NANYANG TECHNOLOGICAL UNIVERSITY

CZ2006 Software Engineering

Stardom Final Report

Supervisor: Dr. Kwoh Chee Keong

Team Members: Wan Liuyang

Chia Ming En

Qian Cheng

Zhao Fang Yuan

Ning Hao Yan

Huang Wei

Project Web Address:

http://ntusurvey.sfdye.com

(Support Firefox, Chrome, Safari, IE9(without compatibility setting))

Wiki Address:
http://ntusurvey.wikispaces.com/CZ2006+Software+Engineering+Course+Project+Wiki

SVN Address:

https://172.21.147.100:8083/svn/Stardom

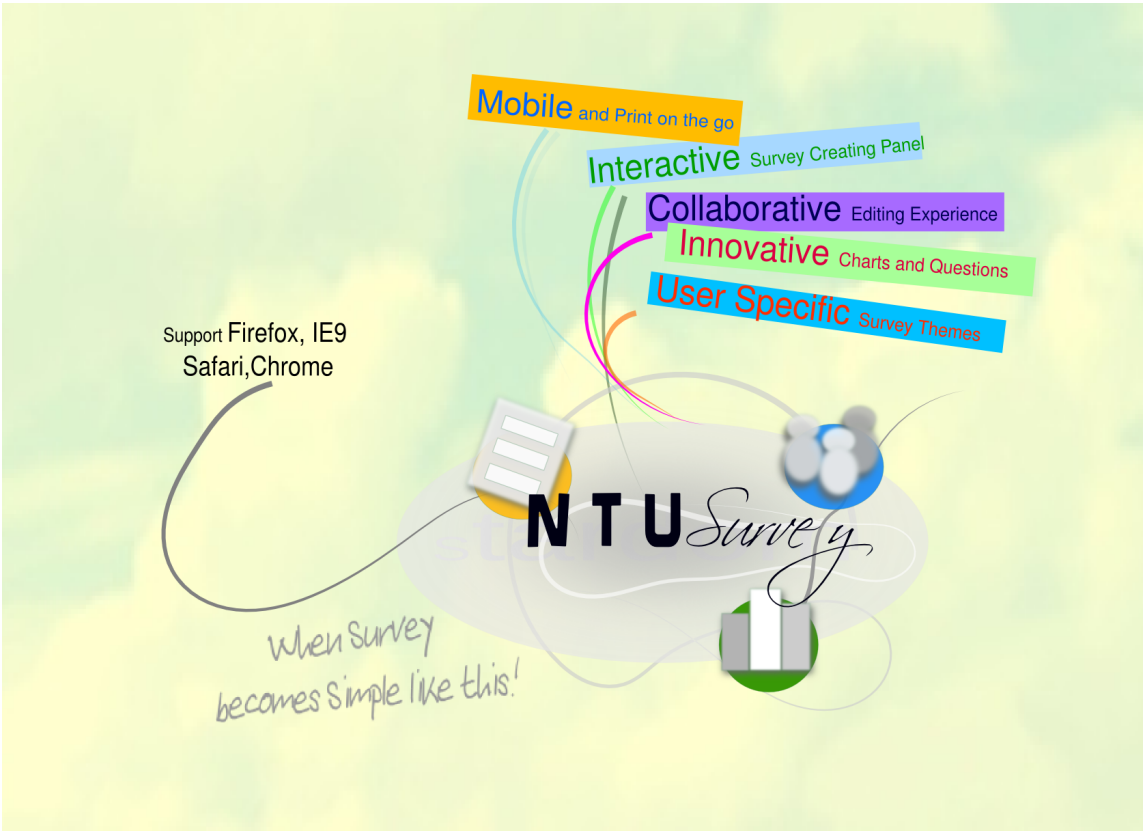
Acknowledgement

We are a group of students from School of Computer Engineering, Nanyang Technological University. This website is based on our course project of CZ2006 Software Engineering. The name "stardom" is given in confidence that this website will outshine all other projects in this course. The survey system is created in one semester with our best effort to beautify and perfect it. In this process, we keep examining the functions of various sub-components and also check our mistake and make adjustment. We spend countless effort in coding as well as testing various functions and subsystems. With all these effort, we hope and aim to best extend our ability in benefiting NTU students and faculties in research study or any other purposes. Hopes you enjoy using our website!

We would like to thank Dr. Kwoh in his guidance during project development, as well as the valuable teaching from Dr. Miao and Dr. Chang in guiding us through the whole process.

Stardom

Nov 2012



Content

1. Project Team Information

1. Wan Liuyang (manager)
2. Chia Mingen
3. Qian Cheng
4. Zhao Fangyuan
5. Ning Haoyan
6. Chu Xiaoqi
7. Huang Wei

2. Software Requirement Specification

2.1 Product Description

2.1.1 Product Vision

From profitable business to academic-excellent institute, a successful story usually starts by a discovery of the need from people. This requires a sharp eye of the market with supporting data, which takes quantum amount of effort to collect. One common data collection tool is paper survey. It can quickly gather a large amount of data, but at the same time, consumes large amount of paper, which is obviously not a sustainable solution. Then 'Can we make survey sustainable and green'? Yes, with the wide application of network technology, e-survey now become the 'wonder-tool', and NTUSurvey is a typical example.

NTUSurvey is a real-time data collection system. It supports online survey design and management. It serves as a free and convenient platform for NTU staffs and students to gather information for campus feedback, research, marketing, etc. NTUSurvey aims itself as a highly flexible survey creation system, while maintaining safety and reliable feature for data collection. It allows users to select various question types. Users can quickly share the survey and easily obtain the insight of data through inbuilt visual application. This survey platform can be helpful in collecting reliable results, bridge the gap between student feedback and decision makers, thus assist in better decision making around campus, and accelerate the process of building a vivid campus community. In summary, the mission of NTUSurvey is to deliver the most reliable survey results with the most simple survey-creation process, to benefit every NTU users.

2.1.2Business Requirements

NTUSurvey should be free of charge, dedicate itself to serve NTU staffs and students. The interface should be clean and clear. Little and almost zero learning time is required from users. It should help obtain the correct response and gather the information into various charts. Testing process is a must. It should be conducted before, during, and after coding process. NTUSurvey should also be evaluated by respondents and users out of the project team. Besides, this survey system must be completed within 2 months, yet maintaining a high standard which meets and exceeds user requirements.

2.1.3 Stakeholders and Users

Stakeholders of NTUSurvey are project developers, system users, and project supervisor. Project developers contribute to the creation of the survey website, and they are the members of management team. System users are students and staff in NTU. They can use NTUSurvey to market for campus events, or conduct academic research. They join in as a member of system and will be benefited by this free e-service. Project supervisor is lab supervisor who provides valuable guidance for software development process, and will evaluate the product. In addition, common users are people who respond to surveys, or say, survey respondents.

2.1.4 Project Scope

NTUsurvey is a web-based survey system with three key features. It supports creation of survey, deployment of survey questionnaire, and insightful data visualization. Users can design questions online, re-edit questions, and share their design easily with friends and classmates for them to respond. For every login, users are presented with up-to-date data results from the respondents. Moreover, visual applications are built to give a better insight of data. The system provides at least four question types upon creating survey. Background data logging keep recording valuable data when respondents are filling out questionnaires. The combination of novelty and utility is the theme of NTUSurvey's interface and data visualization. Project delivery is expected to the end of semester 1, 2012-2013. The system will launch in campus to facilitate data inquiry via web service.

2.1.5 Assumptions

Users are assumed to have experience in using computer and internet. Before using the system, Internet browser is assumed to be installed correctly and internet connection is available.

2.1.6 Constraints

Time is limited within two months' working schedule. This 7-members team is limited in manpower as compared to competing teams, which requires each individual's devotion and support to make this project into a sound reality.

2.2 Functional Requirement

2.2.1 Register

2.2.1.1 user must fill in unregistered username, email, password and confirm password.

2.2.1.2 system must check if the username and e mail is registered. If yes, display error message

2.2.1.3 system must be able to detect if the password entered two times is the same. If not, display error message to ask the user to re-enter

2.2.1.4 system must be able to check if the particular filed satisfies the specific requirement.

2.2.1.5 system must be able to check if all the fields are filled when user wants to submit the registration form.

2.2.1.6 system must send the user a e mail with a activation link to the e mail registered.

2.2.1.7 user is only allowed to log in after activation

2.2.2 Log in

2.2.2.1 user must fill in user name or survey and password to enter

2.2.2.2 system must be able to check if the username/ e mail and password is correct. If not correct, system must be ble to show error message

2.2.3 Manage profile

2.2.3.1 user is able to edit on the profile for example name and date of birth

2.2.3.2 user is able to reset the password using e mail address

2.2.4 Create survey

2.2.4.1 user is able to choose the question type

2.2.4.2 After enter the question, the question is able to edit again

2.2.4.3 questions is able to move up and down

2.2.4.4 user is able to save the current survey questions

2.2.4.5 user is able to specify the deadline of the survey

2.2.4.6 user is able to specify the limit of words in paragraph and single line questions.-

2.2.4.7 user is able to select a desired background theme for his survey

2.2.5 Delete survey

2.2.5.1 user is able to select his survey to delete

2.2.5.2 system must display a warning message

2.2.6 Edit survey

2.2.6.1 user is able to select a survey to edit on the survey

2.2.6.2 system is able to detect if the survey is out of date and do not allow it to be edited

2.2.7 Response to survey

2.2.7.1 survey respondents is able to complete the survey question

2.2.7.2 system is able to check if the answer satisfies the requirement of the questions. (eg, word limit)

2.2.7.3 system is able to locate the respondent’s IP

2.2.8 View survey results

2.2.8.1 user is able to view the survey results once there is response to the survey

2.2.8.2 user is able select a desired chart from a group of charts to display the survey results.

2.2.9 Manage database

2.2.9.1 admin have the right to delete the surveys and users

2.3 Data Requirements

2.3.1 username must be a unique string of 4 to x characters

2.3.2 the password must be a string of x to x characters containing…

2.3.3 a valid e mail address must be given

2.4. Non-Functional Requirement

2.4.1 Usability: Different functionality is presented for different system roles for selection directly after successful login. System (Help and Error) messages will guide user for each functionality.

2.4.2 Reliability: System can handle wrong login password and username by giving error message. System can rejects invalid range of survey deadline; system can also rejects respondents after the survey deadline. System encrypt user password. System backup site is provided to prevent loss of data due to system crash.

2.4.3 Performance: The system shall not down for no more than 1 hour.

2.4.4 Supportability: Object oriented feature is applied for future system development. Documentation and coding comments are consistent and traceable (with index number).

2.5. Interface Requirements

2.5.1 User Interfaces

Login: This is for the Administrator and member to get into the system. It requires a user name/e-mail and password.

Registration: This utility is to create new member for the system.

User Account: This enables the user to check and change the information about the account. User can also view his own survey here.

Dashboard: This page able the users to edit, delete, share the survey or any other actions on the survey. This page shows all the surveys of the particular user.

Survey Creation: Registered user can create a new survey by offering a set of questions and answers.

Survey Completion: Once the survey is finished, a completion page will be shown.

Survey Report: User can view the result of survey.

Figure 1.1: Main page

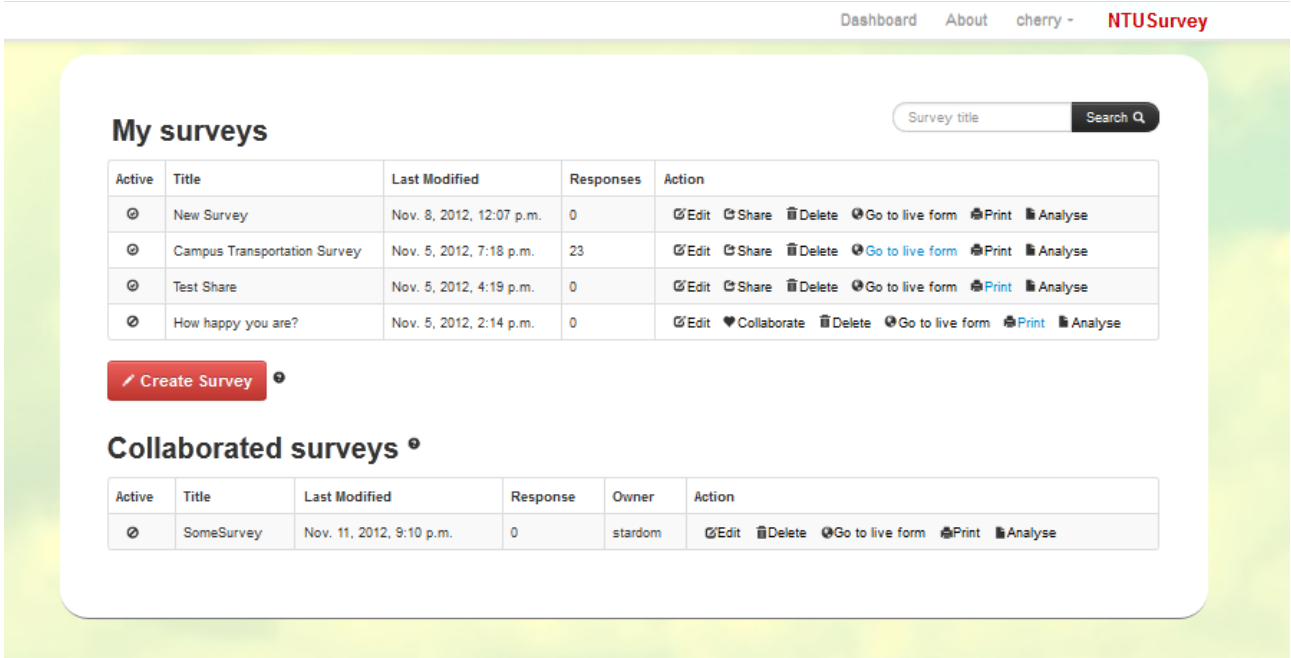


Figure 1.2: Logged in interface

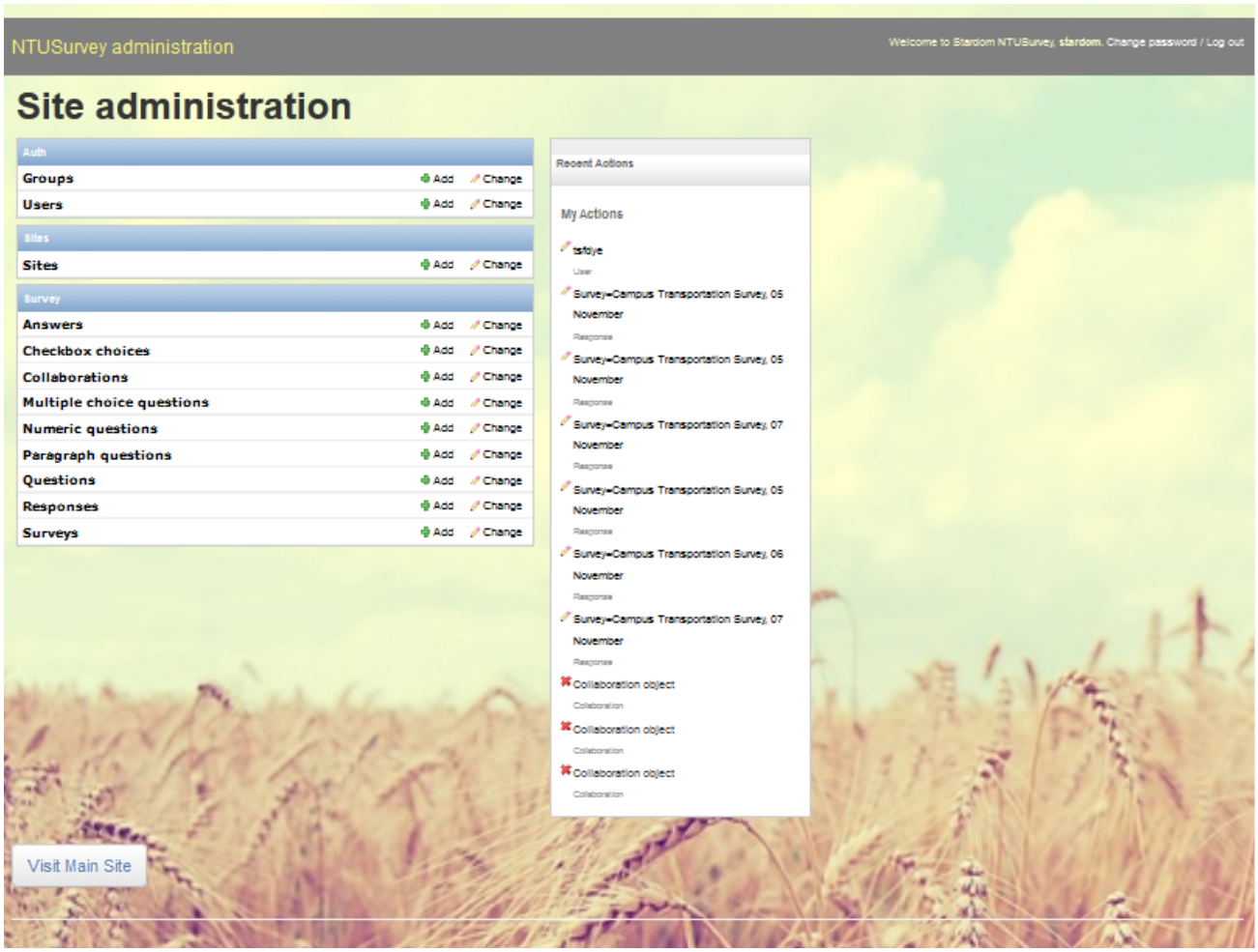


Figure 1.3: administration interface

2.5.2 Hardware Interfaces

Server: Mac OS X with Apache 2.2.17
This server interface is currently out of our scope of the system

2.5.3 Software Interfaces

Client: A browser supporting JavaScript, Unity3D.
Frontend: HTML, CSS, JavaScript, Unity3D
Backend: MySQL database, Django-1.4.1 frame work,

2.5.4 Communication Interfaces

The system uses python as server language and hence requires HTTP for transmission of data. More over this allows easy interaction between the various clients and the server.

3. Use Case Model

Use case is a list of steps, typically defining interactions between a role and a system, to achieve a goal. The actor can be a human or an external system. In this section, a use case dedicated to our project is attached. A detailed description of each use case descripts the flow of the specific use case.

3.1 Use Case Diagram-top level

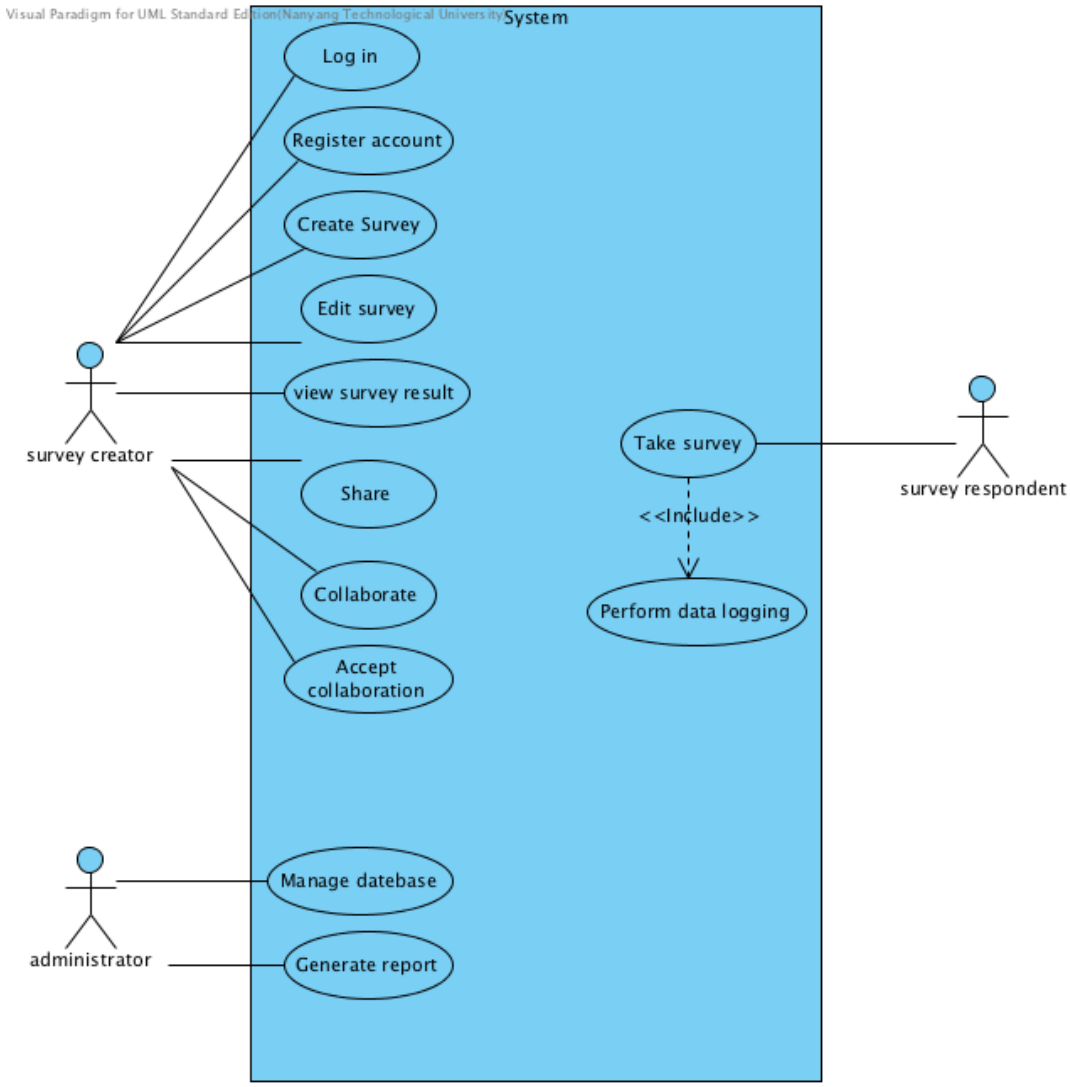


Figure 1.4: Use case diagram

3.2 Use Case Description

3.2.1 Login

Use Case ID:	UC - 01
Use Case Name:	**Login**

Actors:	Initiated by SurveyCreator
Description:	The goal of use case Login is to describes how a SurveyCreator login into NTUSurvey system.
Trigger:	SurveyCreator tries to login into the system
Preconditions:	SurveyCreator has already registered an account with the system
Postconditions:	SurveyCreator logged into the system
Normal Flow:	1. System display the login screen 2. SurveyCreator enters username/ e-mail 3. SurveyCreator enters password 4. SurveyCreator presses the login button 5. System validates the entered username/ e-mail and password A1: invalid username or password 6. System logs SurveyCreator into the system
Alternative Flows:	A1: invalid username or password 1. The system display an error message informing SurveyCreator that the username or password is invalid 2. SurveyCreator redirected back to login screen
Exceptions:	Nil
Includes:	Nil
Priority:	Top Priority

Frequency of Use:	50 times per day
Business Rules:	Nil
Special Requirements:	Nil
Assumptions:	Nil
Notes and Issues:	Nil

3.2.2 Register

Use Case ID:	UC - 02
Use Case Name:	**RegisterAccount**

Actors:	Initiated by SurveyCreator
Description:	The goal of use case RegisterAccount is to allow a user to register an account into NTUSurvey system.
Trigger:	A web user tries to register an account in the NTUsurvey system so that he/she could create surveys.
Preconditions:	SurveyCreator has navigated to online login interface.
Postconditions:	A new user account of SurveyCreator is added in system database.
Normal Flow:	1. SurveyCreator activates the interface of RegisterAccount. 2. The system asks SurveyCreator to enter email and password. 3. SurveyCreator filled in email and password and confirmed input. 4. The system validates if there exists an identical email in the database.

	A1: The email already exists in the database. 5. SurveyCreator chooses to edit profile. 6. System updates user profile.
Alternative Flows:	A1: The email already exists in the database. 1. System displays a message saying that the email belongs to an existing account. 2. SurveyCreator acknowledges the message. 2. System goes back to UC02-Normal Flow point 3.
Exceptions:	SurveyCreator has registered account already.
Includes:	Nil
Priority:	Top Priority
Frequency of Use:	50 times per day
Business Rules:	Nil
Special Requirements:	Nil
Assumptions:	Nil
Notes and Issues:	Registration may require some additional information like First Name, Last Name, Course of Study, etc. Necessary validations are to carry out. Examples: weak passwords are not allowed, the email provided should be valid.

3.2.3 Create survey

Use Case ID:	UC - 03
Use Case Name:	**CreateSurvey**

Actors:	SurveyCreator
Description:	The goal of use case CreateSurvey is to allow survey creators to construct survey questionnaires to obtain statistically useful information from survey respondents about a given topic.
Trigger:	SurveyCreator tries to construct a survey questionnaire.
Preconditions:	The survey creator must has successfully log-in already.
Postconditions:	A new survey questionnaire is constructed. Targeted respondents can take the survey if the survey is already published. Creator can re-edit it after it is saved and unpublish it after it is published.
Normal Flow:	1. Survey Creator activates the function to construct a new survey (i.e. Click “Create survey” button). 2. System shows the interface for Survey Creator to design questionnaire content. 3. Survey creator can click the title, description and deadline of the survey, and change them. 4. Survey creator selects a question type for each question to be asked, click “add”. Then a question will show out. 5. Once a question clicked, options of this question will be present. Survey creator could change the details, the order of questions, delete it, and then finish constructing. 6. The survey creator chooses to save it or publish the survey for respondents to take online. 7. System will validate the survey, if there is any illegal input, save process will end, and warning message will be presented at the bottom of the survey.

	8. If pass the validation, the system adds the survey to the list of surveys created by survey creator, and redirect to dashboard or published page.
Alternative Flows:	Nil
Exceptions:	Nil
Includes:	Nil
Priority:	Top priority
Frequency of Use:	10 times per day
Business Rules:	Nil
Special Requirements:	Survey title, description, deadline and attributes in each question must obey the validation rules.
Assumptions:	The survey content will always be legitimate; otherwise the administrator can take action to terminate it.
Notes and Issues:	Creator cannot creator two surveys with the same title. The exact time of creation should be recorded.

3.2.4 Edit survey

Use Case ID:	UC - 04
Use Case Name:	**EditSurvey**

Actors:	SurveyCreator
Description:	The goal of use case EditSurvey is to allow survey creators to edit the surveys that have been saved under their accounts in the system.
Trigger:	A SurveyCreator tries to edit a survey previously saved.
Preconditions:	The SurveyCreator must successfully login.

	A survey must have been previously saved by the same SurveyCreator.
Postconditions:	Changes made to the survey are saved. Targeted respondents can take the survey if it's successfully deployed.
Normal Flow:	1. The use case starts when a SurveyCreator wants to edit a survey previously saved. 2. In dashboard, the survey creator selects a particular survey and click "edit." 3. The system directs the survey creator to another web page where the survey questionnaire is in edit mode. 4. The survey creator makes changes to the survey questionnaire, and presses save or publish (unpublish or update if published). 5. System will validate the survey, if there is any illegal input, save process will end, and warning message will be presented at the bottom of the survey. 6. If pass the validation, the system update the information of this survey. And record the last modified time.
Alternative Flows:	Nil
Exceptions:	Nil
Includes:	Nil
Priority:	Top priority
Frequency of Use:	30 times per day
Business Rules:	Nil
Special Requirements:	Survey title, description, deadline and attributes in each question must obey the validation rules. The time of last edit should be recorded and shown to the

	survey creator.
Assumptions:	The survey creator will always make certain changes when the survey questionnaire is in edit mode so that there's no need for the system to detect changes.
Notes and Issues:	Nil

3.2.5 Data logging

Use Case ID:	UC - 05
Use Case Name:	**PerformDataLogging**

Actors:	Nil (abstract use case)
Description:	The goal of use case PerformDataLogging is to obtain situational data (e.g., recording a respondent's IP, etc.) and behavioral data related to each respondent's behavior when filling out the questionnaire.
Trigger:	A web users starts to take a survey from the system.
Preconditions:	This use case must be called by TakeSurvey use case.
Postconditions:	SurveyRespondent completes the survey questionnaire. SurveyRespondent's data is successfully captured by carrying out background data logging.
Normal Flow:	1. The use case starts when the calling use case TakeSurvey is being executed. 2. System carries out background data logging to acquire certain situational and behavioral data from the survey respondent. A1: The internet connection breaks up. 3. SurveyRespondent completes the survey. 4. The background data from the respondent is recorded

	in the database.
Alternative Flows:	A1: The internet connection breaks up. 1. The use case ends.
Exceptions:	Nil
Includes:	Nil
Priority:	Medium priority
Frequency of Use:	50 times per day
Business Rules:	Nil
Special Requirements:	Internet connection is stable.
Assumptions:	Internet connection remains stable throughout the process.
Notes and Issues:	Nil

3.2.6 Take survey

Use Case ID:	UC - 06
Use Case Name:	**TakeSurvey**

Actors:	SurveyRespondent
Description:	The goal of this use case is to allow Survey Respondent to complete the survey
Trigger:	Survey Respondentchoose to start the survey
Preconditions:	An URL must be given to Survey Respondent by the Survey Creatorto access the survey
Postconditions:	The result of the survey will be store in database

Normal Flow:	1. SurveyRespondents access the survey through the URL given by the survey creator 2. The survey will display the question set by the creator 3. SurveyRespondent finish the survey and click “submit” button. 4. System will validate the answers, if there is any illegal input, submit process will end, and warning message will be presented at the bottom of the survey. 5.If pass the validation, the result of the survey will be stored in the database. Also the IP and the time used for the survey will be record. 6. SurveyRespondent will be redirect to submit success page.
Alternative Flows:	Nil
Exceptions:	Nil
Includes:	Perform Data Logging
Priority:	Medium priority
Frequency of Use:	50 times per day
Business Rules:	Nil
Special Requirements:	The input must obey the validation rules which specified by both survey creator and system.
Assumptions:	Internet connection remains stable throughout the process.
Notes and Issues:	Nil

3.2.7 View results

Use Case ID:	UC - 07
Use Case Name:	**ViewSurveyResult**

Actors:	SurveyCreator
Description:	The goal of use case ViewSurveyResult is to allow survey creators to view the result from survey respondents for all the survey he/she has created in a statistically and organized manner
Trigger:	Survey Creatorchoose to view the survey result from their control panel
Preconditions:	The SurveyCreator must successfully login. The SurveyCreator can only access the survey he/she created
Postconditions:	Nil
Normal Flow:	1. The use case starts when a survey creator want to view the result of the survey that he/she has previously created 2. System displays a list of titles of the survey 3. SurveyCreator select a particular survey title 4. The system fetches the necessary information from the database and generate a report of the survey result.
Alternative Flows:	Nil
Exceptions:	Nil
Includes:	Perform Data Logging
Priority:	Medium priority
Frequency of Use:	10 times per day
Business Rules:	Nil

Special Requirements:	The report of survey result consists of the number of survey respondents, text/graphic representation of survey respondent's answer for each question, time take for survey respondent to complete the survey etc.
Assumptions:	Nil
Notes and Issues:	Nil

3.2.8 Share Survey

Use Case ID:	UC - 08
Use Case Name:	**Share Survey**

Actors:	SurveyCreator
Description:	The goal of use case Share Survey is to describes how a SurveyCreator can share the survey with Survey Respondent.
Trigger:	SurveyCreator click the share button on the survey
Preconditions:	SurveyCreator must have created the survey which he/she want to share
Postconditions:	System will send an email with the survey link to the email address specify by the SurveyCreator
Normal Flow:	1. SurveyCreator click the share button on the survey 2. System display a popup form 3. SurveyCreator enters the email address that he/she want to share the survey with 4. SurveyCreator enter the message for the email 5. System validates the entered email address A1: invalid email address 6. System send the survey link to the email address specify by the SurveyCreator 7. System display a success message
Alternative Flows:	A1: invalid email address

	1. The system display an error message informing SurveyCreator that the email address is invalid
Exceptions:	Nil
Includes:	Nil
Priority:	Top priority
Frequency of Use:	50 times per day
Business Rules:	Nil
Special Requirements:	Nil
Assumptions:	Nil
Notes and Issues:	Nil

3.2.9 Collaborate Survey

Use Case ID:	UC - 09
Use Case Name:	**Collaborate Survey**

Actors:	SurveyCreator
Description:	The goal of use case Collaborate Survey is to describes how a SurveyCreator can invite other SurveyCreator to collaborate on a survey together
Trigger:	SurveyCreator click the collaborate button on the survey
Preconditions:	SurveyCreator must have created the survey which he/she want to collaborate
Postconditions:	System will send an email with a link to accept the invitation to the email address specify by the SurveyCreator
Normal Flow:	1. SurveyCreator click the collaborate button on the survey

	2. System display a popup form 3. SurveyCreator invite collaborator by entering the username/email of the SurveyCreator that he/she want to collaborate with. 5. System validates the entered username/email A1: SurveyCreator specified does not exist 6. System send the email with a link to accept the invitation to the email address specify by the SurveyCreator 7. System display a success message
Alternative Flows:	A1: SurveyCreator specified does not exist 1. The system display an error message informing SurveyCreator that the collaborator specified does not exist
Exceptions:	Nil
Includes:	Nil
Priority:	Top priority
Frequency of Use:	50 times per day
Business Rules:	Nil
Special Requirements:	Nil
Assumptions:	Nil
Notes and Issues:	Nil

3.2.10 Accept Collaboration

Use Case ID:	UC - 10
Use Case Name:	**Accept Collaboration**

Actors:	SurveyCreator
Description:	The goal of use case Accept Collaboration is to describes how a SurveyCreator accept the invitation of other SurveyCreator to collaborate on a survey
Trigger:	SurveyCreator click the link in the invitation email
Preconditions:	SurveyCreator must have received the invitation email
Postconditions:	SurveyCreator will be able to work on a survey collaboratively with other SurveyCreator
Normal Flow:	1. SurveyCreator click on the link in the invitation email 2. System update the list of collaborated survey of SurveyCreator 3. System show a success message informing SurveyCreator that he/she have accepted the invitation. 4. SurveyCreator can access the collaborated survey in their account page
Alternative Flows:	Nil
Exceptions:	Nil
Includes:	Nil
Priority:	Top priority
Frequency of Use:	50 times per day
Business Rules:	Nil
Special Requirements:	Nil
Assumptions:	Nil

Notes and Issues:	Nil

3.2.11 Print

Use Case ID:	UC - 11
Use Case Name:	**Print**

Actors:	SurveyCreator
Description:	The goal of use case Print is to describes how the system display a survey in print format
Trigger:	SurveyCreator click the print button
Preconditions:	SurveyCreator must have created the survey
Postconditions:	Survey is displayed in print format
Normal Flow:	1. SurveyCreator click on the print button 2. System get all the question of the survey 3. System generate a QR code for the survey 4. System display survey in print mode
Alternative Flows:	Nil
Exceptions:	Nil
Includes:	Nil
Priority:	Top priority
Frequency of Use:	50 times per day
Business Rules:	Nil
Special Requirements:	Nil
Assumptions:	Nil
Notes and Issues:	Nil

3.2.12 Generate Data Logging Report

Use Case ID:	UC - 08
Use Case Name:	**GenerateDataLoggingReport**

Actors:	Administrator
Description:	The goal of this use case is to display the data collected from data logging system
Trigger:	Administrator choose to view the data collected from the data logging system
Preconditions:	The administrator must successfully login.
Postconditions:	Nil
Normal Flow:	1. The use case starts when a administrator clicks on the "Generate Data Logging Report" button 2. System fetches the data collected by the data logging system from database and data are interpreted and displayed in report format
Alternative Flows:	Nil
Exceptions:	Nil
Includes:	Nil
Priority:	Medium priority
Frequency of Use:	10 times per day
Business Rules:	Nil
Special Requirements:	Nil
Assumptions:	Nil
Notes and Issues:	Nil

3.2.13 Manage database

Use Case ID:	UC - 09
Use Case Name:	**ManageDatabase**

Actors:	Administrator
Description:	The goal of this use case is to allow administrator to manage the system
Trigger:	Nil
Preconditions:	The administrator must successfully login.
Postconditions:	Nil
Normal Flow:	1. The use case starts when Administrator click on the "Manage user" button 2. System displays a list of survey creator 3. Administrator will be able to delete the survey creator from the system
Alternative Flows:	1. The use case starts when a administrator click on the "Manage survey" button 2. The system lists out all the survey in the database 3. Administrator will be able to delete the survey from the system
Exceptions:	Nil
Includes:	Nil
Priority:	Low priority
Frequency of Use:	5 times per day
Business Rules:	Nil
Special Requirements:	Nil
Assumptions:	Nil

Notes and Issues:	Nil

4. Class Diagrams

Class diagram is a static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations and the relationships among the classes.

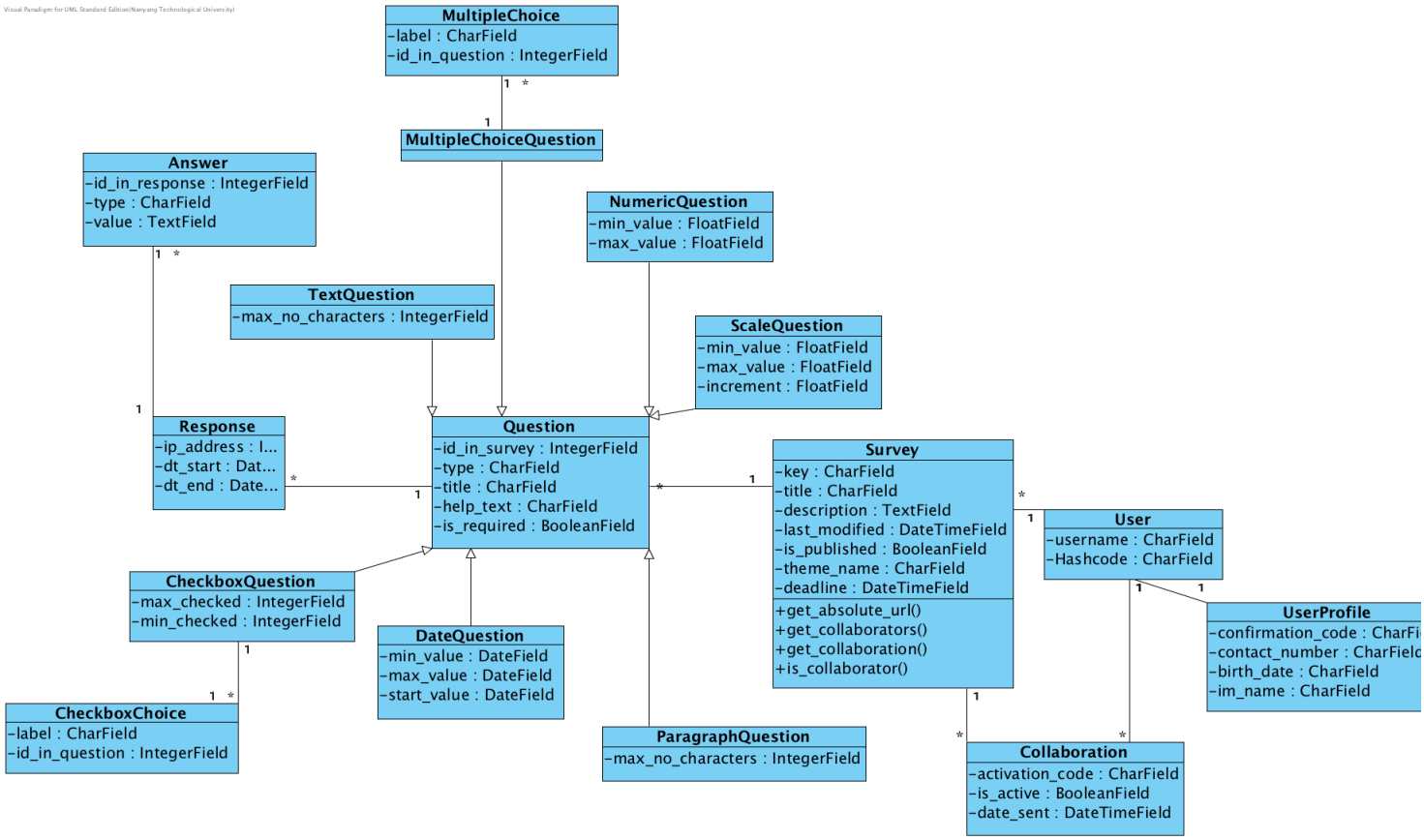
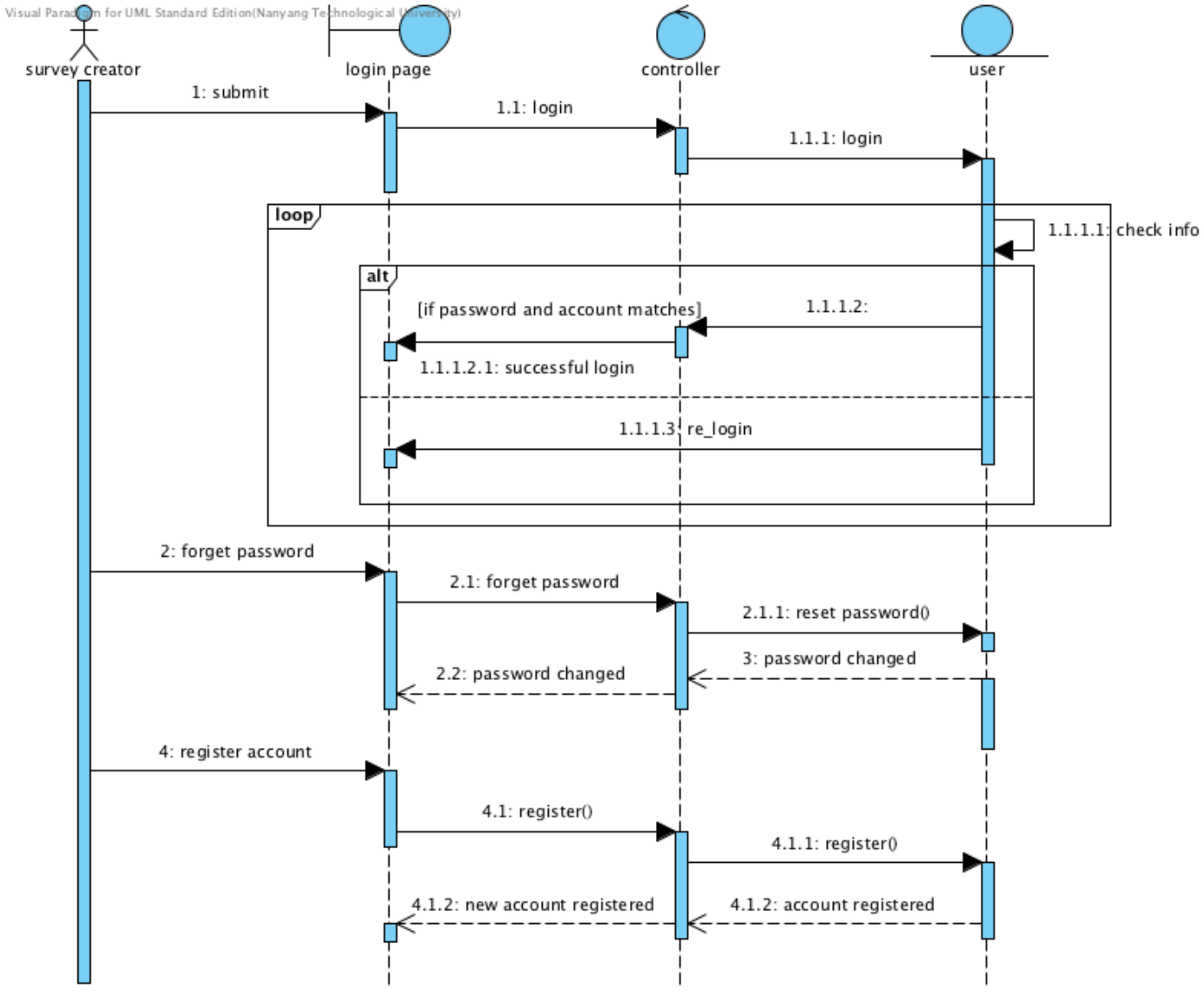


Figure 1.5: Class diagram

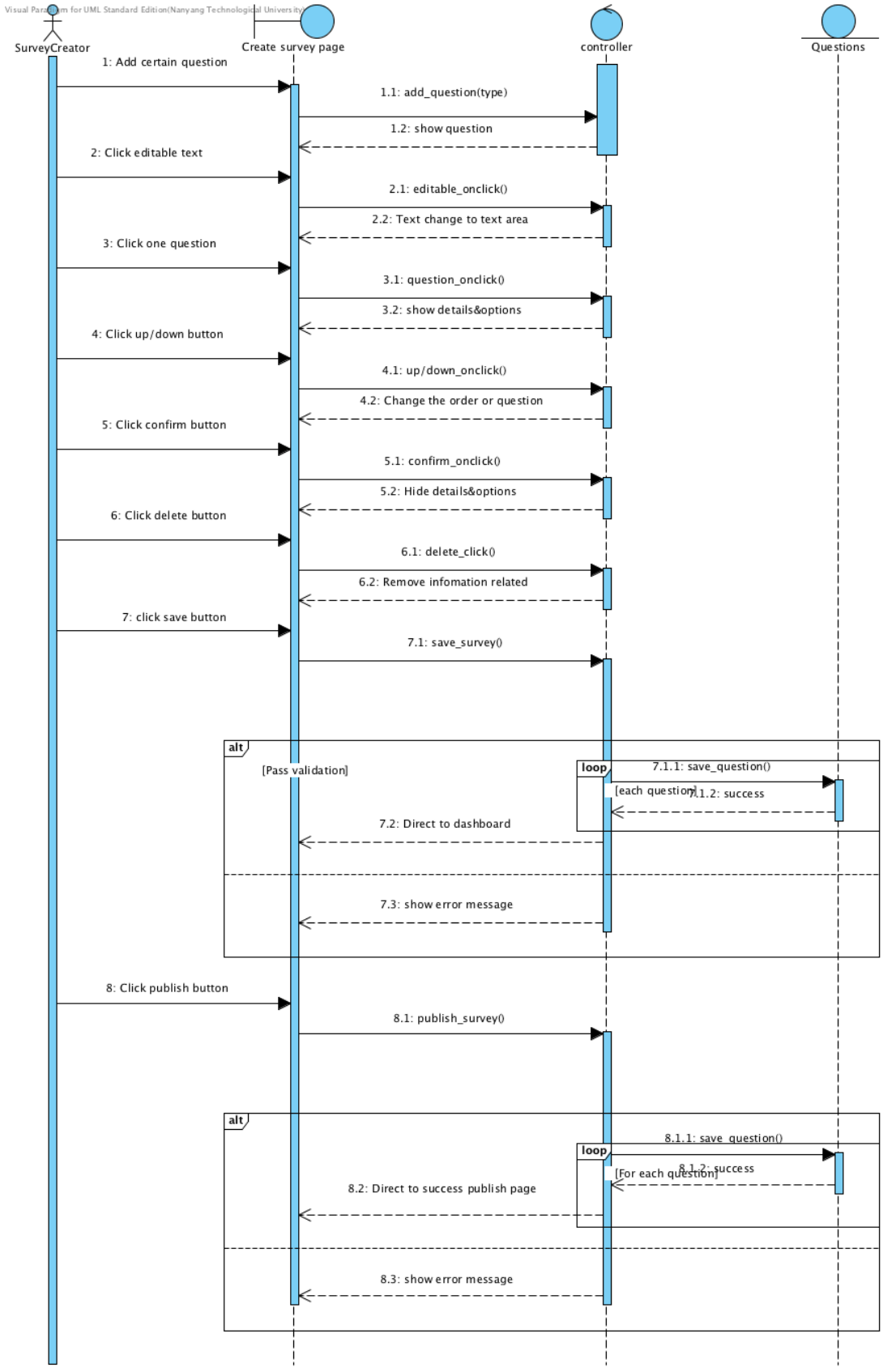
5. Sequence Diagrams

Sequence diagram is an interactive diagram that shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.

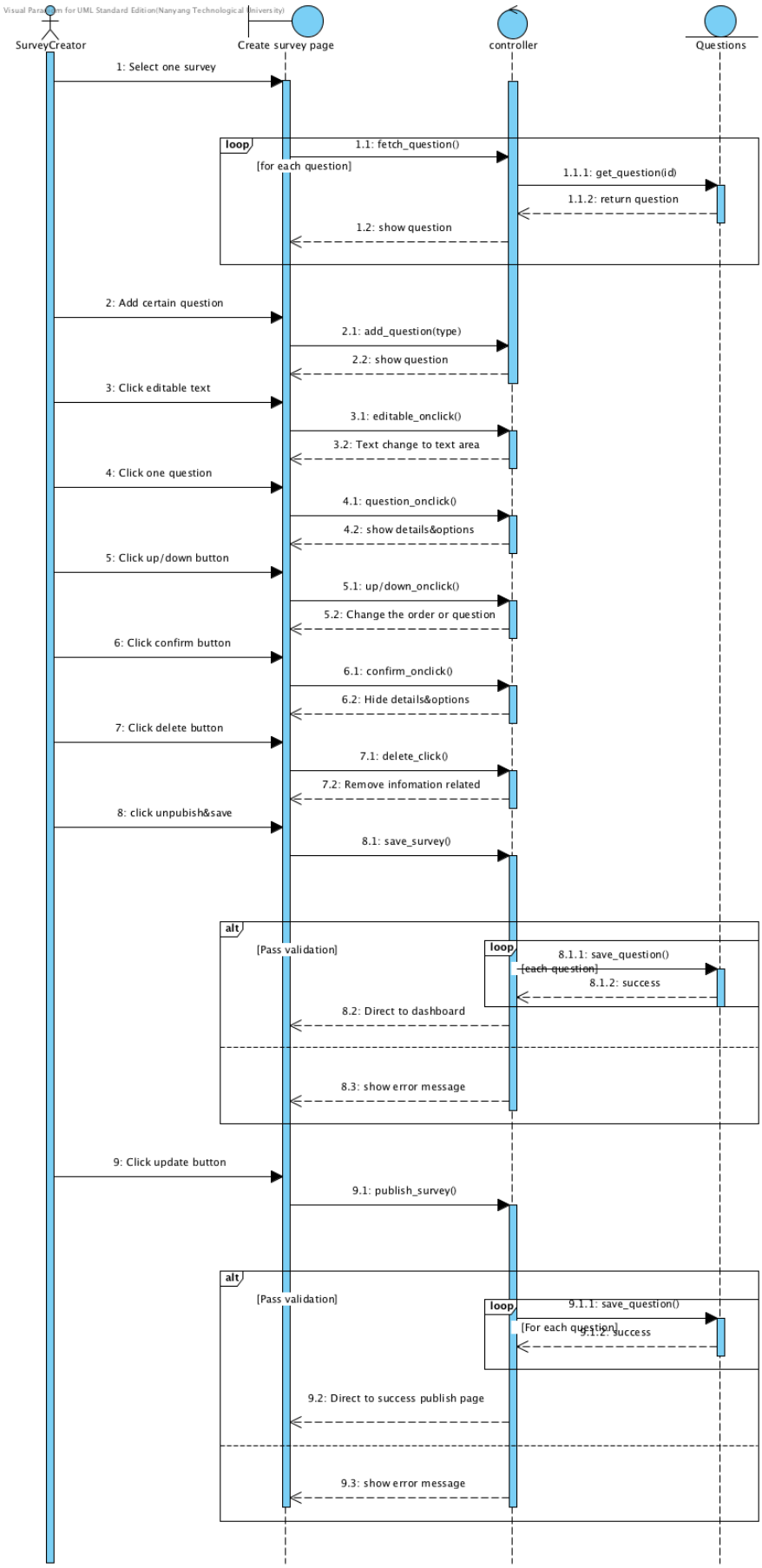
5.1 Register and login



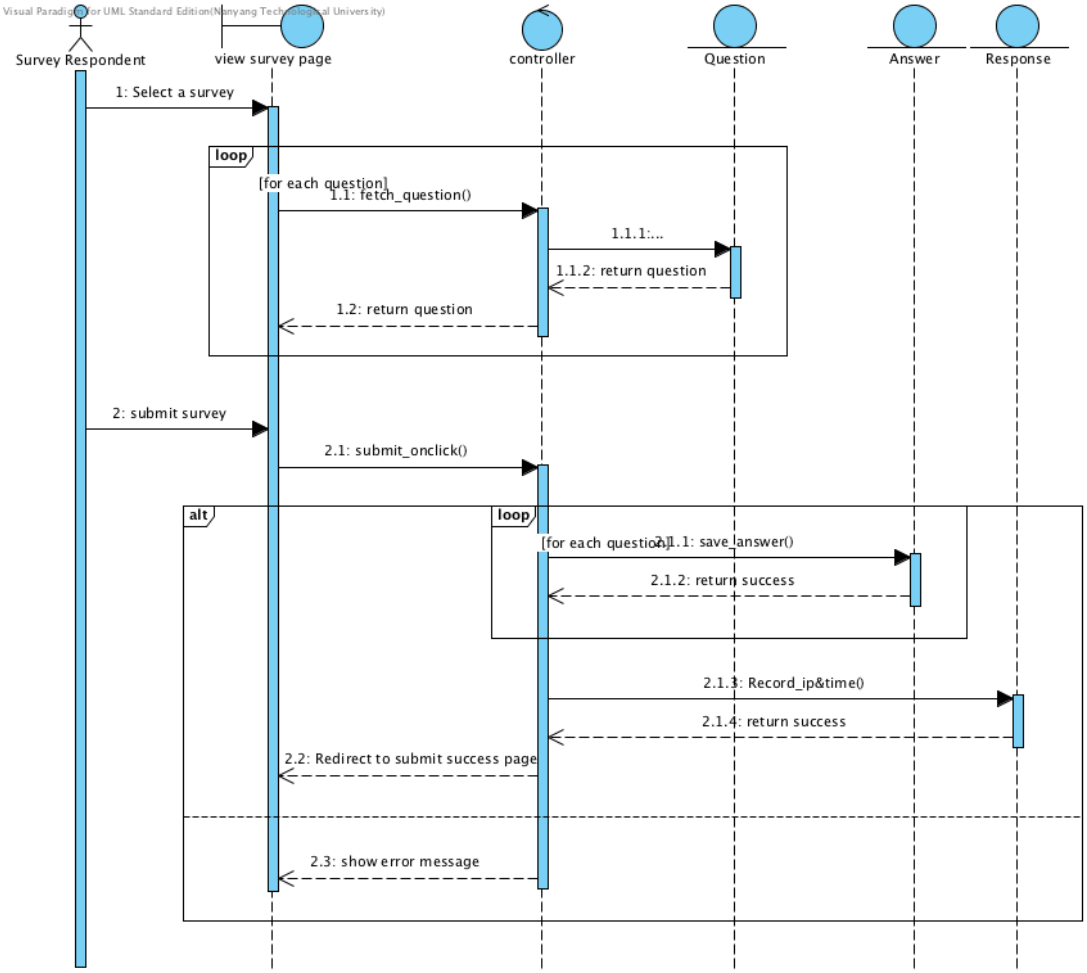
5.2 Create survey



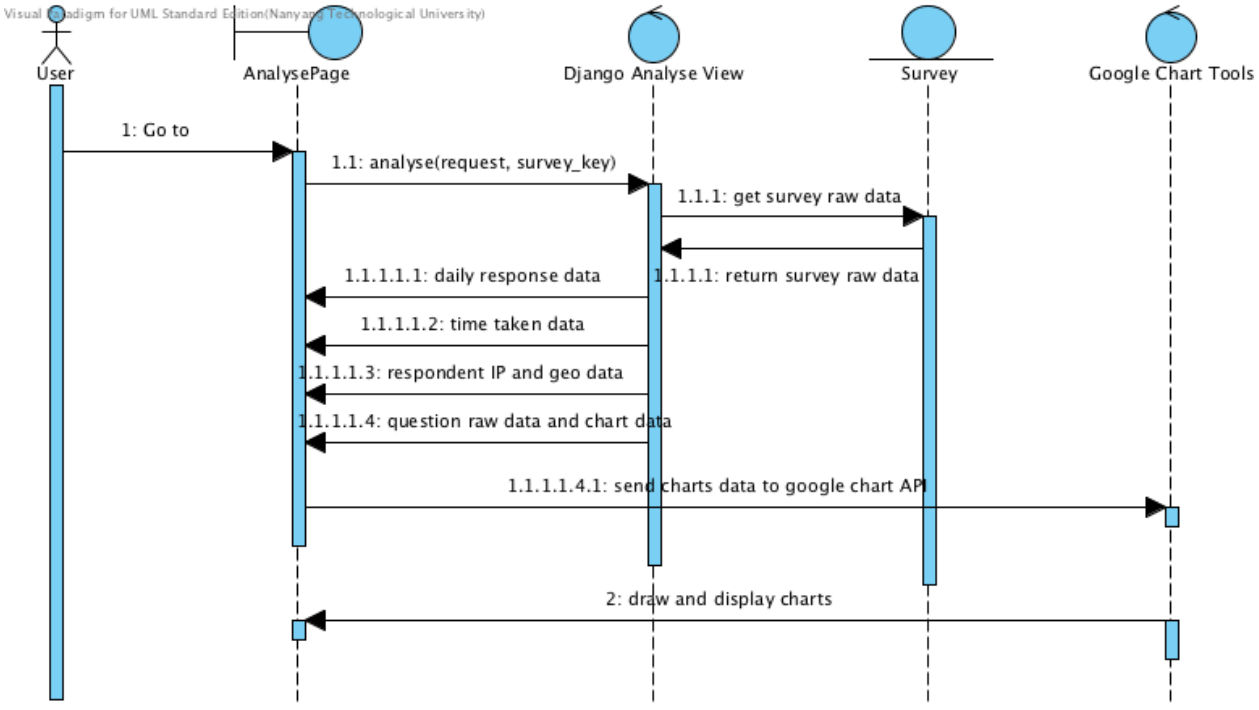
5.3 Edit survey



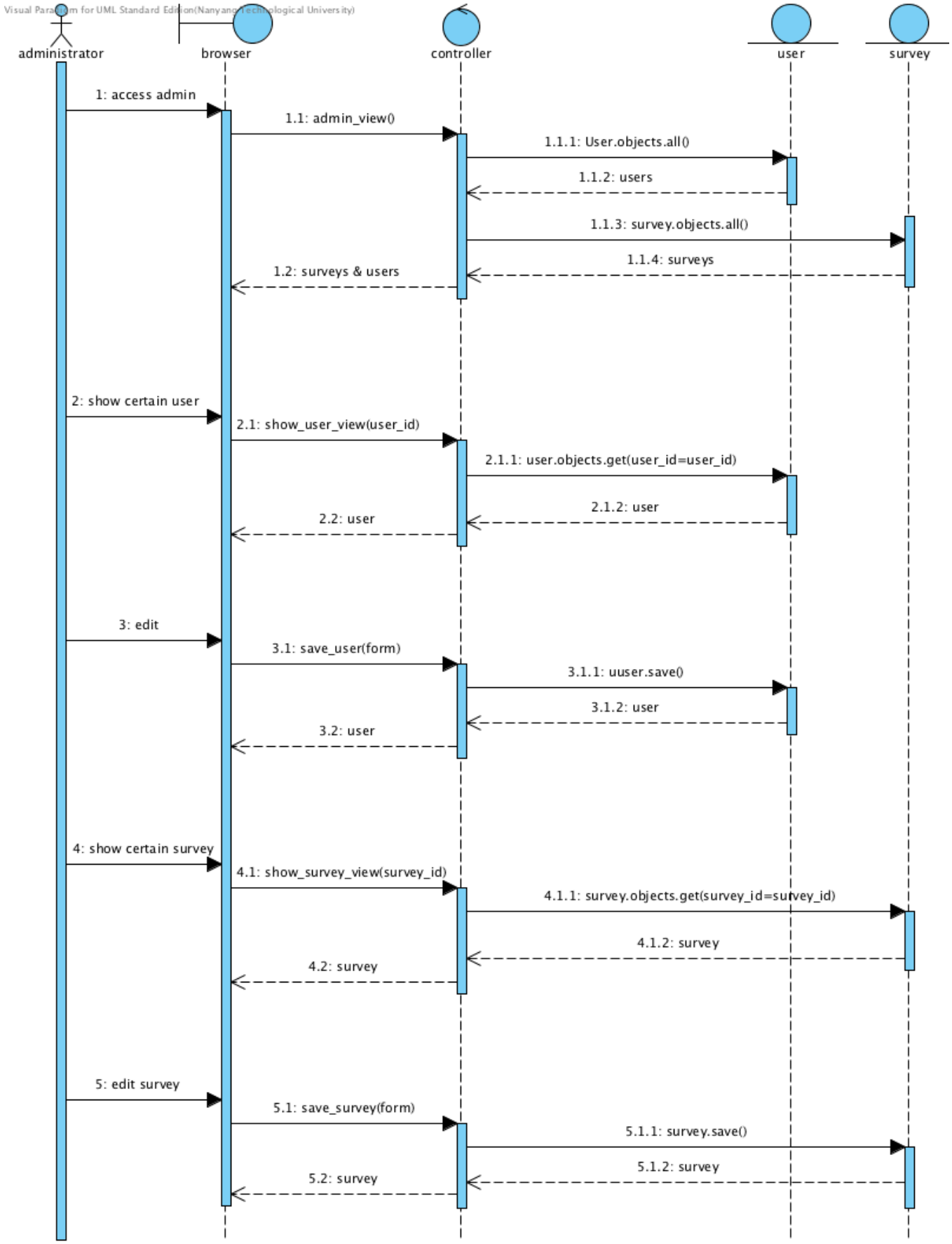
5.4 Take survey



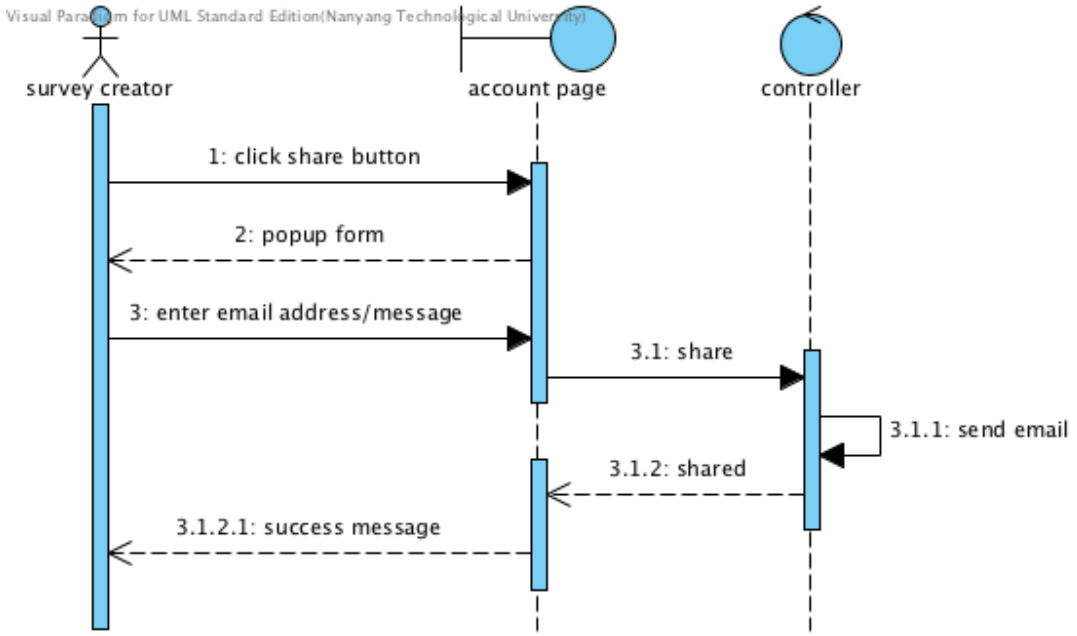
5.5 View survey results



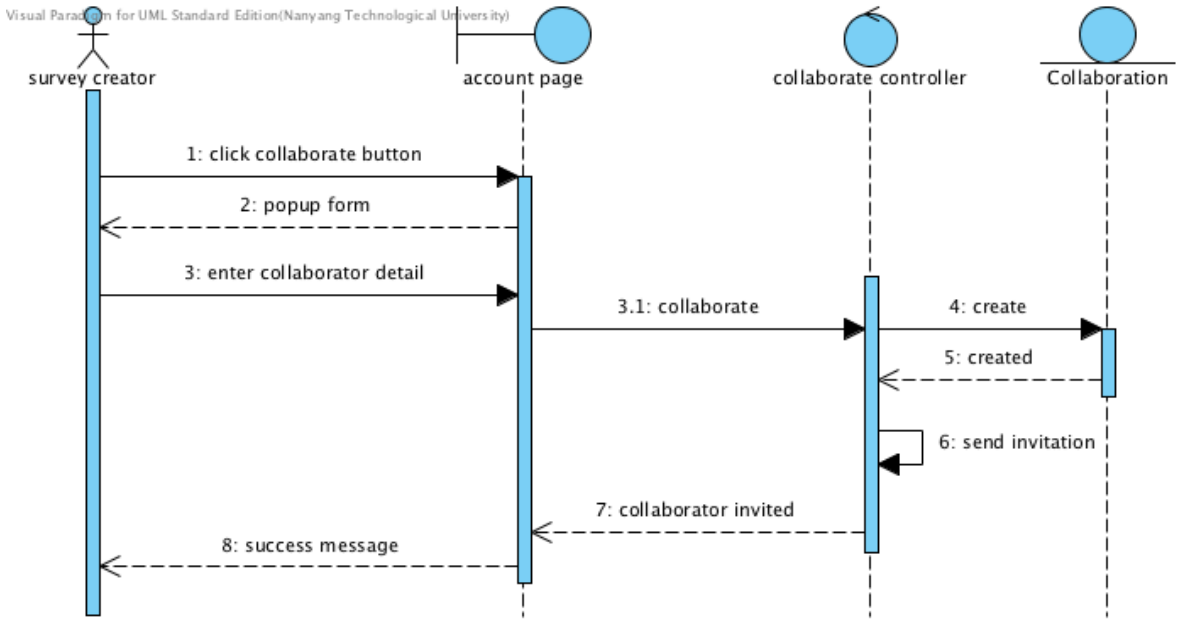
5.6 manage database



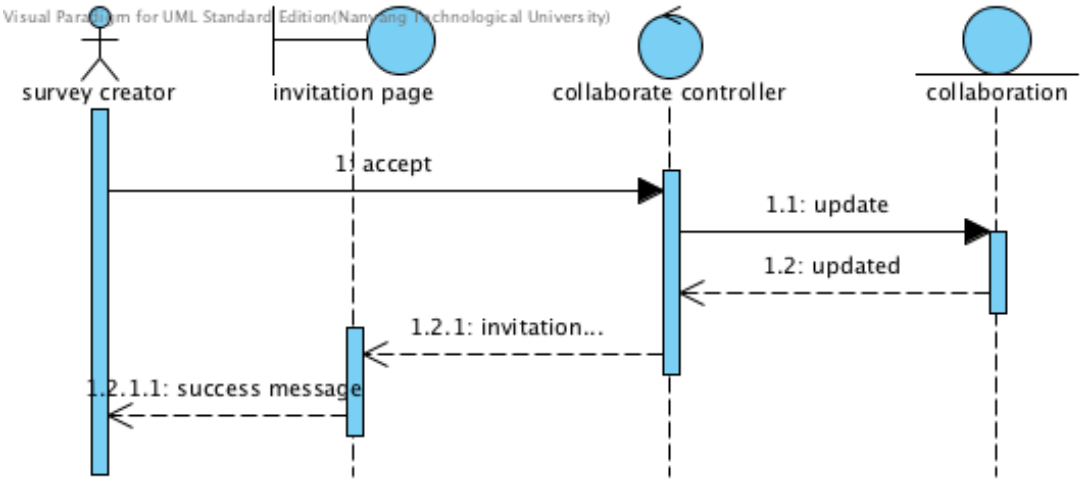
5.7 Share



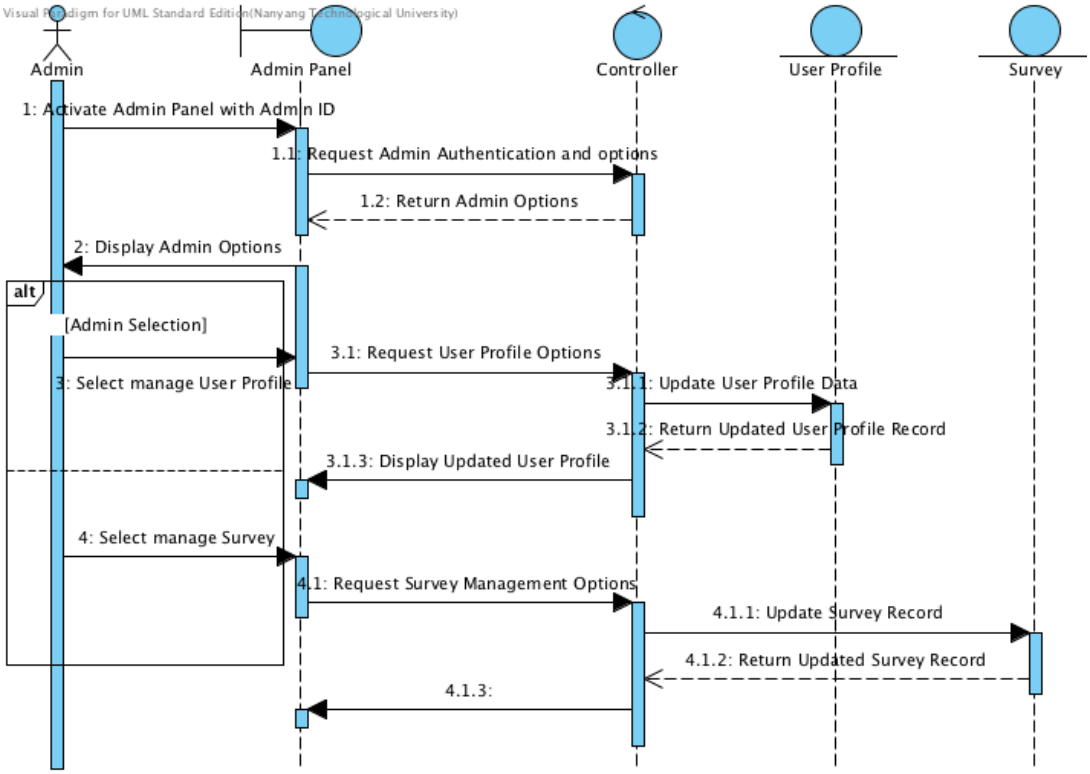
5.8 Collaborate



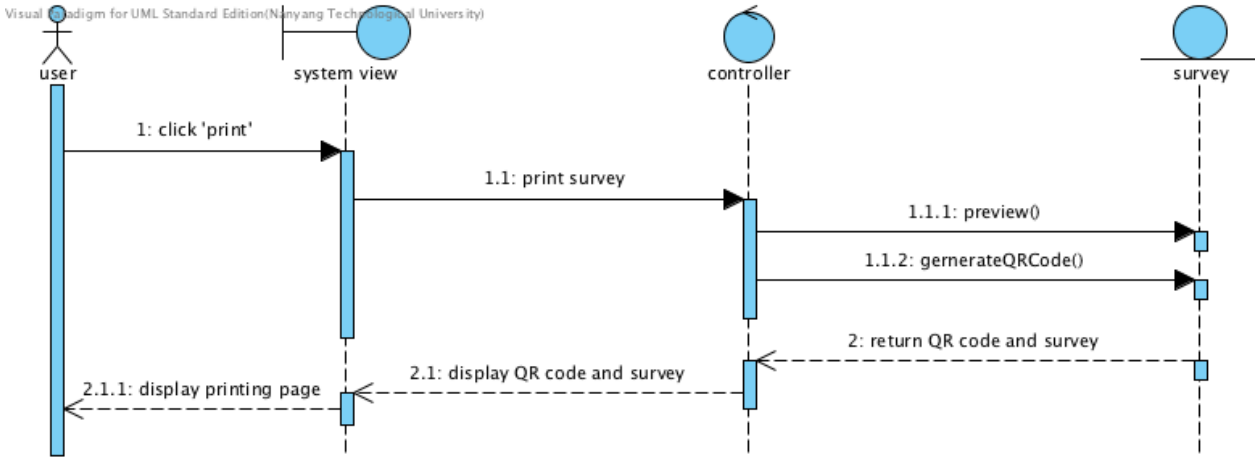
5.9 Accept collaboration



5.10 Admin



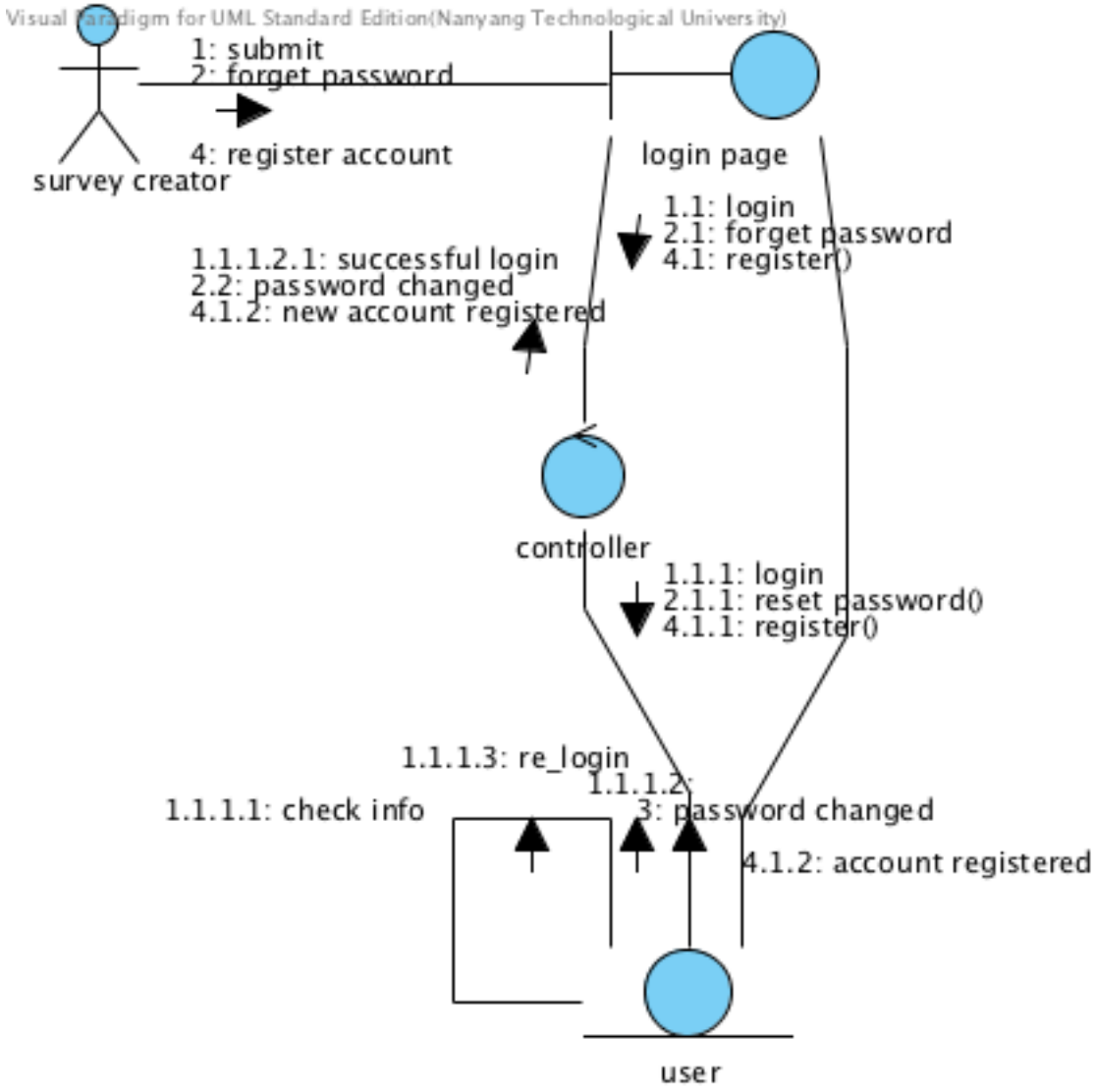
5.11 Print


Visual Paradigm for UML Standard Edition(Nanyang Technological University)
user
system view
controller
survey
1: click 'print'
1.1: print survey
1.1.1: preview()
1.1.2: gernerateQRCode()
2: return QR code and survey
2.1: display QR code and survey
2.1.1: display printing page

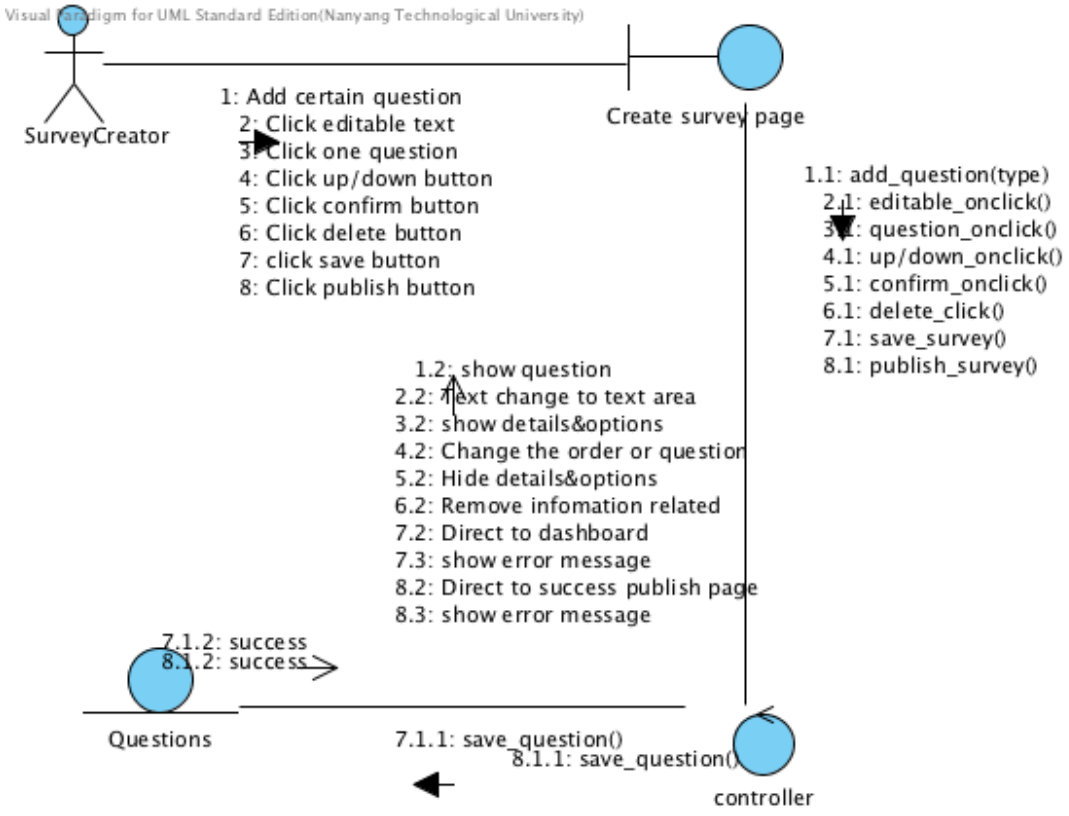
6. Communication diagram

A Communication diagram models the interactions between objects or parts in terms of sequenced messages. Communication diagrams represent a combination of information taken from class, sequence and use case diagram describing both the static structure and dynamic behavior of a system.

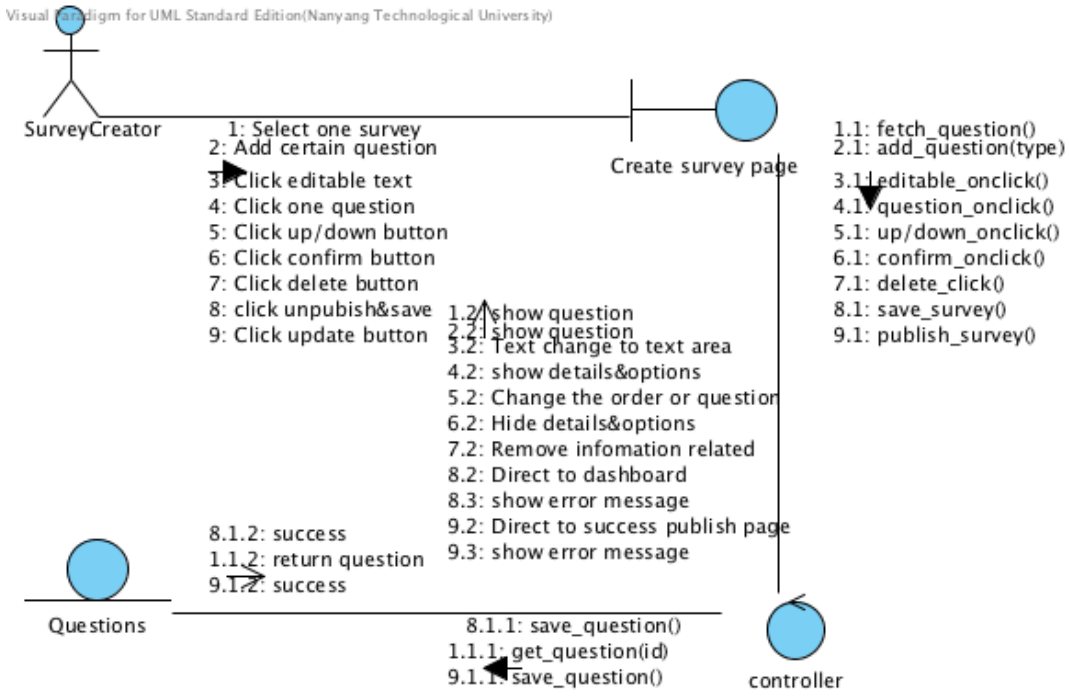
6.1 Register and login



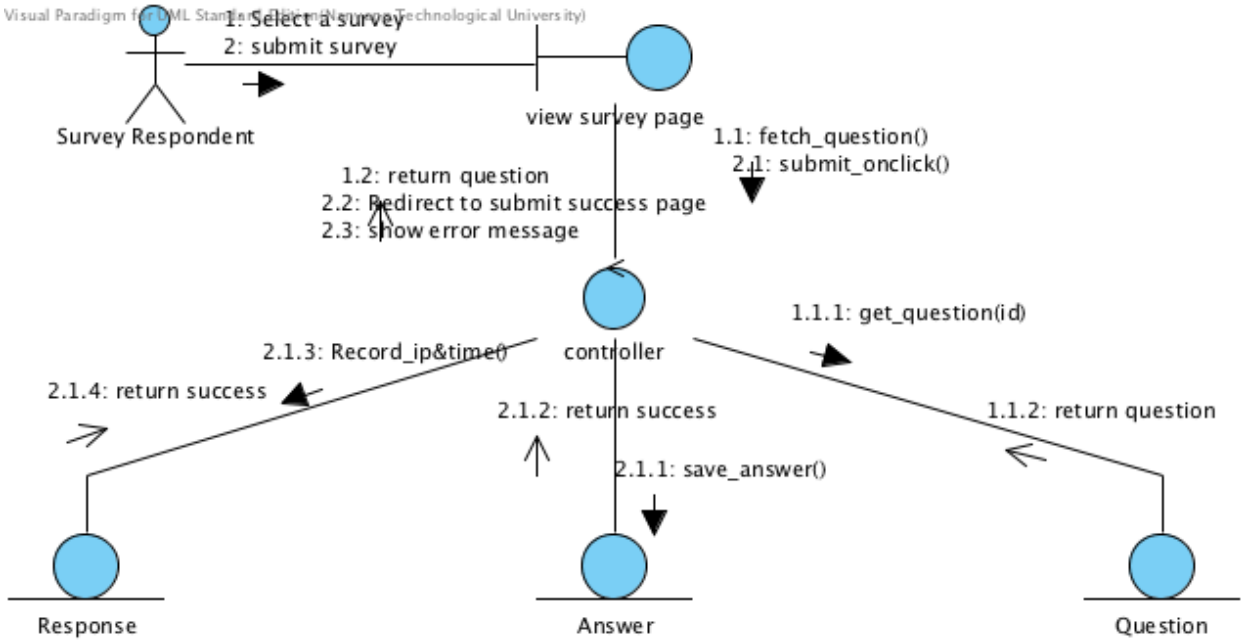
6.2 Create survey



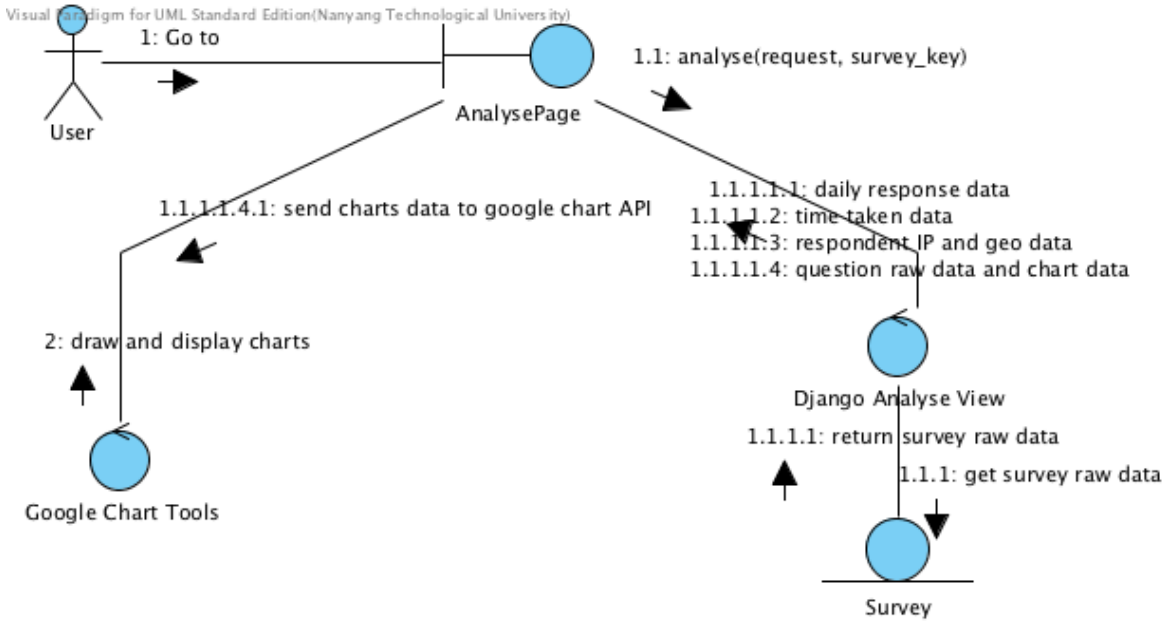
6.3 Edit survey



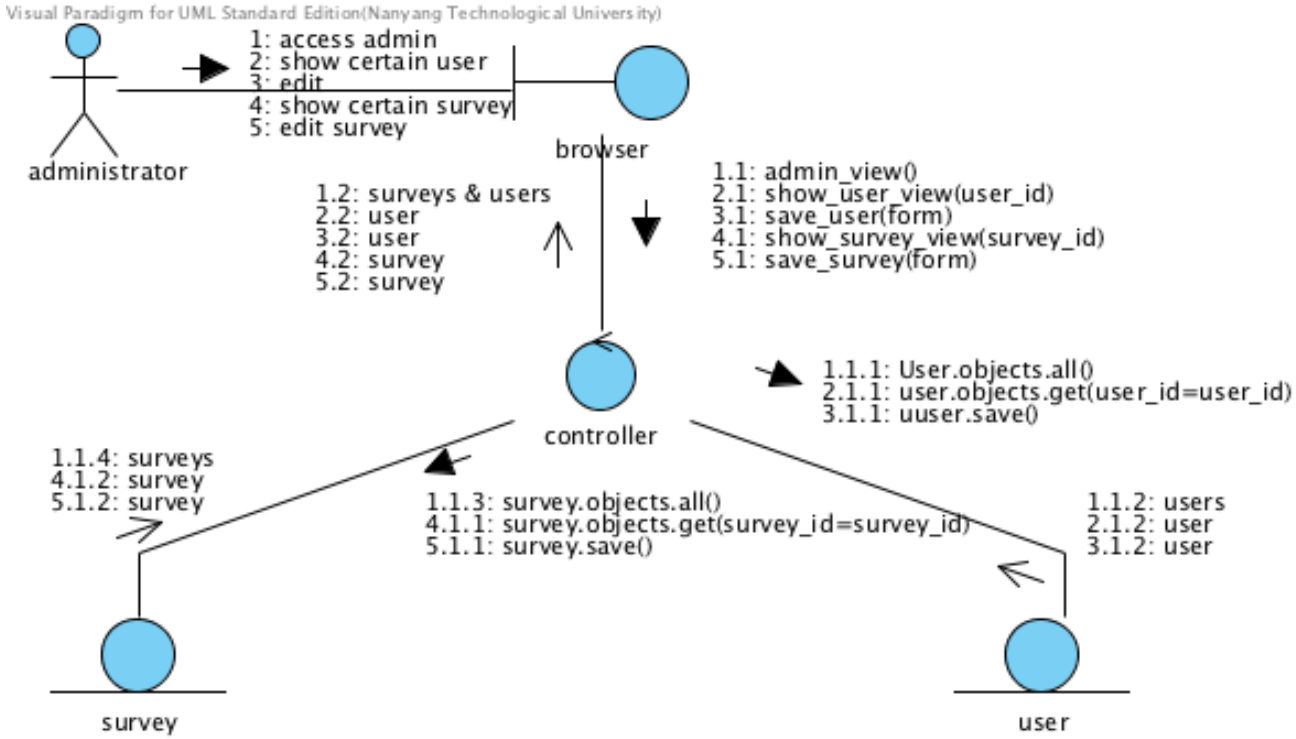
6.4 Take survey



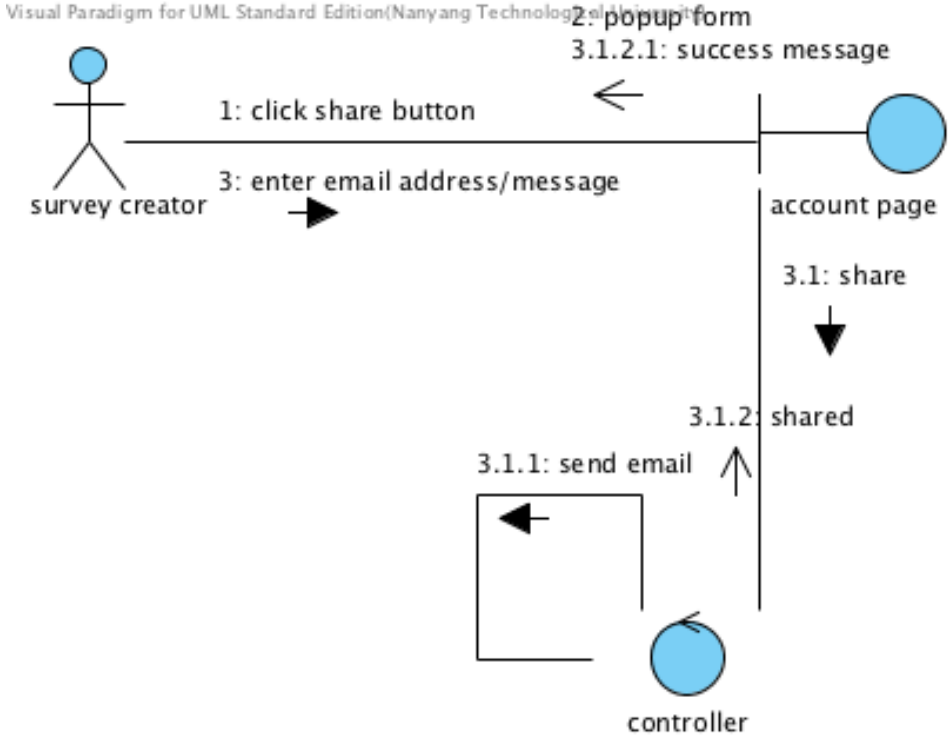
6.5 View results



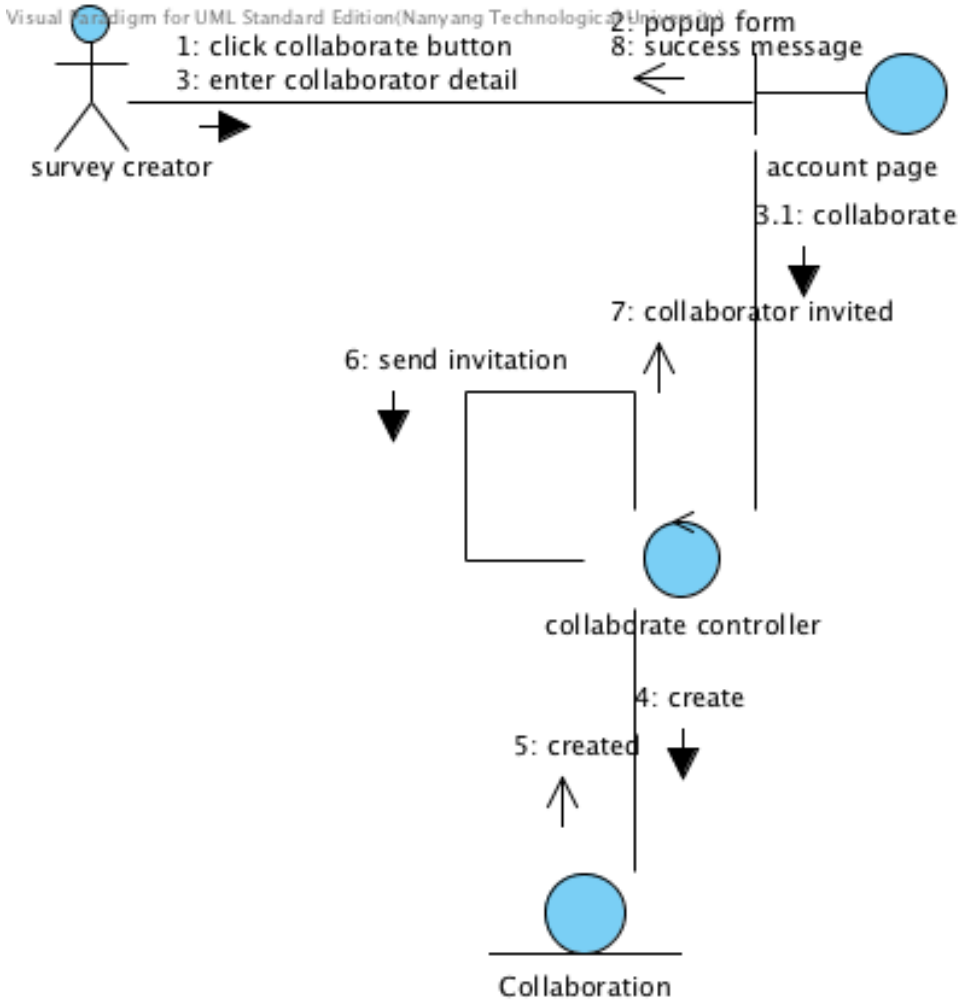
6.6 manage database



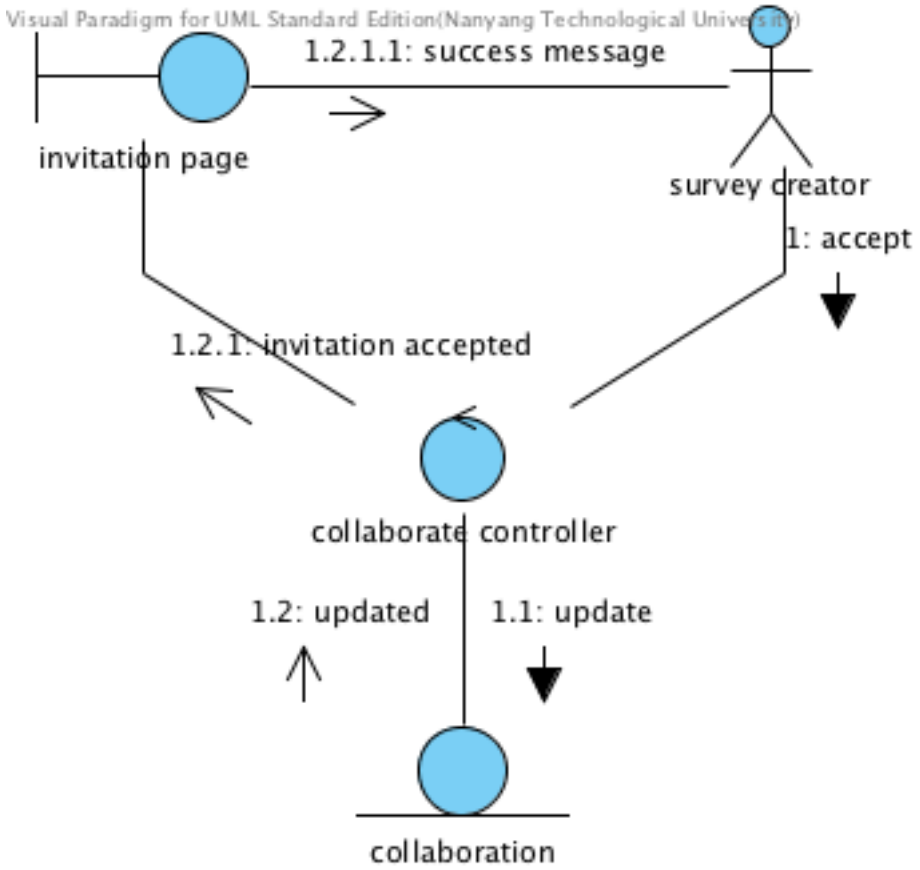
6.7 Share



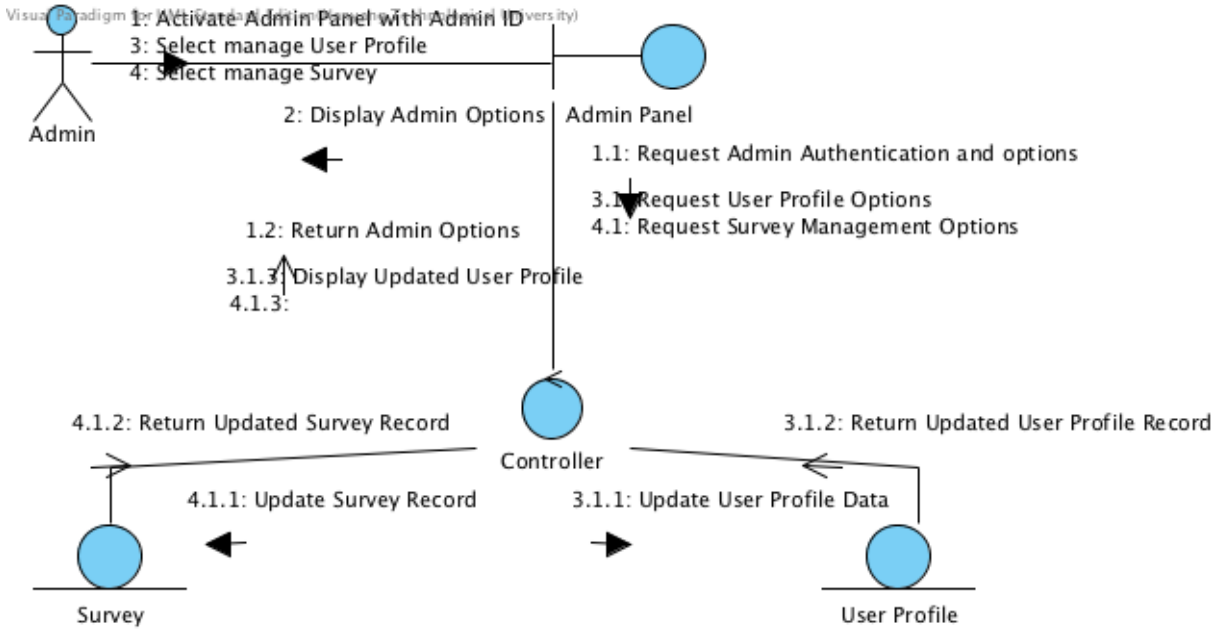
6.8 Collaborate



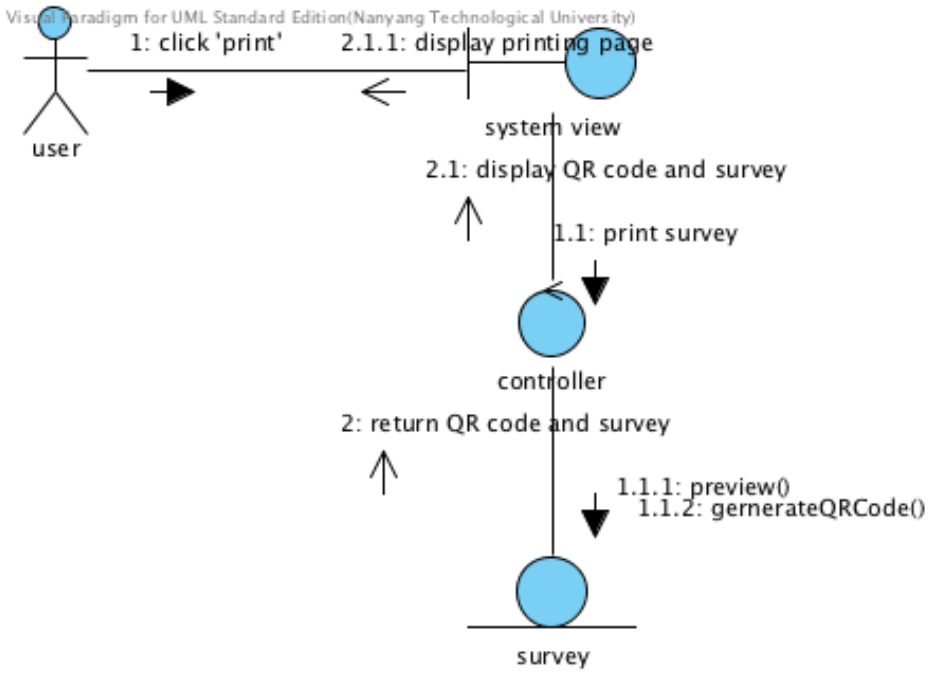
6.9 Accept Collaboration



6.10 Admin



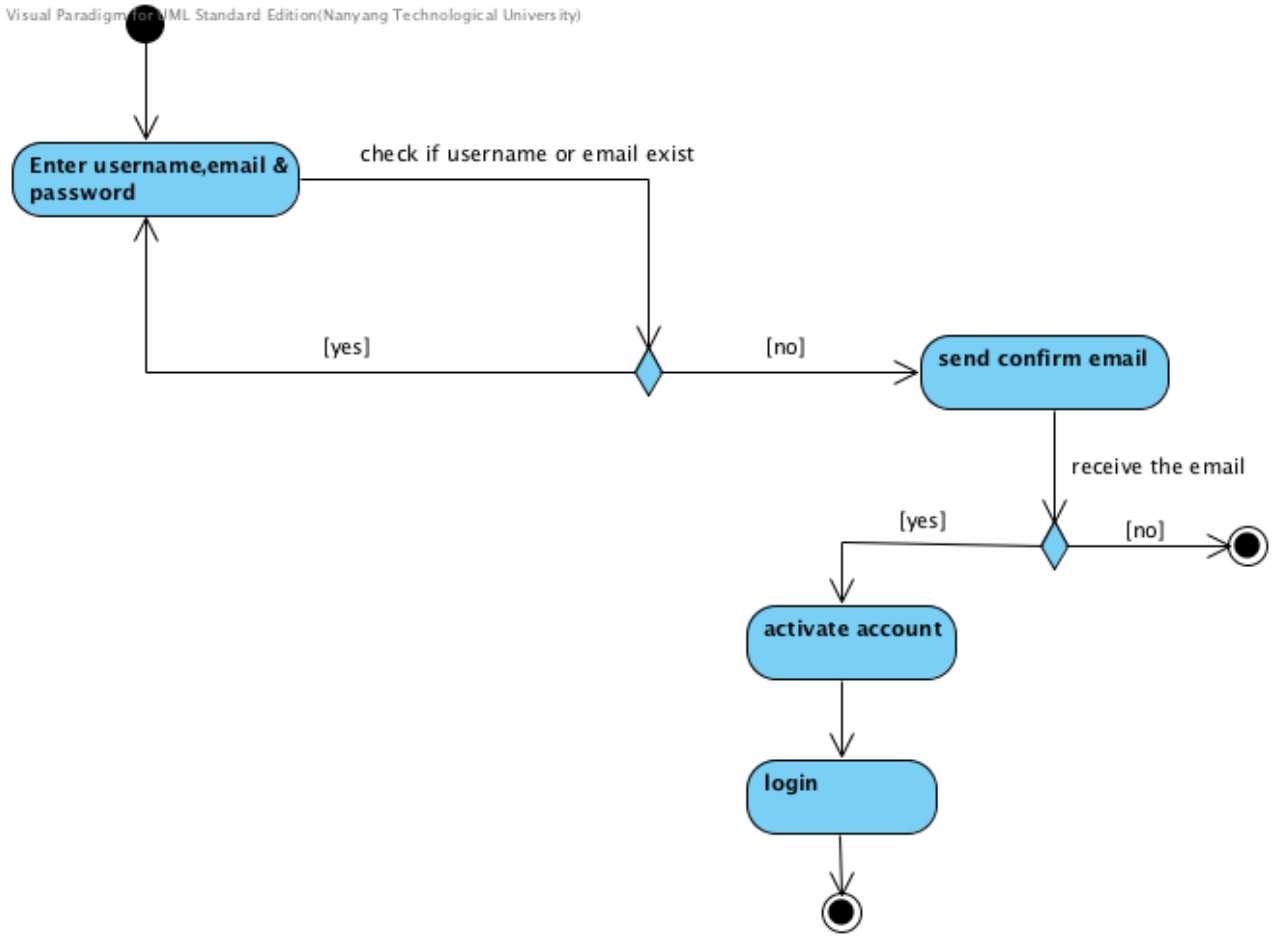
6.11 Print


Visual Paradigm for UML Standard Edition(Nanyang Technological University)
1: click 'print'
2.1.1: display printing page
user
system view
2.1: display QR code and survey
1.1: print survey
controller
2: return QR code and survey
1.1.1: preview()
1.1.2: gernerateQRCode()
survey

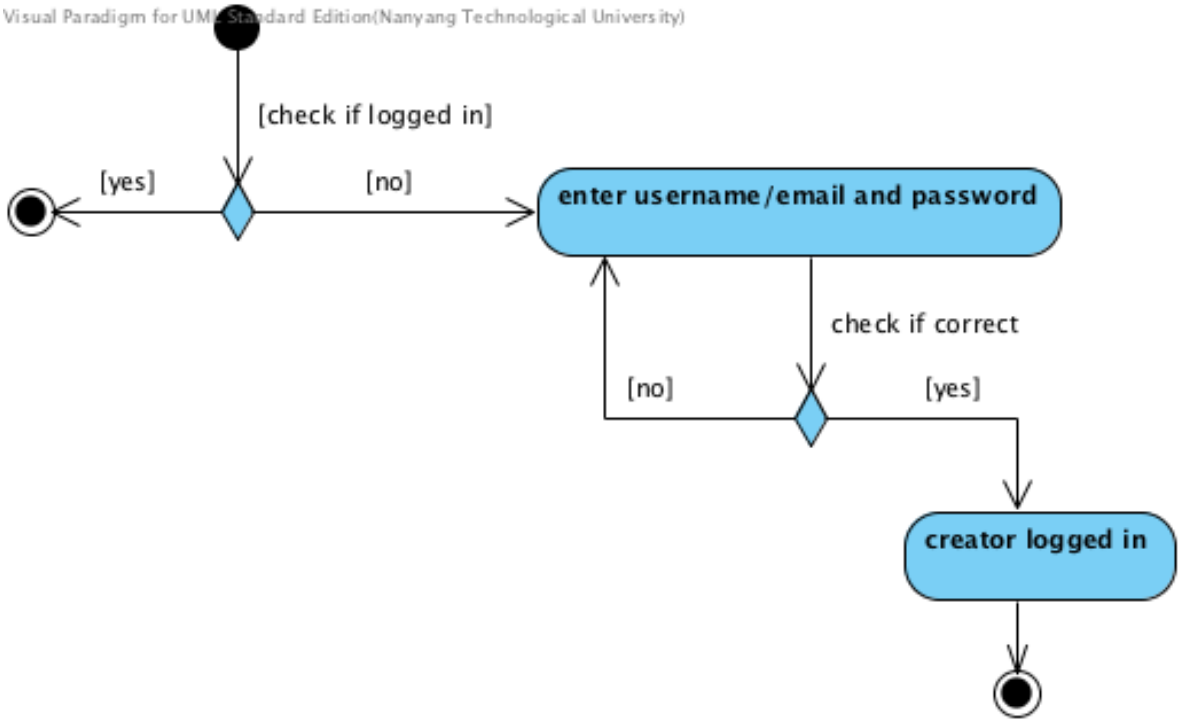
7. Activity Diagrams

Activity diagram is used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control

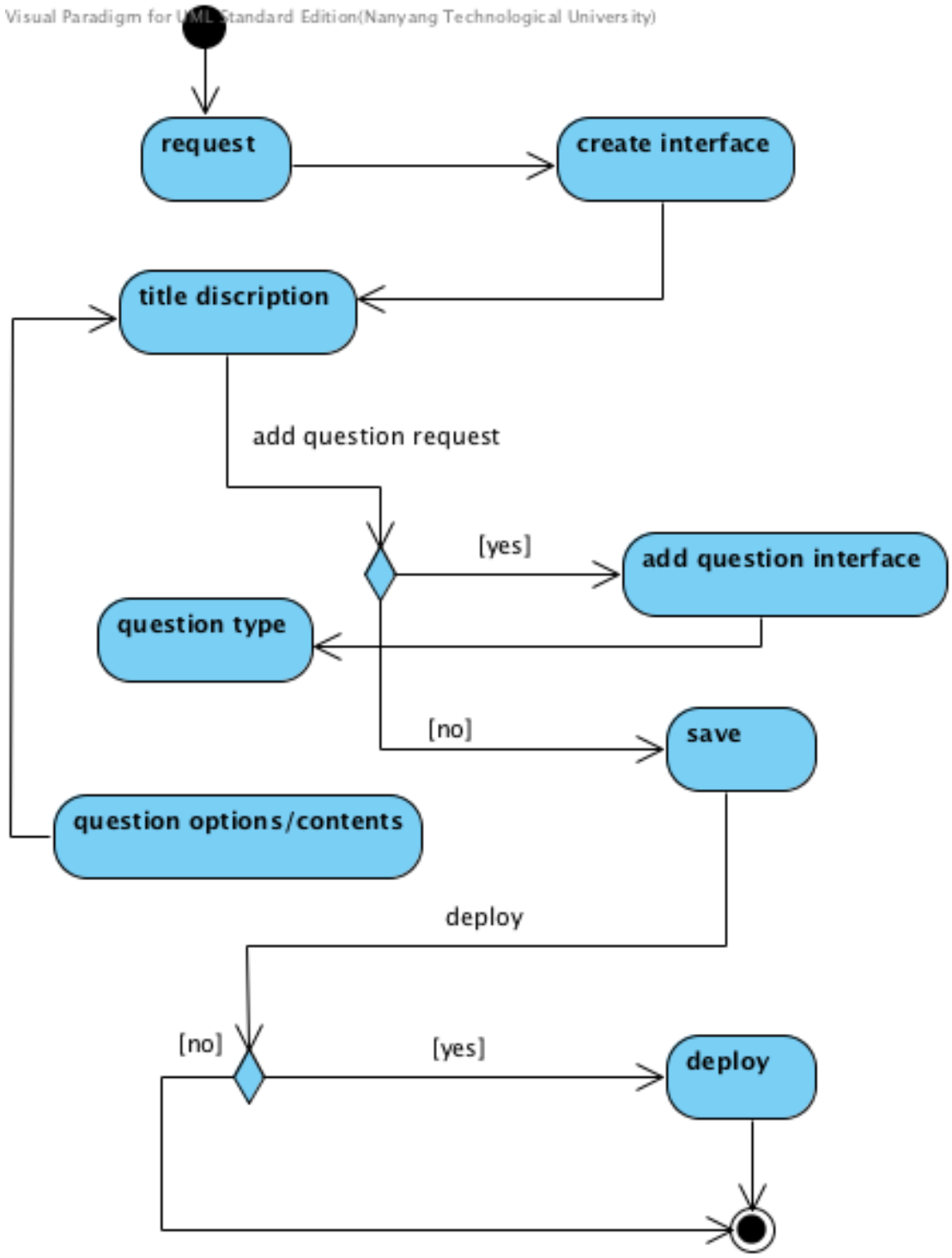
7.1 Register



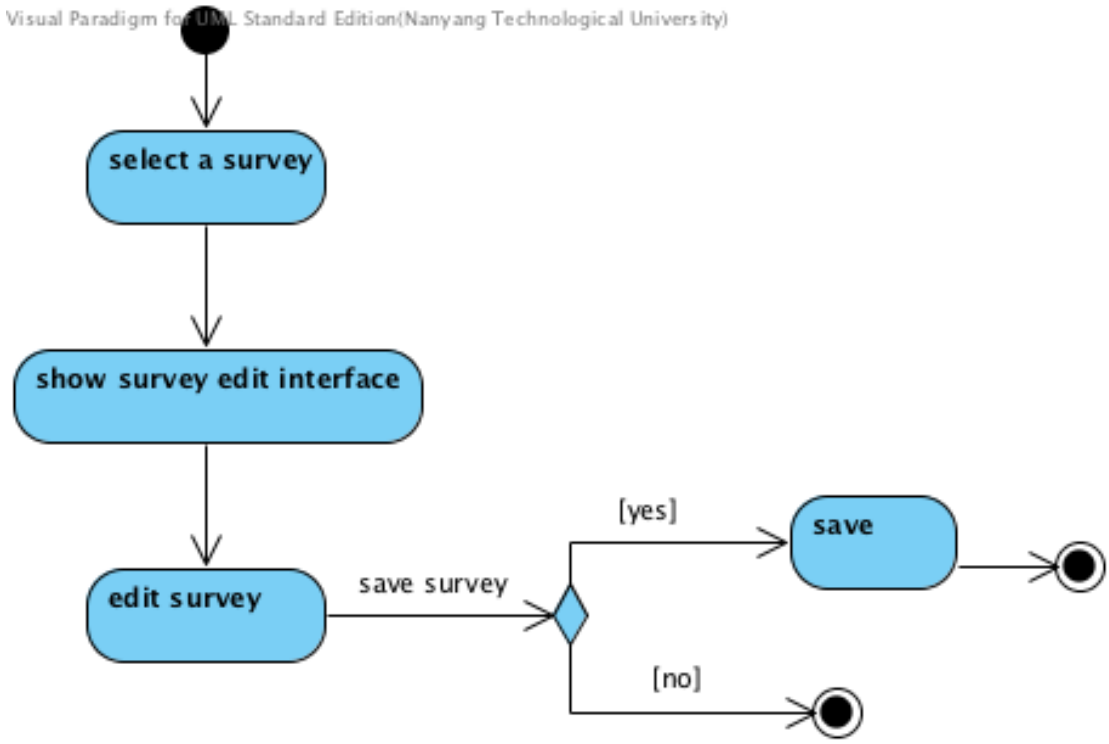
7.2 Log in


Visual Paradigm for UML Standard Edition(Nanyang Technological University)
[check if logged in]
[yes]
[no]
enter username/email and password
check if correct
[no]
[yes]
creator logged in

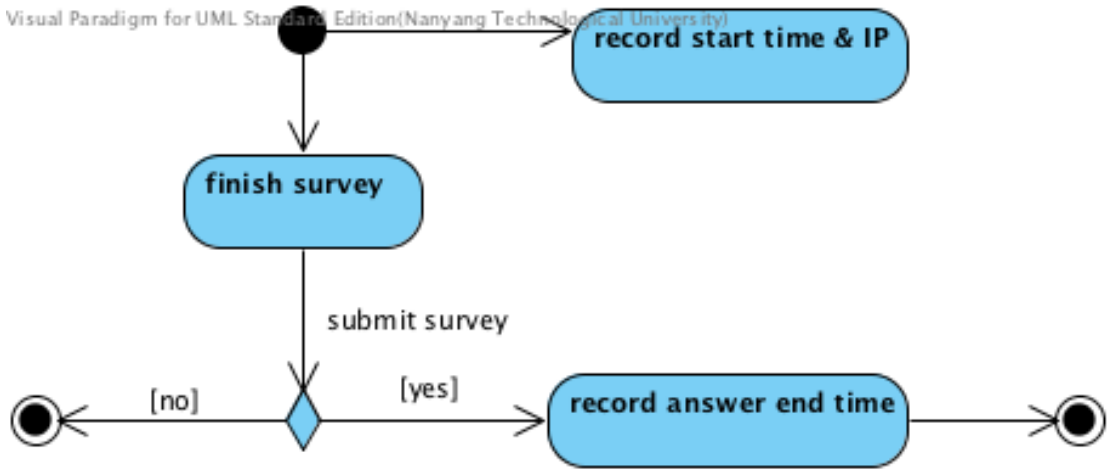
7.3 Create survey



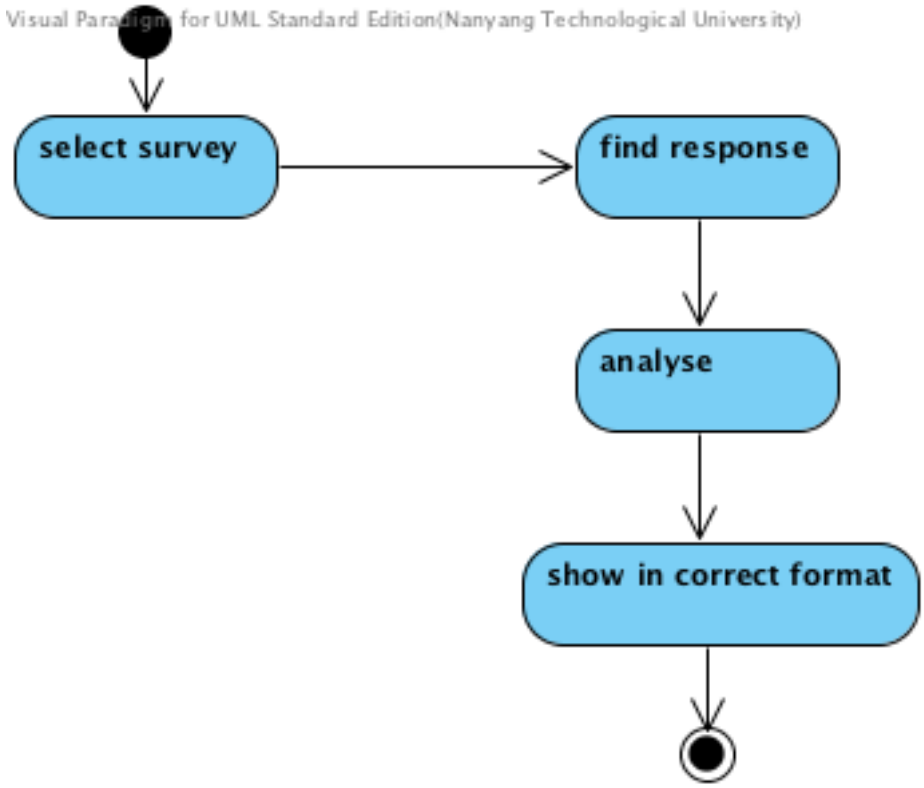
7.4 Edit survey



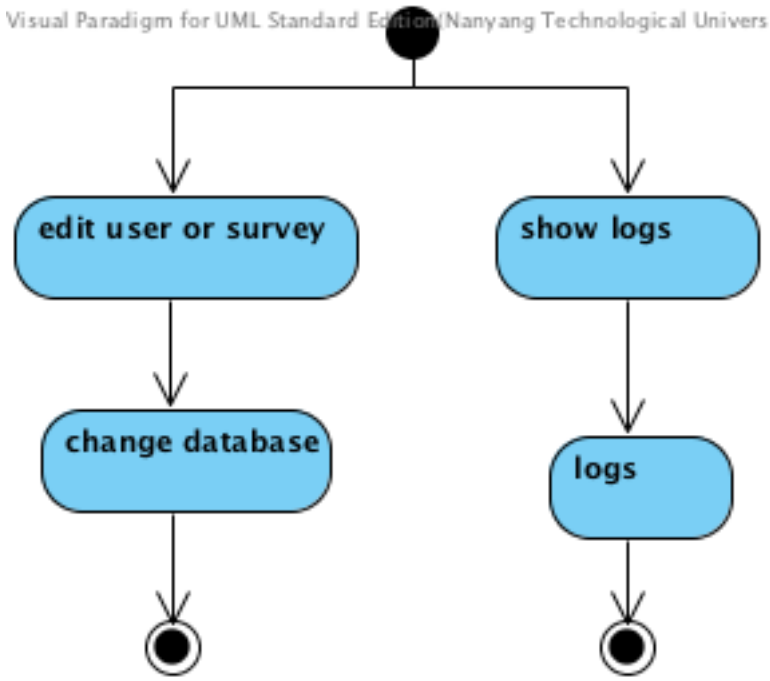
7.5 Take survey



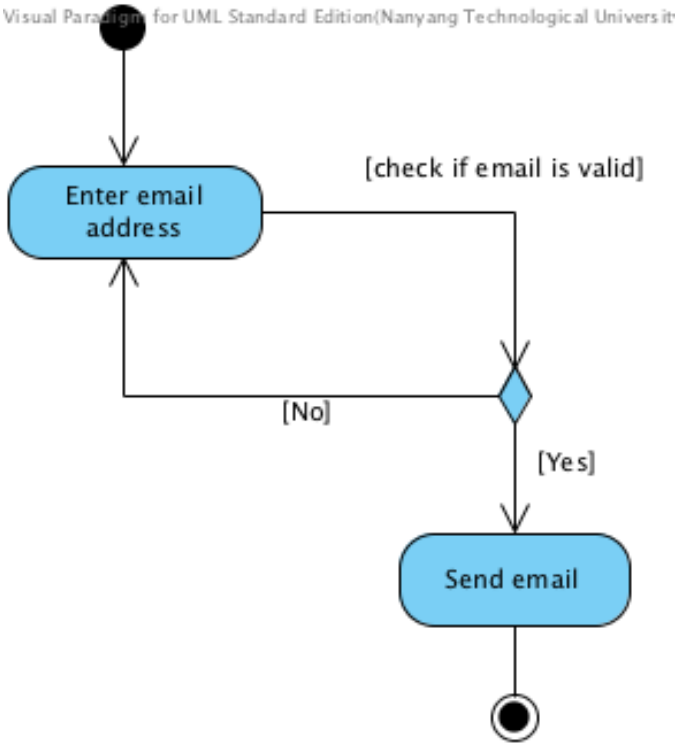
7.6 view results



7.7 Manage database



7.8 Share survey



8. Testing

Both blackbox and whitebox testing are used to test our system. Whitebox is used to do unit testing which will test the individual hardware or software of related units. We use this whitebox testing techniques to veritfy that the code does what is intended to do. Codes are written and testing to ensure the return value is as expected.

Blackbox is used for functional and system testing. Using the blackbox testing techniques, we examine the customer requirements specification and plan the test cases to ensure the functionality is as expected by the developers and most importantly, the customers. Various test cases are examined to make sure the functionality specified in the requirement specification works. We also tested on various system environment to ensure the results are the same for all system within our specification.

During the deployment process, we keep on testing on our functionality, hardware and software. The purpose is to prevent the system from mal-function in case of any changes in our system during the implementation process.

8.1 Blackbox testing

8.1.1 Login

Test ID	Description	Expected Result	Actual result
1	1. key in correct user name 2. key in correct password	Successful login	pass
2	1. key in wrong user name 2. key in correct password	Deny login. Display “Sorry, that's not a valid username or password”	pass
3	1. key in correct user name 2. key in wrong password	Deny login. Display “Sorry, that's not a valid username or password”	pass

8.1.2 Register account

Test ID	Description	Expected Result	Actual result
4	Precondition: user click on “register” button 1. enter the e-mail address 2. enter the password 3. re-type the password 4. click the “confirm button”	New account is registered Activation email sent	pass
4	Precondition: user click on “register” button Precondition: the e-mail is already registered 1. enter the e-mail address 2. enter the password 3. re-type the password 4. click the “confirm button”	Deny message is displayed “account already existed”	pass
6	Precondition: user click on “register” button 1. enter the e-mail address 2. enter the password 3. re-type the password	Deny message is displayed “unmatched password”	pass

	and it is different from the previous password 4. click the “confirm button”		

8.1.3 Create survey

Test ID	Description	Expected Result	Actual result
7	Precondition: user is successfully logged in 1. user choose the question type 2. all question is entered according to the requirement and specified the character limitation 3. click “save” button to save the survey in dashboard 4. click “publish” button to publish the survey	Survey is successfully created and URL is generated	pass

8.1.4 Edit survey

Test ID	Description	Expected Result	Actual result
8	Precondition: user is successfully logged in Precondition: there is a survey to edit	A list of survey is shown to wait for the user to choose and edit	Pass
	1. select a survey to edit	Survey is open and wait for edition	
	2. question is being edited		
	3. click “save” button	Direct to the updated dashboard	
9	Precondition: user is successfully logged in Precondition: there is a survey to edit	A list of survey is shown to wait for the user to choose and edit	pass
	1. select a survey to edit	Survey is open and wait for edition	
	2. no change is being done		
	3. click “save” button	Direct to dashboard	

8.1.5 Take survey

Test ID	Description	Expected Result	Actual result
10	1. open the specific survey 2. answer the survey according to question type requirements 3. click “submit answer” button	Successfully submit and display “Thank you for participating”	pass
11	1. open the specific survey		Pass
	2. answer the text and paragraph questions: exceed the maximum character limit	Warning message “No. of characters can not be exceeded”	Pass
	3. answer multiple, scale, date and time questions: select an option and then change to another option	The original option is deselected and the new option is being selected.	Pass
	4. Answer numeric question: answer entered is not a number	Warning message “Legal degits are required”	Pass
	5. Answer field is empty. click “submit answer” button	Warning message ”This question is required”	Pass

8.1.6 View survey results

Test ID	Description	Expected Result	Actual result
12	Precondition: user is successfully logged in Precondition: the user have already created a survey Precondition: There is already response to the survey	A list of survey is shown	Pass
	1. select a survey to view the result	The survey result page is being opened	

8.1.7 Manage database

Test ID	Description	Expected Result	Actual result
13	Precondition: the admin is logged in 1. select a user from the user list 2. click “delete” button	Message shown “Are you sure you want to delete the user?”	pass
	3. click “Yes” button	The user is successfully deleted	
14	Precondition: the admin is logged in 1. select a user from the user list but there is no user registered	Error message “There is no user to delete”	pass
15	Precondition: the admin is logged in 1. select a survey from the survey list 2. click “delete” button	Message shown “Are you sure you want to delete the survey?”	pass
	3. click “Yes” button	The survey is successfully deleted	
16	Precondition: the admin is logged in 1. select a survey from the survey list but there is no survey created	Error message “There is no survey to delete”	pass

8.1.8 Share

Test ID	Description	Expected Result	Actual result
17	Precondition: the survey creator is logged in 1. Go to dashboard and click on the share button 2. Enter valid email address 3. Click on share button	Message shown “you have successfully share with xxx”	pass

18	Precondition: the survey creator is logged in 1. Go to dashboard and click on the share button 2. Enter a invalid email address 3. Click on share button	Error message “e-mail address is not valid”	pass

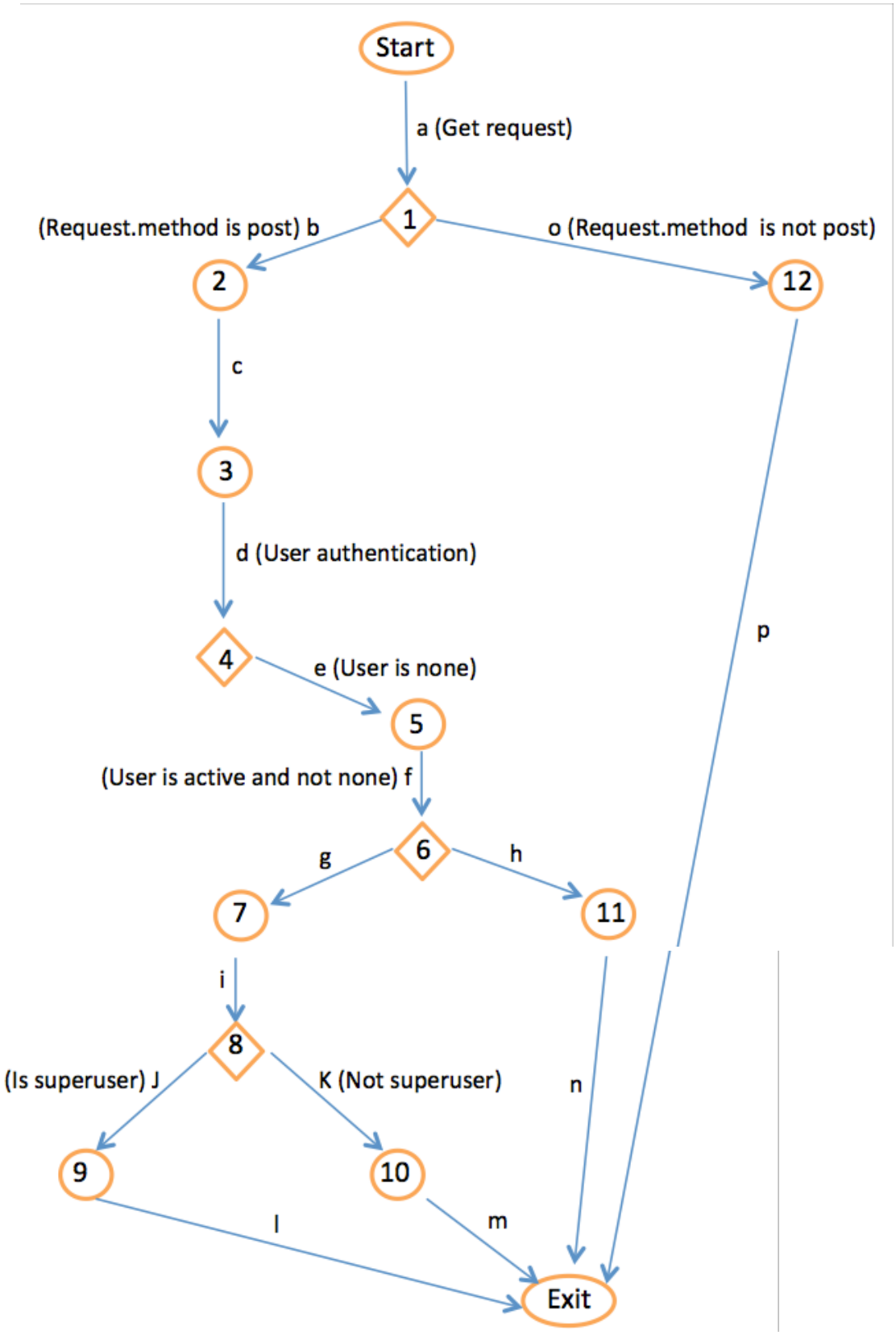
8.1.9 Collaborate

Test ID	Description	Expected Result	Actual result
19	Precondition: the survey creator is logged in 1. Go to dashboard and click on the collaborate button 2. Enter valid email address or username 3. Click on invite button	Message shown “you have successfully invite xxx” as collaborator	pass
20	Precondition: the survey creator is logged in 1. Go to dashboard and click on the collaborate button 2. Enter invalid email address or username 3. Click on invite button	Error message “e-mail address is not valid ” or “user xxx not found”	pass

8.2 Whitebox testing

8.2.1 Login

```
def login_view(request):
    if request.method == 'POST':                                    (1)
        username_or_email = request.POST.get('username_or_email')
        password = request.POST.get('password')                     (2)
        # Login using username or email
        user = auth.authenticate(username=username_or_email, password=password)   (3)
        if user is None:                                            (4)
            user = auth.authenticate(email=username_or_email, password=password)  (5)

        # User must have activated their account through email
        if user is not None and user.is_active:                     (6)
            # Correct password, and the user is marked "active"
            auth.login(request, user)                               (7)
            # Redirect to a success page.
            if user.is_superuser:                                   (8)

                return HttpResponseRedirect("/admin")               (9)
            else:
                return HttpResponseRedirect("/account")             (10)
        else:
            # Show an error page
            return render_to_response("account/login.html", {'error': True},   (11)
context_instance=RequestContext(request))
    else:
        return render_to_response("account/login.html", {}, context_instance=RequestContext(request))   (12)
```


Start
a (Get request)
1
(Request.method is post) b
o (Request.method is not post)
2
12
c
3
d (User authentication)
4
e (User is none)
5
(User is active and not none) f
6
g
h
7
11
i
8
(Is superuser) J
K (Not superuser)
n
9
10
l
m
Exit
p

4 Independent Test Paths

Path1: a-b-c-d-e-f-g-i-j-l

Path2: a-b-c-d-e-f-g-i-k-m

Path3: a-b-c-d-e-f-h-n

Path4: a-o-p

Path1

Input an existing and active Admin account

Username: ray Password: ray007

Expected result:

Redirect to admin page

Path2

Input and an existing and active normal user account

Username: Xiao Ming Password: 1234

Expected result:

Redirect to account page

Path3

Input a non-existing or inactive user account

Username: Da Ming Password: 4321

Expected result:

Redirect to Login page

Path4

Change form method to ’get’ in account\login.html page

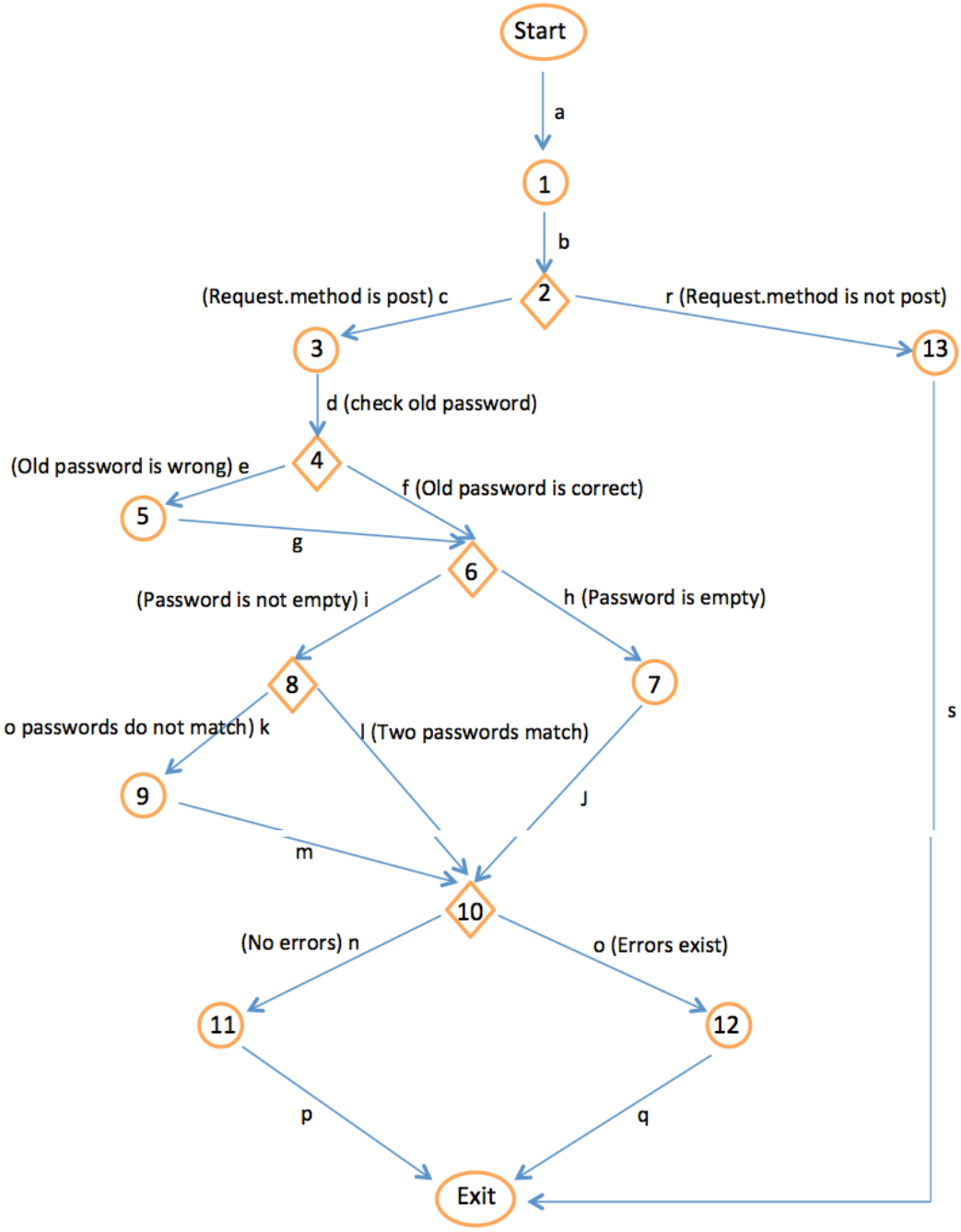
Input valid username and password

Expected result:

Redirect to Login page

8.2.2 Change password

```
def change_password_view(request):
    errors = []                                                    1
    if request.method == 'POST':                                   2
        old_password = request.POST.get('old_password')
        password1 = request.POST.get('password1')                  3
        password2 = request.POST.get('password2')

        if not request.user.check_password(old_password):          4
            errors.append("Your old password is wrong.")           5

        if len(password1) == 0:                                    6
            errors.append("New password field cannot be empty.")   7
        elif password1 != password2:                               8
            errors.append("The two password fields didn't match.") 9

        if len(errors) > 0:                                        10
            return render_to_response("account/change_password.html", {'errors': errors},
            context_instance=RequestContext(request))              11
    else:
        request.user.set_password(password1)
        request.user.save()
        messages.success(request, "You have successfully changed your password")
        return HttpResponseRedirect("/")                           12
else:
    return render_to_response("account/change_password.html", {'errors': errors},
        context_instance=RequestContext(request))                  13
```


Start
a
1
b
2
(Request.method is post) c
r (Request.method is not post)
3
13
d (check old password)
4
(Old password is wrong) e
f (Old password is correct)
5
g
6
(Password is not empty) i
h (Password is empty)
8
7
o passwords do not match) k
l (Two passwords match)
s
9
J
m
10
(No errors) n
o (Errors exist)
11
12
p
q
Exit

8 Independent Test Paths

Path1: a-b-c-d-e-g-i-k-m-o-q

Path2: a-b-c-d-e-g-i-l-o-q

Path3: a-b-c-d-e-g-h-j-o-q

Path4: a-b-c-d-f-h-j-o-q

Path5: a-b-c-d-f-i-k-m-o-q

Path6: a-b-c-d-f-i-l-n-p

Path7: a-b-r-s

The current legitimate user's username is 'User1' and password is '12345'

Path1

Inputs:

Old password: 11111

New password: 67890

Confirm password: 00000

Expected result:

Error messages are:

"Your old password is wrong."

"The two password fields didn't match."

Path2

Inputs:

Old password: 22222

New password: 00000

Confirm password: 00000

Expected result:

Error message is:

"Your old password is wrong."

Path3

Inputs:

Old password: 33333

New password: *empty*

Confirm password: *empty*

Expected result:

Error message is:

"Your old password is wrong."

"New password field cannot be empty."

Path4

Inputs:

Old password: 12345

New password: *empty*

Confirm password: *empty*

Expected result:

Error message is:

"New password field cannot be empty."

Path5

Inputs:

Old password: 12345

New password: *abcdwww*

Confirm password: *abcdyyy*

Expected result:

Error message is:

"The two password fields didn't match."

Path6

Inputs:

Old password: 12345

New password: *abcdef*

Confirm password: *abcdef*

Expected result:

Successful message is:

"You have successfully changed your password"

Path7

Inputs:

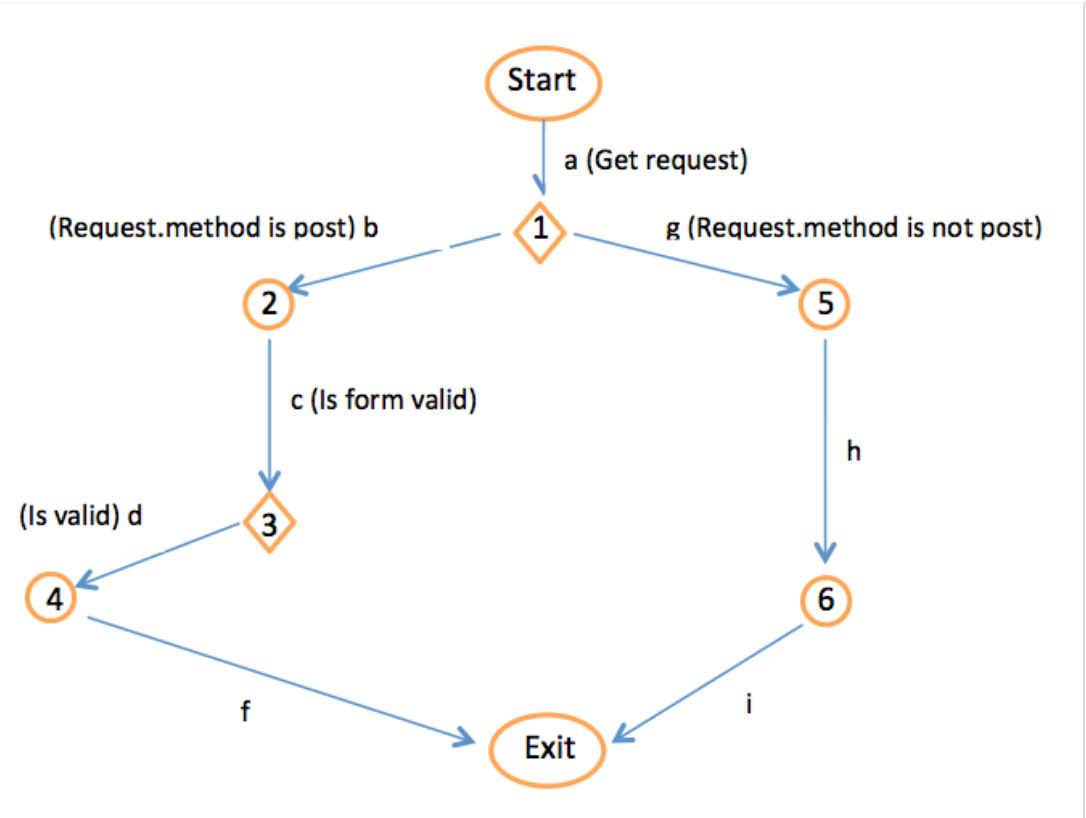
Change form request.method to ‘get’

Expected result:

Direct to "account/change_password.html" page

8.2.3 register

```
def register(request):
    if request.method == 'POST':                                          1
        form = RegistrationForm(request.POST)                             2

        if form.is_valid():                                               3
            new_user = form.save()
            new_user.is_active = False
            new_user.save()

            confirmation_code = sha.new(new_user.username).hexdigest()
            profile = UserProfile(user=new_user, confirmation_code=confirmation_code[0:9])
            profile.save()

            send_registration_confirmation(new_user)

            return render_to_response("account/registration_complete.html")   4
    else:
        form = RegistrationForm()                                         5
    return render_to_response("account/register.html", {'form': form},
context_instance=RequestContext(request))                                 6
```



2 Independent Test Paths

Path1: a-b-c-d-f

Path2: a-g-h-i

Path1

Inputs:

A valid username (not existing in the database): NewUser1

A valid password: newpasswrd

Expected result:

Redirect to account/registration_complete.html page

Path2

Inputs:

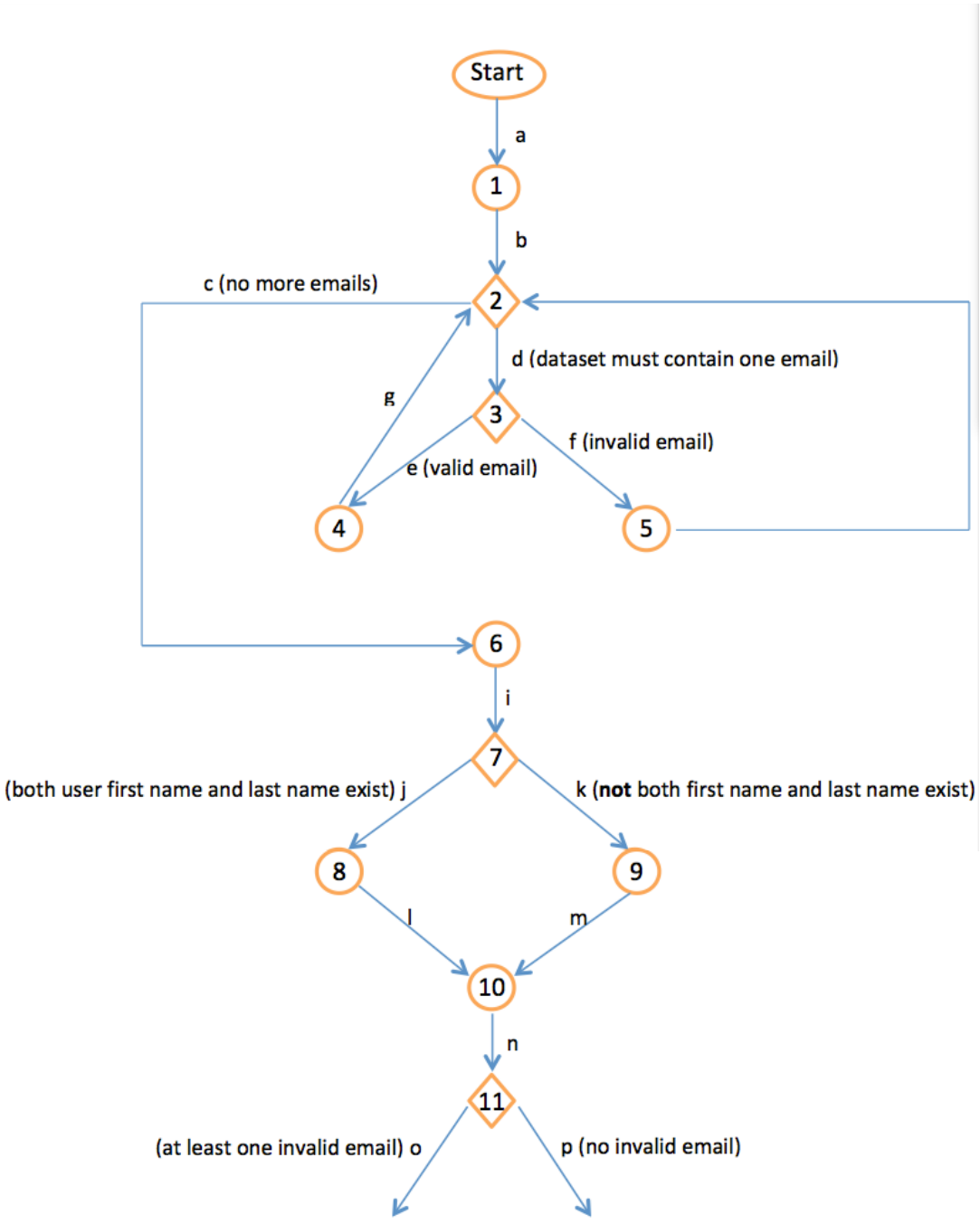
Change form request.method to ‘get’

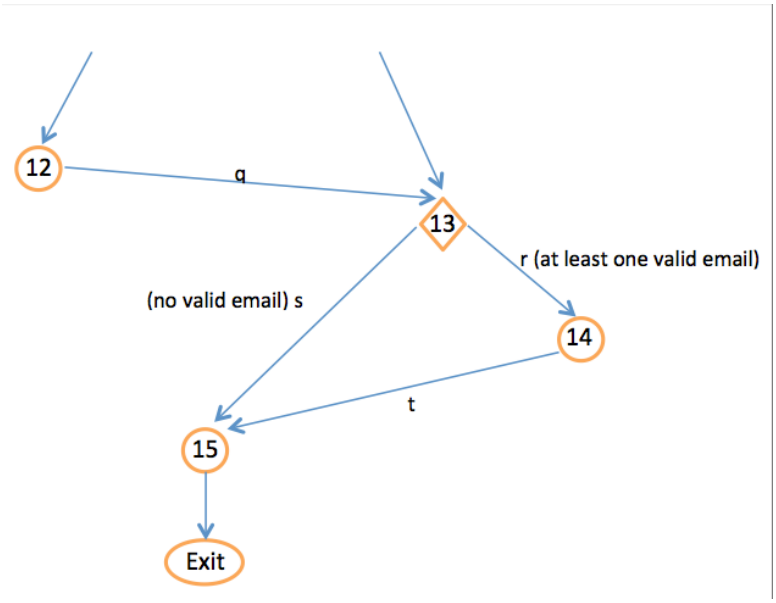
Expected result:

Redirect to account/register.html page

8.2.4 Share

```
def share_survey(request):
    survey_id = request.POST.get('survey_id');
    email_data = request.POST.get('collaborators');
    owner_message = request.POST.get('owner_message');                    (1)
    success_emails = []
    fail_emails = []
    emails = [e.strip() for e in email_data.split(',')]

    for e in emails:                                                      (2)
        if is_valid_email(e):                                             (3)
            success_emails.append(e)                                      (4)
        else:
            fail_emails.append(e)                                         (5)

    survey = Survey.objects.get(id=survey_id)                             (6)
    survey_url = "http://%s%s" % (get_current_site(request).domain, survey.get_absolute_url())

    if request.user.first_name and request.user.last_name:                (7)
        message = "%s %s (%s) invites you to take the survey.\n\nFollow this link to open the
survey:\n%s\n\n%s" % (request.user.first_name, request.user.last_name, request.user.email, survey_url,
owner_message)                                                            (8)
    else:
        message = "%s (%s) invites you to take the survey.\n\nFollow this link to open the
survey:\n%s\n\n%s" % (request.user.username, request.user.email, survey_url, owner_message)   (9)

    send_mail('You are invited to do a survey', message, "noreply@%s" % get_current_site(request).domain,
success_emails)                                                           (10)

    if len(fail_emails) > 0:                                              (11)
        messages.error(request, "Email %s is invalid" % (', ').join(fail_emails))   (12)
    if len(success_emails) > 0:                                           (13)
        messages.success(request, 'You have successfully share with %s' % (', ').join(success_emails))   (14)

    return HttpResponseRedirect("/account")                               (15)
```


Start
a
1
b
c (no more emails)
2
d (dataset must contain one email)
g
3
f (invalid email)
e (valid email)
4
5
6
i
7
(both user first name and last name exist) j
k (not both first name and last name exist)
8
9
l
m
10
n
11
(at least one invalid email) o
p (no invalid email)



6 different paths

Path1: a-b-d-e-g-c-i-j-l-n-p-r-t

Path2: a-b-d-e-g-c-i-k-m-n-p-r-t

Path3: a-b-d-f-h-c-i-j-l-n-o-q-s

Path4: a-b-d-f-h-c-i-k-m-n-o-q-s

Path5: a-b-d-e-g-d-f-h-c-i-j-l-n-o-q-r-t

Path6: a-b-d-e-g-d-f-h-c-i-k-m-n-o-q-r-t

Path1

Inputs:

Current user's username: KBMamba

Email: 7rayrecovery@gmail.com

First name: Kobe, last name: Bryant

Emails entered for sharing: whuang4@e.ntu.edu.sg

Expected result:

1. Redirect to account page, displaying the message: 'You have successfully share with whuang4@e.ntu.edu.sg'
2. An email has been sent to whuang4@e.ntu.edu.sg with the content: 'Kobe Bryant (7rayrecovery@gmail.com)invites you to take the survey'

Path2

Inputs:

Current user's username: KBMamba

Email: 7rayrecovery@gmail.com

First name: *empty*, last name: *empty*

Emails entered for sharing: whuang4@e.ntu.edu.sg

Expected result:

1. Redirect to account page, displaying the message: 'You have successfully share with whuang4@e.ntu.edu.sg'
2. An email has been sent to whuang4@e.ntu.edu.sg with the content: 'KBMamba (7rayrecovery@gmail.com)invites you to take the survey'

Path3

Inputs:

Current user's username: KBMamba

Email: 7rayrecovery@gmail.com

First name: Kobe, last name: Bryant

Emails entered for sharing: abcd

Expected result:

Redirect to account page, displaying the message: 'Email abcd is invalid'

Path4

Inputs:

Current user's username: KBMamba

Email: 7rayrecovery@gmail.com

First name: *empty*, last name: *empty*

Emails entered for sharing: efgh

Expected result:

Redirect to account page, displaying the message: 'Email efgh is invalid'

Path5

Inputs:

Current user's username: KBMamba

Email: 7rayrecovery@gmail.com

First name: Kobe, last name: Bryant

Emails entered for sharing: whuang4@e.ntu.edu.sg, aaaa@bbbb

Expected result:

1. Redirect to account page, displaying the message:
   'Email aaaa@bbbb is invalid' in red
   'You have successfully share with whuang4@e.ntu.edu.sg' in green
2. An email has been sent to whuang4@e.ntu.edu.sg with the content:
   'Kobe Bryant (7rayrecovery@gmail.com) invites you to take the survey'

Path6

Inputs:

Current user's username: KBMamba

Email: 7rayrecovery@gmail.com

First name: *empty*, last name: *empty*

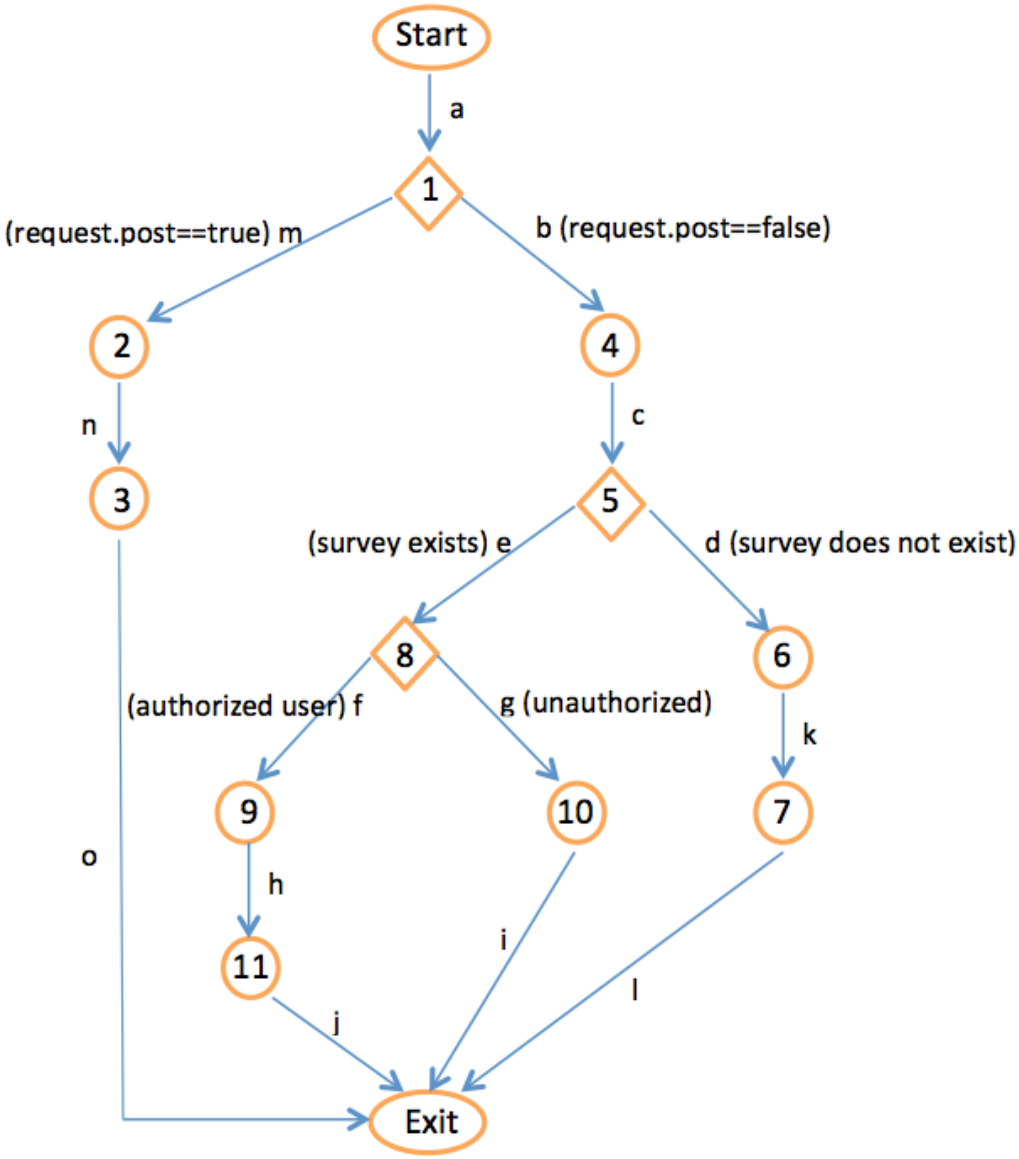
Emails entered for sharing: whuang4@e.ntu.edu.sg, aaaa@bbbb

Expected result:

1. Redirect to account page, displaying the message:
   'Email aaaa@bbbb is invalid' in red
   'You have successfully share with whuang4@e.ntu.edu.sg' in green
2. An email has been sent to whuang4@e.ntu.edu.sg with the content:
   'KBMamba (7rayrecovery@gmail.com) invites you to take the survey'

8.2.5 Delete survey

```
def delete_survey(request, survey_key=""):
    if request.POST:                                                    (1)
        surveyID = int(request.POST.get("surveyID"))
        survey = Survey.objects.get(id=surveyID)                        (2)
        survey.delete()
        dict = {}
        return HttpResponse(simplejson.dumps(dict),mimetype='application/javascript')   (3)
    else:
        try:                                                            (4)
            survey = Survey.objects.get(key=survey_key)                 (5)
        except Survey.DoesNotExist:                                     (6)
            raise Http404                                               (7)

        if request.user == survey.user or survey.is_collaborator(request.user):   (8)
            survey_title = survey.title                                 (9)
            survey.delete()
            messages.success(request, 'Survey "%s deleted" successfully' % survey_title)
    else:
        return error_jump(request,"unauthorized")                       (10)

    return HttpResponseRedirect('/account')                             (11)
```



4 different paths

Path1: a-b-c-e-f-h-j

Path2: a-b-c-e-g-i

Path3: a-b-c-d-k-l

Path4: a-m-n-o

Path1

Inputs:

Request.post == false

The survey to delete exists and the id =2, title = ‘Love’

The current user is an authorized user whose username is LJKing

Expected result:

Redirect to account page and the message says ‘Survey “Love” deleted successfully’

Path2

Inputs:

Request.post == false

The current user is an unauthorized user

Expected result:

Redirect to error page, displaying the message "Sorry, you are unauthorized to do this. Please check again."

Path3

Inputs:

Request.post == false

The survey to delete does not exist

The current user is an authorized user whose username is LJKing

Expected result:

Raise Http404

Path4

Inputs:

Request.post == true

The id of the survey to delete is 1

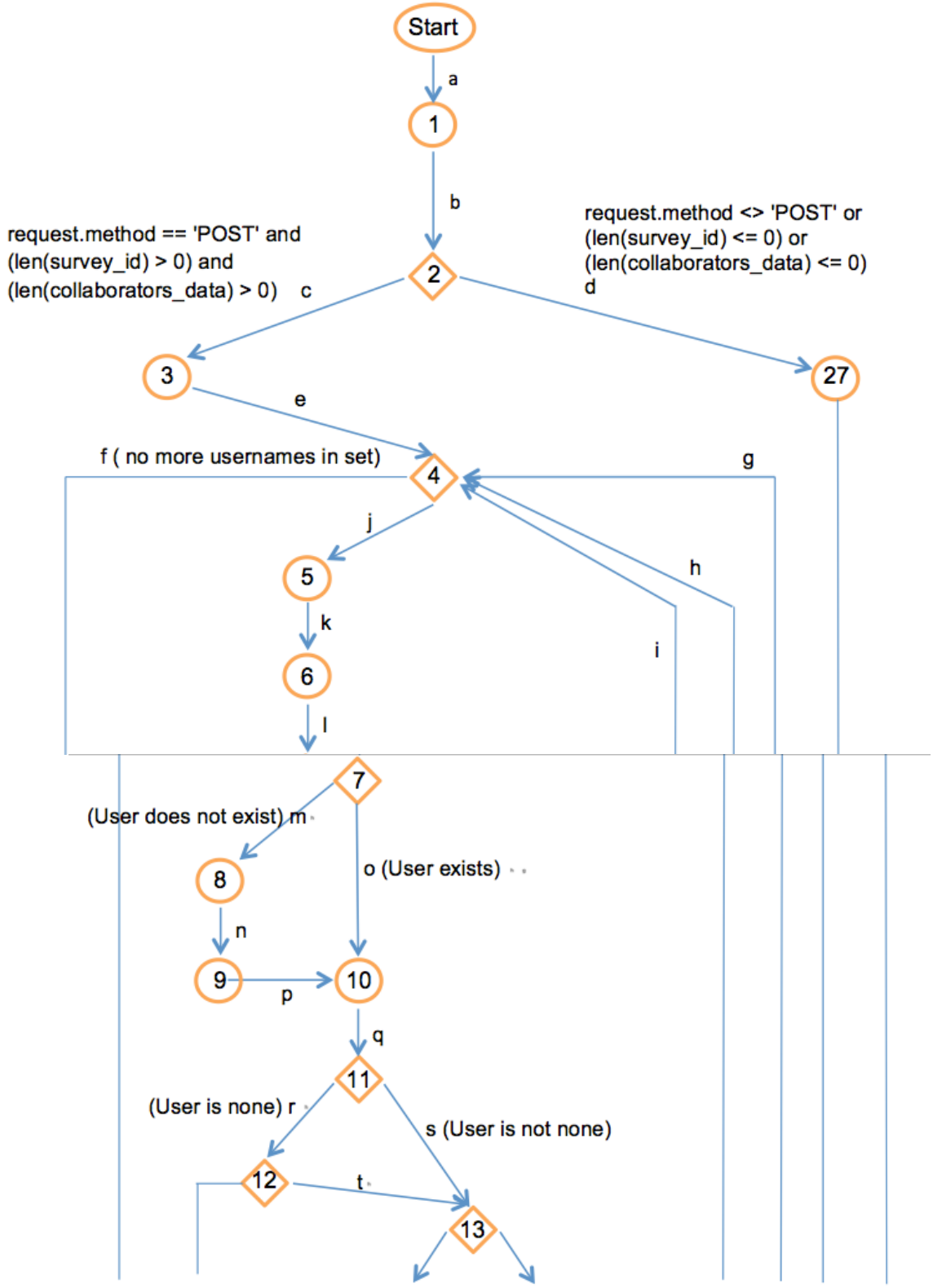
Expected result:

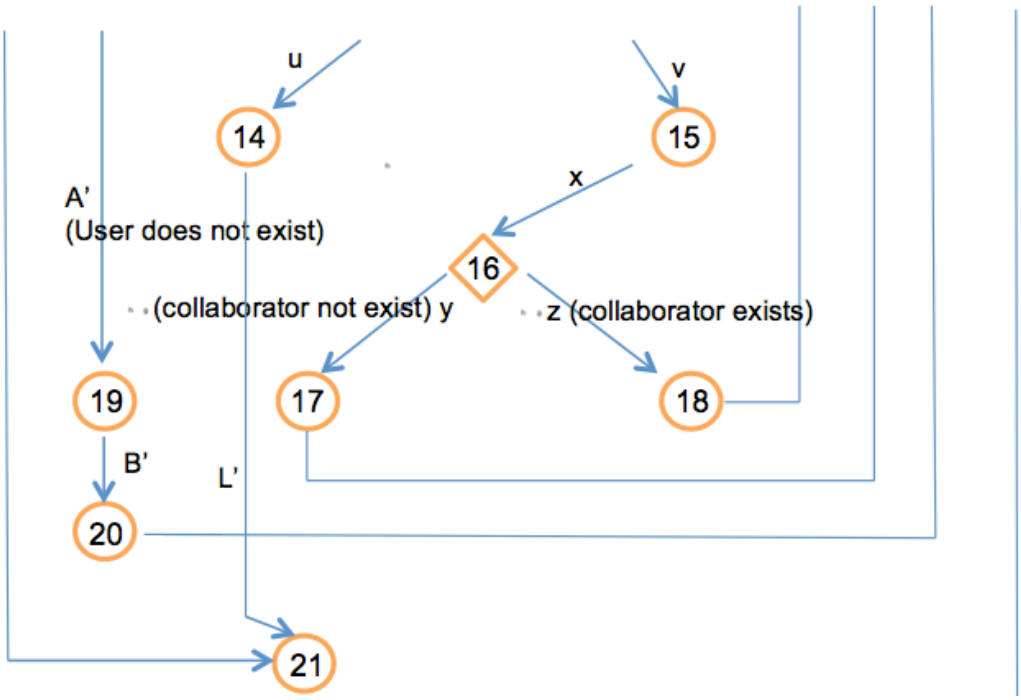
Survey (id=1) is deleted

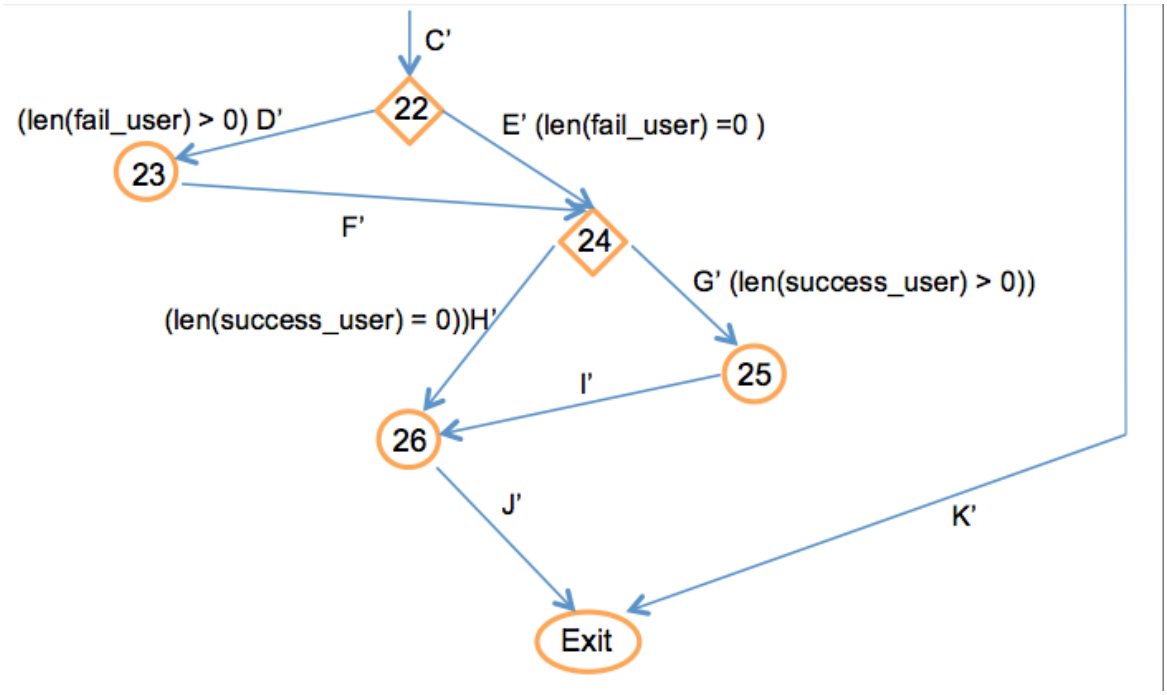
8.2.6 Collaborate

```
def invite(request):
    survey_id = request.POST.get('survey_id');                                  (1)
    collaborators_data = request.POST.get('collaborators');

    # Check whether is a valid request
    if request.method == 'POST' and (len(survey_id) > 0) and (len(collaborators_data) > 0):   (2)
        collaborators_username = [u.strip() for u in collaborators_data.split(',')]
        emails = []
        survey = Survey.objects.get(id=survey_id)                               (3)
        success_user = []
        fail_user = []
        for username in collaborators_username:                                 (4)
            user = None                                                         (5)
            try:                                                                (6)
                user = User.objects.get(username=username)                      (7)
            except User.DoesNotExist:                                           (8)
                pass                                                            (9)
            try:                                                                (10)
                if user is None:                                                (11)
                    user = User.objects.get(email=username)                     (12)

                if user.id == request.user.id:                                  (13)
                    messages.error(request, "You cannot add yourself as collaborator")   (14)
                else:
                    activation_code = sha.new(user.username).hexdigest()        (15)
                    activation_code = activation_code[0:9]
```

```
                if not Collaboration.objects.filter(user=user, survey=survey).exists():
                    Collaboration.objects.create(user=user, survey=survey, activation_code=activation_code)
                    emails.append(('Invitation for collaboration of survey', "http://%s/collaborate/accept/%s" %
(get_current_site(request).domain, activation_code), 'noreply@localhost', [user.email]))
                    success_user.append(username)
                else:
                    messages.error(request, "You have already added %s as collaborator" % username)
            except User.DoesNotExist:
                fail_user.append(username)


        send_mass_mail(emails)


        if len(fail_user) > 0:
            messages.error(request, "User %s not found" % (', ').join(fail_user))
        if len(success_user) > 0:
            messages.success(request, 'You have successfully invite %s as collaborator' % (',
').join(success_user))

        return HttpResponseRedirect("/account")
        #return HttpResponse(str(tuple(emails)))
    else:
        return HttpResponseRedirect("/account")
```


Start
a
1
b
request.method == 'POST' and
(len(survey_id) > 0) and
(len(collaborators_data) > 0) c
request.method <> 'POST' or
(len(survey_id) <= 0) or
(len(collaborators_data) <= 0)
d
2
3
27
e
f (no more usernames in set)
4
g
j
5
h
k
i
6
l
7
(User does not exist) m
o (User exists)
8
n
9
10
p
q
11
(User is none) r
s (User is not none)
12
t
13


u
v
14
15
x
A'
(User does not exist)
16
(collaborator not exist) y
z (collaborator exists)
19
17
18
B'
L'
20
21


C'
22
(len(fail_user) > 0) D'
E' (len(fail_user) =0)
23
F'
24
G' (len(success_user) > 0))
(len(success_user) = 0))H'
25
I'
26
J'
K'
Exit

6 Independent Test Paths

Path1: a-b-c-e-j-k-l-o-q-s-v-x-y-h-f-C'-E'-G'-I'-J'

Path2: a-b-c-e-j-k-l-o-q-s-u-C'-E'-H'-J'

Path3: a-b-c-e-j-k-l-o-q-s-v-x-y-h-j-k-l-o-q-s-v-x-z-i-f-C'-E'-G'-I'-J'

Path4: a-b-c-e-j-k-l-m-n-p-q-r-t-v-x-y-h-f-C'-E'-G'-I'-J'

Path5: a-b-c-e-j-k-l-m-n-p-q-r-A'-B'-g-f-C'-D'-F'-H'-J'

Path6: a-b-c-e-j-k-l-o-q-s-v-x-y-h-j-k-l-m-n-p-q-r-A'-B'-g-f-C'-D'-F'-G'-I'-J'

Path1

Inputs:

The survey for collaboration has id of 1

Input one collaborator: username is 12345 (an active user account)

Expected result:

1. An invitation link to collaborate on the survey will be sent to user 12345's email account.
2. Displaying successfully message : 'You have successfully invite 12345 as collaborator

Path2

Inputs:

The survey for collaboration has id of 1

Input one collaborator: username is 11111 (an active user account), but it is the same as the current user who's logged in.

Expected result:

1. Display error message: 'You cannot add yourself as collaborator'

Path3

Inputs:

The survey for collaboration has id of 1

Input two collaborators: the first collaborator's username is 11111 (an active user account), the second one is 12345 (an active user account but is already a collaborator for the survey)

Expected result:

1. An invitation link to collaborate on the survey will be sent to user 11111's email account.
2. Displaying successfully message : 'You have successfully invite 11111 as collaborator'
3. Displaying error message: 'You have already added 12345 as collaborator'

Path4

Inputs

The survey for collaboration has id of 1

Input one collaborator: user-email is 395215047@qq.com (an active user account email).

Expected result:

1. An invitation link to collaborate on the survey will be sent to 395215047@qq.com email account.
2. Displaying successfully message : 'You have successfully invite 395215047@qq.com as collaborator'

Path5

Inputs:

The survey for collaboration has id of 1

Input one collaborator: username is abcd (a non-existing user account).

Expected result:

1. Display error message: ‘User abcd not found’

Path6

Inputs:

The survey for collaboration has id of 1

Input two collaborators: first username is cherry (an active user account), the second is asdf (a non-existing username)

Expected result:

1. An invitation link to collaborate on the survey will be sent to user cherry’s email account.
2. Displaying successfully message : 'You have successfully invite cherry as collaborator'
3. Displaying error message: ‘User asdf not found’

8.3 Auto-testing

Django’s unit tests use a Python standard library module: unittest. This module defines tests in class-based approach.

For a given Django application, the test runner looks for unit tests in two places:
• The models.py file. The test runner looks for any subclass of unittest.TestCase in this module.
• A file called tests.py in the application directory – i.e., the directory that holds models.py. Again, the test runner looks for any subclass of unittest.TestCase in this module.

Below is part of the codes for testing models in this system.

```
class SurveyTest(TestCase):
   def setUp(self):
      #set up survey
      self.survey = Survey(title='Test survey')
      self.user = User()
      self.user.save()
      self.survey.user = self.user
      self.survey.save()

      # Q1. Multiple choice (id_in_response=0)
      mcq = MultipleChoiceQuestion(type='multiplechoice',
         title='A test for Multiple choice question',
         id_in_survey=0,
         is_required=True,
         survey=self.survey
      )
      mcq.save()

      mc1 = MultipleChoice(label='test label1', id_in_question=0, question=mcq)
      mc1.save()
      mc2 = MultipleChoice(label='test label2', id_in_question=1, question=mcq)
      mc2.save()

      mc3 = MultipleChoice(label='test label3', id_in_question=2, question=mcq)
      mc3.save()

      mc4 = MultipleChoice(label='test label4', id_in_question=3, question=mcq)
      mc4.save()

      response = Response(survey=self.survey, dt_start=datetime.now(),
dt_end=datetime(2012, 10, 19))
      response.save()
      answer = Answer(response=response, type='multiplechoice', id_in_response=0,
value='0')
      answer.save()

      response = Response(survey=self.survey, dt_start=datetime.now(),
dt_end=datetime(2012, 10, 19))
      response.save()
      answer = Answer(response=response, type='multiplechoice', id_in_response=0,
value='1')
      answer.save()

      response = Response(survey=self.survey, dt_start=datetime.now(),
dt_end=datetime(2012, 10, 19))
      response.save()
      answer = Answer(response=response, type='multiplechoice', id_in_response=0,
value='1')
      answer.save()
```

```
        response = Response(survey=self.survey, dt_start=datetime.now(),
dt_end=datetime(2012, 9, 7))
        response.save()
        answer = Answer(response=response, type='multiplechoice', id_in_response=0,
value='2')
        answer.save()

        response = Response(survey=self.survey, dt_start=datetime.now(),
dt_end=datetime(2012, 10, 19))
        response.save()
        answer = Answer(response=response, type='multiplechoice', id_in_response=0,
value='2')
        answer.save()

        response = Response(survey=self.survey, dt_start=datetime.now(),
dt_end=datetime(2012, 10, 20))
        response.save()
        answer = Answer(response=response, type='multiplechoice', id_in_response=0,
value='2')
        answer.save()

        response = Response(survey=self.survey, dt_start=datetime.now(),
dt_end=datetime(2012, 10, 20))
        response.save()
        answer = Answer(response=response, type='multiplechoice', id_in_response=0,
value='3')
        answer.save()


        # Q2. Numeric question (id_in_response=1)

        nq = NumericQuestion(type='numeric',
            title='A test for numeric question',
            id_in_survey=1,
            is_required=True,
            min_value=0.0,
            max_value=100.0,
            survey=self.survey)

        response = Response(survey=self.survey, dt_start=datetime.now(),
dt_end=datetime(2012, 10, 19))
        response.save()
        answer = Answer(response=response, type='numeric', id_in_response=1,
value='1.0')
        answer.save()

        response = Response(survey=self.survey, dt_start=datetime.now(),
dt_end=datetime(2012, 10, 19))
        response.save()
        answer = Answer(response=response, type='numeric', id_in_response=1,
value='2.0')
```

```
        answer.save()

        response = Response(survey=self.survey, dt_start=datetime.now(),
dt_end=datetime(2012, 10, 19))
        response.save()
        answer = Answer(response=response, type='numeric', id_in_response=1,
value='3.0')
        answer.save()

        response = Response(survey=self.survey, dt_start=datetime.now(),
dt_end=datetime(2012, 10, 19))
        response.save()
        answer = Answer(response=response, type='numeric', id_in_response=1,
value='4.0')
        answer.save()

        # Q3. Text question (id_in_response=2)

        tq = TextQuestion(type='text',
            title='A test for text question',
            id_in_survey=2,
            is_required=True,
            survey=self.survey)

        response = Response(survey=self.survey, dt_start=datetime.now(),
dt_end=datetime.now())
        response.save()
        answer = Answer(response=response, type='text', id_in_response=2, value='A
sample text for text question')
        answer.save()

        cq = CheckboxQuestion(type='checkbox',
            title='A test for checkbox question',
            id_in_survey=3,
            max_checked=4,
            min_checked=1,
            is_required=True,
            survey=self.survey)
        cq.save()

        cq1 = CheckboxChoice(label='test label1', id_in_question=0, question=cq)
        cq1.save()
        cq2 = CheckboxChoice(label='test label2', id_in_question=1, question=cq)
        cq2.save()
        cq3 = CheckboxChoice(label='test label3', id_in_question=2, question=cq)
        cq3.save()
        cq4 = CheckboxChoice(label='test label4', id_in_question=3, question=cq)
        cq4.save()

        response = Response(survey=self.survey, dt_start=datetime.now(),
dt_end=datetime(2012, 10, 28))
```

```
        response.save()
        answer = Answer(response=response, type='checkbox', id_in_response=3,
value='0')
        answer.save()

        response = Response(survey=self.survey, dt_start=datetime.now(),
dt_end=datetime(2012, 10, 29))
        response.save()
        answer = Answer(response=response, type='checkbox', id_in_response=3,
value='3')
        answer.save()

        response = Response(survey=self.survey, dt_start=datetime.now(),
dt_end=datetime(2012, 10, 27))
        response.save()
        answer = Answer(response=response, type='checkbox', id_in_response=3,
value='2')
        answer.save()

        response = Response(survey=self.survey, dt_start=datetime.now(),
dt_end=datetime(2012, 10, 28))
        response.save()
        answer = Answer(response=response, type='checkbox', id_in_response=3,
value='1')
        answer.save()

        response = Response(survey=self.survey, dt_start=datetime.now(),
dt_end=datetime(2012, 10, 28))
        response.save()
        answer = Answer(response=response, type='checkbox', id_in_response=3,
value='1')
        answer.save()

        response = Response(survey=self.survey, dt_start=datetime.now(),
dt_end=datetime(2012, 10, 28))
        response.save()
        answer = Answer(response=response, type='checkbox', id_in_response=3,
value='3')
        answer.save()

        sq = ScaleQuestion(type='scale',
            title='A test for scale question',
            id_in_survey=4,
            min_value = 1.0,
            max_value =5.0,
            increment =1.0,
            survey=self.survey)
        sq.save()

        response = Response(survey=self.survey, dt_start=datetime.now(),
dt_end=datetime(2012, 10, 28))
```

```
        response.save()
        answer = Answer(response=response, type='scale', id_in_response=4, value='1')
        answer.save()


    def test_multiple_choice(self):
        # 1 response for choice 0, 2 response for choice 1, 3 responses for choice 2, 1
response for choice 3

        queryset_answers = Answer.objects.filter(response__survey=self.survey,
id_in_response=0)
        self.assertEqual(queryset_answers.filter(value__exact=0).count(), 1)
        self.assertEqual(queryset_answers.filter(value__exact=1).count(), 2)
        self.assertEqual(queryset_answers.filter(value__exact=2).count(), 3)
        self.assertEqual(queryset_answers.filter(value__exact=3).count(), 1)



    def test_numeric(self):
        queryset_answers = Answer.objects.filter(response__survey=self.survey,
id_in_response=1)
        self.assertEqual(queryset_answers.aggregate(Min('value'))['value__min'], 1.0)
        self.assertEqual(queryset_answers.aggregate(Max('value'))['value__max'], 4.0)
        self.assertEqual(queryset_answers.aggregate(Avg('value'))['value__avg'], 2.5)

    def test_text(self):
        queryset_answers = Answer.objects.filter(response__survey=self.survey,
id_in_response=2)
        self.assertEqual(queryset_answers[0].value, 'A sample text for text question')

    def test_check_box(self):
        queryset_answers = Answer.objects.filter(response__survey=self.survey,
id_in_response=3)
        self.assertEqual(queryset_answers.filter(value__exact=0).count(), 1)
        self.assertEqual(queryset_answers.filter(value__exact=1).count(), 2)
        self.assertEqual(queryset_answers.filter(value__exact=2).count(), 1)
        self.assertEqual(queryset_answers.filter(value__exact=3).count(), 2)

    def test_scale(self):
        queryset_answers = Answer.objects.filter(response__survey=self.survey,
id_in_response=4)
        self.assertEqual(str(queryset_answers[0].value), '1')
```

Testing Result:

Positive, no error detected in the system.

8.4 Real User Testing

Real User Testing is implemented by creating a survey called campus transportation survey. Users are invited to experience the web system. 23 Responses are collected and the result is satisfactory.

9. Discussion

9.1 Explain how you derived your analytical and design models

Django
Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design.

Developed by a fast-moving online-news operation, Django was designed to handle two challenges: the intensive deadlines of a newsroom and the stringent requirements of the experienced Web developers who wrote it. It lets you build high-performing, elegant Web applications quickly.

Consideration
Most of team members have learned Python in first-year course CZ1003, with the easy-to-learn and flexible feature of Python, we believe Django it the best framework for our project.

DRY Principle
The DRY (Don't Repeat Yourself) Principle states:
Every piece of knowledge must have a single, unambiguous, authoritative representation within a system.

Django adheres strongly to the DRY principle. Django and all of its third-party packages are open-source, which gives us a great advantage to build amazing applications on the shoulders of other people's great work. Besides, Django has many ready-to-use features such as registration system, debug system, internationalization that provide us with much convenience.

To practice DRY principle, in this NTU Survey web application, we have adopted some interesting libraries :

1. south
   Database migration tools for Django.
   documentation: https://github.com/zocolab/django-qrcode

2. django-qrcode
   Generate QR code based on URL that redirects to survey form
   documentation: https://github.com/zocolab/django-qrcode

3. short-uuid
   Generate unique and nice-looking hash code as survey key (eg, key='vytxeTZskVKR7C7WgdSP3d')
   documentation: https://github.com/stochastic-technologies/shortuuid

4. pygeoip
   Map IP address to geographic location, used with Google chart tools API to plot the geo-chart
   documentation: https://github.com/appliedsec/pygeoip

   Also, we use google chart tools(https://developers.google.com/chart/) Javascript API to generate and display all our chart. With these great packages, we can just install and follow the documentation to implement our design requirements. This is especially good when we need only worry about feature to implement instead of re-inventing the wheels.

MVC

Model–View–Controller (MVC) is an architecture that separates the representation of information from the user's interaction with it. The model consists of application data and business rules, and the controller mediates input, converting it to commands for the model or view. A view can be any output representation of data, such as a chart or a diagram. The central idea behind MVC is code reusability and separation of concerns.

Django's MTV

Django releases the MVC framework. However, the terms are called differently in Django, T is for templates(View) and V is for Views(Controller).

M - Models (MVC's model)
T - Templates(MVC's view)
V - Views (MVC's Controller)

To make it more straight-forward, in Django, M is the models.py in every application, T is the templates files (basically HTML files and Django tags), and V is the views.py containing python fallback functions.

Advantages of using Django

1. Separation of Design and Logic

Using Django gives us much flexibility to separate design and logic. In other words, front-end UI designers need not worry about the complex python code in views.py, and back-end database engineers can focus on the flow of coding rather than the presentation of web pages. To bridge the back-end and front-end, Django has its own way that works fantastically - template tags. For example, in a Django template file, double curly braces {{ varible_name }} can access the value of variable passed from Django view; {% expression %} can be used to perform various purposes right in template,e.g. for loop and if statement.
Documentation:
https://docs.djangoproject.com/en/dev/ref/templates/builtins.

2. No low-level SQL queries

Django uses **ORM**(Object-relational mapping), a programming technique for converting data between incompatible type systems in object-oriented programming languages. With this, we are freed from the complex and troublesome raw SQL queries such as “SELECT a FROM b WHERE c”. Instead, we can write high-low python code like, model_name.objects.filter(condition).
Documentation: https://docs.djangoproject.com/en/dev/topics/db/queries/.

9.2 Explain how you performed your user-interface design

Our aim of the user-interface is user-friendly. On each main function of our system, we included an intuitive help message which will guide the user whenever he needs help. Not only that, we also have our help video that benefits to visual learner.

This is an interactive website which users are able to share our website to their friends in Facebook, twitter or other social networking website.

In helping the user to perform efficient work, we enable the search toolbar to be auto-complete. This feature is upmost helpful when a user has a huge amount of survey database and desired to search for one particular old survey. Instead of struggling to recall the name of the survey or scroll down to find the survey, auto-complete toolbar helps the user to find all the surveys that contains the keyword that he entered.

9.3 Describe difficulties encountered and solutions applied

The first difficulty is the installation of the program needed. Lots of time is spend on searching for reason and solve the problems of installation. As most of our member is new to Django, CSS, HTML, a lot of effort is putted in to get familiar with those language. During the process of programming, problem faced may not have a answer at the moment. Members spend upmost time in reading tutorials and try out different methods.

We suffered a long time from the version control conflict. As different people is editing the code at the same time, a conflict always occurs which will made additional problem to our system. After some trial and errors, we found out that Pycharm has a characteristic which is GUI that will handle the control conflict. It will compare the files and found out the conflicting files and prompt the user some options to choose how to deal with the files.

During the implementation of our major function “create survey” we encounter some difficulties which confuse us for quite a while. The online sample code for the creation of question are not user friendly as we wished for our website. We went through lots of samples and found no code meets our expectation. So we took a hard time refer to materials and samples and develop our own unique code that not only function fantastic and also user-friendly.

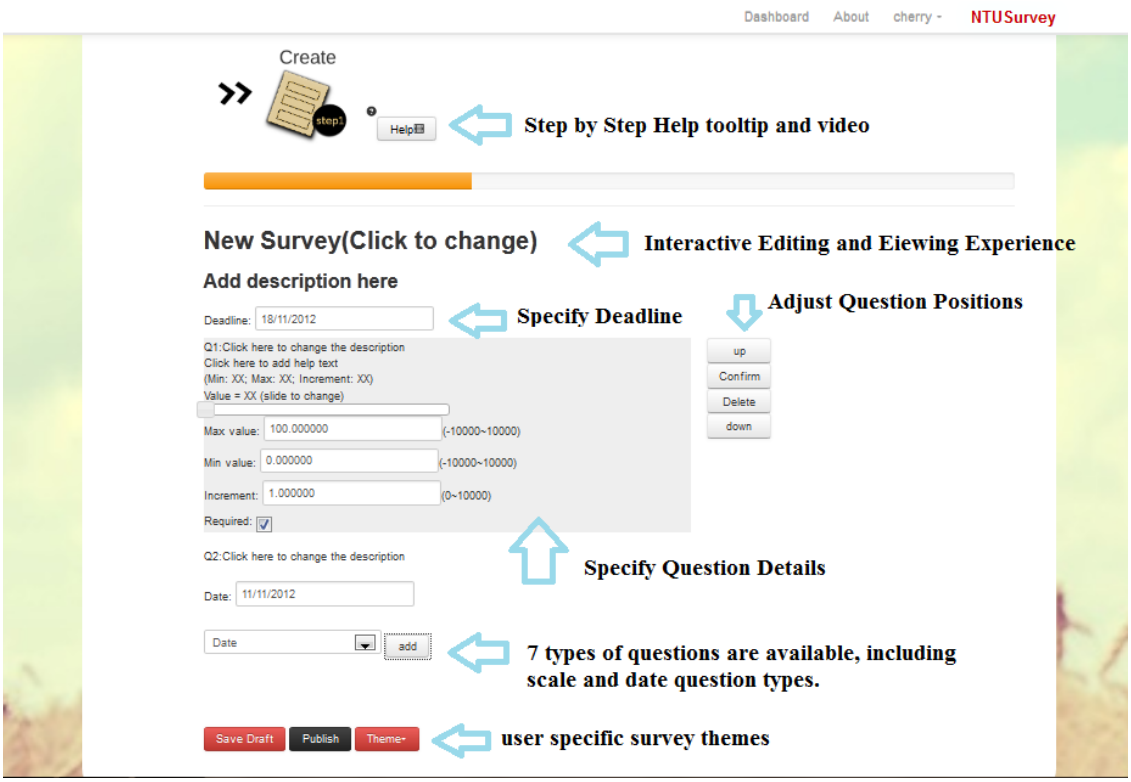


Figure: 1.6: The create survey function we finally come up with

Browser compatibility is always a big issue for website. After running on different browsers, we discovered that our system is able to run on major web browser such as safari, Chrome and Firefox. However there is a issue with IE. Our system can only run on IE9 without compatibility setting. In order to solve this, we made a validation in our code so that when a user open our website using IE lower than IE9, the system will prompt a message to ask the user to update to IE9 or change to another browser.

After the system is done, it is necessary to test our system. However, testing has not been teach at that time. We not only have to deploy on bonus functions and also spend time to read on testing. We read on textbook, lecture notes and search for online materials for steps on how to do testing. After get an idea of how to do the test cases, we then share with the members to share the work and test all parts of out functions and system.

10. Bonus implementation

10.1 Collaboration

We added in a collaboration button for each unpublished survey. Survey creator is able to click on the button to invite others to join this survey as a survey editors. After being invited, the user can also edit the survey question. In order to do this, the invited user must be a registered member of stardom also. Survey creator click on the collaboration button and type in the username to invite him. If the username entered is not valid, an error message will be shown. After which, a activation e-mail will be send to the collaborator. Once activated, he is able to play his role as a collaborator. The survey creator can remove the collaborator is he wanted to do so. However, collaborators do not have the right to publish the survey, this right remains for the survey creators.

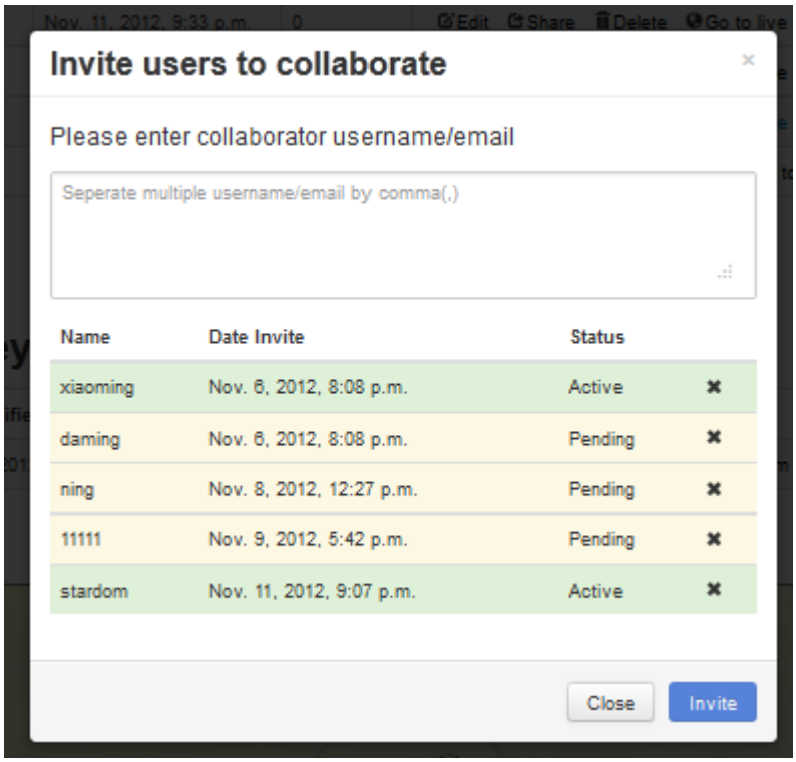


Figure 1.7: collaborate interface

10.2 Share

Each survey is attached with a share button which enables the survey creators to share the survey with their friends to invite them to give response to survey. After click on the button, survey creator is able to enter the username or e-mail addresses and separate the addresses using a comma. Invited respondents are able to answer the survey with the link provided in the e-mail. If the user invited is not exist or the e-mail address is not valid, a error message will be shown.

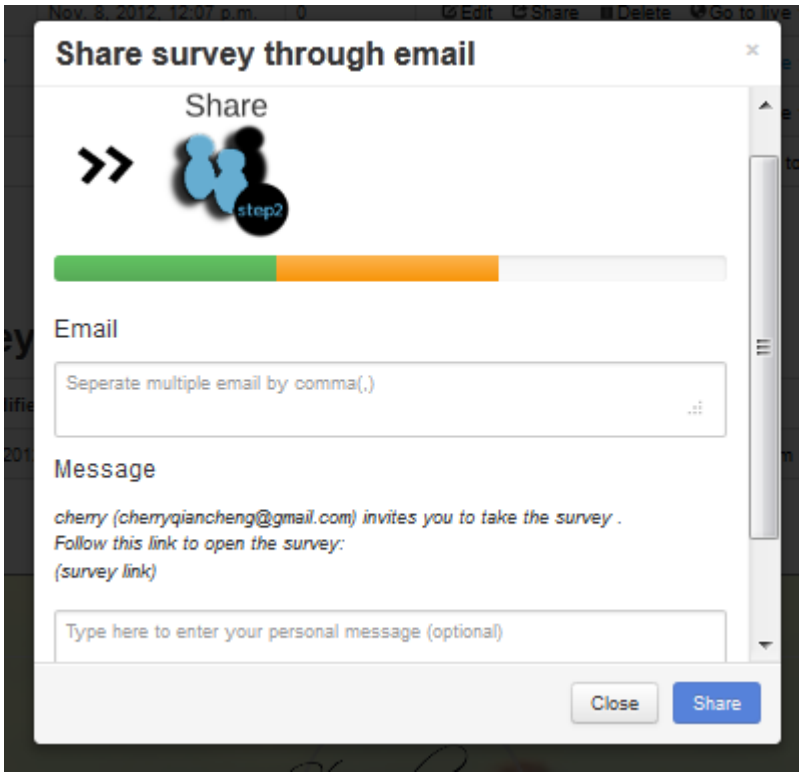


Figure1.8: Share the survey to friends

10.3 Additional question types

Apart from the required question types, we included another two additional question time to meet the needs of the users.

Scale question: The creator is able to set a initial, maximum and minimum value for the bar. Increment value can also be set. To answer this question, just drag from left to right to a desired value.

Date question: The range of date and initial date can be set by the user. To answer the question, respondent may write the date in the box or select a date from the calendar.

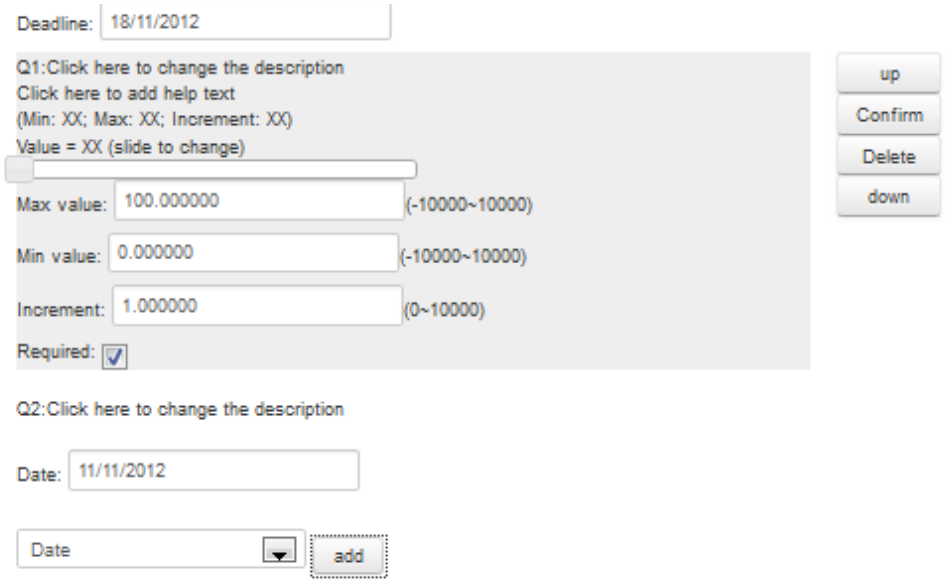


Figure 1.9: Additional question type, scale and date question

10.4 Validation

During the process of creating the survey question, the creator is able to specify if the specific question is compulsory for the respondents to answer. For the question that are compulsory, a red * is displayed beside the published question. Questions are default as compulsory. To make changes, just unselect the “required” button in the specific question.

10.5 Share to social networking website

On the bottom of the page, sharing button for various social networking website. For example: Facebook, twitter. Click on the button to share this website with their friend.

Figure 2.0: sharing function available

10.6 Deadline

On the survey creation page, the survey creator is able to state a period of time which the respondents can response to the survey. After the deadline is meet, the survey is closed and no one is allowed to answer the survey.

10.7 Visual charts

Besides the bar chart, more choices are given to the survey creator to visualise the results. In addition to bar chart, we have geo chart, pie chart, column chart, and line chart. Each chart has its own characteristic to best show the results for each question. The raw data will be displayed below each chart or when the cursor is moved to a specific area.

Geo chart: The IP address will be recorded for each respondents and it will be mapped to the location and displayed on a world map. When cursor is moved to a shaded region, the country name and number of respondents will be shown.

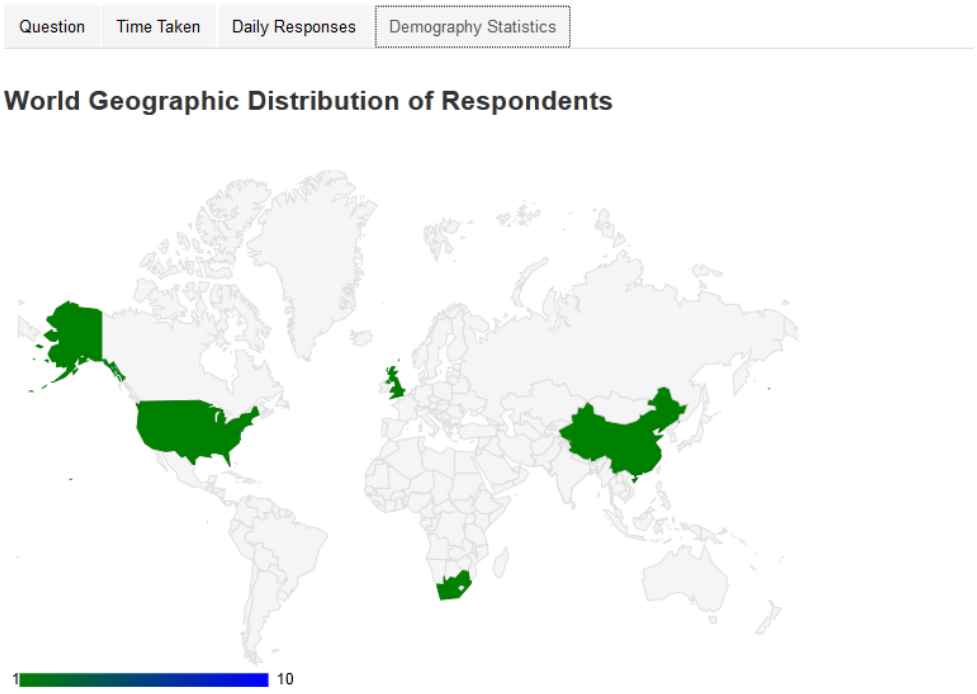


Figure 2.1: Geo chart generated from real user testing on the system

Pie chart: Pie chart will be used to display the answer for multiple choice question. And the raw data is displayed below the chart.

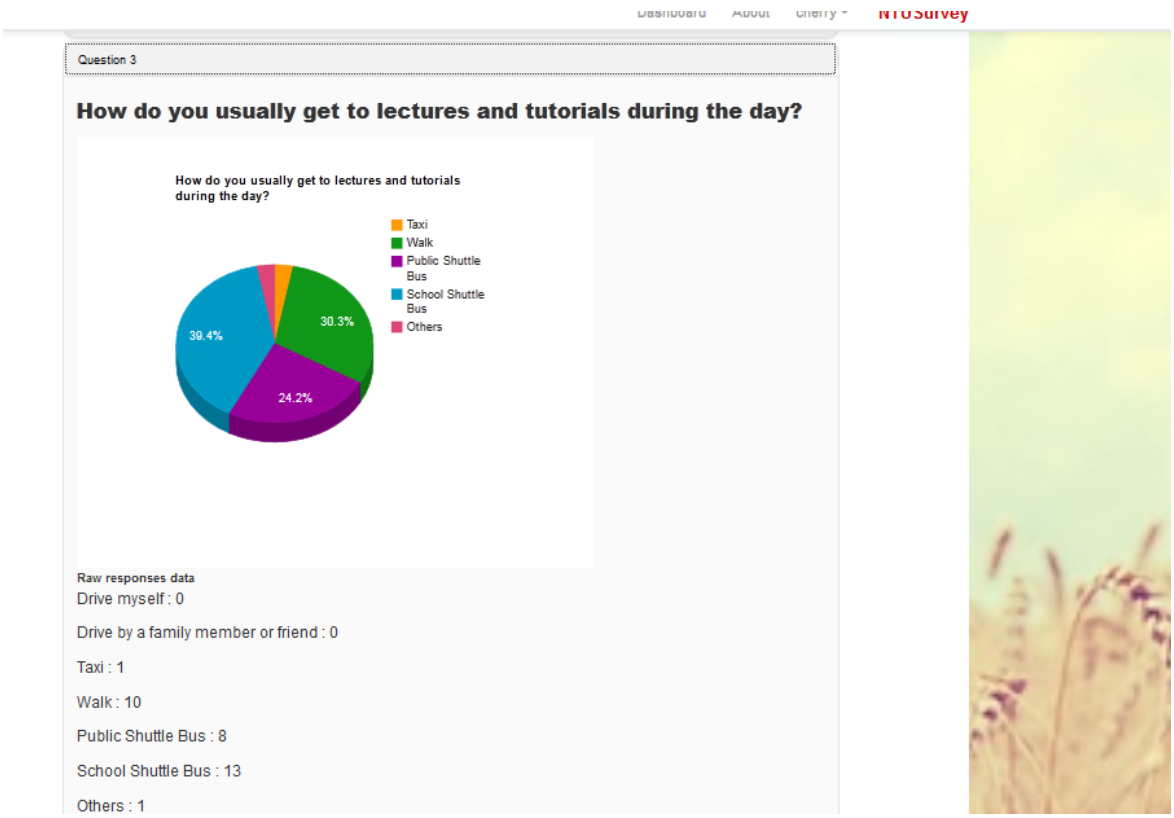


Figure 2.2: Pie chart based on the real user testing

Column chart: column chart will be used for scale question and the raw data of each entry is displayed below the chart.

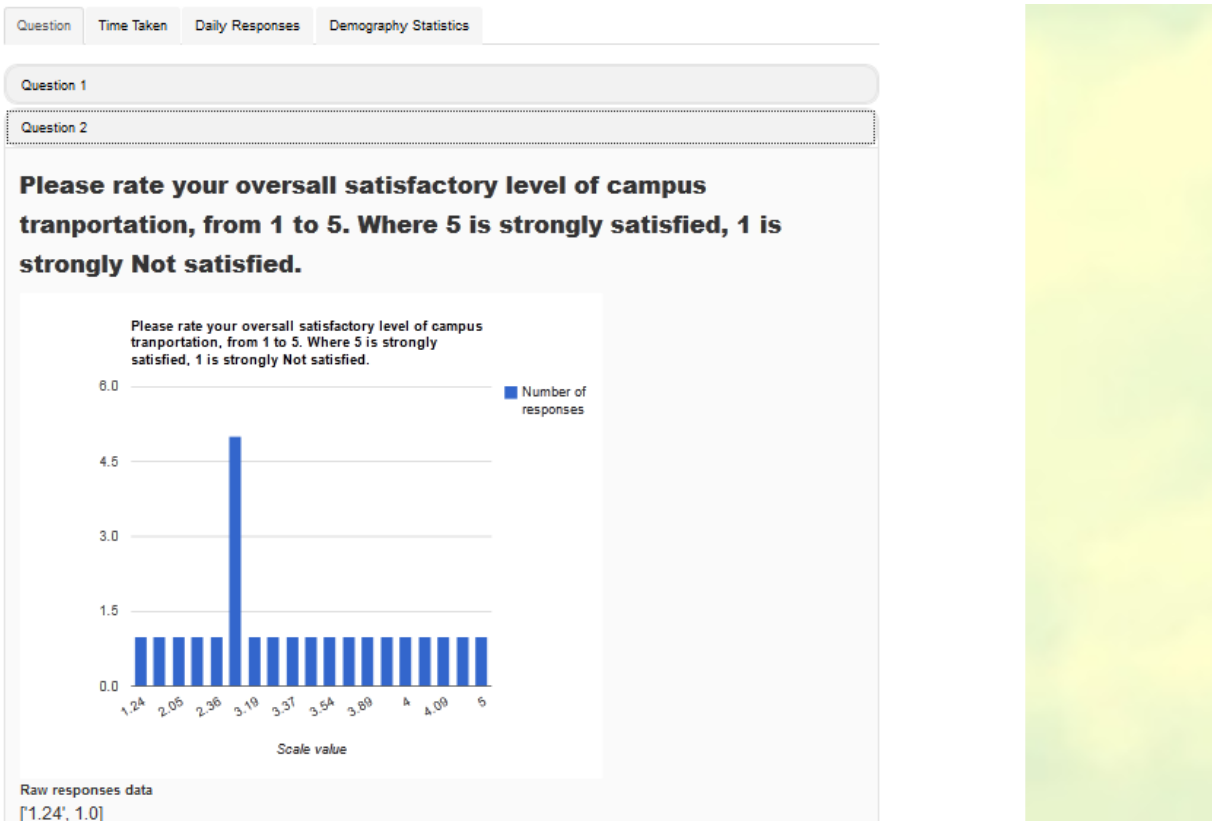


Figure 2.3: Column chart generated based on real user testing

Line chart: line chart is used to show the number of respondents each day. Move the cursor to the turning point on the line will show the number of respondents on that particular day.

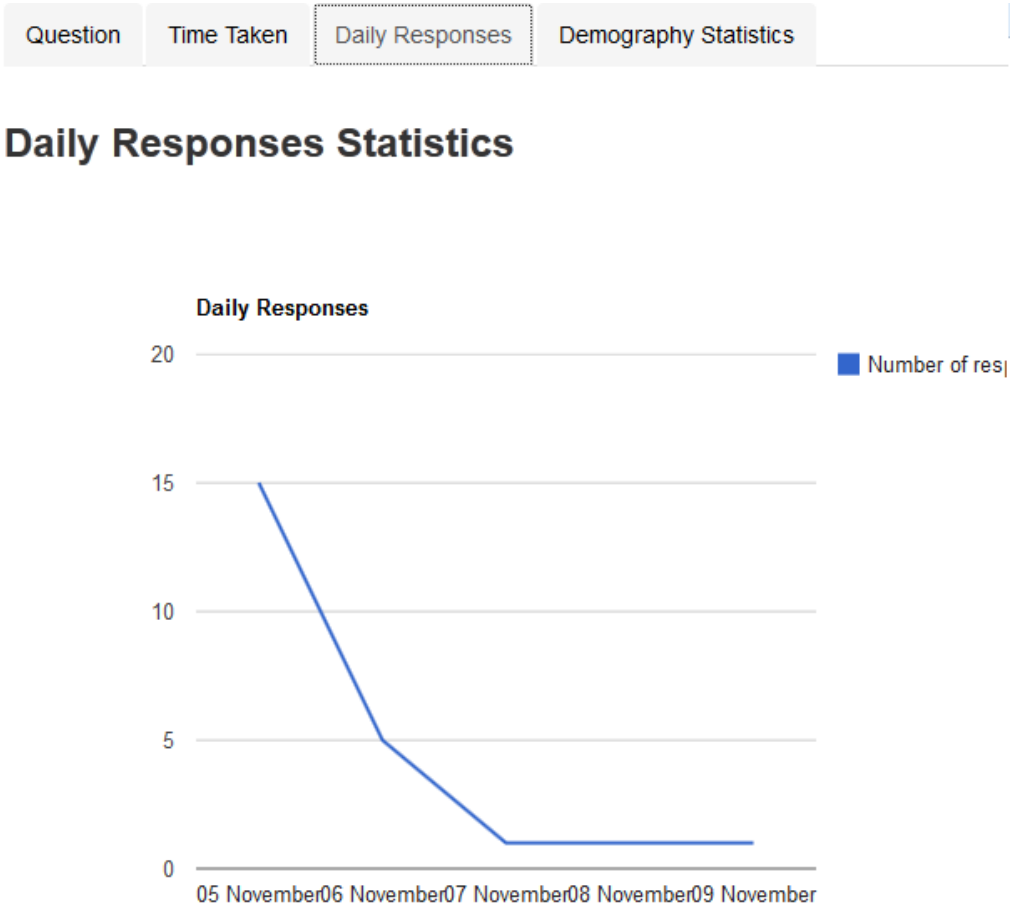
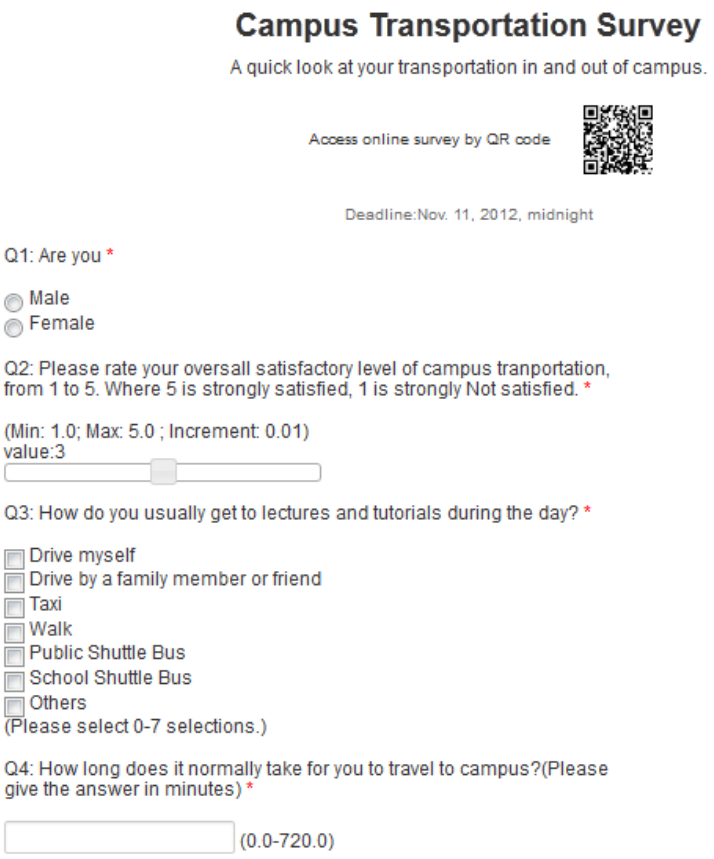


Figure 2.4: Line chart generated based on the number of respondents each day after the survey is published

10.8 Print survey

In case the survey creator hope to distribute the survey by paper or keep as references, print function is available for each survey. This extend availability of the survey apart from the web version. A print friendly page will be displayed with the QR code on top of the survey. This page can also be saved as PDF document to keep as a reference and print it later.



Campus Transportation Survey

A quick look at your transportation in and out of campus.

Access online survey by QR code

Deadline:Nov. 11, 2012, midnight

Q1: Are you *

- Male
- Female

Q2: Please rate your oversall satisfactory level of campus tranportation, from 1 to 5. Where 5 is strongly satisfied, 1 is strongly Not satisfied. *

(Min: 1.0; Max: 5.0 ; Increment: 0.01)
value:3

Q3: How do you usually get to lectures and tutorials during the day? *

- Drive myself
- Drive by a family member or friend
- Taxi
- Walk
- Public Shuttle Bus
- School Shuttle Bus
- Others

(Please select 0-7 selections.)

Q4: How long does it normally take for you to travel to campus?(Please give the answer in minutes) *

(0.0-720.0)

Figure 2.5: A user friendly print page

10.9 QR code

On the survey printout, a unit QR code is attached for easy reference. For example, if the survey printout is pasted on the notice board, interested user is able to scan the QR code and answer the question using smart phones or tablets. With the help of the QR code, the survey can be accessed on mobile devices.

10.10 Theme

During the process of creating the survey, the creator is given the choice to choose a desired background of the survey instead of the default theme. This makes the survey more attractive and interesting which in turn attract more respondents.

10.11 Mobile

Our system can support any PC or laptop with browser include safari, chrome, IE9, Firefox. With the increasing trend of user smart phone and Ipad to aid study, we also follow the trend to develop a mobile survey system. Our survey system can be supported on smart phones such as Iphone or android and also Tablet PC such as Ipad. There is two way to visit the survey questions on mobile devices. One is open the website with the URL link. The other method is to scan the barcode of the specific survey and the barcode will direct you to the survey question site. To do this, a app to scan the barcode must be installed first.

Figure 2.6: Visit the site using Iphone

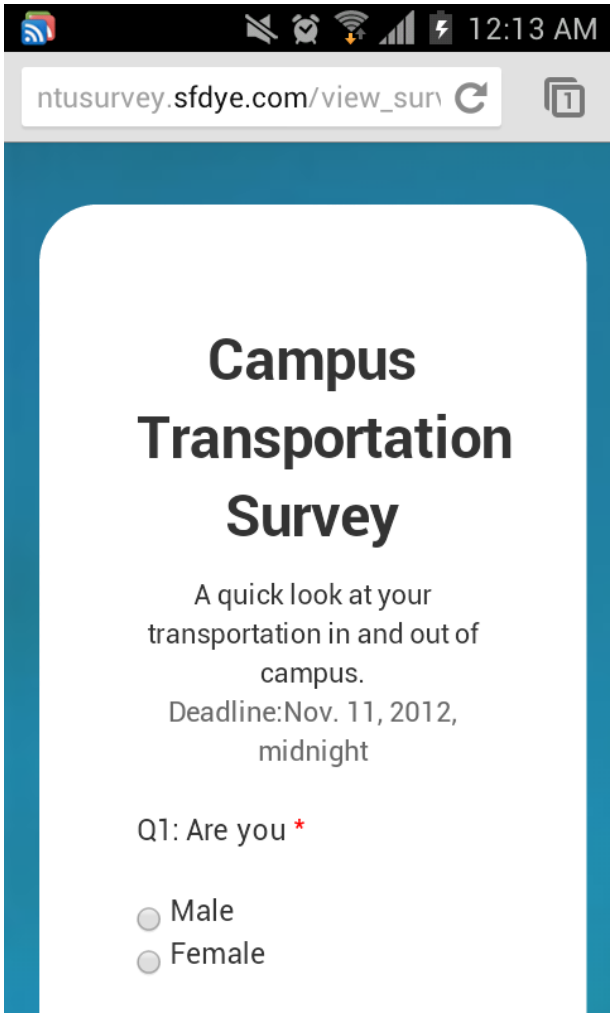


Figure 2.7: Visit the site using android phone

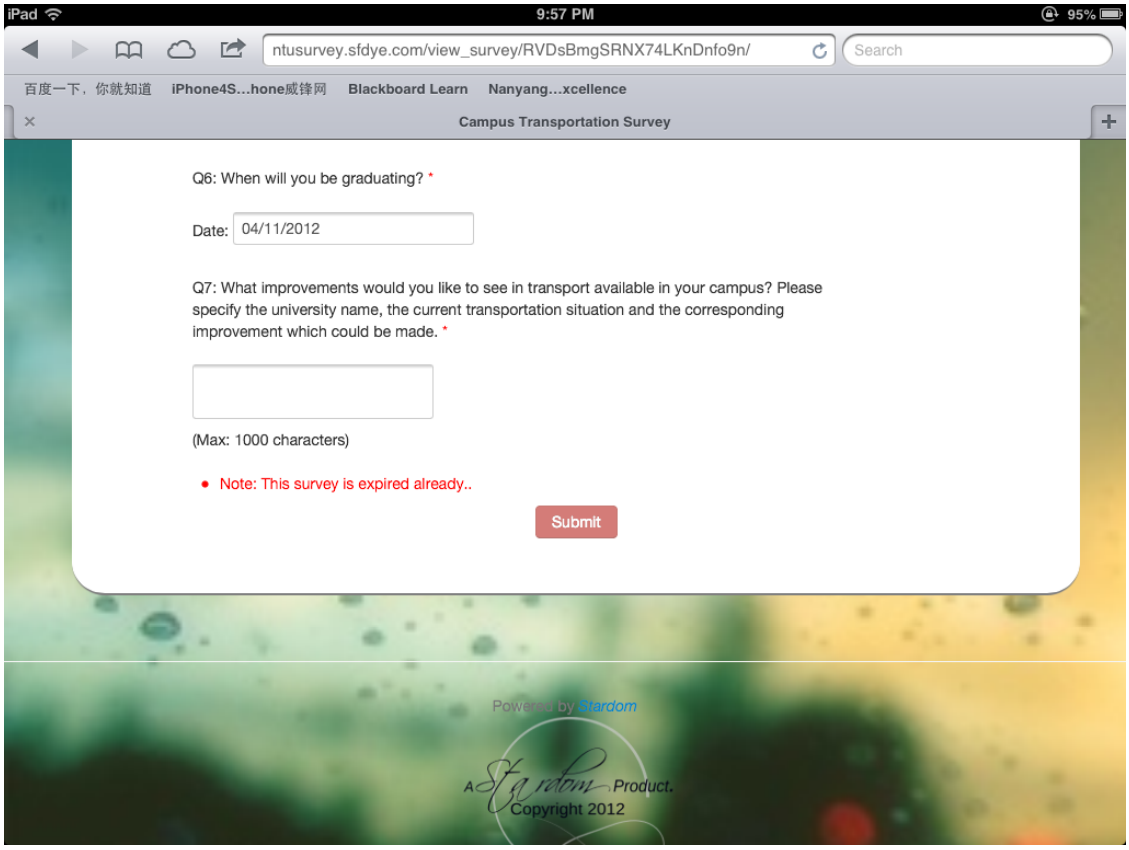


Figure 2.8: Visit the site using ipad

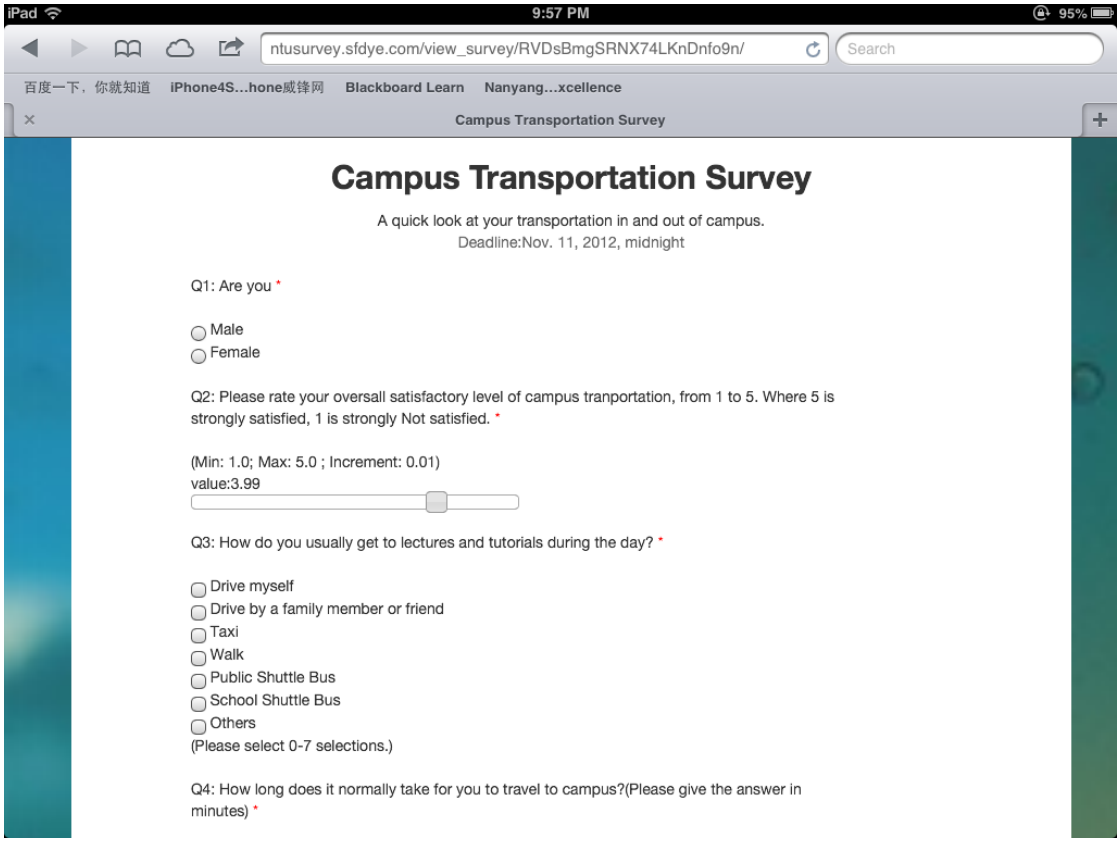


Figure 2.9: Visit an unexpired survey question using Ipad

11. Work Breakdown Structure

	SRS	Analysis doc	Design doc	Testing doc	Final report
Wan Liuyang	14	8	9	10	8
Chia Mingen	14	8	9	0	8
Chu Xiaoqi	15	9	8	0	8
Zhao Fangyuan	14	8	8	10	9
Ning Haoyan	14	50	50	20	50
Huang Wei	15	8	8	60	9
Qian Cheng	14	9	8	0	8
Total (100%)	100	100	100	100	100

	Django views	Django templates	Django Models
Wan Liuyang	17	0	30
Chia Mingen	17	0	30
Chu Xiaoqi	50	0	30
Zhao Fangyuan	0	50	0
Ning Haoyan	0	0	0
Huang Wei	16	0	10
Qian Cheng	0	50	0
Total (100%)	100	100	100

	database	Web server	SVN	Issue tracking
Wan Liuyang	30	100	40	30
Chia Mingen	30	0	10	12
Chu Xiaoqi	30	0	10	11
Zhao Fangyuan	0	0	10	12
Ning Haoyan	0	0	10	11
Huang Wei	10	0	10	12
Qian Cheng	0	0	10	12
Total (100%)	100	100	100	100

12. Glossary

actor: A person, software system, or hardware device that interacts with a system to achieve a useful goal. Also called a *user role*.

assumption: A statement that is believed to be true in the absence of proof or definitive knowledge.

business requirement: A high-level business objective of the organization that builds a product or of a customer who procures it.

constraint: A restriction that is imposed on the choices available to the developer for the design and construction of a product.

functional requirement: A statement of a piece of required functionality or a behaviour that a system will exhibit under specific conditions.

scope: The portion of the ultimate product vision that the current project will address. The scope draws the boundary between what's in and what's out for the project.

software requirements specification (SRS): A collection of the functional and non-functional requirements for a software product.

stakeholder: A person, group, or organization that is actively involved in a project, is affected by its outcome, or can influence its outcome.

use case: A description of an interaction between an actor and a system that results in an outcome that provides value to the actor.

use case diagram: An analysis model that identifies the actors who can interact with a system to accomplish valuable goals and the various use cases that each actor will perform.

vision: A long-term strategic concept of the ultimate purpose and form of a new system.

13. Reference

Yanic Inghelbrecht. (2008/7/6). A quick introduction to UML sequence diagram. visit date: 2012/8/20， from: trace modeler: http://www.tracemodeler.com/articles/a_quick_introduction_to_uml_sequence_diagrams/

Douban. (2011/10/27). background. visit date: 2012/10/10， from: douban: http://site.douban.com/122585/

Twitter. (2012/2/4). bootstrap. visit date: 2012/9/10， from: bootstrap: http://twitter.github.com/bootstrap/

Glyphicon. (2012/1/2). glyphicon. visit date: 2012/10/4， from: glyphicon: http://glyphicons.com

Django. (2011/3/12). django documentation. visit date: 2012/9/22， from: django: https://docs.djangoproject.com/en/1.4/

Django. (2011/10/2). writing your first django app. visit date: 2012/9/25， from: django: https://docs.djangoproject.com/en/1.4/intro/tutorial01/

Light bird. (2009/5/10). django by example: django tutorial. visit date: 2012/8/29， from: Light bird: http://lightbird.net/dbe/

Pycharm. (2009/4/19). pycharm. visit date: 2012/9/10， from: getiing started with pycharm: http://www.jetbrains.com/pycharm/quickstart/index.html

W3school. (2000/12/3). w3school. visit date: 2012/10/1， from: w3school tutorial: http://www.w3schools.com/html/default.asp

Stockoverflow. (2011/12/1). stockoverflow. visit date: 2012/10/2， from: stockoverflow: http://stackoverflow.com

Google. (2012/4/3). google developer. visit date: 2012/9/10， from: google charts: https://developers.google.com/chart/

Altova. (2012/2/1). altova. visit date: 2012/9/13， from: UML sequence diagram: http://www.altova.com/umodel/sequence-diagrams.html

Sparx system. (2011/10/3). sparx system. visit date: 2012/9/12， from: UML2 sequence diagram: http://www.sparxsystems.com.au/resources/uml2_tutorial/uml2_sequencediagram.html

Modelingagile. (2012/3/2). agile modeling. visit date: 2012/9/13， from: UML2 sequence diagram: http://www.agilemodeling.com/artifacts/sequenceDiagram.htm

Guru99. (2011/2/10). what is regression testing. visit date: 2012/10/4， from: guru99: http://www.guru99.com/regression-testing.html

Buzzle. (2011/2/3). software testing-whitebox testing strategy. visit date: 2012/10/4， from: buzzle: http://www.buzzle.com/editorials/4-10-2005-68350.asp

Guru99. (2011/4/1). black box testing. visit date: 2012/10/4， from: guru99: http://www.guru99.com/black-box-testing.html

WilliamsLaurie. (2006/4/1). testing overview. visit date: 2012/10/4， from: Laurie Williams : http://agile.csc.ncsu.edu/SEMaterials/BlackBox.pdf