash = util.UtilityMethods.CalculateMD5Hash(user.userName + DateTime.Now.ToShortTimeString());

Context.Response.Write(js.Serialize(new Response3 { LogIn = 1, twocubeSSO = user.memberHash }));

session.Save(user);

transaction.Commit();

//Context.Response.Write(js.Serialize(new Response3 { LogIn = 0 }));

}

else

{

//DateTime dt = new DateTime();

member.memberHash = util.UtilityMethods.CalculateMD5Hash(member.userName + DateTime.Now.ToShortTimeString());

Context.Response.Write(js.Serialize(new Response3 { LogIn = 1, twocubeSSO = member.memberHash }));

session.SaveOrUpdate(member);

transaction.Commit();

}

}

}

}

}

}

1. Discussion
2. Explain how you derived your analytical design and models
3. Explain how you performed your UI design
4. Describe difficulties encountered and solutions applied
5. WBS