Vietnam National University Ho Chi Minh City

Ho Chi Minh City of Technology

Computer Science and Engineering Faculty



SOFTWARE ENGINEERING

ASSIGNMENT 4

Professor: Bui Hoai Thang

Teaching Assistant: Bui Cong Tuan

Student: Luong Thanh Nhan – 1820047 – Class B01

Group 4

11/2019

Contents

[Part 1: ASSIGNMENT 1 8](#_Toc25616735)

[1. Introduction (Group work) 8](#_Toc25616736)

[1.1. Purposes 8](#_Toc25616737)

[1.2. Definitions 8](#_Toc25616738)

[1.3. Overview 8](#_Toc25616739)

[2. Overall Description (Group work) 8](#_Toc25616740)

[2.1. System Use Case Diagram 8](#_Toc25616741)

[2.2. System Context Diagram 11](#_Toc25616742)

[3. Functional Requirement Specifications (Group work) 11](#_Toc25616743)

[3.1. Room Information and Schedule 11](#_Toc25616744)

[3.1.1. Search room for information 11](#_Toc25616745)

[3.1.2. Search room for booking 11](#_Toc25616746)

[3.1.3. Book room 11](#_Toc25616747)

[3.1.4. Cancel room 12](#_Toc25616748)

[3.1.5. Modify room 12](#_Toc25616749)

[3.1.6. Review booking 12](#_Toc25616750)

[3.1.7. Confirm by email 12](#_Toc25616751)

[3.1.8. Remind by email 12](#_Toc25616752)

[3.1.9. Campus map 12](#_Toc25616753)

[3.1.10. Recommend room 12](#_Toc25616754)

[3.1.11. Search Lecturer 12](#_Toc25616755)

[3.1.12. Borrow Laboratory 12](#_Toc25616756)

[3.2. Electrical Automation System 13](#_Toc25616757)

[3.2.1. Turn on automatically 13](#_Toc25616758)

[3.2.2. Turn off automatically 13](#_Toc25616759)

[3.2.3. Alert not functional 13](#_Toc25616760)

[3.2.4. Alarm automatically 13](#_Toc25616761)

[3.3. Security Staff 13](#_Toc25616762)

[3.3.1. Search for electrical devices 13](#_Toc25616763)

[3.3.2. Search for active room 13](#_Toc25616764)

[3.3.3. Control electrical devices remotely 13](#_Toc25616765)

[3.3.4. Control electrical devices in-person 13](#_Toc25616766)

[3.3.5. Monitor camera 13](#_Toc25616767)

[3.3.6. Enter the Campus 14](#_Toc25616768)

[3.4. Sensor and Camera System 14](#_Toc25616769)

[3.4.1. Check number of people 14](#_Toc25616770)

[3.4.2. Automatically control the light 14](#_Toc25616771)

[3.4.3. Alert detecting human 14](#_Toc25616772)

[3.4.4. Detect fire 14](#_Toc25616773)

[3.5. SOS Alert 14](#_Toc25616774)

[3.6. Statistical and Reporting System 14](#_Toc25616775)

[3.6.1. View electrical usage 14](#_Toc25616776)

[3.6.2. View room usage 14](#_Toc25616777)

[3.6.3. Automatically send report 14](#_Toc25616778)

[3.6.4. User feedback 15](#_Toc25616779)

[4. Non-Functional Requirements Specification (Group work and Individual work) 15](#_Toc25616780)

[4.1. Group work - Non-Functional Requirements Specification 15](#_Toc25616781)

[4.1.1. Efficiency requirements 15](#_Toc25616782)

[4.1.2. Regulatory 16](#_Toc25616783)

[4.1.3. Usability 16](#_Toc25616784)

[4.1.4. Reliability 16](#_Toc25616785)

[4.1.5. Confidentiality 16](#_Toc25616786)

[4.1.6. Survivability 17](#_Toc25616787)

[4.2. Individual - Other Non-Functional Requirements 17](#_Toc25616788)

[5. Use-case detail/scenario for use-case and specific non-function requirement (Individual work) 18](#_Toc25616789)

[5.1 Check Number Of People 18](#_Toc25616790)

[5.1.1. Context Model: 18](#_Toc25616791)

[5.1.2. Use case diagram: 18](#_Toc25616792)

[5.1.3. Use case scenario tabular: 19](#_Toc25616793)

[5.1.4. Non-function requirement 20](#_Toc25616794)

[5.2 Send SOS 20](#_Toc25616795)

[5.2.1. Context Model: 20](#_Toc25616796)

[5.2.2. Use-case Diagram: 21](#_Toc25616797)

[5.2.3. Use case scenario tabular: 21](#_Toc25616798)

[5.2.4. Non-functional requirement 22](#_Toc25616799)

[5.3. Borrow Equipment 22](#_Toc25616800)

[5.3.1. Context Model: 22](#_Toc25616801)

[5.3.2. Use-case Diagram: 22](#_Toc25616802)

[5.3.3. Use case scenario tabular: 23](#_Toc25616803)

[5.3.4 Non-functional requirements 23](#_Toc25616804)

[5.4. Search for Lecturer 24](#_Toc25616805)

[5.4.1. Context Model: 24](#_Toc25616806)

[5.4.2. Use-case Diagram: 24](#_Toc25616807)

[5.4.3. Use case scenario tabular: 24](#_Toc25616808)

[5.4.4 Non-functional requirement 25](#_Toc25616809)

[5.5. Get Identification 26](#_Toc25616810)

[5.5.1. Context Model: 26](#_Toc25616811)

[5.5.2. Use-case Diagram: 26](#_Toc25616812)

[5.5.3. Use case scenario tabular: 26](#_Toc25616813)

[5.5.4 Non-functional requirement 27](#_Toc25616814)

[5.6. Verify Identification 27](#_Toc25616815)

[5.6.1. Context Model: 27](#_Toc25616816)

[5.6.2. Use-case Diagram: 27](#_Toc25616817)

[5.6.3. Use case scenario tabular: 28](#_Toc25616818)

[5.6.4 Non-functional requirement 28](#_Toc25616819)

[6. Other non-interactive functional requirement (Individual work) 29](#_Toc25616820)

[6.1. Send notifications about quotes 29](#_Toc25616821)

[6.2. Remind exam schedule. 29](#_Toc25616822)

[6.3. Check students’ attendance 29](#_Toc25616823)

[7. Conclusion (Individual work) 29](#_Toc25616824)

[Part 2: ASSIGNMENT 2 30](#_Toc25616825)

[1. Sequence Diagram 30](#_Toc25616826)

[1.1 Check Number of People 30](#_Toc25616827)

[1.2 Send SOS 32](#_Toc25616828)

[1.3 Borrow Equipment 34](#_Toc25616829)

[1.4 Search for Lecturer 36](#_Toc25616830)

[1.5. Get Identification 38](#_Toc25616831)

[1.6 Verify Identification 40](#_Toc25616832)

[2. State-chart Diagram 42](#_Toc25616833)

[2.1 Lab equipment state-chart diagram 42](#_Toc25616834)

[3. Activity Diagram 44](#_Toc25616835)

[3.1 Verify identification activity 44](#_Toc25616836)

[3.2 Borrow lab equipment activity 45](#_Toc25616837)

[3.3 Check number of people activity 46](#_Toc25616838)

[3.4 Send SOS activity 47](#_Toc25616839)

[4. Conclusion 48](#_Toc25616840)

[Part 3: ASSIGNMENT 3 49](#_Toc25616841)

[1. Deployment view (Group work) 49](#_Toc25616842)

[2. Development view 50](#_Toc25616843)

[2.1 Check Number of People 50](#_Toc25616844)

[2. 2 Check Authorization 50](#_Toc25616845)

[2. 3 Send SOS 51](#_Toc25616846)

[3. Conclusion 52](#_Toc25616847)

[Part 4: ASSIGNMENT 4 53](#_Toc25616848)

[1. Group work 53](#_Toc25616849)

[1.1. Software Structure – MVC Paradigm 53](#_Toc25616850)

[1.2. System Class Diagram 54](#_Toc25616851)

[2. Individual work 56](#_Toc25616852)

[2.1. Borrow Lab Equipment 56](#_Toc25616853)

[2.1.1. Module Interface 56](#_Toc25616854)

[2.1.2. Class Diagram 56](#_Toc25616855)

[2.1.3. Sequence Diagram 59](#_Toc25616856)

[2.1.4. State-chart Diagram: 60](#_Toc25616857)

[2.1.5. Design Pattern: 61](#_Toc25616858)

[2.2. Send SOS 61](#_Toc25616859)

[2.2.1. Module Interface 61](#_Toc25616860)

[2.2.2. Class Diagram 61](#_Toc25616861)

[2.2.3. Sequence Diagram 64](#_Toc25616862)

[2.2.4. Design Pattern 65](#_Toc25616863)

[2.3. Check Number of People 65](#_Toc25616864)

[2.3.1. Module Interface 65](#_Toc25616865)

[2.3.2. Class Diagram 65](#_Toc25616866)

[2.3.3. Sequence Diagram 68](#_Toc25616867)

[2.3.4. Design Pattern 69](#_Toc25616868)

[2.4. Working demonstrations 69](#_Toc25616869)

[2.4.1. Borrow Equipment 69](#_Toc25616870)

[2.4.2. Send SOS 71](#_Toc25616871)

[2.4.3. Check Number of People 72](#_Toc25616872)

[3. Conclusion 73](#_Toc25616873)

[4. Group list 73](#_Toc25616874)

# Part 1: ASSIGNMENT 1

# 1. Introduction (Group work)

## 1.1. Purposes

The purpose of this report is specifying a Smart Campus System (SCAMS) in detail including system design, features, methods and operation of the system and how the system interacts with variety range of users such as lecturers, students, security staffs,guesses. The system is expected to make the university smart.

## 1.2. Definitions

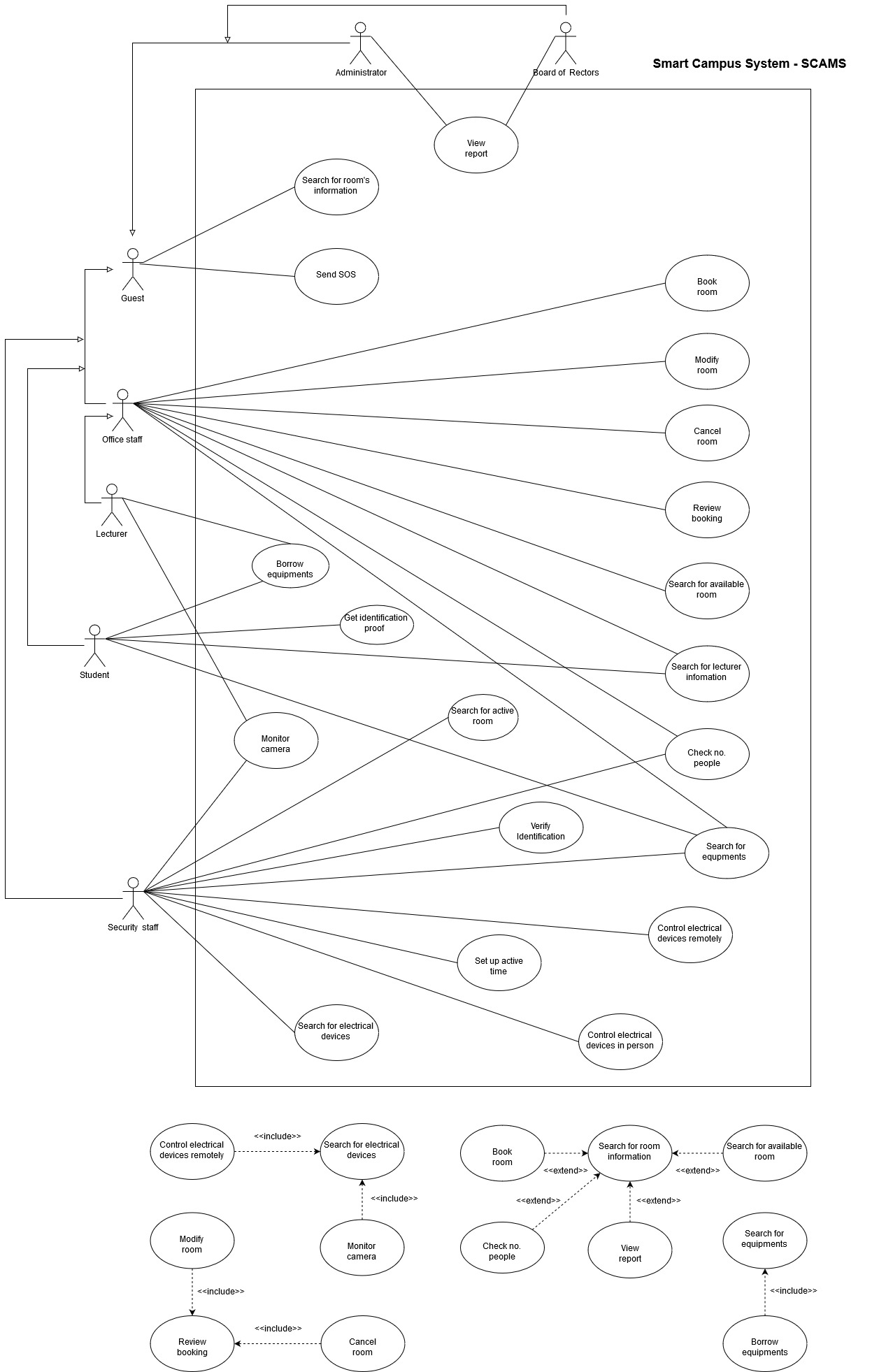
SCAMS: Smart Campus System

## 1.3. Overview

* Section 1: Group work - Introduction to the SCAMS
* Section 2: Group work - Overall description of the SCAMS
* Section 3: Group work and individual work - Functional requirements
* Section 4: Group work and individual work - Non-Functional requirements
* Section 5: Individual - Use-case detail/scenario for use-case the student is in-charged
* Section 6: Individual - Non-interactive functional requirements
* Section 7: Individual – Conclusion
* Section 8: Further notes

# 2. Overall Description (Group work)

## 2.1. System Use Case Diagram



*Figure 1.1: System Use-case Diagram For Smart Campus System (SCAMS)*

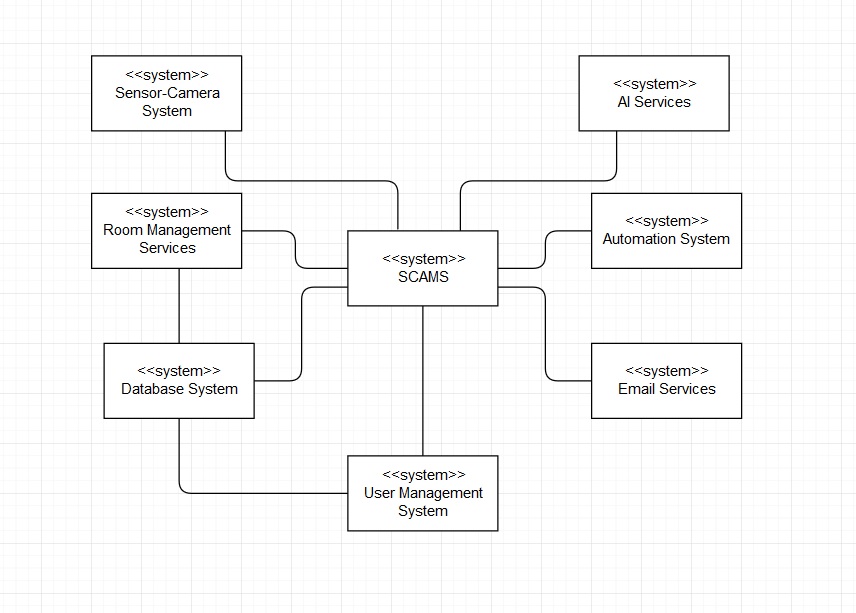
The SCAMS system includes/connects to the following components:

* **Room Information and Schedule**: connects Room Management Service (ROMS) of the university to fetch room data, book and modify room for different purposes.
* **IoT system**: connects to the IoT system of the university to view information, control electrical devices.
* **User Database System**: connects to the User Database System of the university to fetch user information and verify user.
* **Infrastructure Database**: connects to the Infrastructure Database of the university to fetch infrastructure and facility information .
* **AI service**: connects to the external AI service for detecting human.
* Emailing System: connects to the external email service to send email automatically to users and other stakeholders.
* **End User Interface System**: includes a system of interfaces for end user on different platforms.
* **Statistics and Reports System**: includes a systems of statistics and reports for different purposes.

End-users of SCAMS system:

* Lecturers
* Security Staffs
* Office Staffs
* Students
* Guests
* Board of Rectors
* System Administrators

## 2.2. System Context Diagram



*Figure 1.2: Context-model Diagram*

# 3. Functional Requirement Specifications (Group work)

## 3.1. Room Information and Schedule

### 3.1.1. Search room for information

All users can search for room information (available or not, capacity, current lecture/event information, direction to the room) using date, building, room number.

### 3.1.2. Search room for booking

Lecturers and office staff can search room to see its information (available or not, capacity) for booking purpose, using search criteria: date, building, room number.

### 3.1.3. Book room

Lecturers and office staff can book a room in one or more class periods for different purposes (teaching, event).

### 3.1.4. Cancel room

Lecturers and office staff can cancel a room they booked if they change their schedule.

### 3.1.5. Modify room

Lecturers and office staff can modify a room they booked if they change their schedule.

### 3.1.6. Review booking

Lecturers and office staff can review all of their booked rooms on the application

### 3.1.7. Confirm by email

Email will be sent to the lecturer/office staff email address when he/she finished booking/modify/canceling room.

### 3.1.8. Remind by email

User (lecturer & office staff) will receive a reminder of his/her booked room's information & schedule before the lecture.

### 3.1.9. Campus map

User can open the whole campus map on the application.

### 3.1.10. Recommend room

The system recommend lecturer the suitable room for them.

### 3.1.11. Search Lecturer

Lecturers, Students, Office Staff and Security Staff can search for lecturers information.

### 3.1.12. Borrow Laboratory

User can register for borrowing laboratory equipment.

## 3.2. Electrical Automation System

### 3.2.1. Turn on automatically

The doors and its electrical (lights, fans) devices will be turned on automatically before the lecture if they are off.

### 3.2.2. Turn off automatically

Electrical devices will be turned off and doors will be closed after the end of lecture & after the last person left the room if there are no next lecture in the same room.

### 3.2.3. Alert not functional

An alert will be sent to the security staff if doors the automation system is function properly (not on or of as it has to be).

### 3.2.4. Alarm automatically

The sound will be alarmed at if there is still some person in the room after the closed hour of the university.

## 3.3. Security Staff

### 3.3.1. Search for electrical devices

Security staff can search for electrical devices by room's ID, building's ID.

### 3.3.2. Search for active room

Security can search for rooms in the campus which is still in active after the closed hour.

### 3.3.3. Control electrical devices remotely

Security staff can control electrical devices using the application.

### 3.3.4. Control electrical devices in-person

Security staff use code on the system to access control panel attached on each room, corridor, building to control in-person.

### 3.3.5. Monitor camera

User (security staff) can monitor rooms, corridor and building using camera.

### 3.3.6. Enter the Campus

User can use the changeable QR code provided by the system to verify identification in order to enter the campus instead of using the ID card.

## 3.4. Sensor and Camera System

### 3.4.1. Check number of people

User can check for the number of people in the classroom currently.

### 3.4.2. Automatically control the light

The lights in corridors can be turned on/off automatically based on human detecting system.

### 3.4.3. Alert detecting human

The system can send an alert to the security staff in case of detecting human.

### 3.4.4. Detect fire

The system is able to detect fire based on a system of sensor detecting smoke and fire.

## 3.5. SOS Alert

User can send SOS alert to every user in the campus if he/she is in danger.

## 3.6. Statistical and Reporting System

### 3.6.1. View electrical usage

System Administrators and Board of Rectors can view reports on the usage of electrical devices.

### 3.6.2. View room usage

System Administrators and Board of Rectors can view reports on the usage of electrical devices in a period of time.

### 3.6.3. Automatically send report

A monthly report will be generated and sent automatically to the System Administrators and the Board of Rectors via email.

### 3.6.4. User feedback

Any user can provide information and suggestions about the use of electrical devices to the System Administrators using the reporting box.

# 4. Non-Functional Requirements Specification (Group work and Individual work)

## 4.1. Group work - Non-Functional Requirements Specification

### 4.1.1. Efficiency requirements

* Response Time:
* The maximum start up time of the system is 2 seconds
* The maximum receiving search result time is 4 seconds.
* The maximum react time of the system for a particular job is 2 seconds.
* The doors and its electrical (lights, fans) devices will be turned on automatically 15 minutes before the lecture if they are off.
* Electrical devices will be turned off and doors will be closed immediately after the end of lecture & after the last person left the room if there are no next lecture in the same room.
* Report for System Administrator and Board of Rectors will be generated before 5pm on the last working day of the month.
* The maximum interaction for each activity (searching, booking, cancelling, modifying, monitoring, SOS messaging, report retrieving) is 2.
* Space:
* At least 20 percent of the processor capacity and storage space available to the system shall be unused at normal load seasonal periods.
* Performance:
* At least 80 percent of the request must be successful in 1 minute.

### 4.1.2. Regulatory

* The system shall not affect other system while using shared resources.

### 4.1.3. Usability

* User Interface:
* The interface must be available in English and Vietnamese.
* Learnability:
* The product shall be able to used by adult members (age 18 to 60) without training.

### 4.1.4. Reliability

* Readiness:
* The system shall be at least 99 percent available everyday, maintenance is not included.
* Unless the system is non‐operational, the system shall send a user an email informing them that the system is unavailable.
* Fault tolerance:
* The system should ensure operation in case of errors.
* Resilience:
* The system should recover operation in at most 3 hours in case of incidents.

### 4.1.5. Confidentiality

* Security:
* The system should provide multi-layer security for accessing management tools.
* Integrity:
* Username and password of users should be encrypted.
* Authentication:
* The system should provide two-factor authentication.
* The system should authenticate by a sequence of 4 numbers (PIN) besides password.

### 4.1.6. Survivability

* Analytical ability:
* The system should be able to identify issues itself and provide reports.
* Modifiability:
* In source code, function calls shall not be nested more than two levels deep.
* A junior developer who has at least 6 months of experience programming should be able to add a new feature, including code modifications and testing.
* Adaptation ability:
* The system can operate on several platforms: web and mobile.
* The system should display on different resolution based on devices.
* The system must be able to interface with any HTML (HyperText Markup Language) browser.
* Installability:
* The system should be installed easily and automatically.
* The installation time must be within 5 minutes.
* When a new version of the main system is released, it shall be possible to upgrade to it from any previous version.

## 4.2. Individual - Other Non-Functional Requirements

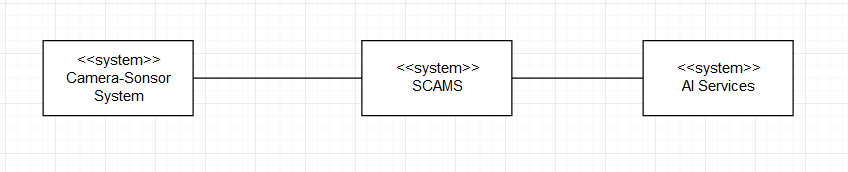
* The action of controlling the devices can be transfer to another security staff or their supervisor in case of the current one can not be able to do it.
* The notification of abnormal room will be pushed to security staff’s mobile phone after closed hour if they exist.
* The number of people in room must be refreshed in every 30 seconds.

*And some specific non-functional requirements in the next section (section 5)*

# 5. Use-case detail/scenario for use-case and specific non-function requirement (Individual work)

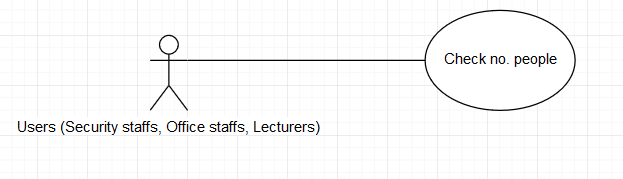
## 5.1 Check Number Of People

### 5.1.1. Context Model:



*Figure 1.3: Context model for checking number of people*

### 5.1.2. Use case diagram:



*Figure 1.4: Use-case Diagram for checking number of people*

### 5.1.3. Use case scenario tabular:

|  |  |
| --- | --- |
| Use case name | Check no. people |
| Actor | Security staffs, Office staffs, Lecturers |
| Description | The users can check the number of people in a class for many reasons such as:   * When lecturers are not in the class while Teaching Assistant is taking that class. The lecturer can check if the class if full of students or not. * Security staffs can check for security issues in self-study rooms. * The office staff who is not able to join an event at a room directly can check if the room is full of attendant or not. |
| Preconditions | The users logged in into SCAMS application and is at the SCAMS homepage |
| Triggers | None |
| Normal Flow | 1. Users selects “Search” on SCAMS home page 2. System represents a search panel with multiple filters (date) and (building, room number). Date and building must be filled 3. Users selects date, building and room’s number from filters and then click/touch on “Search” button. 4. System represents room’s information including room’s number, the building and a button “Check No. People” (which is only enabled for Security staffs, Office staffs and Lecturers) next to it. 5. Users click/touch on “Check No. People” button 6. System lets users know the number of people in the class by showing a pop-up. 7. Users quit the pop-up 8. Users quit page 9. Users return to homepage |
| Exception | *Exception at step 5.*  The AI services provided by external company doesn’t work. The system will display a pop-up notifying “This Service Is Not Available” |
| Alternative Flows | *Alternative 1: at step 3 and step 4*  3a. Users select building  3b. System represents a list of several room’s information including number of rooms which are located in that building and a button “Check No. People” (which is only enabled for Security staffs, Office staffs and Lecturers) next to it.  *Continue step 5 in the normal flow*  *Alternative 2: at step 3 and step 4*  3a. Users select room  3b. System represents a list of several room’s information including name of the building whom they are located in and a button “Check No. People” (which is only enabled for security staffs, office staffs and lecturers) next to it.  *Continue step 5 in the normal flow* |

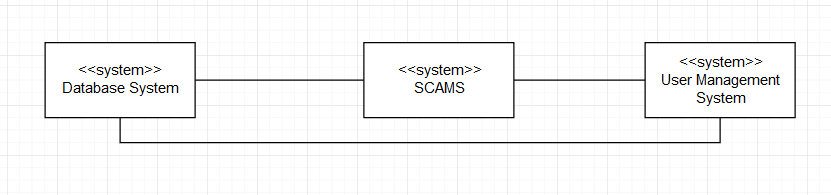
### 5.1.4. Non-function requirement

The system has to make sure that only security staffs, office staffs and lecturers have access to this features.

The system has to make sure that it operates constantly 99 percent of all time.

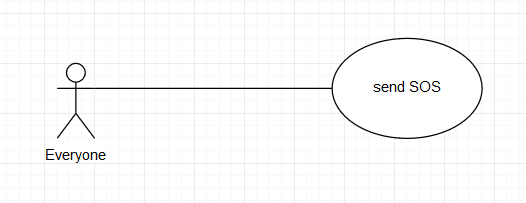
## 5.2 Send SOS

### 5.2.1. Context Model:



*Figure 1.5: Context model for sending SOS*

### 5.2.2. Use-case Diagram:



*Figure 1.6: Use-case Diagram for sending SOS*

### 5.2.3. Use case scenario tabular:

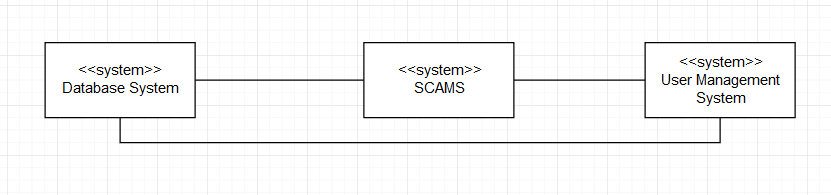
|  |  |
| --- | --- |
| Use case name | Send SOS |
| Actor | Everyone |
| Description | The users can send a SOS alert to everyone else in the system in case of emergency:   * When users are in a dangerous situation (the campus is huge, so evildoers can attack users for robbing) |
| Preconditions | The users logged in into SCAMS application and is at the SCAMS homepage |
| Triggers | None |
| Normal Flow | 1. Users selects tab “SOS” on SCAMS home page  2. System represents a page with text input field and a red button  3. Users fill the content  4. Users tap twice on the red button  5. System represents a pop-up with a notification “Send SOS Successfully” |
| Exception | *Exception at step 4:*  If the input is not filled, a default content will be sent  *Exception at step 5:*  The alert cannot be sent (alternative 2). |
| Alternative Flows | *Alternative 1: at step 3*  3a. User skip this step and move to step 4  *Continue step 4 in the normal flow*  *Alternative 2: at step 5*  5a. System represents a pop-up with a notification “Fail to send SOS” and a button “Try again”  5b. Users click/tap on the button  *Continue until step 5 in the normal flow* |

### 5.2.4. Non-functional requirement

The system has to make sure that SOS alert will be sent with 99 percent of success.

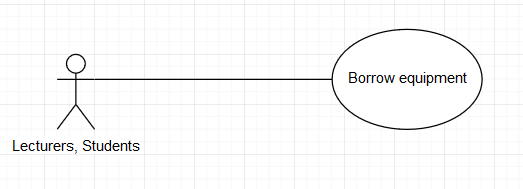
## 5.3. Borrow Equipment

### 5.3.1. Context Model:



*Figure 1.7: Context model for borrowing equipment*

### 5.3.2. Use-case Diagram:



*Figure 1.8: Use-case Diagram for borrowing equipment*

### 5.3.3. Use case scenario tabular:

|  |  |
| --- | --- |
| Use case name | Borrow equipment |
| Actor | Lecturers, Students |
| Description | The users can borrow equipment for working purpose |
| Preconditions | The users logged in into SCAMS application and is at the SCAMS homepage |
| Triggers | None |
| Normal Flow | 1. Users selects tab “Equipment” on SCAMS home page  2. System represents a page with a list of equipment and their ID, name, location and status (available or not) and a “Search” field which contains “ID” and “Name” filters.  3. Users choose the equipment from the list  4. System represents a page with a form including input “Borrowing time”, “Reason”.  5. Users fill in the form  6. Users click/tap “request” button  7. System represents a page with a notification “request is pending”  8. Users quit page  9. Users return to homepage |
| Exception | *None* |
| Alternative Flows | *Alternative 1: at step 2*  3a. Users fill ID or name of the equipment in the field.  3b. Users click/tap “Search” button  3c. System represents a page with a list of equipment and their name, ID, location and status (available or not)  *Continue step 3 in the normal flow* |

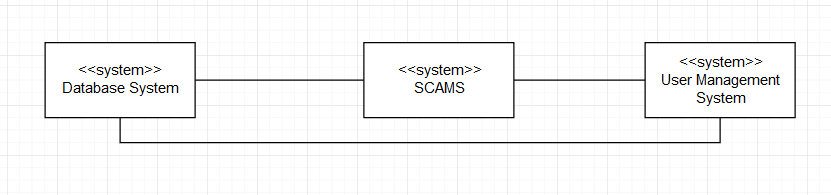
### 5.3.4 Non-functional requirements

The system has to make sure that the borrowing request should be processed in 1 hour.

The system has to make sure that it operates constantly 99 percent of all time.

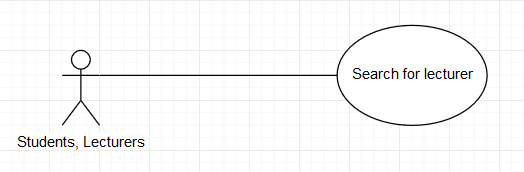
## 5.4. Search for Lecturer

### 5.4.1. Context Model:



*Figure 1.9: Context model for searching for lecturer*

### 5.4.2. Use-case Diagram:



*Figure 1.10: Use-case Diagram for searching for lecturer*

### 5.4.3. Use case scenario tabular:

|  |  |
| --- | --- |
| Use case name | Search for lecturer |
| Actor | Students, Lecturers |
| Description | The users can search for lecturer’s information for contacting purpose such as:   * Portrait photo * Class’s schedule * Email * Phone number * Working room location |
| Preconditions | The users logged in into SCAMS application and is at the SCAMS homepage |
| Triggers | None |
| Normal Flow | 1. Users selects tab “Lecture Information” on SCAMS home page  2. System represents a page with a “Search” field which contains “ID”, “Name”, “Faculty” filters.  3. Users fill in the inputs  4. Users click/tap “Search” button.  5. System represents a page with a list of lecturer’s information that all matches with ID, name and faculty.  6. Users choose lecturer  7. System represents a page with the lecturers’ information in detail.  8. Users search for information that needed  9. Users quit page  10. Users return to homepage |
| Exception | *Exception at step 4: at step 3*  Users do not fill in the field (alternative 2) |
| Alternative Flows | *Alternative 1: at step 3*  3a. Users only fill ID or name or faculty in the field  3b. Users click/tap “Search” button  3c. System represents a page with a list of lecturer’s information that only matches ID or name or faculty.  *Continue step 6 in the normal flow*  *Alternative 2: at step 3*  3a. Users do not fill in the field  3b. Users click/tap “Search” button  3c. System represents a page with a list of all lecturers’ information  *Continue step 6 in the normal flow* |

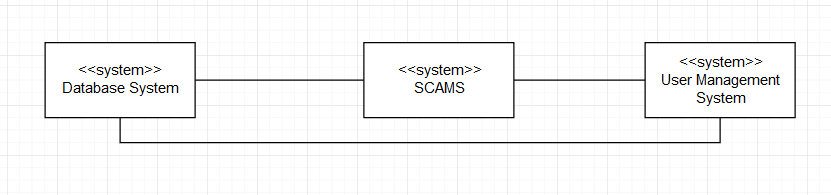
### 5.4.4 Non-functional requirement

The system should have a friendly UI design; content should be readable by people who are color blind.

The system has to make sure that it operates constantly 99 percent of all time.

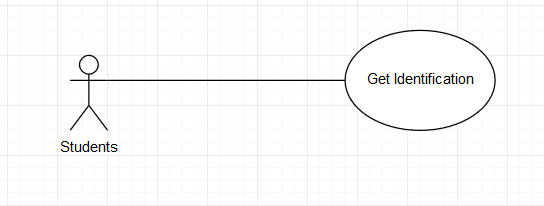
## 5.5. Get Identification

### 5.5.1. Context Model:



*Figure 1.11: Context model for Getting identification*

### 5.5.2. Use-case Diagram:



*Figure 1.12: Use-case Diagram for getting identification*

### 5.5.3. Use case scenario tabular:

|  |  |
| --- | --- |
| Use case name | Get Identification |
| Actor | Students |
| Description | The users can get a proof of identification |
| Preconditions | The users logged in into SCAMS application and is at the SCAMS homepage |
| Triggers | None |
| Normal Flow | 1. Users selects tab “Profile” on SCAMS home page  2. System represents a page with profile of the student  3. Users click/tap “Get Identification” button  4. System represent a pop-up with a QR code  5. Users show the code for security staff to scan  6. The system represent a page with a notification which notifies valid identification  7. Users quit page  8. Users return to homepage |
| Exception | *None* |
| Alternative Flows | *None* |

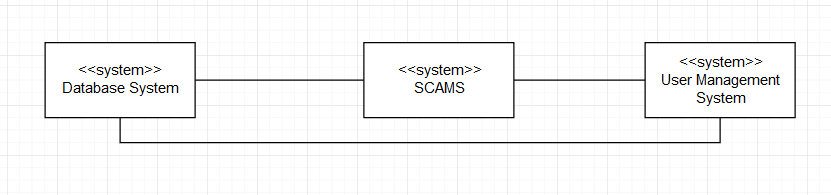
### 5.5.4 Non-functional requirement

The system has to make sure that the QR code should change every day for security purpose.

The system has to make sure that it operates constantly 99 percent of all time.

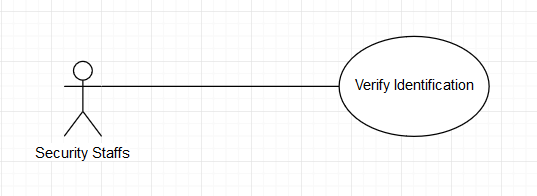
## 5.6. Verify Identification

### 5.6.1. Context Model:



*Figure 1.13: Context model for verifying identification*

### 5.6.2. Use-case Diagram:



*Figure 1.14: Use-case Diagram for verifying identification*

### 5.6.3. Use case scenario tabular:

|  |  |
| --- | --- |
| Use case name | Verify Identification |
| Actor | Security Staffs |
| Description | The security can check if a people is a valid student or not |
| Preconditions | The users logged in into SCAMS application and is at the SCAMS homepage |
| Triggers | None |
| Normal Flow | 1. Users selects tab “Profile” on SCAMS home page  2. System represents a page with profile of the security staff  3. Users click/tap “Verify Identification” button  4. System represent a camera interface for scanning students’ QR code  5. Users scan the QR code which is provided by the student  6. The system represent a page with a notification which notifies valid identification  7. Users quit page  8. Users return to homepage |
| Exception | *Exception 1 at step 6:*  The system represents a page with a notification which notifies invalid identification  *Exception 2 at step 6:*  The system does not recognize the QR code (alternative 1) |
| Alternative Flows | *Alternative 1: at step 6*  6a. System represents a pop-up with “send report” button  6b. Users click/tap on the button  *Continue step 7 in the normal flow* |

### 5.6.4 Non-functional requirement

The system has to make sure that only security staffs have this ability and it can recognize the code immediately.

The system has to make sure that it operates constantly and accurately 99 percent of all time.

# 6. Other non-interactive functional requirement (Individual work)

## 6.1. Send notifications about quotes

On the mobile phone, SCAMS application is able to send a notification about quotes by famous people to lock screen periodically and automatically

## 6.2. Remind exam schedule.

System is able to automatically send a notification which reminds students for upcoming exam schedule.

## 6.3. Check students’ attendance

The system is able to check students’ attendance through camera system and AI services, and then automatically send a report to lecturers who is responsible for that class.

# 7. Conclusion (Individual work)

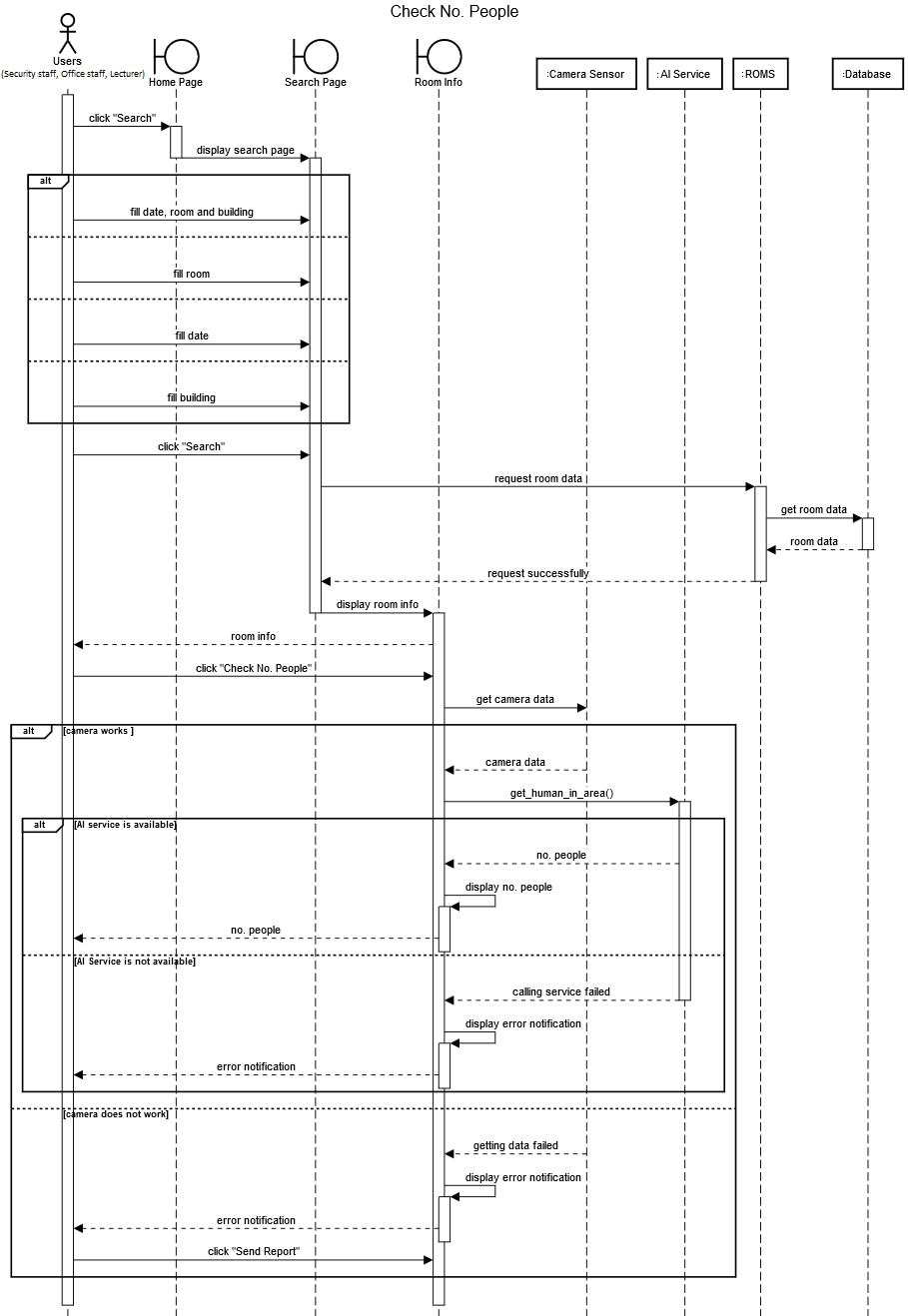
After team-working and personal working. This report has completed tasks in assignment-1 about functional requirements and non-functional requirements.

# Part 2: ASSIGNMENT 2

# 1. Sequence Diagram

## 1.1 Check Number of People

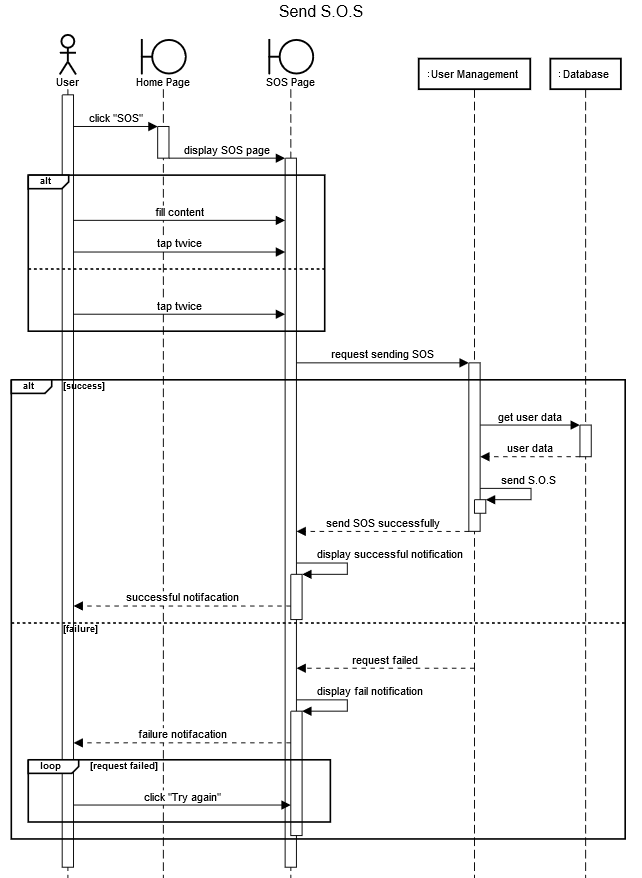
* Brief Description:
* Users open search page
* Users fill in the fields
* Users search for room to check number of people
* SCAMS get data from camera system
* SCAMS transfer data from camera system to AI Service
* AI Service operates for counting number of people
* AI Service return number of people to SCAMS
* SCAMS display number of people to users
* Sequence Diagram:



*Figure 2.1: Sequence diagram for checking no. people*

## 1.2 Send SOS

* Sequence Diagram:

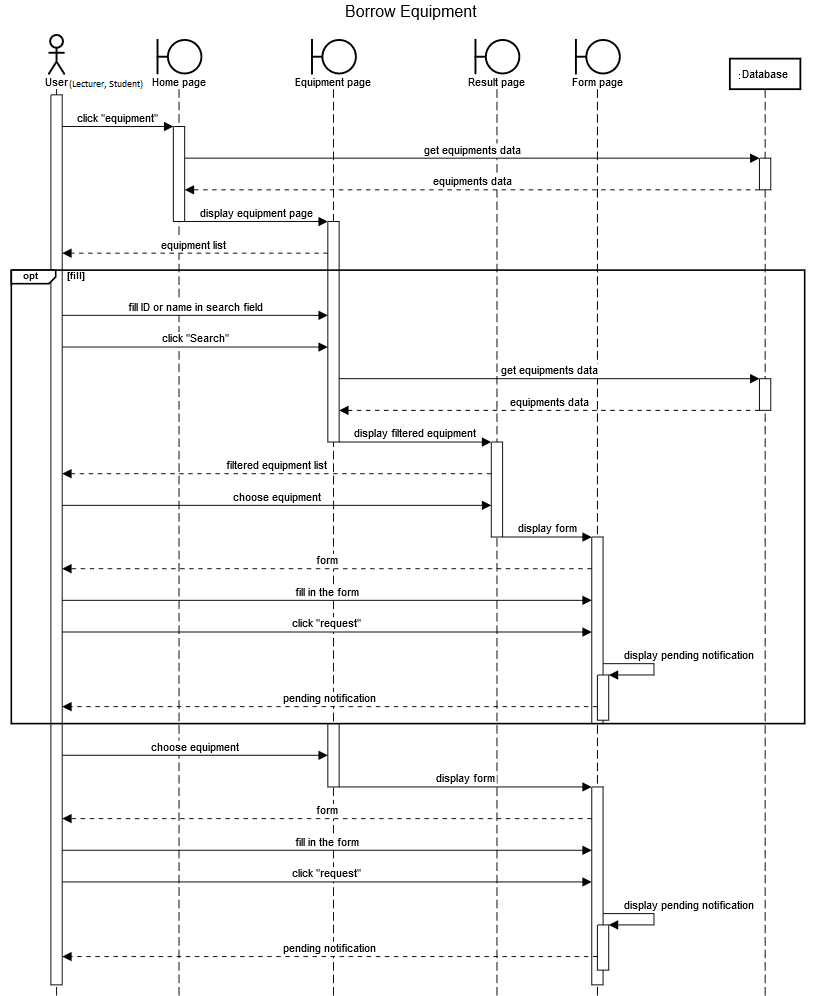


*Figure 2.2: Sequence diagram for sending SOS*

* Brief Description:
* Users open SOS page
* Users fill form (optional)
* Users tap twice on the screen to request sending SOS
* SCAMS send request to User Management System
* User Management System operates sending SOS
* User Management System return the result to SCAMS
* SCAMS display notification to users

## 1.3 Borrow Equipment

* Sequence Diagram:

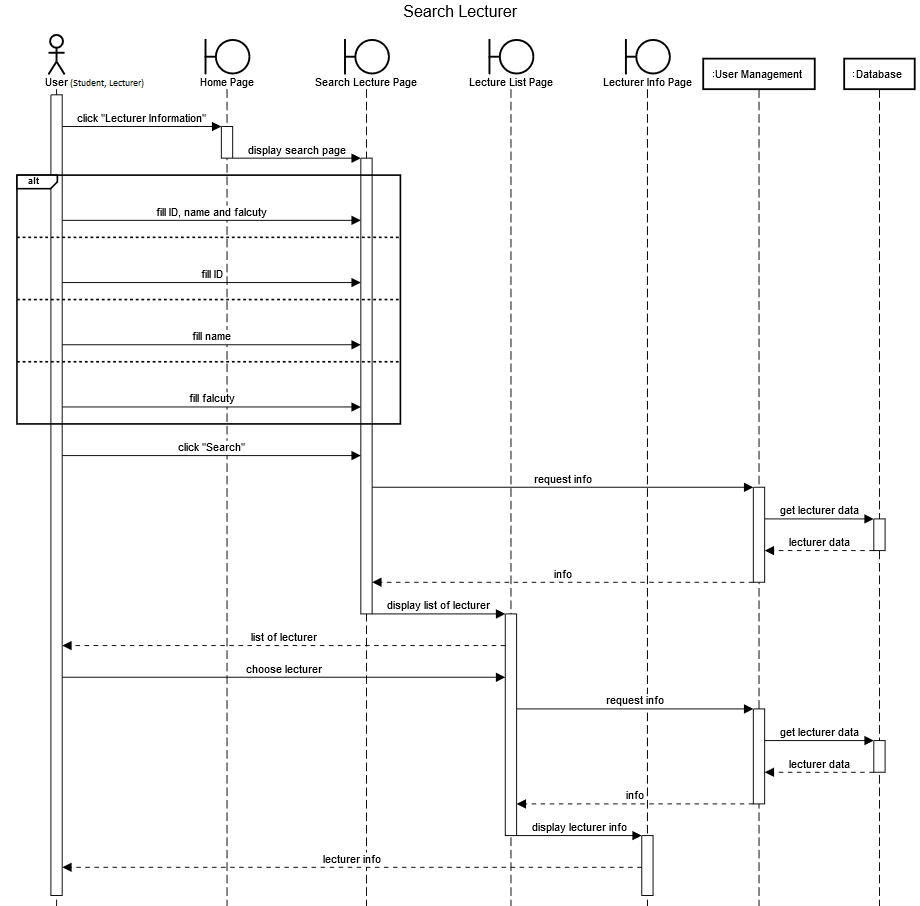


*Figure 2.3: Sequence diagram for borrowing equipment*

* Brief Description:
* Users open equipment page
* Users fill in the fields (optional)
* Users search for the equipment (optional)
* SCAMS display a list of equipment
* Users choose equipment
* Users request for borrowing
* SCAMS display notification to users

## 1.4 Search for Lecturer

* Sequence Diagram:

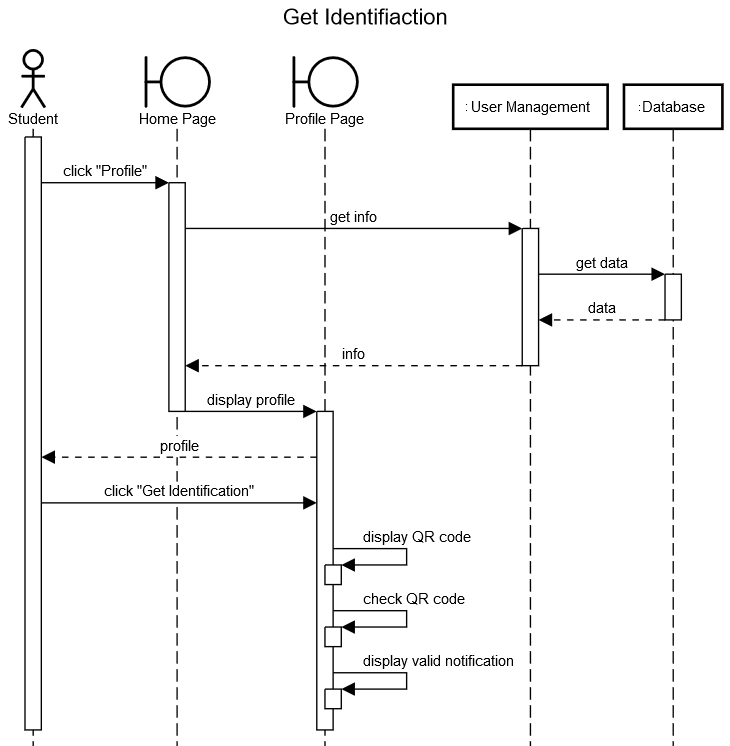


*Figure 2.4: Sequence diagram for searching for lecturer*

* Brief Description:
* Users open lecturer page
* Users fill in the fields
* Users search for lecturer
* SCAMS display a list of lecturer
* Users choose a lecturer to see information
* SCAMS display lecturer information to users

## 1.5. Get Identification

* Sequence Diagram:

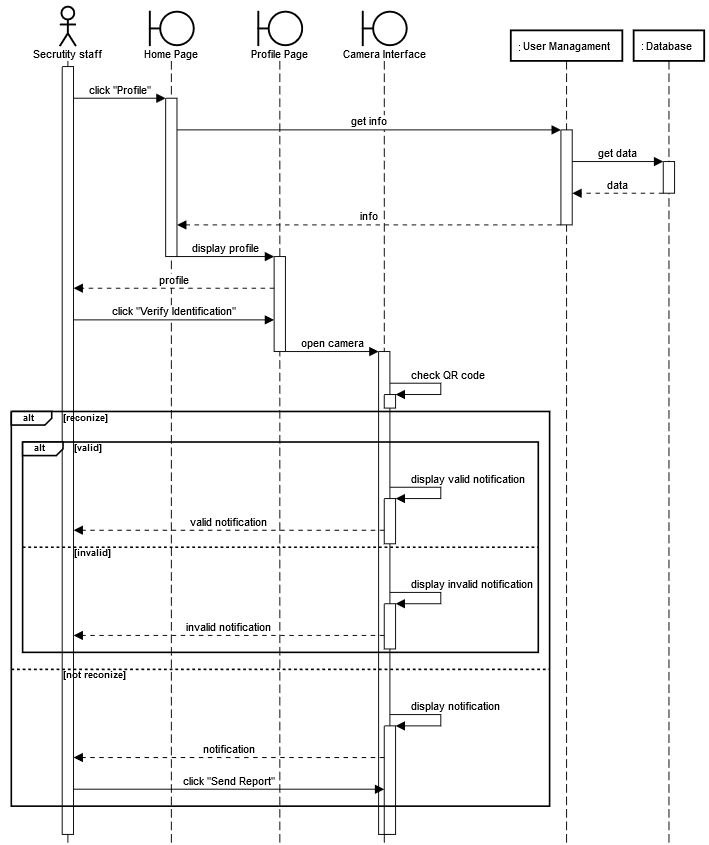


*Figure 2.5: Sequence diagram for getting identification*

* Brief Description:
* Users open profile page
* Users click “get identification”
* SCAMS create a QR code
* Users use the QR code to verify identification
* SCAMS display notification to users

## 1.6 Verify Identification

* Sequence Diagram:



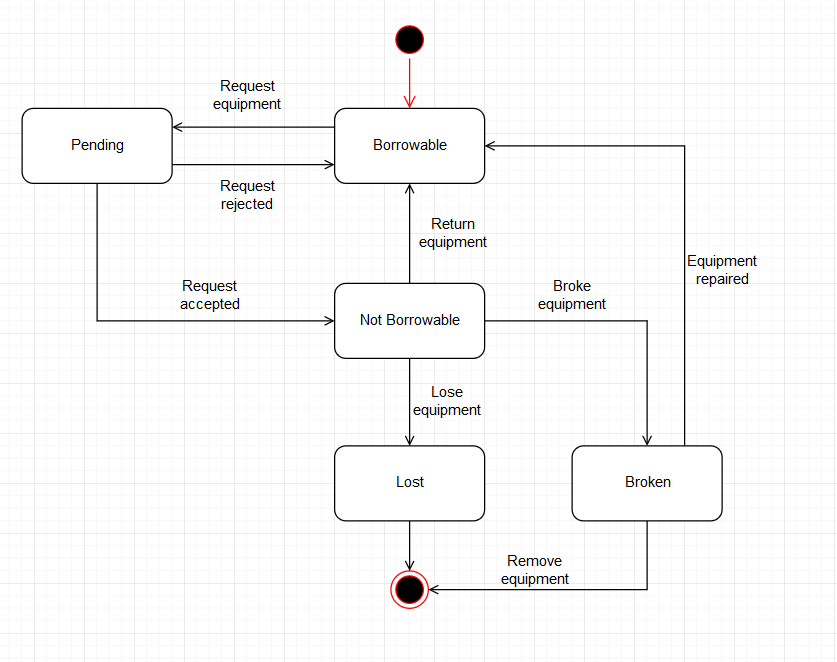
*Figure 2.6: Sequence diagram for verifying identification*

* Brief Description:
* Users open profile page
* Users click “verify identification”
* SCAMS represent a camera interface
* Users scan QR code which provided by the student
* SCAMS check QR code
* SCAMS display notification to users

# 2. State-chart Diagram

## 2.1 Lab equipment state-chart diagram

* State-chart diagram:



*Figure 2.7: State-chart diagram for lab equipment*

* State tabular:

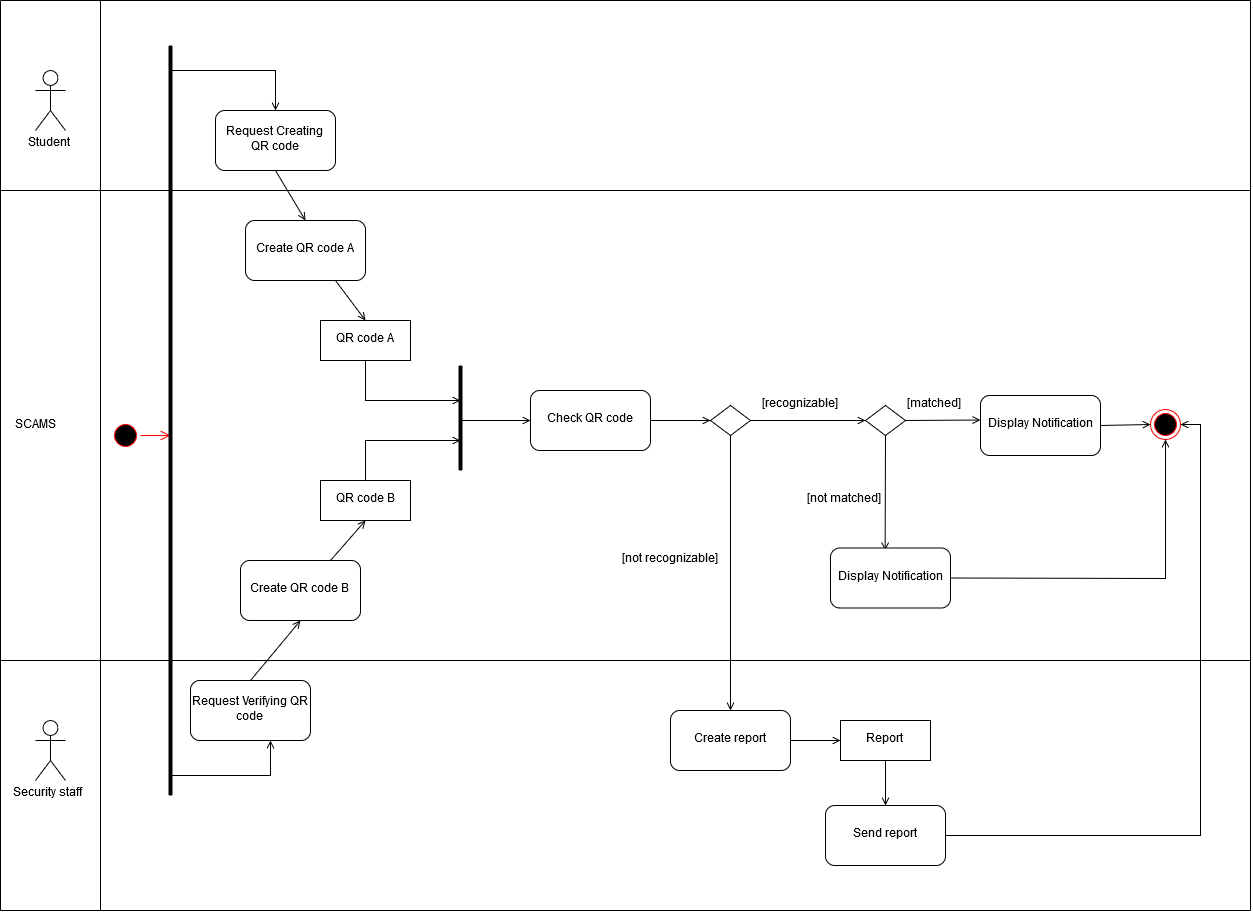
|  |  |
| --- | --- |
| State | Description |
| Borrowable | The equipment is available for borrowing |
| Not borrowable | The equipment is not available for borrowing |
| Pending | The equipment is waiting for confirming |
| Lost | The equipment is now lost |
| Broken | The equipment is now broken |

* Stimulus tabular:

|  |  |
| --- | --- |
| Stimulus | Description |
| Request equipment | The user requests for borrowing equipment |
| Accept request | The office staff accepts request |
| Reject request | The office staff rejects request |
| Return equipment | The user return equipment |
| Lose equipment | The user loses equipment |
| Broke equipment | The user makes equipment broken |
| Repair equipment | The equipment has been repaired |
| Remove equipment | The office staff remove the equipment |

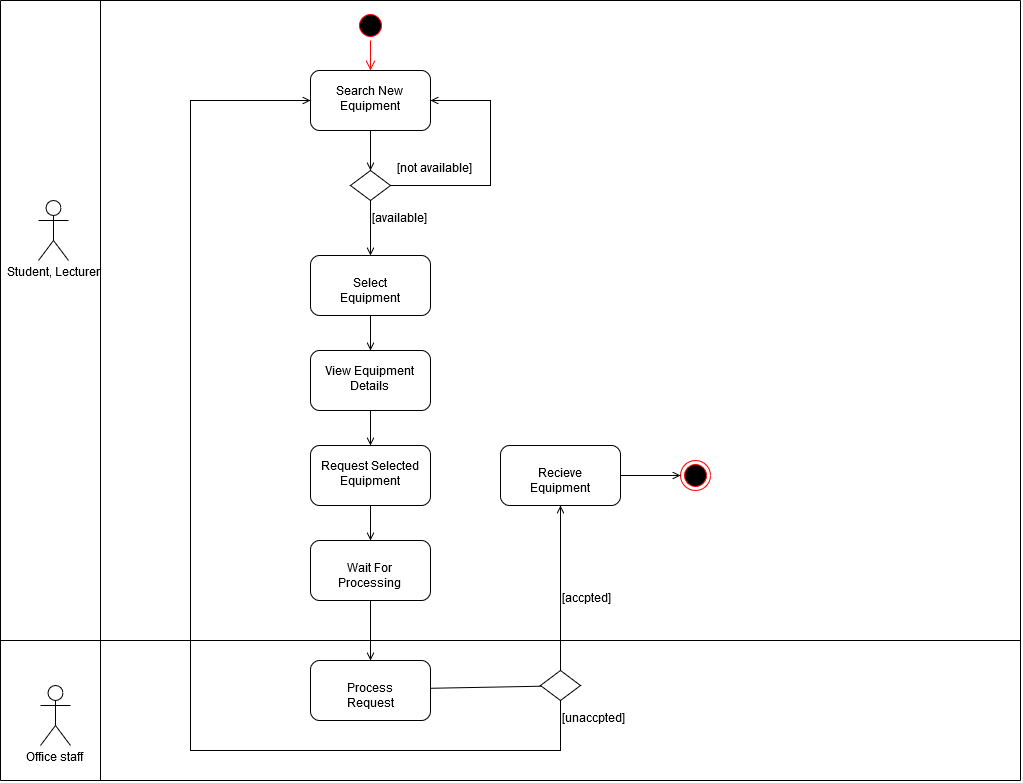
# 3. Activity Diagram

## 3.1 Verify identification activity



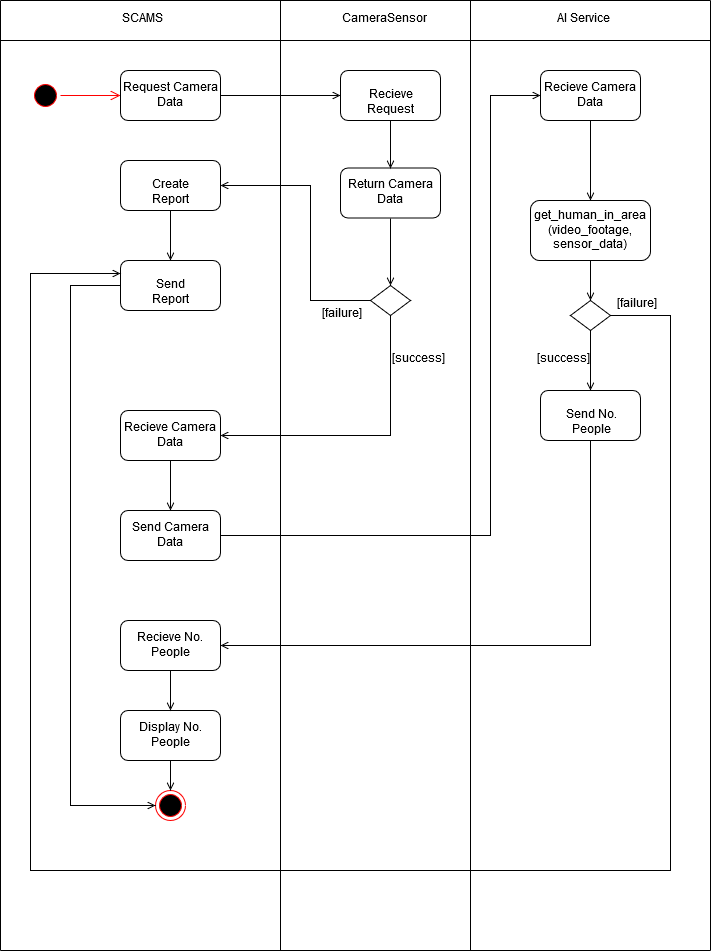
*Figure 2.8: Activity diagram for verifying identification*

## 3.2 Borrow lab equipment activity



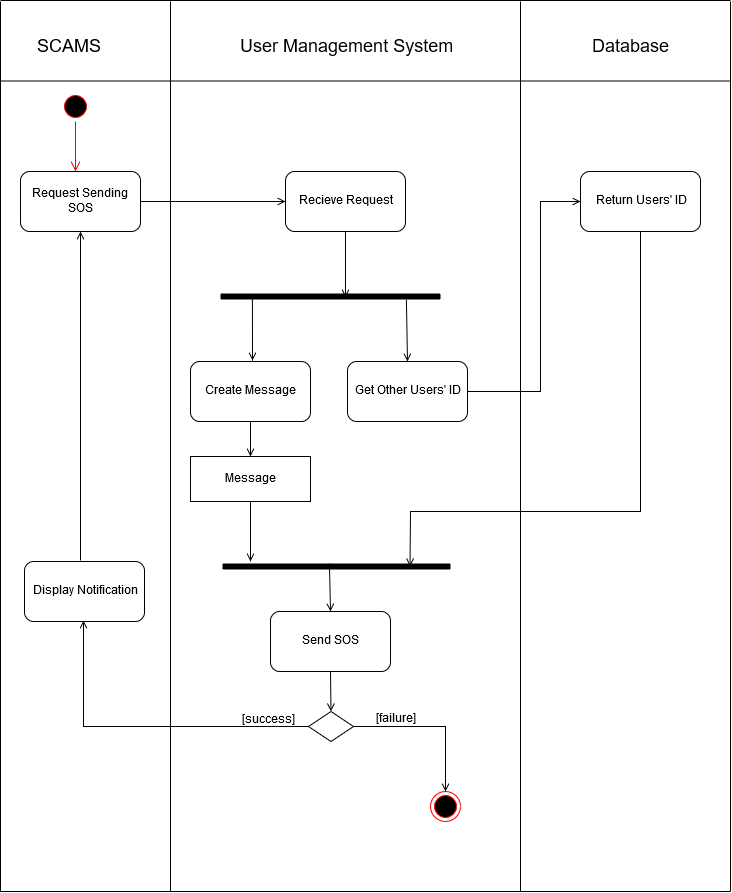
*Figure 2.9: Activity diagram for borrowing lab equipment*

## 3.3 Check number of people activity



*Figure 2.10: Activity diagram for checking number of people*

## 3.4 Send SOS activity



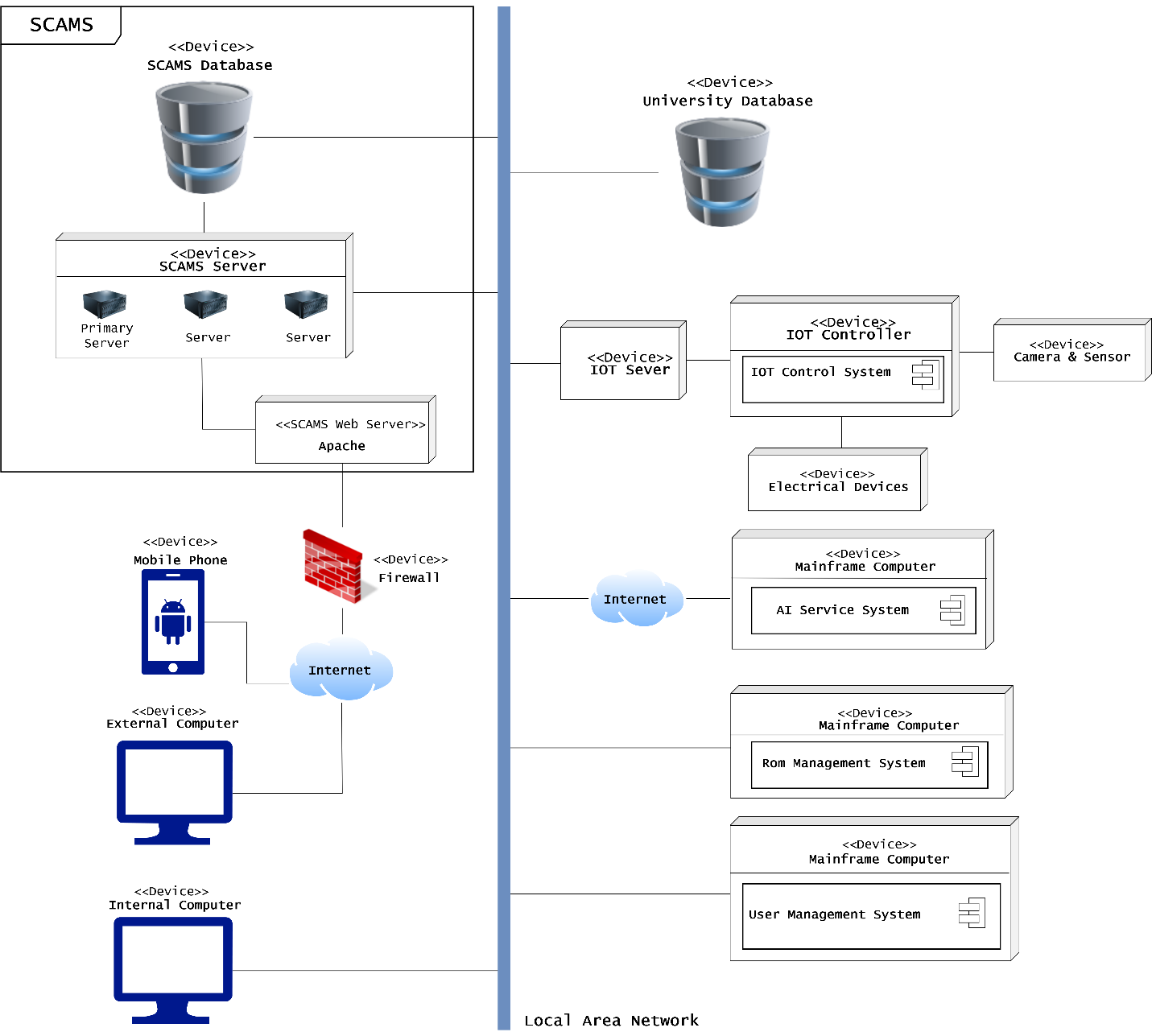
*Figure 2.11: Activity diagram for sending SOS activity*

# 4. Conclusion

This report has completed tasks in assignment-2 about sequence diagram, state-chart diagram and activity diagram.

# Part 3: ASSIGNMENT 3

# 1. Deployment view (Group work)

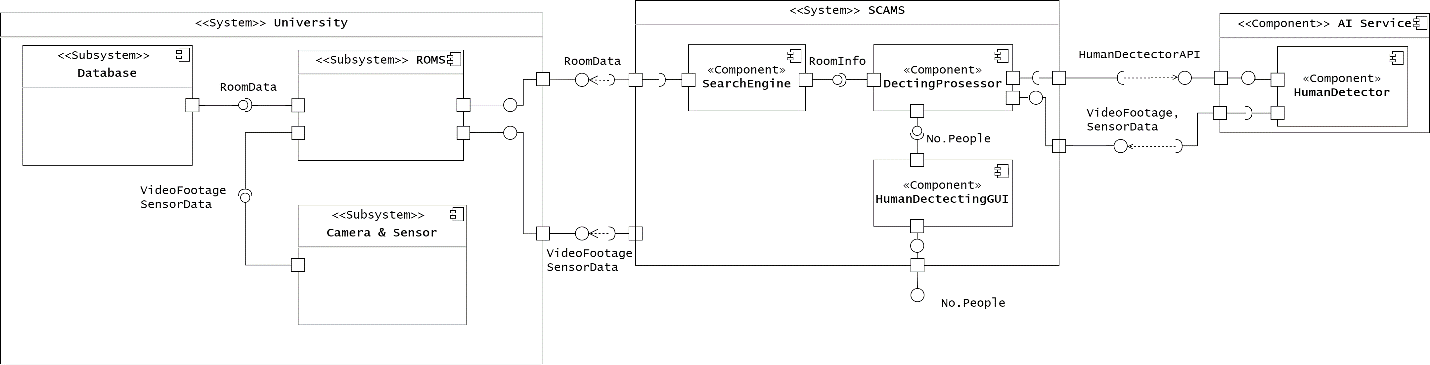


*Figure 3.1: SCAMS Deployment View*

The above figure shows deployment view of our system. Elements inside the system and other elements that the system connects to (University Database, IOT Server, ROMS, User Management System and Internal Computer) are connecting together through LAN. User connects through the internet have to pass the Firewall to access the application. The IOT Server connects to the IOT controller and IOT devices through specific IOT connection protocol (will be defined later on). Connection with AI Service is through the internet.

# 2. Development view

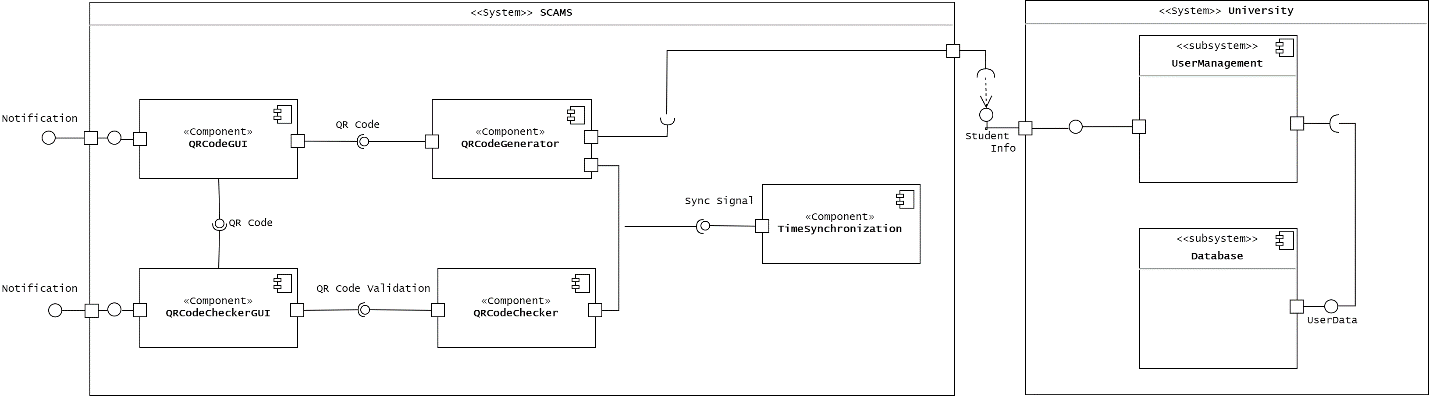
## 2.1 Check Number of People



*Figure 3.2: Development View of Checking Number of People*

The figure above shows the development view of checking number of people; it means that it shows the relationship between components to explain the operation of checking number of people function.

## 2. 2 Check Authorization



*Figure 3.3: Development View of Checking Authorization*

The figure above shows the development view of checking authorization; it means that it shows the relationship between components to explain the operation of checking authorization function.

The purpose of the “TimeSynchronization” component is that it synchronizes the QR code of the student and the QR code checker of the security staff. It means that a QR code which is created by a student will expire in a period of time, this help avoid cheating from student.

## 2. 3 Send SOS



*Figure 3.4: Development View of Sending SOS*

The figure above shows the development view of sending SOS; it means that it shows the relationship between components to explain the operation of sending function.

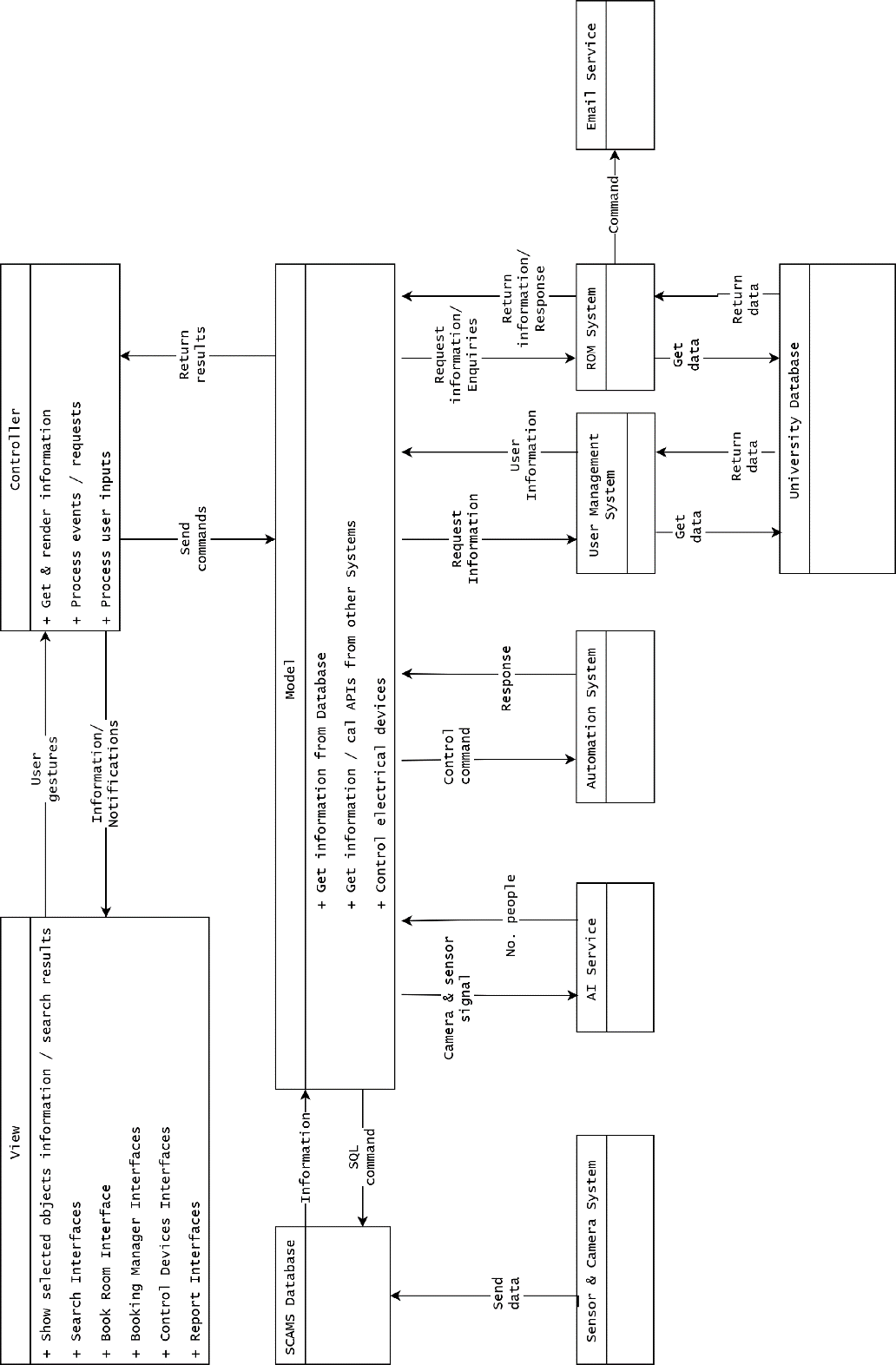
# 3. Conclusion

This report has completed tasks in assignment-3 about deployment view and development view for student who is in charge of their functions.

# Part 4: ASSIGNMENT 4

# 1. Group work

## 1.1. Software Structure – MVC Paradigm



*Figure 4.1: SCAMS MVC Paradigm*

Our MVC view is quite different from the original MVC view, where Model updates View directly. Here Model gets all information and send it to Controller to render the view.

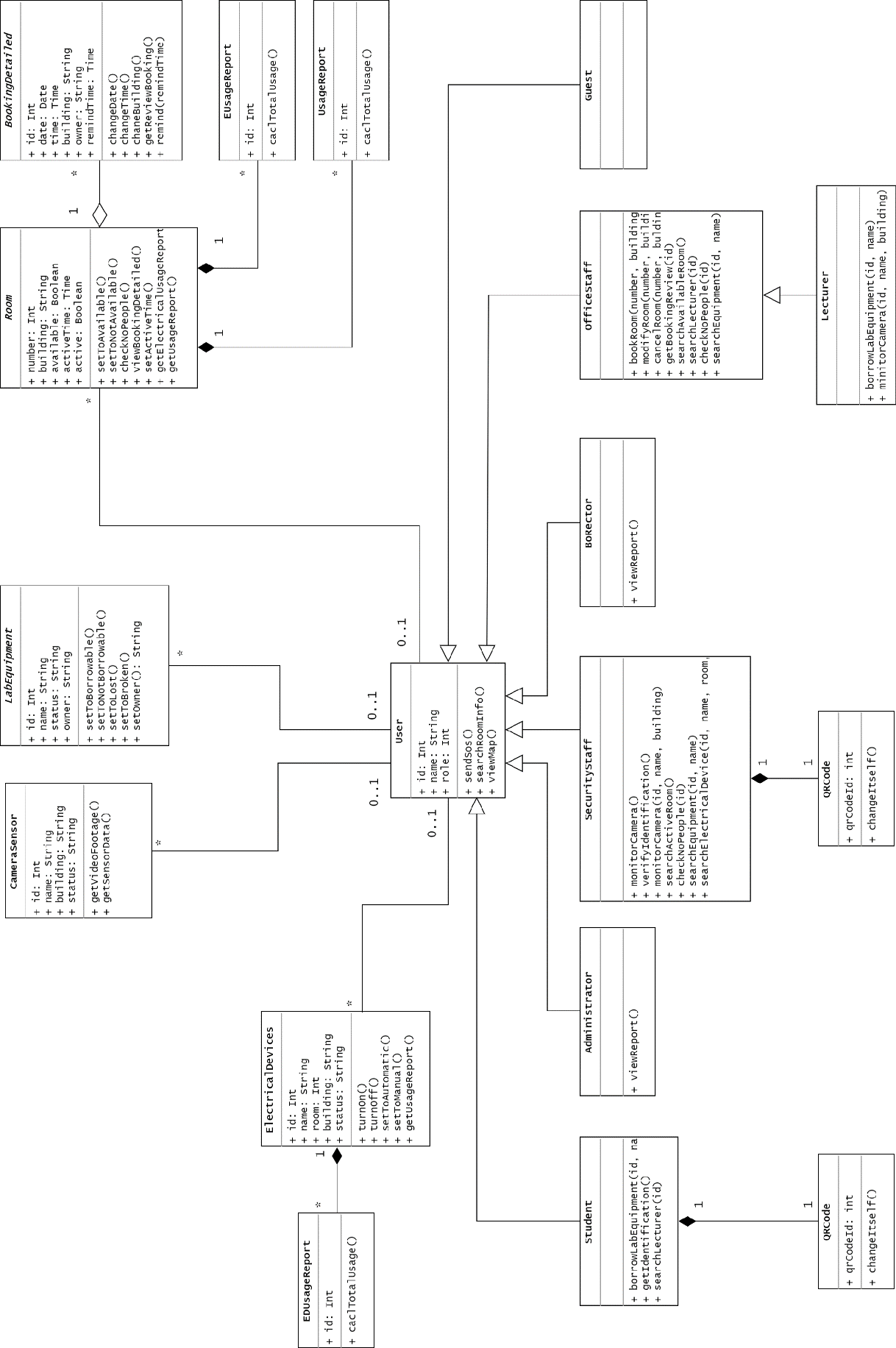
From this viewpoint:

1. User inputs some actions/ information (clicking button, completing form, etc.) from the View.
2. Controller will determine what to do with User’s gesture/information, and send it off to the right Model.
3. Models are Classes in our system, it will process the information from Controller, connects to Databases, call APIs from external Systems and return results to the Controller.
4. After receiving the result, Controller will send back information to the User with appropriate View.

## 1.2. System Class Diagram

The below diagram is class diagram for our system. This is only the Domain Classes with some primary methods.

Each member in our team will implement these methods as closure for the part they are in charge of into Implement Level Class Diagrams.

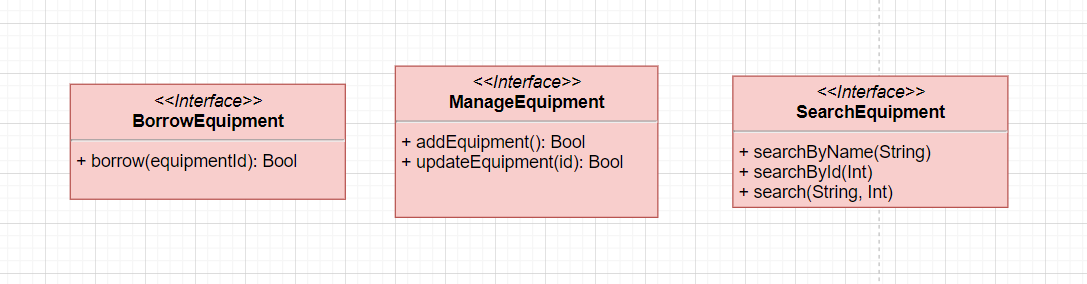


*Figure 4.2: SCAMS Class Diagram*

# 2. Individual work

## 2.1. Borrow Lab Equipment

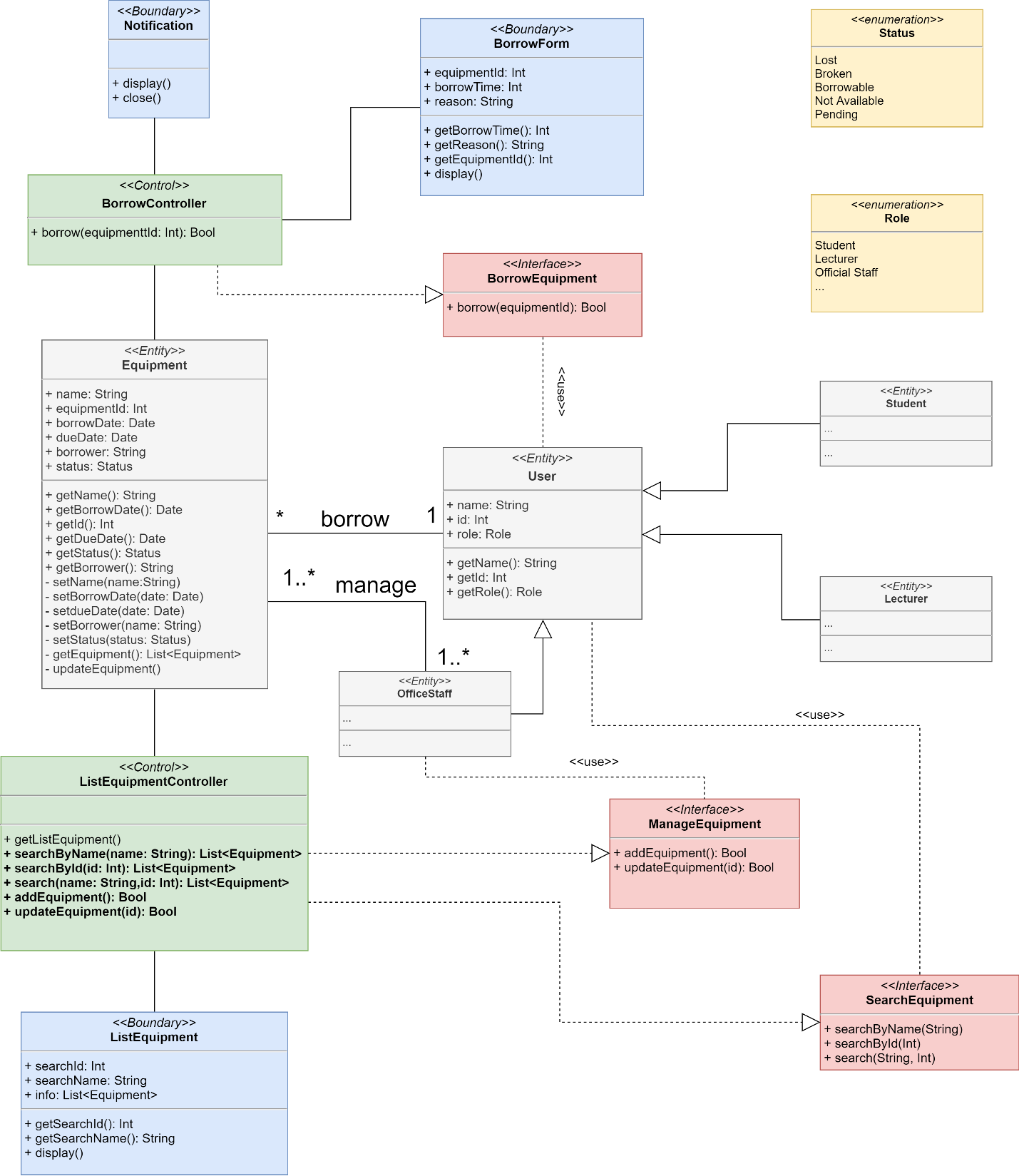
### 2.1.1. Module Interface



*Figure 4.3: Module Interface for Borrow Lab Equipment*

### 2.1.2. Class Diagram

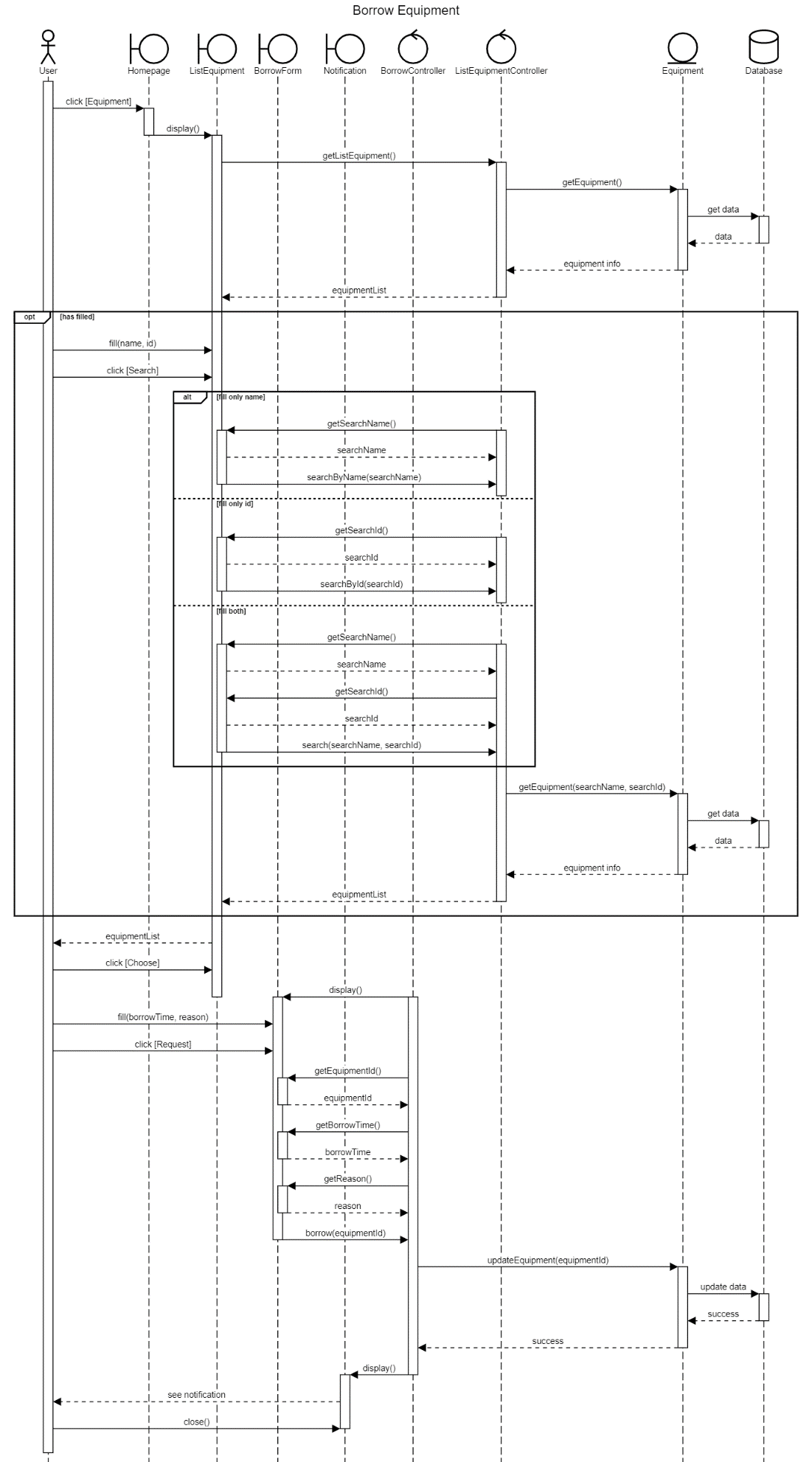
Note that “…” means there are several methods or attributes which is not showed here in this class diagram but still existed. This diagram only shows methods and attributes which relates to the function. The reason why doing this is for synchronizing content among group members.



*Figure 4.4: Class Diagram at detail level*

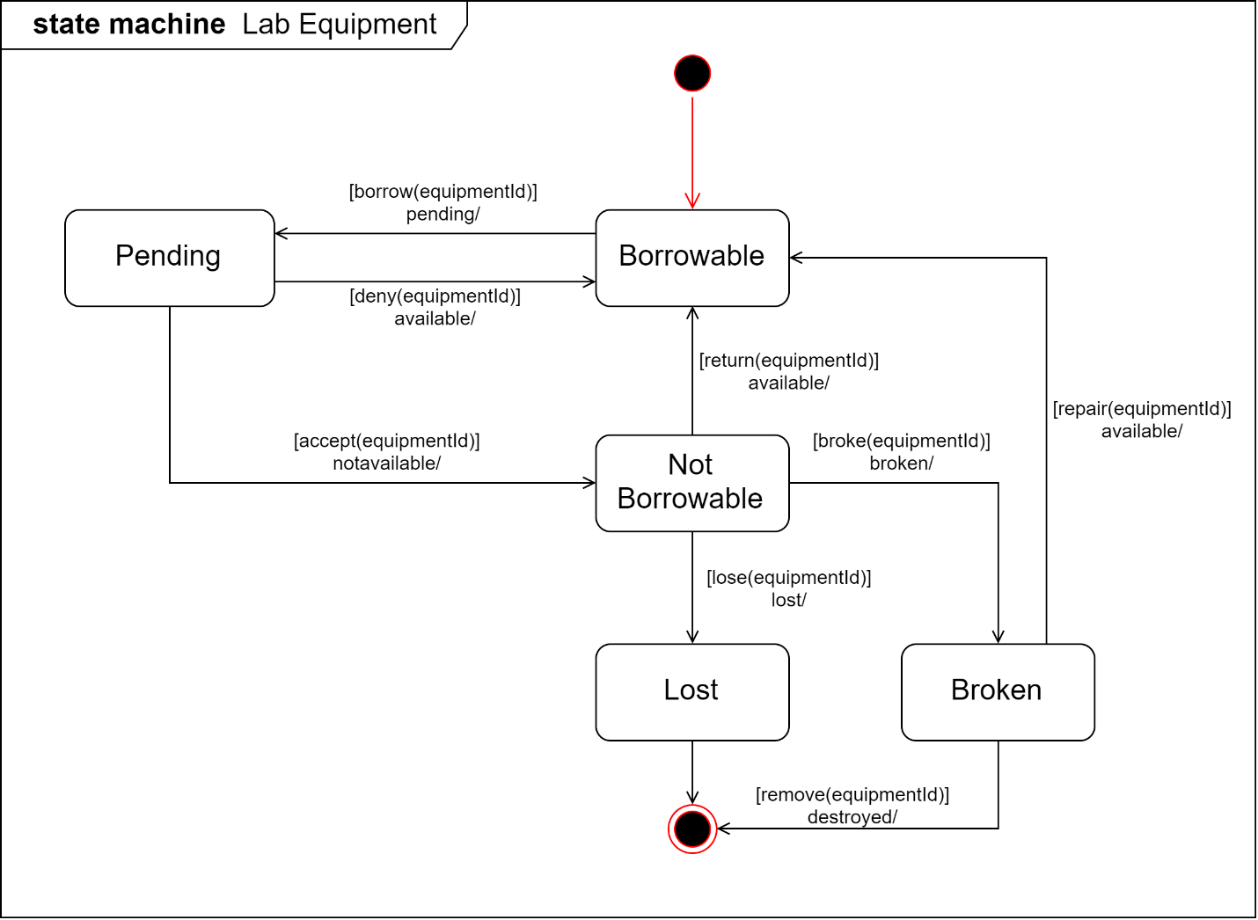
|  |  |  |
| --- | --- | --- |
| Class | Method | Description |
| Equipment | getName() | Get name of equipment |
| getBorrowDate() | Get borrow date of equipment |
| getId() | Get id number of equipment |
| getDueDate() | Get due date of equipment |
| getStatus() | Get status of equipment |
| getBorrower() | Get name of who borrows equipment |
| setName() | Set name for equipment |
| setBorrowDate() | Set borrow date for equipment |
| setDueDate() | Set due date for equipment |
| setBorrower() | Set name for who borrows equipment |
| setStatus() | Set status for equipment |
| getEquipement() | Return a list of all equipment |
| updateEquipment() | Update information of equipment such as status, borrower, due date,… |
| User | getName() | Get name of user |
| getId() | Get id number of user |
| getRole() | Get role of user |
| BorrowController | Borrow(equipmentId) | Get information from BorrowForm, then call method updateEquipment() to update information of equipment. And finally send notification to actor. |
| ListEquipmentController | searchByName(name) | Return a list of equipment with match name to display out |
| searchById(id) | Return a list of equipment with match id number to display out |
| search(name, id) | Return a list of equipment with match both name and id number to display out |
| getListEquipment() | Call method getEquipment then display out |
| addEquipment() | Create new object of class Equipment |
| updateEquipment(id) | Call method updateEquipment() of the equipment matched id |
| Notification | display() | Update view and display out |
| close() | Close view UI |
| BorrowForm | getBorrowTime() | Get borrow time which entered by user |
| getEquipmentId() | Get equipment id which entered by user |
| getReason() | Get reason why user needs to borrow |
| display() | Update view and display out |
| ListEquipment | getSearchId() | Get id which entered by user |
|  | getSearchName() | Get name which entered by user |
|  | Display() | Update view and display out |

### 2.1.3. Sequence Diagram



*Figure 4.5: Sequence Diagram for Borrow Lab Equipment*

### 2.1.4. State-chart Diagram:



*Figure 4.6: State-chart Diagram at detail level*

All above methods have been defined in updateEquipment method (described in Python pseudocode below):

def updateEquipment(self):

    def deny(id):

        self.status = borrowable

    def accept(id):

        self.status = notAvailable

    def lose(id):

        self.status = lost

    def broke(id):

        self.status = broken

    def repair(id):

        self.status = borrowable

    def return(id):

        self.status = borrowable

    def remove(id):

        self.status = notAvailable

### 2.1.5. Design Pattern:

MVC design pattern is applied for this function:

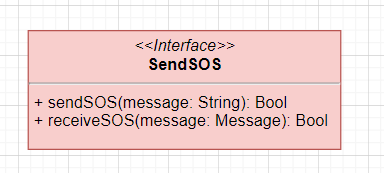
* **Model**: *<<class>>Equipment*: This is data model which is manipulated by the controller.
* **View**:
  + *<<class>>ListEquipment*
  + *<<class>>Notification*
  + *<<class>>BorrowForm*

Every view has an display() method that is called by controller to update view and show to the user.

* **Control**: *<<class>>ListEquipmentController, <<class>>BorrowController*: Controller will handle event of the user through view, and manipulate data of the model.

## 2.2. Send SOS

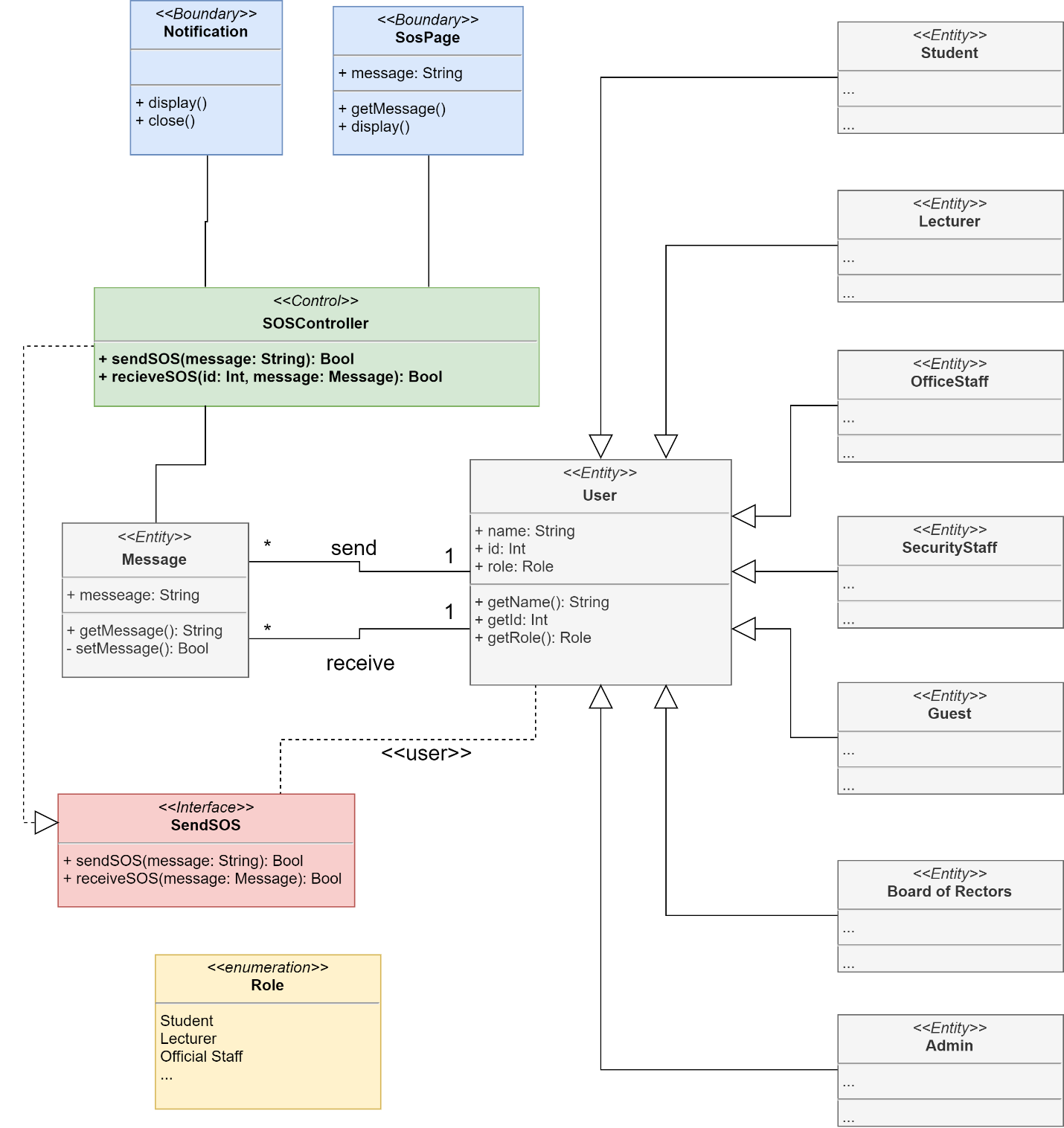
### 2.2.1. Module Interface



*Figure 4.7: Module Interface for Send SOS*

### 2.2.2. Class Diagram

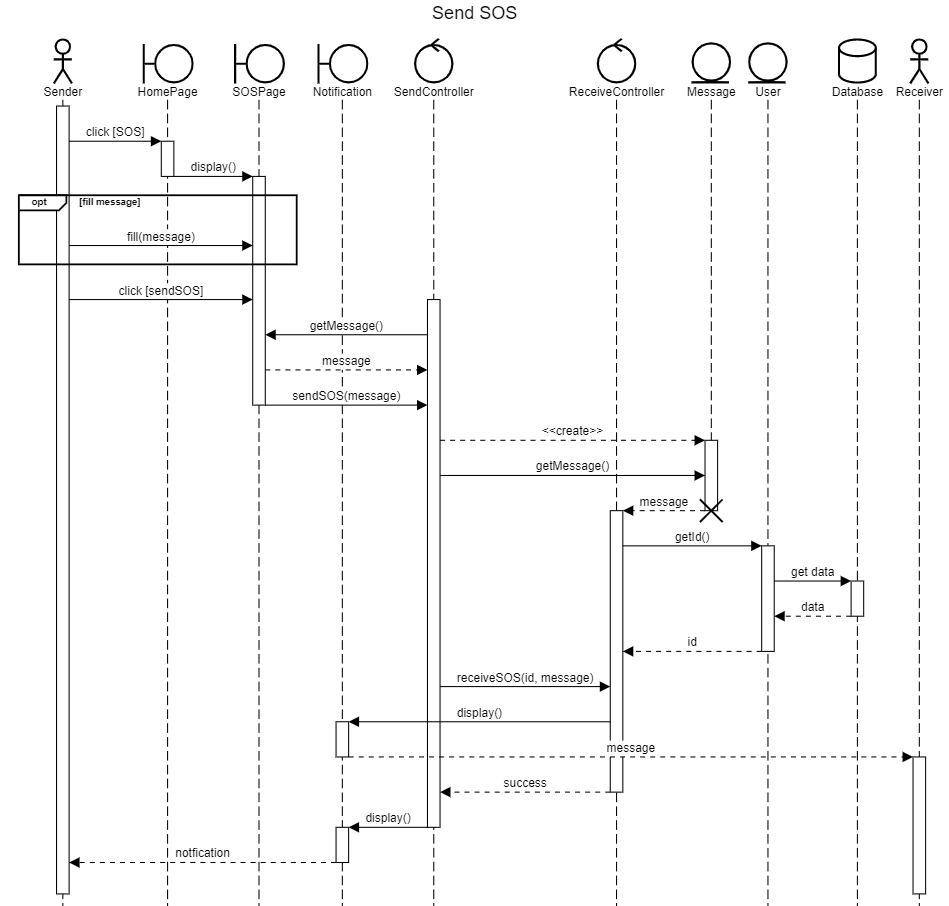
Note that “…” means there are several methods or attributes which is not showed here in this class diagram but still existed. This diagram only shows methods and attributes which relates to the function. The reason why doing this is for synchronizing content among group members.



*Figure 4.8: Class Diagram for Send SOS*

|  |  |  |
| --- | --- | --- |
| Class | Method | Description |
| User | getName() | Get name of user |
| getId() | Get id number of user |
| getRole() | Get role of user |
| Message | getMessage() | Get message which is entered by sender |
| setMessage() | Set message to send |
| SOSController | sendSOS(message) | Handle event send request from sender |
| receiveSOS(id, message) | Handle when receiver receives message with ‘id’ is the id number of the receiver. |
| Notification | display() | Update view and display out |
| close() | Close view UI |
| SOSPage | getMessage() | Get message which is entered by the sender |
| display() | Update view and display out |

### 2.2.3. Sequence Diagram



*Figure 4.9: Sequence Diagram for Send SOS*

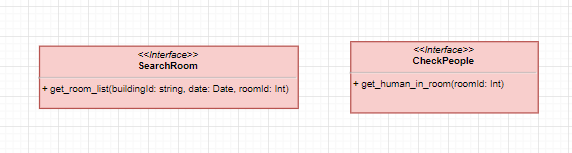
### 2.2.4. Design Pattern

MVC design pattern is applied for this function:

* **Model**: *<<class>>Message*, <<class>>User: This is data model which is manipulated by the controller.
* **View**: *<<class>>SOSPage, <<class>> Notification:* Every view has an display() method that is called by controller to update view and show to the user.
* **Control**: *<<class>>SOSController:* Controller will handle event of the user through view, and manipulate data of the model.

## 2.3. Check Number of People

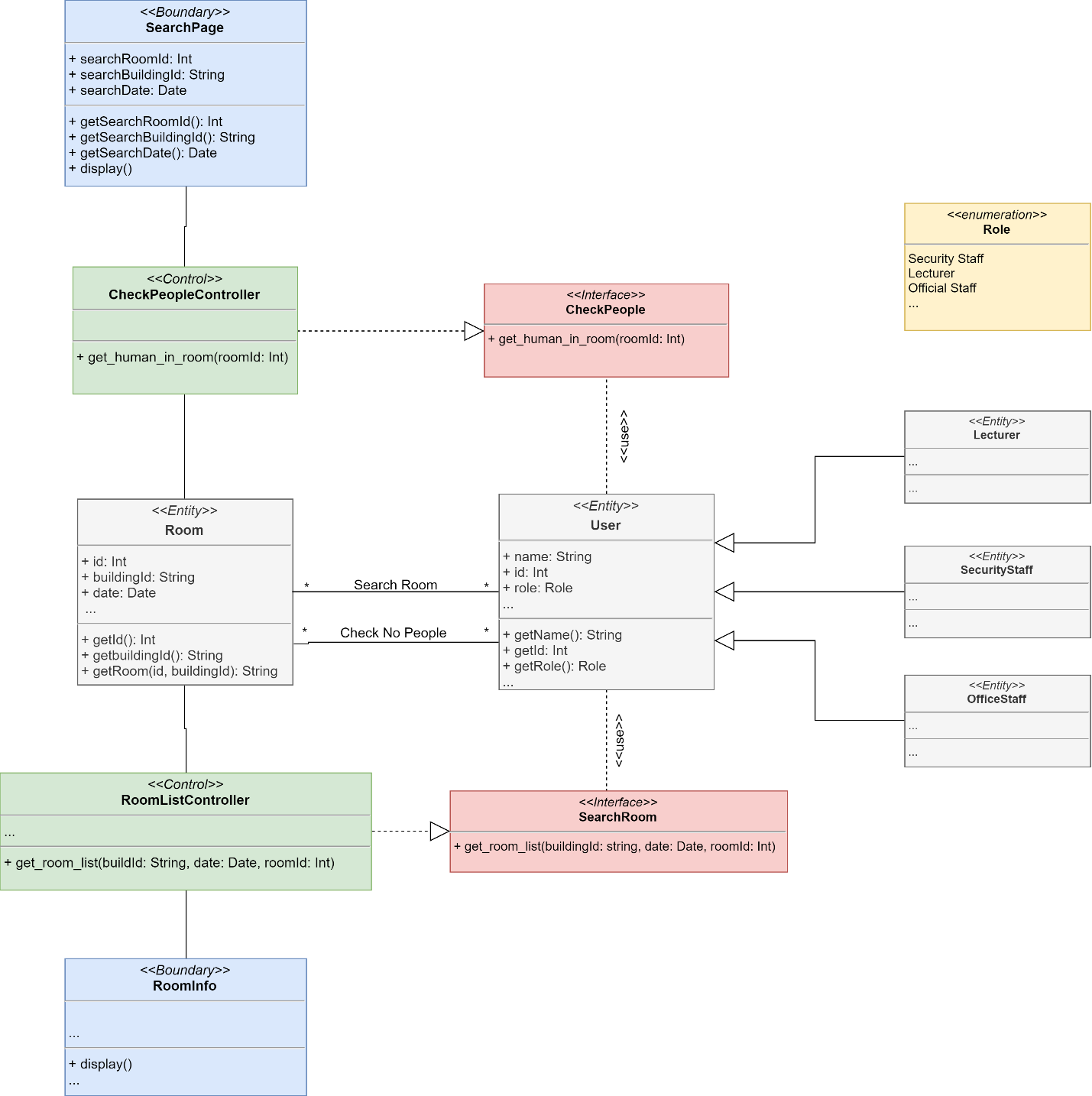
### 2.3.1. Module Interface



*Figure 4.10: Module Interface for Check Number of People*

### 2.3.2. Class Diagram

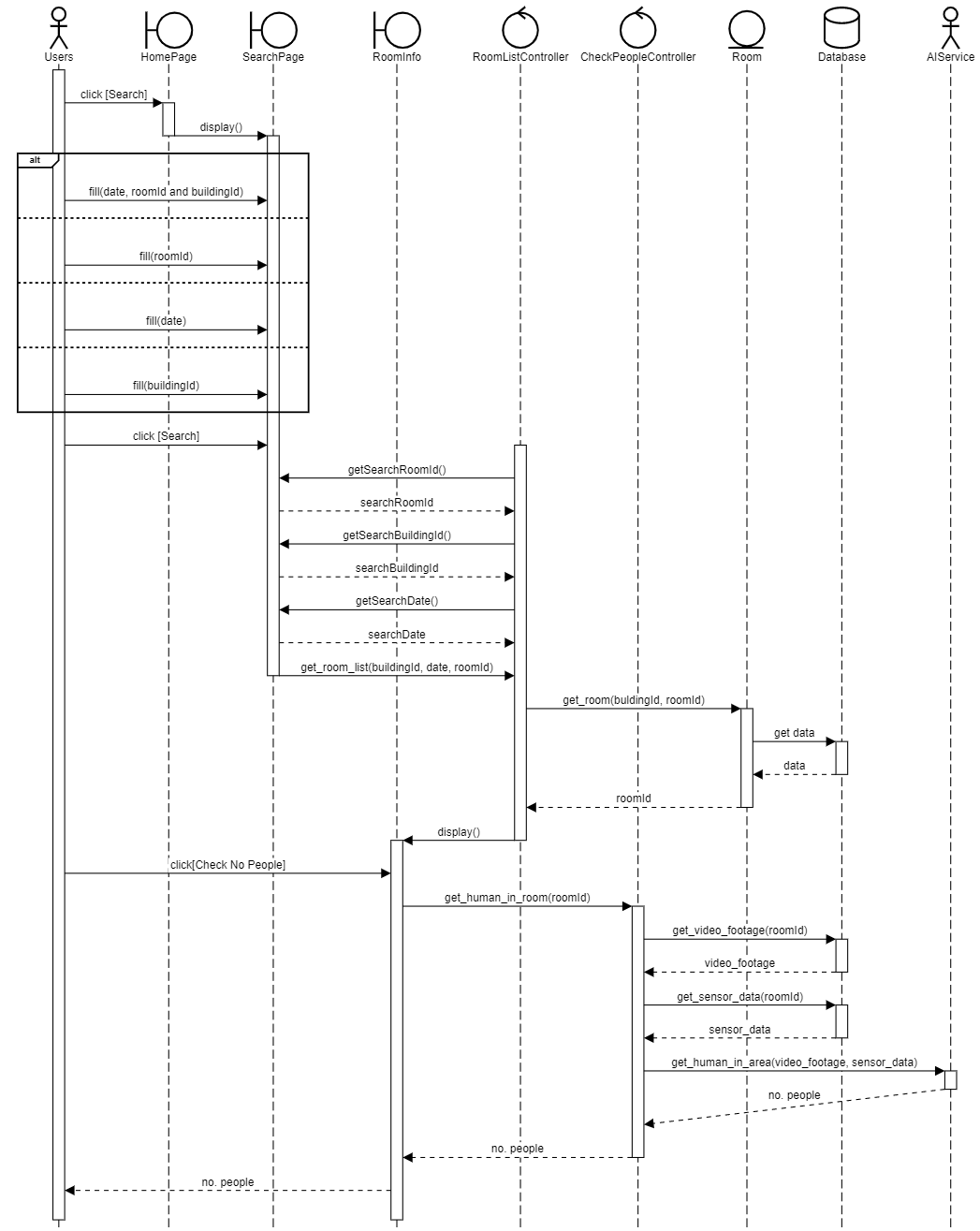
Note that “…” means there are several methods or attributes which are not shown here in this class diagram but do exist. This diagram only shows methods and attributes which relates to the function. The reason why doing this is for synchronizing content among group members and enhancing readable document.



*Figure 4.11: Class Diagram for Check Number of People*

|  |  |  |
| --- | --- | --- |
| Class | Method | Description |
| User | getName() | Get name of user |
| getId() | Get id number of user |
| getRole() | Get role of user |
| Room | getId() | Get id of room |
| getbuildingId() | Get id of the building which the room belongs to |
| getRoom(id, buildingId) | Get room information to display |
| CheckPeople Controller | get\_human\_in\_room(roomId) | Handle request check no. people by call API get\_human\_in\_area(video\_footage, sensor\_data) from AI service |
| RoomListController | get\_room\_list(buildingId, date, roomId) | Handle request get room list by call method get\_room(buildingId, roomId) to get list of room and their information |
| SearchPage | getSearchRoomId() | Get id of the room which is entered by user |
| getSearchBuildingId() | Get id of the building which the room belongs to entered by user |
| getSearchDate() | Get date entered by user |
| display() | Update view and display out |
| RoomInfo | display() | Update view and display out |

### 2.3.3. Sequence Diagram



*Figure 4.12: Sequence Diagram for Check Number of People*

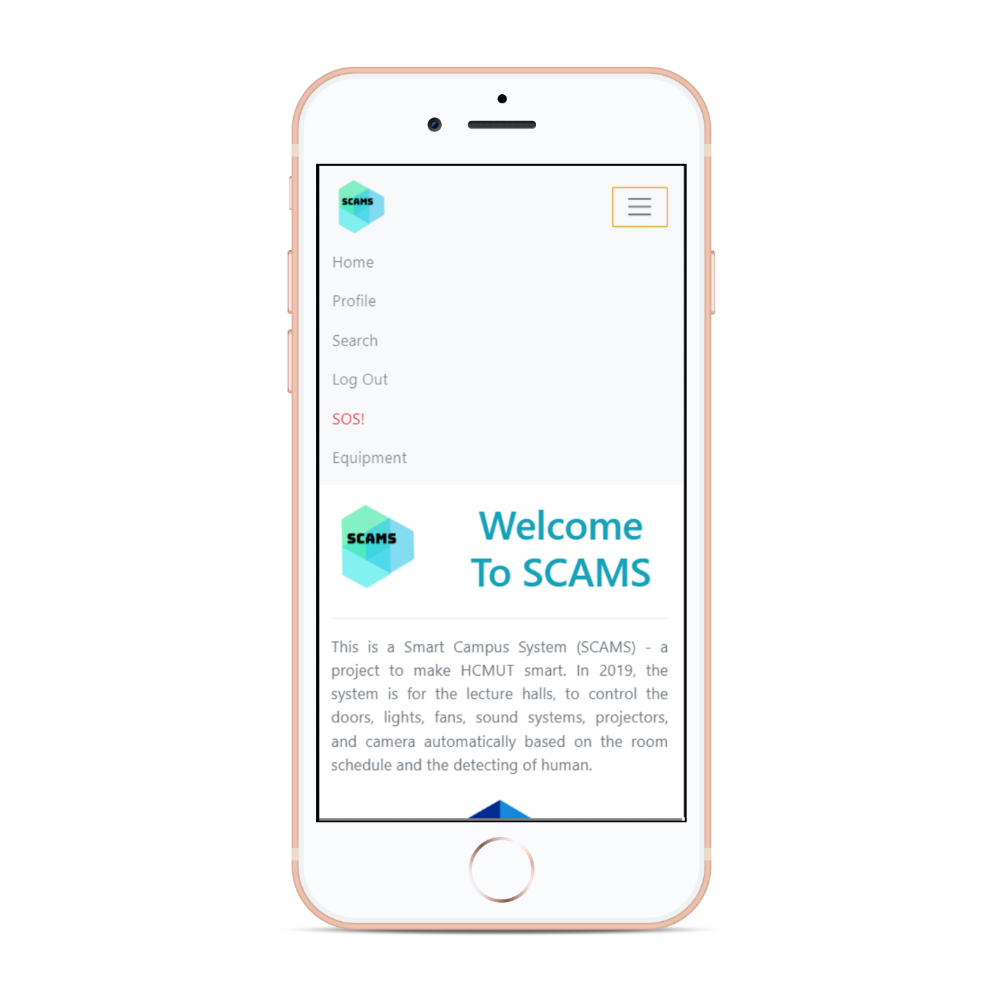
### 2.3.4. Design Pattern

MVC design pattern is applied for this function:

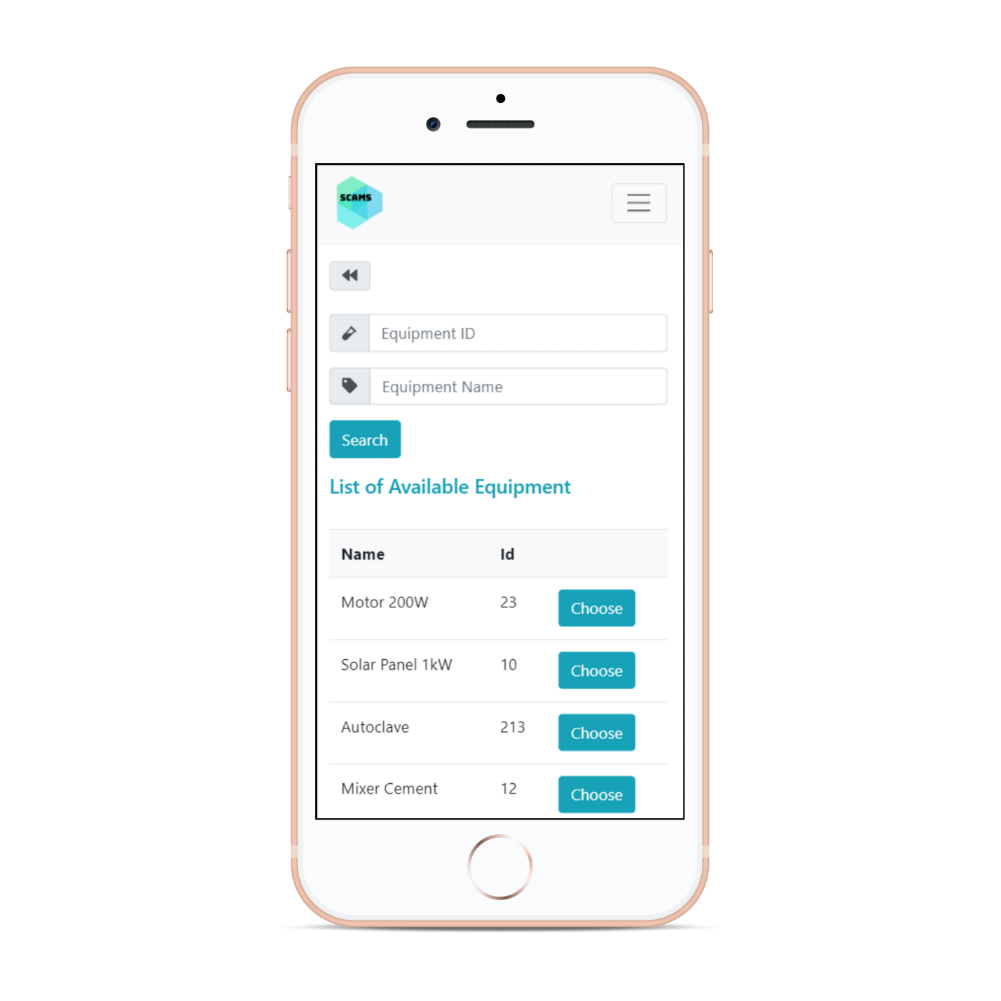
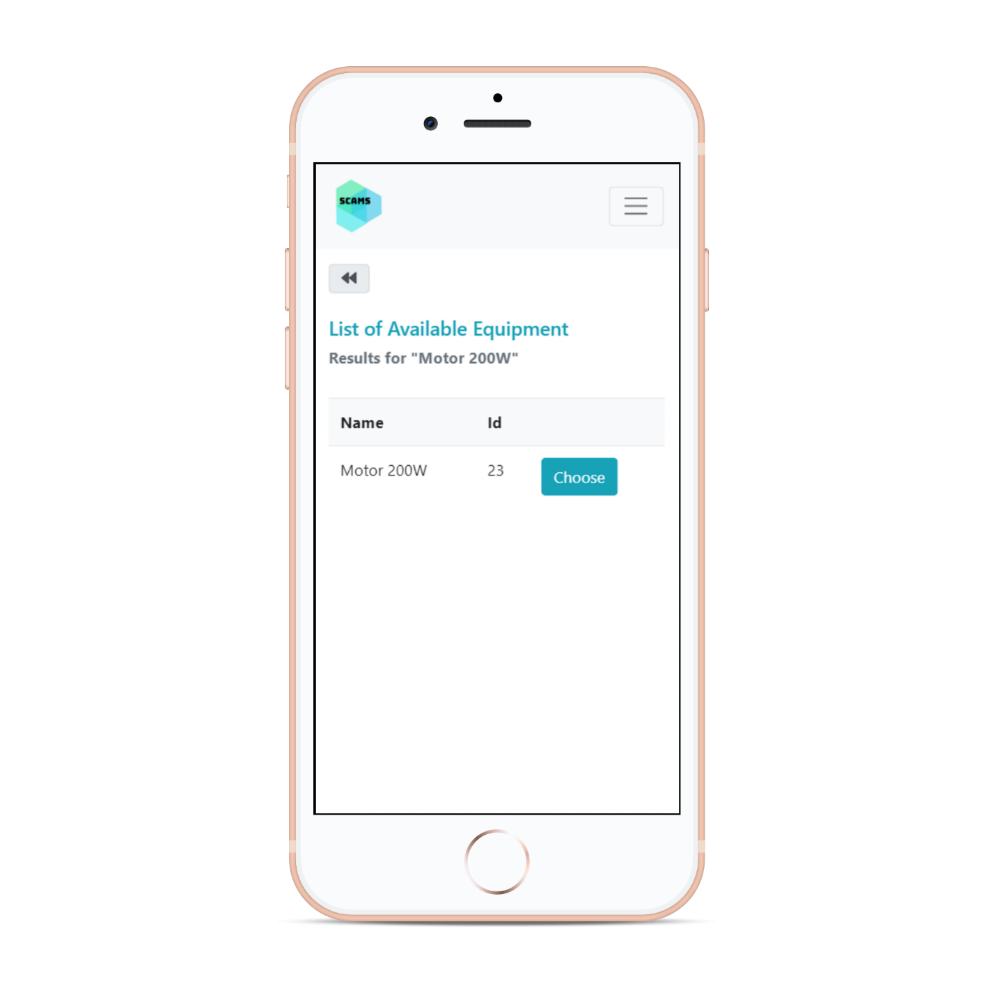
* **Model**: *<<class>>Room*, <<class>>User: This is data model which is manipulated by the controller.
* **View**: *<<class>>RoomInfo, <<class>> SearchPage:* Every view has a display() method that is called by controller to update view and show to the user.
* **Control**: *<<class>>CheckPeopleController, <<class>>RoomListController:* Controller will handle event of the user through view, and manipulate data of the model.

## 2.4. Working demonstrations

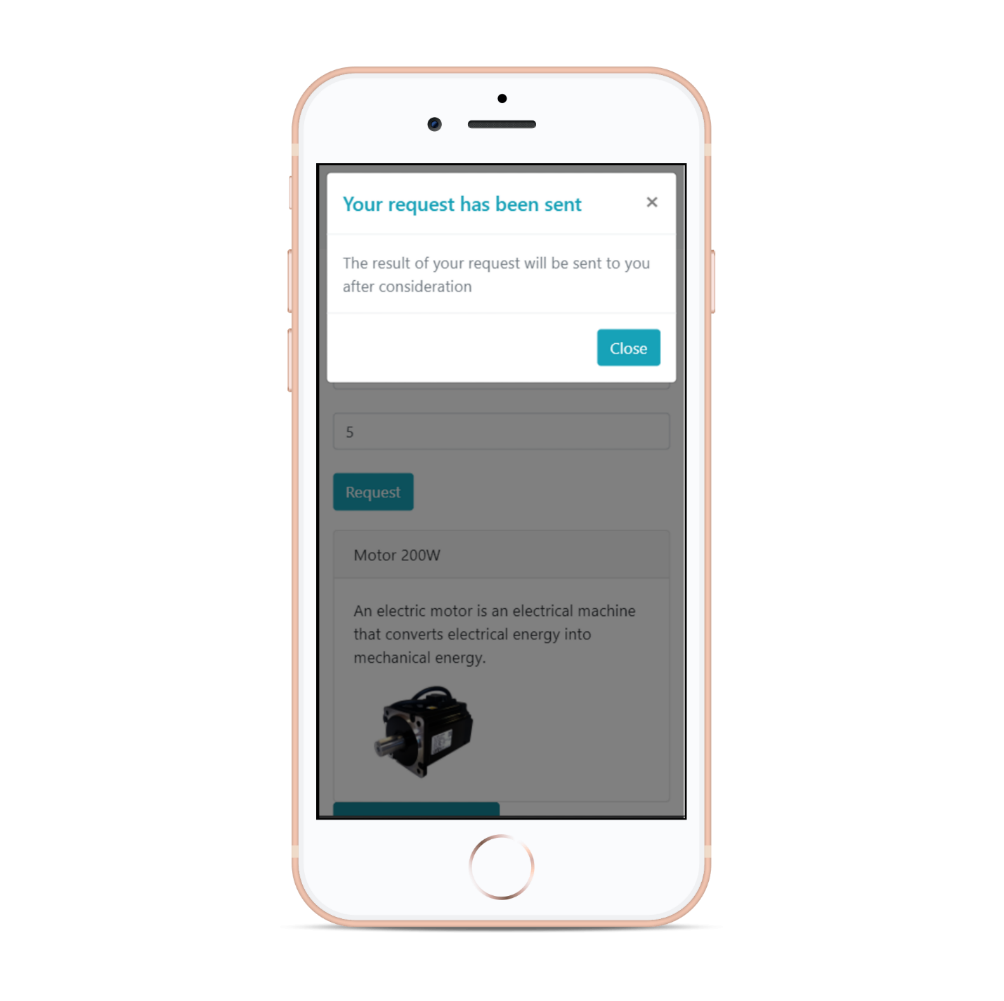
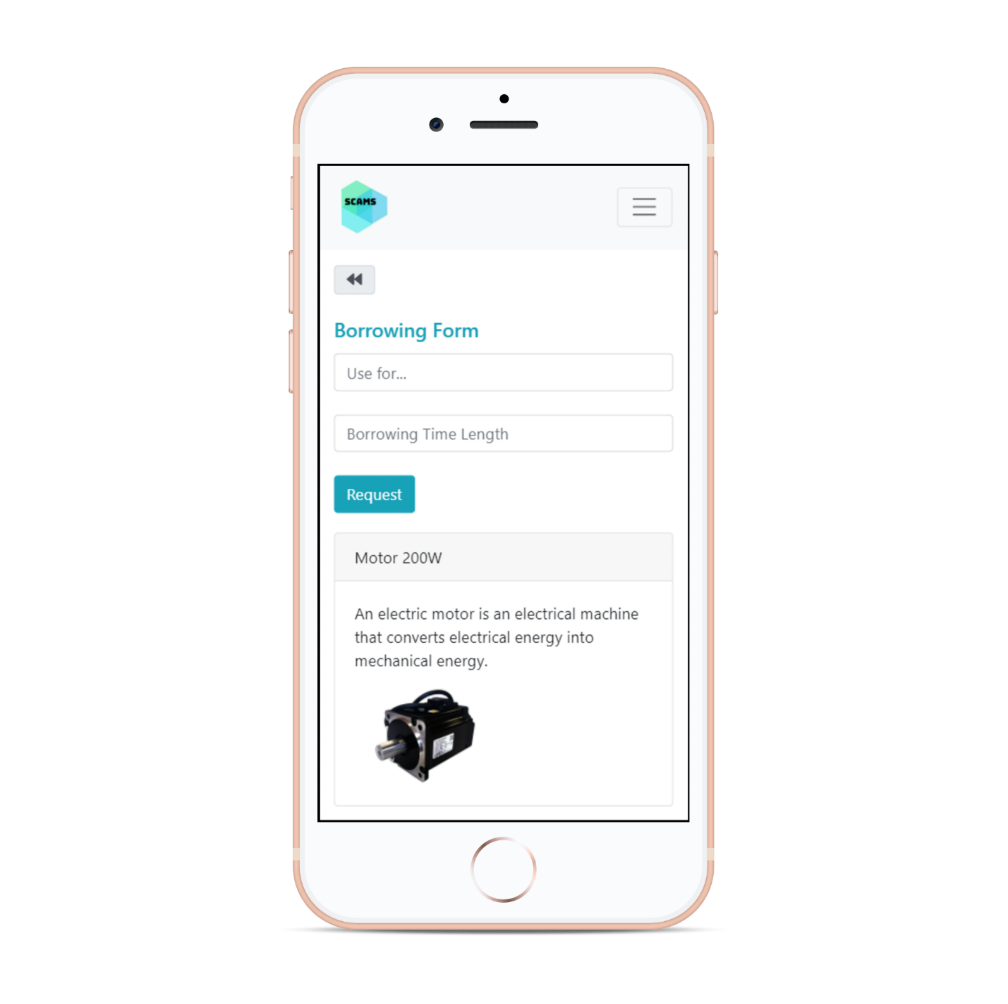
### 2.4.1. Borrow Equipment

*Figure 4.13: Home Page Figure 4.14: Menu on Home Screen*

*Figure 4.15: Search Equipment Figure 4.16: Search Equipment Result*

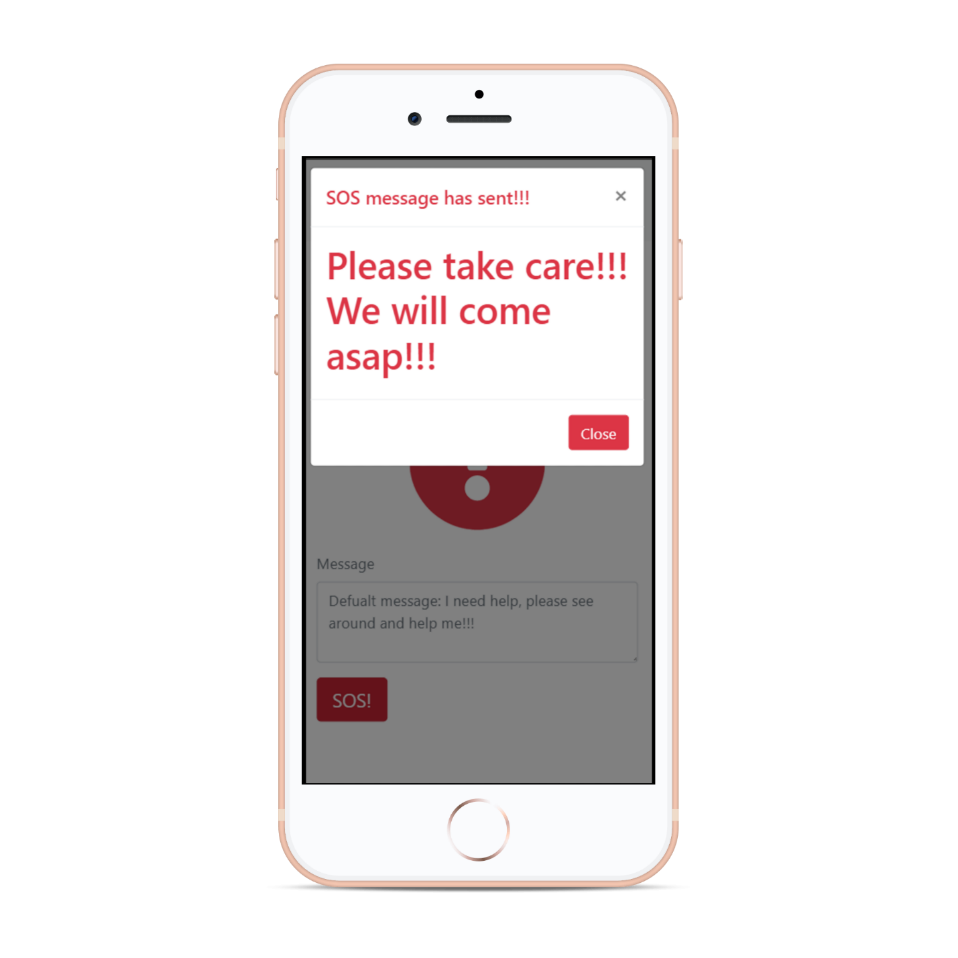


*Figure 4.17: Borrow Equipment Figure 4.18: Request sent successful*

### 2.4.2. Send SOS

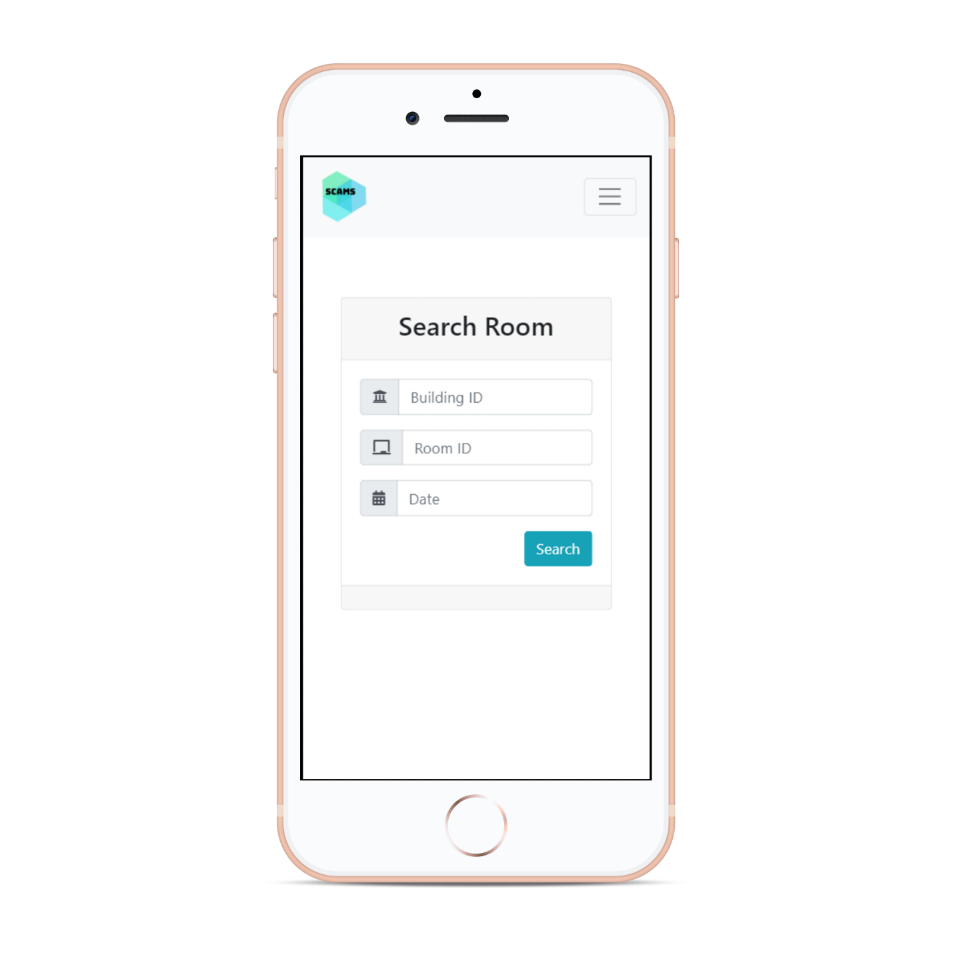
 

*Figure 4.19: From Home Page to access SOS Message*

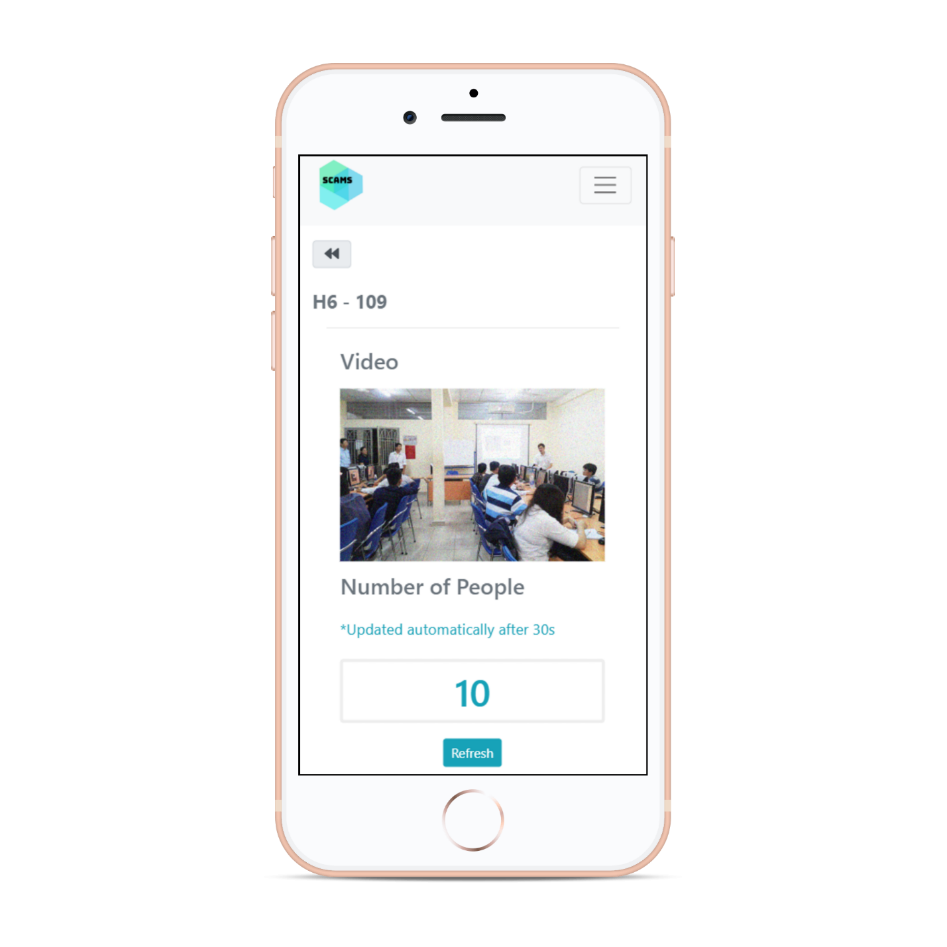
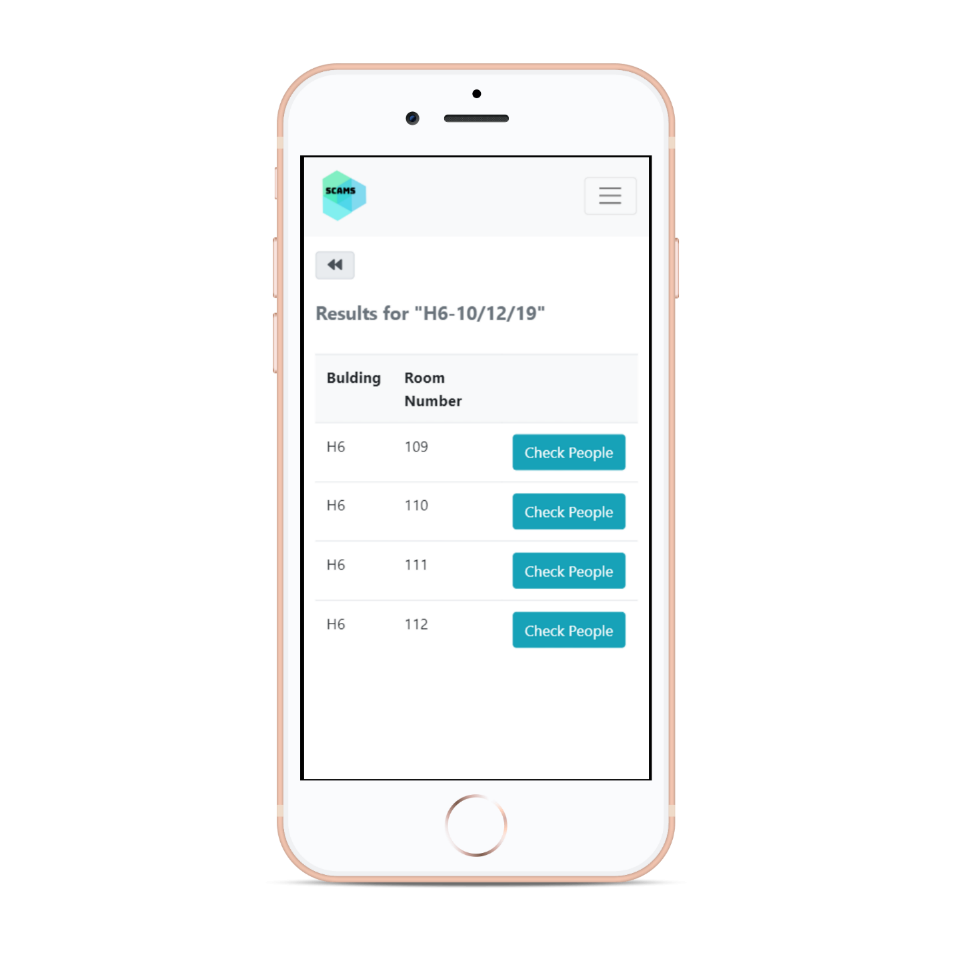
 

*Figure 4.20: SOS Page Figure 4.21: SOS sent successful*

### 2.4.3. Check Number of People

*Figure 4.21: From Home Page to access Search Room Page*



*Figure 4.22: Room List Result Figure 4.23: Check No. of People Page*

# 3. Conclusion

This report has completed tasks in assignment-4 about software design document at detail level.

# 4. Group list

|  |  |  |
| --- | --- | --- |
| Name | Student’s ID | Distribution |
| Tran Thi Ngoc Diep | 1827005 | 100% |
| Ngo Hoang Ngoc Thanh | 1827030 | 100% |
| Pham Tran Dinh Huy | 1820031 | 100% |
| Nguyen Dang Khoa | 1820034 | 100% |
| Luong Thanh Nhan | 1820047 | 100% |