SECJ2203: Software Engineering

System Documentation

Project Title: FABU Alumni System

Version 1.0

22nd May 2021

School of Computing, Faculty of Engineering

Prepared by: Seventh

Revision Page

a. Overview

The software design in this report is the first version of the alumni management system. This report consists of the introduction, system-specific requirement, system architectural design, detailed description of components, data design, user interface design, requirement matrix, test cases, and test approach analysis.

b. Target Audience

The target audiences of the alumni management system are the alumni in the Faculty of Built Environment and Surveying(FABU) and the FABU staff in the University of Technology Malaysia (UTM).

c. Project Team Members

Name	Matric Number	Role	Task	Status
Ng Jing Er	A19EC0115	Team Leader	Communication Interfaces Other Requirements State Diagram Use Case(Mobile - View news,View event,View charity,View report)	Completed
Goh Jo Ey	A19EC0047	Member	User Interface Performance Requirements Use Case(User - Login, Add User, Remove User, Create Account)	Completed
Chiam Wooi Chin	A19EC0034	Member	Hardware Interface Design Constraints Use Case (Alumni - Update Architecture Licensing, Mobile - View alumni profile,View alumni list, View dashboard)	Completed
Ong Yin Ren	A19EC0204	Member	Software Interfaces Software System Attributes State Diagram Use Case(Alumni - Filter friend, Add friend, Delete friend, Search friend)	Completed

Siti Fatimah Az Zahra	A19EC3032	Member	Introduction Domain diagram Use Case(Alumni-View status,Upload Status,Edit Status,Delete Status , User-Update Profile)	Completed

d. Version Control History

Version	Primary Author(s)	Description of Version	Date Completed
Version 1.0	**Ng Jing Er** **Goh Jo Ey** **Ong Yin Ren** **Chiam Wooi Chin** **Siti Fatimah Az Zahra**	**This version includes interface requirements, system features, performance requirements, design constraints, software system attributes and other requirements of the alumni system.**	**25th May 2021**

Table of Contents

1. Introduction

System Documentation (SD) comprises Software Requirement Specification (SRS), Software Design Document (SDD) and Software Testing Documentation (STD). The first one is Software Requirement Specification (SRS) documentation is documented purposely to specify the requirements for the system. It explains in detail the purpose and features of the system, the design of interfaces for this system, definitions for the functional and non-functional specifications of the system, behaviour of the system, design constraints or the limitations of the environment to meet each requirement. This information is collected by interviewing one of the stakeholders of the system and this technique is the most common technique for gathering requirements.

Next, the Software Design Document (SDD) is to outline the description of the requirements in the design diagrams of the system to assist the development of the system. This SDD is focusing on the graphical documentation in detail including use case models, sequence diagrams, object behaviour models and other supporting requirement information.

One of the important facets of software development life cycle is testing. Therefore, Software Testing Documentation (STD) is the third document in this SD that involves the documentation of items that should be developed before or during the testing of a system. Documentation for software testing helps in estimating the testing effort required, test coverage, requirement tracking, and more. In general, it helps to reduce cost and time in system development. In this testing documentation, it shows the result of the software testing of the system, hence, we can figure out the weakness and improve it.

1.1 Purpose

This SD describes the system for software development to progress seamlessly with a great comprehension of what is to be built and how it is expected to be built. The purposes of this documentation are as listed below:

1. To provide the necessary information for the intended audience of the documentation
2. To identify the requirements for the system
3. To outline the description of the requirements in the graphical documentation
4. To provide evidence of progress in the development process and to monitor the process.
5. To ease system implementation and modification
6. To narrow down the communication gaps among developers, designers, stakeholders, users, and the management.
7. To provide the result of the software testing of the system
8. To provide a means to determine in advance what will occur and when.

The audiences of this SD include system developers, system testers, system designer, database admin, project managers, system users, FABU alumni and the stakeholders. This system is believed to increase productivity, efficient and overall ease of use.

1.2 Scope

The software product is FABU Alumni System. It is a system that is made for the community in particular the alumni and the staff of Faculty of Built Environment & Surveying, UTM also known as FABU. This system provides a computerized platform to replace the manual system. It can be accessed by both PC and mobile phone. The alumni can use this system to view the dashboard and reports while keeping up to date for the latest news, charity and events that are happening in FABU. This system also allows the alumni to manage their friends and status. They also can update architecture licensing in this system. Meanwhile, the admin is responsible for creating accounts and removing the users. The staff use this system mainly to manage and update the news, charity, and events in the system. All users can update their profile picture and login details. In general, this system eases the information transfer from the staff to the alumni and creates an efficient platform for the users.

The advantages of FABU Alumni System include to provide a systematic and computerized platform for the FABU community to organize and publicize the events which focus more on themselves. Moreover, alumni of FABU are able to give the donation to FABU directly. He also hopes that FABU alumni are able to search for their old friends through this system since every FABU alumni will create an account for it. They can search other alumni through some keywords for example name and batch. Besides, alumni are also benefited from this system because if they keep close relationships with FABU association, they could have a huge network with a lot of professional contacts not only lecturers but also other alumni. The staff also profited from the system because this system significantly reduces the workload of the staff and creates an efficient platform to manage the information of the news, charity, and events of the faculty. Adding to that the admin can add and delete accounts so this can avoid data redundancy.

1.3 Definitions, Acronyms and Abbreviation

No.	Terms/ Acronyms/ Abbreviation	Definitions
1.	FABU	Fakulti Alam Bina dan Ukur (Faculty of Built Environment and Surveying.)

2.	UTM	Universiti Teknologi Malaysia
3.	SD	System Documentation
4.	SRS	Software Requirement Specification
5.	SDD	Software Design Document
6.	STD	Software Testing Documentation
7.	UML	Unified Modelling Language a standardized modelling language consisting of an integrated set of diagrams
8.	Domain model	Domain model is a conceptual model to describe and model real world entities and the relationships between them. Domain class objects reflect business concern, policies, and constraints
9.	Use case diagram	Use case diagrams are used for interaction modelling. The purpose of a use case diagram is to capture the dynamic aspect of a system.

1.4 References

1. Sommerville, I. 2016. “Software Engineering”, 10th Edition, Pearson.

2. Imran, M., Alghamdi, A. A., & Ahmad, B. (2016). Software Engineering: Architecture, Design and Frameworks.

3. Charland, A., & LeRoux, B. (2011). Mobile Application Development: Web vs. Native. Queue, 9(4) 20. Doi: 10.1145/19989.1968203

1.5 Overview

The SD Document is divided into nine sections with some of them have various subsections. The sections of the Software Design Document are:

1.0 Introduction: This section consists of five subsections that covers the system purpose, system scope, system overview, references, document overview and the definitions of terms, acronyms and abbreviation used in this documentation. Overall, it explains about the contents of System Documentation (SD0 and the system, its background, its functionalities, the audiences, and the references used to complete this system.

2.0 Specific Requirements: This section has six subsections that includes external interface requirements, system features, performance requirements, design constraints, software system attributes and other requirements. Generally, in the first subsection it discusses user interfaces, hardware interfaces, software interfaces and communication interfaces of the system Meanwhile, it describes the system features using use case description for each function that the system offers. In addition, this section briefly explains about the design constraints, performance requirements and software system attributes.

3.0 System Architectural Design: In this section, it describes the architectural design. The system should be divided into a set of communicating subsystems and components. It describes the decomposition, the components, the interfaces, and the rationales.

4.0 Detailed Description of Components: Here we describe the detailed description of components consisting of a complete package diagram to show the files and subfiles contained within the system. Also, the details then elaborated according to the content of the package diagram where every file has their class diagram, sequence diagram, entity, method, and the algorithm table.

5.0 Data Design: Data design contains data description that contains every entity name and their description. Next is a data dictionary which lists out the attribute name. type and description for each entity.

6.0 User Interface Design: This section explains about the user interface design. Every screen for every transaction in the system is attached. The screens show a step by step process when the user starts to open the application, register, login, main page and upload. Moreover, it also contains the screen for all the actors including which are only accessible for specific actors.

2. Specific Requirements

2.1 External Interface Requirements

2.1.1 User Interfaces

1. **Login Interface**

When the user clicks the "Login" button, the user is required to enter the username and password they set it before. In order to access the platform, the username and password entered by the user should be matched with the information stored in the database of the user. If the user doesn't have an account, the user should choose to register instead of log in to the system.

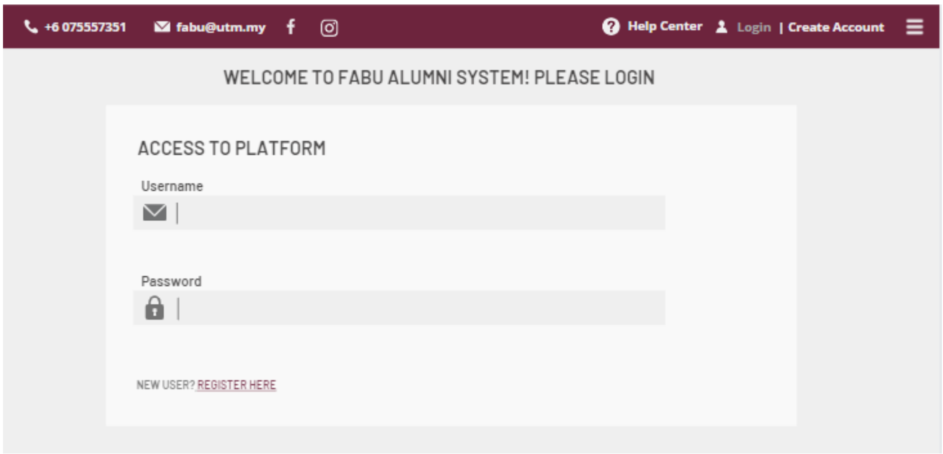


Figure 2.1.1.1: Login Interface

2. **Create account**

Each alumni or staff is allowed to create an account in the system one time only. If the user clicks the "Create Account" button, the user will be redirected to the create account interface by the system. The user should choose their role as staff or alumni. In order for the user to register an account successfully, the id entered by the user should exist in the database of the system. If the id entered by the user is not found, an error message will be displayed to the user, and the user cannot further access the next page. If the id entered by the user is found, the alumni or staff should fill out the registration form that includes their full name, phone number, role, id, password, address, employment status, education status, license status for alumni. The information entered by the user should be in the correct format that is set in the system, otherwise, an error message will be prompted to ask the user to enter the information in the correct format.

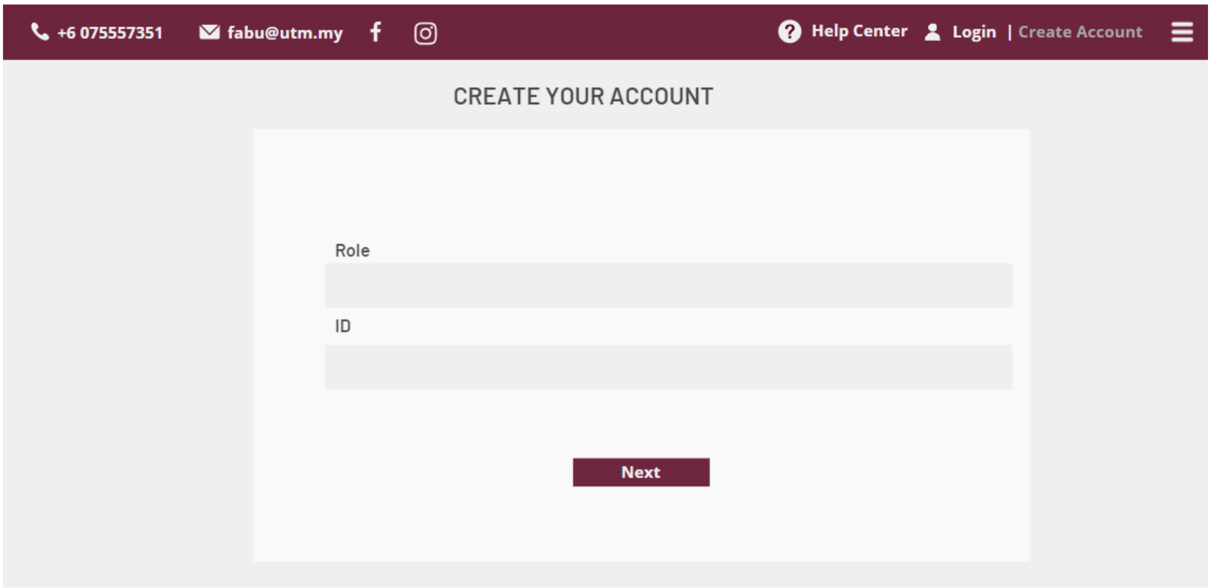


Figure 2.1.1.1.2: Create Account Interface

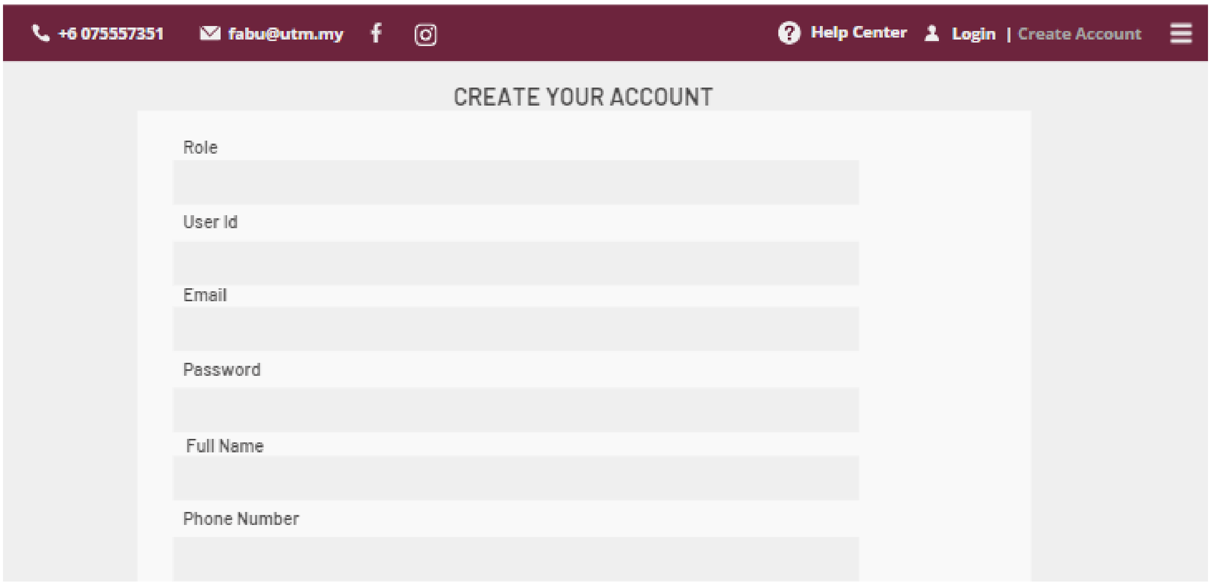
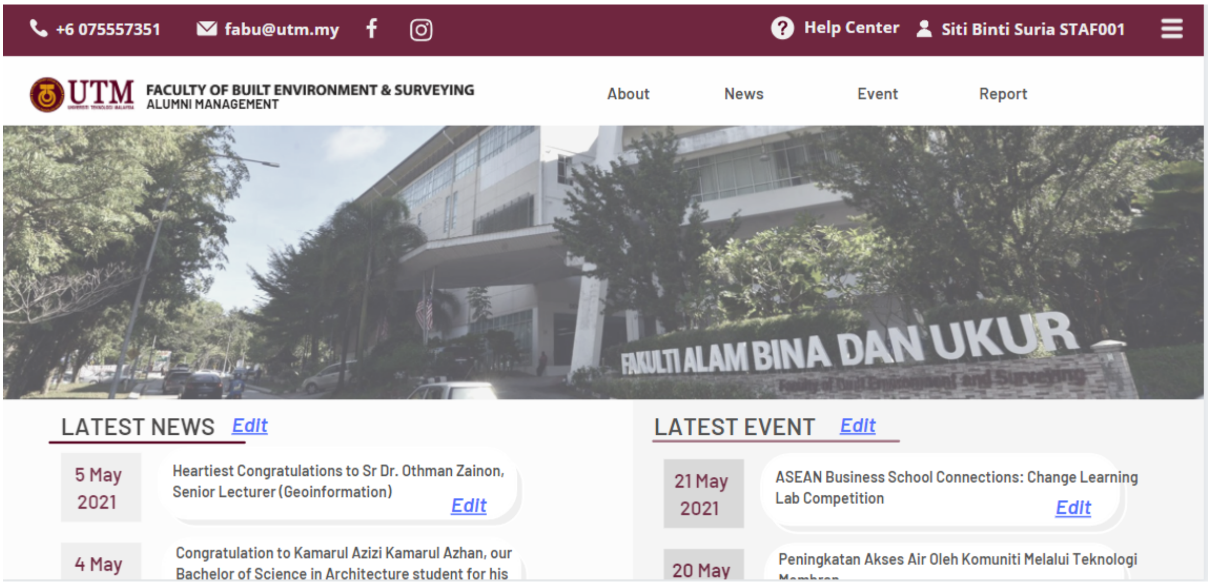


Figure 2.1.1.1.3: Create Account Interface

3. Main Page Interface

The alumni can view the news, events, and reports of the Faculty of Built Environment and Surveying(FABU). The main page of the alumni display the latest news, and the latest event held by FABU. There is a navigation bar that allows the user to view the news, events and the report that are updated by the admin of the FABU. The staff can choose to update and edit either one of the news, events and reports one time by clicking the edit or add button.



5 May 2021 — Heartiest Congratulations to Sr Dr. Othman Zainon, Senior Lecturer (Geoinformation) Edit

4 May — Congratulation to Kamarul Azizi Kamarul Azhan, our Bachelor of Science in Architecture student for his

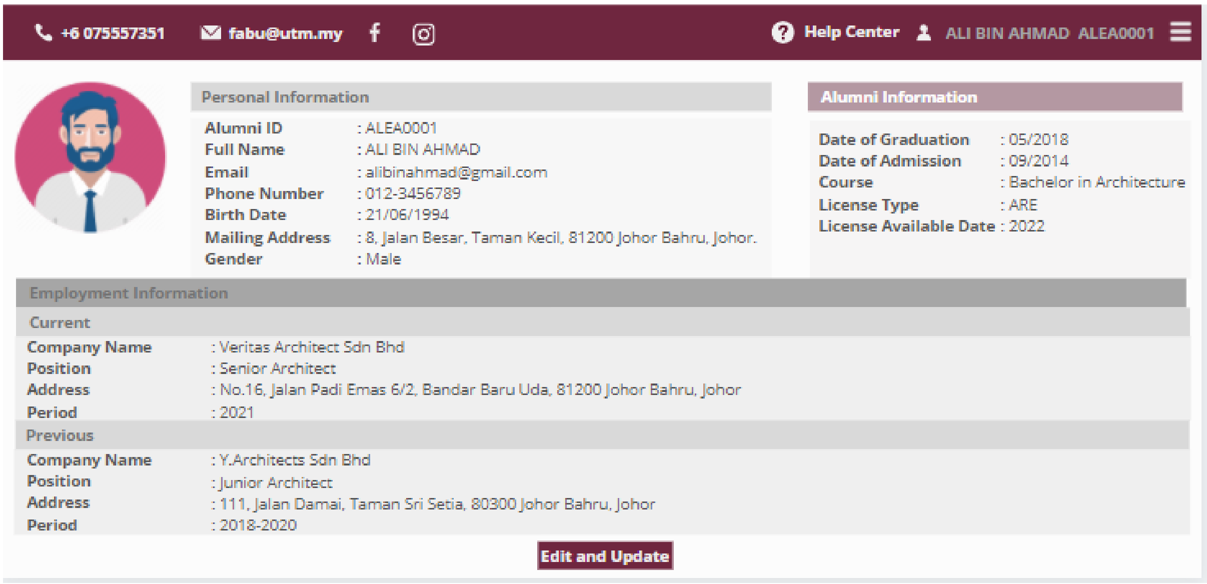
21 May 2021 — ASEAN Business School Connections: Change Learning Lab Competition Edit

20 May — Peningkatan Akses Air Oleh Komuniti Melalui Teknologi

Figure 2.1.1.1.4: Main Page

4. User profile

The alumni’s user profile displays the personal information, alumni information, and employment status of the existing user. The alumni id, full name, gender is not allowed to change by the alumni. Other than that, the alumni can choose to edit and update their information anytime. If there is any error in the alumni id, full name, gender, the alumni should contact the admin of the system to review the issue. The staff’s personal information consists of the personal information and the staff information. The staff can choose to edit and update their information anytime. Same as the alumni, staff id, full name, gender is not allowed to change by the staff.



Personal Information

Alumni ID	: ALEA0001
Full Name	: ALI BIN AHMAD
Email	: alibinahmad@gmail.com
Phone Number	: 012-3456789
Birth Date	: 21/06/1994
Mailing Address	: 8, Jalan Besar, Taman Kecil, 81200 Johor Bahru, Johor.
Gender	: Male

Alumni Information

Date of Graduation	: 05/2018
Date of Admission	: 09/2014
Course	: Bachelor in Architecture
License Type	: ARE
License Available Date	: 2022

Employment Information

Current

Company Name	: Veritas Architect Sdn Bhd
Position	: Senior Architect
Address	: No.16, Jalan Padi Emas 6/2, Bandar Baru Uda, 81200 Johor Bahru, Johor
Period	: 2021

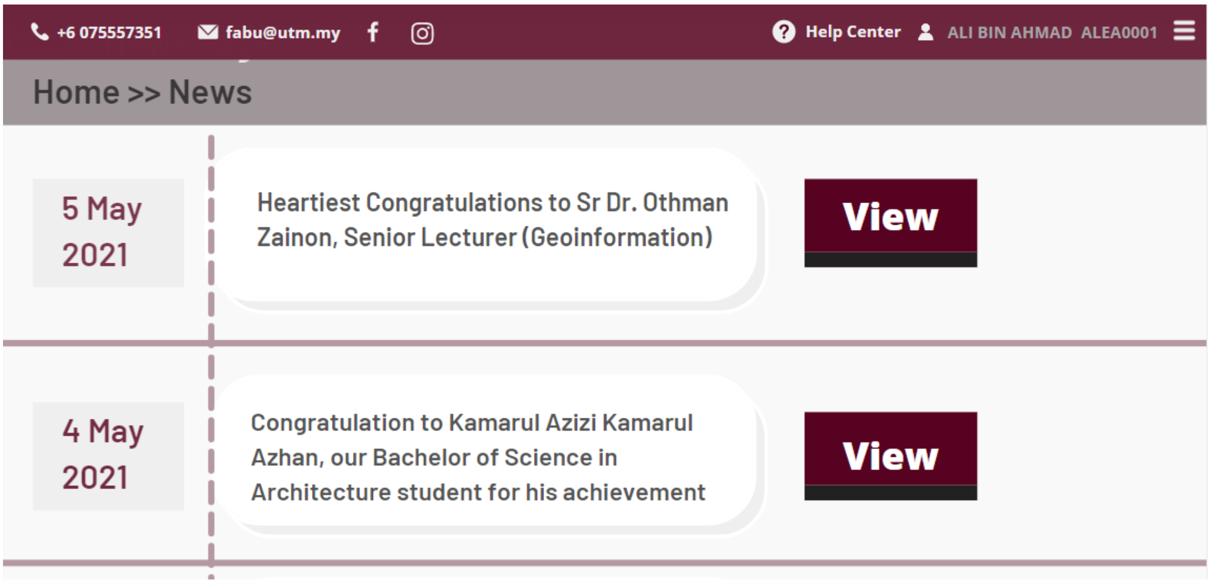
Previous

Company Name	: Y.Architects Sdn Bhd
Position	: Junior Architect
Address	: 111, Jalan Damai, Taman Sri Setia, 80300 Johor Bahru, Johor
Period	: 2018-2020

Figure 2.1.1.1.5: User Profile

5. News Page

The alumni can choose to view the news available on the news page. The news is sorted by the dates, from the latest to the oldest. The staff can choose to edit one news at one time. The staff also can choose to add news to the system through the “Add News” button at the right of the interface and the task should be done in one hour,



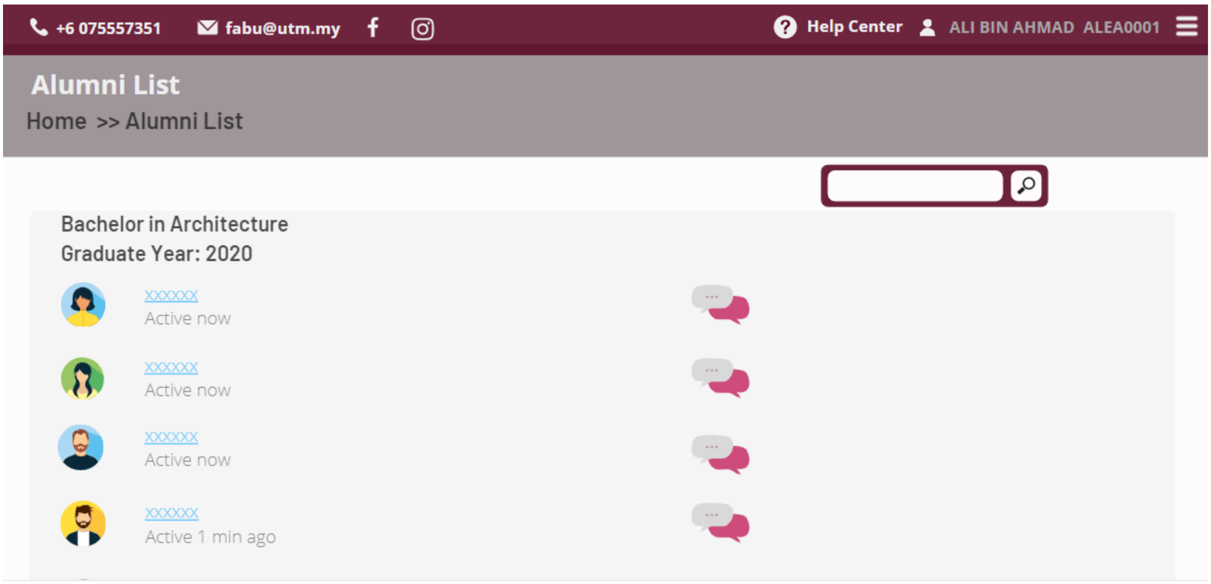
5 May 2021 — Heartiest Congratulations to Sr Dr. Othman Zainon, Senior Lecturer (Geoinformation)

4 May 2021 — Congratulation to Kamarul Azizi Kamarul Azhan, our Bachelor of Science in Architecture student for his achievement

Figure 2.1.1.1.6: View News Interface

6. Alumni List

A list of alumni sorted by the course and the graduate year is shown in the alumni list interface. The alumni can search for their friends through the search bar at the right of the interface. The search bar allows the user to search one person's name one time. Through the alumni list, the alumni can click to view other profiles by clicking the hyperlink. The alumni also can choose to chat by clicking the message icon at the right of the alumni's profile. The admin of the system can manage the alumni by clicking the "Manage" button. The admin can either remove or edit one alumni in the system at one time and the task should be done in 1 hour.



Bachelor in Architecture
Graduate Year: 2020

xxxxxx
Active now

xxxxxx
Active now

xxxxxx
Active now

xxxxxx
Active 1 min ago

Figure 2.1.1.1.7: Alumni List Interface

7. Dashboard

The dashboard consists of the status of the alumni, their friend's status, and the friend list. The alumni can choose to share their status on the dashboard to allow their friend to view his/her updated status. The active activity of the friends is shown below their name in the friend list.

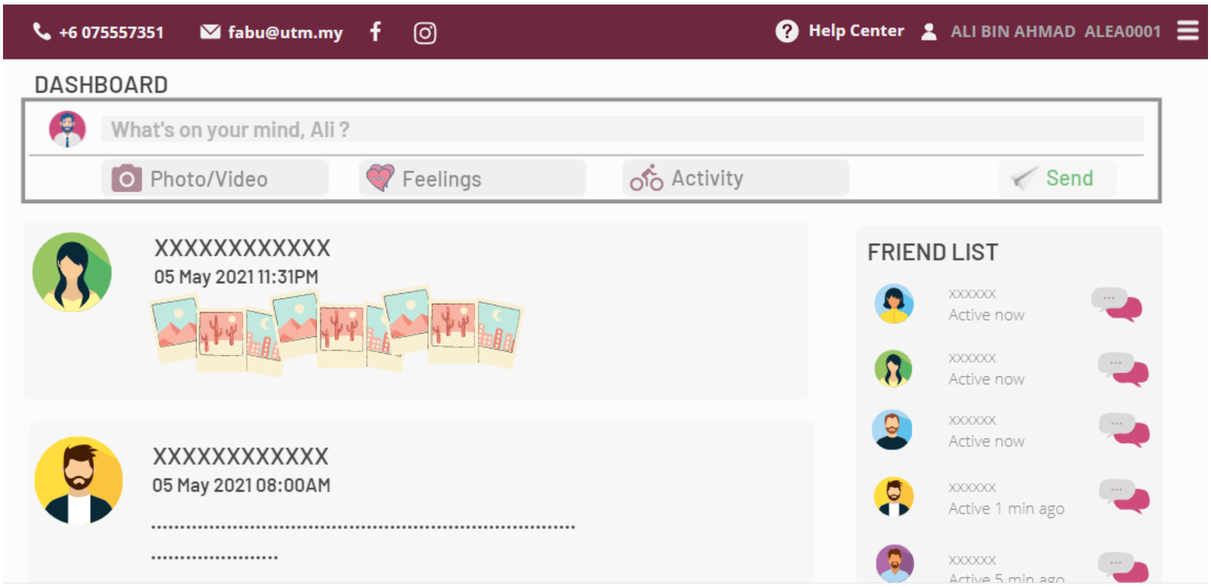


Figure 2.1.1.1.8: Dashboard Interface

8. Event Information Page

When the alumni clicks the specific event on the event information page, the user can view the detailed information of the event. The alumni can choose either to join the event or sponsor the event. The staff can change the event details by clicking the "Edit" button and it should be done in 1 hour.

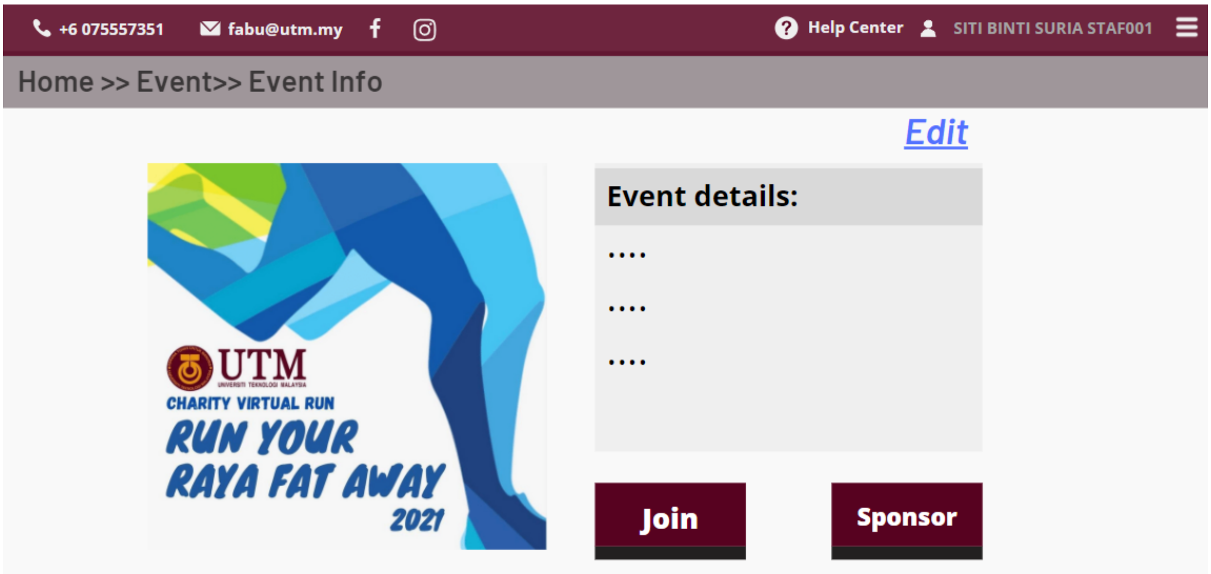
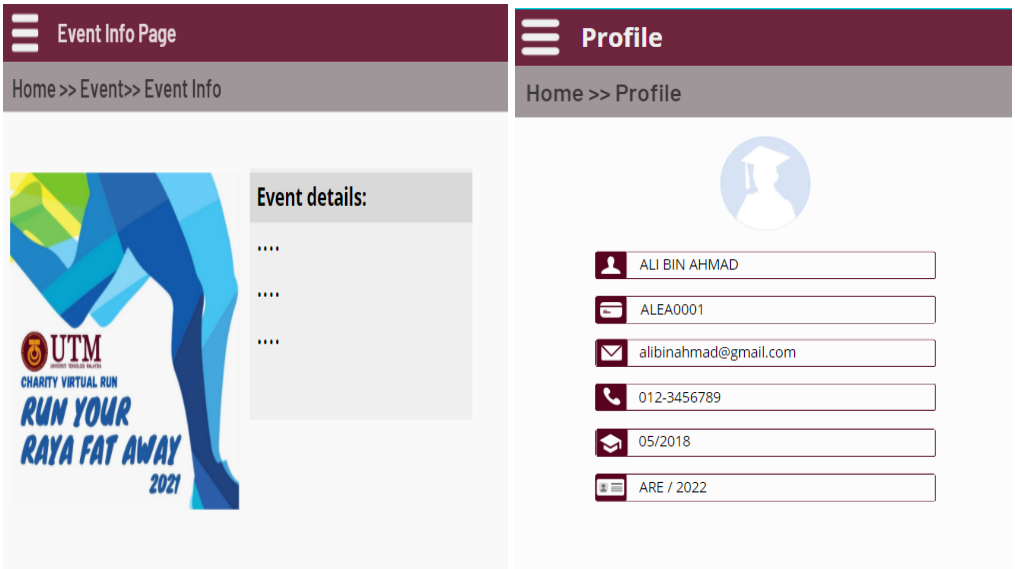


Figure 2.1.1.1.9: Event Information Interface

9. Mobile Interface

For the mobile interface, user interaction is restricted. Only the user with the alumni role is allowed to access the mobile interface. The alumni only can view the alumni detail, profile detail, news, charity, event, sponsor, and the alumni report.



ALI BIN AHMAD

ALEA0001

alibinahmad@gmail.com

012-3456789

05/2018

ARE / 2022

Figure 2.1.1.1.10: Mobile Interface

2.1.2 Hardware Interfaces

In this system, 2 servers are used which the web application will be hosted on one of the department's Linux servers and connected to one of the school Oracle Database servers. The web server is listening on the web standard port, port 80. The system is a web-based application; clients are required to use a modern web browser such as Mozilla Firefox 1.5, Internet Explorer 6 and Enable Cookies. The computer must have an Internet connection in order to be able to access the system. The Online Alumni System is also supported on mobile devices such as cell phones. All components are able to be executed on personal computers with Windows OS platforms and other platforms like Linux, Unix with processor Intel Pentium 4 Dual Core, RAM 512 MB or above, and hard disk: 40 GB or above.

2.1.3 Software Interfaces

1. **Google Chrome (Web Browser)**

   Name: Chrome

   Mnemonic:

   Specification number: 90.0.0.0

   Version number : 90.0.4430.93 (Windows, MacOS),

   : 90.0.4430.91 (Android),

   : 87.0.4280.163 (iOS)

   Source: https://www.google.com/chrome/

   Discussion: Chrome is used to open the UTM Alumni website

2. **C++ (Programming Language)**

   Name: Dev C++

   Mnemonic:

   Specification number: 5.11

   Version number: 5.11

   Source: https://www.cplusplus.com/

   Discussion: It is an object-oriented language that can be used for both high-level and low-level functions. It has an intermediate difficulty level and can be used for everything from displaying video game graphics to extracting data from large data sets. This universal programming language works on all types of operating systems and can be used to develop any type of software.

3. **Microsoft SQL Server (Database)**

   Name: SQL Server 2019

Mnemonic:

Specification number: 15.0.0.0

Version number: 15.0.2000.5

Source:https://www.microsoft.com/en-us/sql-server/sql-server-2019?rtc=1

Discussion: SQL server database is used to store and record all the information in the alumni management system

2.1.4 Communication Interfaces

There are some communication interfaces involved in the system, which include:

1. Email protocol

   Email is the main communication device for the Alumnis to communicate with the staff. Both the Alumni and staff are able to send and receive messages via Email.

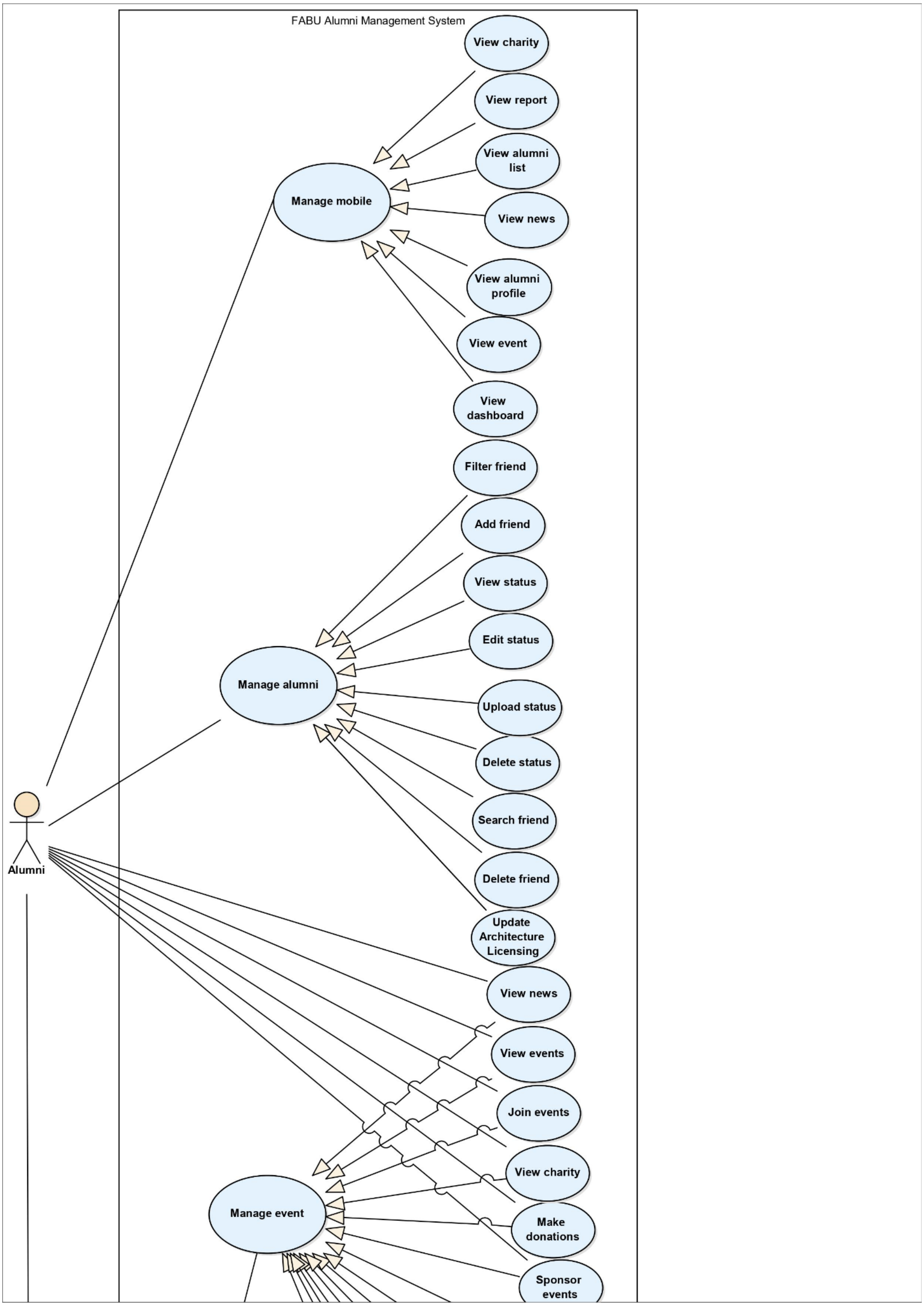
2. Transmission Control Protocol (TCP) and Internet Protocol (IP)

   TCP is used for data transmission. TCP works with and complements IP. It is used for communicating over the network. It divides any message into a series of packets that are sent from source to destination and there it gets reassembled at the destination.

3. HyperText Transfer Protocol Secure (HTTPS)

   HTTP is used for transferring data between the client browser (request) and the web server (response) in the hypertext format, same in case of HTTPS except that the transferring of data is done in an encrypted format which is secure from the hackers or modification of data throughout the transfer of packets.

2.2 System Features



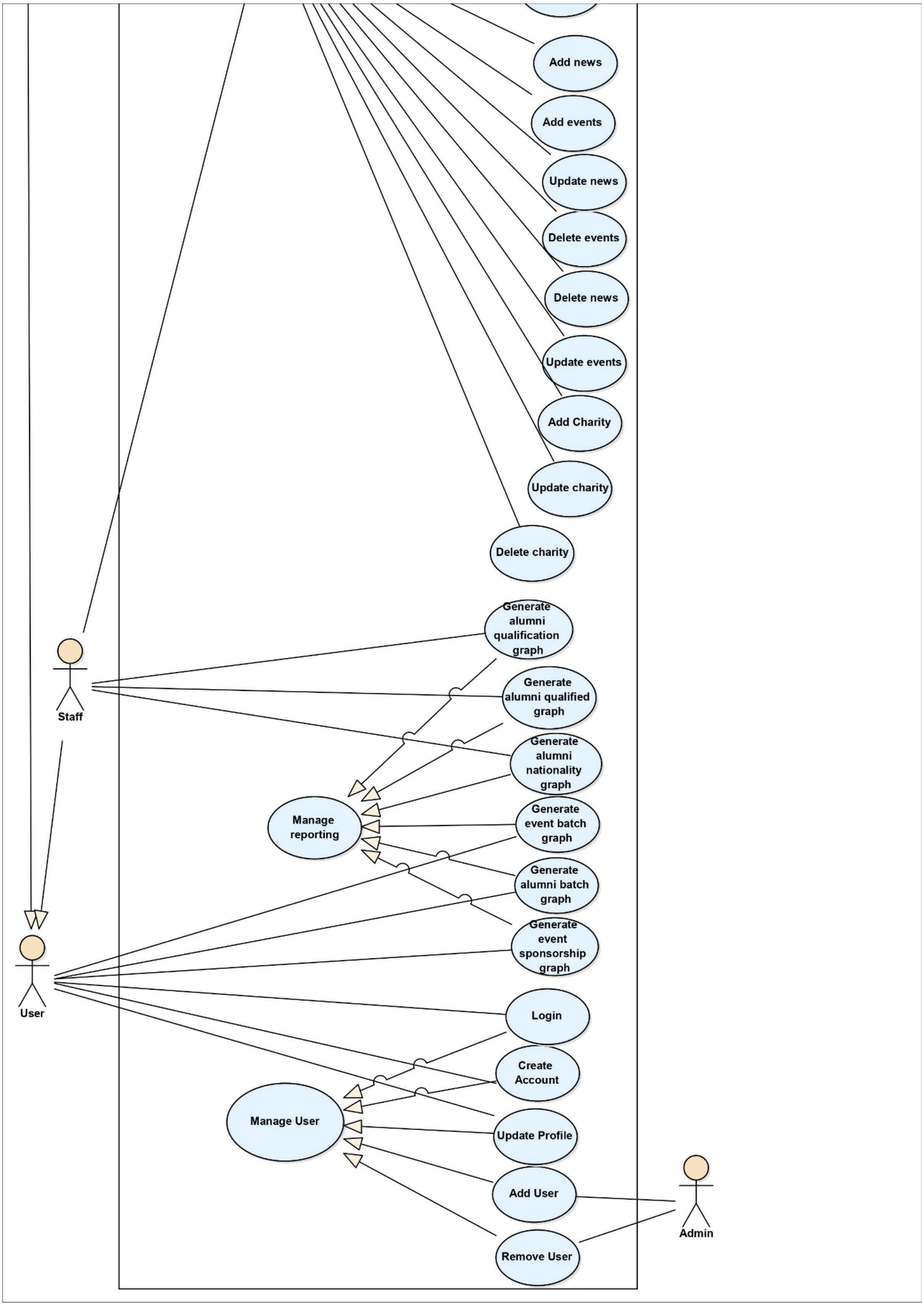


Figure 2.2.1: Use Case Diagram for FABU ALUMNI SYSTEM

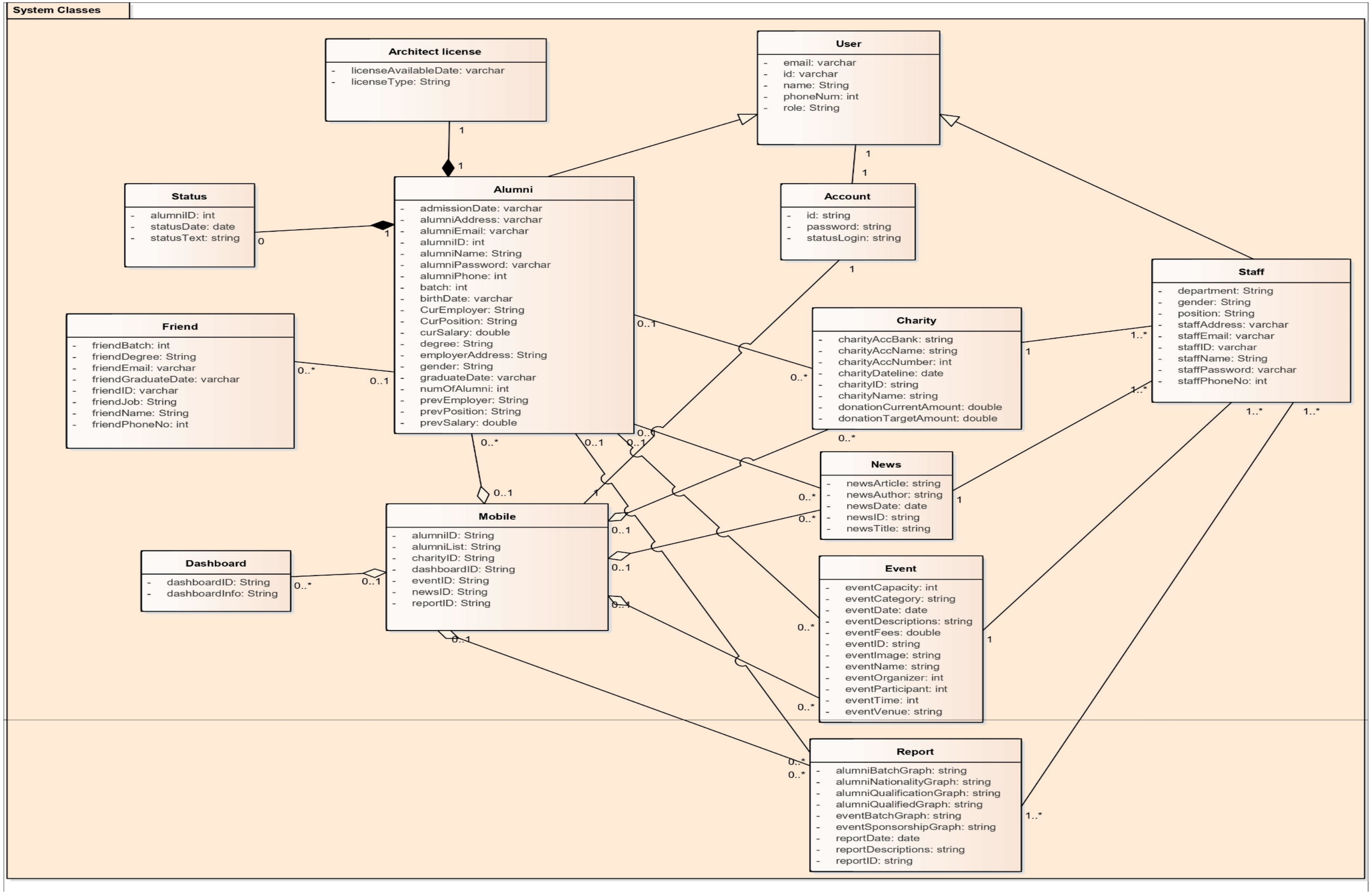


Figure 2.2.2: Domain Model for FABU ALUMNI SYSTEM

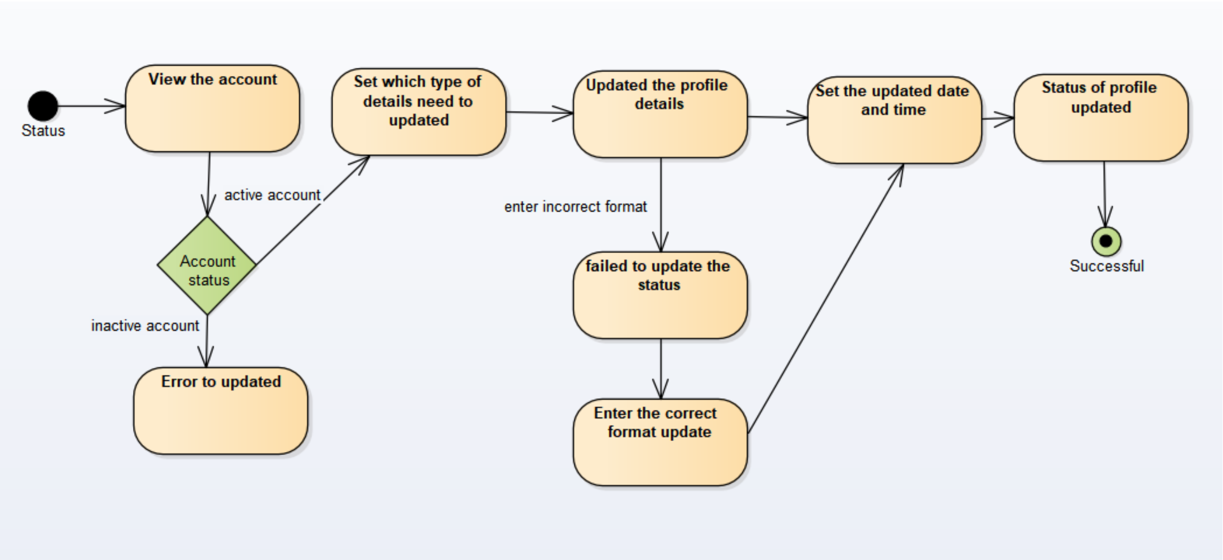

Status
View the account
Set which type of details need to updated
Updated the profile details
Set the updated date and time
Status of profile updated
active account
Account status
inactive account
Error to updated
enter incorrect format
failed to update the status
Enter the correct format update
Successful

Figure 2.2.3: State Machine Diagram for class Status

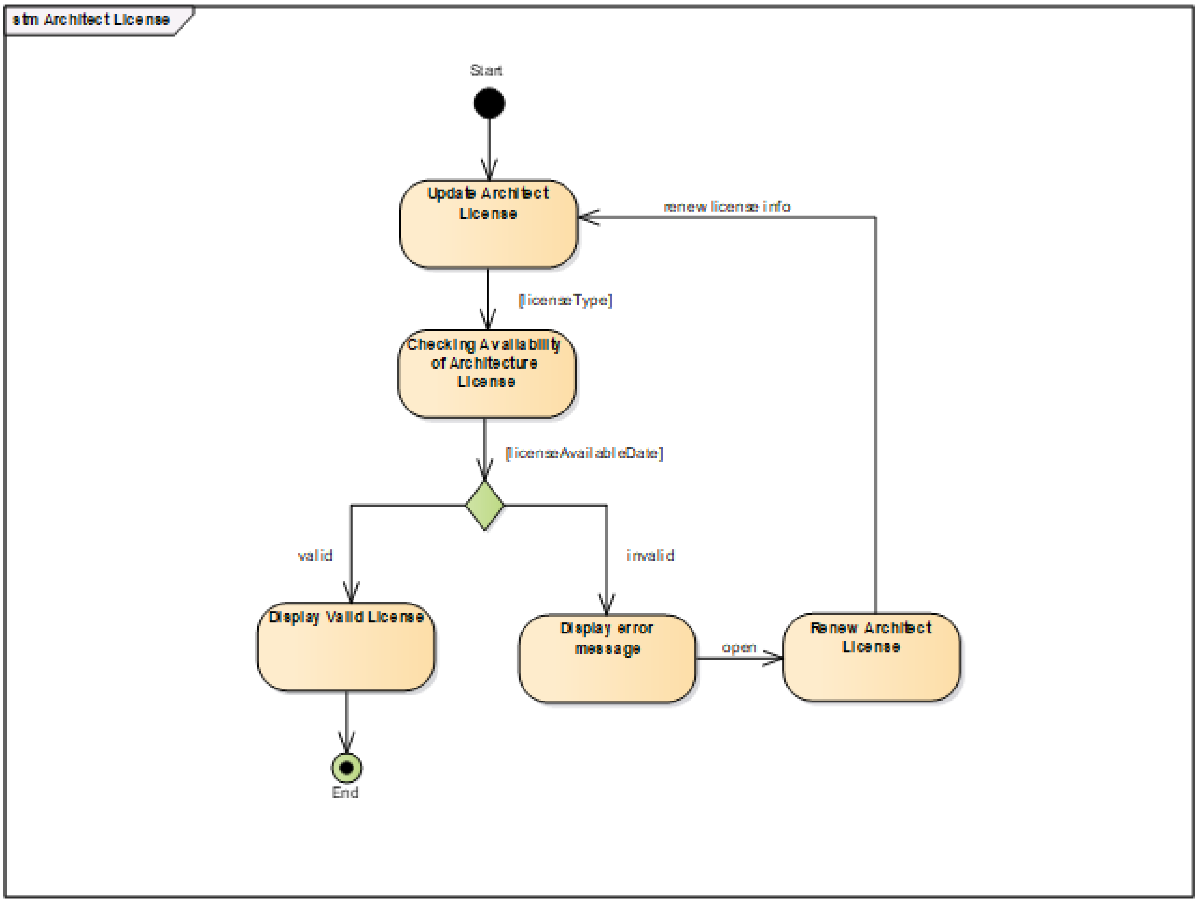


Figure 2.2.4: State Machine Diagram for class Architect License

2.2.1 UC0001: Use Case <Add User>

SCENARIOS

Basic Path. Basic Path

1. The user selects "add user" in the system
2. The system display add user page
3. The user selects the category of the user to be added
4. The system displays the add user page for specific role
5. The user enter the user id that need to be added
6. The system check the user id entered
Exception: 6a. User Id Existed
7. The user enters the information of the new user
8. The system validates the information
Exception: 8a. Wrong format
9. The system adds the information to the database of the system
10. The system prompt a successful message to the user

Exception. Wrong format

1. The system displays the error message to the user
2. The system ask the user to correct the error information

Exception. User Id Existed

1. The system prompt a message to inform the user id entered is exist in the system
2. The system ask the user to correct the user id

CONSTRAINTS

Post-condition. The user successfully added a new user with specific role in the system

[Approved, Weight is 0.]

Pre-condition. The user has not added into the system before.

[Approved, Weight is 1.]

OUTGOING STRUCTURAL RELATIONSHIPS
Generalization from Add User to Manage User [Direction is 'Source -> Destination'.]

CONNECTORS	
UseCaseLink	Source -> Destination
From:	Admin : Actor, Public
To:	Add User : UseCase, Public

Table 2.2.1.1: Use Case Description for <Add User>

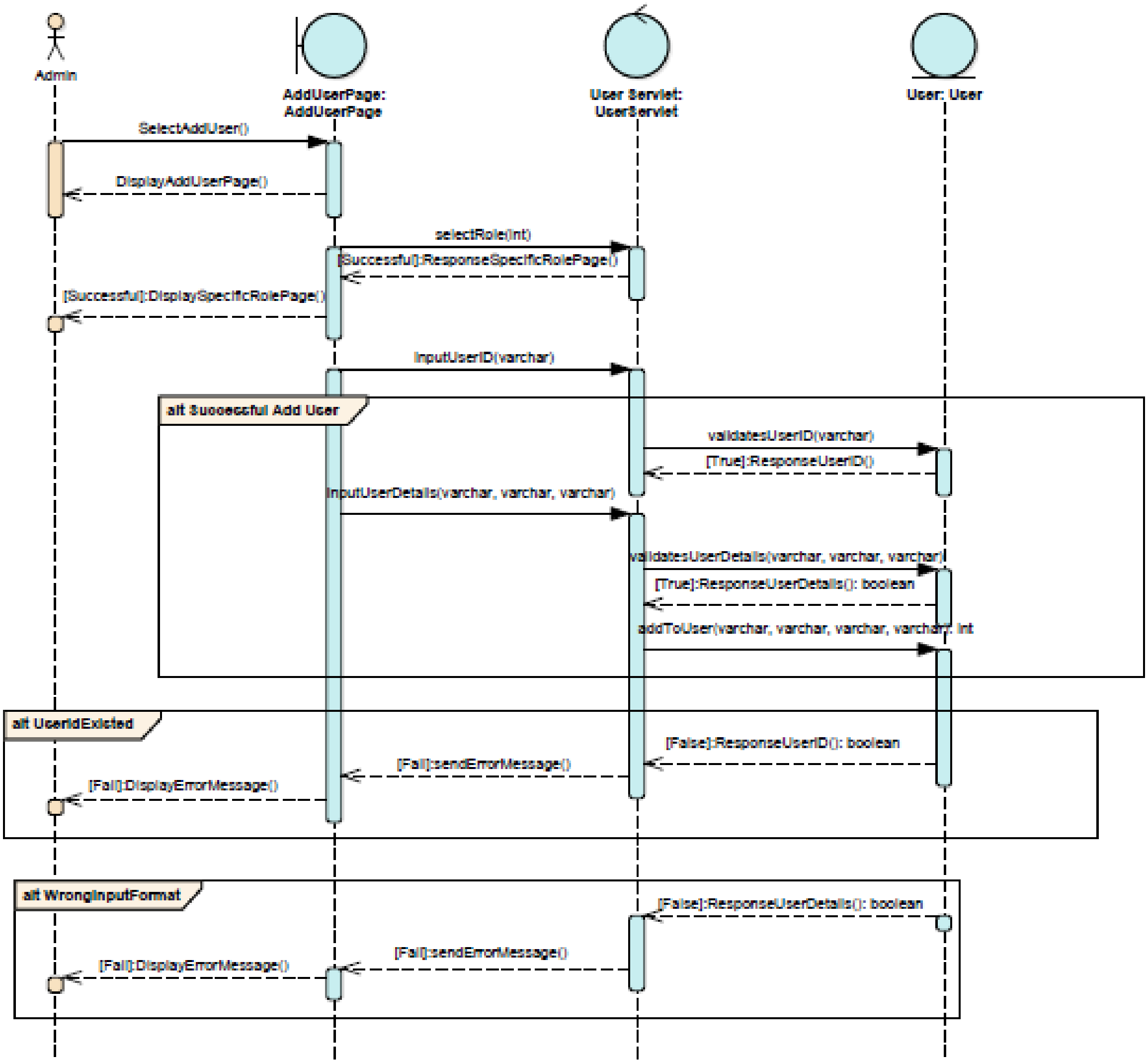

Admin
AddUserPage: AddUserPage
User Servlet: UserServlet
User: User
SelectAddUser()
DisplayAddUserPage()
selectRole(int)
[Successful]:ResponseSpecificRolePage()
[Successful]:DisplaySpecificRolePage()
InputUserID(varchar)
alt Successful Add User
validatesUserID(varchar)
[True]:ResponseUserID()
InputUserDetails(varchar, varchar, varchar)
validatesUserDetails(varchar, varchar, varchar)
[True]:ResponseUserDetails(): boolean
addToUser(varchar, varchar, varchar, varchar): int
alt UserIdExisted
[False]:ResponseUserID(): boolean
[Fail]:sendErrorMessage()
[Fail]:DisplayErrorMessage()
alt WrongInputFormat
[False]:ResponseUserDetails(): boolean
[Fail]:sendErrorMessage()
[Fail]:DisplayErrorMessage()

Figure 2.2.1.2: Sequence Diagram for <Add User>

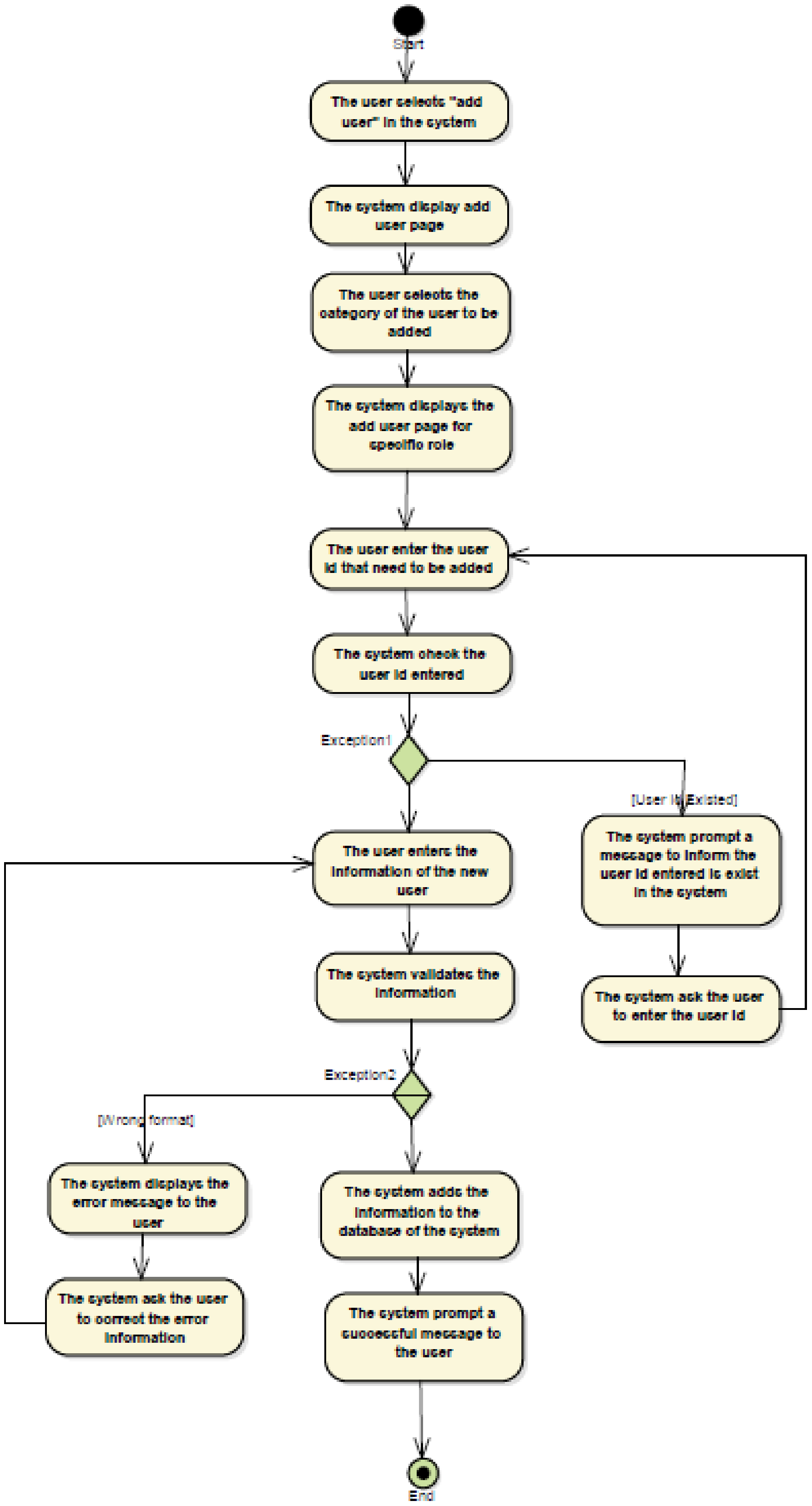

Start
The user selects "add user" in the system
The system display add user page
The user selects the category of the user to be added
The system displays the add user page for specific role
The user enter the user id that need to be added
The system check the user id entered
Exception1
[User Id Existed]
The system prompt a message to inform the user id entered is exist in the system
The system ask the user to enter the user id
The user enters the information of the new user
The system validates the information
Exception2
[Wrong format]
The system displays the error message to the user
The system ask the user to correct the error information
The system adds the information to the database of the system
The system prompt a successful message to the user
End

Figure 2.2.1.3: Activity Diagram for <Add User>

2.2.2 UC0002: Use Case <Create Account>

SCENARIOS
Basic Path. Basic Path 1. User selects "register account" in the system 2. The system shows the registration page 3. User enter the user id (alumni id) 4. The system retrieves the data from the database Exception: 4a. Data not found 5. The user check and complete the personal information 6. The system validates the information Exception: 6a. Incomplete personal details Exception: 6b. Wrong format 7. The system update the database 8. The system send an email to the user to confirm the registrations
Exception. Data not found 1. The system displays the error message 2. The system asks the user to enter valid id
Exception. Incomplete personal details 1. The system shows the error message 2. The system ask the user to complete the incomplete information
Exception. Wrong format 1. The system shows the error message 2. The system ask the user to correct the error format

CONSTRAINTS
Pre-condition. User have not created account before. [Approved, Weight is 0.]
Post-condition. User create account with specific role successfully. [Approved, Weight is 1.]

OUTGOING STRUCTURAL RELATIONSHIPS
Generalization from Create Account to Manage User [Direction is 'Source -> Destination'.]

CONNECTORS

UseCaseLink Source -> Destination
From: User : Actor, Public
To: Create Account : UseCase, Public

Table 2.2.2.1: Use Case Description for <Create Account>

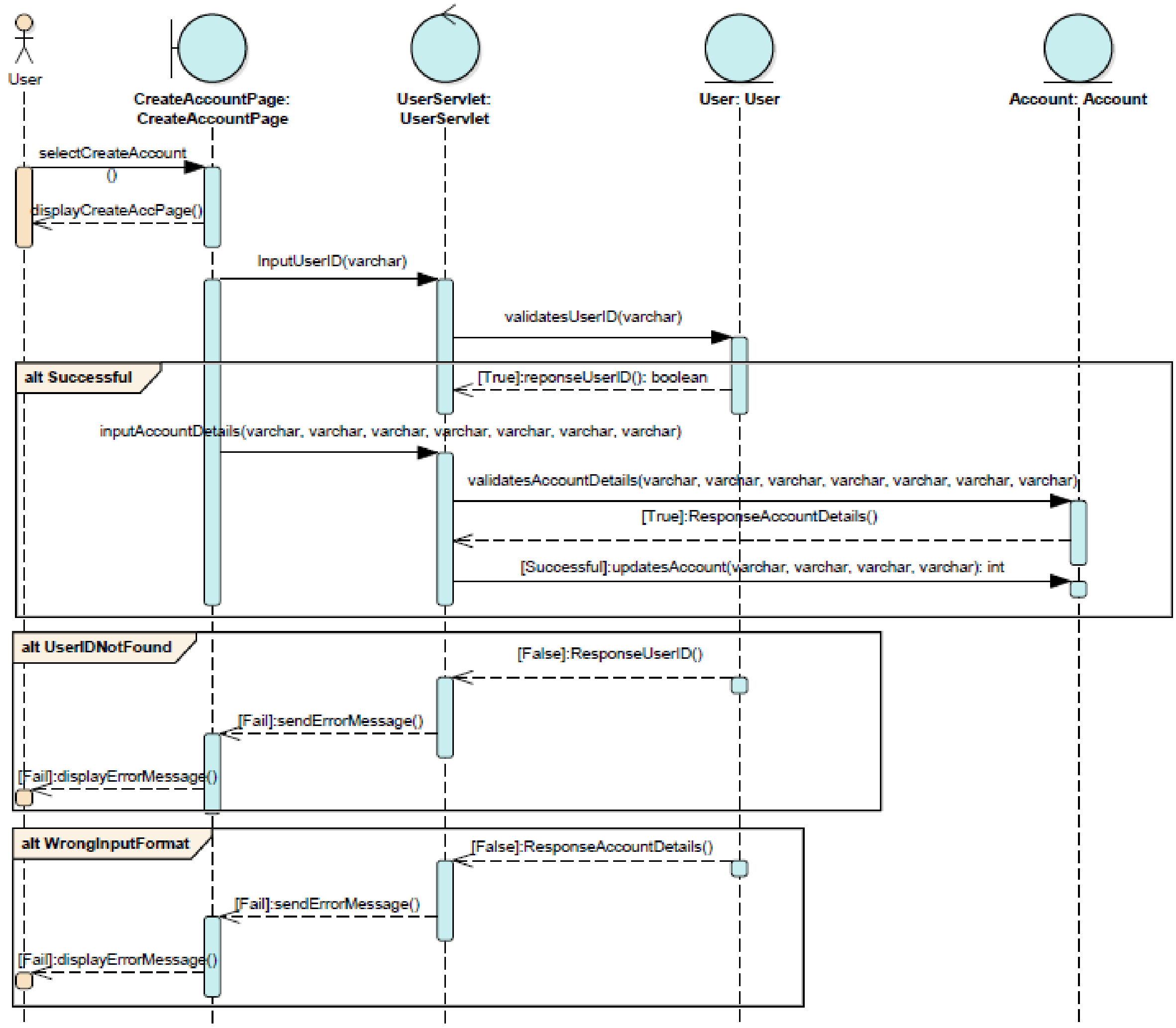

User
CreateAccountPage: CreateAccountPage
UserServlet: UserServlet
User: User
Account: Account
selectCreateAccount ()
displayCreateAccPage()
InputUserID(varchar)
validatesUserID(varchar)
alt Successful
[True]:reponseUserID(): boolean
inputAccountDetails(varchar, varchar, varchar, varchar, varchar, varchar, varchar)
validatesAccountDetails(varchar, varchar, varchar, varchar, varchar, varchar, varchar)
[True]:ResponseAccountDetails()
[Successful]:updatesAccount(varchar, varchar, varchar, varchar): int
alt UserIDNotFound
[False]:ResponseUserID()
[Fail]:sendErrorMessage()
[Fail]:displayErrorMessage()
alt WrongInputFormat
[False]:ResponseAccountDetails()
[Fail]:sendErrorMessage()
[Fail]:displayErrorMessage()

Figure 2.2.2.2: Sequence Diagram for <Create Account>

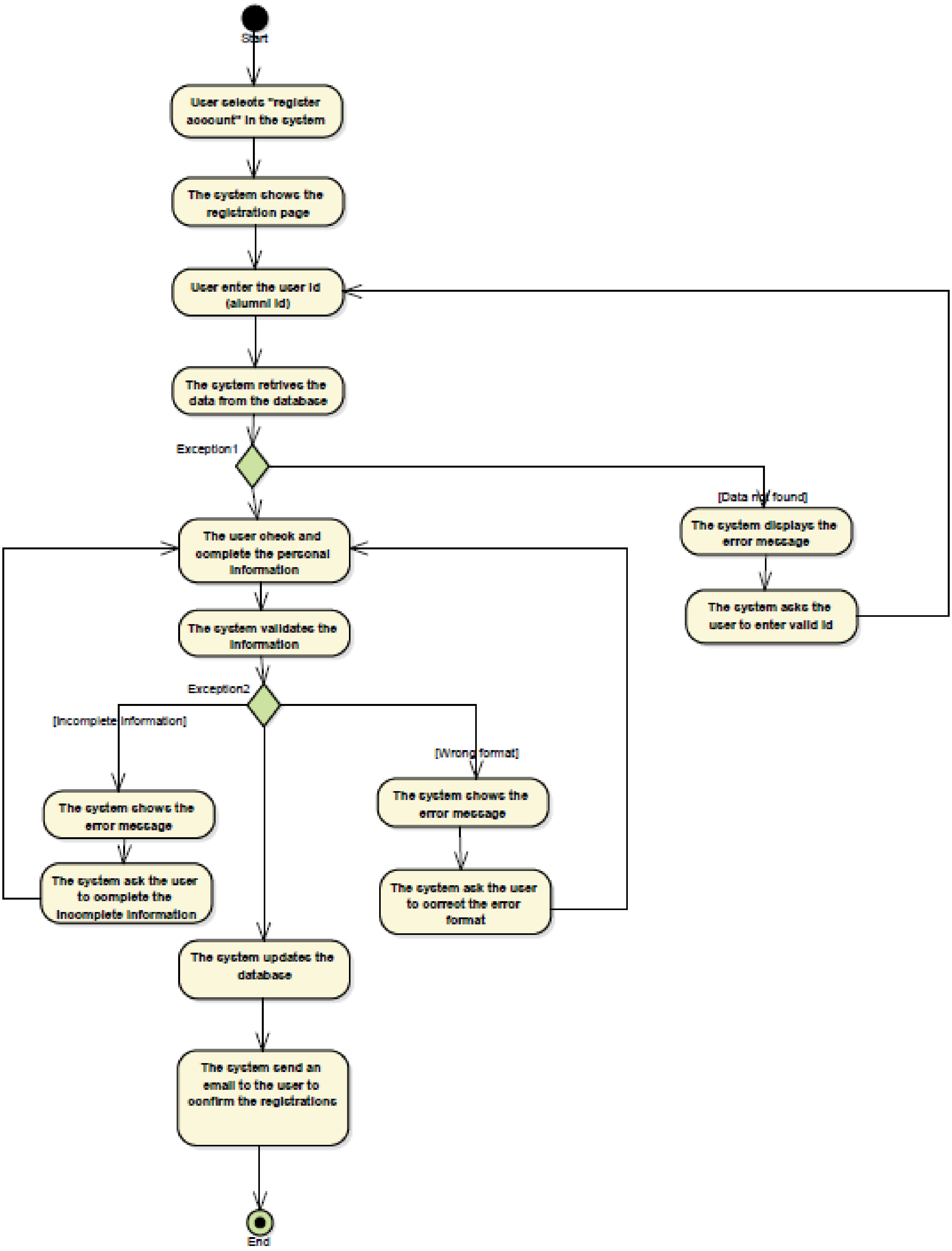

Start
User selects "register account" in the system
The system shows the registration page
User enter the user id (alumni id)
The system retrives the data from the database
Exception1
[Data not found]
The system displays the error message
The system asks the user to enter valid id
The user check and complete the personal information
The system validates the information
Exception2
[Incomplete information]
The system shows the error message
The system ask the user to complete the incomplete information
[Wrong format]
The system shows the error message
The system ask the user to correct the error format
The system updates the database
The system send an email to the user to confirm the registrations
End

Figure 2.2.2.3: Activity Diagram for <Create Account>

2.2.3 UC0003: Use Case <Login>

SCENARIOS

Basic Path. Basic Path
1. User select "log in" in the system.
2. System displays the login page.
3. The user enters the username and password
4. The system validates the credential
    Exception: 4a. Username error
    Exception: 4b. Password error
5. The system direct the user to the main page

Exception. Username error
1. The system shows the message the username is invalid
2. The system ask the user enter the correct username

Exception. Password error
1. The system shows the message the password is invalid
2. The system ask the user enter the correct password

CONSTRAINTS

Pre-condition. User have a registered account.
[Approved, Weight is 0.]

Post-condition. User successfully access the system and load into the main page.
[Approved, Weight is 2.]

OUTGOING STRUCTURAL RELATIONSHIPS

Generalization from Login to Manage User
[Direction is 'Source -> Destination'.]

CONNECTORS

UseCaseLink Source -> Destination
From: User : Actor, Public
To: Login : UseCase, Public

Table 2.2.3.1: Use Case Description for <Login>

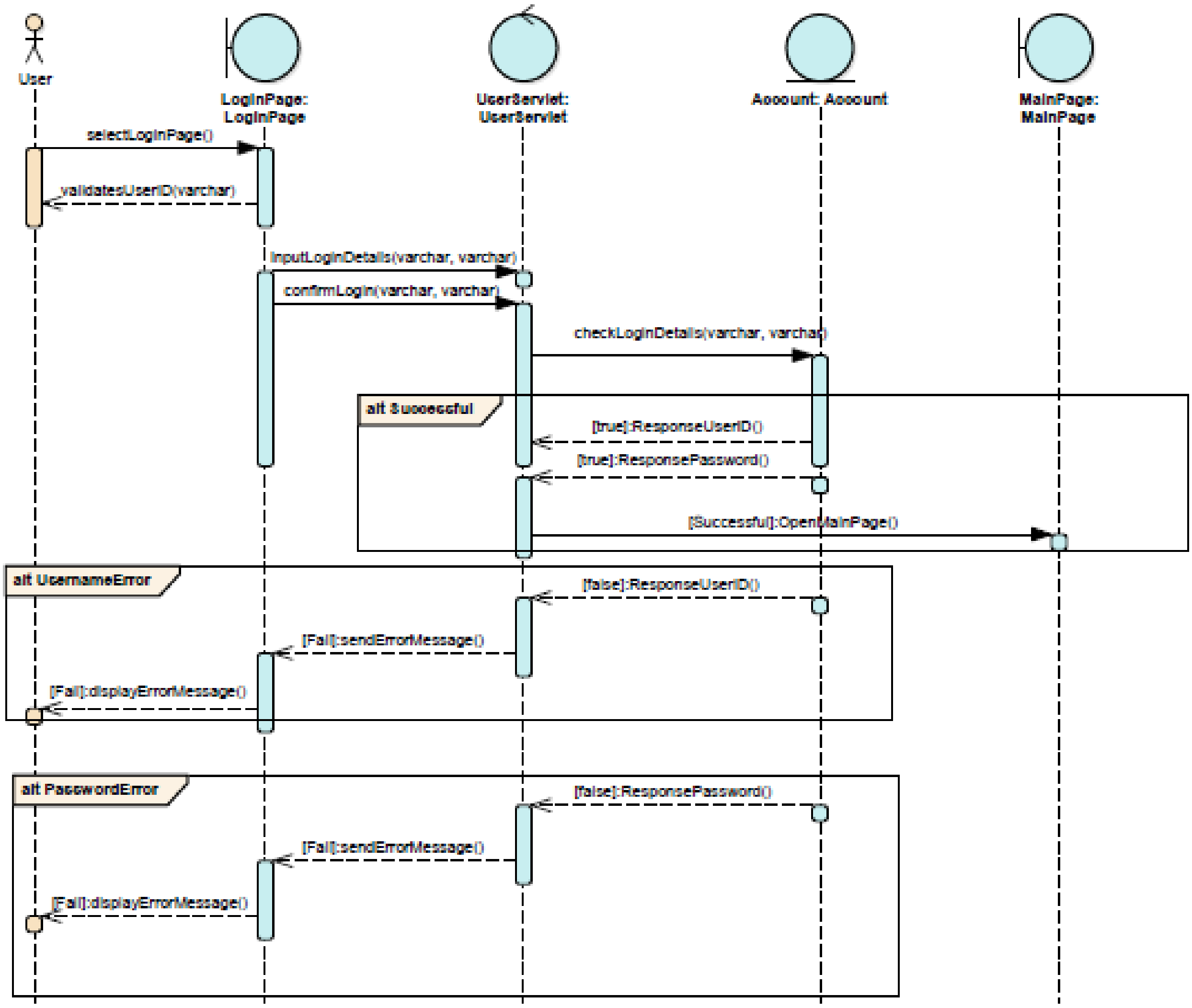

User
LoginPage: LoginPage
UserServlet: UserServlet
Account: Account
MainPage: MainPage
selectLoginPage()
validatesUserID(varchar)
inputLoginDetails(varchar, varchar)
confirmLogin(varchar, varchar)
checkLoginDetails(varchar, varchar)
alt Successful
[true]:ResponseUserID()
[true]:ResponsePassword()
[Successful]:OpenMainPage()
alt UsernameError
[false]:ResponseUserID()
[Fail]:sendErrorMessage()
[Fail]:displayErrorMessage()
alt PasswordError
[false]:ResponsePassword()
[Fail]:sendErrorMessage()
[Fail]:displayErrorMessage()

Figure 2.2.3.2: Sequence Diagram for <Login>

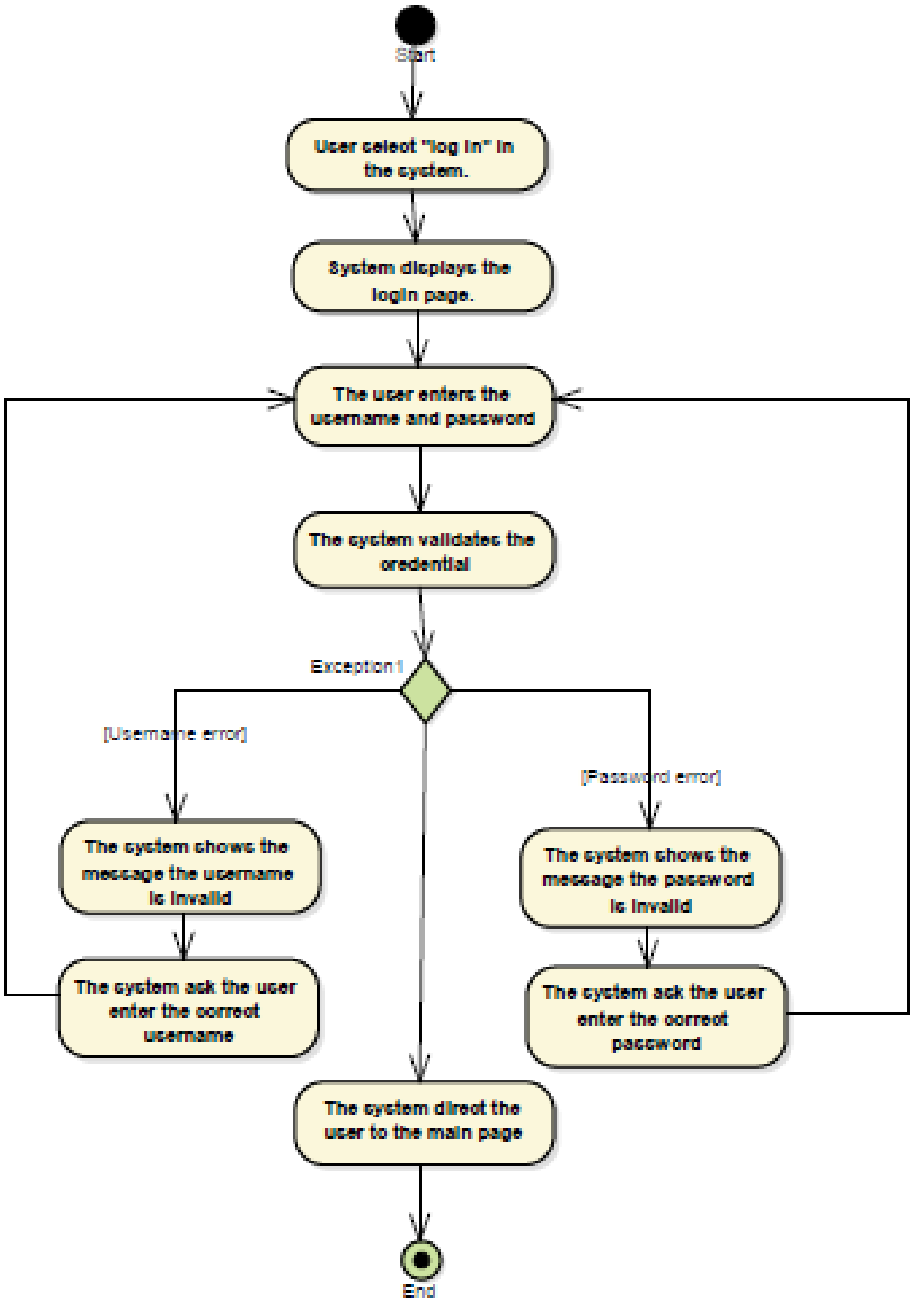

Start
User select "log in" in the system.
System displays the login page.
The user enters the username and password
The system validates the credential
Exception1
[Username error]
[Password error]
The system shows the message the username is invalid
The system ask the user enter the correct username
The system shows the message the password is invalid
The system ask the user enter the correct password
The system direct the user to the main page
End

Figure 2.2.3.3: Activity Diagram for <Login>

2.2.4 UC0004: Use Case <Remove Account>

SCENARIOS

Basic Path. Basic Path

1. The user selects "remove user"
2. The user enters and searches the user id that needs to be removed
3. The system retrieves the user id from the database
Exception: 3a. Id not found
4. The system displays the information of the user with the entered user id
5. The system prompt a message for the remove confirmation
6. The user selects the "confirm remove" button
7. The system removes the entered user information from the database
8. The system show a successfully removed user message

Exception. Id not found
1. The system shows an error message to the user
2. The system asks the user to enter a correct user id

CONSTRAINTS

Pre-condition. The user has been added into the system before.
[Approved, Weight is 0.]

Post-condition. The user is removed from the database of the system
[Approved, Weight is 1.]

OUTGOING STRUCTURAL RELATIONSHIPS

Generalization from Remove User to Manage User
[Direction is 'Source -> Destination'.]

CONNECTORS

UseCaseLink Source -> Destination
From: Admin : Actor, Public
To: Remove User : UseCase, Public

Table 2.2.4.1: Use Case Description for <Remove Account>

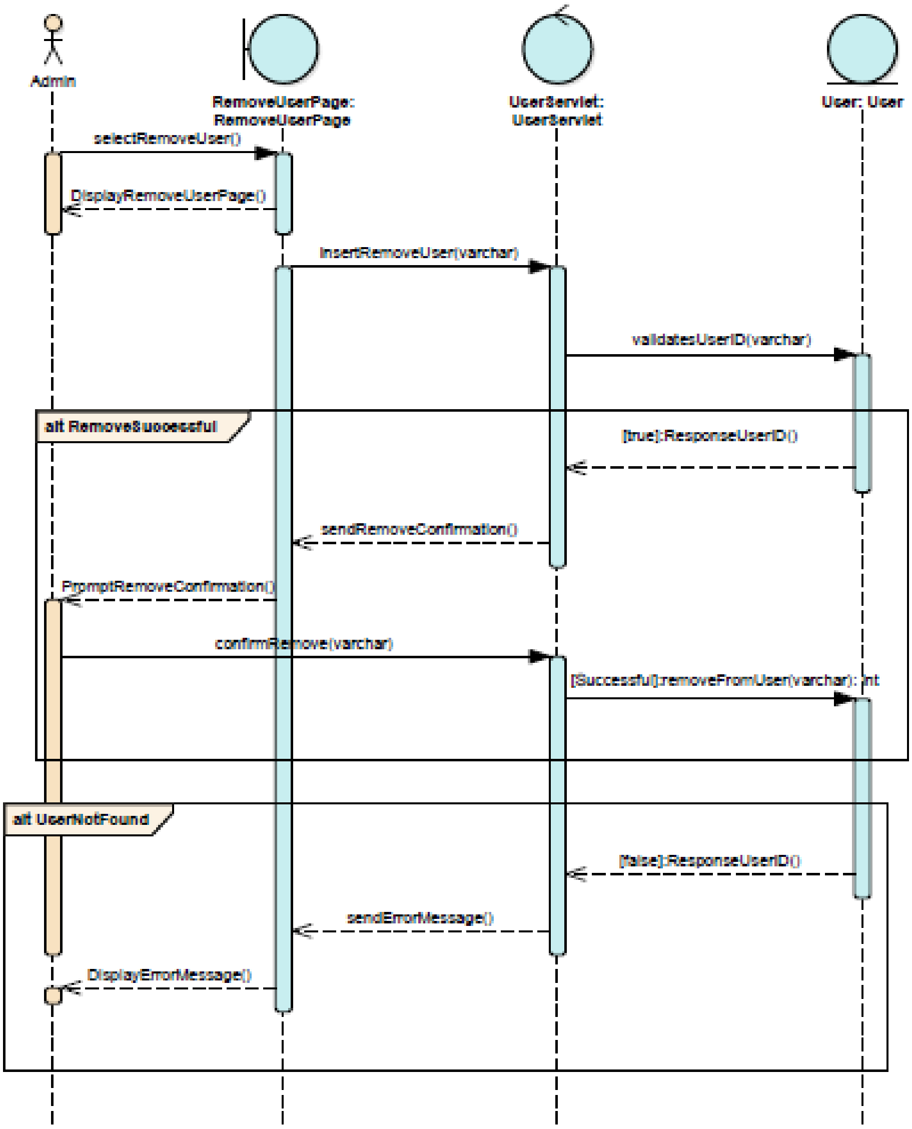


Figure 2.2.4.2: Sequence Diagram for <Remove Account>

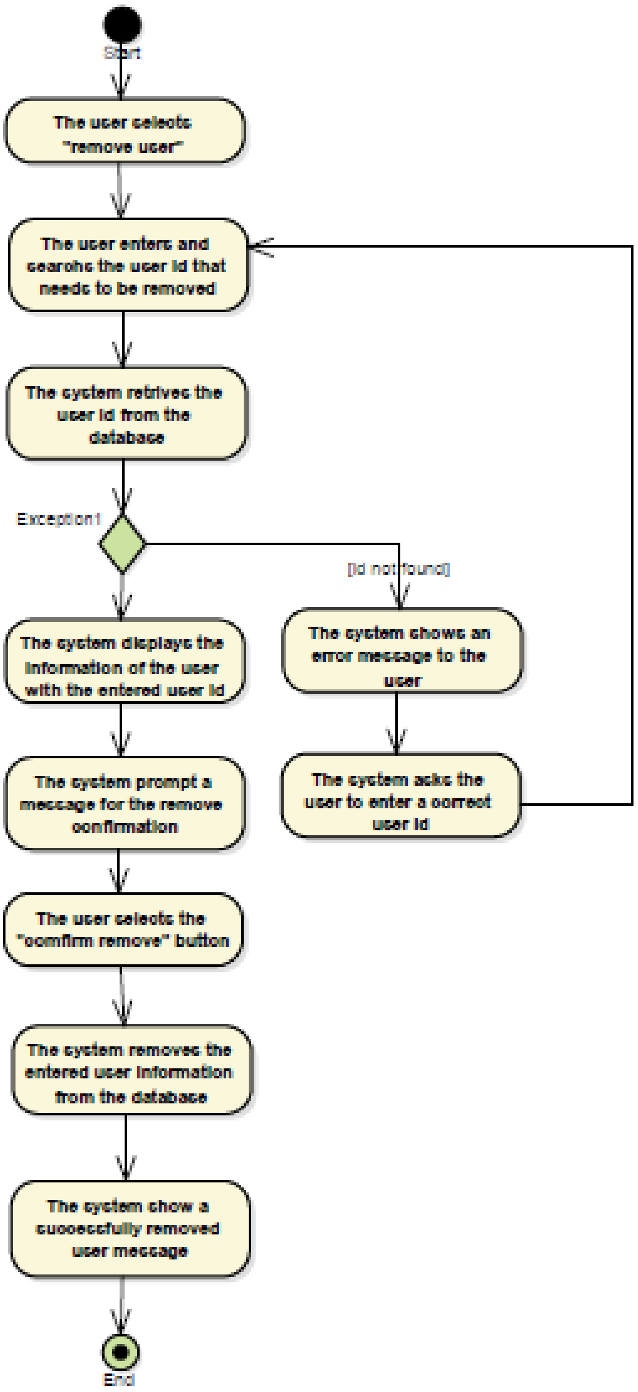

Start
The user selects "remove user"
The user enters and searchs the user id that needs to be removed
The system retrives the user id from the database
Exception1
[Id not found]
The system displays the information of the user with the entered user id
The system shows an error message to the user
The system asks the user to enter a correct user id
The system prompt a message for the remove confirmation
The user selects the "comfirm remove" button
The system removes the entered user information from the database
The system show a successfully removed user message
End

Figure 2.2.4.3: Activity Diagram for <Remove Account>

2.2.5 UC0005: Use Case <View News>

SCENARIOS
Basic Path. View News **1. The user selects ‘News’.** **2. The user clicks ‘View’ button for selected news.** **3. The system retrieves data information of selected news.** **4. The system displays news information page.** **Alternate: 4a. No Information** **5. The use case ends.**
Alternate. No Information **1. If there is no information, the system displays “No Information” message.**

CONSTRAINTS
Pre-condition. The user logged in accounts and already accessed the main page. **[Mandatory, Weight is 0.]**
Post-condition. 1The user accessed the ‘News Information’ page. **[Implemented, Weight is 1.]**

OUTGOING STRUCTURAL RELATIONSHIPS
Generalization from View news to Manage mobile **[Direction is 'Source -> Destination'.]**

Table 2.2.5.1: Use Case Description for <View News>

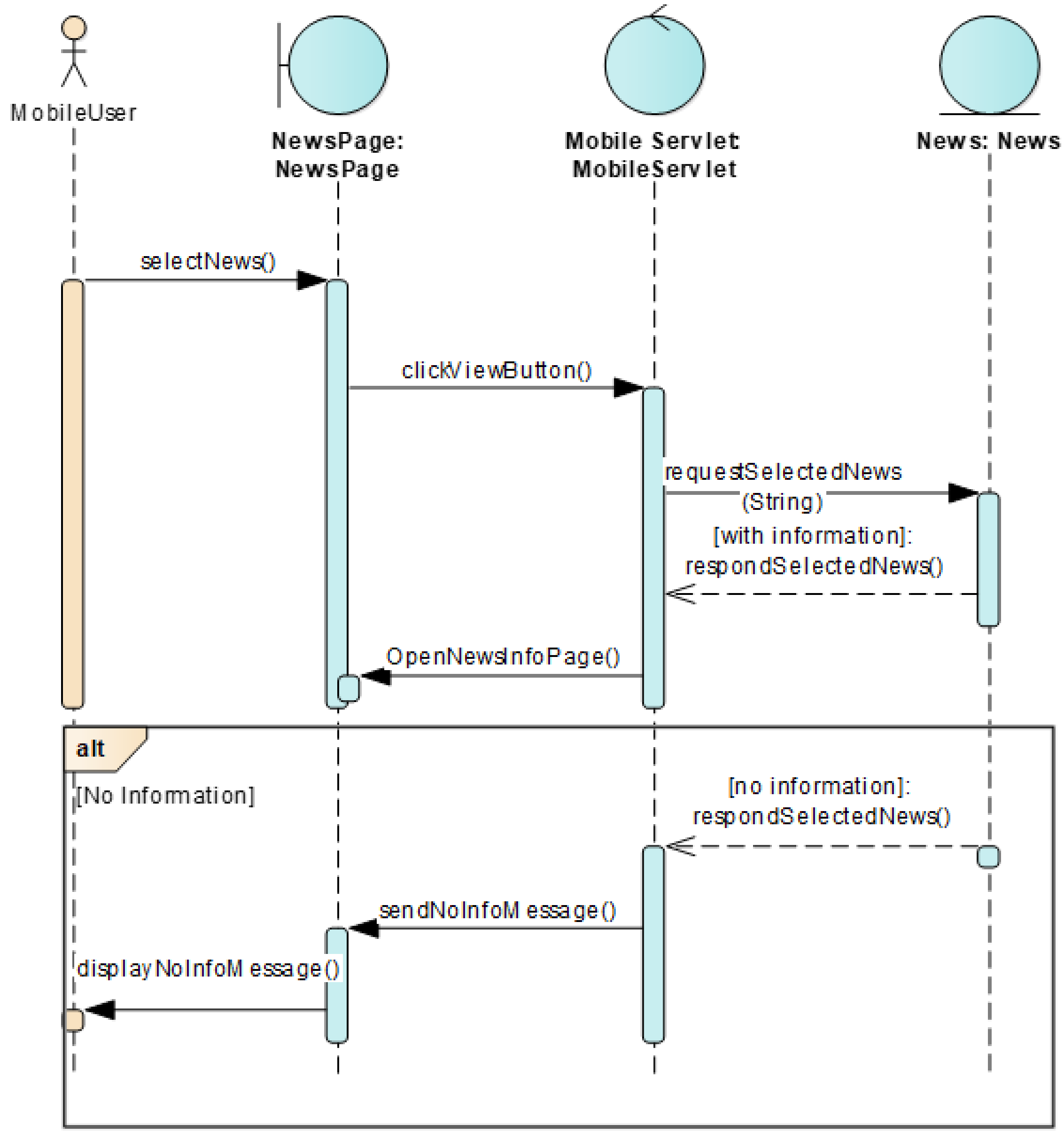


Figure 2.2.5.2: Sequence Diagram for <View News>

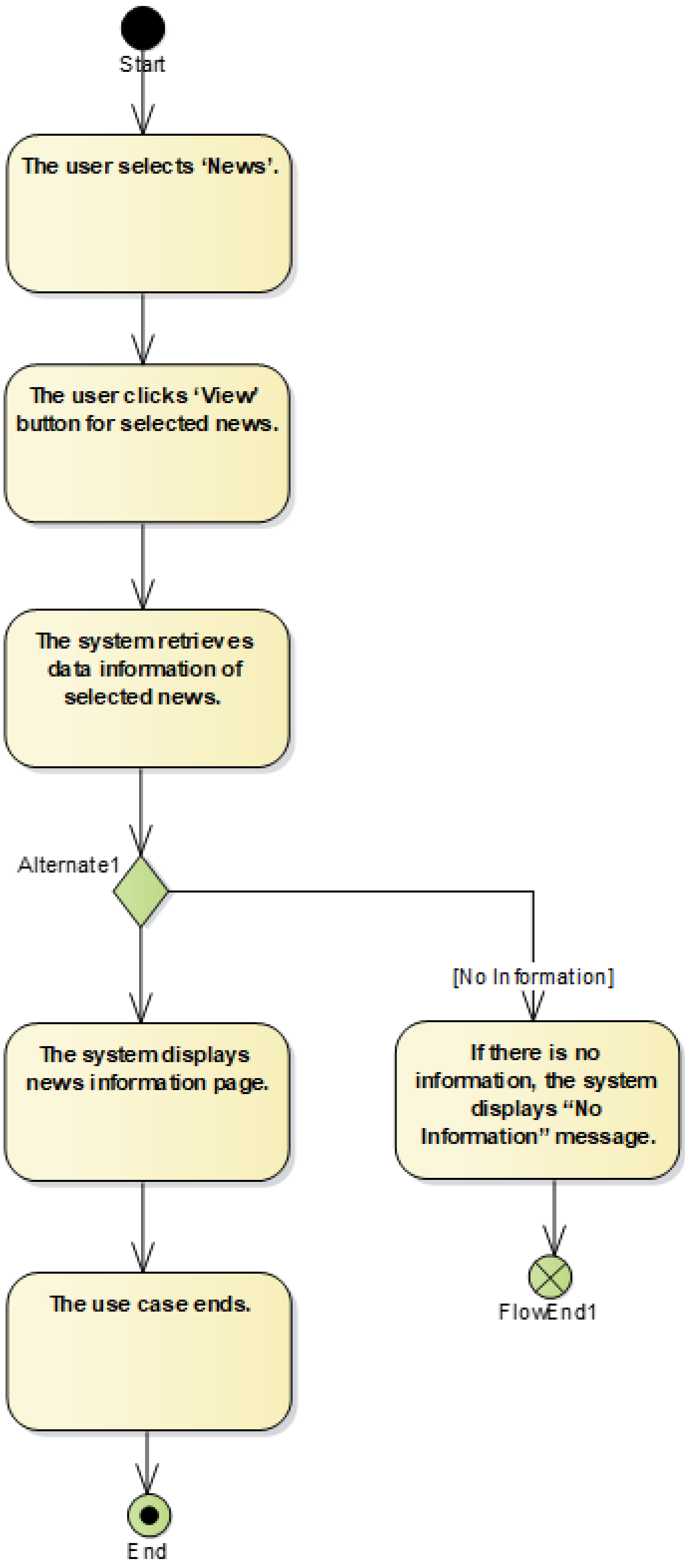

Start
The user selects 'News'.
The user clicks 'View' button for selected news.
The system retrieves data information of selected news.
Alternate1
[No Information]
The system displays news information page.
If there is no information, the system displays "No Information" message.
The use case ends.
FlowEnd1
End

Figure 2.2.5.3: Activity Diagram for <View News >

2.2.6 UC0006: Use Case <View Event>

SCENARIOS
Basic Path. View Event 1. The user selects ‘Events’. 2. The user clicks ‘View’ button for selected event. 3. The system retrieves data of selected event. 4. The system displays event information page. Alternate: 4a. No Information 5. The use case ends.
Alternate. No Information 1. If there is no information, the system displays “No Information” message.

CONSTRAINTS
Pre-condition. The user logged in accounts and already accessed the main page. [Mandatory, Weight is 0.]
Post-condition. The user accessed the ‘Events Information’ page. [Implemented, Weight is 1.]

OUTGOING STRUCTURAL RELATIONSHIPS
Generalization from View event to Manage mobile [Direction is 'Source -> Destination'.]

Table 2.2.6.1: Use Case Description for <View Event>

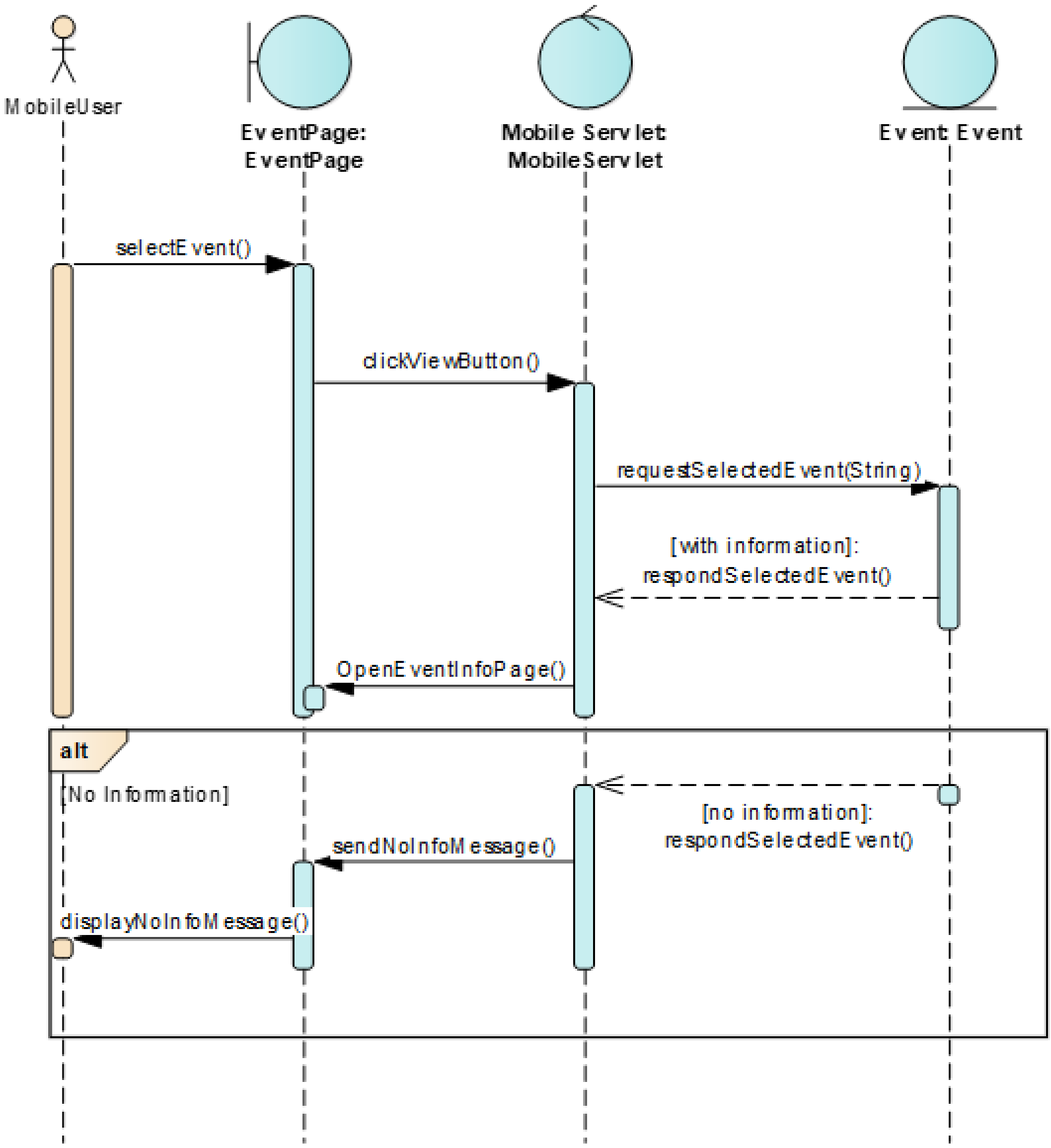


Figure 2.2.6.2: Sequence Diagram for <View Event>

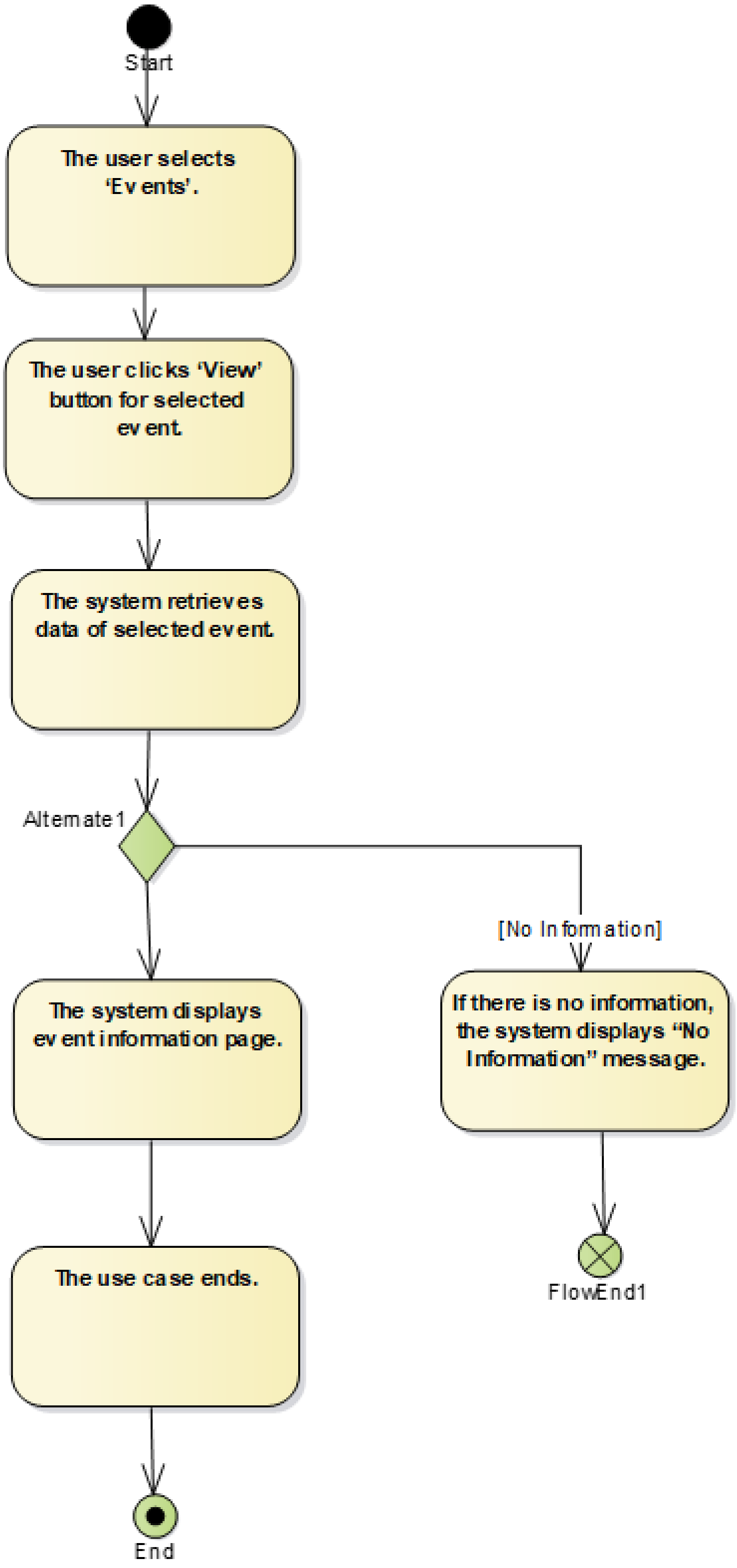

Start
The user selects 'Events'.
The user clicks 'View' button for selected event.
The system retrieves data of selected event.
Alternate1
[No Information]
The system displays event information page.
If there is no information, the system displays "No Information" message.
The use case ends.
FlowEnd1
End

Figure 2.2.6.3: Activity Diagram for <View Event>

2.2.7 UC0007: Use Case <View Charity>

SCENARIOS
Basic Path. View Charity 1. The user selects ‘Charity’. 2. The user clicks the ‘View’ button for selected charity. 3. The system retrieves data information of selected charity. 4. The system displays charity information. Alternate: 4a. No information 5. The use case ends.
Alternate. No information 1. If there is no information, the system displays a “No Information” message.

CONSTRAINTS
Pre-condition. The user logged in accounts and already accessed the main page. [Mandatory, Weight is 0.]
Post-condition. The user accessed to 'Charity Information' page. [Implemented, Weight is 1.]

OUTGOING STRUCTURAL RELATIONSHIPS
Generalization from View charity to Manage mobile [Direction is 'Source -> Destination'.]

Table 2.2.7.1: Use Case Description for <View Charity>

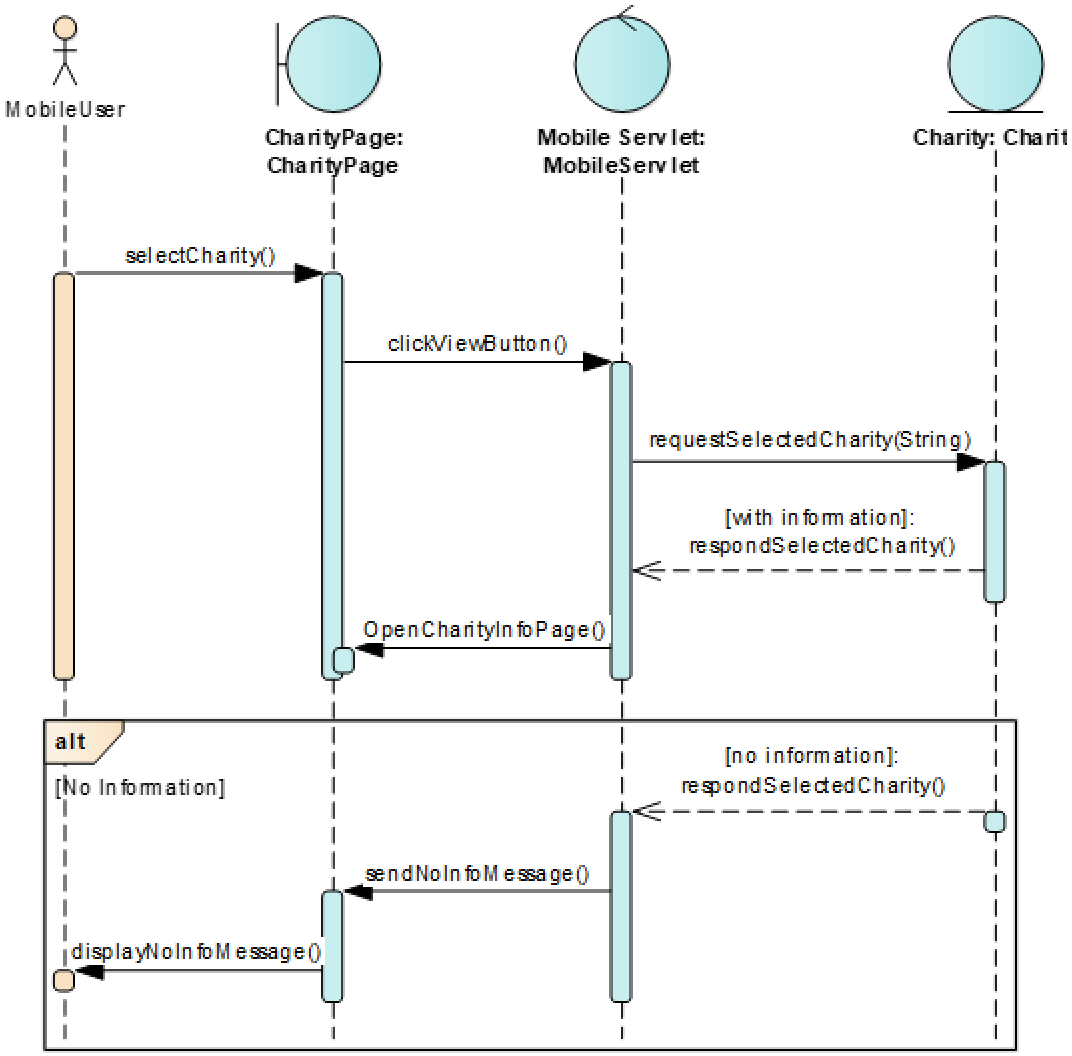


Figure 2.2.7.2: Sequence Diagram for <View Charity>

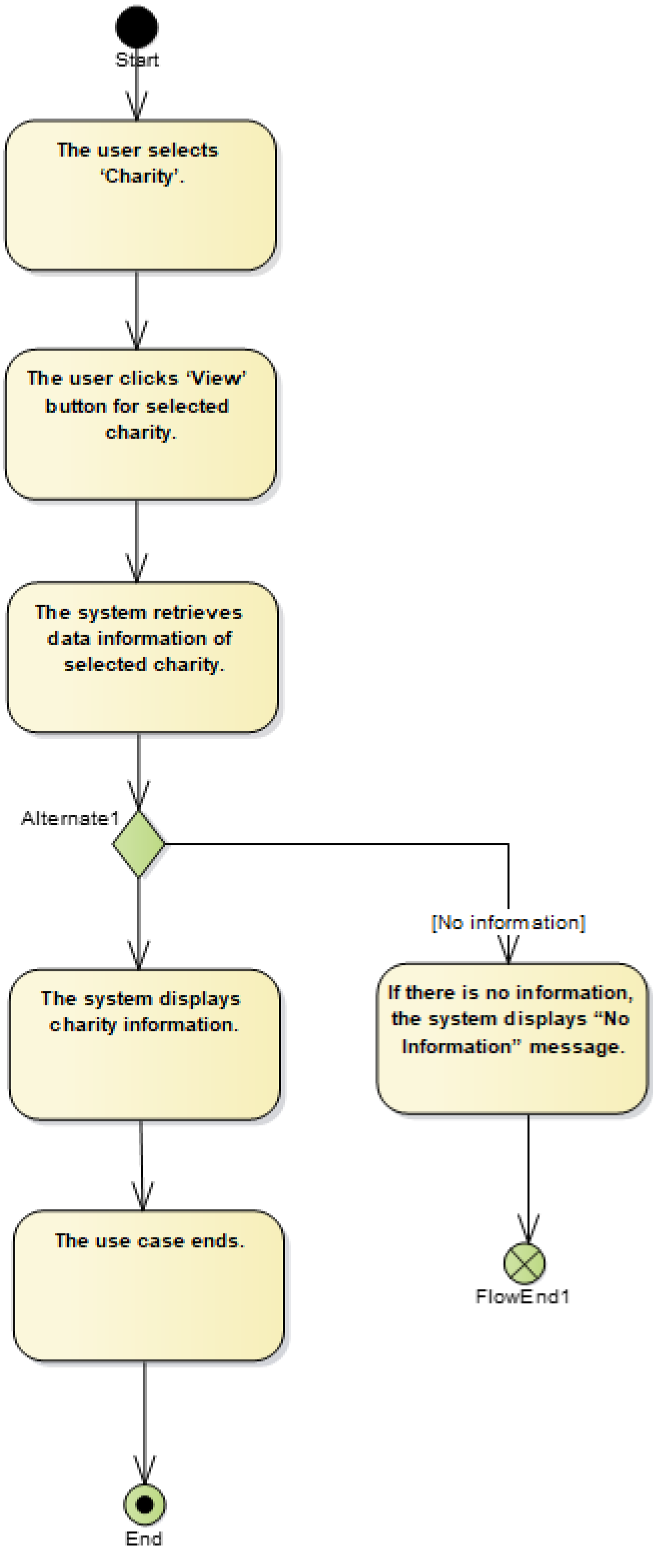

Start
The user selects 'Charity'.
The user clicks 'View' button for selected charity.
The system retrieves data information of selected charity.
Alternate1
[No information]
The system displays charity information.
If there is no information, the system displays "No Information" message.
The use case ends.
FlowEnd1
End

Figure 2.2.7.3: Activity Diagram for <View Charity>

2.2.8 UC0008: Use Case <View Report>

SCENARIOS

Basic Path. View Report

1. The user selects ‘Report’.

2. The user clicks ‘View’ button for selected report category.

3. The system retrieves data information of selected report generated by staff.

4. The system displays report information.

Alternate: 4a. No Information

5. The use case ends.

Alternate. No Information

1. If there is no information, the system displays “No Information” message.

CONSTRAINTS

Pre-condition. The user logged in accounts and already accessed the main page.

[Mandatory, Weight is 0.]

Post-condition. The user accessed to ‘Report Information’ page.

[Implemented, Weight is 1.]

OUTGOING STRUCTURAL RELATIONSHIPS

Generalization from View report to Manage mobile

[Direction is 'Source -> Destination'.]

Table 2.2.8.1: Use Case Description for <View Report>

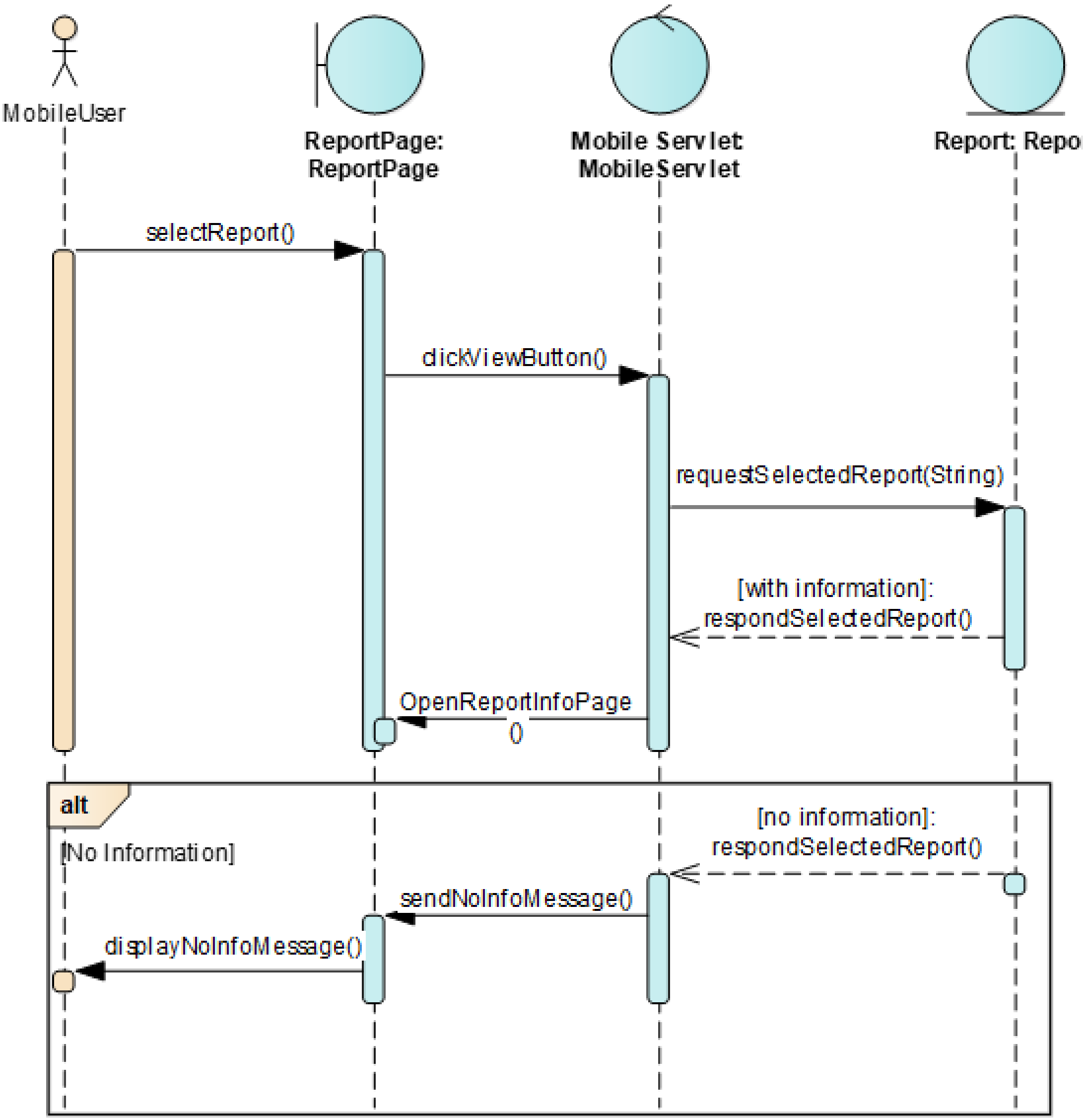


Figure 2.2.8.2: Sequence Diagram for <View Report>

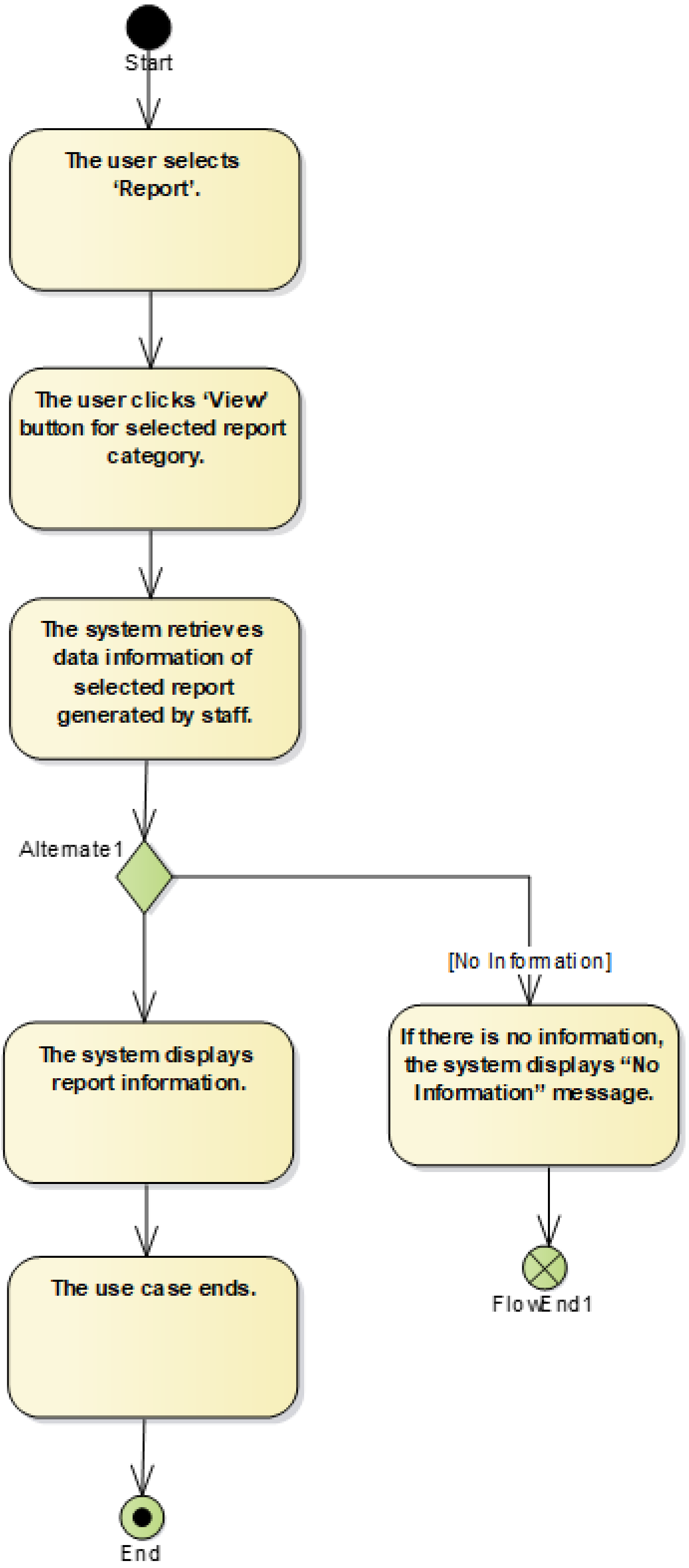

Start
The user selects 'Report'.
The user clicks 'View' button for selected report category.
The system retrieves data information of selected report generated by staff.
Alternate1
[No Information]
The system displays report information.
If there is no information, the system displays "No Information" message.
The use case ends.
FlowEnd1
End

Figure 2.2.8.3: Activity Diagram for <View Report>

2.2.9 UC0009: Use Case <Update Architecture Licensing>

SCENARIOS
Basic Path. Basic Path 1. The use case begins when alumni select the “Update Architecture Licensing” from the main menu. 2. The system displays the existing architecture license of alumni. 3. The system asks the alumni to renew the architecture license. 4. The alumni update the latest architecture licensing. 5. The system validates the updated license. Exception: 5a. invalid 6. The system updates the architecture license.
Exception. invalid 1. The alumni update invalid architecture license 2. The system display an error with the architecture license updated by the alumni.

CONSTRAINTS
Pre-condition. Alumni logged on into the system [Approved, Weight is 0.]
Pre-condition. The system must have an existing architecture license of alumni. [Approved, Weight is 1.]
Post-condition. The architecture license is updated. [Approved, Weight is 2.]

Table 2.2.9.1: Use Case Description for <Update Architecture Licensing>

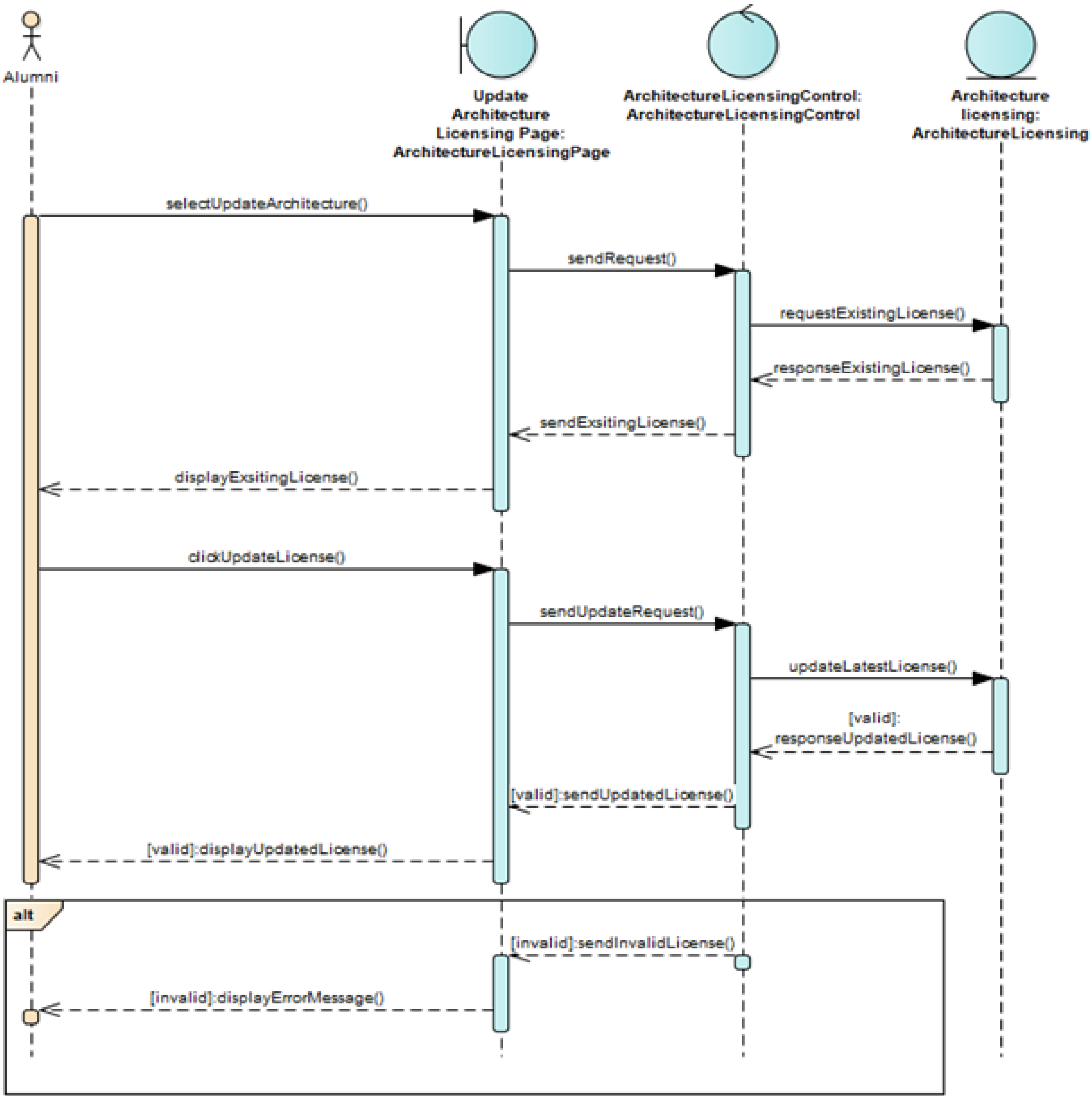

Alumni
Update Architecture Licensing Page: ArchitectureLicensingPage
ArchitectureLicensingControl: ArchitectureLicensingControl
Architecture licensing: ArchitectureLicensing
selectUpdateArchitecture()
sendRequest()
requestExistingLicense()
responseExistingLicense()
sendExsitingLicense()
displayExsitingLicense()
clickUpdateLicense()
sendUpdateRequest()
updateLatestLicense()
[valid]: responseUpdatedLicense()
[valid]:sendUpdatedLicense()
[valid]:displayUpdatedLicense()
alt
[invalid]:sendInvalidLicense()
[invalid]:displayErrorMessage()

Figure 2.2.9.2: Sequence Diagram for <Update Architecture Licensing>

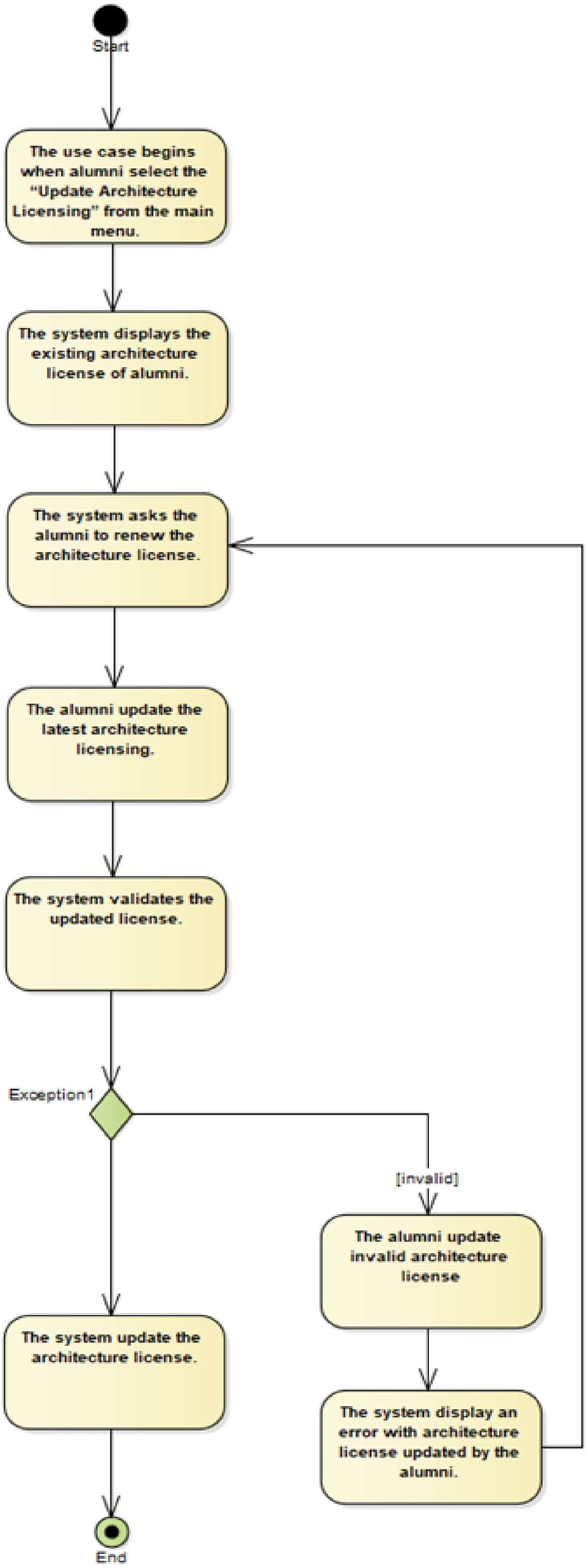

Start
The use case begins when alumni select the "Update Architecture Licensing" from the main menu.
The system displays the existing architecture license of alumni.
The system asks the alumni to renew the architecture license.
The alumni update the latest architecture licensing.
The system validates the updated license.
Exception1
[invalid]
The alumni update invalid architecture license
The system display an error with architecture license updated by the alumni.
The system update the architecture license.
End

Figure 2.2.9.3: Activity Diagram for <Update Architecture Licensing>

2.2.10 UC0010: Use Case <View Alumni List>

SCENARIOS
Basic Path. Basic Path 1. The use case begins when alumni select the “Alumni List” from the main menu. 2. The system retrieves the information from the alumni list. 3. The system prepares and displays the alumni list on the “Alumni List” page. Alternate: 3a. alternate 1 4. Alumni view the alumni list information.
Alternate. alternate 1 1. If there is no alumni list information. 2. The system informs the alumni that the alumni list is not update yet.

CONSTRAINTS
Pre-condition. Alumni logged on into the system [Approved, Weight is 0.]
Post-condition. Alumni will be able to view the alumni list. [Approved, Weight is 1.]

Table 2.2.10.1: Use Case Description for <View Alumni List>

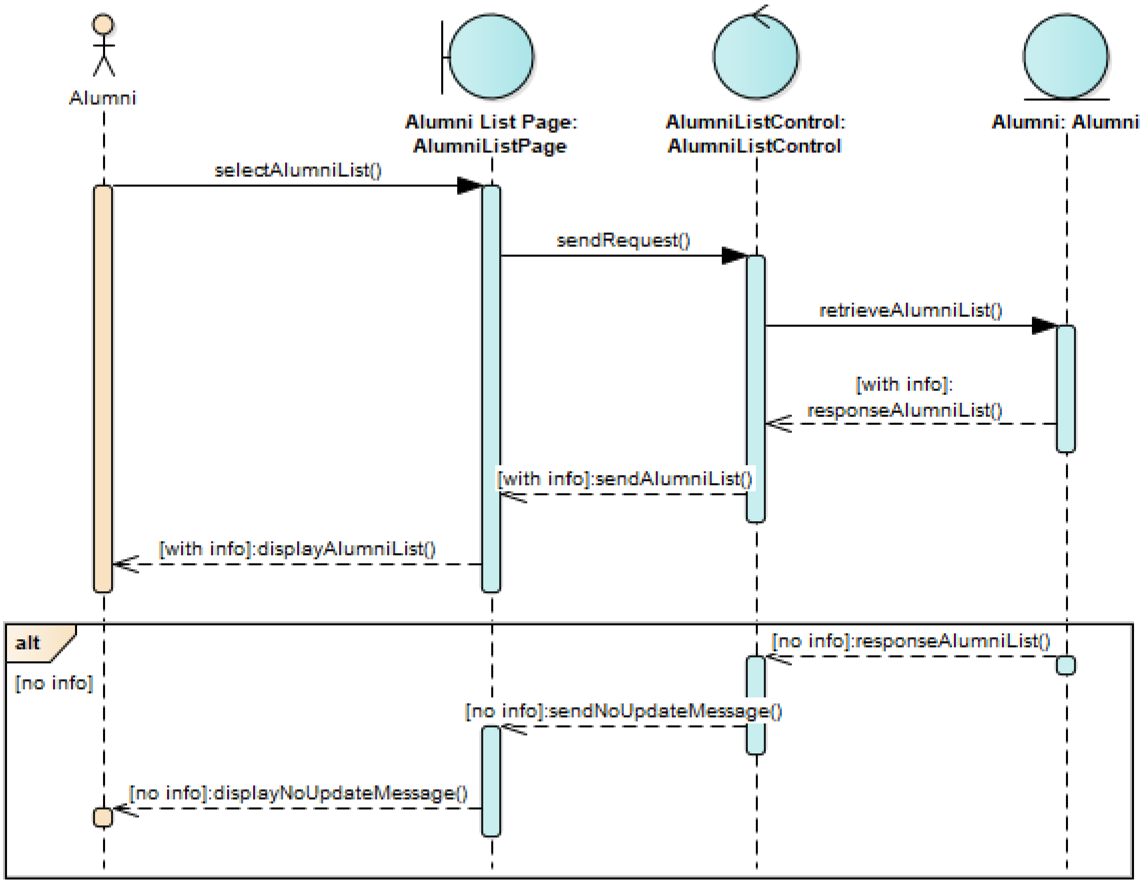

Alumni
Alumni List Page: AlumniListPage
AlumniListControl: AlumniListControl
Alumni: Alumni
selectAlumniList()
sendRequest()
retrieveAlumniList()
[with info]: responseAlumniList()
[with info]:sendAlumniList()
[with info]:displayAlumniList()
alt
[no info]
[no info]:responseAlumniList()
[no info]:sendNoUpdateMessage()
[no info]:displayNoUpdateMessage()

Figure 2.2.10.2: Sequence Diagram for <View Alumni List>

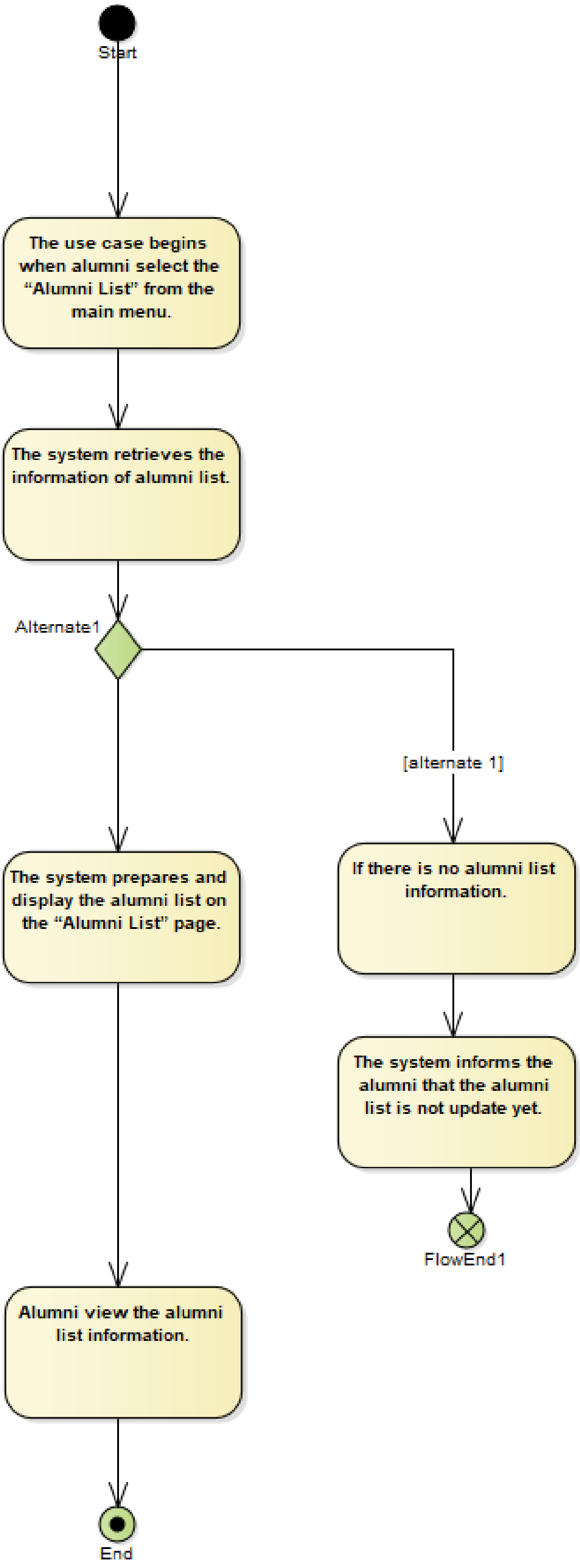

Start
The use case begins when alumni select the “Alumni List” from the main menu.
The system retrieves the information of alumni list.
Alternate1
[alternate 1]
The system prepares and display the alumni list on the “Alumni List” page.
If there is no alumni list information.
The system informs the alumni that the alumni list is not update yet.
FlowEnd1
Alumni view the alumni list information.
End

Figure 2.2.10.3: Activity Diagram for <View Alumni List>

2.2.11 UC0011: Use Case <View Alumni Profile>

SCENARIOS
Basic Path. Basic Path 1. The use case begins when alumni select the “Profile” from the main menu. 2. The system retrieves the information of alumni profiles. 3. The system prepares and displays the alumni profile on the “Profile” page. Alternate: 3a. alternate 1 4. Alumni view their profile information.
Alternate. alternate 1 1. If there is no alumni profile information 2. The system informs the alumni that the alumni profile information is not update yet.

CONSTRAINTS
Pre-condition. Alumni logged on into the system [Approved, Weight is 0.]
Post-condition. Alumni will be able to view their profile. [Approved, Weight is 1.]

Table 2.2.11.1: Use Case Description for <View Alumni Profile>

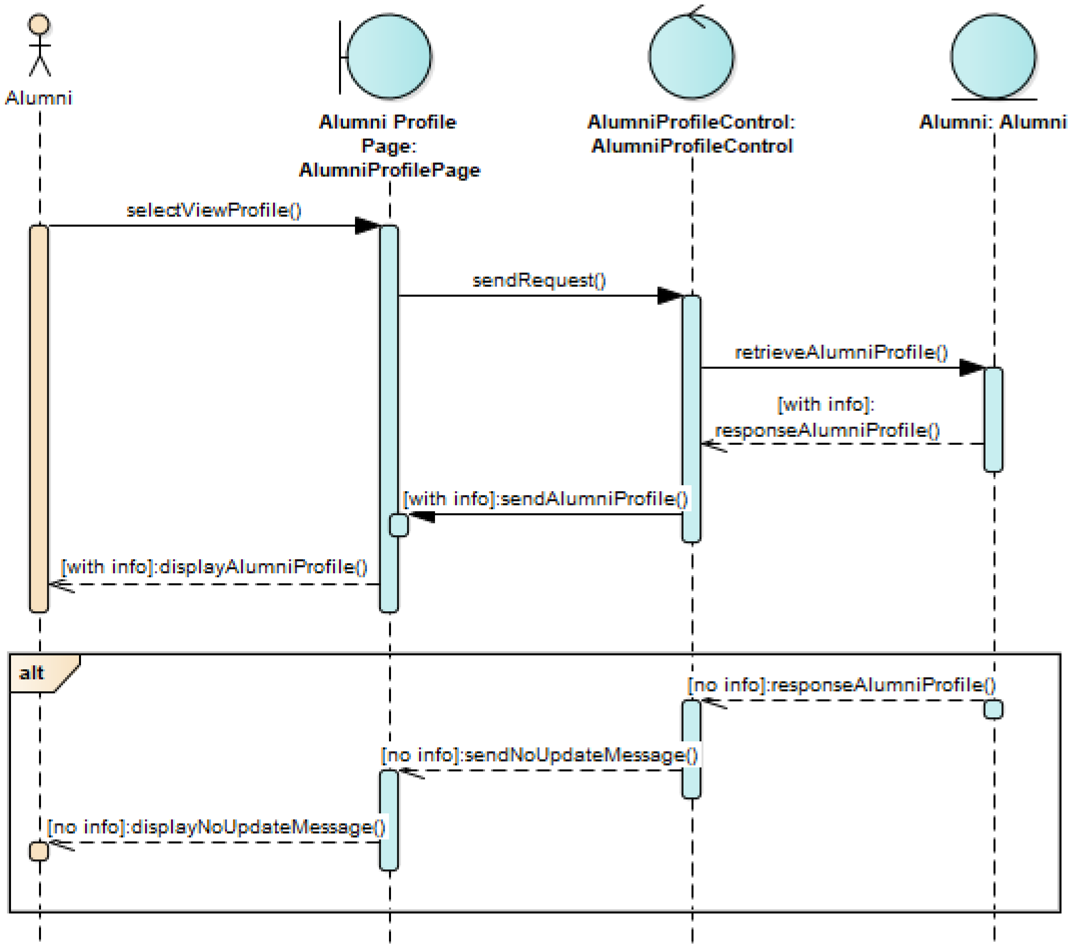

Alumni
Alumni Profile Page: AlumniProfilePage
AlumniProfileControl: AlumniProfileControl
Alumni: Alumni
selectViewProfile()
sendRequest()
retrieveAlumniProfile()
[with info]: responseAlumniProfile()
[with info]:sendAlumniProfile()
[with info]:displayAlumniProfile()
alt
[no info]:responseAlumniProfile()
[no info]:sendNoUpdateMessage()
[no info]:displayNoUpdateMessage()

Figure 2.2.11.2: Sequence Diagram for <View Alumni Profile>

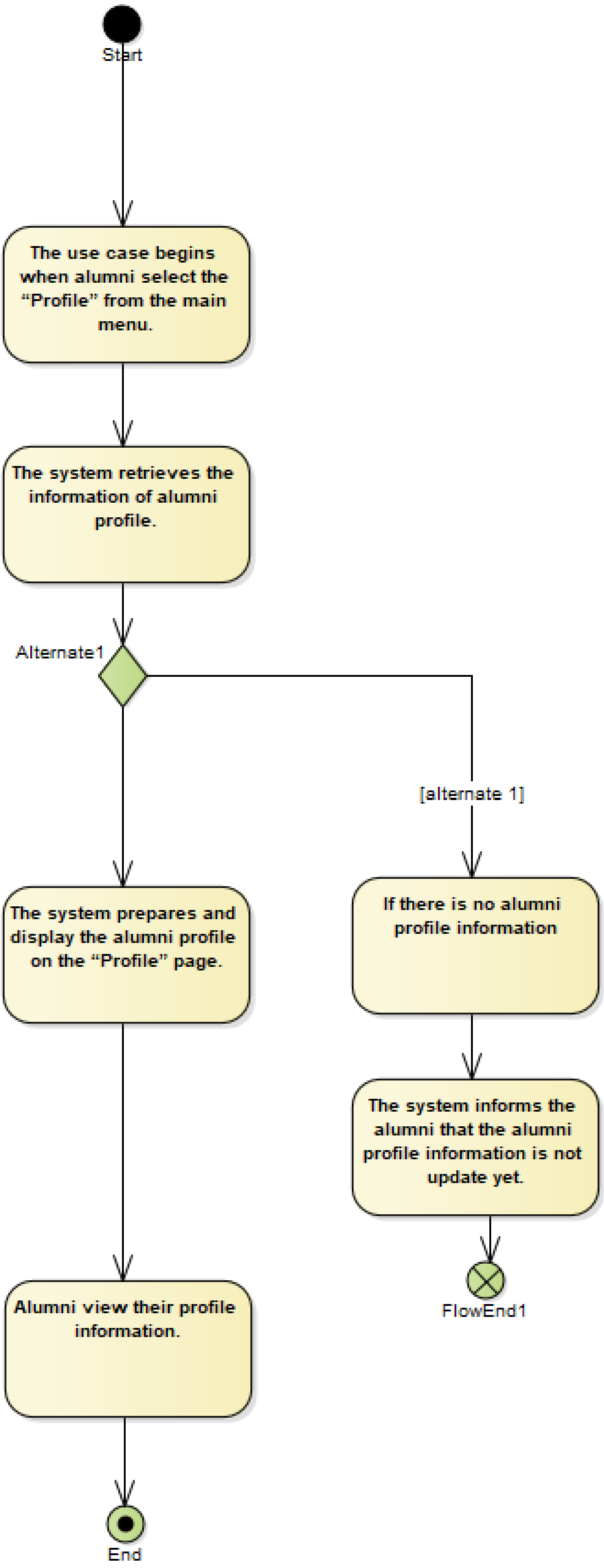

Start
The use case begins when alumni select the “Profile” from the main menu.
The system retrieves the information of alumni profile.
Alternate1
[alternate 1]
The system prepares and display the alumni profile on the “Profile” page.
If there is no alumni profile information
The system informs the alumni that the alumni profile information is not update yet.
FlowEnd1
Alumni view their profile information.
End

Figure 2.2.11.3: Activity Diagram for <View Alumni Profile>

2.2.12 UC0012: Use Case <View Dashboard>

SCENARIOS
Basic Path. Basic Path 1. The use case begins when alumni select the “Dashboard” from the main menu. 2. The system retrieves the information from the dashboard. 3. The system prepares and displays the dashboard on the "Dashboard” page. Alternate: 3a. alternate 1 4. Alumni view the dashboard information.
Alternate. alternate 1 1. If there is no dashboard information. 2. The system informs the alumni that the dashboard is not update yet.

CONSTRAINTS
Pre-condition. Alumni logged on into the system [Approved, Weight is 0.]
Post-condition. Alumni will be able to view the dashboard. [Approved, Weight is 1.]

Table 2.2.12.1: Use Case Description for <View Dashboard>

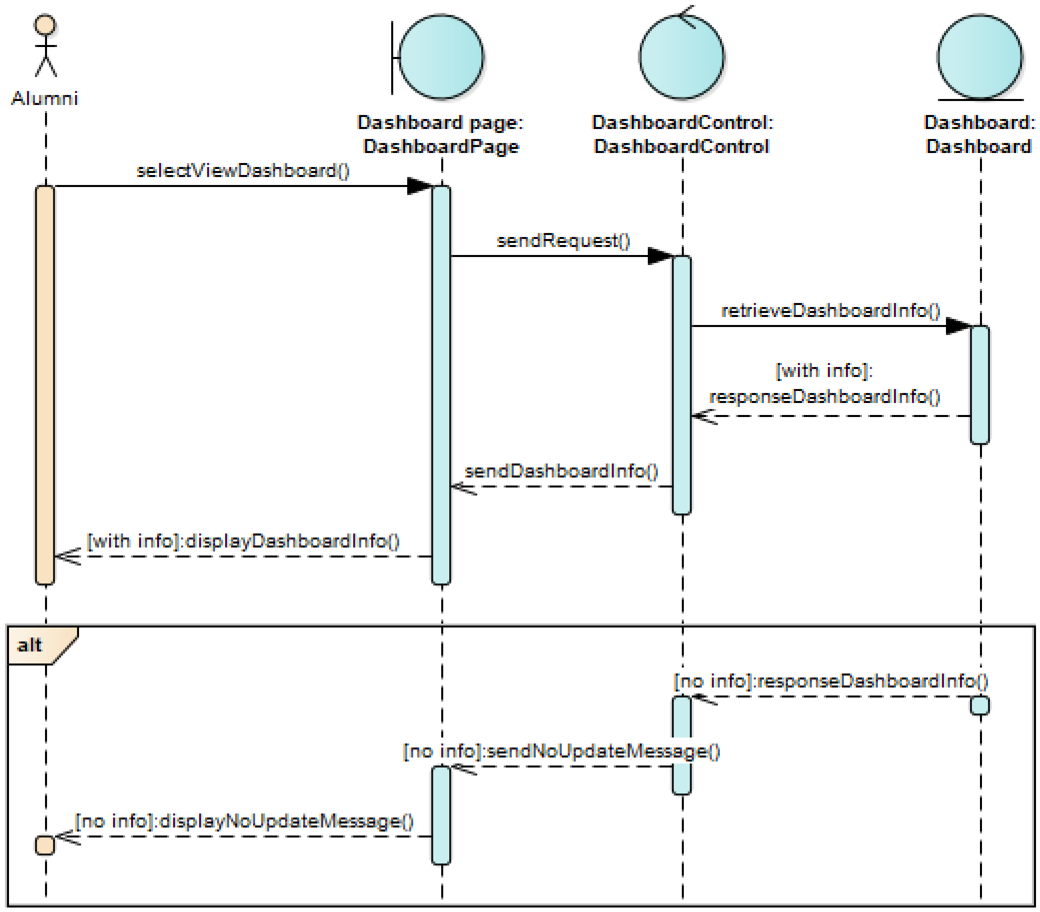

Alumni
Dashboard page: DashboardPage
DashboardControl: DashboardControl
Dashboard: Dashboard
selectViewDashboard()
sendRequest()
retrieveDashboardInfo()
[with info]: responseDashboardInfo()
sendDashboardInfo()
[with info]:displayDashboardInfo()
alt
[no info]:responseDashboardInfo()
[no info]:sendNoUpdateMessage()
[no info]:displayNoUpdateMessage()

Figure 2.2.12.2: Sequence Diagram for <View Dashboard>

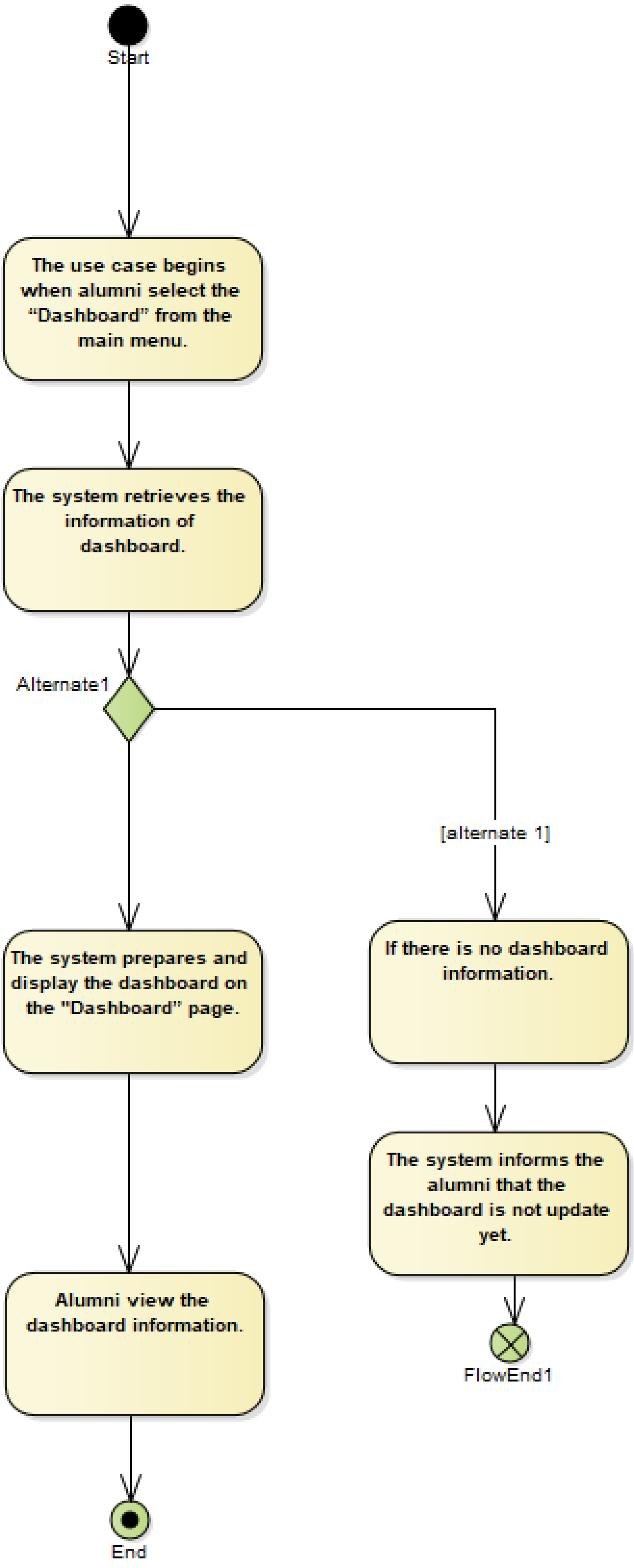

Start
The use case begins when alumni select the “Dashboard” from the main menu.
The system retrieves the information of dashboard.
Alternate1
[alternate 1]
The system prepares and display the dashboard on the "Dashboard” page.
If there is no dashboard information.
The system informs the alumni that the dashboard is not update yet.
FlowEnd1
Alumni view the dashboard information.
End

Figure 2.2.12.3: Activity Diagram for <View Dashboard>

2.2.13 UC0013: Use Case <Filter Friend>

SCENARIOS
Basic Path. Basic Path 1.The alumni select “filter friend”. 2. The alumni select the filter criteria. 3. The system retrieves the friend information Alternate: 3a. alternate 1 4.The system search the friend with matched criteria Alternate: 4a. alternate 2 5.The system retrieves the user information 6.The system displays the result of the friend
Alternate. alternate 1 1. If there is no friend found Alternate. alternate 2 1. If there is no match result

CONSTRAINTS
Pre-condition. Alumni logged on into the system [Approved, Weight is 0.]
Post-condition. Alumni will be able to view the friend list on the dashboard. [Approved, Weight is 1.]

Table 2.2.13.1: Use Case Description for <Filter Friend>

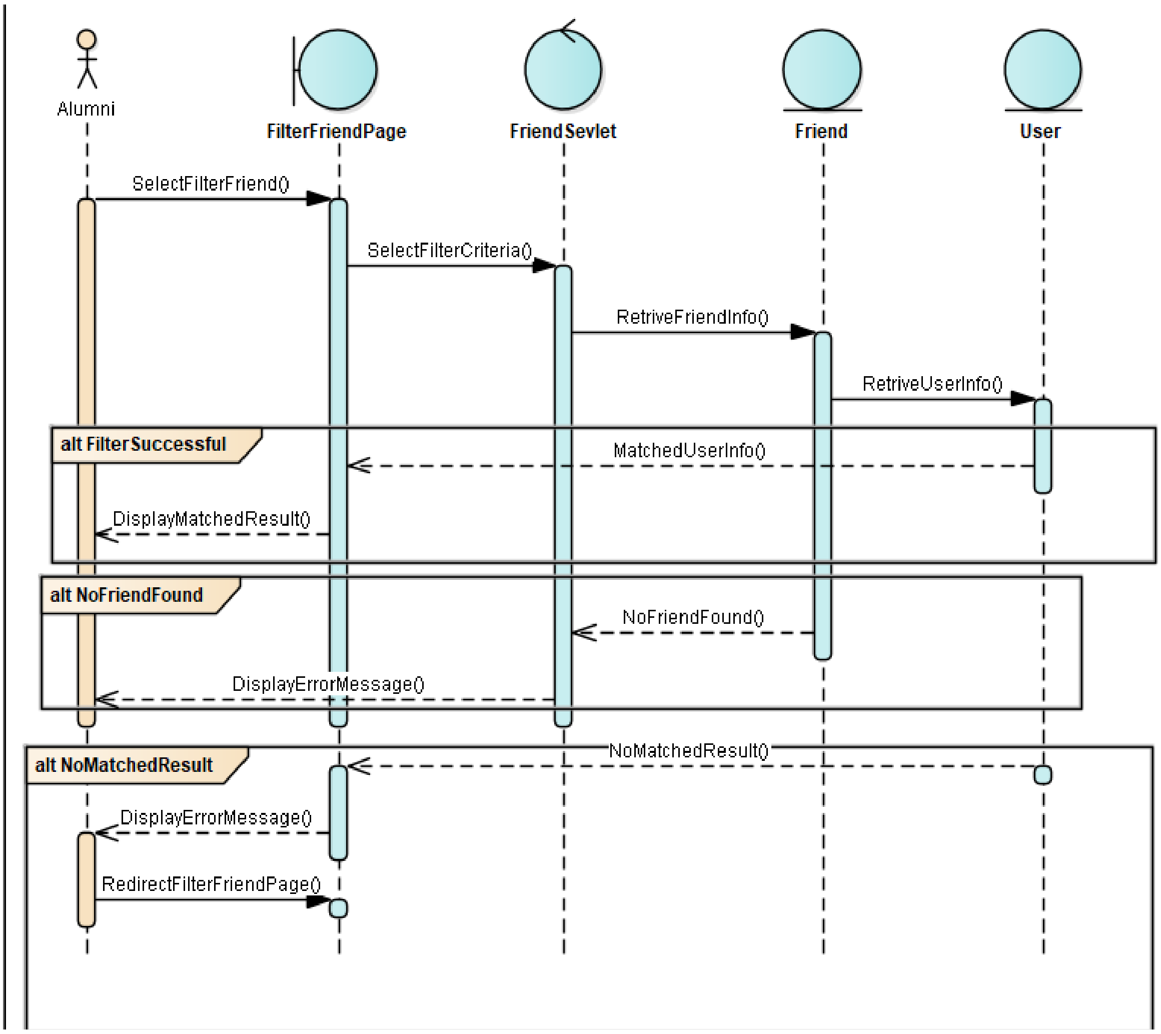


Figure 2.2.13.2: Sequence Diagram for <Filter Friend>

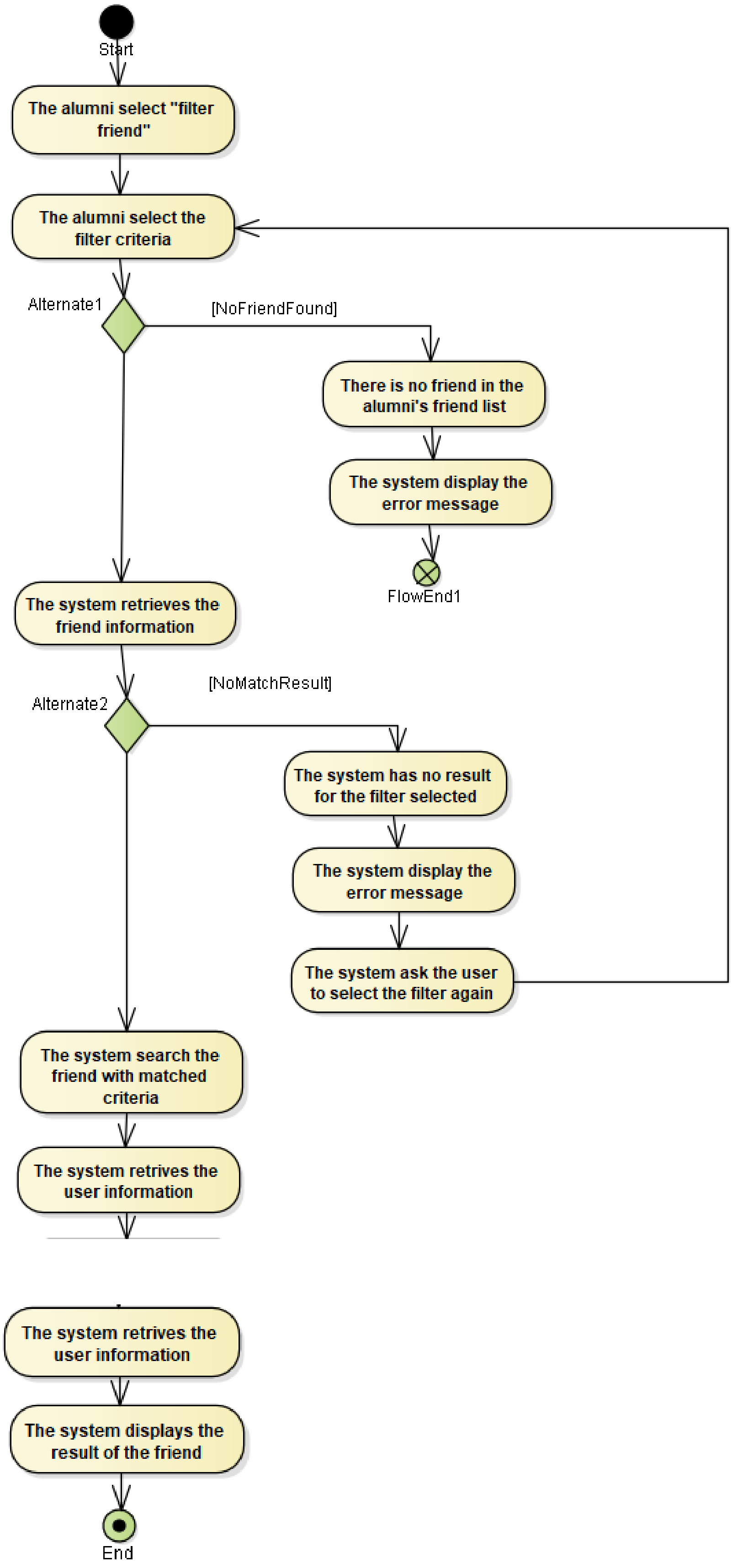

Start
The alumni select "filter friend"
The alumni select the filter criteria
Alternate1
[NoFriendFound]
There is no friend in the alumni's friend list
The system display the error message
FlowEnd1
The system retrieves the friend information
[NoMatchResult]
Alternate2
The system has no result for the filter selected
The system display the error message
The system ask the user to select the filter again
The system search the friend with matched criteria
The system retrives the user information
The system retrives the user information
The system displays the result of the friend
End

Figure 2.2.13.3: Activity Diagram for <Filter Friend>

2.2.14 UC0014: Use Case <Add Friend>

SCENARIOS
Basic Path. Basic Path 1. The alumni selects "add friend" in the system 2. The system display add friend page 3. The alumni selects the category of the friend to be added 4. The system retrieves the user info with specific category Alternate: 4a. alternate 1 5. The alumni select the friend that want to be added 6. The system adds the friend into the system 8. The system prompt a successful message to the alumni
Alternate. Alternate 1 1. If there is no user found

CONSTRAINTS
Post-condition. The alumni successfully added a new friend with specific role in the system [Approved, Weight is 0.]
Pre-condition. The user logged into the system. [Approved, Weight is 1.]

Table 2.2.14.1: Use Case Description for <Add Friend>

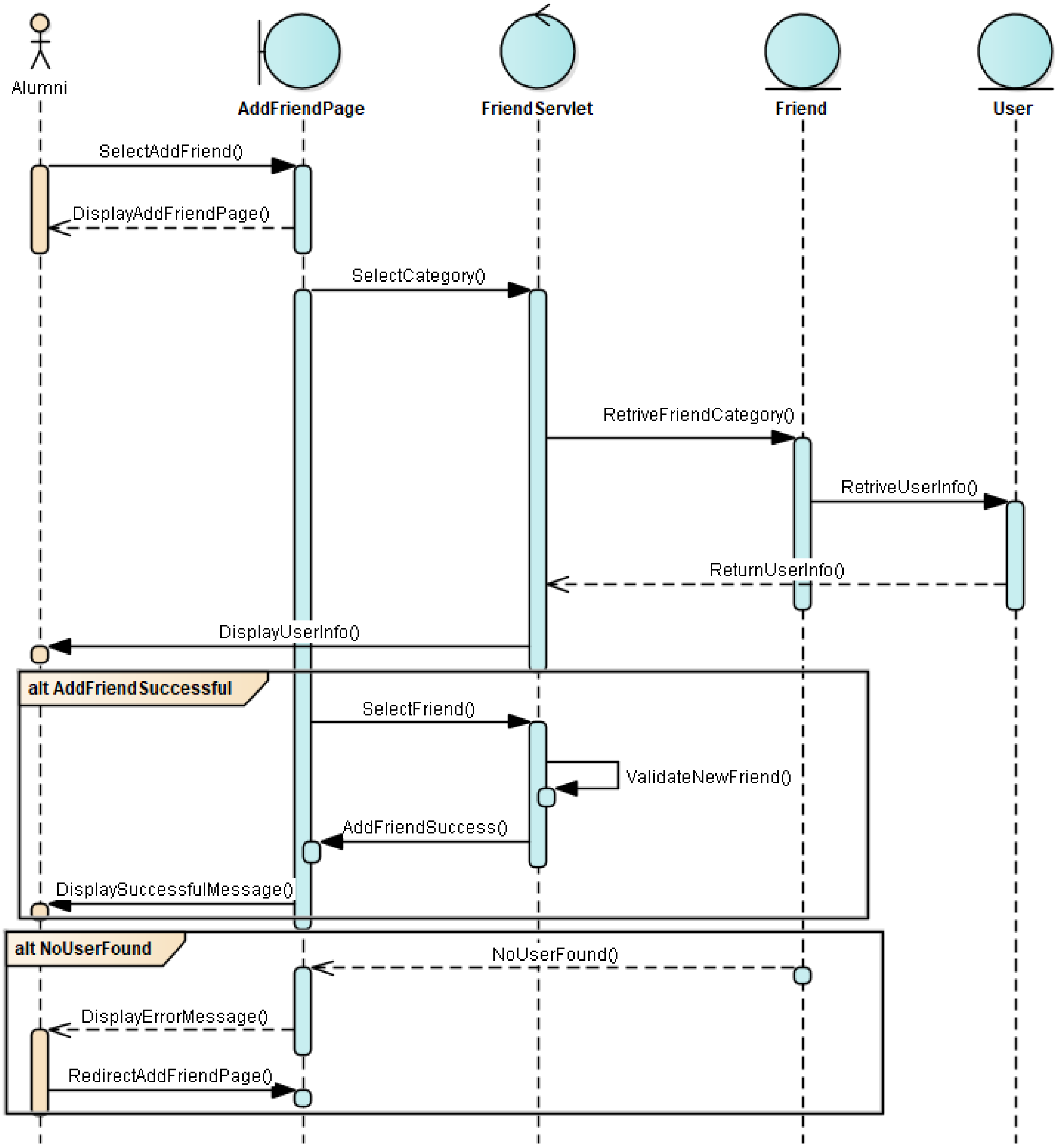


Figure 2.2.14.2: Sequence Diagram for <Add Friend>

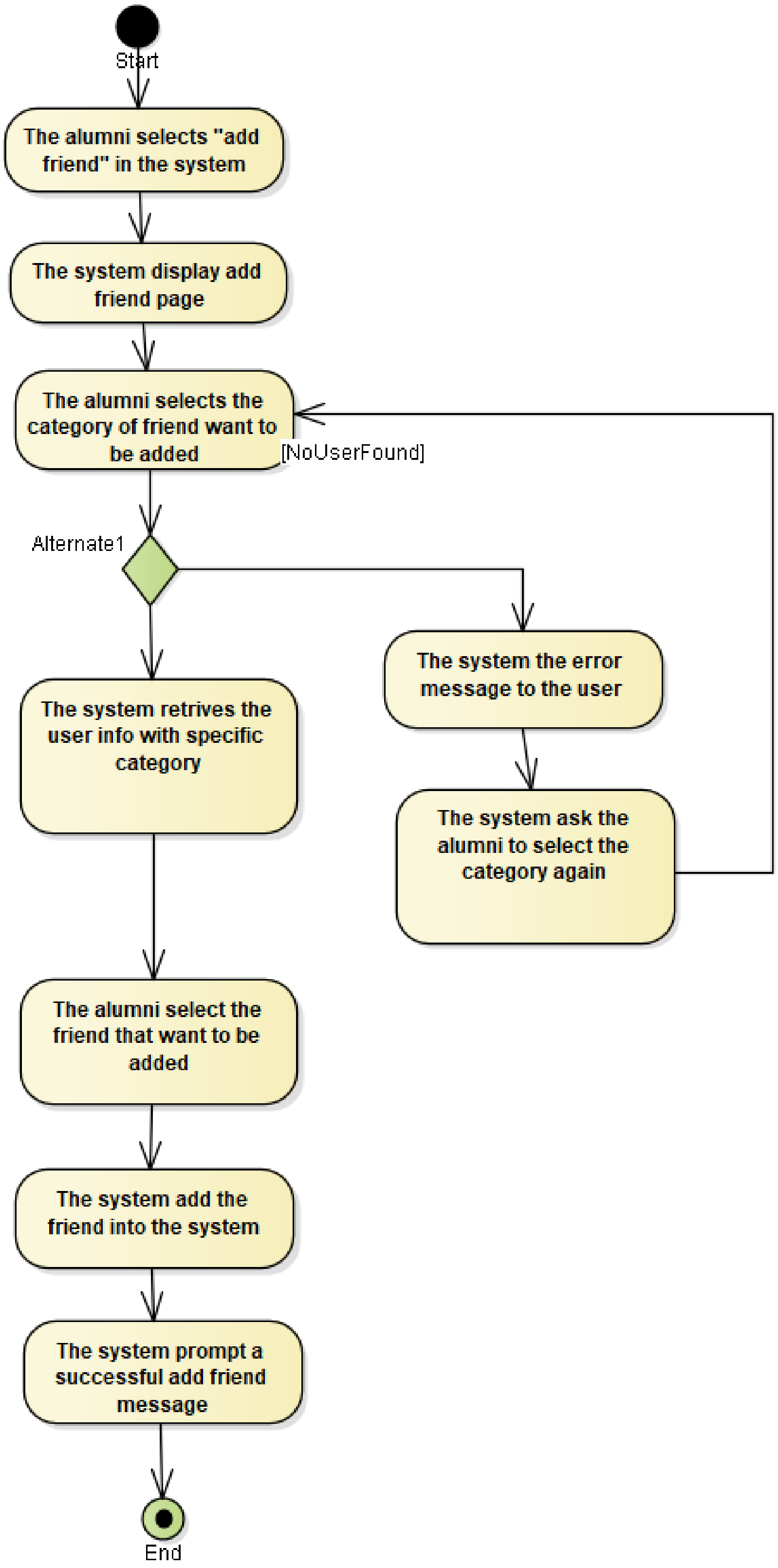

Start
The alumni selects "add friend" in the system
The system display add friend page
The alumni selects the category of friend want to be added
[NoUserFound]
Alternate1
The system the error message to the user
The system ask the alumni to select the category again
The system retrives the user info with specific category
The alumni select the friend that want to be added
The system add the friend into the system
The system prompt a successful add friend message
End

Figure 2.2.14.3: Activity Diagram for <Add Friend>

2.2.15 UC0015: Use Case <Delete Friend>

SCENARIOS
Basic Path. Basic Path **1. The alumni selects "delete friend"** **2. The alumni enters the friendID that needs to be removed** **3. The system search the ID entered** **4. The system retrieve the friend from database** **Alternate: 4a. Alternate 1** **5. The system display the selected friend info** **6. The system prompt a message for the remove confirmation** **7. The alumni select “confirm delete”** **8. The system removes the selected friend information from the database** **9. The system show a successful removed friend message**
Alternate. Alternate 1 **1. If ID not found**

CONSTRAINTS
Pre-condition. The alumni logged into the system. **[Approved, Weight is 0.]**
Post-condition. The friend is removed from the database of the system **[Approved, Weight is 1.]**

Table 2.2.15.1: Use Case Description for <Delete Friend>

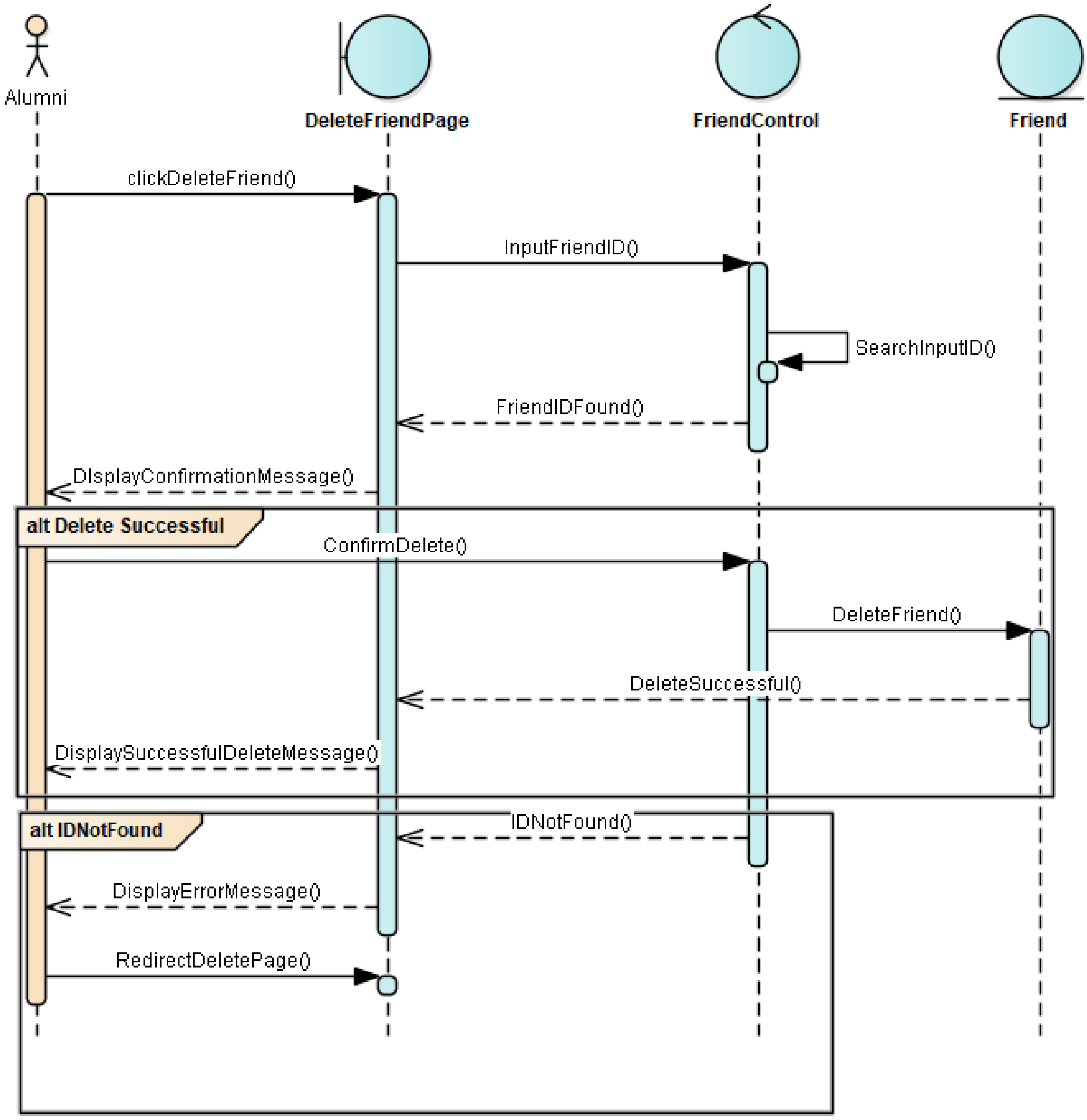


Figure 2.2.15.2: Sequence Diagram for <Delete Friend>

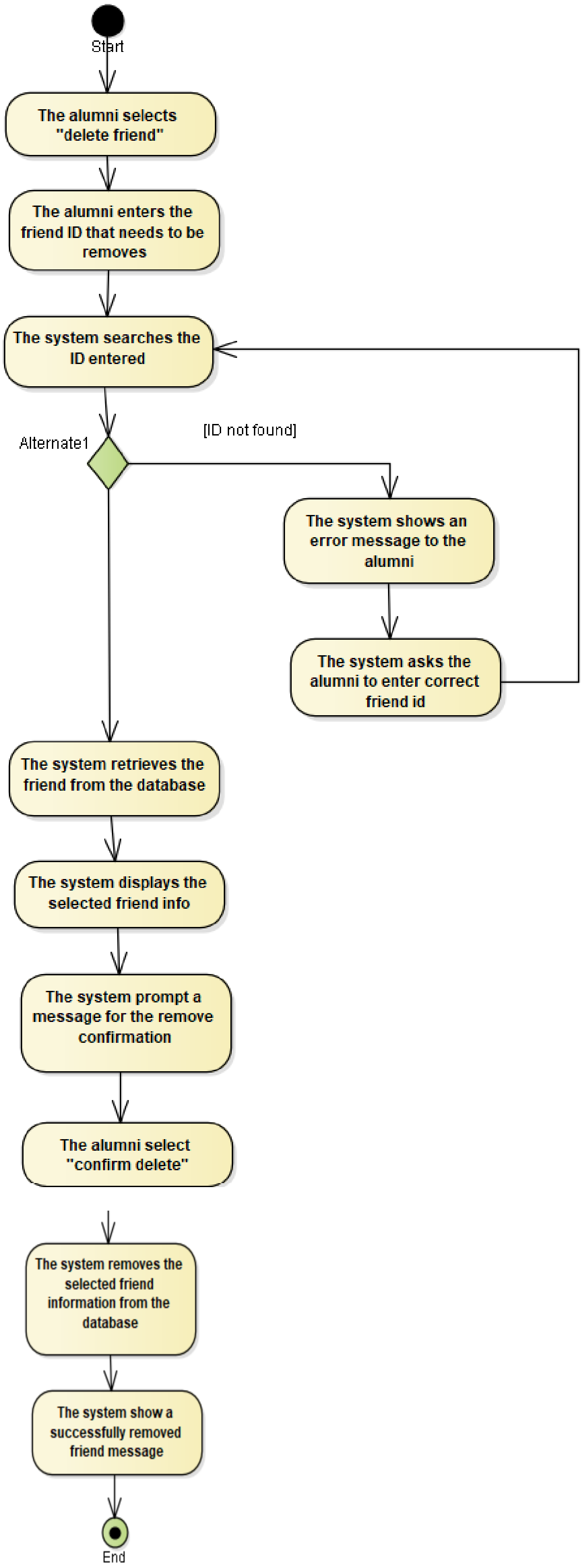

Start
The alumni selects "delete friend"
The alumni enters the friend ID that needs to be removes
The system searches the ID entered
Alternate1
[ID not found]
The system shows an error message to the alumni
The system asks the alumni to enter correct friend id
The system retrieves the friend from the database
The system displays the selected friend info
The system prompt a message for the remove confirmation
The alumni select "confirm delete"
The system removes the selected friend information from the database
The system show a successfully removed friend message
End

Figure 2.2.15.3: Activity Diagram for <Delete Friend>

2.2.16 UC0016: Use Case <Search Friend>

SCENARIOS
Basic Path. Basic Path 1. The alumni selects "search friend" 2. The alumni enters friend that wanted to search 3. The system retrieves the friend from the database Alternate: 3a. Alternate 1 4. The system displays the information of the alumni with the entered friend id 5. The system display the friend information
Alternate. Alternate 1 1. If ID not found

CONSTRAINTS
Pre-condition. The user logged into the system [Approved, Weight is 0.]
Post-condition. The friend is searched by the alumni [Approved, Weight is 1.]

Table 2.2.16.1: Use Case Description for <Search Friend>

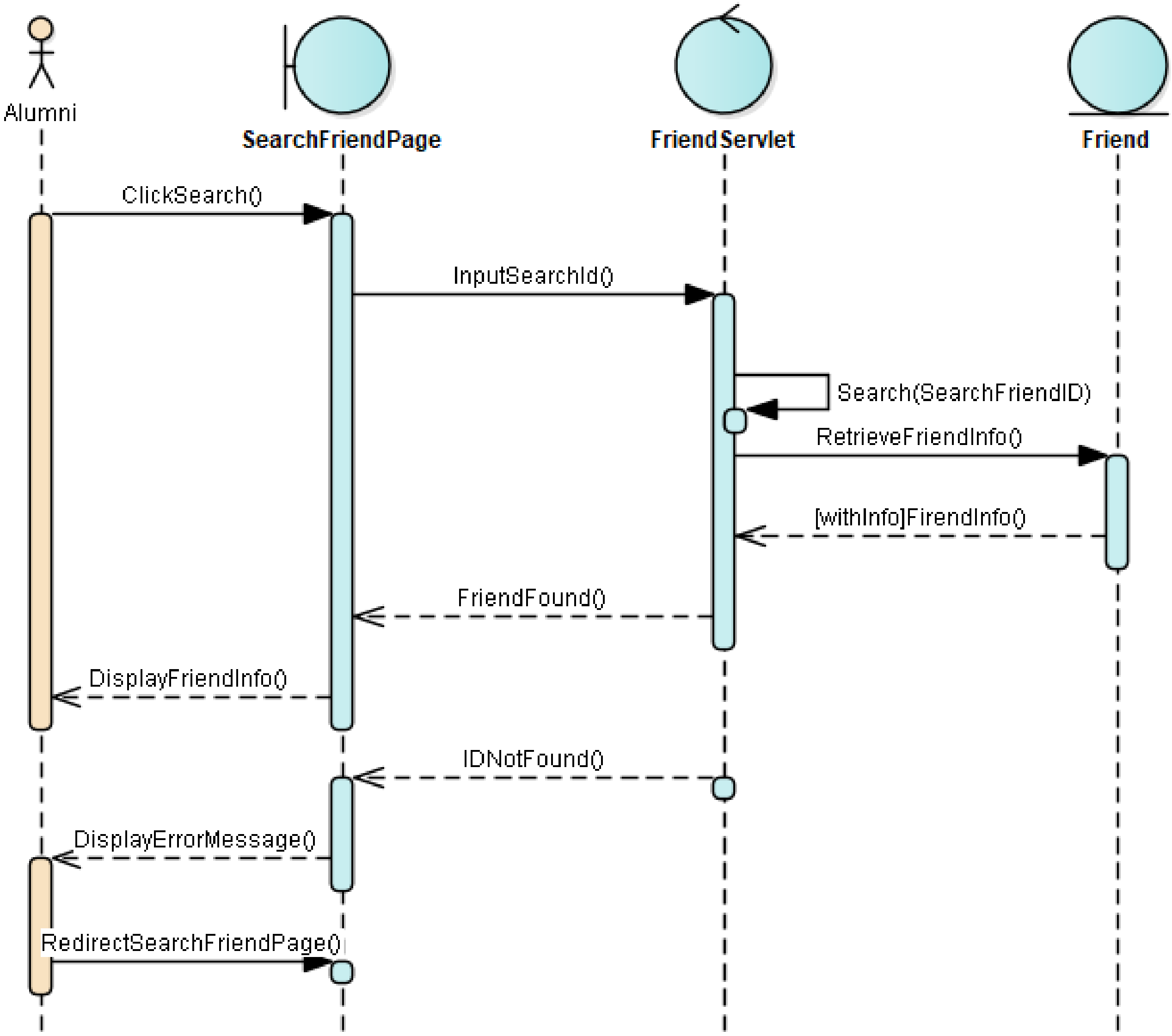


Figure 2.2.16.2: Sequence Diagram for <Search Friend>

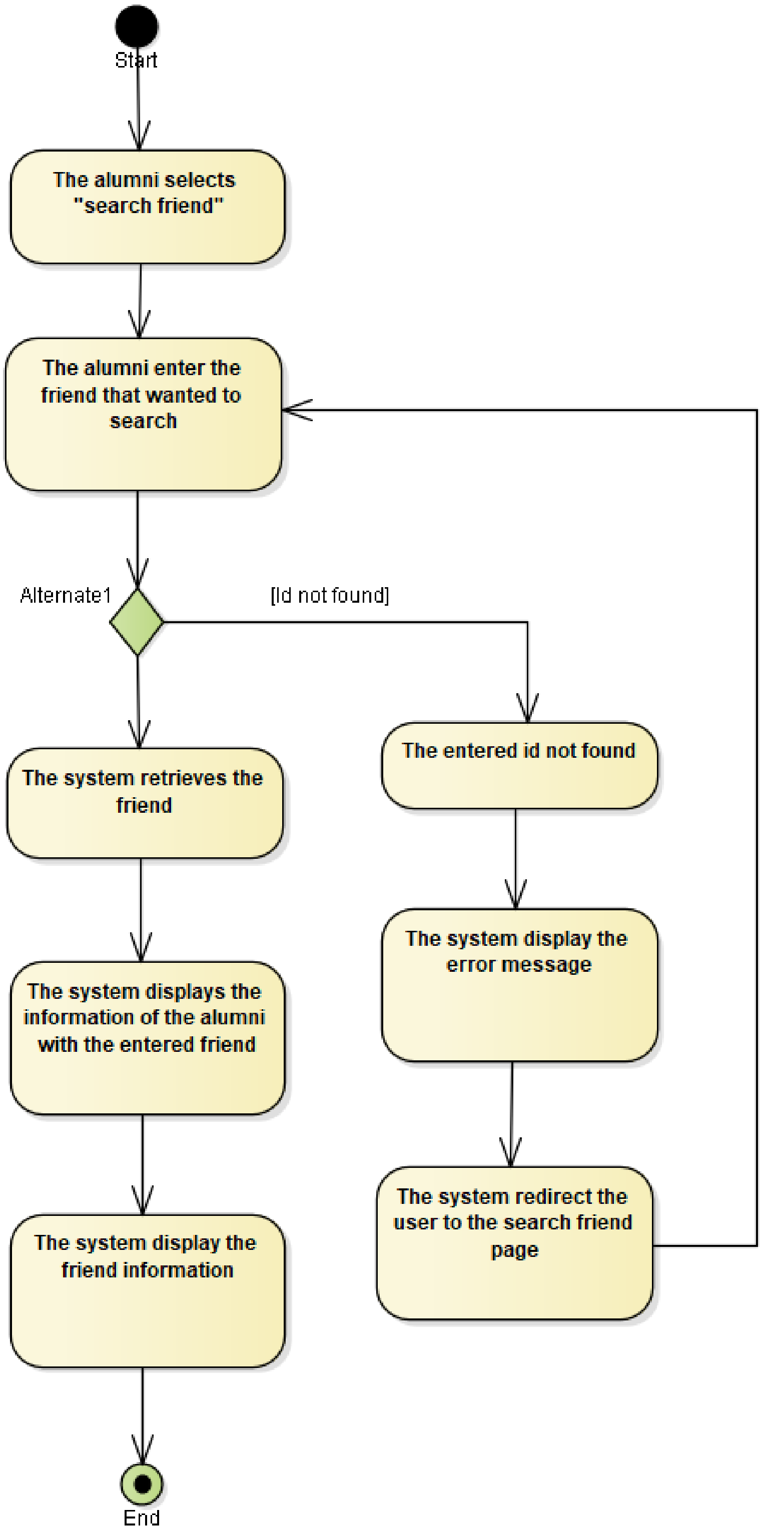

Start
The alumni selects "search friend"
The alumni enter the friend that wanted to search
Alternate1
[Id not found]
The system retrieves the friend
The entered id not found
The system displays the information of the alumni with the entered friend
The system display the error message
The system display the friend information
The system redirect the user to the search friend page
End

Figure 2.2.16.3: Activity Diagram for <Search Friend>

2.2.17 UC0017: Use Case <Update Profile>

SCENARIOS
Basic Path. Basic Path 1. The user click to view profile interface 2. The user click to update the profile 3. The system display the update profile page 4. The user modify the information in the profile 5. The system validates the information entered Exception: 5a. Wrong Information Input 6. The system updated the profile 7. The system prompt a successful message to the user
Exception. Wrong Information Input 1. The system displays the error message 2. The system ask the user to correct the error information

CONSTRAINTS
Post-condition. The user successfully update his/her profile [Approved, Weight is 0.]
Pre-condition. The user have created an account [Approved, Weight is 1.]

OUTGOING STRUCTURAL RELATIONSHIPS
Generalization from Update Profile to Manage User [Direction is 'Source -> Destination'.]

CONNECTORS

UseCaseLink Source -> Destination
From: User : Actor, Public
To: Update Profile : UseCase, Public

Figure 2.2.17.1: Use Case Description for <Update Profile>

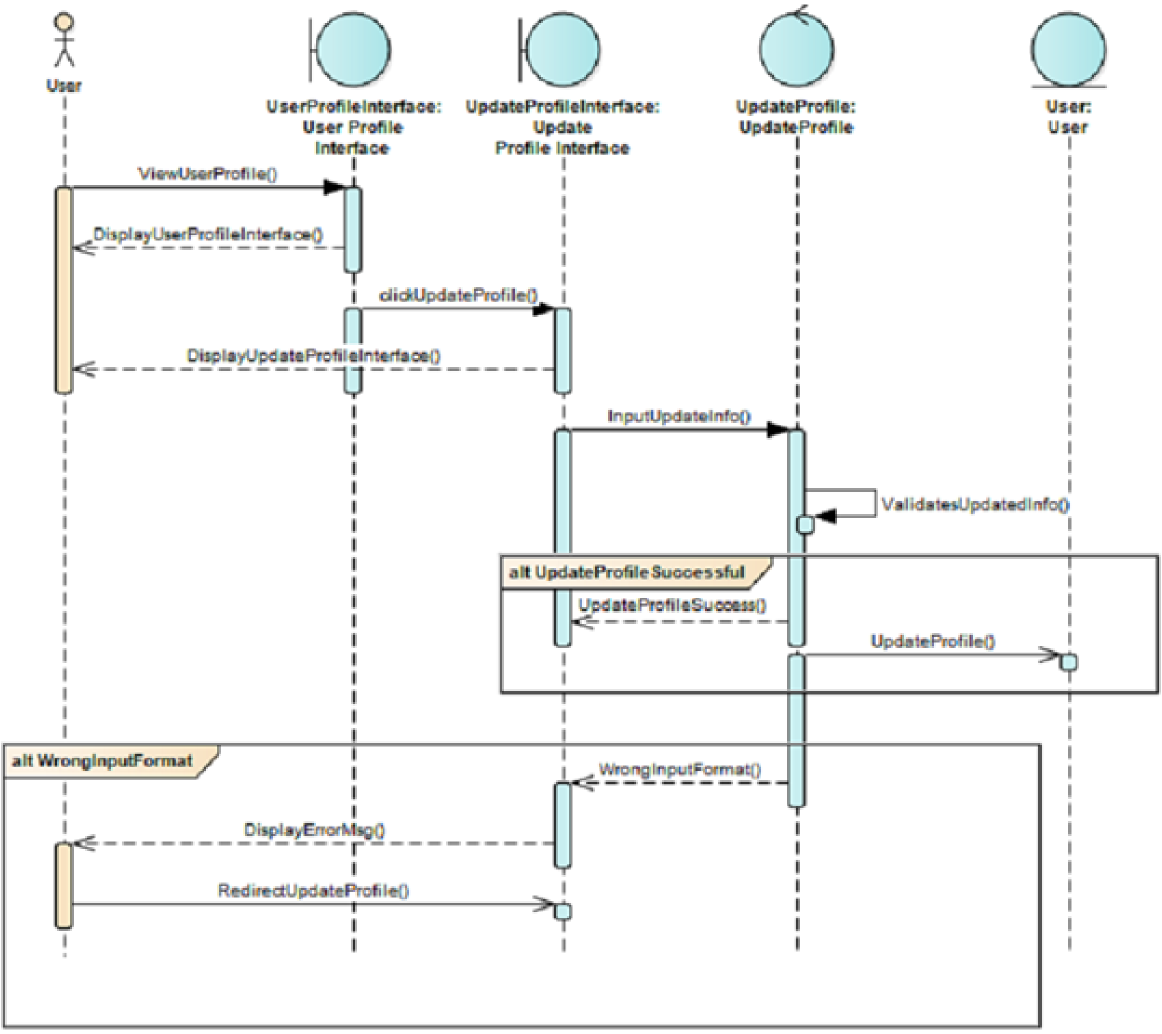

User
UserProfileInterface: User Profile Interface
UpdateProfileInterface: Update Profile Interface
UpdateProfile: UpdateProfile
User: User
ViewUserProfile()
DisplayUserProfileInterface()
clickUpdateProfile()
DisplayUpdateProfileInterface()
InputUpdateInfo()
ValidatesUpdatedInfo()
alt UpdateProfile Successful
UpdateProfileSuccess()
UpdateProfile()
alt WrongInputFormat
WrongInputFormat()
DisplayErrorMsg()
RedirectUpdateProfile()

Figure 2.2.17.2: Activity Diagram for <Update Profile>

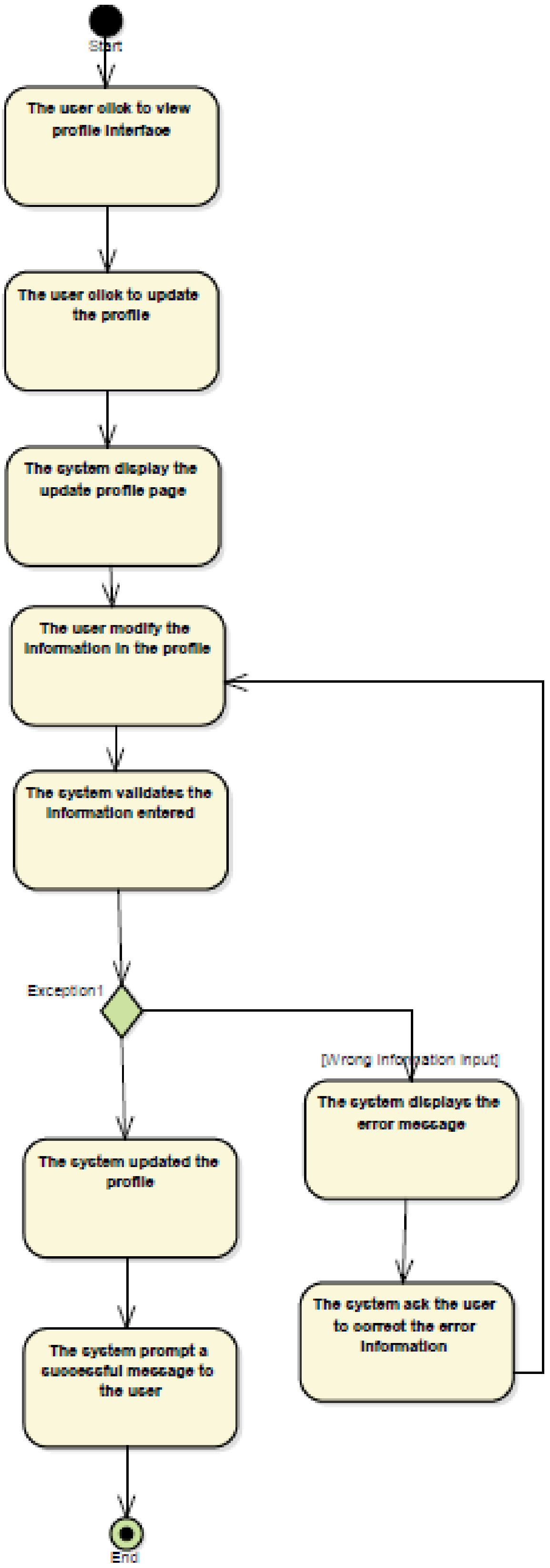

Start
The user click to view profile interface
The user click to update the profile
The system display the update profile page
The user modify the information in the profile
The system validates the information entered
Exception1
[Wrong information input]
The system displays the error message
The system ask the user to correct the error information
The system updated the profile
The system prompt a successful message to the user
End

Figure 2.2.17.3: Activity Diagram for <Update Profile>

2.2.18 UC0018: Use Case <View Status>

SCENARIOS
1. The user selects ‘Status’. **2. The system displays a status information page.** **3. The user select a status to view** **4. The system displays available status** **Alternate: 4a. No Information** **5. The use case ends.**
Alternate **If there is no status information to display, the system display “No Information”**
CONSTRAINTS
Pre-condition. The user logged into account [Approved, Weight is 0.]
Post-condition: The user accessed the main page [Implemented, weight 1]

OUTGOING STRUCTURAL RELATIONSHIP
Generalization from View Status to Manage mobile **[Direction is 'Source -> Destination'.**

Figure 2.2.18.1: Use Case Description for <View Status>

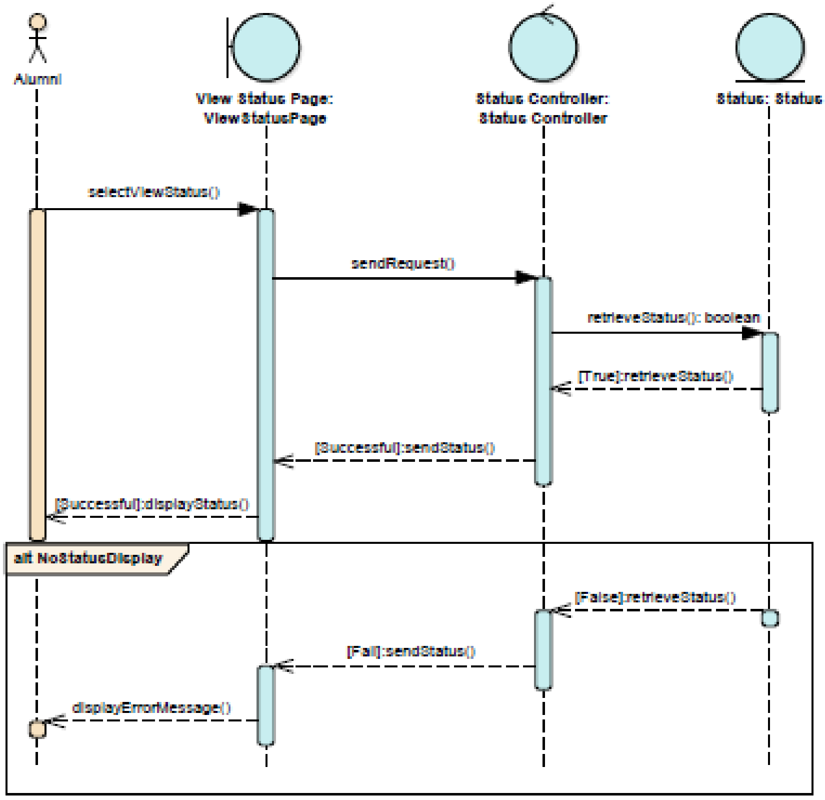


Figure 2.2.18.2: Sequence Diagram for <View Status>

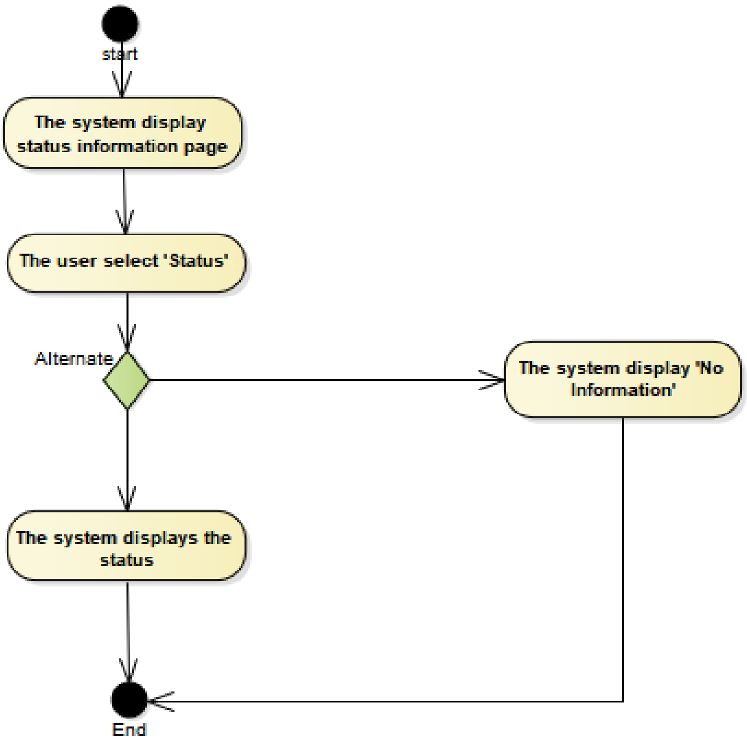

start
The system display status information page
The user select 'Status'
Alternate
The system display 'No Information'
The system displays the status
End

Figure 2.2.18.3: Activity Diagram for <View Status>

2.2.19 UC0019: Use Case <Upload Status>

SCENARIOS
Basic Path. Update Status 1. The use case begins when the user selects ‘Upload Status’. 2. The user insert status 3. The user uploads the status 4. The user confirms the upload 4a. Alternate. The user cancels the update 5. The system saves and upload the status 6. The use case ends
Alternate If the user cancels the update, the system will not change anything
CONSTRAINTS
Pre-condition. The user logged into account and posted a status before [Approved, Weight is 0.]
Post-condition: The user accessed the Status [Approved, weight 1]

OUTGOING STRUCTURAL RELATIONSHIP
Generalization from Upload Status to Manage mobile [Direction is 'Source -> Destination'.

Figure 2.2.19.1: Use Case Description for <Upload Status>

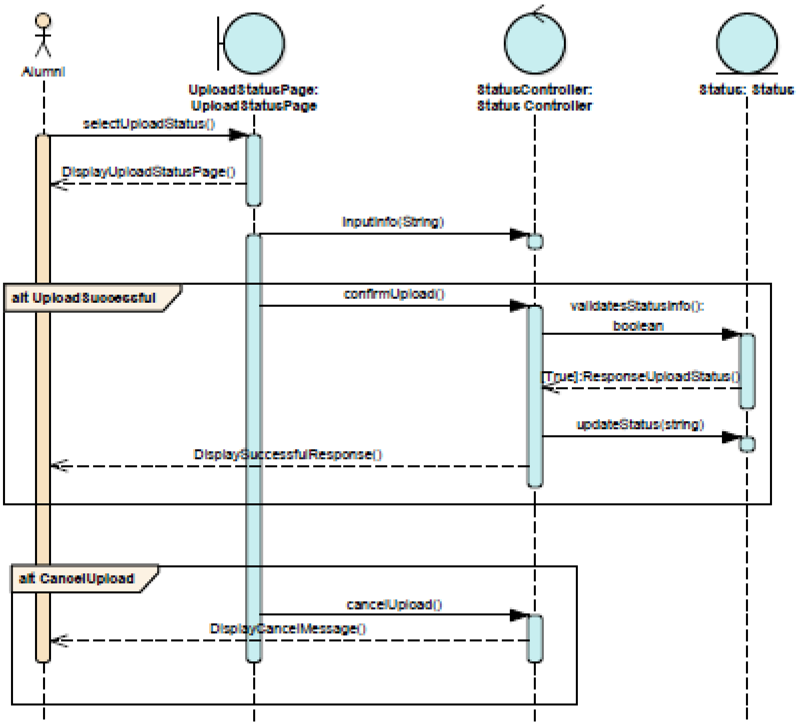


Figure 2.2.19.2: Sequence Diagram for <Upload Status>

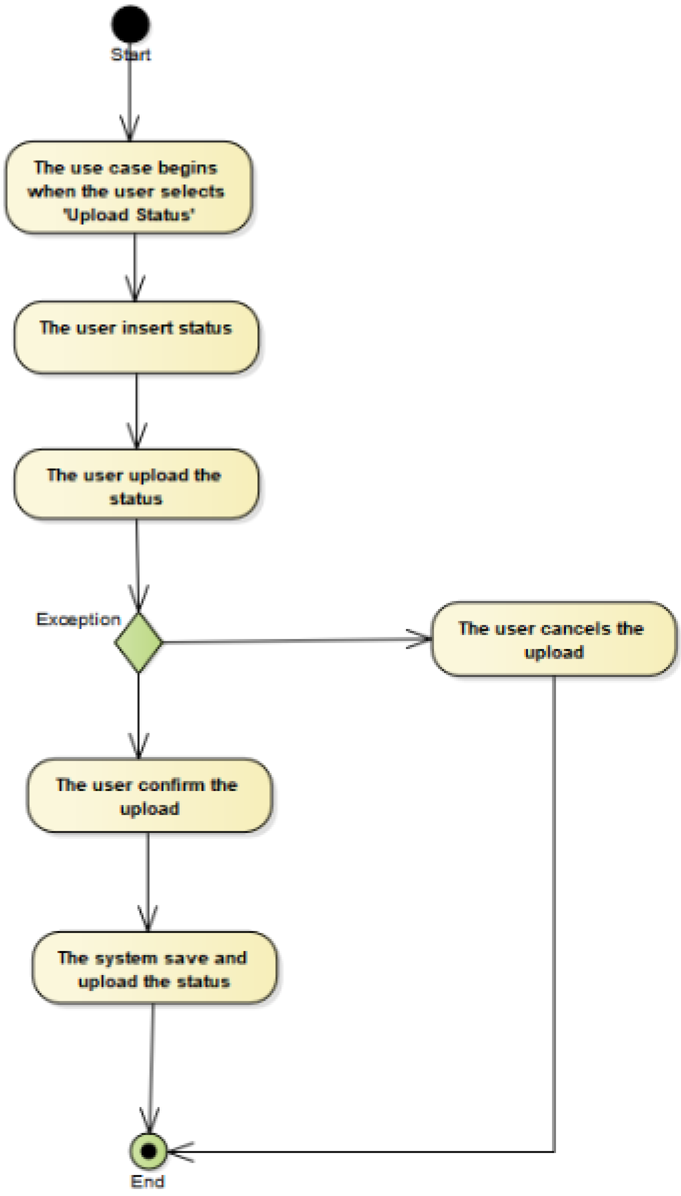

Start
The use case begins when the user selects 'Upload Status'
The user insert status
The user upload the status
Exception
The user cancels the upload
The user confirm the upload
The system save and upload the status
End

Figure 2.2.19.3: Activity Diagram for <Upload Status>

2.2.20 UC0020: Use Case <Edit Status>

SCENARIOS
Basic Path. Basic Path 1. The alumni click to view status 2. The alumni click "edit status" button 3. The system display edit status interface 4. The alumni edit the status Alternate: 4a. Cancel Edit 5. The alumni click "done edit" button 6. The system update the status 7. The system prompt a successful edit message
Alternate. Cancel Edit 1. The alumni click "cancel edit" button 2. The system redirect the user to status interface

CONSTRAINTS
Pre-condition. The user logged into account and posted a status before [Approved, Weight is 0.]
Post-condition. The user edit the status successfully [Approved, Weight is 1.]

OUTGOING STRUCTURAL RELATIONSHIPS
Generalization from Edit status to Manage alumni [Direction is 'Source -> Destination'.]

Figure 2.2.20.1: Use Case Description for <Edit Status>

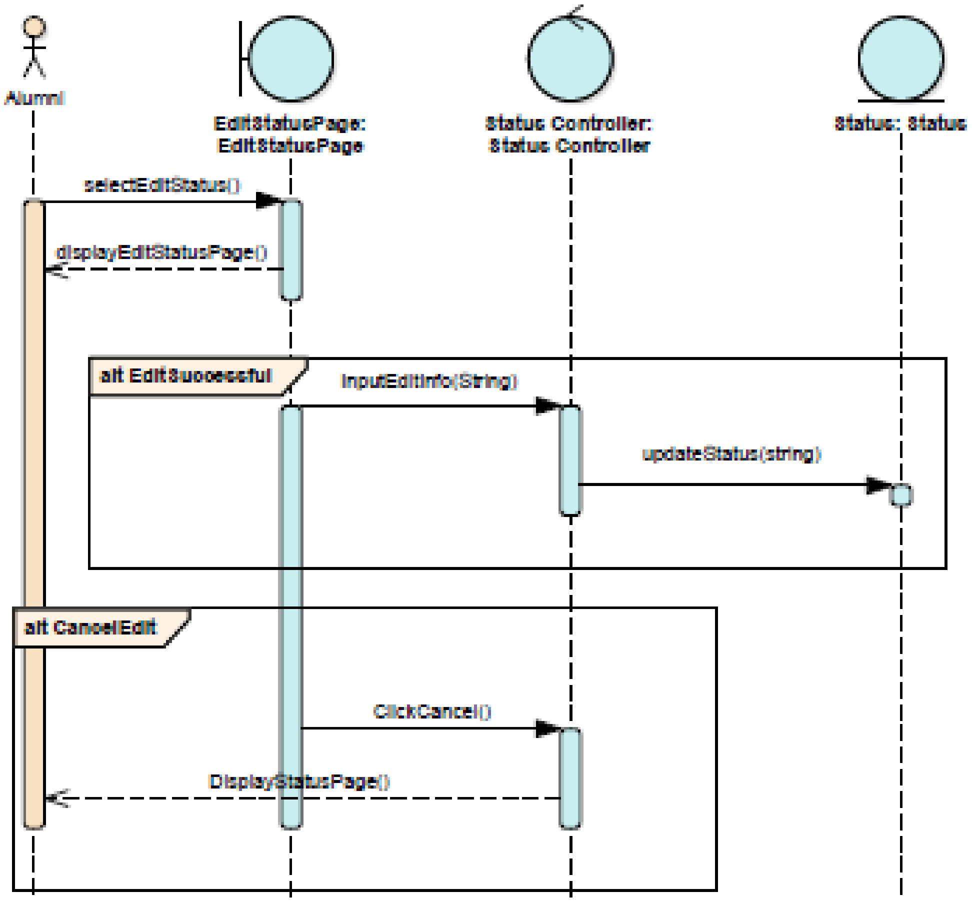


Figure 2.20.1.2: Sequence Diagram for <Edit Status>

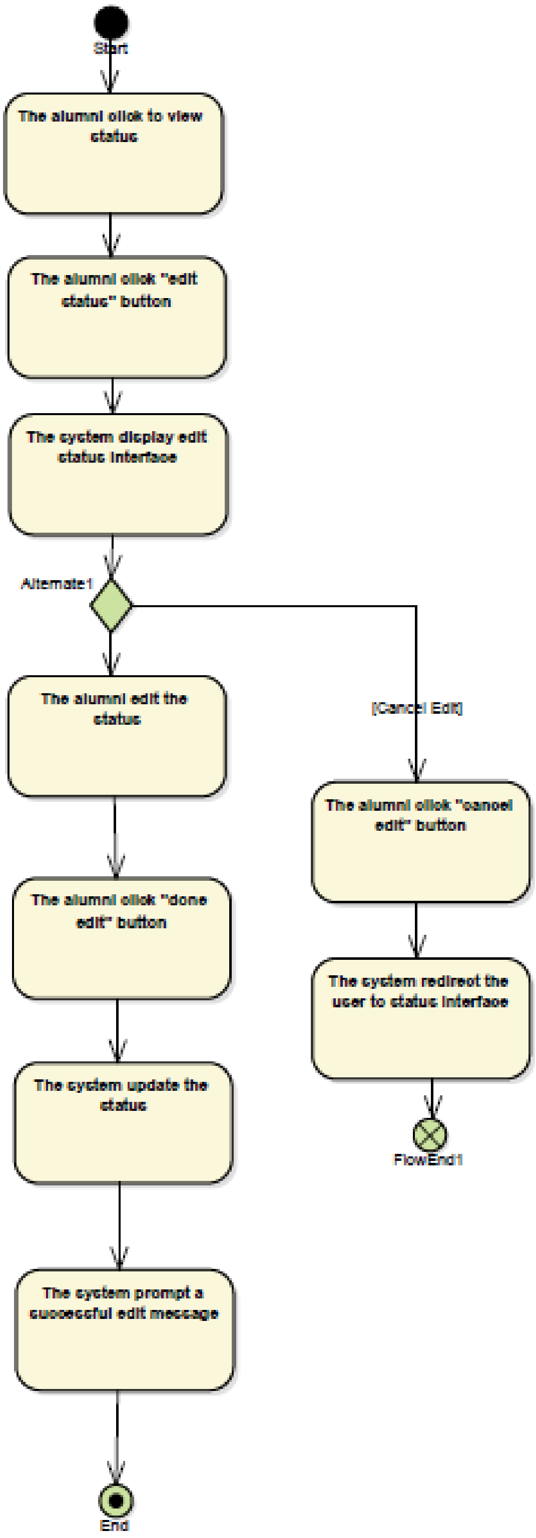

Start
The alumni click to view status
The alumni click "edit status" button
The system display edit status interface
Alternate1
[Cancel Edit]
The alumni edit the status
The alumni click "cancel edit" button
The alumni click "done edit" button
The system redirect the user to status interface
FlowEnd1
The system update the status
The system prompt a successful edit message
End

Figure 2.2.20.3: Activity Diagram for <Edit Status>

2.2.21 UC0021: Use Case <Delete Status>

SCENARIOS
Basic Path. Basic Path 1. The alumni click to view status 2. The alumni click "delete status" button 3. The system prompt a confirmation message Alternate: 3a. Cancel Delete 4. The alumni click "confirm delete" 5. The system delete status from the status database 6. The system display a successful delete message
Alternate. Cancel Delete 1. The alumni click "cancel delete" button 2. The system redirect the user to status interface

CONSTRAINTS
Pre-condition. The alumni posted a status before [Approved, Weight is 0.]
Post-condition. The alumni deleted a status successfully [Approved, Weight is 1.]

OUTGOING STRUCTURAL RELATIONSHIPS
Generalization from Delete status to Manage alumni [Direction is 'Source -> Destination'.]

Figure 2.2.21.1: Use Case Description for <Delete Status>

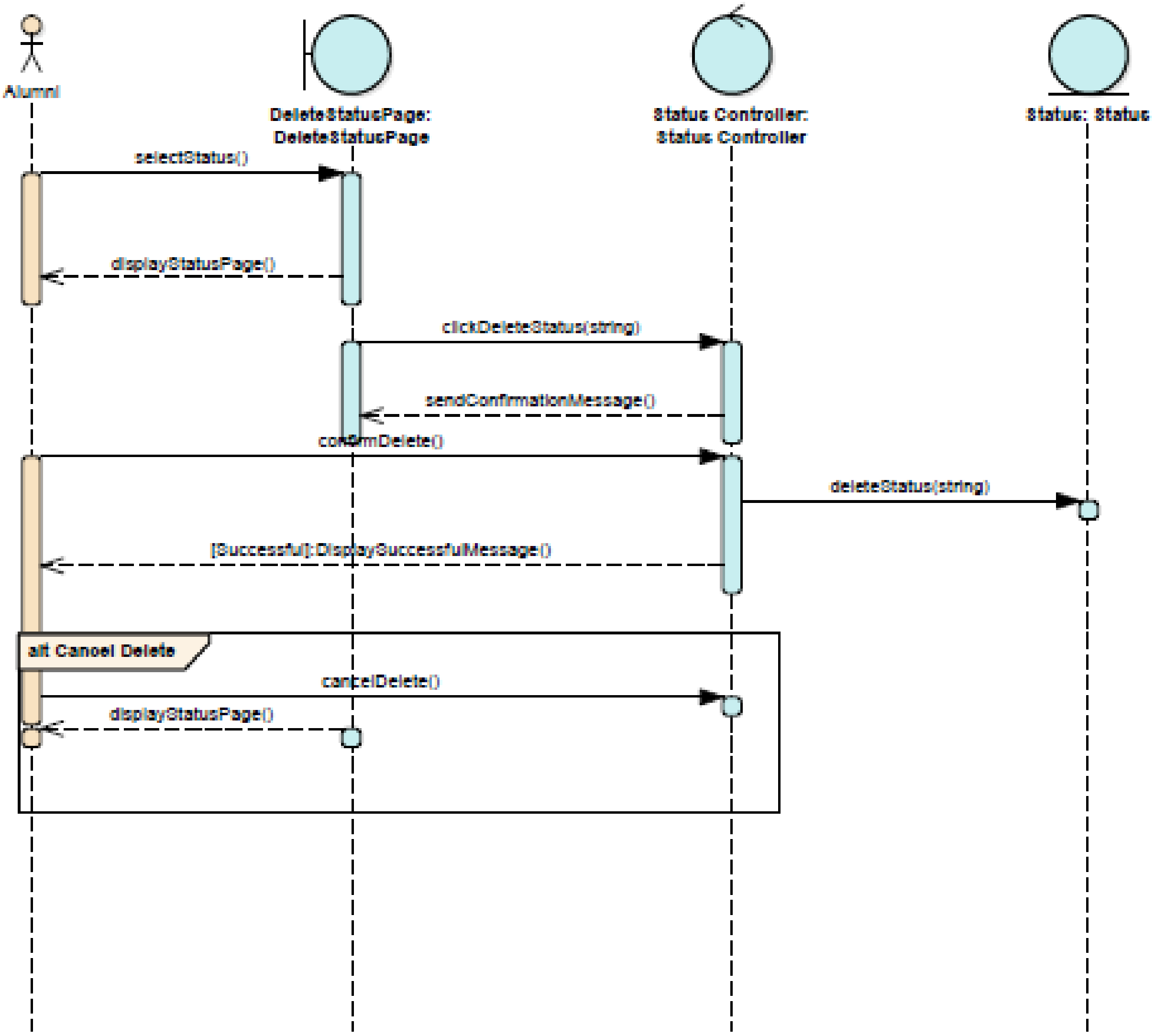

Alumni
DeleteStatusPage: DeleteStatusPage
Status Controller: Status Controller
Status: Status
selectStatus()
displayStatusPage()
clickDeleteStatus(string)
sendConfirmationMessage()
confirmDelete()
deleteStatus(string)
[Successful]: DisplaySuccessfulMessage()
alt Cancel Delete
cancelDelete()
displayStatusPage()

Figure 2.2.21.2: Sequence Diagram for <Delete Status>

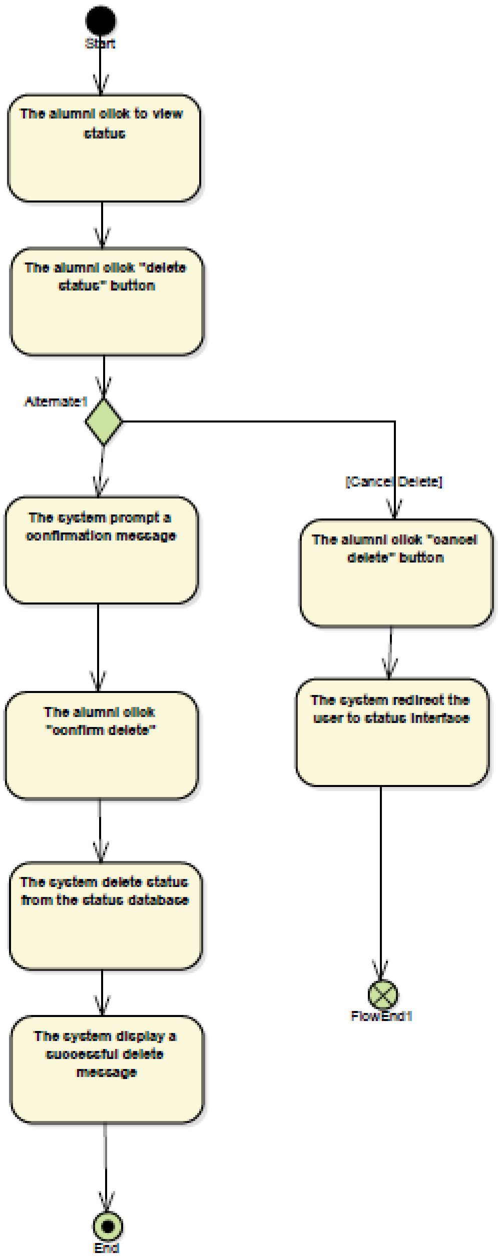

Start
The alumni click to view status
The alumni click "delete status" button
Alternate1
[Cancel Delete]
The system prompt a confirmation message
The alumni click "cancel delete" button
The alumni click "confirm delete"
The system redirect the user to status interface
The system delete status from the status database
FlowEnd1
The system display a successful delete message
End

Figure 2.2.21.3: Activity Diagram for <Delete Status>

2.3 Performance Requirements

The system should be capable of supporting at least 10000 users at the same time. For the response time, the system should display the pages within milliseconds while if there is any delay for the pages, the delay should be less than 8 seconds. The system should be able to load 50 pages in at most 1 second. For the requirements, the system should be able to process 10000 requests per second in the peak load and it should be able to accept every request submitted. Other transitions besides payment transactions should be able to be performed for the whole day. On the other hand, the payment transaction for the sponsorship and donations should be able to be performed from 12.15 a.m. until 11.59 p.m. everyday.

2.4 Design Constraints

- This system must be equipped by all devices that can access this website, such as using a laptop or mobile phone.
- The way this system will be present needs to follow basic knowledge of other websites so that users can easily access and understand.
- The information of all the users must be stored in a central database that is accessible by the Online Alumni System.
- The users can access the Online Alumni System from any computer that has Internet browsing capabilities and an Internet connection.
- The users must have their correct usernames and passwords to enter into the Online Alumni System.
- The web portal will be constrained by the capacity of the database. Since the database may be forced to queue incoming requests and therefore increase the time it takes to fetch data.

- The Internet connection is also a constraint for the system. Since the system needs to fetch data from the database over the Internet, it is important that there is an Internet connection for the system to function.
- The response time to load for the request should not be longer than 2 seconds.

2.5 Software System Attributes

- This system is available on any web browser.
- Users should be logged in by the username and password uniquely that they registered.
- User can update profile anytime at the profile page
- This system is available for user to join and view the event by user whereas and able for staff to edit
- New alumni users should create an account for the first-time user.

2.6 Other Requirements

- Security

  The system should protect the user password against unauthorized access.

  The system should not allow alumni to update and delete other user’s information.

- Accuracy

  The system should display accurate graph information to the user.

  The system should display accurate alumni information to the user.

- Mobility

  The user should be able to access the system with both PC and mobile.

- Usability

  The system shall display good interfaces that achieve Shneiderman's Eight Golden Rules.

  The system shall be user-friendly and easy to use for alumni and staff.

- Availability

  The staff shall be able to manage the event, charity and news at any time during the day