

**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 12

2.7 Glossary 28

2.8 References 28

2.9 Revision History 28

3. Use Case/Activity Diagrams 29

4. Analytical Model – Class Diagrams 30

5. Design Model – Sequence Diagrams 31

5.1 Account Management 31

5.1.1 Create member account 31

5.1.2 Login 32

5.1.3 View member account 33

5.1.4 Update member account 33

5.1.5 Update password 34

5.2 Survey Management 35

5.2.1 Create Survey 35

5.2.2 Create Survey Questions 35

5.2.3 Create Survey Option 36

5.2.4 View Survey 37

5.3 Survey Report Management 38

5.3.1 Generate Survey Report 38

5.3.2 View Survey Report 39

5.3.3 Export Survey Report 39

5.4 Respondents Actions 40

5.4.1 Submit Survey 40

5.5 Ground Data Logging 41

5.5.1 Data Logging 41

6. Testing 42

6.1 Unit testing 42

6.2 Functional Testing 63

7.1 Discussion 80

8.1 Explain how you derived your analytical design and models 80

9.1 Explain how you performed your UI design 80

10.1 Describe difficulties encountered and solutions applied 80

* **Project Team Information**

|  |  |  |  |
| --- | --- | --- | --- |
| **Team Name:** | twoCube™ | | |
| **Project Title:** | NTUSurvey | | |
| **Team Website URL:** | [http://twocube1.elasticbeanstalk.com](http://twocube1.elasticbeanstalk.com/) | | |
| **Project Description:** | [http://twocube1.elasticbeanstalk.com](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 |

* **Refined Software Requirement Specification**
* Product Description
* 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
* 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.

* 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.

* 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.

* 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.
* Constraints

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

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

**Overview**

**For Member:**

* Create Member account
* Login
* Update Member account
* Update password
* Create survey
* Create survey questions
* Edit, delete and reorder survey questions
* Create survey options
* View survey
* View survey statistics
* 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 or alternatively use Facebook account to login

For system login, username and password will be compared with the username and password in database. System save username session

For Facebook login, system will compare Facebook ID with the current Facebook ID exist in the database. If exist, it will load all account data in the database. If it is not, system will get several data to be input into the database such as Facebook ID, first name, last name and email

* **Update Member account**

System display user’s information

Users can change other details

System will save changes by updating the database

* **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

Allow users to edit, delete and reorder 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.

Allows users to use images as question options

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 set minimum and maximum value for slider 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.

* 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 |  |
| memberPassword | varchar(255) | Member’s login password | Min: 6 chars |
| memberAge | Int4(2) | Member’s age |  |
| dateOfBirthday | timestamp | Member’s birth date |  |
| memberLocation | varchar(255) | Member’s location |  |
| memberEmail | varchar(255 | Member’s email |  |
| memberFBID | Varchar(255) | Member’s Facebook ID |  |
| 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 |  |

* Non-functional requirements
* Compatibility

The website must be compatible with all HTML5 compliant browsers.

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

* 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.

* Security

Access shall be controlled with usernames and passwords

Access to the database will be done via web services.

* 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.

* 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.

* 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.

* System Maintenance

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

* Interface Requirements
* *User Interfaces*

User interfaces describes how users interact and engage with a specific page on a website. The goal of user interface is to make user’s interaction as simple and interactive as possible.

Overview of the screen must be clear. For instance, break the screen up into sections such that at the top section it will display the logo and navigation links, on the middle of the page it will be the contents of what the users are expected to see and lastly at the bottom of the page it will be footer.

User must also be able to navigate from one page to another page without any problem or broken links. Also, the links should be obvious to where the users would be directed to (it should not direct user to login page when user clicks on registration page). Error message should also be displayed whenever there is an error faced by the users and provide solutions on how the users can solve the problem or error.

Requirement of user interface is that users need familiar with the handling basic UI components such as radio buttons, dropdown list, and checkboxes.

It provides a user-friendly graphic interface for any users who come to twoCubeTM website, performing tasks such as:

* Registering for an account

Registration should be able to create new users in the database successfully with user name, password and email address

* Login

This should require a username and password to successfully login to the website

* Create survey

This should allow the users to create survey without any problem

* Viewing survey results

This should enables user to generate analytic reports on the survey created

In addition, the overview of the website must be consistent in all aspects. For example, the logo should be at the top left hand corners on all pages. The colour and size of the font must also be consistent such that it does not confuse and frustrate the users (i.e **bold** only when the content is important) to reduce ambiguity.

* *Hardware interfaces*

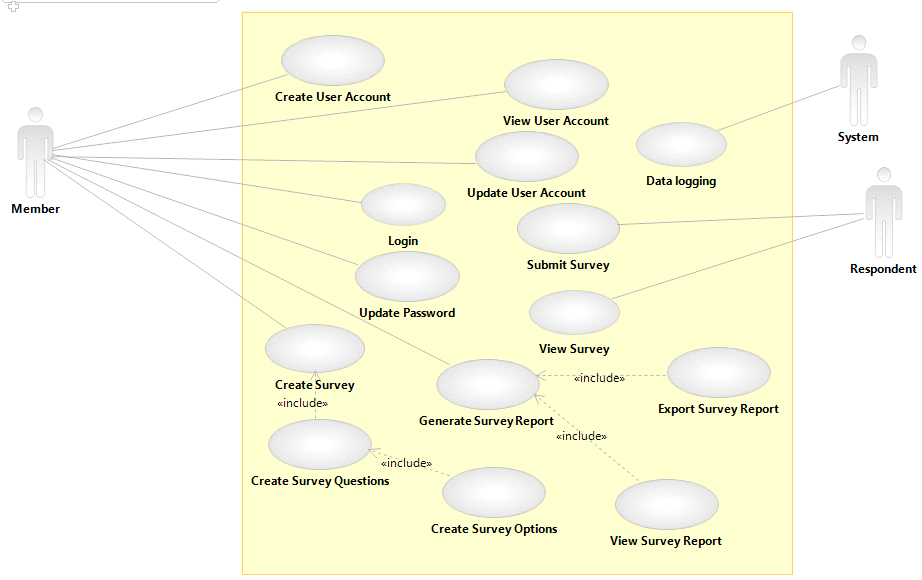
Hardware devices such as PCs, notebooks, tablets, mobile phone devices are supported by the website.

It also requires Android 2.2 above for the mobile application.

* *Software interfaces*

twoCubeTM website enables users to interact with the server and get access to the Internet.

Some of the software requirements would be:

* Html5 browser
* Datebase: postgresql
* Application: c# webservices
* Web Server: IIS7 running on a scalable cloud
* Flash
* Use Case Model
* Use Case Diagram
* 
* 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: | * Upon success, the Member entered data is stored in the Member’s account. Confirmation is sent to the email address. * If the Member fails enters invalid data or cancels the account creation request, no account will be created. |
| Normal Flow: | * The Member enters the required Member account information and request the system to save the entered Member account information. * The system validates the entered Member account information. * The entered information is then stored in the Member’s account. * 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: | * Member must exist. * Member must have a valid Member id and password |
| Postconditions: | * Member logged on to the system successfully. |
| Normal Flow: | * Member enters the Member id and password when prompt. * System verifies Member id and password. * System displays Member home page. |
| Alternative Flows: | * Member logged in through “Facebook authenticator” * System verify if any Facebook account is logged in * Facebook check for permission * Facebook check for authorization code * 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: | 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: | * Upon success, the Member entered data is updated in the Member’s account. * 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: | * The Member update the required Member account information and request the system to save the updated Member account information. * The system validates the updated Member account information. * The updated information is then stored in the Member’s account. * The system notifies the Member that the account has been updated. |
| Alternative Flows: |  |
| Exceptions: | Member cancellation  UC03.0.E.1 Member case ends and account is not updated.  Invalid information  UC03.0.E.2 System will prompt Member to re-enter the information.  UC03.0.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: | UC04 | | |
| 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: | * Upon success, the password is updated in the Member’s account. * If the Member enter mismatched password or cancels the update password request, there will be no change to the Member’s account. |
| Normal Flow: | * 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. * The system validates if the old password matches with the initial password. * The updated information is then stored in the Member’s account. * The system notifies the Member that the password has been changed. |
| Alternative Flows: |  |
| Exceptions: | Member cancellation  UC04.0.E.1 Member case ends and password 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:   * Mismatched password |
| Includes: |  |
| Priority: |  |
| Frequency of Use: |  |
| Business Rules: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | UC05 | | |
| 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: | * Member must be logged on to the system. |
| Postconditions: | * Survey created and posted up for respondents to start their survey. |
| Normal Flow: | * Member initiates the “Create Survey” button. * System displays the form for Member to enter survey title and description. * Member enters the corresponding fields as needed. * System validates the fields Member has entered. * System proceeds to UC06 Create Survey Questions to proceed on to create questions for the survey. |
| Alternative Flows: |  |
| Exceptions: | Member exits page before saving.  UC05.0.E.3 Use case ends and survey not created.  Required fields not entered.  UC05.0.E.4 System prompts Member to enter the required fields.  UC05.0.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: | UC06 | | |
| 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: | * Survey question created. |
| Normal Flow: | * Member initiates the create survey question button. * System redirects Member to the create survey question page. * Member enters the question. * System validates question entered. * System proceeds to UC07 Create Survey Options to create options for the question. |
| Alternative Flows: |  |
| Exceptions: | Required fields not entered.  UC06.0.E.1 System prompts Member to enter required fields before proceeding. |
| Includes: | UC07 Create Survey Options |
| Priority: |  |
| Frequency of Use: |  |
| Business Rules: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | UC07 | | |
| 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: | * Survey and survey question is created. |
| Postconditions: | * Option is added after the question. |
| Normal Flow: | * Member has to create survey. System proceeds to UC06. * Member has to add a question. System proceeds to UC07. * System show several types of options and wait for Member selection. * Member fills in the option description for the respondent to choose. * Option is added. * 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.  UC07.0.E.1 System prompts the Member to add at least 1 option of any type. |
| Includes: |  |
| Priority: | UC05 Create Survey, UC06 Create Survey Question. |
| Frequency of Use: |  |
| Business Rules: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | UC08 | | |
| 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: | Respondent |
| Description: | This use case allows users to view the survey they have created. |
| Trigger: | Member initiates the view survey feature. |
| Preconditions: | * Member must be logged on to the system. * An existing survey created by Member must exist. |
| Postconditions: | * Users are able to view the survey that they have created. |
| Normal Flow: | * Member initiates the “View Survey” button. * System displays details of the survey questions. |
| Alternative Flows: |  |
| Exceptions: | Users enter invalid Member id and password.  UC08.0.E.1 System prompts Member to reenter id and password.  There is no existing survey.  UC08.0.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: | UC9 | | |
| 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: | * Survey must exist. |
| Postconditions: | * Report generated successfully. |
| Normal Flow: | * Member initiates the “Generate Report” button. * System generates the statistical report of the survey questions. * Member views the generated report and initiates the “Save Report” button. * System saved the generated report. |
| Alternative Flows: |  |
| Exceptions: | Member exits page without initiating the “Save Report” button.  UC9.0.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: | UC10 | | |
| 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: | * Report has to be generated. |
| Postconditions: | * Users are able to view the survey report that they have generated. |
| Normal Flow: | * Member initiates the “View Survey Report” button. * System displays details of the survey report – for example charts. |
| Alternative Flows: |  |
| Exceptions: |  |
| Includes: | UC9 Generate Survey Report |
| Priority: |  |
| Frequency of Use: |  |
| Business Rules: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | UC11 | | |
| 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: | * The Member has to be logged in. * The Member has to have a survey created before. Status has to be “closed” when exporting. |
| Postconditions: | * The survey result is generated in a specified format. |
| Normal Flow: | * The Member will choose which file format the survey will be exported as. * The Member will choose what type of data format the survey will show. * Member will also filter what kind of information (specific questions or answers) will be exported. * The system will generate the specified report in the chosen format. * The system will prompt the Member to save the exported survey report. |
| Alternative Flows: |  |
| Exceptions: | Member cancellation  UC11.0.E.1 Member case ends and account is not updated.  Member exits page before closing  UC11.0.E.2 The process will be cancelled, thus the report will not be generated. |
| Includes: | UC9 Generate Survey Report |
| Priority: |  |
| Frequency of Use: |  |
| Business Rules: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | UC12 | | |
| 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: | * Respondent must have completed all required-to-do questions. |
| Postconditions: | * Survey submission completed. |
| Normal Flow: | * Respondent initiate the submit button. * System verifies respondent has completed all the required questions. * System redirect respondents to thank you page. |
| Alternative Flows: |  |
| Exceptions: | Respondents did not answer all required-to-do questions.  UC12.0.E.2 System prompts respondents to complete all required-to-do questions before submission.  UC12.0.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: | UC13 | | |
| 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: | * One respondent must only do survey once. |
| Postconditions: | * Respondent’s IP address logged. * Time taken for respondents to complete the survey is logged. |
| Normal Flow: | * Respondents initiate to do the survey. * System validate if IP address has already been logged. * Respondents start completing the survey. * System times the speed of respondents completing the survey. |
| Alternative Flows: |  |
| Exceptions: | IP address already logged.  UC13.0.E.2 System prompts respondents that they have already did the survey once.  UC13.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: |  |

* Glossary

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

* 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.

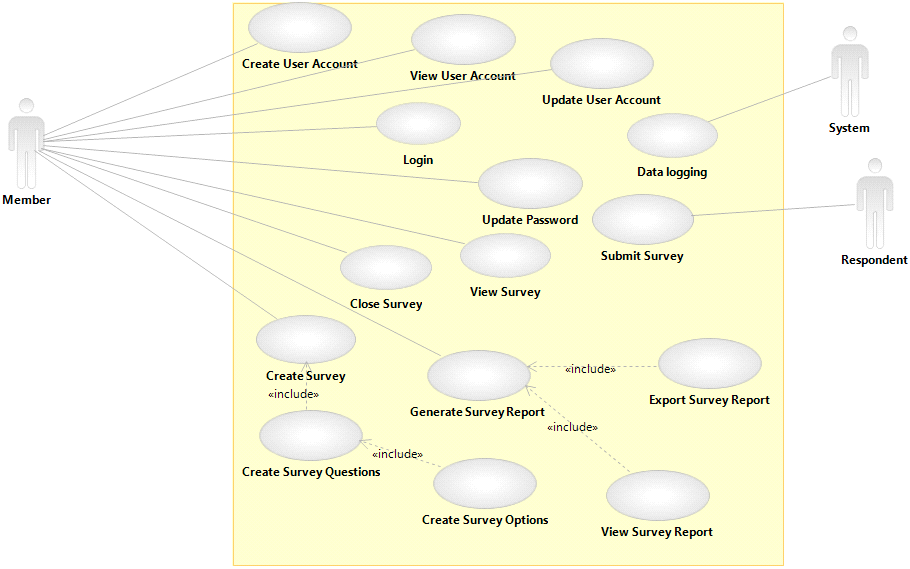
|  |  |  |  |
| --- | --- | --- | --- |
| **Document No.** | **Document Title** | **Date** | **Author** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

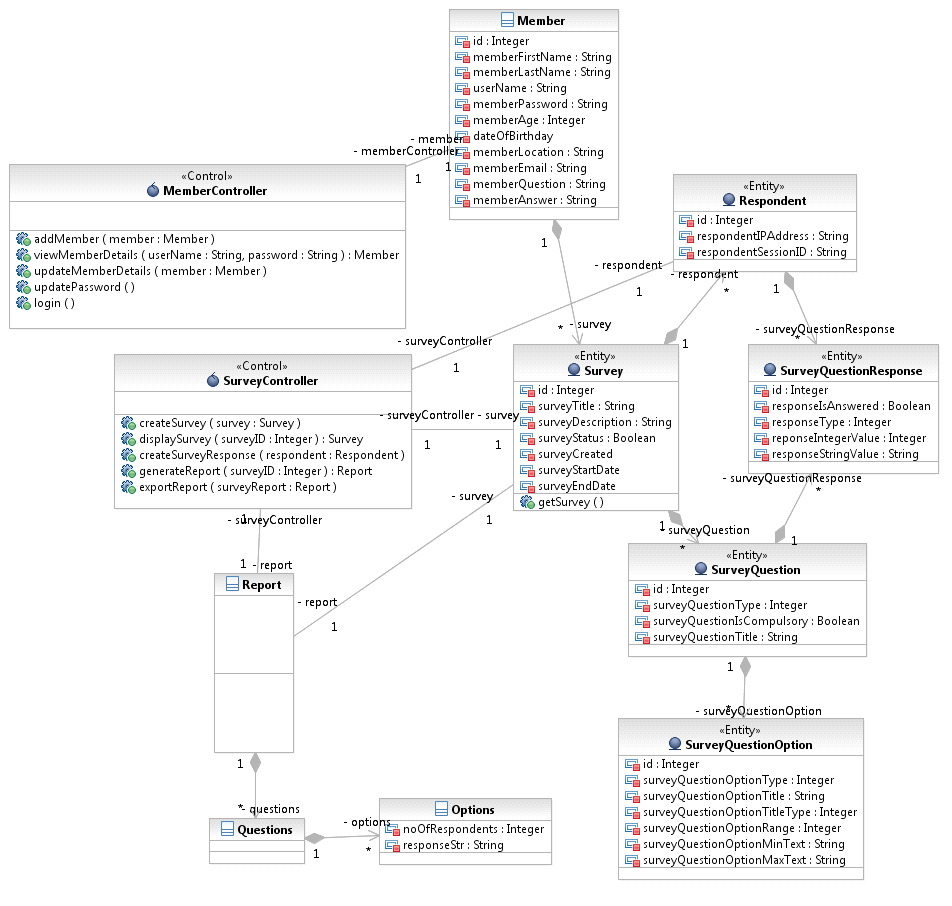
* Revision History

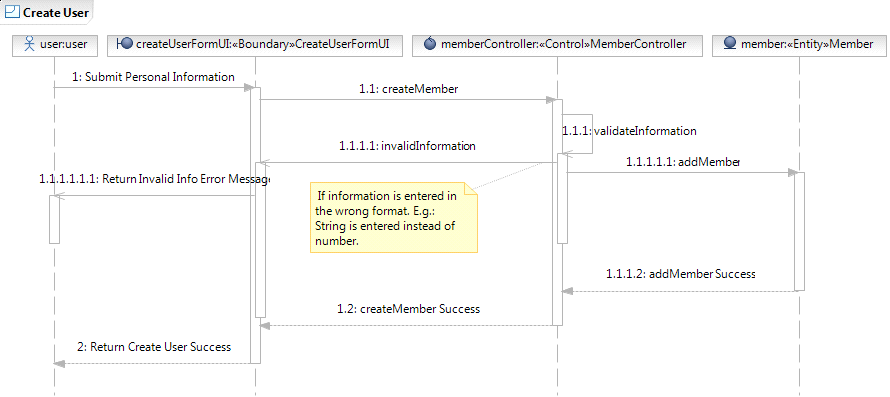
Identify changes to the SRS.

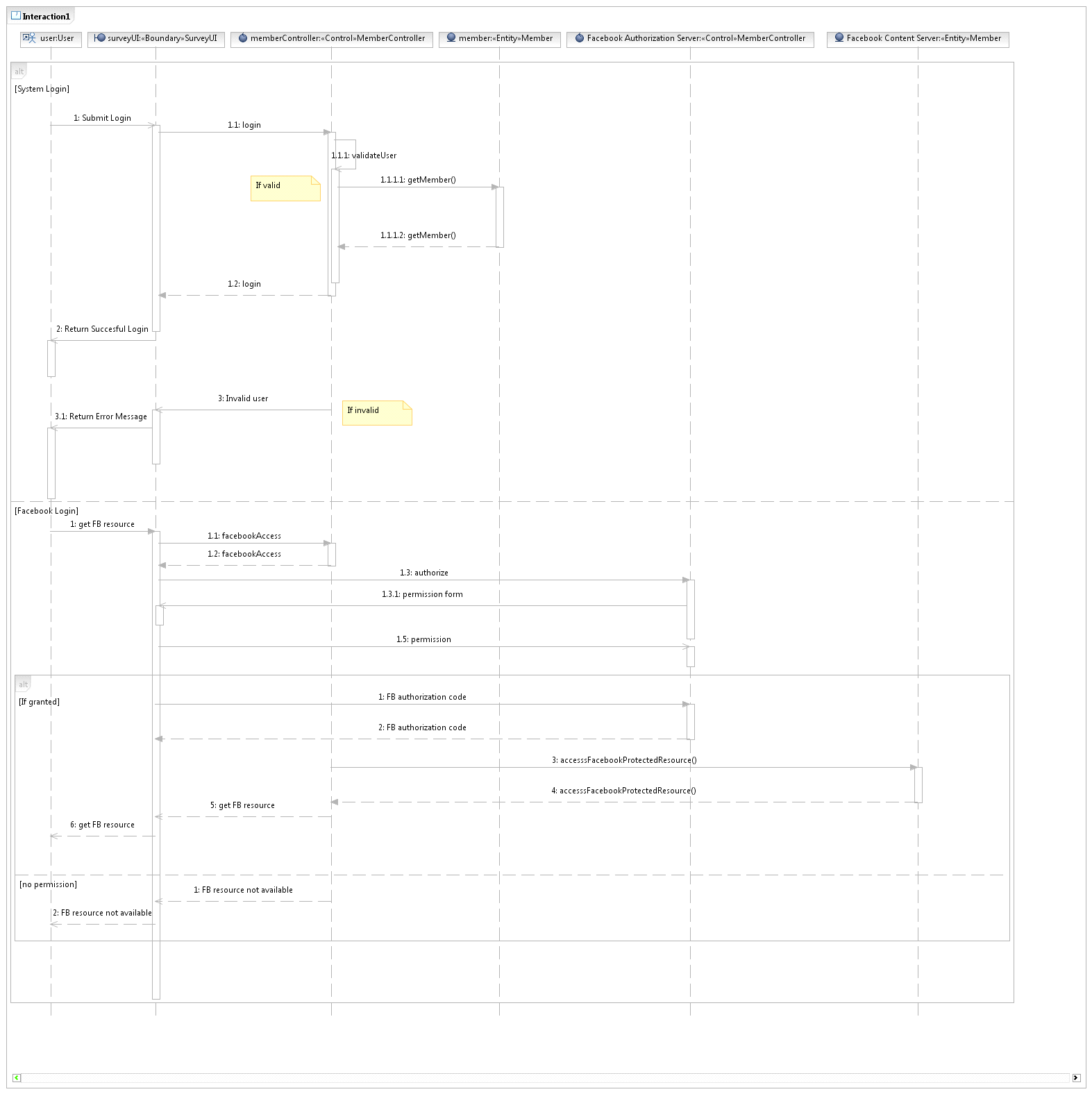
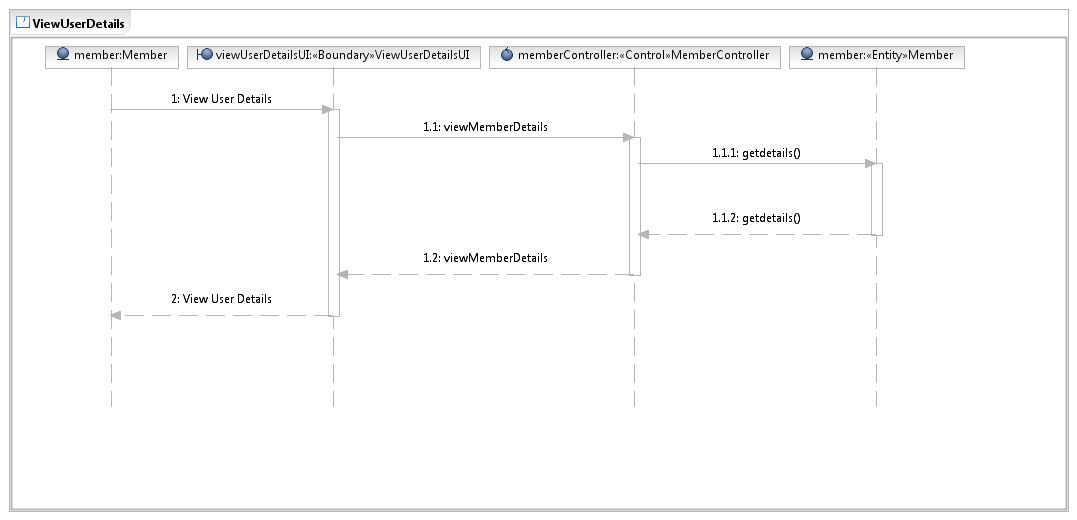
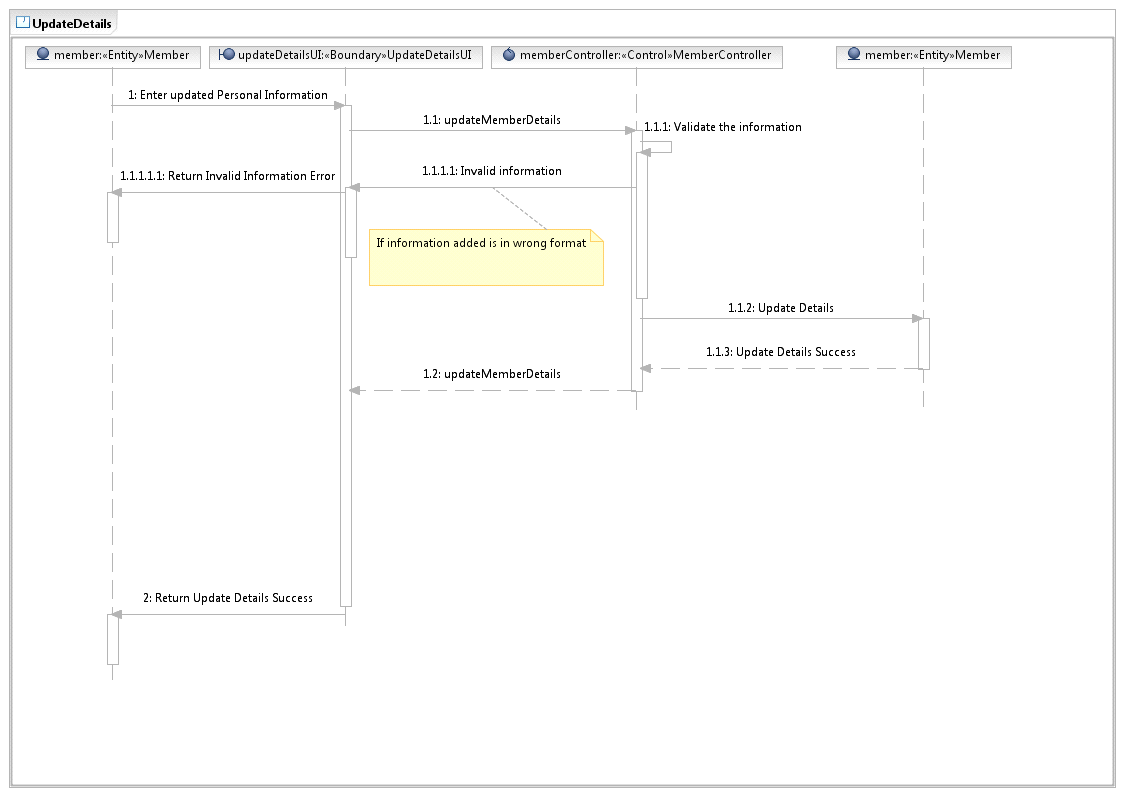
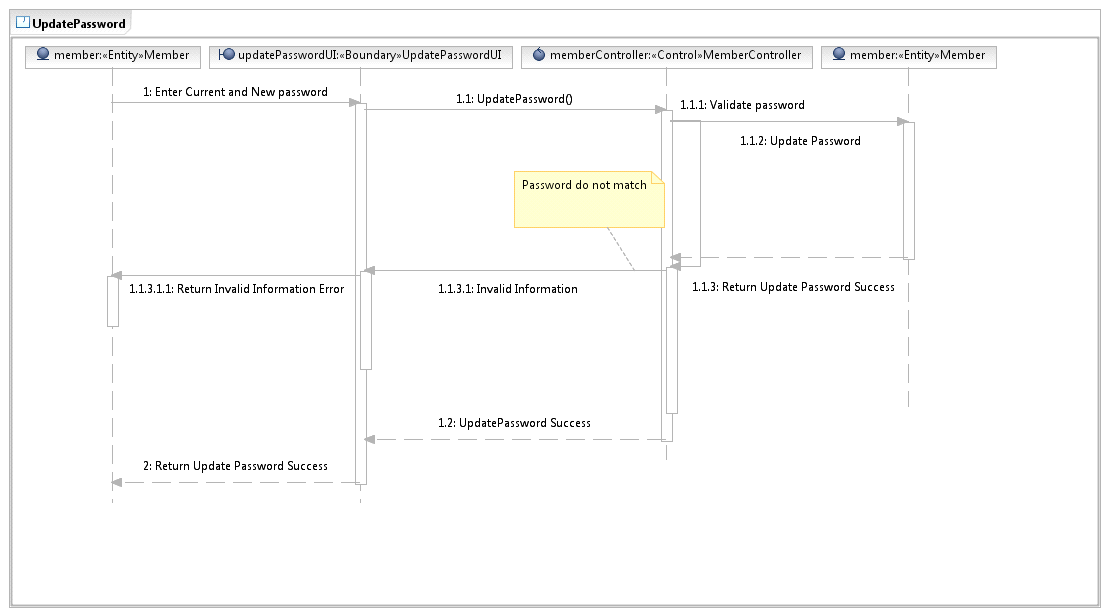
|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Name** | **Description** |
| Ver. 1.1 | 31/08/2012 | SRS Version 1.1. | Section 1: Project Team Information added  Section 2.1: Refined Software Requirements added  Section 2.2: Functional Requirements added  Section 2.4: Non-Functional Requirements added  Section 10.1: WBS added |
| Ver. 1.2 | 07/09/2012 | SRS Version 1.2. | Section 2.6: Use Case Model added  Section 3: Use Case/Activity Diagrams added |
| Ver. 1.3 | 21/09/2012 | SRS Version 1.3. | Section 2.1: Refined Software Requirements edited  Section 4: Analytical Model – Class Diagrams added  Section 5: Design Model – Sequence Diagrams added |
| Ver. 1.3 | 12/10/2012 | SRS Version 1.3. | Section 2.6: Use Case Model added  Section 3: Use Case/Activity Diagrams added  Section 4: Analytical Model – Class Diagrams edited  Section 5: Design Model – Sequence Diagrams edited |
| Ver. 1.4 | 03/11/2012 | SRS Version 1.4. | Section 6: Testing added |
|  |  |  |  |

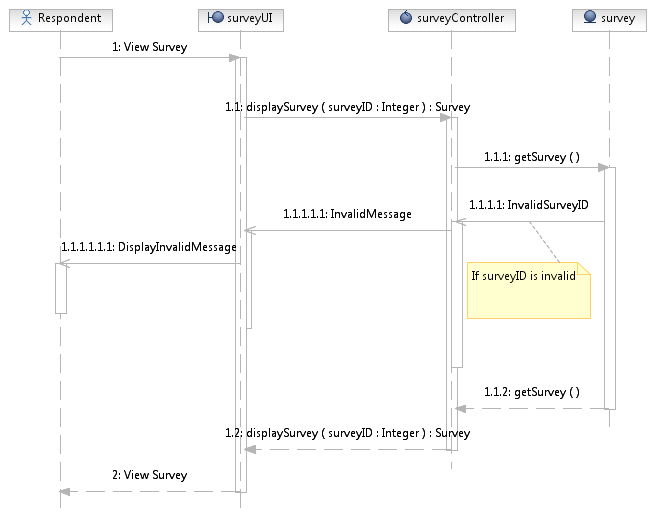
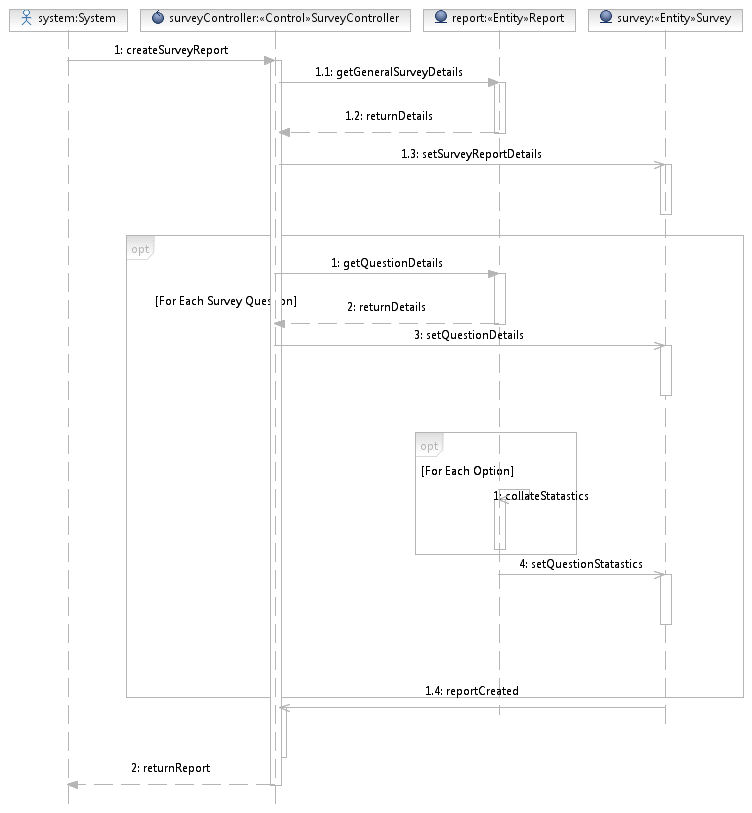
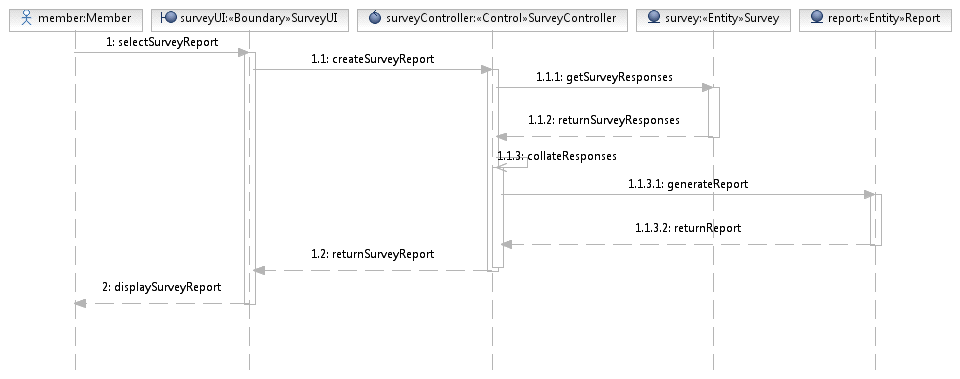
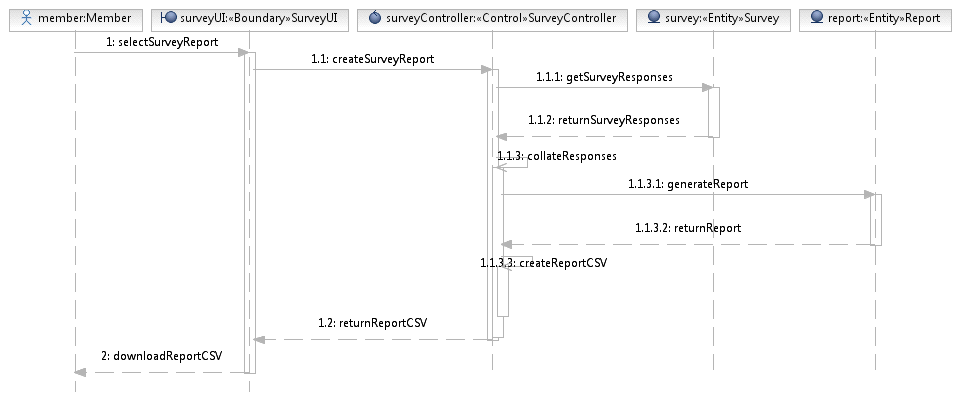
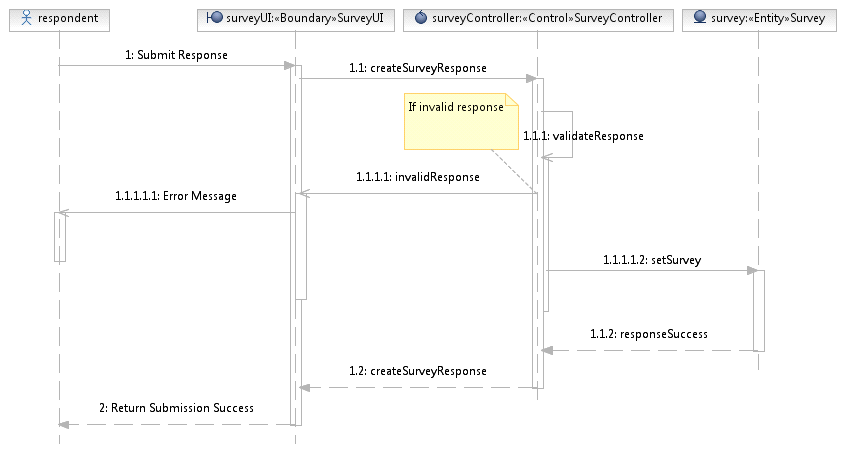
* **Use Case/Activity Diagrams**



* **Analytical Model – Class Diagrams**
* 
* **Design Model – Sequence Diagrams**
* Account Management
* Create member account



* Login
* View member account
* 
* Update member account
* 
* Update password
* 
* Survey Management
* Create Survey
* 
* Create Survey Questions
* 
* Create Survey Option
* 

* View Survey
* 
* Survey Report Management
* Generate Survey Report
* 
* View Survey Report
* 
* Export Survey Report
* 
* Respondents Actions
* Submit Survey
* 
* Ground Data Logging

Data Logging

* **Testing**
* 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();

}

}

}

}

}

}

* Functional Testing

**createSurvey.html**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Functional Requirements | Execution Details | System response | Remarks |
| Survey Form | User must be able to enter title and description for the survey form | Create survey form by clicking on “Form Properties” tab and edit in the text box for title and description | System instantly displays the changes while user updates the textbox. | Success |
|  | User must be able to add survey questions. | Create survey questions by choosing the question from “Add a Field” tab | System instantly displays question types and details correspondingly to the question added on the right side of the panel | Success |
|  | User must be able to reorder survey questions. | Reorder survey questions by clicking on the question and drag to desired positon. | System instantly displays the new order of survey questions. | Success |
|  | User must be able to edit survey questions. | Edit survey infomation by clicking on the corresponding field. | System instantly displays the changes while user updates the field. | Success |
|  | User must be able to delete survey questions. | Delete survey questions by clicking on the delete icon positioned at the bottom right hand corner of the survey question. | System instantly removes the survey question from the current form. | Success |
| Single Line Text | User must be able to update the description of text type question | Add text type question by clicking on the survey question and go to “Field Properties”.  Default vaule is “text”.  For instance, user changes the question title to “What is your opinion?” | System instantly displays the changes while user updates the field. | Success |
| Number | User must be able to update the description of number type question | Update field label by clicking on the survey question to go to “Field Properties”  Default value is “Number”.  For instance, user changes the word to “Please enter your number:” | System instantly displays the changes while user updates the field to “Please enter your number”. | Success |
| Paragraph Text | User must be able to update the description of paragraph text type of question | Update text type question by clicking on the survey question and to editing in the “Field Properties”  Default value is “Paragraph"  For instance, user changes the question title to “From the investigation, what would you like to provide to us? | System instantly displays the changes while user updates the field. | Success |
| Checkbox | User must be able to update the description of checkbox type question and options | Update field label by clicking on the survey question to go to “Field Properties”  Default value is “Checkboxes”.  For instance, user changes the word to “What days of the week are you available? (Check all that apply)” and options to “Monday”, “Tuesday”, “Wednesday”, “Thursday”, “Friday” and “All of the above” | System instantly displays the changes while user updates the field to “What days of the week are you available? (check all that apply)” | Success |
| Multiple Choice | User must be able to update the descriptions and options of multiple choice type of question | Update description and option by clicking on the survey question to go to “Field Properties”.  Defualt vaule is "Multiple Choice”.  For instance, user changes the desciption to “What is your favourite colour?”  Option can be edited also such as “Red”, “Black”, “White”. | System instantly displays the changes while user updates the field. | Success |
| Drop Down | User must be able to update the descriptions and options of dropdown question and options. | Update field label by clicking on the survey question to go to “Field Properties”  Default value is “Drop down”.  For instance, user changes the word to “What is your favourite colour?” and options to “Red”, “Blue”, “Green”, “Yellow‘ and “None of the above” | System instantly displays the changes while user updates the field to “What is your favourite colour?” | Success |
| Name | User must be able to update the format of name | Update name by clicking on the survey question to go to “Field Properties”  Default value is “Name”.  For instance, user can choose the format of the name want to display | System instantly displays the changes while user updates the field.  Eg: Simple Name    Eg: Extended Name | Success |
| Date | User must be able to update the descriptions and format of date question. | Update field label by clicking on the survey question to go to “Field Properties”  Default value is “Date”.  For instance, user changes the word to “Please enter your birthday:” | System instantly displays the changes while user updates the field to “Please enter your birthday:” | Success |
|  | User must be able to update the descriptions and format of date question. | Update field label by clicking on the survey question to go to “Field Properties”  Default format is “MM/DD/YYYY”.  For instance, user changes the format to “DD/MM/YYYY” | System instantly displays the changes while user updates the format to “DD/MM/YYYY” | Success |
| Phone | User must be able to update the descriptions and format of phone question. | Update field label by clicking on the survey question to go to “Field Properties”  Default value is “Phone”.  For instance, user changes the word to “Please enter your phone number:” | System instantly displays the changes while user updates the field to “Please enter your phone number:” | Success |
|  | User must be able to update the descriptions and format of phone question. | Update field label by clicking on the survey question to go to “Field Properties”  Default value is “(###)-###-####”.  For instance, user changes the format to “International” | System instantly displays the changes while user updates the field to “International” format. | Success |
| Website | User must be able to update the descriptions of website question. | Update field label by clicking on the survey question to go to “Field Properties”  Default value is “Web Site”.  For instance, user changes the word to “Please your company website:” | System instantly displays the changes while user updates the field to “Please your company website:” | Success |
| Price | User must be able to update the format of currency type question | Update field label by clicking on the survey question to go to “Field Properties”  Default value is “$Dollar”  For instance, user changes the currency format to pound. | System instantly displays the changes while user updates the field.  Eg: $ Dollar    Eg: $ Pound | Success |
| Email | User must be able to update the descriptions of email question. | Update field label by clicking on the survey question to go to “Field Properties”  Default value is “Email”.  For instance, user changes the word to “Please enter your e-mail address:” | System instantly displays the changes while user updates the field to “Please enter your e-mail address:” | Success |
| File Uploaded | User must be able to update the descriptions of file upload question. | Update field label by clicking on the survey question to go to “Field Properties”  Default value is “Upload a File”.  For instance, user changes the word to “Upload file here:” | System instantly displays the changes while user updates the field to “Upload file here:” | Success |
| Slider | User must be able to update the descriptions of slider question and options | Update field label by clicking on the survey question to go to “Field Properties”.  Default value is “Slider”.  For instance, user changes the word to “How will you rate this website? (1 to 10, 10 is the most)” | System instantly displays the changes while user updates the field to “How will you rate this website? (1 to 10, 10 is the most)” | Success |
|  | User must be able to update the options of the slider question | Update field label by clicking on the survey question to go to “Field Properties”  Default value of slider range is “10”.  Default value of minimum text is “1.”  Default value of maximum text is “10.” | System instantly displays the changes while user updates the minimum text vaule to “2” and maximum text vaule to “8”. | Success |
| Scaler | User must be able to update the descriptions of scalar question and options. | Update field label by clicking on the survey question to go to “Field Properties”  Default value is “Scaler”.  For instance, user changes the word to “Please rate your attractiveness on a scale of 1 to 5.” | System instantly displays the changes while user updates the field to “Please rate your attractiveness on a scale of 1 to 5.” | Success |
|  | User must be able to update the descriptions of scalar question and options. | Update field label by clicking on the survey question to go to “Field Properties”  Default value of scale amount is “5”.  Default value of worse text is “Worst.”  Default value of best text is “Best.”  For instance, user changes the scale amount to “10”, worst text to “Ugly” and best text to “Pretty” | System instantly displays the changes while user updates the scale amount to “10”, worst text to “Ugly” and best text to “Pretty”. | Success |
| Satisfactory | User must be able to update the descriptions of satisfactory type of question and its options. | Update field label by clicking on the survey question to go to “Field Properties”  Default value is “Satisfactory”.  For instance, user changes the word to “How do you rate our service provide to you” | System instantly displays the changes while user updates the field | Success |
|  | User must be able to update the options of the question for text input | Update field label by clicking on the survey question to go to “Field Properties”  Default value of options are “Very unsatisfactory, Unsatisfactory, Neutral, Satisfactory, Very Satisfactory”  For instance, user changes the options to “Very poor, Poor, Neutral, Good, Excellent” | System instantly displays the changes while user updates the field | Success |
|  | User must be able to update the options of the question for input image url | Update field label by clicking on the survey question to go to “Field Properties”  Default value of options are empty, user need to input the url link of the image in order to output the image  For instance, user input the image link into the field | System instantly displays the image while user updates the field | Success |
| Save Form | User must be able to save form after create the survey questions. | Save survey questions by clicking on the “Save Form” tab | System will save the survey questions into the database and publish the survey | Success |

viewsurveylist.html

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Functional Requirements | Execution Details | System response | Remarks |
| View Created Survey | User must be able to see a list of surveys created in the account | Click on “View Created Survey” link | System should display a list of survey created by the user. Created survey should have a link to view survey report and a download link as well as a survey link to go to the survey | Success |
|  | User must be able to see “No Surveys have been created” message if there is no existing surveys | Click on “View Created Survey” link | System should display a message as shown below: | Success |
| View Survey Report | User must be able to view graphs and statistics of the survey responses | Click on “View Survey Report” link | System should display the statistic results on the survey responses in graphs and charts.   * Pie Chart for radio button and scale radio questions      * Bar Graph for checkbox questions      * List for date, numerical inputs, text and textarea inputs questions | Success |
|  | User must be able to view the respondent data logging (time taken for the respondents to complete the survey) and the number of respondents who responded to the survey | Click on “View Survey Report” link | System should display the average time that respondents taken to complete the survey and the number of respondents who took and responded to the survey | Success |
| Download CSV Report - Choose to save | User can download a version of a Comma-Seperated-Values (CSV) Report | User clicks on “Download Report” link | System will prompt the user to choose where they want to save the CSV report. | Success |
| Download CSV Report - Save the Report | After the user chooses where to save the CSV report, the user can then click the save button to store the CSV file | User clicks on the “Save” button | System will save the CSV Report in the desired place. | Success |

register.html

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Functional Requirements | Execution Details | System response | Remarks |
| Register | User must be able to register an account. | User keys in the valid response for every fields.  User clicks on “Register” link. | System will prompt user to enter user’s personal and account information.  Fields are all correctly entered. Registration done! | Success |
|  | User will be able to see “Enter valid email address” if email address format is incorrect. | User keys in an invalid email address.  User clicks on “Register” link. | A pop-out will appear to prompt the user that he/she has entered an invalid email. | Success |
|  | User will be able to see “Enter valid username” if username format is incorrect. | User keys in an invalid username.  User clicks on “Register” link. | A pop-out will appear to prompt the user that he/she has entered an invalid username. | Success |
|  | User will be able to see “Enter another username” if username is already taken. | User keys in a duplicated username.  User clicks on “Register” link. | A pop-out will appear to prompt the user that he/she has entered a username that has already been taken by another user. | Success |
|  | User will be able to see “Enter valid password” if password is too short. | User keys in an invalid password.  User clicks on “Register” link. | A pop-out will appear to prompt the user that he/she has entered an invalid password. | Success |
|  | User will be able to see “Password don’t match” if password and confirm password field does not match. | User keys in different password on two different fields. | A pop-out will appear to prompt the user that he/she has entered an unmatched password. | Success |

EditUserDetails.htm

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Functional Requirements | Execution Details | System response | Remarks |
| Edit Details | User must have an account and is able to update the account | User edit the fields and he/she has to re-enter password in order to update the details. | System will prompt that details are updated if all the fields are entered correctly. | Success |
|  | User tries to update with incorrect password. | User key in an invalid password when “update” button is clicked. | System will prompt the user that he/she has entered an incorrect password. | Success |
|  | User tries to update the account details with blank field. | User leave 1 or more fields empty when “update” button is clicked. | System will prompt the user that he/she has to fill in all the fields before proceeding. | Success |
|  | Email address format is not entered correctly. | User key in an invalid email address format when “update” button is clicked. | System will prompt the user that he/she has entered an invalid email. | Success |
|  | Email address entered is already in use. | User enters an email address that has already been used. | System will prompt the user that the email address has already been used. | Success |

UpdatePassword.htm

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Functional Requirements | Execution Details | System response | Remarks |
| Update password | User must have an account and is able to update the account password | User key in the old password, new password and confirm the password before proceeding to update. | System will prompt that password are updated successfully if all the fields are entered correctly. | Success |
|  | User tries to update with incorrect password. | User key in an invalid password that does not match with the username when “update” button is clicked. | System will prompt the user that he/she has entered an incorrect password. | Success |
|  | User tries to update with mismatch password. | User key in mismatch password for the new and confirm password field when “update” button is clicked. | System will prompt the user that the passwords entered are mismatch. | Success |
|  | User tries to update with password with less than 6 characters long. | User key in password with less than 6 characters when “update” button is clicked. | System will prompt the user that the password should be at least 6 characters long. | Success |

Login.htm

|  |  |  |  |
| --- | --- | --- | --- |
| **Functional Requirement** | **Execution Details** | **System Response** | **Remarks** |
| Non-registered username during system login | Fill in the username with the non-registered username. Fill in the password field with the correct password. Click on submit. | Not successful login and system will send alert saying username or password not found. | Successfully implement the function |
| Wrong password during system login | Fill in the username with the registered username. Fill in the password field with the wrong password. Click on submit. | Not successful login and system will send alert saying username or password not found. | Successfully implement the function |
| System login function | Fill in the username with the registered username. Fill in the password field with the correct password. Click on submit. | Registered user is logged in. User is redirect to registered user page. | Successfully implement the function |
| System facebook login function | Click on the "Login with Facebook" button. Log in using the registered Facebook account. | User is logged in to the page.  The facebook button will change to welcoming message and then user will be redirected to the home page of registered user. | Successfully implement the function |
| Non-registered facebook user login function | Click on the "Login with Facebook" button. Log in using the non-registered Facebook account. | Facebook user is registered. User data is captured and saved in the database. Then it will continue with the same response as the registered Facebook user login function. | Successfully implement the function |

* **Discussion**

Due to our inexperience in Microsoft CLR languages, we initially struggled to setup the development environment.

A major and early problem was the setting up of the persistence layer. Due to the requirement for a quick and hassle free environment where we could just focus on business logic, we decided to use a ORM instead of handling SQL queries.

Therefore we chose nHibernate. However, there were few examples on how to get nHibernate to work with postgresql. Ultimately though, we managed to setup nHibernate and so made all persistence calls/interactions much easier.

* **Explain how you derived your analytical design and models**

We derived our analytical designs and models from the functional requirements as listed in section 2.2 and also the use cases from section 2.6 and 3. By looking at the functional requirements, we are able to determine what at the things we want our program to be able to do, and thus, develop use cases. The use cases tell us what the users can do to perform the various functionalities as depicted in section 2.2.

For example, one of the functionalities of the program is to enable users to register an account with our system, hence, becoming a member. Thus, this feature is added into the use case diagram as depicted in section 2.6.1. Also, the corresponding use case description is added in section 2.6.2. Finally, from there, we determine that we require a member class store all the details a user has keyed in. Hence, a member class is added to the analytical design.

* **Explain how you performed your UI design**

Our UI design has gone through many stages of changes to become the final product that it is now. Firstly, we look at commercial websites and learn from those websites on how we can design our UI. We also looked for plug-ins we can put into our system to make user interaction with our system much easier and more dynamic. In the initial stages of designing the UI, we come up with a lo-fi prototype which is hand-drawn on a hard copy to visualize how we want our system to look like. Subsequently, we come up with a hi-fi prototype which has a closer resemblance to our final UI design. Finally, we added in other features to beautify our website, (eg. a slideshow of the features of our website) as finish touches to our UI design.

* **Describe difficulties encountered and solutions applied**

We have faced technical limitations as Amazon places a high cost on computational cycles consumed and database transactions. Thus, we have moved most of application logic to client.

Another issue is the standard method of serving webpages by server generating HTML pages and sending them to the client not flexible enough to allow a mobile application. Thus, we used a Stateless RESTful Service Oriented Architecture

Another difficulties is that most of us are new to Javascript and C#. However we went on research from the web and with the help of programmer in our team, we managed to overcome this problem and has completed this project successfully.

* **Work breakdown structure**