SOFTWARE REQUIREMENT SPECIFICATION  
DOCUMENT

presence.if.its.ac.id

Intended for:

Teknik Informatika ITS

Prepared by:   
  
Muhammad Rifatullah 5114100118  
Aditya Ikhsan 5114100186

Informatics Department - Institut Teknologi Sepuluh Nopember

Kampus ITS KeputihSukolilo Surabaya

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Jurusan**  **Teknik Informatika ITS** | Nomor Dokumen | | Halaman |
| *SKPL-001* | | 1/#52 |
| Revisi | - | DD MM YYYY |

DAFTAR PERUBAHAN

|  |  |
| --- | --- |
| Revisi | Deskripsi |
| A |  |
| B |  |
| C |  |
| D |  |
| E |  |
| F |  |
| G |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| INDEX  TGL | - | A | B | C | D | E | F | G |
| Ditulis oleh |  |  |  |  |  |  |  |  |
| Diperiksa oleh |  |  |  |  |  |  |  |  |
| Disetujui oleh |  |  |  |  |  |  |  |  |

Daftar Halaman Perubahan

|  |  |  |  |
| --- | --- | --- | --- |
| Halaman | Revisi | Halaman | Revisi |
|  |  |  |  |

TABLE OF CONTENT

[1 Foreword 9](#_Toc466980674)

[1.1 Document Objective 9](#_Toc466980675)

[1.2 Problem Scope 9](#_Toc466980676)

[1.3 Definition and Term 9](#_Toc466980677)

[1.4 Naming and Numbering Rule 10](#_Toc466980678)

[1.5 Reference 10](#_Toc466980679)

[1.6 Document Summary 10](#_Toc466980680)

[2 Software General Description 11](#_Toc466980681)

[2.1 System Description 11](#_Toc466980682)

[2.2 User Characteristic 11](#_Toc466980683)

[2.3 Limitations 12](#_Toc466980684)

[2.4 Operational Environment 12](#_Toc466980685)

[3 Requirements Description 12](#_Toc466980686)

[3.1 External User Interface 12](#_Toc466980687)

[3.1.1 User Interface 12](#_Toc466980688)

[3.1.2 Hardware Interface 12](#_Toc466980689)

[3.1.3 Software Interface 13](#_Toc466980690)

[3.1.4 Communication Interface 13](#_Toc466980691)

[3.2 Functional Description 13](#_Toc466980692)

[3.2.1 Use Case Diagram 13](#_Toc466980693)

[3.2.2 Function 1: Filling presence by scanning KTM 13](#_Toc466980694)

[3.2.3 Function 2 : Check students’ own presence history 16](#_Toc466980695)

[3.2.4 Function 3: Edit user account 18](#_Toc466980696)

[3.2.5 Function 4: Add User Account 21](#_Toc466980697)

[3.2.6 Function 5: Delete User Account 24](#_Toc466980698)

[3.2.7 Function 6: See summary of presence in each class 27](#_Toc466980699)

[3.2.8 Funtion 7: Check every student presence history 28](#_Toc466980700)

[3.3 Class Description 31](#_Toc466980701)

[3.3.1 Class Diagram 31](#_Toc466980702)

[3.3.2 Class Description 32](#_Toc466980703)

[3.3.3 Entity Description 33](#_Toc466980704)

Daftar Tabel

[Table 1‑1 Naming and Numbering Rule 10](#_Toc466980876)

[Table 2‑1 User Characteristic 12](#_Toc466980877)

[Table 3‑1 Filling Presence Specification 14](#_Toc466980878)

[Table 3‑2 Use Case Specification Check students’ own presence history 17](#_Toc466980879)

[Table 3‑3 Use Case Specification Edit User Account 19](#_Toc466980880)

[Table 3‑4 Use Case Specification Add User Account 22](#_Toc466980881)

[Table 3‑5 Use Case Specification Delete User Account 25](#_Toc466980882)

[Table 3‑6 Use Case Specification See Summary of Presence in each class 28](#_Toc466980883)

[Table 3‑7 Use Case Specification Check Every Student Presence History 29](#_Toc466980884)

[Table 3‑8 Class Description 32](#_Toc466980885)

Daftar Gambar

[GAMBAR 3‑1 Use Case Diagram 13](#_Toc466980903)

[GAMBAR 3‑2 Activity Diagram: Filling Presence by Scanning KTM 15](#_Toc466980904)

[GAMBAR 3‑3 Check their own presence history activity diagram 17](#_Toc466980905)

[GAMBAR 3‑4 Edit user account activity diagram 20](#_Toc466980906)

[GAMBAR 3‑5 Add user account activity diagram 23](#_Toc466980907)

[GAMBAR 3‑6 Activity Diagram Delete User Account 26](#_Toc466980908)

[GAMBAR 3‑7 Activity Diagram See Summary of Presence in Each Class 28](#_Toc466980909)

[GAMBAR 3‑8 Check Every Student Presence History Activity Diagram 30](#_Toc466980910)

[GAMBAR 3‑9 Class Diagram 31](#_Toc466980911)

[GAMBAR 3‑11 Physical Data Model 33](#_Toc466980912)

# Foreword

## Document Objective

This document contains Software Requirement Specification for student presence system. The purpose of writing this document is to give detailed explanation about the software that is going to be built.

This document is going to be used as a foundation when developing the software and as an evaluation medium during the development process. Through this document, the development process is expected to be in order and detailed so that no ambiguity will arise for the developers of this software.

## Problem Scope

The student presence system in Informatics Department is a web based information system. This system is capable of doing the following stuff:

1. Store presence information in Informatics Department
2. Input media for verifying student presence in Informatics Department

This system is expected to make student easier to see their own presence history and make presence more original and validated.

## Definition and Term

Here are the following definition list and important terms that is used in this document:

* SRS : *Software Requirements Specification*, atau

SKPL : Spesifikasi Kebutuhan Perangkat Lunak

Dokumen hasil analisis yang berisi spesifikasi kebutuhan perangkat lunak.

* IEEE : *Institute of Electrical and Electronics Engineering*

Standar internasional untuk pengembangan dan perancangan produk.

* ANSI : *American National Standard Institute*

Lembaga Standardisasi Amerika.

* TBD : *To Be Defined*
* LAN : Local Area Network
* SIBES : Sistem Informasi Beasiswa

## Naming and Numbering Rule

This document uses a variation of naming and numbering for specific section. The naming and numbering is based on the table below

|  |  |
| --- | --- |
| **Hal/Bagian** | **Aturan Penomoran/Penamaan** |
| Functional Requirement | SKPL-FXX : Menunjukkan kebutuhan fungsional ke-XX |
| Non Functional Requirement | SKPL-NFXX : Menunjukkan kebutuhan non fungsional ke-XX |
| Functional Requirement Summary | SKPL-Fxxxdimana xxx adalah tiga digit bilangan bulat dimulai dari 000 |
| Non Functional Requirement Summary | SKPL-NFxxxdimana xxx adalah tiga digit bilangan bulat dimulai dari 000 |

Table ‑ Naming and Numbering Rule

## Reference

The following documents are used as a reference for this SRS

1. *Software Requirement Spesification Document (SRS) – IEEE* 1999 by Karl E. Wiegers.
2. Panduan Penggunaan dan Pengisian Spesifikasi Perangkat Lunak (SKPL), Jurusan Teknik Informatika, Institut Teknologi Sepuluh November.
3. Panduan Penggunaan dan Pengisian Spesifikasi Perangkat Lunak (SKPL), Jurusan Teknik Informatika, Institut Teknologi Bandung.

## Document Summary

Dokumen ini secara garis besar terdiri dari tiga bab dengan perincian sebagai berikut:

* Chapter 1 Foreword, discuss about the introduction of this SRS document. The content of the foreword are document objective, problem scope, definition and term, naming and numbering rule, reference and document summary
* Chapter 2 Software General Description. Discuss about the definition of the software and also the assumption and dependencies that is used in the making of this software
* Chapter 3 Functional Description. Discuss about the specific requirements of this software including user interface, functional requirement, non functional requirement, software attribute, performance requirement and other requirements from this software

# Software General Description

## System Description

Student presence system is a system that stores student presence information in Informatics Department and a media for student to input their presence. Inside this system, there are two users which are student and administration staff. Student has the right to access their own presence history and input presence by scanning their KTM on the scanner. Admin is responsible for importing student data and class from Integra to the system. This software can be accessed by all Informatics Department student and administration staff.

From student point of view, system can display their own presence history in each class. Besides that, this system is also a media for inputting presence by scanning students’ KTM

From admin point of view, system can handle data import from Integra to this system. Besides that admin can also edit, add and delete student information. Admin has the right to access all students’ presence history and the list of student who is already beyond the absent limit.

This system is equipped with access rights based on login data for security and safety of the system.

## User Characteristic

The characteristic of user from Presence.if.its.ac.id is explained on the table below:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Number** | **User Category** | **Job** | **Application access right** | **Minimum Ability** |
| 1. | Admin | Organize the system | Able to organize(add, edit, update) student data.  Able to view all student presence history | 1.Able to operate computer  2.Able to use web and database |
| 2. | Student | Accessing the system | Able to see their own presence history  Able to scan their KTM for presence input | 1.Able to operate computer  2.Able to use internet  3. Must own KTM |
| 3 | Lecturer | Organize class activity | Edit schedule, able to start and stop class | 1.Able to operate computer  2.Able to scan ID Card  3.Must own Lecturer Card |

Table ‑ User Characteristic

## Limitations

The development of presence.if.its.ac.id have the following limitations:

1. This system use HTML and PHP language
2. The user interface is in a form of simple design
3. From hardware point of view, there are some limitations such as limited storage and memory space and input only in a form of text and number
4. Other support software that is used are mysql workbench and sublime text 3

## Operational Environment

Presence.if.its.ac.id works on the following environment::

Operating system : Ubuntu

Operating system version : Ubuntu Server/14.04

DBMS : postgreSQL

Framework : Laravel (MVC)

# Requirements Description

## External User Interface

### User Interface

Presence.if.its.ac.iduses a GUI. User can input data by using keyboard and mouse and Windows operating system

### Hardware Interface

Presence.if.its.ac.id runs on a server. The ktm scanner is available on each class and connected through LAN cable to the server

### Software Interface

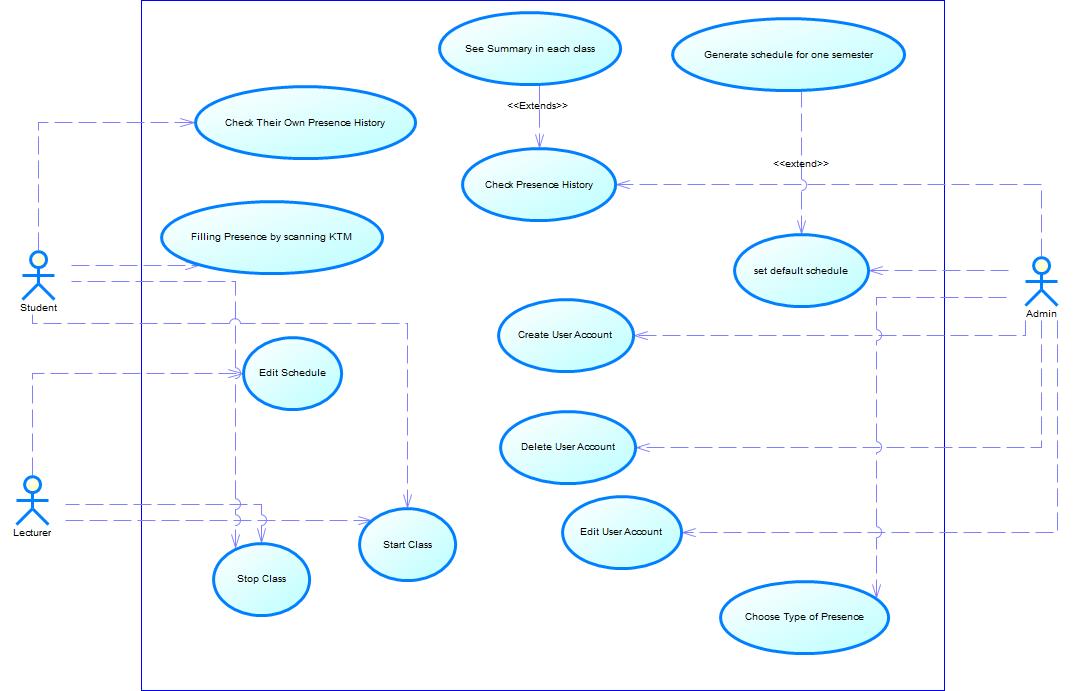
Presence.if.its.ac.id is a system that is built using HTML, PHP, postgreSQL and works on Ubuntu operating system

### Communication Interface

Presence.if.its.ac.id by using internet as the medium of communication

## Functional Description

### Use Case Diagram



GAMBAR ‑ Use Case Diagram

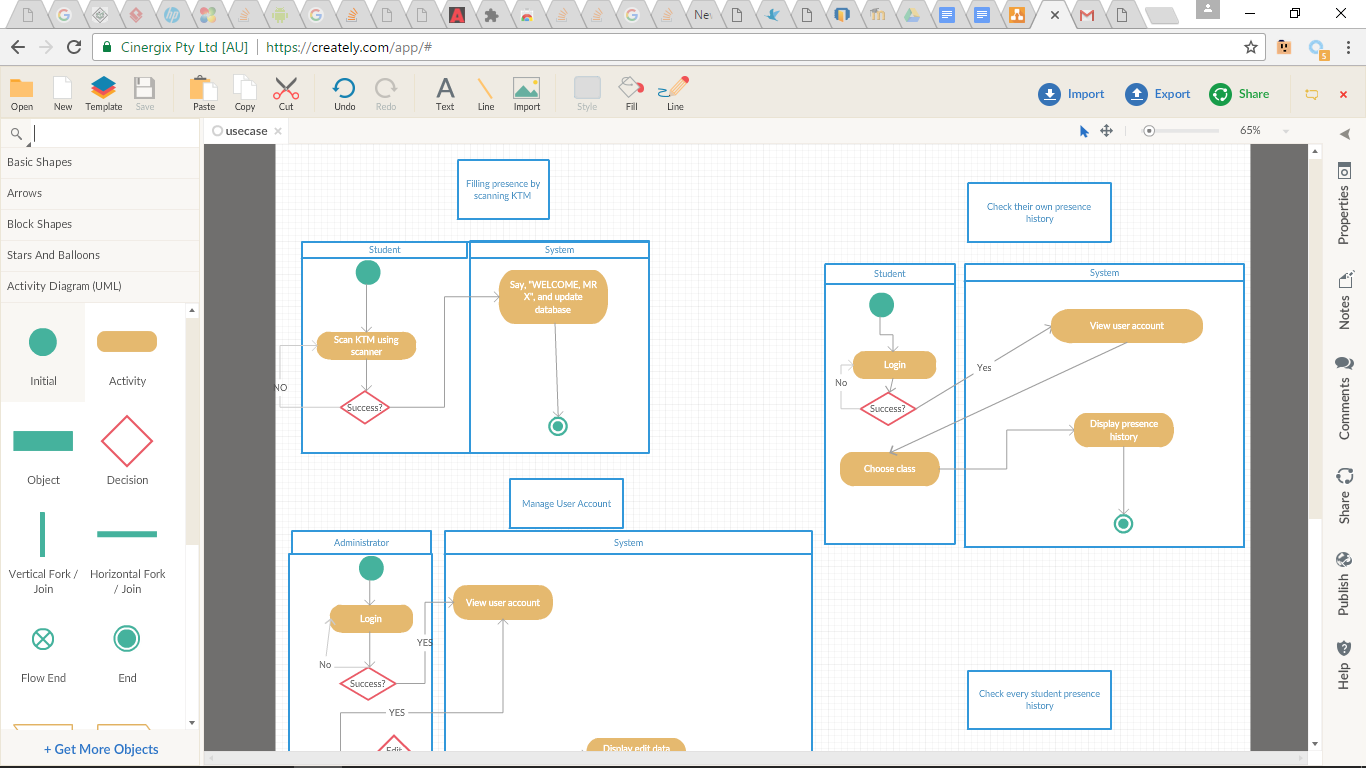
### Function 1: Filling presence by scanning KTM

**3.2.2.1 Scenario 1: Filling presence by scanning KTM**

|  |  |
| --- | --- |
| Use Case Name | Filling presence by scanning KTM |
| Description | Student have to scan their KTM in order to validate their presence |
| Use Case ID | UC01 |
| Relation | - |
| Primary Actor | Student |
| Secondary Actor | KTM Scanner |
| Initial state | Student have KTM |
| Final state | Scanning validation success or scanning validation failed |
|
| Main Flow | |
| Student | System |
| 1. Student scan their KTM in scanner | 2. KTM is validated  3. System update the database |
| Alternative Flow | |
| User is not registered in system | |
| 1. Student scan their KTM in scanner   3. Student cannot fill presence | 2. KTM validation failed |

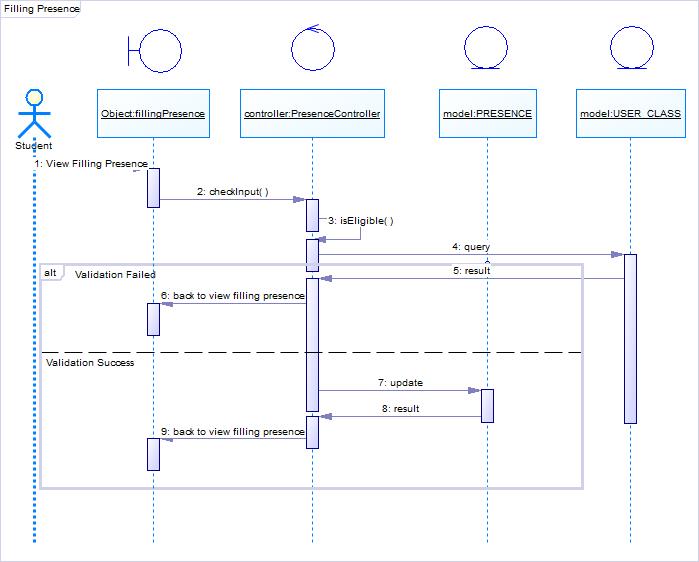
Table ‑ Filling Presence Specification

#### Activity Diagram: Filling Presence by Scanning KTM

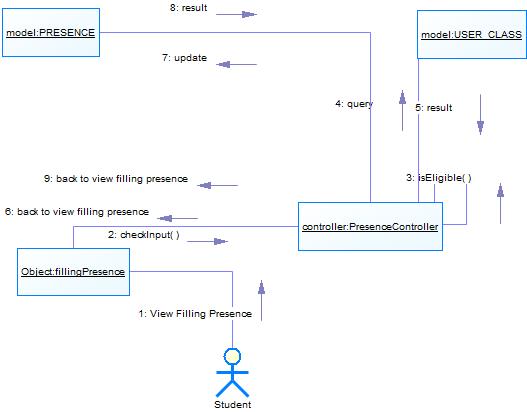


GAMBAR ‑ Activity Diagram: Filling Presence by Scanning KTM

#### Sequence Diagram: Filling Presence by Scanning KTM



#### Collaboration Diagram: Filling Presence by Scanning KTM



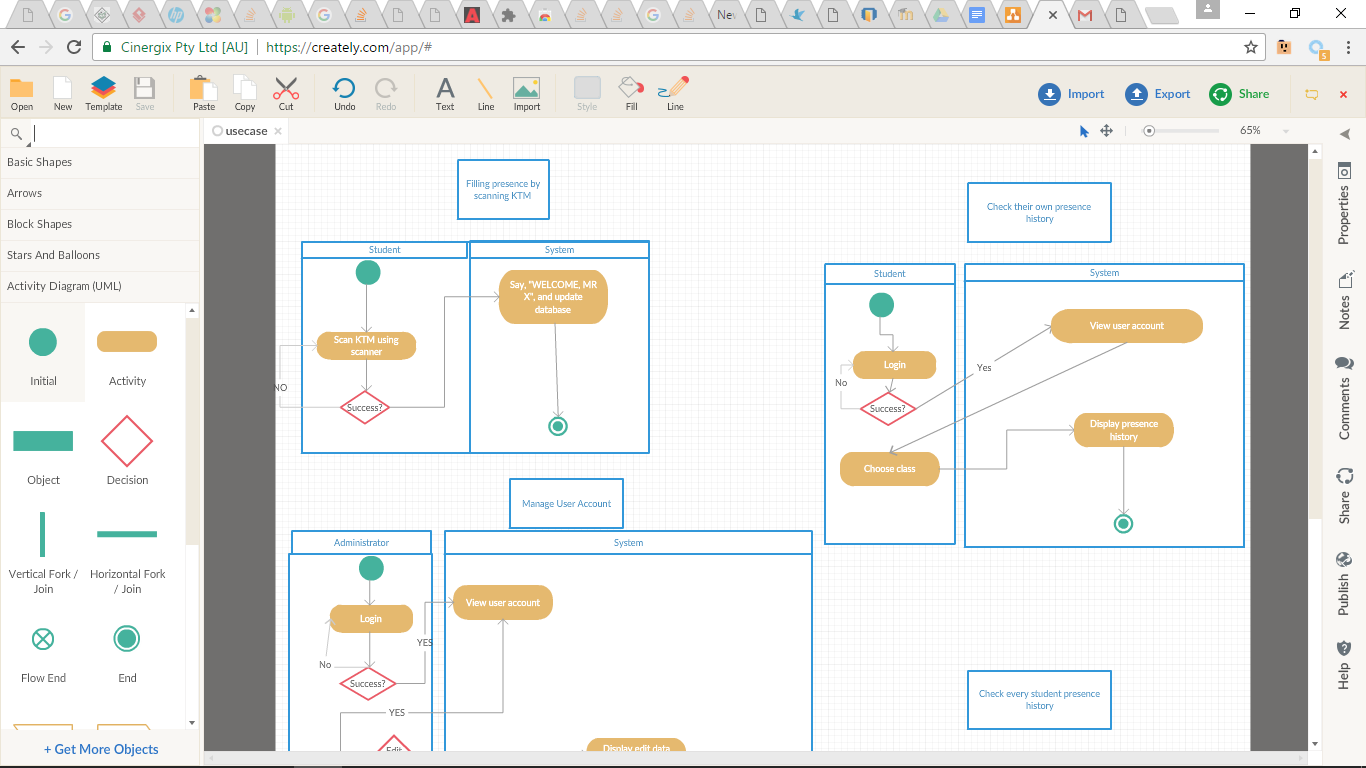
### Function 2 : Check students’ own presence history

#### Scenario: Check students’ own presence history

|  |  |
| --- | --- |
| Use Case Name | Check students’ own presence history |
| Description | Student can check their presence history from the classes that they attend |
| Use Case ID | UC02 |
| Relation | - |
| Primary Actor | Student |
| Secondary Actor | - |
| Initial Condition | Student have logged in |
| Final Condition | Student can check their own presence history |
|
| Main Flow | |
| Student | System |
| 1. Student login   3. Student pick class | 2. System validate login   1. Login succeed continue to view user account 2. Login failed go back to login screen   4. System display the student presence history |
| Alternative flow | |
| Login failed | |
| Student | System |
| 1a. Back to login form | 2. System validate login  a.Login failed go back to login screen |

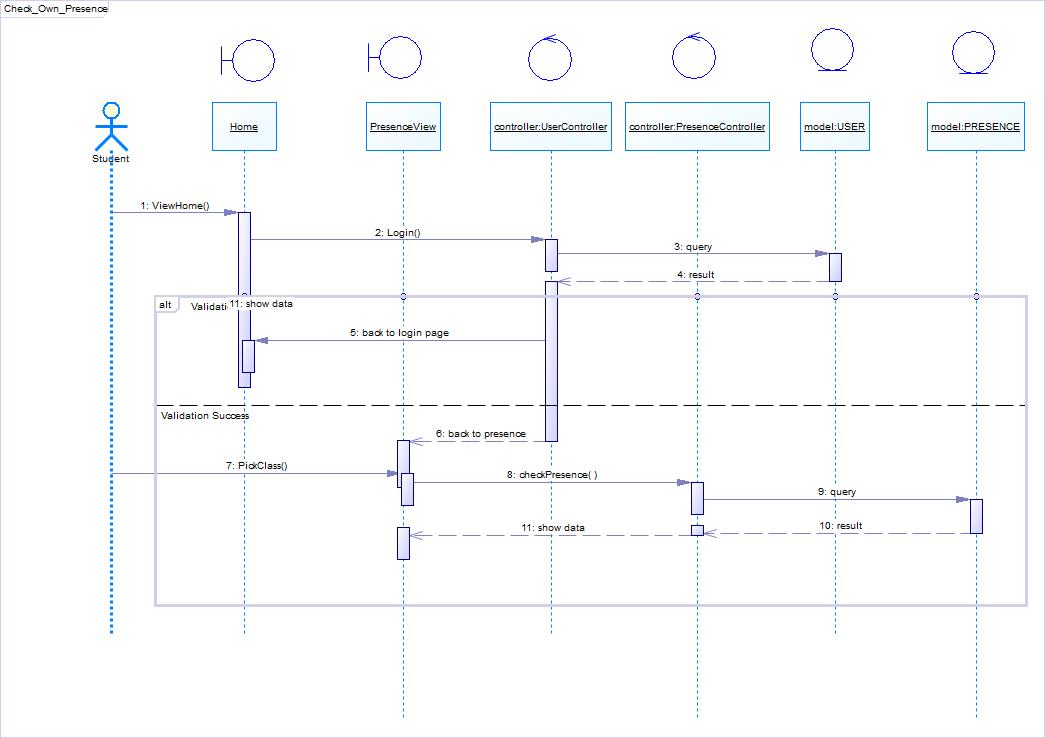
Table ‑ Use Case Specification Check students’ own presence history

#### Activity Diagram: Check Students’ own presence history

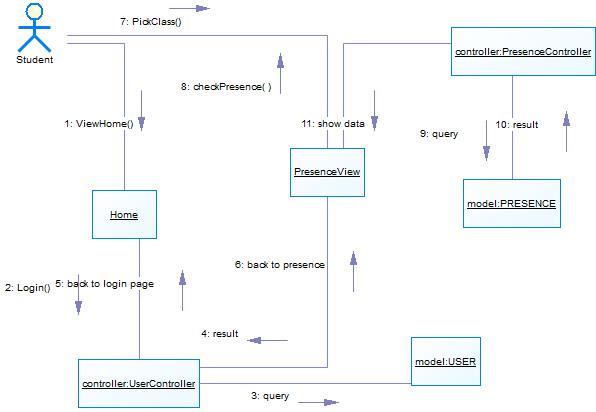


GAMBAR ‑ Check their own presence history activity diagram

#### Sequence Diagram: Check students’ own presence history



#### Collaboration Diagram: Check Students’ Presence History



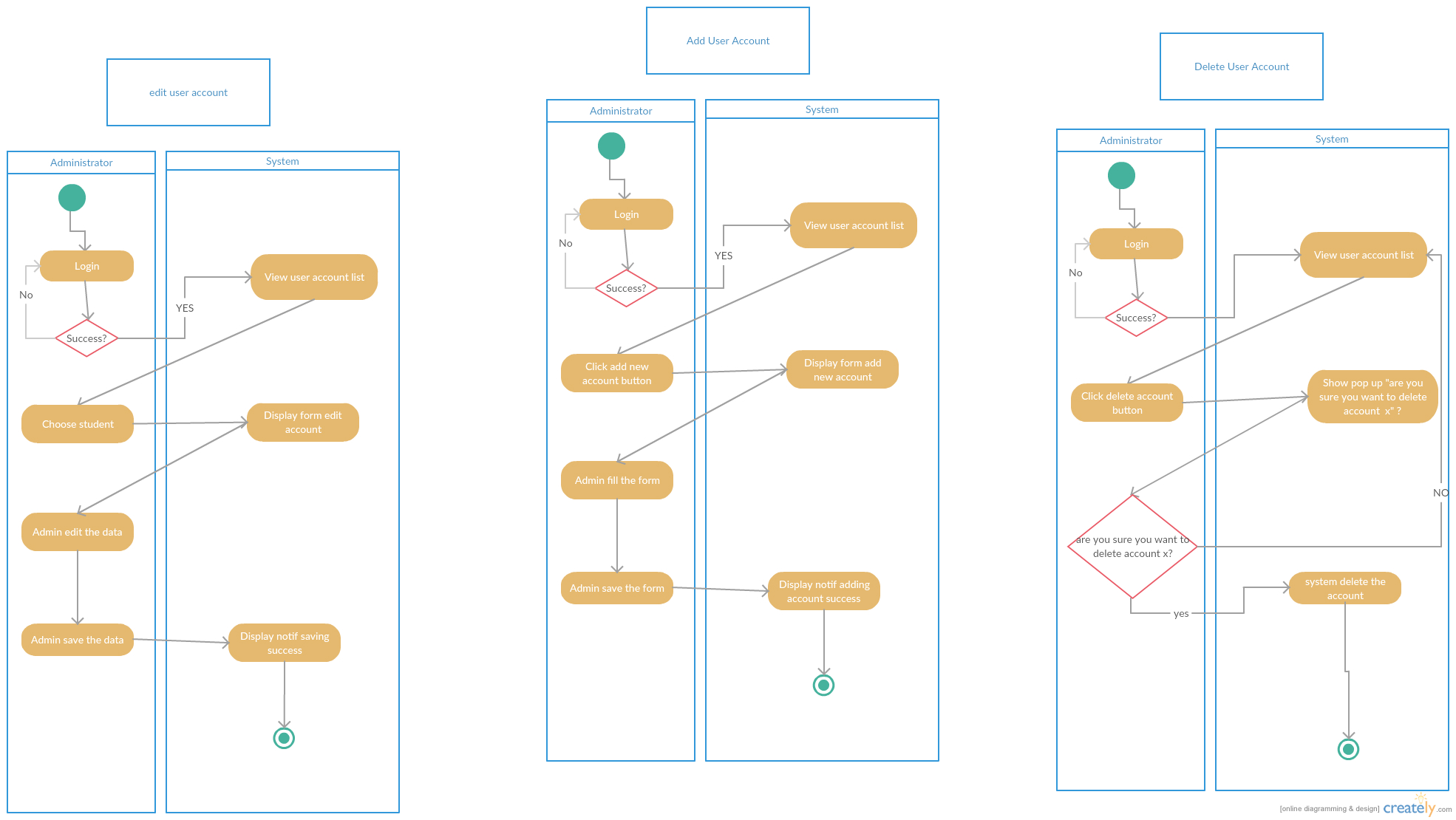
### Function 3: Edit user account

#### Scenario: Edit user account

|  |  |  |
| --- | --- | --- |
| Use Case Name | Edit User Account | |
| Description | Administrator is the one who is responsible for adding, editing and deleting student user account. Administrator is also responsible for assigning each student to their chosen class based on integra | |
| Use Case ID | UC03 | |
| Relation | - | |
| Primary Actor | Administrator | |
| Secondary Actor | - | |
| Initial Condition | Student have completed FRS | |
| Final Condition | Admin can add, edit or delete user account | |
|
| Main Flow | | |
| Administrator | System | |
| 1. Admin login to system  2. Admin choose student  4. Admin edit the data | 3. System show form editor  5. System update the data in database  6. System redirect to homepage | |
| Alternative Flow | | |
| Login failed | | |
| Admin | | System |
| 1a. Back to login form | | 2. System validate login  a.Login failed go back to login screen |

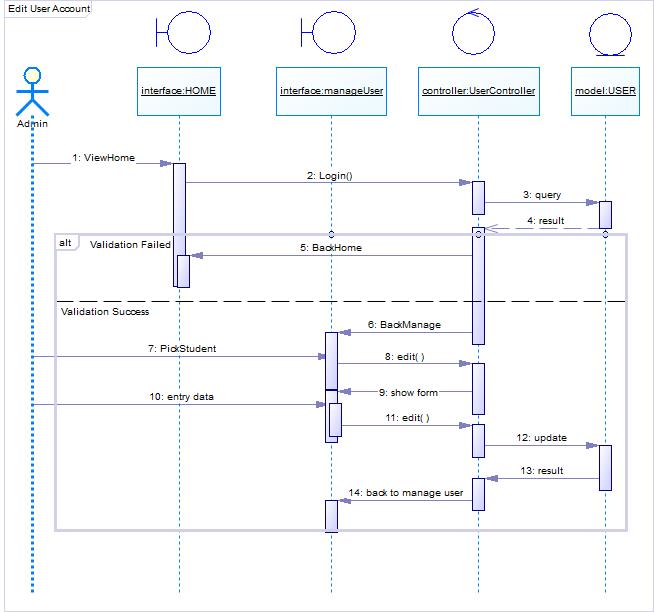
Table ‑ Use Case Specification Edit User Account

#### Activity Diagram: Edit User Account

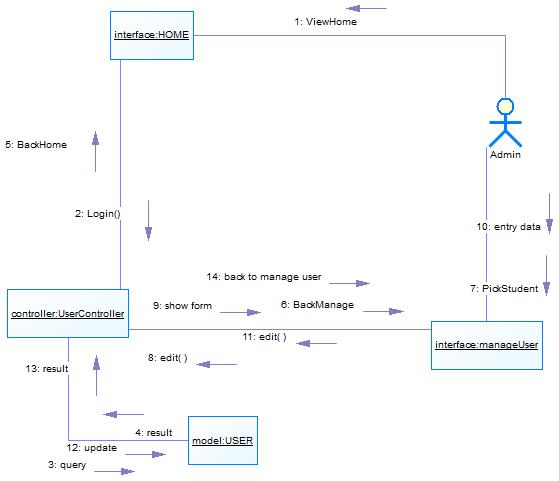


GAMBAR ‑ Edit user account activity diagram

#### Sequence Diagram: Edit User Account



#### Collaboration Diagram: Edit User Account



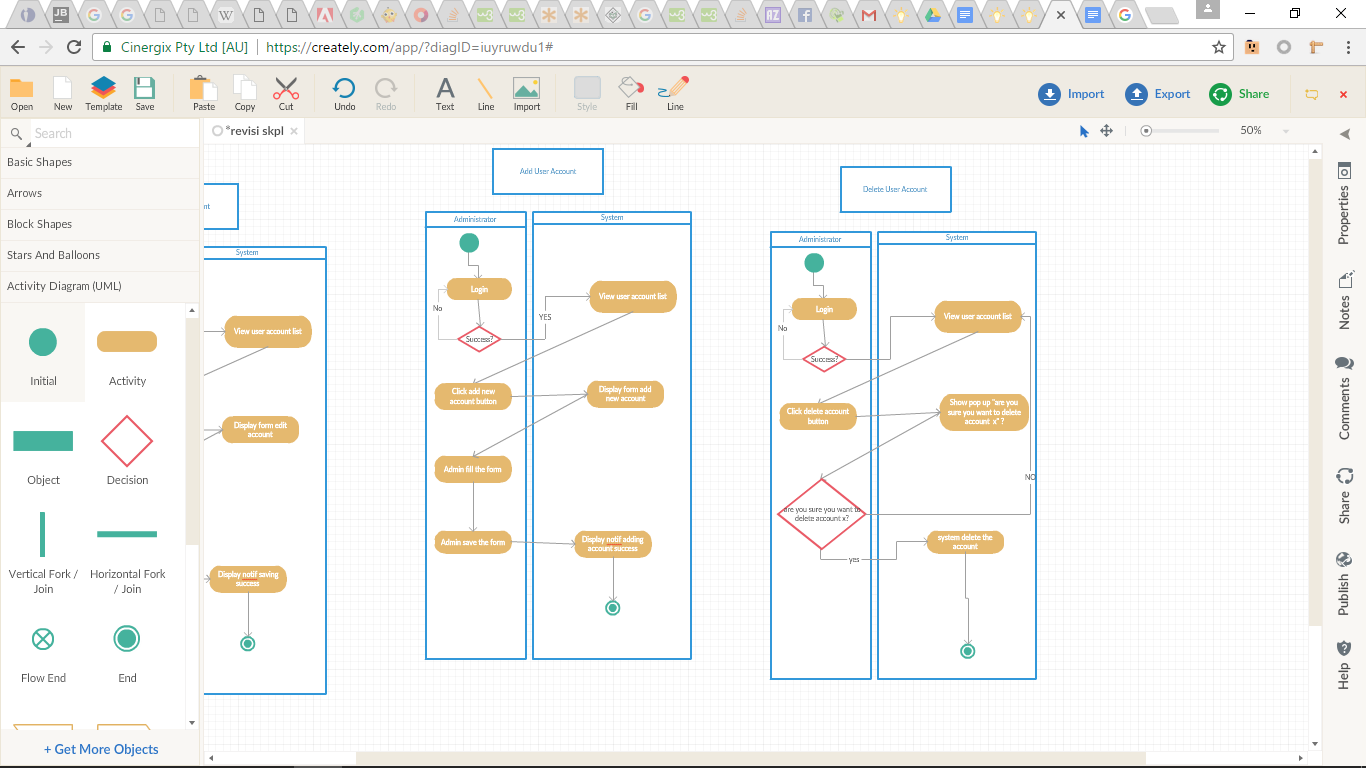
### Function 4: Add User Account

#### Scenario: Add User Account

|  |  |
| --- | --- |
| Use Case Name | Add User Account |
| Description | Administrator is the one who is responsible for adding, editing and deleting student user account. Administrator is also responsible for assigning each student to their chosen class based on integra |
| Use Case ID | UC04 |
| Relation | - |
| Primary Actor | Administrator |
| Secondary Actor | - |
| Initial Condition | Student have completed FRS |
| Final Condition | Admin can add user account |
|
| Main Flow | |
| Administrator | System |
| 1.admin login to Integra  2.Click add user  4. Input new data and save | 2. Show User List  3. System form editor  5. System update the database  6. System redirect to homepage |
| Alternative Flow | |
| Login Failed | |
| Admin | System |
| 1a. Back to login form | 2. System validate login  a.Login failed go back to login screen |

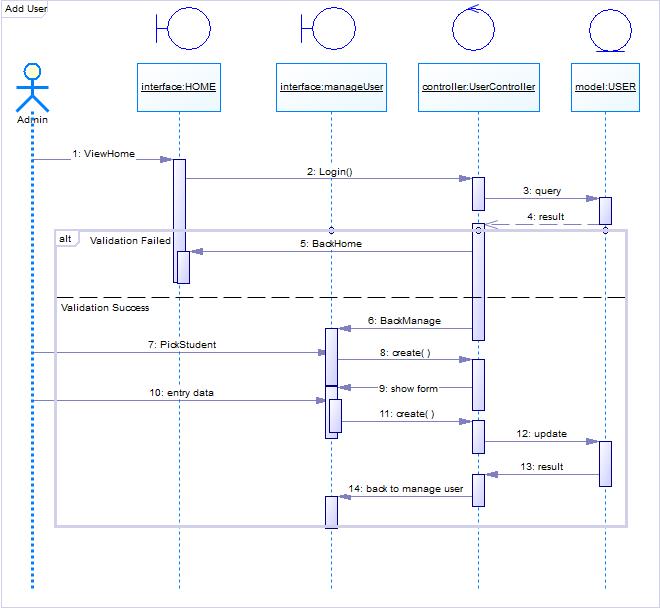
Table ‑ Use Case Specification Add User Account

#### Activity Diagram: Add user account

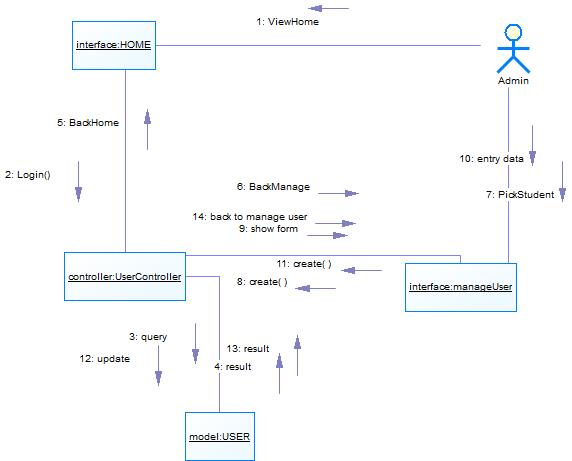


GAMBAR ‑ Add user account activity diagram

#### Sequence Diagram: Add User Account



#### Collaboration Diagram: Add User Account



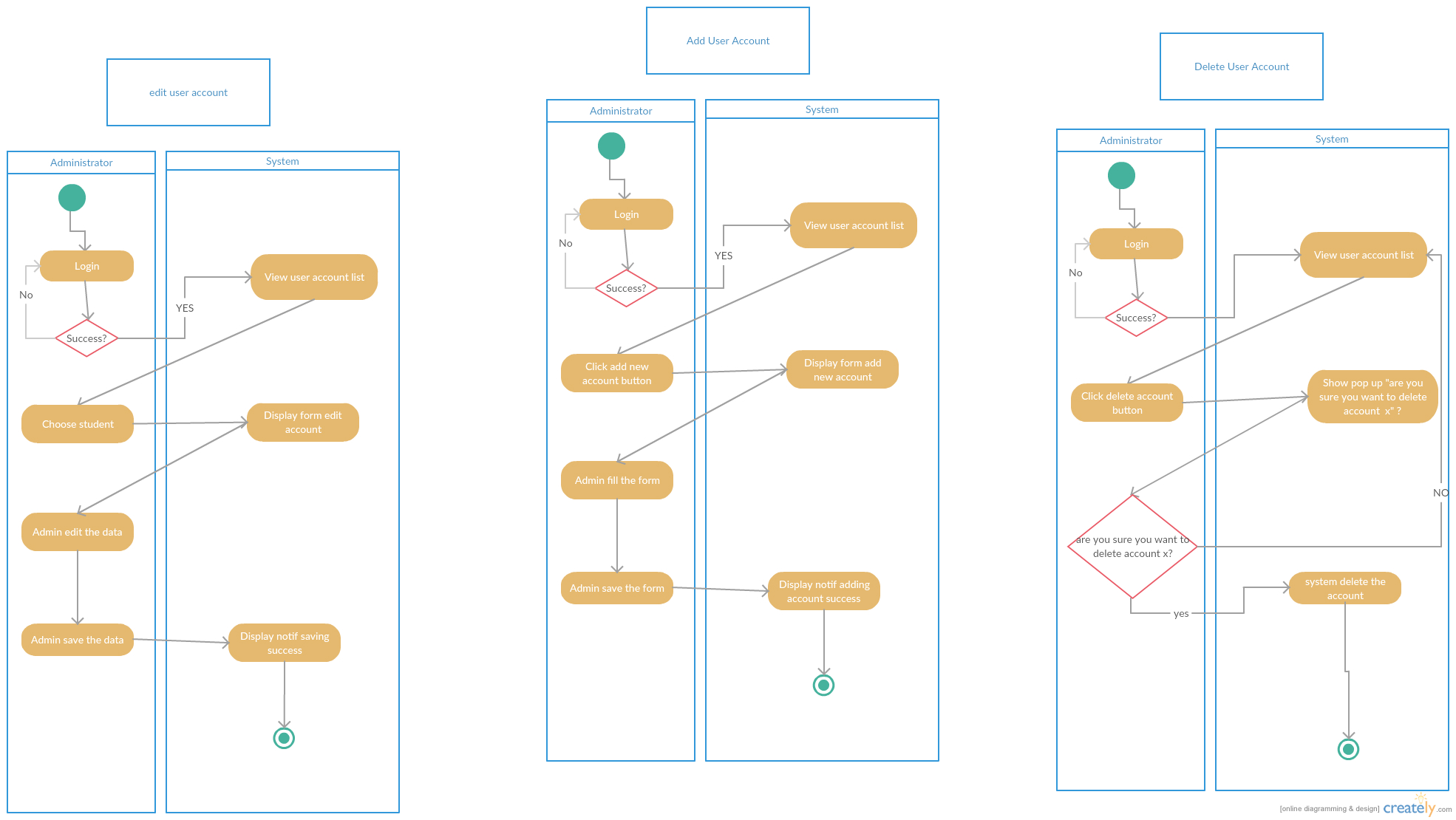
### Function 5: Delete User Account

#### Scenario: Delete User Account

|  |  |
| --- | --- |
| Use Case Name | Delete User Account |
| Description | Administrator is the one who is responsible for adding, editing and deleting student user account. Administrator is also responsible for assigning each student to their chosen class based on integra |
| Use Case ID | UC05 |
| Relation | - |
| Primary Actor | Administrator |
| Secondary Actor | - |
| Initial Condition | Student have completed FRS |
| Final Condition | Admin can delete user account |
|
| Main Flow | |
| Administrator | System |
| 1.Admin login to Integra  3.Admin click delete on selected account  5.User choose yes | 2. System show user list  4. System show pop up asking “are you sure you want to delete” ?  6. System delete the user  7. System redirect to homepage |
| Alternative Flow  5.1 Admin click no  5.2 System redirect to step 2 | |

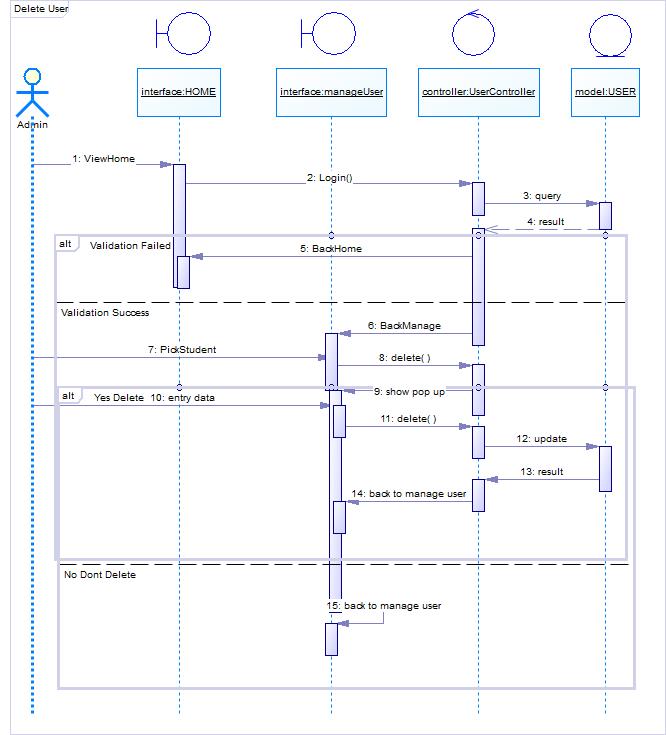
Table ‑ Use Case Specification Delete User Account

#### Activity Diagram: Delete user account

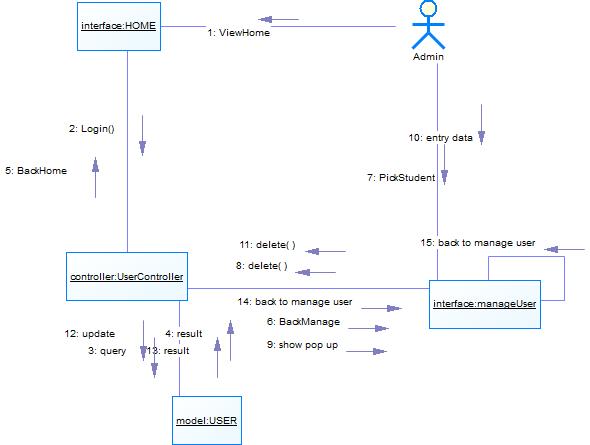


GAMBAR ‑ Activity Diagram Delete User Account

#### Sequence Diagram: Delete User Account



#### Collaboration Diagram: Delete User Account



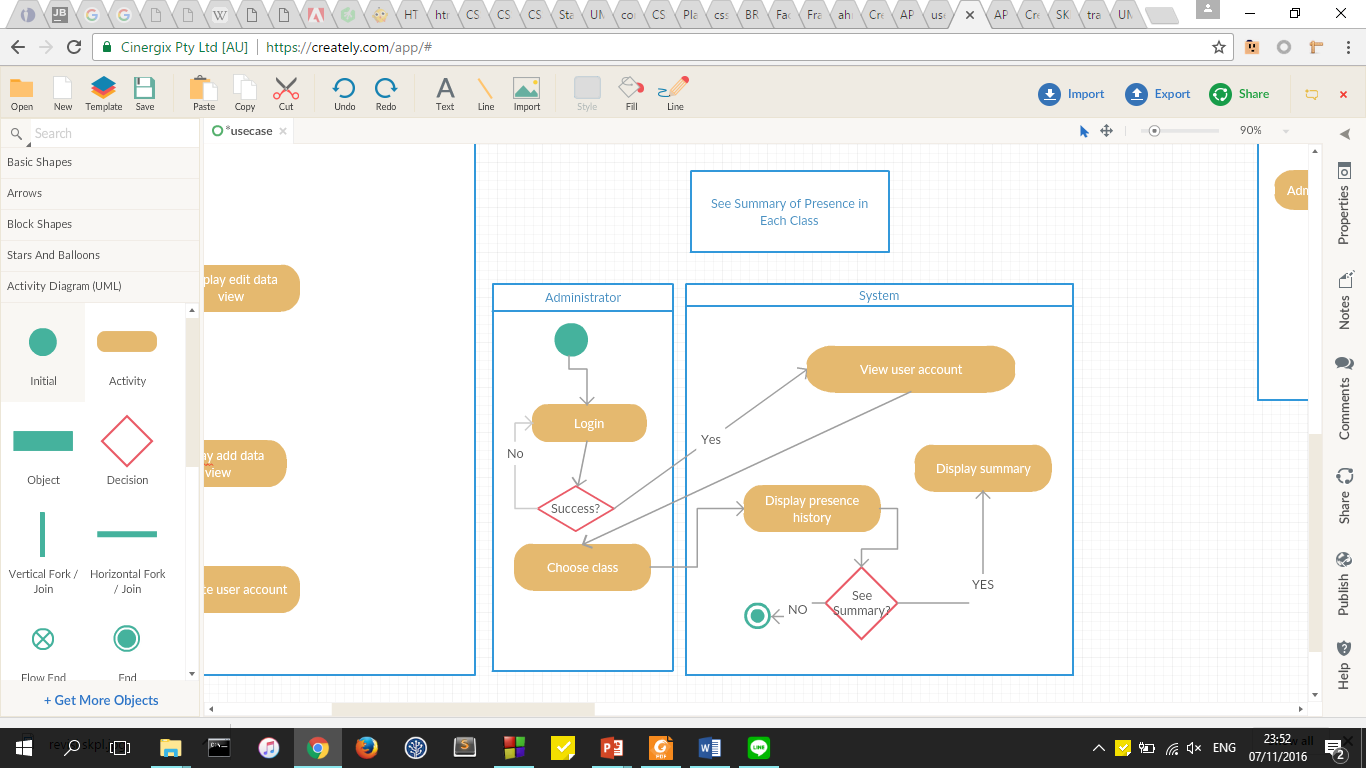
### Function 6: See summary of presence in each class

#### Scenario: See summary of presence in each class

|  |  |
| --- | --- |
| Use Case Name | See summary of presence in each class |
| Description | Administrator is the only one who can check the summary of presence of every class |
| Use Case ID | UC06 |
| Relation | Extend to usecase UC07 |
| Primary Actor | Administrator |
| Secondary Actor | - |
| Initial Condition | Student have input their presence using KTM |
| Final Condition | Admin can see the summary of each student in every class |
|
| Main Flow | |
| Administrator | System |
| 1. Admin login   3. Admin click summary link | 2. System validate login   1. Validation success 2. Validation failed   4. System display the summary of absent in each class |
| Alternative flow | |
| Username or password is not correct | |
| Administrator | System |
| 1. Admin login | 2. System validation failed. Return to login page |

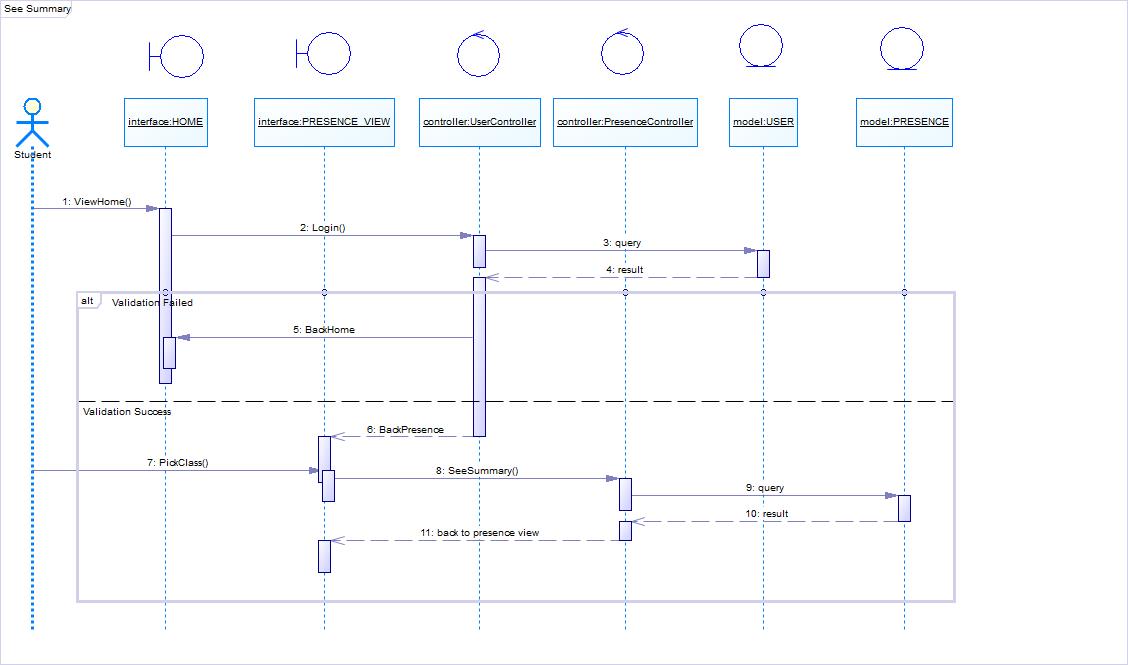
Table ‑ Use Case Specification See Summary of Presence in each class

#### Activity Diagram: See summary of presence in each class

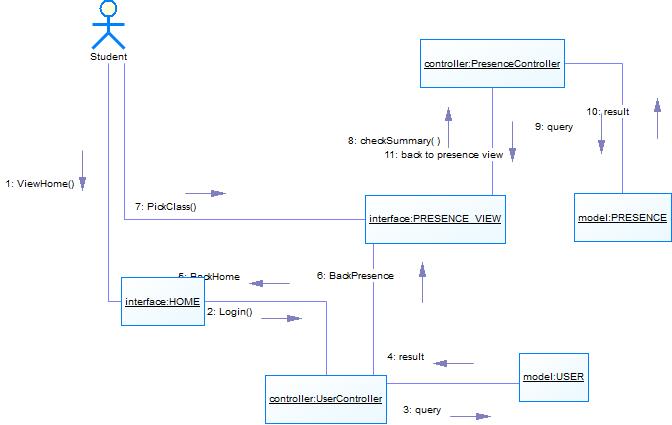


GAMBAR ‑ Activity Diagram See Summary of Presence in Each Class

#### Sequence Diagram: See Summary of Presence in Each Class



#### Collaboration Diagram: See Summary of Presence in Each Class



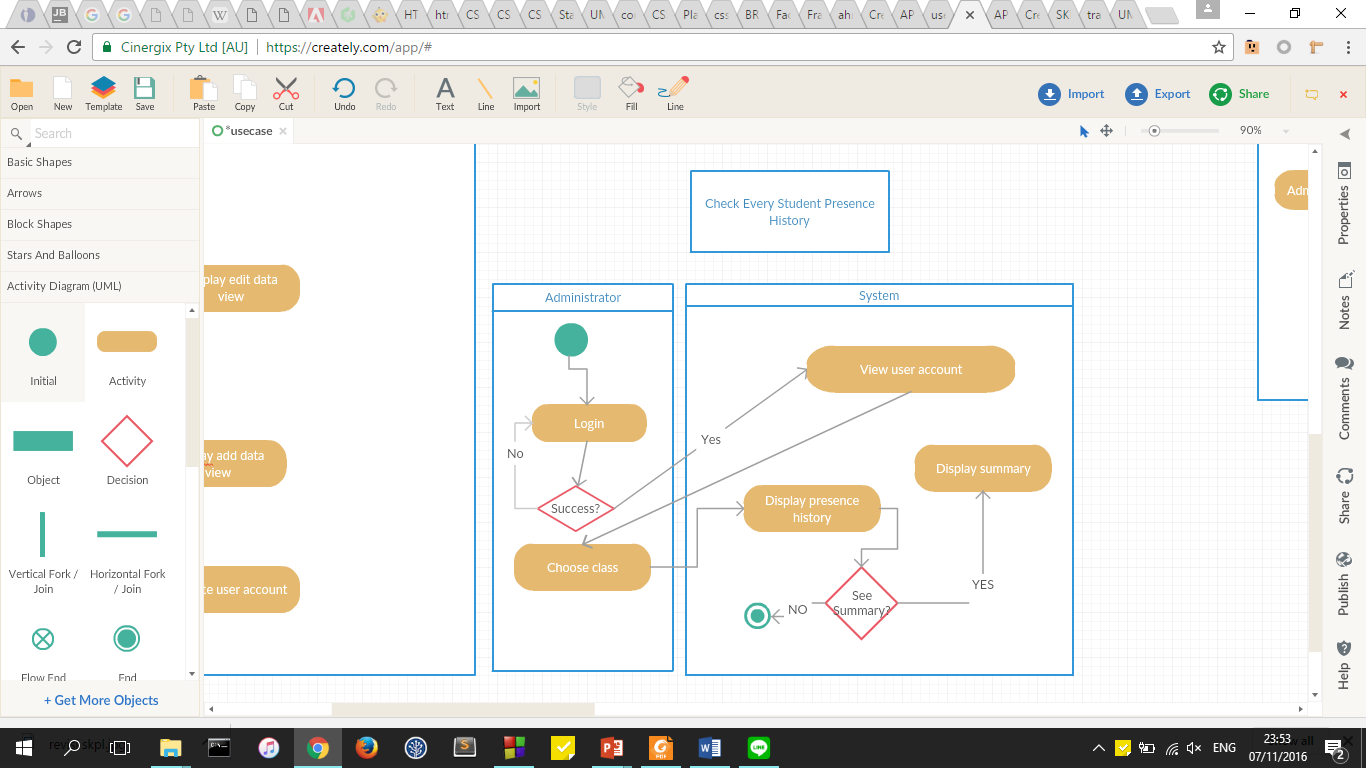
### Function 7: Check every student presence history

#### Scenario: Check every student presence history

|  |  |
| --- | --- |
| Use Case Name | Check every student presence history |
| Description | Student can check their presence history from the classes that they attend |
| Use Case ID | UC07 |
| Relation | - |
| Primary Actor | Administrator |
| Secondary Actor | - |
| Initial Condition | Admin have logged in |
| Final Condition | Admin can view the summary of each class |
|
| Main Flow | |
| Administrator | * System |
| 1. Admin login   3. Admin click summary for each student | 2. System validate login   1. Validation success 2. Validation failed   4. System display the summary of absent of each student |
| Alternative flow | |
| Username or password is not correct | |
| Administrator | System |
| 1. Admin login | 2. System validation failed. Return to login page |

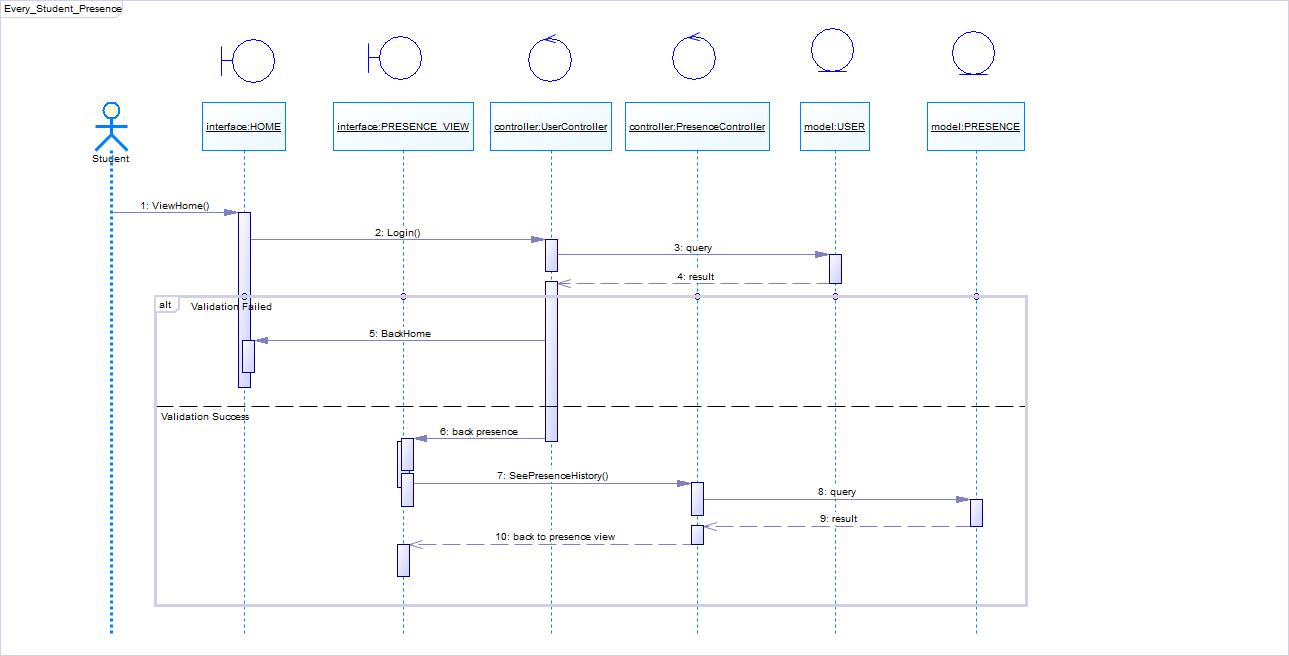
Table ‑ Use Case Specification Check Every Student Presence History

#### Activity Diagram: Check Every Student Presence History

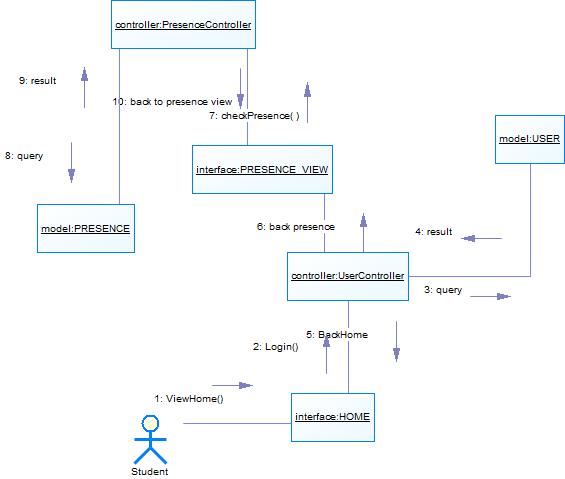


GAMBAR ‑ Check Every Student Presence History Activity Diagram

#### Sequence Diagram: Check Every Student Presence History



#### Collaboration Diagram: Check Every Student Presence History



### Function 8: Choose type of presence

#### Scenario: Choose Type of Presence

|  |  |
| --- | --- |
| Use Case Name | Choose Type of Presence |
| Description | Admin can select the type of presence according to the permission or condition of the student |
| Use Case ID | UC08 |
| Relation | - |
| Primary Actor | Administrator |
| Secondary Actor | - |
| Initial Condition | Admin have logged in |
| Final Condition | Admin can set the presence according to the selection |
|
| Main Flow | |
| Administrator | * System |
| 1. Admin login  3. Admin view student list  4. Admin choose student  6. Admin choose the presence option and save the data | 2. System validate login   1. Validation success 2. Validation failed   5. System display the presence option  7. System update database  8. System redirect to homepage |
| Alternative flow | |
| Username or password is not correct | |
| Administrator | System |
| 1. Admin login | 2. System validation failed. Return to login page |

Table ‑ Use case specification Choose type of presence

#### Activity Diagram: Choose Type of Presence

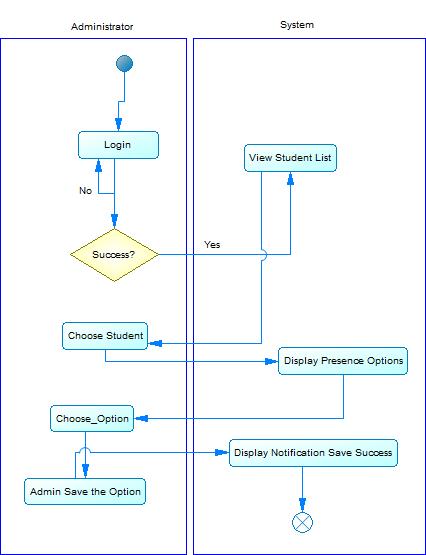
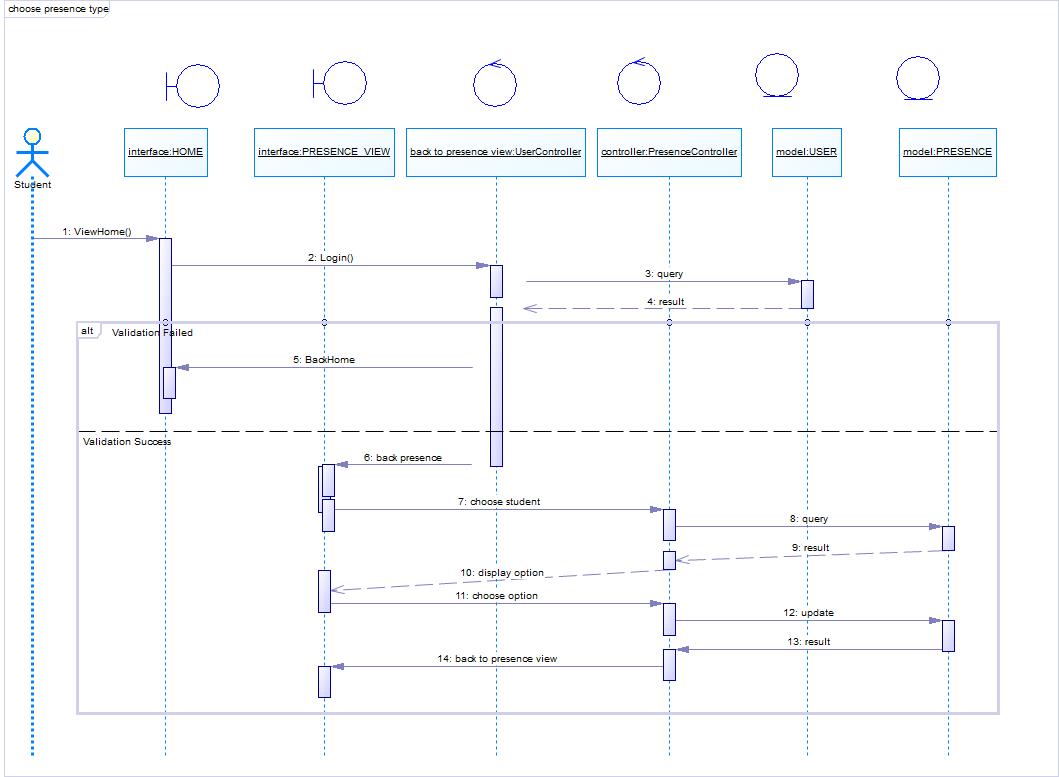
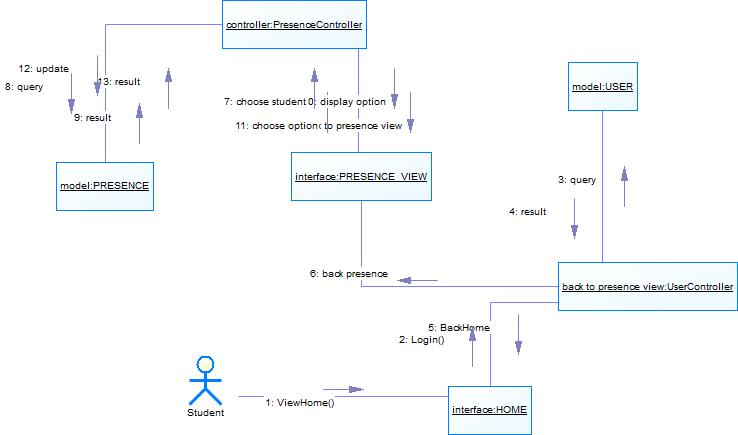


Table ‑ Choose Type of Presence Activity Diagram

**3.2.9.2: Sequence Diagram: Choose Type of Presence**



* + - 1. **Collaboration Diagram: Choose Type of Presence**



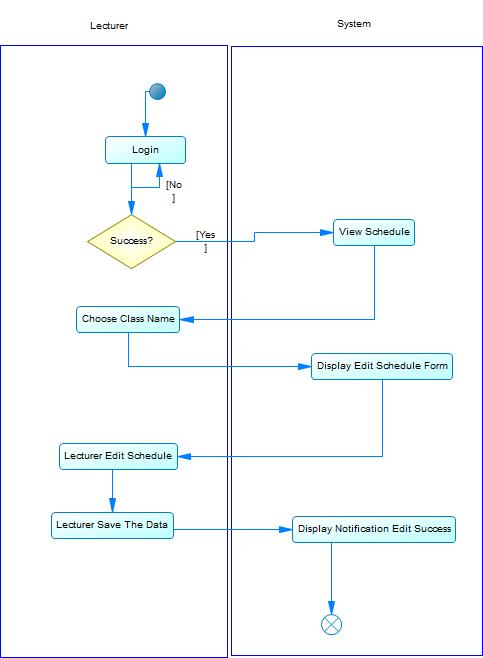
#### 

### Function 9 : Edit Schedule

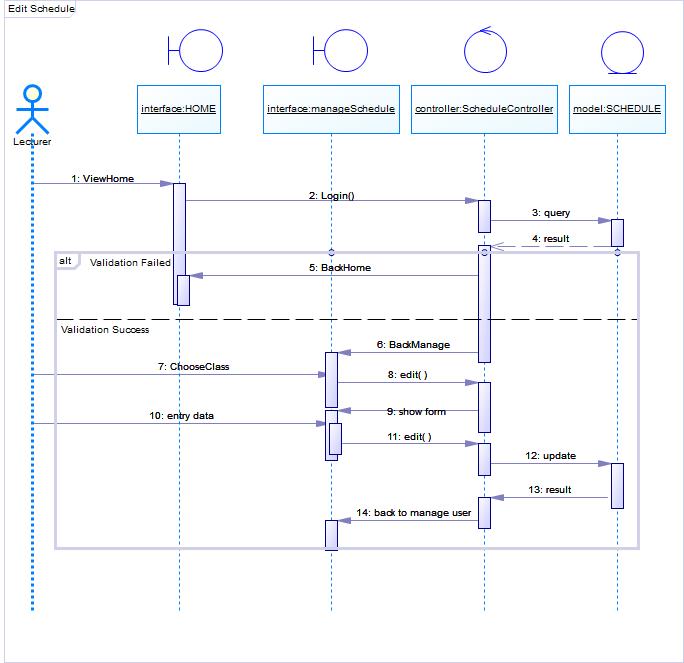
#### Scenario: Edit Schedule

|  |  |
| --- | --- |
| Use Case Name | Edit Schedule |
| Description | If the lecturer want to change the class schedule, they can edit the schedule according to their own time and class |
| Use Case ID | UC09 |
| Relation | - |
| Primary Actor | Lecturer |
| Secondary Actor | - |
| Initial Condition | Lecturer have logged in |
| Final Condition | Lecturer can edit the schedule |
|
| Main Flow | |
| Administrator | * System |
| 1. Lecturer login  2. Lecturer choose class  4. Lecturer edit the data | 3. System show edit schedule form  5. System update data in database  6. System redirect to homepage |
| Alternative flow | |
| Username or password is not correct | |
| Administrator | System |
| 1. Lecturer login | 2. System validation failed. Return to login page |

#### Activity Diagram: Edit Schedule

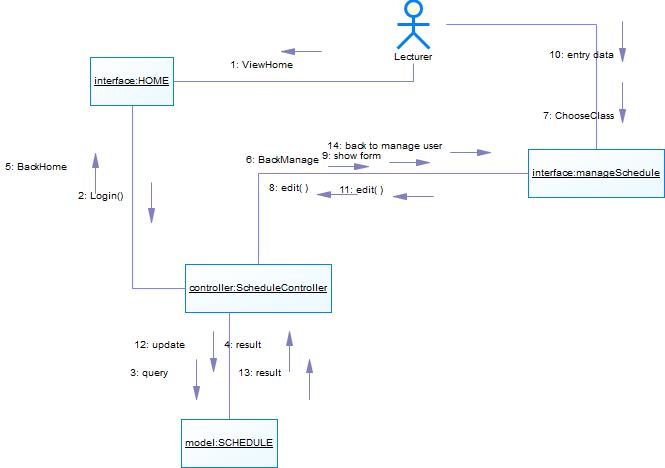


#### Sequence Diagram: Edit Schedule



3.2.10.3 Collaboration Diagram: Edit Schedule

#### Collaboration Diagram: Edit Schedule Collaboration Diagram: Choose Type of Presence

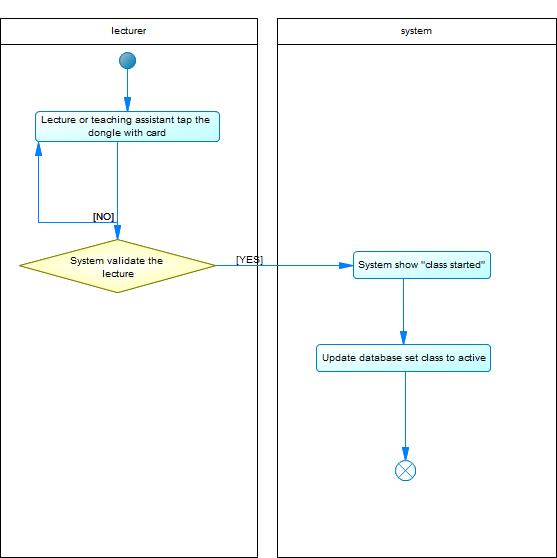


### Start Class

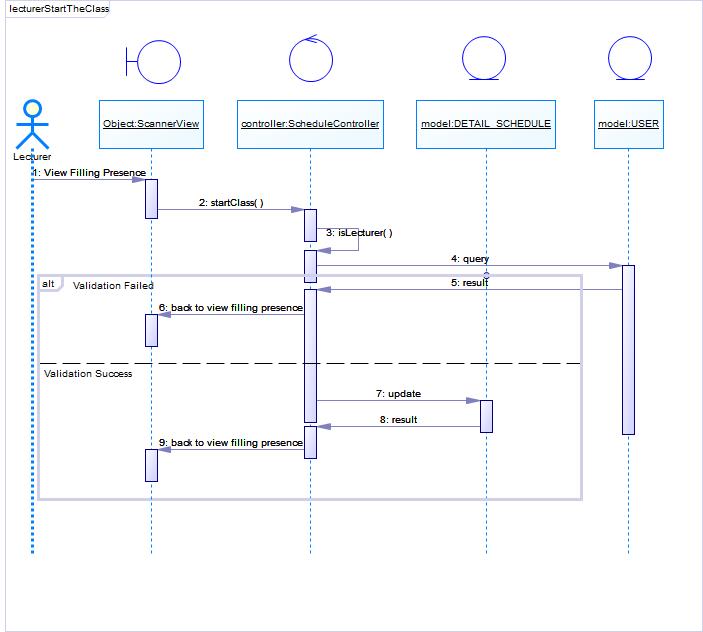
#### Scenario: Start Class

|  |  |
| --- | --- |
| Use Case Name | Start Class |
| Description | If the lecturer want to start the class, they need to tap their lecturer card in the scanner so that student can fill the presence |
| Use Case ID | UC10 |
| Relation | - |
| Primary Actor | Lecturer |
| Secondary Actor | Scanner |
| Initial Condition | Lecturer have lecturer card |
| Final Condition | Scanning validation success or scanning validation failed |
|
| Main Flow | |
| Lecturer | * System |
| 1. Lecturer scan lecturer card in scanner | 2. Lecturer card is validated  3. System update the database |
| Alternative flow | |
| User is not registered in system | |
| 1. Lecturer scan lecturer card  3. Lecturer cannot start the class | 2. Lecturer card validation failed |

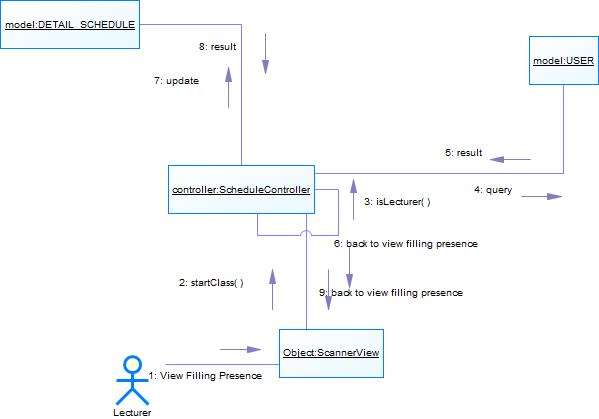
#### Activity Diagram: Start Class



#### Sequence Diagram: Start Class



#### Collaboration Diagram: Start Class

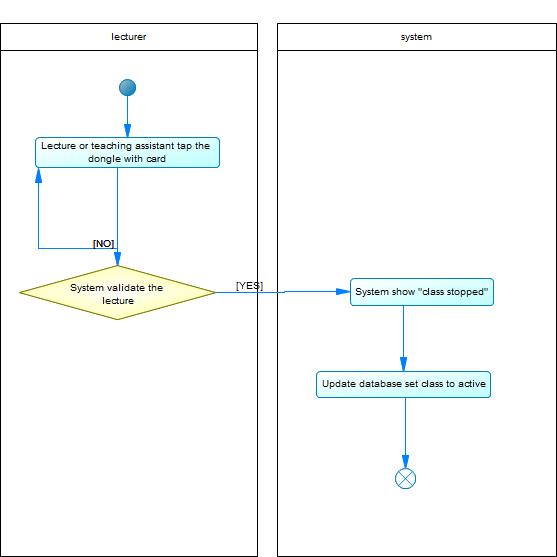


### Stop Class

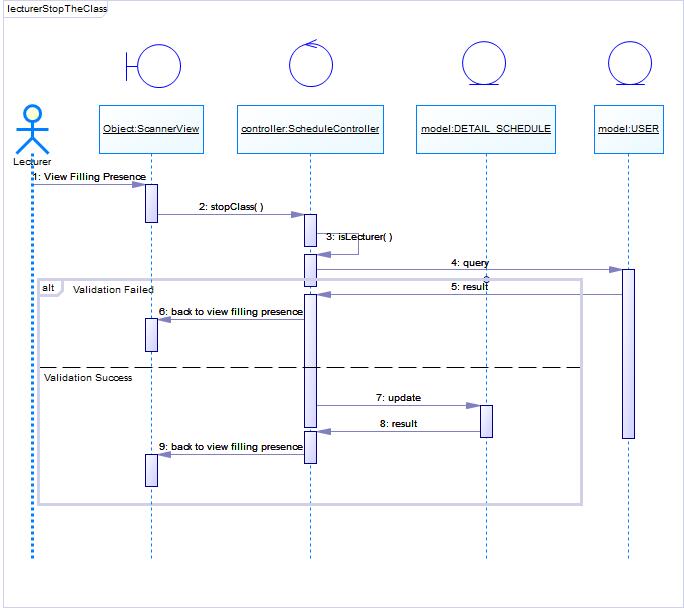
#### Scenario: Stop Class

|  |  |
| --- | --- |
| Use Case Name | Stop Class |
| Description | If the lecturer want to start the class, they need to tap their lecturer card in the scanner so that student can fill the presence |
| Use Case ID | UC11 |
| Relation | - |
| Primary Actor | Lecturer |
| Secondary Actor | Scanner |
| Initial Condition | Lecturer have lecturer card |
| Final Condition | Scanning validation success or scanning validation failed |
|
| Main Flow | |
| Lecturer | * System |
| 1. Lecturer scan lecturer card in scanner | 2. Lecturer card is validated  3. System update the database |
| Alternative flow | |
| User is not registered in system | |
| 1. Lecturer scan lecturer card  3. Lecturer cannot stop the class | 2. Lecturer card validation failed |

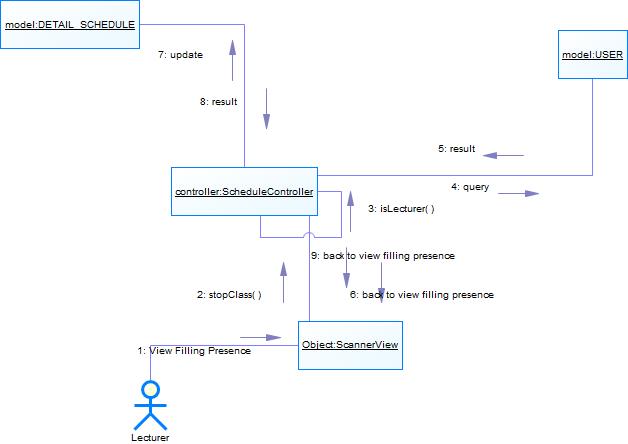
#### Activity Diagram: Stop Class



#### Sequence Diagram: Stop Class



#### Collaboration Diagram: Stop Class

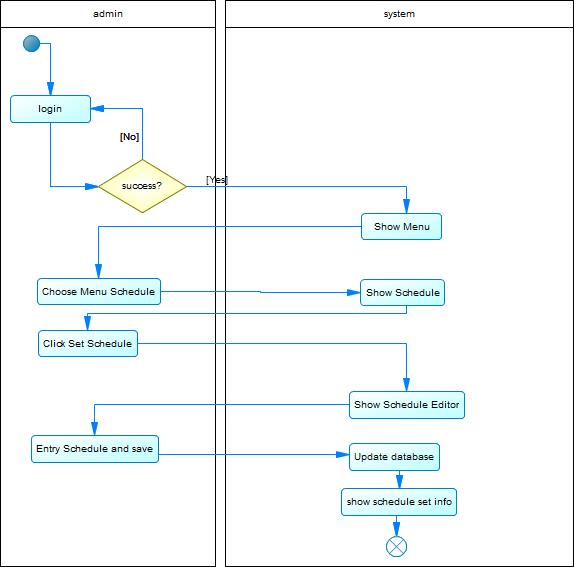


### Set default schedule

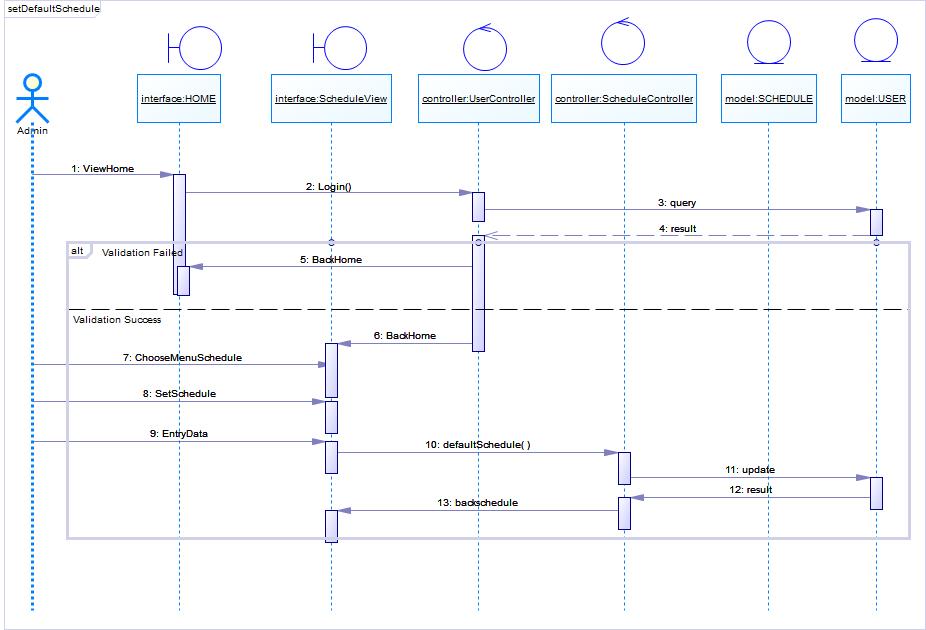
#### Scenario: Set default schedule

|  |  |
| --- | --- |
| Use Case Name | Set default schedule |
| Description | Admin can set the default schedule in a week based on the class, lecturer and room |
| Use Case ID | UC12 |
| Relation | - |
| Primary Actor | Administrator |
| Secondary Actor | - |
| Initial Condition | Admin have logged in |
| Final Condition | Admin can view the summary of default schedule |
|
| Main Flow | |
| Administrator | * System |
| 1. Admin login  3. Admin click schedule  4. Admin set schedule | 2. System validate login   1. Validation success 2. Validation failed   5. System update database  6. Return back to schedule view |
| Alternative flow | |
| Username or password is not correct | |
| Administrator | System |
| 1. Admin login | 2. System validation failed. Return to login page |

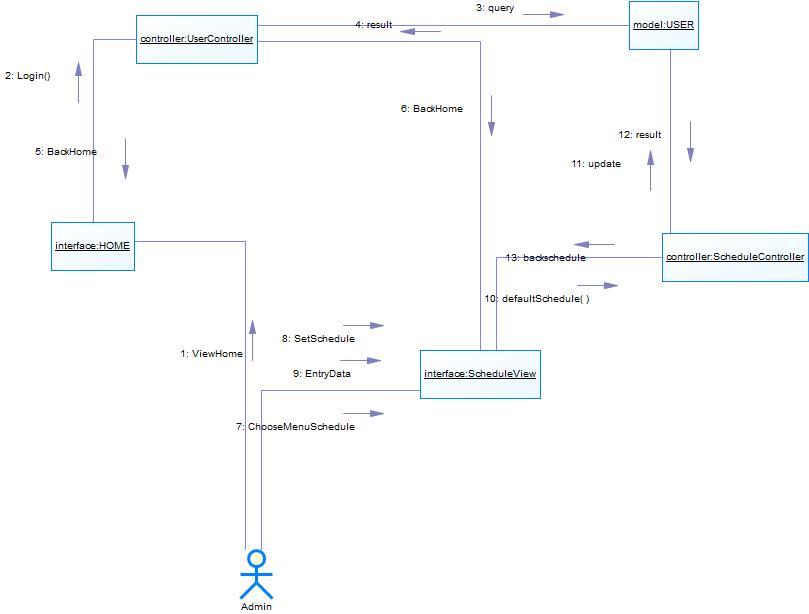
#### Activity Diagram: Set default schedule



#### Sequence Diagram: Set default schedule



#### Collaboration Diagram: Set default schedule

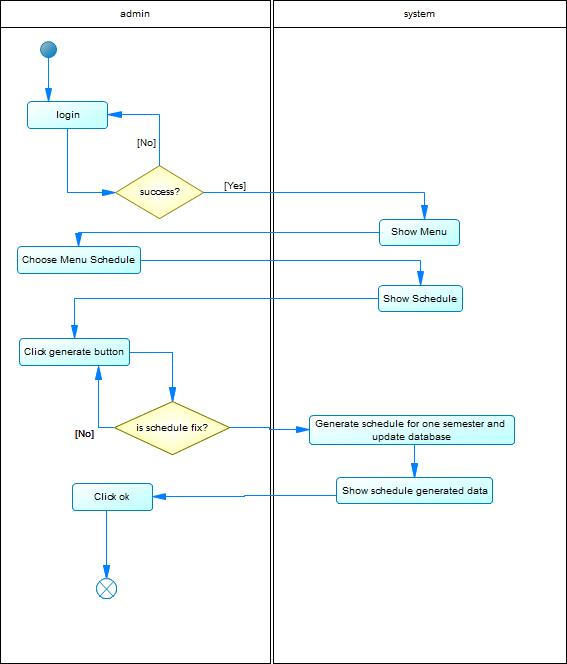


### Generate schedule for one semester

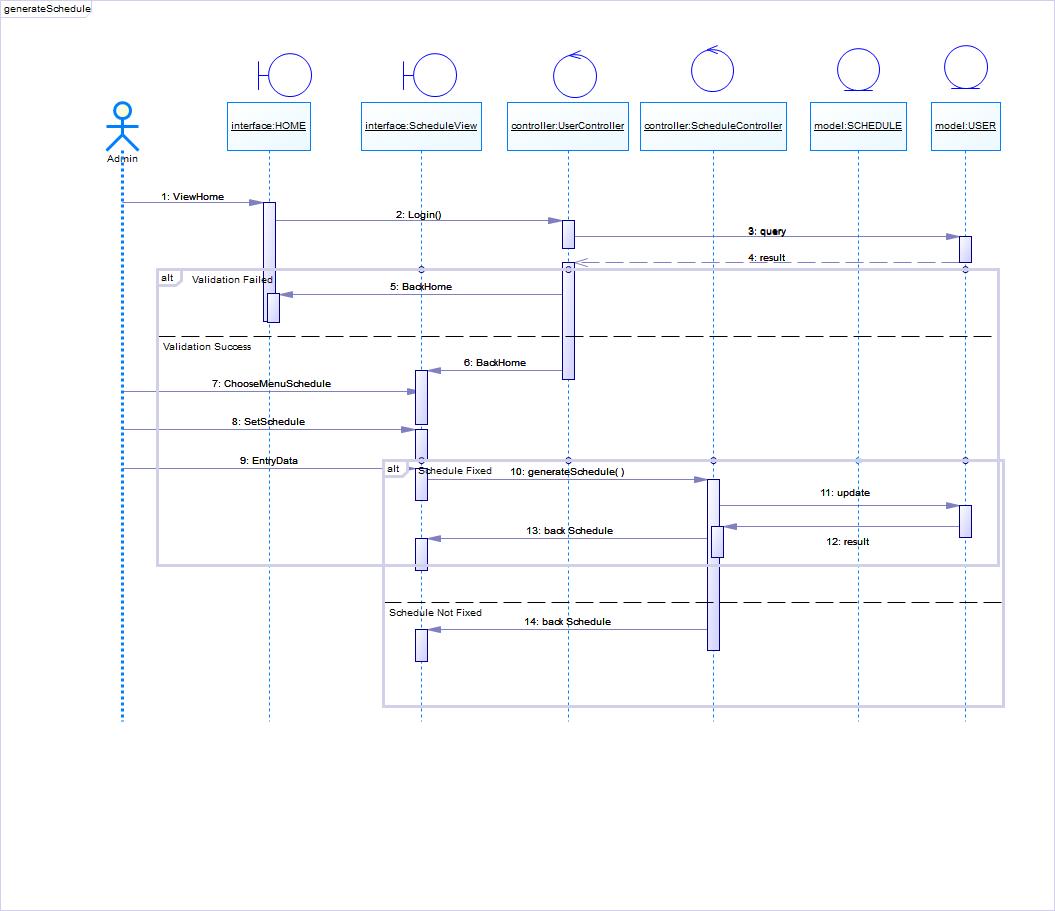
#### Scenario: Generate schedule for one semester

|  |  |
| --- | --- |
| Use Case Name | Generate schedule for one semester |
| Description | Admin can generate schedule for one semester based on the default schedule |
| Use Case ID | UC13 |
| Relation | - |
| Primary Actor | Administrator |
| Secondary Actor | - |
| Initial Condition | Admin have logged in |
| Final Condition | Admin can view the summary of default schedule |
|
| Main Flow | |
| Administrator | * System |
| 1. Admin login  3. Admin click schedule  4. Admin generate schedule | 2. System validate login   1. Validation success 2. Validation failed   5. System update database  6. Return back to schedule view |
| Alternative flow | |
| Username or password is not correct | |
| Administrator | System |
| 1. Admin login | 2. System validation failed. Return to login page |

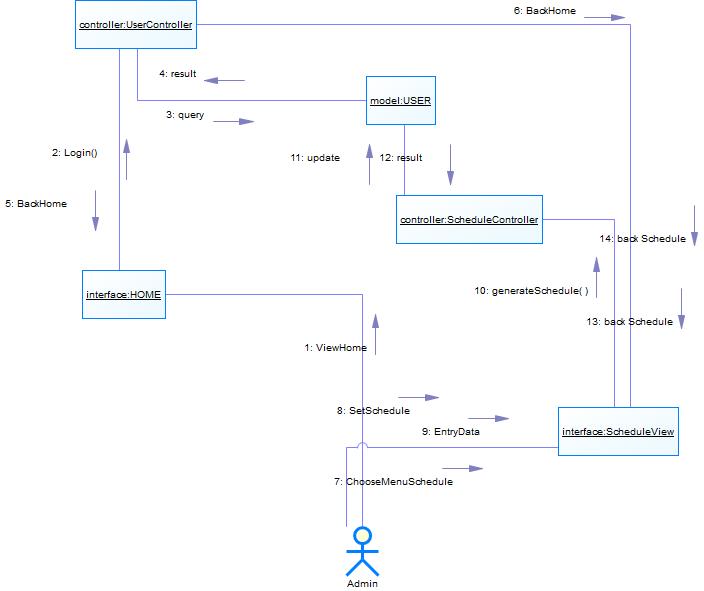
#### Activity Diagram: Generate schedule for one semester



#### Sequence Diagram: Generate schedule for one semester

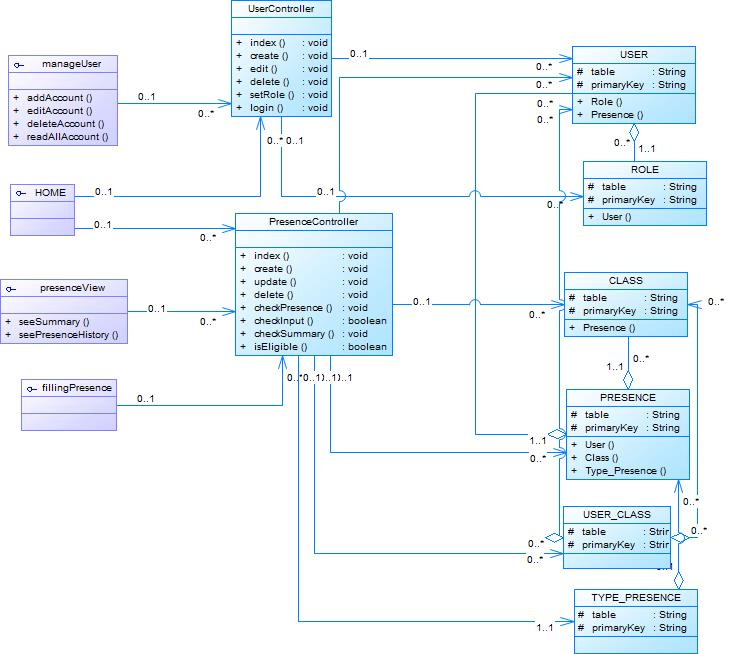


#### Collaboration Diagram: Generate schedule for one semester



## Class Description

### Class Diagram



GAMBAR ‑ Class Diagram

### Class Description

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Name | Method | Attribute | Function |
| 1. | Insert Data Control | insert() |  | Insert Data to Database |
| 2. | Update Data Control | update() |  | Update data to Database |
| 3. | Delete Data Control | delete() |  | Delete Data to Database |
| 4. | Set user role Control | setRole() |  | Set user role in database |
| 5. | Check student presence control | checkPresence() |  | Check presence in database |
| 6. | Check student presence summary control | checkSummary() |  | Check summary of student presence in database |
| 7. | Validate KTM scan | checkInput() |  | Validate student data from KTM in database |
| 8. | Check whether student can input presence | isEligible() |  | Validate the presence of student from database |
| 9. | Login validation | Login() |  | Validate login from user |

Table ‑ Class Description

### Entity Description

GAMBAR ‑ Physical Data Model

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Name | Attribute | Method | Function |
| 1. | ROLE | ID\_ROLE : integer  NAMA\_ROLE: varchar  DESC: varchar  CREATED\_AT: timestamp  UPDATED\_AT: timestamp  DELETED\_AT: timestamp | setRole() | Used for determining the role of user |
| 2. | USER | ID\_USER: integer  NAMA: varchar  ALAMAT: varchar  CREATED\_AT: timestamp  UPDATED\_AT: timestamp  DELETED\_AT: timestamp | Create()  Edit()  Delete()  Login() | Used for adding, deleteing, and editing user |
| 3. | PRESENCE | ID\_PRESENCE: integer  CREATED\_AT: timestamp  UPDATED\_AT: timestamp  DELETED\_AT: timestamp | checkPresence()  checkSummary() | Used for storing presence data |
| 4. | TYPE PRESENCE | ID\_TYPE: integer  NAME\_TYPE: varchar  DESC: varchar  CREATED\_AT: timestamp  UPDATED\_AT: timestamp  DELETED\_AT: timestamp |  | Used for determining the type of presence |
| 5. | CLASS | ID\_CLASS: integer  CLASS\_NAME: varchar  MAX: numeric  COUNT: numeric  TAHUN\_AJARAN: varchar  SEMESTER: integer  CREATED\_AT: timestamp  UPDATED\_AT: timestamp  DELETED\_AT: timestamp |  | Used for adding, editing, deleting class |
| 6 | USER CLASS | ID\_UC:integer  CREATED\_AT: timestamp  UPDATED\_AT: timestamp  DELETED\_AT: timestamp |  | Used for determining student class |

## Data Flow Diagram

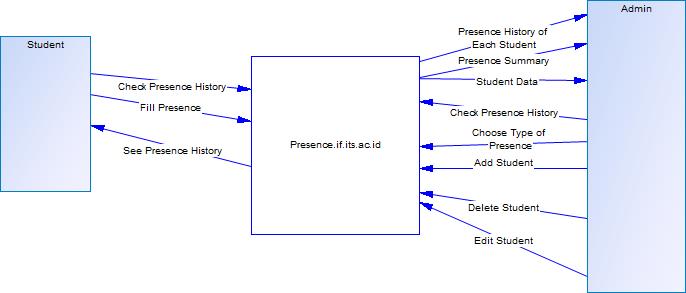


Figure ‑Level 0

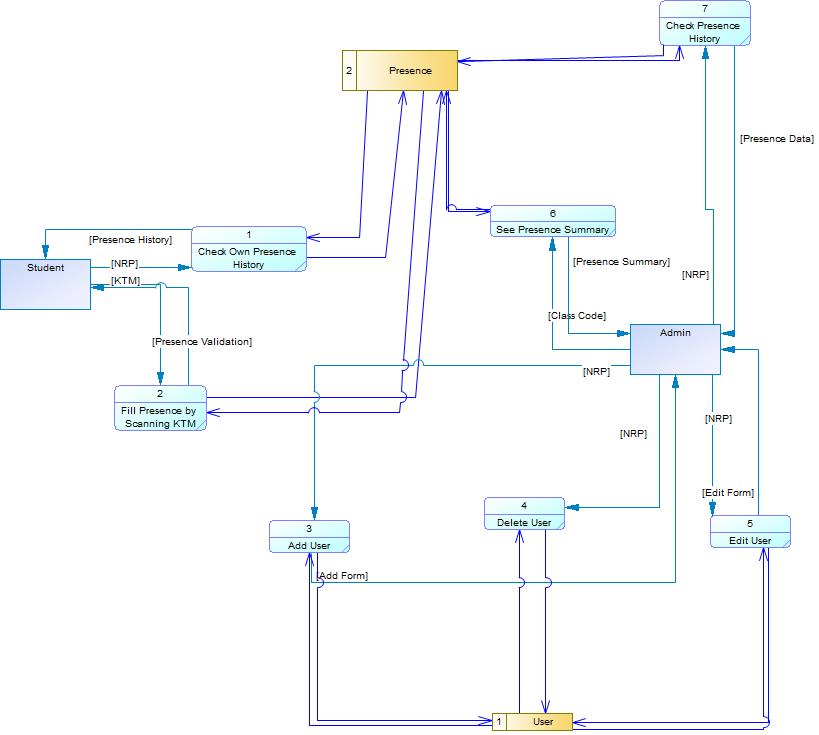


Figure ‑Level 1

**3.5. Requirement Summary**

**3.5.1 Non Functional Requirement**

|  |  |  |
| --- | --- | --- |
| **SKPL-Id** | **Parameter** | **Requirement** |
| SKPL-N01 | Availability | This application has to be available 24 hours in a day and 365 days in a year because this application will be used to store the presence data of Department of Informatics Student. |
| SKPL-N02 | Reliability | This application has to be reliable because it will store important data for Department of Informatics. There will be external factor that will determine the reliability of this application. The external factor are scanning device, internet network and electricity |
| SKPL-N03 | Ergonomy | This application has to have good ergonomy so that user can use this application with ease and not confusing. |
| SKPL-N04 | Portability | This application can be implemented in more than 5 computer and more than 10 KTM scanning device |
| SKPL-N05 | Memory | This application can store up to 100GB of data |
| SKPL-N06 | Response time | Database can be accessed in less than 2 second |
| SKPL-N07 | Safety | The network is protected by firewall |
| SKPL-N08 | Security | This application is protected by username and password. Admin and student have different access rights. |

**3.5.2 Functional Requirement Summary**

|  |  |
| --- | --- |
| **SKPL-Id** | **Keterangan** |
| SKPL-F000 | See summary of presence in each class |
| SKPL-F001 | See presence history of every student |
| SKPL-F002 | See presence data of the student |
| SKPL-F003 | Add new user |
| SKPL-F004 | Edit user |
| SKPL-F005 | Delete user |
| SKPL-F006 | Set presence type |
| SKPL-F007 | Fill presence by scanning KTM |
| SKPL-F008 | Start Class |
| SKPL-F009 | Stop Class |
| SKPL-F010 | Edit Schedule |
| SKPL-F011 | Set Default Schedule |
| SKPL-F012 | Generate schedule for one semester |
| SKPL-F013 | Choose Type of Presence |