NEXT

QuickManage
Software Requirements Specification
2.2

Revision History

Date	Version	Description	Author
19/03/2013	2.0	First published version	Trinh Luan, Ngo Tan Thinh, Nguyen Manh Hung, Cao Phi Hung, Dao Tien Minh
14/04/2013	2.1	Second published version with more details and clarifications based on the first published version as well as new features addded to the program	Trinh Luan, Ngo Tan Thinh, Nguyen Manh Hung, Cao Phi Hung, Dao Tien Minh
09/05/2013	2.2	Thrid published version add more functions into the documents	Trinh Luan, Ngo Tan Thinh, Nguyen Manh Hung, Cao Phi Hung, Dao Tien Minh

Contents

1. Introduction 5
- 1.1 Purpose 5
- 1.2 Scope 5
- 1.3 Assumptions and Dependencies 5
- 1.4 References 5

2. Specific Requirements 5
- 2.1 Actors 5
- 2.2 Use Case Diagram 6
- 2.3 Use Case Specifications 7
  - 2.3.1 Add Manager 7
  - 2.3.2 Add Staff 9
  - 2.3.3 Add Teacher 10
  - 2.3.4 Add Student 12
  - 2.3.5 Add Class 13
  - 2.3.6 View Manager 15
  - 2.3.7 View Staff 15
  - 2.3.8 View Teacher 16
  - 2.3.9 View Student 16
  - 2.3.10 View Class 17
  - 2.3.11 Edit Manager 18
  - 2.3.12 Edit Staff 18
  - 2.3.13 Edit Teacher 19
  - 2.3.14 Edit Student 20
  - 2.3.15 Edit Class 21
  - 2.3.16 Delete Manager 21
  - 2.3.17 Delete Staff 22
  - 2.3.18 Delete Teacher 23
  - 2.3.19 Delete Student 23
  - 2.3.20 Delete Class 24
  - 2.3.21 Delete Class type 25
  - 2.3.22 Delete Room 25
  - 2.3.23 Activate/Deactivate Manager 26
  - 2.3.24 Activate/Deactivate Staff 27
  - 2.3.25 Activate/Deactivate Teacher 28
  - 2.3.26 Activate/Deactivate Student 28
  - 2.3.27 Assign Teacher to Class 29
  - 2.3.28 Enroll the Student 30
  - 2.3.29 Copy Class 30
  - 2.3.30 Search Manager 31
  - 2.3.31 Search Staff 31
  - 2.3.32 Search Teacher 32
  - 2.3.33 Search Student 32
  - 2.3.34 Search Class 33
  - 2.3.35 Generate Invoice 33
  - 2.3.36 View Report 34

2.3.37 Payslip 35
2.4 Non-Functional Requirements 35
2.4.1 Installability 35
2.4.2 Usability 36
2.4.3 Performance 36
2.4.4 Scalability 36

Software Requirements Specification

1. Introduction

1.1 Purpose

The purpose of this document is to provide specific requirements of some new features in assignment 2 which include a set of functional and non-functional requirements, Actors and Use Case diagram to develop the QuickManage program. It will explain the purpose and features of the program, the interfaces of the program, what the program will do and some other install ability, usability, performance and scalability related to the program. We are focusing on making the Invoice Form for each student. We also provide Junit test case for our program.

1.2 Scope

This program will be a QuickManage for a local music and art school in HCMC. It will be designed to maximize the user's management by providing the friendly UI and extremely easy to use for the people who have no skill about controlling the program. More specifically, this program is designed to allow the manager and staff to manage all information about the teacher, student, class and also other manager/staff.

This document is limited within the school and the users within, as all the data of the managers, staffs, teachers, classes and students are stored in the database of the school. The use and access of the database is only available for managers and staffs. The managers and staffs are able to access information from school via this program.

1.3 Assumptions and Dependencies

No specific assumption or dependencies.

1.4 References

Oracle. 1995. *How to Use File Choosers*. [ONLINE] Available at:http://docs.oracle.com/javase/tutorial/uiswing/components/filechooser.html. [Accessed 19 March 13].

mkyong. 2008. *How To Resize An Image In Java ?*. [ONLINE] Available at:http://www.mkyong.com/java/how-to-resize-an-image-in-java/. [Accessed 19 March 13].

Javarevisited.2012.How To Copy File In Java Program http://javarevisited.blogspot.com/2012/05/how-to-copy-file-in-java-program.html [Accessed 30 March 13]

2. Specific Requirements

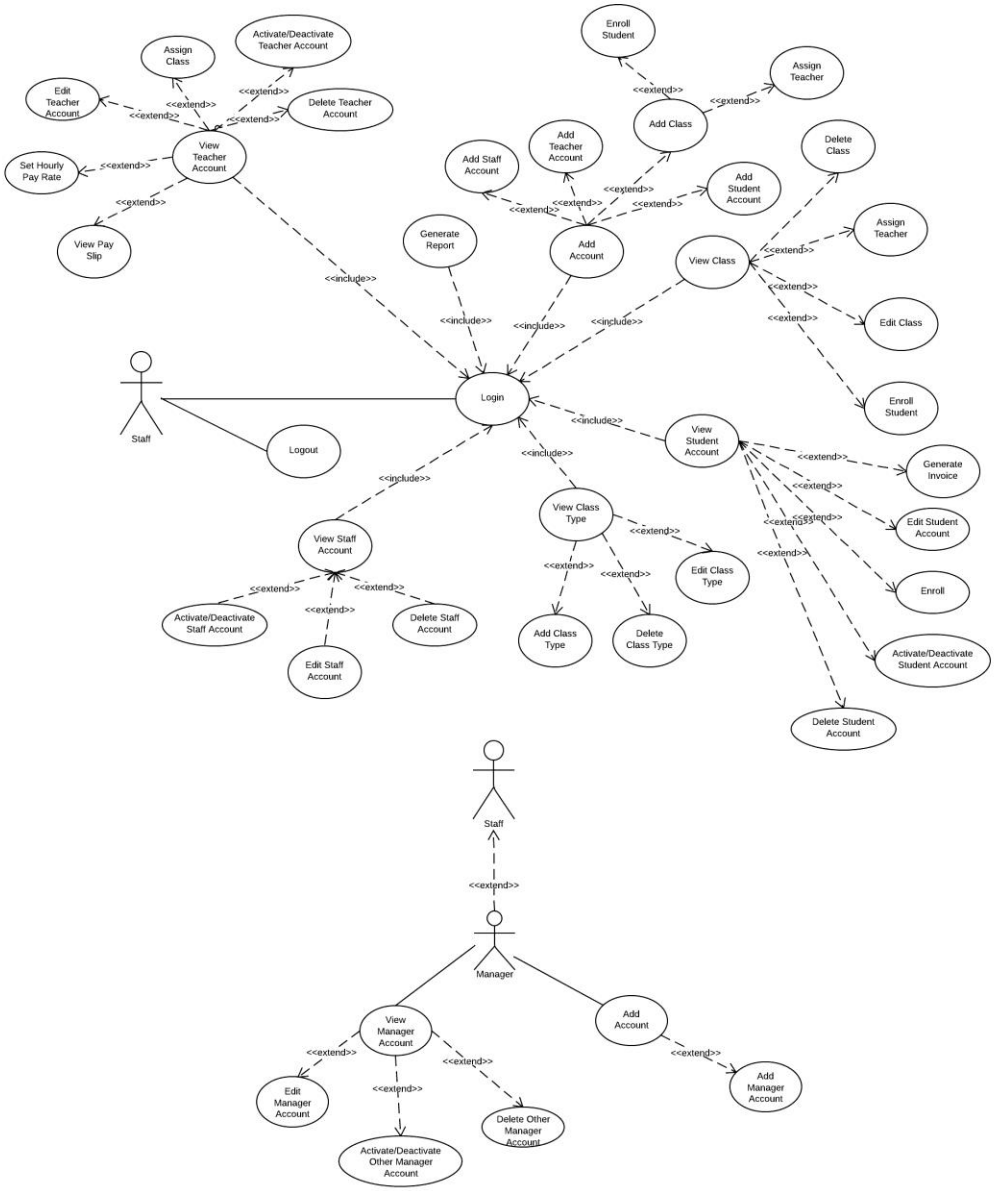
2.1 Actors

	Manager	Staff
Add/View/Edit/Delete/Active/Deactivate All Accounts, Change Class type, and room information.	X	
Add/View/Edit/Delete/Active/Deactivate Staff/Student/Teacher/Class and change	X	X

room information.		

	Classes	Students	Teachers
Contact Person		X	
Assign/Enroll		X	X
Copy	X		
Invoice		X	

2.2 Use Case Diagram



2.3 Use Case Specifications

2.3.1 Add Manager

Use Case Name: Add Manager	ID: UC001
Actors: Manager	Priority: 1

Brief description: This use case describes how the User use the add function to add Manager account

Pre-conditions: the program must run and the user login should be Manager or Staff only

Trigger: Add button is pressed

Post-conditions: Successful add new Manager account or Staff account or Teacher account or Student account or Class

Normal flow:

1. The use case begins when user start the program
2. Successfully login into the program by only Manager or Staff
3. The user click the Add button on the top left corner then it will pop up the small form so we can choose one type of account to do the add function.
4. It will show the type of form you choose above (Manager/Staff/Teacher/Student/Class)
5. Click the type Manager → The Add Form Manager will pop up
6. User must fill all text fields into the form (include validation)

   6.1 Manager Id: auto generate (Mx) ("M" is manager, x is the number of manager in the system)

   6.2 First Name : can not be blank

   6.3 Last Name: can't be blank

   6.4 DOB: user can choose their Date of Birth by using these combobox

   6.5 Gender: user can choose their Gender

   6.6 Home Phone: include Home Phone Code(combobox) and text field which is (xxxx-xxxx)

   6.7 Mobile Phone: include Mobile Phone Code (combobox) and text field which is (xxx-xxxx)

   6.8 Email: normal email (ex: rmit@edu.com)

   6.9 Address: can't be blank

   6.10 Status: auto Activate (Caused there's no reason to add the Deactivate Manager into the system)

   6.11 Username: can't not be blank should have at least 1 number (ex: managerx)

   6.12 Password: can't be blank (at least 1 uppercase, 1 lower Case, 1 special character)

   6.13 Retype password: must exactly the same with Password

   6.14 Browse: user can choose manager's image

   6.15 Description: description about the manager (can be empty)

7. If the user missing to full all the field above → It will pop the Warning message and show where the user need to fix by the red color. The blue color is correct and don't need to fix.
8. If the user click Cancel. The Add Manager will be close → return to the Quick Manage main From

9. The user click Add button to finish Add
10. Manager object will be saved into the manager file
11. The new Manager will display on the right scroll panel
12. The use case ends successfully

Exceptional flows:

2.1 Wrong user name or password when login into the program. Pop up will show up with the error message "Wrong username or password".

2.2 The user is deactivated. Pop up will show up with the error message "The user is deactivated"

6.1 Missing data inputs. Pop up will show "Please check the red input field"

6.2 Data validate input. Pop up will show "Please check the red input field"

8.1 The form doesn't close when click Cancel button

10.1 Cannot save data into files. Pop up will show "Cant save to manager File"

2.3.2 Add Staff

Use Case Name: Add Staff	ID: UC002
Actors: Staff	Priority: 1
Brief description: This use case describes how the User use the add function to add Staff account	
Pre-conditions: the program must run and the user login should be Manager or Staff only	
Trigger: Add button is pressed	
Post-conditions: Successful add new Staff account	

Normal flow:

1. The use case begins when user start the program
2. Successfully login into the program by only Manager or Staff
3. The user click the Add button on the top left corner then it will pop up the small form so we can choose one type of account to do the add function.
4. It will show the type of form you choose above (Manager/Staff/Teacher/Student/Class)
5. Click the type Staff → The Add Form Staff will pop up
6. User must fill all text fields into the form (include validation)

   6.1 Staff Id: auto generate (Tx) ("T" is manager, x is the number of manager in the system)

   6.2 First Name : cannot be blank

   6.3 Last Name: can't be blank

   6.4 DOB: user can choose their Date of Birth by using these combobox

   6.5 Gender: user can choose their Gender

6.6 Home Phone: include Home Phone Code(combobox) and text field which is (xxxx-xxxx)

6.7 Mobile Phone: include Mobile Phone Code (combobox) and text field which is (xxx-xxxx)

6.8 Email: normal email (ex: rmit@edu.com)

6.9 Address: can't be blank

6.10 Status: auto Activate (Caused there's no reason to add the Deactivate Staff into the system)

6.11 Username: can't not be blank should have at least 1 number (ex: managerx)

6.12 Password: can't be blank (at least 1 uppercase, 1 lower Case, 1 special character)

6.13 Retype password: must exactly the same with Password

6.14 Browse: user can choose image

6.15 Description: description about the staff (can be empty)

7. If the user missing to full all the field above → It will pop the Warning message and show where the user need to fix by the red color. The blue color is correct and don't need to fix.
8. If the user click Cancel. The Add Staff will be close → return to the Quick Manage main From
9. The user click Add button to finish Add
10. Staff object will be saved into the staff file
11. The new Staff will display on the right scroll panel
12. The use case ends successfully

Exceptional flows:

2.1 Wrong user name or password when login into the program. Pop up will show up with the error message "Wrong username or password".

2.3 The user is deactivated. Pop up will show up with the error message "The user is deactivated"

6.1 Missing data inputs. Pop up will show "Please check the red input field"

6.3 Data validate input. Pop up will show "Please check the red input field"

8.2 The form doesn't close when click Cancel button

10.2 Cannot save data into files. Pop up will show "Cant save to staff File"

2.3.3 Add Teacher

Use Case Name: Add Teacher	ID: UC003
Actors: Teacher	Priority: 1
Brief description: This use case describes how the User use the add function to Teacher account.	
Pre-conditions: the program must run and the user login should be Manager or Staff only	
Trigger: Add button is pressed	
Post-conditions: Successful add new Teacher account	

Normal flow:

11 The use case begins when user start the program

12 Successfully login into the program by only Manager or Staff

13 The user click the Add button on the top left corner then it will pop up the small form so we can choose one type of account to do the add function.

14 It will show the type of form you choose above (Manager/Staff/Teacher/Student/Class)

15 Click the type Teacher → The Add Form Teacher will pop up

16 User must fill all text fields into the form (include validation)

16.1+Teacher Id: auto generate (yymmxxx)

16.2+ First Name : can not be blank

16.3+Last Name: can't be blank

16.4+DOB: user can choose their Date of Birth by using these combobox

16.5+Gender: user can choose their Gender

16.6+Home Phone: include Home Phone Code(combobox) and text field which is (xxxx-xxxx)

16.7+Mobile Phone: include Mobile Phone Code (combobox) and text field which is (xxx-xxxx)

16.8+Email: normal email (ex: rmit@edu.com)

16.9+Address: can't be blank

16.10 +Status: auto Activate (Caused there's no reason to add the Deactivate Manager into the system)

16.11 +Skill: choose the teacher skills by click into small check box (9 skills : Piano, Organ, Violin, Singing, Ballet, Hip hop, Guitar, Painting, Photography)

16.12 +Browse: user can choose teacher's image

16.13 +Description: description about the teacher (can be empty)

17 If the user missing to full all the field above → It will pop the Warning message and show where the user need to fix by the red color. The blue color is correct and don't need to fix.

18 If the user click Cancel. The Add Teacher will be close → return to the Quick Manage main From

19 The user click Add button to finish Add

20 Teacher object will be saved into the teacher file

21 The new Teacher will display on the right scroll panel

13. The use case ends successfully

Exceptional flows:

2.1 Wrong user name or password when login into the program. Pop up will show up with the error message "Wrong username or password".

2.4 The user is deactivated. Pop up will show up with the error message "The user is deactivated"

6.1 Missing data inputs. Pop up will show "Please check the red input field"

6.4 Data validate input. Pop up will show “Please check the red input field”

8.3 The form doesn’t close when click Cancel button

21.1 Cannot save data into files. Pop up will show “Cant save to teacher File”

2.3.4 Add Student

Use Case Name: Add Student	ID: UC004
Actors: Manager, Staff	Priority: 1
Brief description: This use case describes how the User use the add function to add Student account .	
Pre-conditions: the program must run and the user login should be Manager or Staff only	
Trigger: Add button is pressed	
Post-conditions: Successful add new Student account	

Normal flow:

22 The use case begins when user start the program

23 Successfully login into the program by only Manager or Staff

24 The user click the Add button on the top left corner then it will pop up the small form so we can choose one type of account to do the add function.

25 It will show the type of form you choose above (Manager/Staff/Teacher/Student/Class)

26 Click the type Student → The Add Form Student will pop up

27 User must fill all text fields into the form (include validation)

27.1+Student Id: auto generate (yymmxxx)

27.2+ First Name : cannot be blank

27.3+Last Name: can’t be blank

27.4+DOB: user can choose their Date of Birth by using these combobox

27.5+Gender: user can choose their Gender

27.6+Home Phone: include Home Phone Code(combobox) and text field which is (xxxx-xxxx)

27.7+Mobile Phone: include Mobile Phone Code (combobox) and text field which is (xxx-xxxx)

27.8+Email: normal email (ex: rmit@edu.com)

27.9+Address: can’t be blank

27.10 +Status: auto Activate (Caused there’s no reason to add the Deactivate Manager into the system)

27.11 +Contact person Name: can’t be blank

27.12 +Contact person Mobile Phone : can’t be blank

27.13 +Contact person Email: can’t be blank

27.14 +Contact person Address: can’t be blank

27.15 +Relationship: choose the relationship with this student (by choose the element in the combobox :

27.16 Father, Mother, Sister, Brother, Other)

27.17 +Browse: user can choose image

27.18 +Description: description about the student (can be empty)

28 If the user missing to full all the field above → It will pop the Warning message and show where the user need to fix by the red color. The blue color is correct and don't need to fix.

29 If the user click Cancel. The Add Student will be close → return to the Quick Manage main From

30 The user click Add button to finish Add

31 Student object will be saved into the student file

32 The new Student will display on the right scroll panel

14. The use case ends successfully

Exceptional flows:

2.1 Wrong user name or password when login into the program. Pop up will show up with the error message "Wrong username or password".

2.5 The user is deactivated. Pop up will show up with the error message "The user is deactivated"

6.1 Missing data inputs. Pop up will show "Please check the red input field"

6.5 Data validate input. Pop up will show "Please check the red input field"

8.4 The form doesn't close when click Cancel button

32.1 Cannot save data into files. Pop up will show "Cant save to student File"

2.3.5 Add Class

Use Case Name: Add Class	ID: UC005
Actors: Class	Priority: 1
Brief description: This use case describes how the User use the add function to add Class.	
Pre-conditions: the program must run and the user login should be Manager or Staff only	
Trigger: Add button is pressed	
Post-conditions: Successful add new Class	

Normal flow:

33 The use case begins when user start the program

34 Successfully login into the program by only Manager or Staff

35 The user click the Add button on the top left corner then it will pop up the small form so we can

choose one type of account to do the add function.

36 It will show the type of form you choose above (Manager/Staff/Teacher/Student/Class)

37 Click the type Class → The Add Form Class will pop up

38 User must fill all text fields into the form (include validation)

38.1+Class Id: auto generate (Cx) ("C" is Class, x is the number of class in the system)

38.2+ Class Name : cannot be blank

38.3+Class Code: can't be blank

38.4+Start date: user can choose Start Date using these combobox Jcalender

38.5+End date: user can choose End Date using these combobox Jcalender

38.6+Tuition: can't be blank

38.7+Class Type: user can choose the type of the class by choose element in the combobox

38.8+Date/Time: user can choose the Date (Mon → Sun) , Time (8:00 → 19:30), Rooms

38.9(1 → 8)

38.10 +User can add student into that class by click the button "Add Student" -> pop up the new Form

38.11 +Click the check box to choose student then click "Add" button to add the student into that class

38.12 + User also can assign Teacher to that class by click "Assign Teacher" → It will pop up the choose Teacher Form. Tick the check box to choose the Teacher

38.13 +Click the button "Add" to finish assign teacher to the class

39 If the user missing to full all the field above → It will pop the Warning message and show where the user need to fix by the red color. The blue color is correct and don't need to fix.

40 If the user click Cancel. The Add Class will be close → return to the Quick Manage main From

41 The user click Add button to finish Add

42 Class object will be saved into the class file

43 The new Class will display on the right scroll panel

15. The use case ends successfully

Exceptional flows:

2.1 Wrong user name or password when login into the program. Pop up will show up with the error message "Wrong username or password".

2.6 The user is deactivated. Pop up will show up with the error message "The user is deactivated"

6.1 Missing data inputs. Pop up will show "Please check the red input field"

6.6 Data validate input. Pop up will show "Please check the red input field"

8.5 The form doesn't close when click Cancel button

43.1 Cannot save data into files. Pop up will show "Cant save to class File"

2.3.6 View Manager

Use Case Name: View Manager	ID: UC006
Actors: Manager	Priority: 2
Brief description: This use case describes how the User use the View function to view the detail of the Manager	
Pre-conditions: the program must run and the user login should be Manager or Staff only	
Trigger: View button is pressed	
Post-conditions: Successful view the detail information of the user's account or Class	
Normal flow: 1. The use case begins when user start the program 2. Successfully login into the program by only Manager or Staff 3. The user choose the type of user by click the combo box next to the Search bar 4. Click → Manager → It will display all Manager in the system in to the scroll pane 5. Choose any manager (The color will turn to orange if u choose that user) 6. The detail of information of that user/class will show on the left detail information panel 7. The use case ends successfully	
Exceptional flows: - Wrong user name or password when login into the program - The program can't start or crash during the process - Dont have any user or class to view	

2.3.7 View Staff

Use Case Name: View staff	ID: UC007
Actors: Staff	Priority: 2
Brief description: This use case describes how the User use the View function to view the detail of the Staff	
Pre-conditions: the program must run and the user login should be Manager or Staff only	
Trigger: View button is pressed	
Post-conditions: Successful view the detail information of the user's account or Class	
Normal flow: 8. The use case begins when user start the program 9. Successfully login into the program by only Manager or Staff	

10. The user choose the type of user by click the combo box next to the Search bar 11. Click → Staff → It will display all Staff in the system in to the scroll pane 12. Choose any Staff (The color will turn to orange if u choose that user) 13. The detail of information of that user/class will show on the left detail information panel 14. The use case ends successfully
Exceptional flows: - Wrong user name or password when login into the program - The program can't start or crash during the process - Dont have any user or class to view

2.3.8 View Teacher

Use Case Name: View Teacher	ID: UC008
Actors: Teacher	Priority: 2
Brief description: This use case describes how the User use the View function to view the detail of the Teacher	
Pre-conditions: the program must run and the user login should be Manager or Staff only	
Trigger: View button is pressed	
Post-conditions: Successful view the detail information of the user's account or Class	
Normal flow: 15. The use case begins when user start the program 16. Successfully login into the program by only Manager or Staff 17. The user choose the type of user by click the combo box next to the Search bar 18. Click → Teacher → It will display all Manager in the system in to the scroll pane 19. Choose any teacher (The color will turn to orange if u choose that user) 20. The detail of information of that user/class will show on the left detail information panel 21. The use case ends successfully	
Exceptional flows: - Wrong user name or password when login into the program - The program can't start or crash during the process - Dont have any user or class to view	

2.3.9 View Student

Use Case Name: View Student	ID: UC009
Actors: Student	Priority: 2

Brief description: This use case describes how the User use the View function to view the detail of the Student
Pre-conditions: the program must run and the user login should be Manager or Staff only
Trigger: View button is pressed
Post-conditions: Successful view the detail information of the user's account or Class
Normal flow: 22. The use case begins when user start the program 23. Successfully login into the program by only Manager or Staff 24. The user choose the type of user by click the combo box next to the Search bar 25. Click → Student → It will display all Manager in the system in to the scroll pane 26. Choose any student (The color will turn to orange if u choose that user) 27. The detail of information of that user/class will show on the left detail information panel 28. The use case ends successfully
Exceptional flows: - Wrong user name or password when login into the program - The program can't start or crash during the process - Dont have any user or class to view

2.3.10 View Class

Use Case Name: View Class	ID: UC010
Actors: Class	Priority: 2
Brief description: This use case describes how the User use the View function to view the detail of the Manager	
Pre-conditions: the program must run and the user login should be Manager or Staff only	
Trigger: View button is pressed	
Post-conditions: Successful view the detail information of the user's account or Class	
Normal flow: 29. The use case begins when user start the program 30. Successfully login into the program by only Manager or Staff 31. The user choose the type of user by click the combo box next to the Search bar 32. Click → Class → It will display all Manager in the system in to the scroll pane 33. Choose any class (The color will turn to orange if u choose that user) 34. The detail of information of that user/class will show on the left detail information panel	

35. The use case ends successfully
Exceptional flows: - Wrong user name or password when login into the program - The program can't start or crash during the process - Dont have any user or class to view

2.3.11 Edit Manager

Use Case Name: Edit Manager	ID: UC011
Actors: Manager	Priority: 3
Brief description: This use case shows how the User use the Edit function to edit user and class	
Pre-conditions: User has to log in as staff or manager. One user/class has to be chosen	
Trigger: Edit button is pressed	
Post-conditions: Successful edit the chosen user/class	
Normal flow: 1. Choose the type of the user want to edit (Click the combobox next to Search bar) 2. Select Manager → Choose any manager to edit (click the manager → the color turn to orange) 3. Click Edit button in the bottom of personal info 4. It will load the manager file. Find the exact Id and get all the information of this Id in the file then put all these information into the edit Form 5. The edit Form will look like the add Form but have filled information 6. Now we can change any text field in that form then click Add button to finish or click Cancel button if don't want to edit 7. User can't change the username when edit 8. Manager object will be saved into the manager file 9. The new information will auto save when Add button is clicked 10. The use case ends successfully	
Exceptional flows: • Invalid inputs • Cannot save new data • The program crash during the process • Don't have any user or class to edit	

2.3.12 Edit Staff

Use Case Name: Edit Staff	ID: UC012
Actors: Staff	Priority: 3

Brief description: This use case shows how the User use the Edit function to edit user and class
Pre-conditions: User has to log in as staff or manager. One user/class has to be chosen
Trigger: Edit button is pressed
Post-conditions: Successful edit the chosen user/class
Normal flow: 11. Choose the type of the user want to edit (Click the combobox next to Search bar) 12. Select Staff → Choose any manager to edit (click Staff → the color turn to orange) 13. Click Edit button in the bottom of personal info 14. It will load the Staff file. Find the exact Id and get all the information of this Id in the file then put all these information into the edit Form 15. The edit Form will look like the add Form but have filled information 16. Now we can change any text field in that form then click Add button to finish or click Cancel button if don't want to edit 17. User can't change the username when edit 18. Staff object will be saved into the Staff file 19. The new information will auto save when Add button is clicked 20. The use case ends successfully
Exceptional flows: • Invalid inputs • Cannot save new data • The program crash during the process • Don't have any user or class to edit

2.3.13 Edit Teacher

Use Case Name: Edit Teacher	ID: UC013
Actors: Teacher	Priority: 3
Brief description: This use case shows how the User use the Edit function to edit user and class	
Pre-conditions: User has to log in as staff or manager. One user/class has to be chosen	
Trigger: Edit button is pressed	
Post-conditions: Successful edit the chosen user/class	
Normal flow: 21. Choose the type of the user want to edit (Click the combobox next to Search bar) 22. Select Teacher → Choose any teacher to edit (click the teacher → the color turn to orange) 23. Click Edit button in the bottom of personal info	

24. It will load the teacher file. Find the exact Id and get all the information of this Id in the file then put all these information into the edit Form
25. The edit Form will look like the add Form but have filled information
26. Now we can change any text field in that form then click Add button to finish or click Cancel button if don't want to edit
27. Teacher object will be saved into the teacher file
28. The new information will auto save when Add button is clicked
29. The use case ends successfully

Exceptional flows:

- Invalid inputs
- Cannot save new data
- The program crash during the process
- Don't have any user or class to edit

2.3.14 Edit Student

Use Case Name: Edit Student	ID: UC014
Actors: Student	Priority: 3
Brief description: This use case shows how the User use the Edit function to edit user and class	
Pre-conditions: User has to log in as staff or manager. One user/class has to be chosen	
Trigger: Edit button is pressed	
Post-conditions: Successful edit the chosen user/class	

Normal flow:

30. Choose the type of the user want to edit (Click the combobox next to Search bar)
31. Select Student → Choose any student to edit (click the student → the color turn to orange)
32. Click Edit button in the bottom of personal info
33. It will load the student file. Find the exact Id and get all the information of this Id in the file then put all these information into the edit Form
34. The edit Form will look like the add Form but have filled information
35. Now we can change any text field in that form then click Add button to finish or click Cancel button if don't want to edit
36. Student object will be saved into the student file
37. The new information will auto save when Add button is clicked
38. The use case ends successfully

Exceptional flows:

- Invalid inputs
- Cannot save new data
- The program crash during the process
- Don't have any user or class to edit

2.3.15 Edit Class

Use Case Name: Edit Class	ID: UC015
Actors: Class	Priority: 3
Brief description: This use case shows how the User use the Edit function to edit user and class	
Pre-conditions: User has to log in as staff or manager. One user/class has to be chosen	
Trigger: Edit button is pressed	
Post-conditions: Successful edit the chosen user/class	

Normal flow:

39. Choose the type of the user want to edit (Click the combobox next to Search bar)
40. Select Class → Choose any class to edit (click the class → the color turn to orange)
41. Click Edit button in the bottom of personal info
42. It will load the class file. Find the exact Id and get all the information of this Id in the file then put all these information into the edit Form
43. The edit Form will look like the add Form but have filled information
44. Now we can change any text field in that form then click Add button to finish or click Cancel button if don't want to edit
45. Class object will be saved into class file
46. The new information will auto save when Add button is clicked
47. The use case ends successfully

Exceptional flows:

- Invalid inputs
- Cannot save new data
- The program crash during the process
- Don't have any user or class to edit

2.3.16 Delete Manager

Use Case Name: Delete Manager	ID: UC016
Actors: Manager	Priority: 4

Brief description: This use case describes how the User use the delete function to delete a user or class
Pre-conditions: User has to log in as Staff or Manager. One user/class has to be chosen
Trigger: Deleted button is pressed
Post-conditions: Successful delete 1 or many user/class
Normal flow: 1. Choose the type of user want to delete(Button next to the Search bar) 2. Choose Manager 3. Login by staff account can't delete manager account → The delete button will deactivated 4. Choose one or many Manager by tick the check box of each user 5. Click Delete button on the top main menu 6. It will get the Id of those selected and loop through the Manager file 7. Find the exact Id and delete that Item then save to the Manager file 8. The class you have ticked will be gone 9. The use case ends successfully
Exceptional flows: 3.1 The Delete button doesn't appear. 6.1 The item still appear when it deleted

2.3.17 Delete Staff

Use Case Name: Delete Staff	ID: UC017
Actors: Staff	Priority: 4
Brief description: This use case describes how the User use the delete function to delete a user or class	
Pre-conditions: User has to log in as Staff or Manager. One user/class has to be chosen	
Trigger: Deleted button is pressed	
Post-conditions: Successful delete 1 or many user/class	
Normal flow: 1. Choose the type of user want to delete(Button next to the Search bar) 2. Choose Staff 3. Choose one or many Staff by tick the check box of each user 4. Click Delete button on the top main menu 5. It will get the Id of those selected and loop through the Staff file 6. Find the exact Id and delete that Item then save to the Staff file	

7. The class you have ticked will be gone 8. The use case ends successfully
Exceptional flows: 3.1 The Delete button doesn't appear. 6.1 The item still appear when it deleted

2.3.18 Delete Teacher

Use Case Name: Delete Teacher	ID: UC018
Actors: Teacher	Priority: 4
Brief description: This use case describes how the User use the delete function to delete a user or class	
Pre-conditions: User has to log in as Staff or Manager. One user/class has to be chosen	
Trigger: Deleted button is pressed	
Post-conditions: Successful delete 1 or many user/class	
Normal flow: 1. Choose the type of user want to delete(Button next to the Search bar) 2. Choose Teacher 3. Choose one or many Teacher by tick the check box of each user 4. Click Delete button on the top main menu 5. It will get the Id of those selected and loop through the Teacher file 6. Find the exact Id and delete that Item then save to the Teacher file 7. The class you have ticked will be gone 8. The use case ends successfully	
Exceptional flows: 3.1 The Delete button doesn't appear. 6.1 The item still appear when it deleted	

2.3.19 Delete Student

Use Case Name: Delete Student	ID: UC019
Actors: Student	Priority: 4
Brief description: This use case describes how the User use the delete function to delete a user or class	
Pre-conditions: User has to log in as Staff or Manager. One user/class has to be chosen	

Trigger: Deleted button is pressed
Post-conditions: Successful delete 1 or many user/class
Normal flow: 1. Choose the type of user want to delete(Button next to the Search bar) 2. Choose Student 3. Choose one or many Student by tick the check box of each user 4. Click Delete button on the top main menu 5. It will get the Id of those selected and loop through the Student file 6. Find the exact Id and delete that Item then save to the Student file 7. The class you have ticked will be gone 8. The use case ends successfully
Exceptional flows: 3.1 The Delete button doesn't appear. 6.1 The item still appear when it deleted

2.3.20 Delete Class

Use Case Name: Delete Class	ID: UC020
Actors: Class	Priority: 4
Brief description: This use case describes how the User use the delete function to delete a user or class	
Pre-conditions: User has to log in as Staff or Manager. One user/class has to be chosen	
Trigger: Deleted button is pressed	
Post-conditions: Successful delete 1 or many user/class	
Normal flow: 9. Choose the type of user want to delete(Button next to the Search bar) 10. Choose Class 11. Choose one or many Classes by tick the check box of each user 12. Click Delete button on the top main menu 13. Pop up the message form "Delete one or many classes will automatically remove these classes for all students who have enrolled? Are you sure?" 14. Click Cancel. It will go back to the QuickManage main form 15. Click Yes. It will get the Id of those selected and loop through the Class file 16. Find the exact Id and delete that Item then save to the Class file	

17. The class you have ticked will be gone 18. The use case ends successfully
Exceptional flows: 3.1 The Delete button doesn't appear. 6.1 The item still appear when it deleted

2.3.21 Delete Class type

Use Case Name: Delete Class Type	ID: UC020
Actors: Class type	Priority: 4
Brief description: This use case describes how the User use the delete function to delete a user or class type	
Pre-conditions: User has to log in as Staff or Manager. One user/class has to be chosen	
Trigger: Deleted button is pressed	
Post-conditions: Successful delete 1 or many user/class	
Normal flow: 19. Choose the type of user want to delete(Button next to the Search bar) 20. Choose Class type 21. Choose one or many Classes by tick the check box of each user 22. Click Delete button on the top main menu 23. Pop up the message form "Delete one or many classes will automatically remove these classes for all students who have enrolled? Are you sure?" 24. Click Cancel. It will go back to the QuickManage main form 25. Click Yes. It will get the Id of those selected and loop through the Class type file 26. Find the exact Id and delete that Item then save to the Class type file 27. The class types you have ticked will be gone 28. The use case ends successfully	
Exceptional flows: 3.1 The Delete button doesn't appear. 6.1 The item still appear when it deleted	

2.3.22 Delete Room

Use Case Name: Delete Room	ID: UC020
Actors: Room	Priority: 4
Brief description: This use case describes how the User use the delete function to delete a user or room	
Pre-conditions: User has to log in as Staff or Manager. One user/class has to be chosen	
Trigger: Deleted button is pressed	
Post-conditions: Successful delete 1 or many user/class	
Normal flow: 29. Choose the type of user want to delete(Button next to the Search bar) 30. Choose Room 31. Choose one or many Classes by tick the check box of each user 32. Click Delete button on the top main menu 33. Click Cancel. It will go back to the QuickManage main form 34. Click Yes. It will get the Id of those selected and loop through the room file 35. Find the exact Id and delete that Item then save to the room file 36. The rooms you have ticked will be gone 37. The use case ends successfully	
Exceptional flows: 3.1 The Delete button doesn't appear. 6.1 The item still appear when it deleted	

2.3.23 Activate/Deactivate Manager

Use Case Name: Active/Deactive Student	ID: UC021
Actors: Manager	Priority: 5
Brief description: This use case describes how the User use the activate/deactivate function to active/deactivate one or many users at one time	
Pre-conditions: User has to log in as manager or staff. One or more accounts has to be chosen	
Trigger: Activate/Deactivate button has to be pressed	
Post-conditions: Chosen account(s) has to be activated/deactivated	
Normal flow: 1. Choose the type of user want to delete(Button next to the Search bar) 2. Choose Manager 3. Choose one or many Manager by tick the check box of each user	

4. Click the Activated button on the top menu
5. It will get the Id loop through the Manager file
6. And change the status of this user from (Activate → Deactivated or Deactivated → Activate)
7. Save to the Manager File
8. The user which is activate will have the green point, red is deactivate

Exceptional flows:

4.1 Click the Activate button but nothing happen

8.1 The View can't update the status of the user which is changed

8.2 Can't save to the file. It will pop up the warning message

8.3 Can't login by using the manager account which is deactivated

2.3.24 Activate/Deactivate Staff

Use Case Name: Active/Deactive Staff	ID: UC022
Actors: Staff	Priority: 5
Brief description: This use case describes how the User use the activate/deactivate function to active/deactivate one or many users at one time	
Pre-conditions: User has to log in as manager or staff. One or more accounts has to be chosen	
Trigger: Activate/Deactivate button has to be pressed	
Post-conditions: Chosen account(s) has to be activated/deactivated	

Normal flow:

1. Choose the type of user want to delete(Button next to the Search bar)
2. Choose Staff
3. Choose one or many Staff by tick the check box of each user
4. Click the Activated button on the top menu
5. It will get the Id loop through the Staff file
6. And change the status of this user from (Activate → Deactivated or Deactivated → Activate)
7. Save to the Staff File
8. The user which is activate will have the green point, red is deactivate

Exceptional flows:

4.1 Click the Activate button but nothing happen

8.1 The View can't update the status of the user which is changed

8.2 Can't save to the file. It will pop up the warning message

8.3 Can't login by using the staff account which is deactivated

2.3.25 Activate/Deactivate Teacher

Use Case Name: Active/Deactivate Teacher	ID: UC023
Actors: Teacher	Priority: 5
Brief description: This use case describes how the User use the activate/deactivate function to active/deactivate one or many users at one time	
Pre-conditions: User has to log in as manager or staff. One or more accounts has to be chosen	
Trigger: Activate/Deactivate button has to be pressed	
Post-conditions: Chosen account(s) has to be activated/deactivated	
Normal flow: 1. Choose the type of user want to delete(Button next to the Search bar) 2. Choose Teacher 3. Choose one or many Teacher by tick the check box of each user 4. Click the Activated button on the top menu 5. It will get the Id loop through the Teacher file 6. And change the status of this user from (Activate → Deactivated or Deactivated → Activate) 7. Save to the Teacher File 8. The user which is activate will have the green point, red is deactivate	
Exceptional flows: 22.1 Click the Activate button but nothing happen 22.2 The View can't update the status of the user which is changed 24.1 Can't save to the file. It will pop up the warning message	

2.3.26 Activate/Deactivate Student

Use Case Name: Active/Deactivate Student	ID: UC024
Actors: Student	Priority: 5
Brief description: This use case describes how the User use the activate/deactivate function to active/deactivate one or many users at one time	
Pre-conditions: User has to log in as manager or staff. One or more accounts has to be chosen	
Trigger: Activate/Deactivate button has to be pressed	
Post-conditions: Chosen account(s) has to be activated/deactivated	
Normal flow:	

9. Choose the type of user want to delete(Button next to the Search bar)
10. Choose Student
11. Choose one or many Student by tick the check box of each user
12. Click the Activated button on the top menu
13. It will get the Id loop through the Student file
14. And change the status of this user from (Activate → Deactivated or Deactivated → Activate)
15. Save to the Student File
16. The user which is activate will have the green point, red is deactivate

Exceptional flows:

22.1 Click the Activate button but nothing happen

22.2 The View can't update the status of the user which is changed

24.1 Can't save to the file. It will pop up the warning message

2.3.27 Assign Teacher to Class

Use Case Name: Assign Teacher to Class	ID: UC025
Actors: Manager/Staff	Priority:
Brief description: This use case describes how a user use the Assign function to assign the teacher to one to many classes	
Pre-conditions: User has to log in as staff or manager	
Trigger: Assign button has to be pressed	
Post-conditions: A class has to be successfully assign to a teacher	

Normal flow:

1. Choose the type of user want to delete(Button next to the Search bar)
2. Choose teacher
3. Tick a teacher on the scroll pane which want to assign (Tick the check box)
4. Click Assign button on the detail information panel on the left side
5. It will show all classes available in the Class list box then we can choose class by click on the time period show on the timetable
6. When finish the step above then the class you assigned will show on the Assigned list box
7. Click Close button to finish
8. The use case ends successfully

Exceptional flows:

- The program crash during the process
- Don't have any teacher or class to assign

2.3.28 Enroll the Student

Use Case Name: Enroll the Student	ID: UC026
Actors: Manager/Staff	Priority: 3
Brief description: This use case describes how the User use the Enroll function to enroll the student to the class	
Pre-conditions: User has to log in as staff or manager	
Trigger: Enroll button has to be presses	
Post-conditions: Successfully enroll the student to the class	
Normal flow: 1. Choose the type of user want to delete(Button next to the Search bar) 2. Choose Student 3. Tick a student which want to enroll on the Student scroll panel (Click the check box student) 4. Click Enroll button on the Student detail information on the left side 5. It will show the Enrollment Form and all classes available in the Class list box then you can click on any class to enroll 6. Click on the time period show on the timetable to enroll then it will show the class you already enrolled in the Enrolled list box 7. The use case ends successfully	
Exceptional flows: - Don’t have any class student to enroll - Class have the same time/clash	

2.3.29 Copy Class

Use Case Name: Copy Class	ID: UC027
Actors: Class	Priority: 3
Brief description: This use case describes how the User copy class by using Copy Class button	
Pre-conditions: User has to log in as staff or manager	
Trigger: Copy Class button has to be presses	
Post-conditions: Successfully copy class information	
Normal flow: 8. Choose the type of user want to delete(Button next to the Search bar) 9. Choose Class 10. Choose the class want to copy(tick the check box of that class) 11. Click Copy Class button	

12. It will get the class ID look through the Class file. Then create new class which have the same properties but have different ID,
13. Add into the Class File
14. The scroll pane will return the new Class

Exceptional flows:

11.1 Copy Class is licked but don't create any new Class

13.1 Can't save to the Class File. It will pop up the warning message

2.3.30 Search Manager

Use Case Name: Search Manager	ID: UC028
Actors: Manager	Priority: 3
Brief description: This use case describes how the User use the Search function to search information	
Pre-conditions: User has to log in as staff or manager	
Trigger: Search button has to be presses	
Post-conditions: Successfully search information for each type of user	

Normal flow:

15. Choose the type of user want to delete(Button next to the Search bar)
16. Choose Manager
17. Type any information user want to search except ID in the Search Bar (in the middle of the screen)
18. Click search button
19. The scroll pane will return the manager have that information

Exceptional flows:

- Can't search the default manager
- Can't search different type of user in one tab

2.3.31 Search Staff

Use Case Name: Search Staff	ID: UC029
Actors: Staff	Priority: 3
Brief description: This use case describes how the User use the Search function to search information	
Pre-conditions: User has to log in as staff or manager	
Trigger: Search button has to be presses	
Post-conditions: Successfully search information for each type of user	
Normal flow:	

20. Choose the type of user want to delete(Button next to the Search bar) 21. Choose Staff 22. Type any information user want to search except ID in the Search Bar (in the middle of the screen) 23. Click search button 24. The scroll pane will return the manager have that information
Exceptional flows: - Can't search the default manager - Can't search different type of user in one tab

2.3.32 Search Teacher

Use Case Name: Search Teacher	ID: UC030
Actors: Teacher	Priority: 3
Brief description: This use case describes how the User use the Search function to search information	
Pre-conditions: User has to log in as staff or manager	
Trigger: Search button has to be presses	
Post-conditions: Successfully search information for each type of user	
Normal flow: 25. Choose the type of user want to delete(Button next to the Search bar) 26. Choose Teacher 27. Type any information user want to search except ID in the Search Bar (in the middle of the screen) 28. Click search button 29. The scroll pane will return the manager have that information	
Exceptional flows: - Can't search the default manager - Can't search different type of user in one tab	

2.3.33 Search Student

Use Case Name: Search Student	ID: UC031
Actors: Student	Priority: 3
Brief description: This use case describes how the User use the Search function to search information	
Pre-conditions: User has to log in as staff or manager	
Trigger: Search button has to be presses	

Post-conditions: Successfully search information for each type of user
Normal flow: 30. Choose the type of user want to delete(Button next to the Search bar) 31. Choose Student 32. Type any information user want to search except ID in the Search Bar (in the middle of the screen) 33. Click search button 34. The scroll pane will return the manager have that information
Exceptional flows: - Can't search the default manager - Can't search different type of user in one tab

2.3.34 Search Class

Use Case Name: Search Class	ID: UC032
Actors: Class	Priority: 3
Brief description: This use case describes how the User use the Search function to search information	
Pre-conditions: User has to log in as staff or manager	
Trigger: Search button has to be presses	
Post-conditions: Successfully search information for each type of user	
Normal flow: 35. Choose the type of user want to delete(Button next to the Search bar) 36. Choose Class 37. Type any information user want to search except ID in the Search Bar (in the middle of the screen) 38. Click search button 39. The scroll pane will return the manager have that information	
Exceptional flows: - Can't search the default manager - Can't search different type of user in one tab	

2.3.35 Generate Invoice

Use Case Name: Generate Invoice	ID: UC033
Actors: Student	Priority: 3
Brief description: This use case describes how the Manager use the Invoice function to print invoice	

information of the student
Pre-conditions: User has to log in as manager or staff
Trigger: Invoice button has to be presses
Post-conditions: Successfully print invoice information for each student
Normal flow: 1) Choose type Student (Button next to the Search bar) 2) Choose the student want to print invoice (The color must change to orange) 3) Click the Invoice button in the bottom of student personal information → Pop up the invoice form 4) Classes: User can choose Paid or Unpaid by choose Radio button 5) It will automatically show the classes which student enrolled in and their total tuition fee 6) If the user click Cancel. It will go back to the QuickManage main form 7) If the manager click Save. It will automatic save this invoice into the invoice file 8) If the student didn't enroll into any class. Can't save the invoice 9) If the user click Print preview → pop up the view which show the invoice before print 10) If the user click History. If there is already have the invoice in the model. It will print out all of these Invoice 11) Manager can delete Invoice by changing from Paid to Unpaid 11.1 Click the History → Pop up all invoice 11.2 Click Delete → pop up the message to make sure the manager want to delete Invoice 12) Staff can't delete the invoice. However, staff can change the Invoice from UnPaid → Paid → It will pop up a Dialog → fill the paid date, paid method and paid note 13) It will automatic save to the invoice 14) User can print 1 Invoice per day
Exceptional flows: 8.1 If the student doesn't enroll into class can't save the invoice 8.2 Staff can't delete the invoice 13.1 Can't save to the invoice file. It will pop up the message box "Cant save to the invoice file"

2.3.36 View Report

Use Case Name: View Report	ID: UC034
Actors: Class	Priority: 3
Brief description: This use case describes how the User use the Report function to view report	
Pre-conditions: User has to log in as staff or manager	
Trigger: Report button has to be presses	

Post-conditions: View report of student which is paid
Normal flow: 1) Click the Report button(next to the button copy class) 2) Choose the Month and the year which user want to see (using combobox) 3) If the user click Cancel. It will go back to the Quick Manage main form 4) If the user click Report 4.1 If in that month and year, there is a paid or unpaid invoice, It will show all the invoice 4.2 If in that month and year, there is no invoice, It will show nothing
Exceptional flows: 4.1 Click button report it doesn't work

2.3.37 Payslip

Use Case Name: Payslip	ID: UC034
Actors: Class	Priority: 3
Brief description: This use case describes how the User use the Payslip	
Pre-conditions: User has to log in as staff or manager	
Trigger: Generate button has to be pressed	
Post-conditions: Teacher's class and payment have already been set.	
Normal flow: 1) Choose teacher from combo box of Control panel form 2) Click at the teacher and choose edit 3) Choose skill and fill in the hourly paid 4) Click at the generate button and click at the next generate button to make a report 5) To view each teacher payslip, choose a teacher for the form to load and click at the view payslip button on the left	
Exceptional flows: 4.1 Click button Generate doesn't work	

2.4 Non-Functional Requirements

2.4.1 Installability

- This program should be runed properly in any computer having Netbean 7.2 with JDK 1.6.

- Source codes should written in Java and able to run in NetBean 7.2

2.4.2 Usability

2.4.2.1) GUI:

- The GUI should be friendly and easy to use with menu bar and tool bar.
- The GUI shall optimize for a resolution of 1024x768 pixels.
- The user should be able to use correctly all the function which is provided in use guide within 5 minutes
-

2.4.2.2) Usage:

- The program should work on following hardware: Pentium 3, 1GB RAM, 1 GB hard disk space and following OS: Win XP/7, Fedora 17/18 or Ubuntu 12.04/12.10.

2.4.2.3) User Guide

- The program should display a User Guide in HTML format in the default browser (such as IE, Firefox Google Chrome, Safari) when the user click F1 hotkey or choose from menu bar of the program.
- The User Guide contains acknowledgements and detail instructions on how to install and how to use each function of the program.

2.4.3 Performance

2.4.3.1) Program data should be auto saved as binary format. The entire image should be store in the same folder that holds the program's executable file

- 2.4.3.2) Program should keep the information even when the application closed and load normally when opens it again. The program should be detect invalid inputs (check validation) and should not crash or stop working for any reasons.

2.4.4 Scalability

The program shall using MVC model to create the application with high cohesion and slow coupling. It should made good use of OO. Our program should have 3 mains package: Model , View, Controller. When producing the program, the group should use RUP,SVN, Unfuddle to keep track the process of the application.