Software Requirements Specification

for

Classroom Allotment System

Version 1.0

Prepared by:
Ajat Prabha (B16CS002)
Saksham Banga (B16CS042)

IIT JODHPUR

12-02-2018

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Classroom Allotment System	12-02-2018	Initial Draft	1.0
Update 1	05-03-2018	Diagram images changed, points properly numbered	

Glossary

CAS	Classroom Allocation/Allotment System
Project	CAS
User	BaseUser derivative (Professor/Admin)
class	Classroom Class instance
Database	File Handling System

1. Introduction

1.1 Purpose

The purpose of this document is to give a detailed explanation about the institute's Classroom Allotment application. This document will provide an insight to the important functions of the same and is intended for all stakeholders and developers associated with the project. It also highlights the necessary constraints, dependencies and other minor details associated specifically with this version of the application.

1.2 Product Scope

The scope of this software is within the Institute to reserve classrooms for lectures, seminars, etc. It is designed to decrease the time required to take permissions etc from the authorities which is usually done by taking handwritten paperwork etc.

1.3 References

1. IEEE Software Requirement Specification Template
2. UML @ Classroom
3. StarUML documentation

2. Overall Description

2.1 Product Perspective

It is a standalone project for classroom request handling intended for professors and administration staff members where a professor can request any empty classroom for a particular time slot and also check whether the class is approved or not.

2.2 Product Requirements

The functions(abstract) are listed below:

1. Professor/Admin account creation(registration).
2. Professor/Admin account deletion.
3. Professor/Admin modify details.
4. Professor can apply for a particular time slot and look at available classrooms in that slot.
5. Professor can also modify an existing slot.
6. Admin can choose to approve or decline the slot requested.
7. After request is approved, professor can view all the approved requests in his profile.

2.3 User Classes and Characteristics

There are typically 3 User Classes which will be used for this project

I. **BaseUser**(Abstract)-No direct objects of BaseUser to be created(ie No entity such as a baseUser exists independently). It contains all the attributes common to all classes inheriting this class(Professor and Admin).
II. **Professor**(inherits BaseUser)-A class based representation of a Professor. Contains attributes like department etc.
III. **Admin**(inherits BaseUser)- A class based representation of an Administration staff member having the power to accept/decline the requests.

2.4 Operating Environment

This program is intended to work on command line interface such as cmd, bash or powershell with a basic compiler(gcc) installed on the computer.

2.5 Assumptions, Dependencies and Constraints

→ **Assumptions**:

It is assumed that at a given time, no 2 users are logged in and using the filesystem.
All Professors are assumed to be at the same status in terms of priority of requests.
All admin staff are assumed to be of the same authority

→ **Dependencies**:

Any computer with a compiler(gcc) installed and a default command line interface to run the compiler is required.

→ **Constraints:**

Instead of a dedicated Relational Database Management System(RDBMS), file handling is used

3. External Interface Requirements

3.1 Hardware Interfaces

Keyboard: User Input to be given by a standard keyboard
Video Display Unit(VDU): Program output to be displayed on stdout source chosen by the computer.

4. Other Nonfunctional Requirements

4.1 Performance Requirements

It is required that at a given time, only 1 user edits the file so that sequential execution of requests is possible. If multiple instances running the same program are opened, it might create problems since simultaneous updation is not possible without making the code distributed.

4.2 Security Requirements

No user can view the classroom details without being in a logged in state. This ensures safety of data and offers accountability for every decision made by the code.

4.3 Software Quality Attributes

The code structure has the following quality attributes:

- Adaptability
- Robustness
- Maintainability
- Portability
- Reusability

4.4 Business Rules

→ An admin has superuser level of access in this project. The admin can approve/decline requests put forward by the professor.
→ A Professor is a client access level which can view and request slots for a particular time and classroom type. More priority is given towards successful allotment of classes for a given time rather than choosing a specific classroom for a given time slot

5. Other Requirements

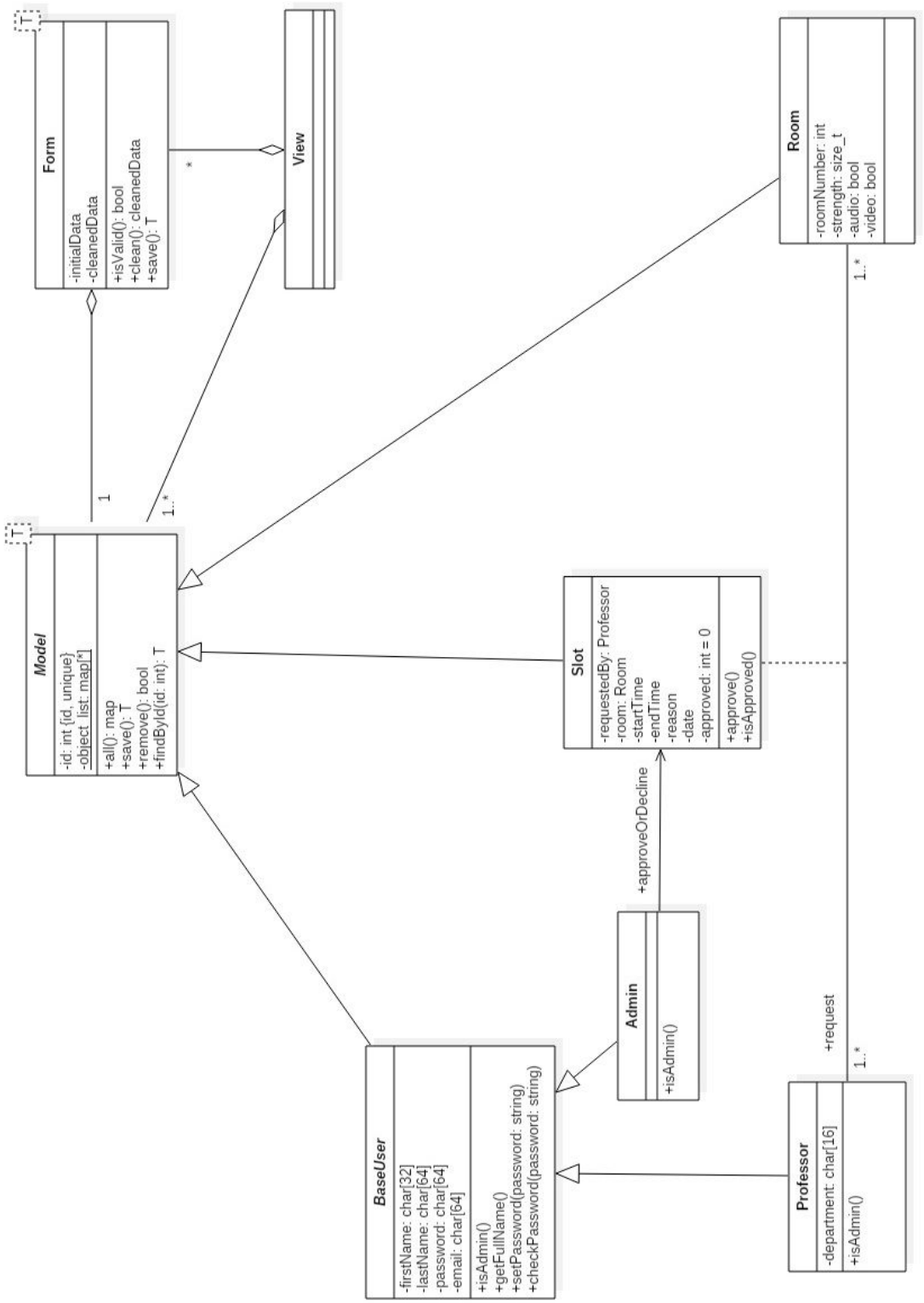
→ File Handling is required in order to simulate a database.

6. Analysis Models

Following is a list of diagrams kept in mind for this project
Note: For activity diagrams of the below use cases, refer to Appendix A.

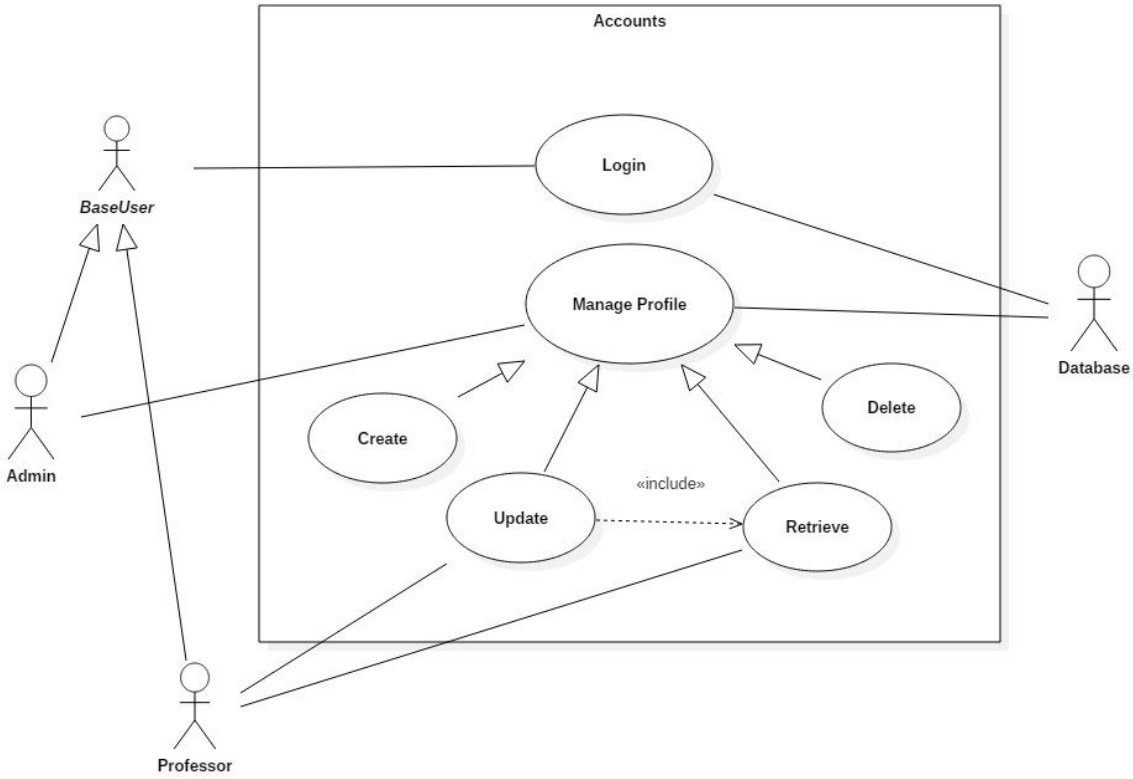
Specific Requirements

CLASS DIAGRAM



Use Case Specifications

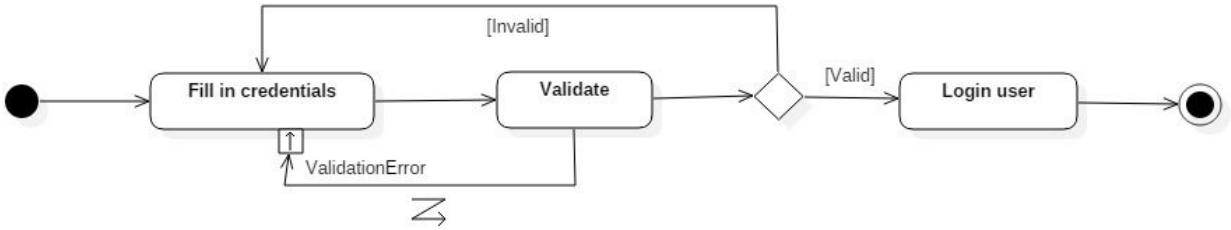
Accounts

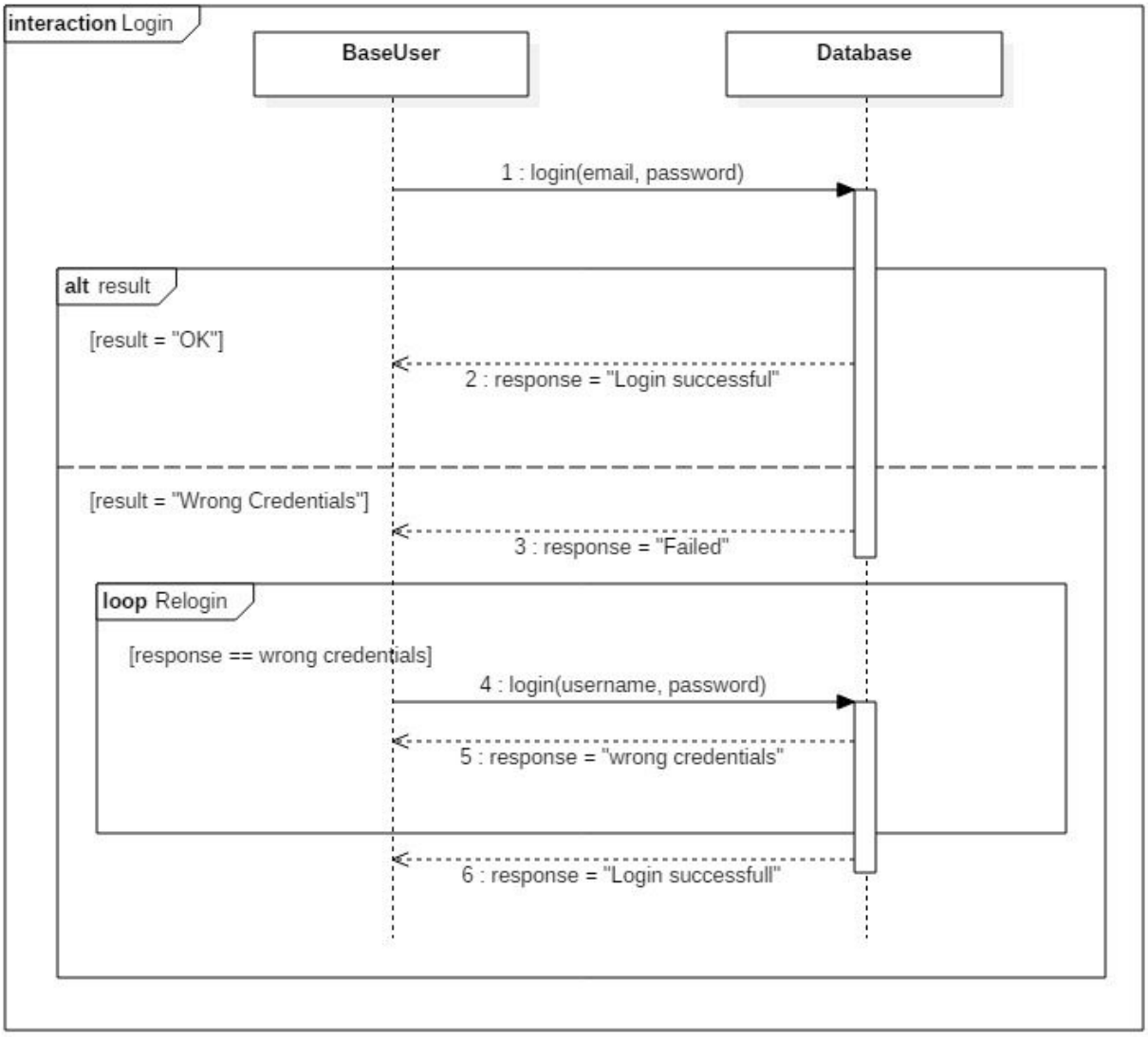


1.Login

Use case Name	User Login
Actor	BaseUser, Database
Trigger	User fills the Login credentials.
Precondition	The User must be registered.
Basic Path	1. User enters its user ID and password into the system's login screen. 2. System appropriately selects BaseUser type(Professor/Admin).
Alternative Path	No alternative path.
Post Condition	The user is taken to his profile page.

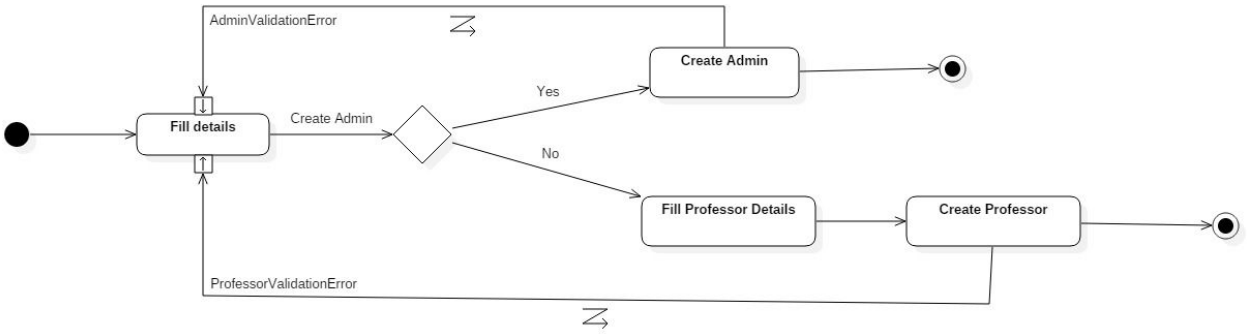
Exception Paths	All fields must be filled.
Others	None

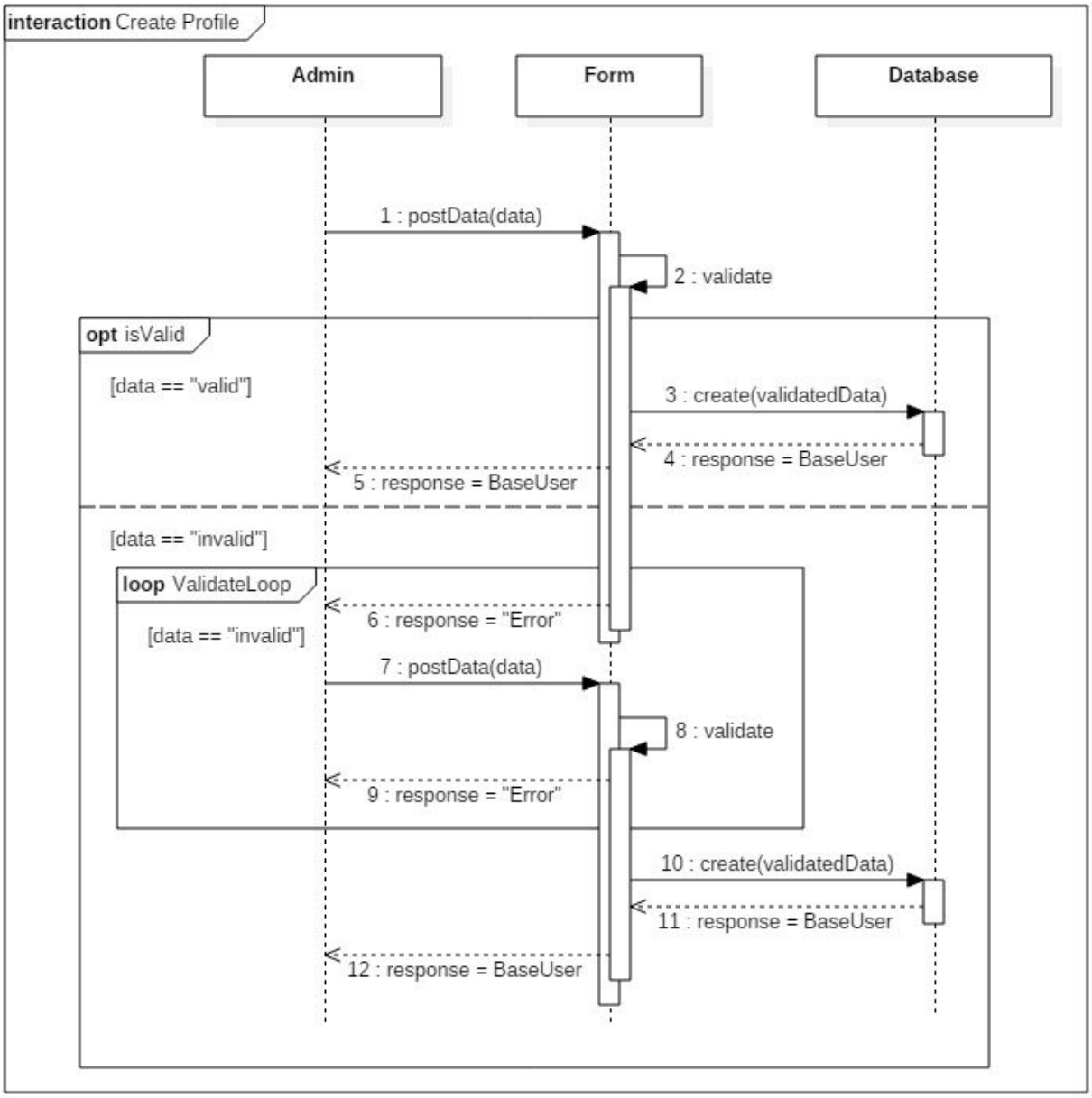




2. Create Account

Use case Name	Create
Actor	Admin, Database
Trigger	Admin chooses register new user option in his profile
Precondition	The User must be logged in and it should be of the admin type.
Basic Path	1. Admin enters the details of the new account(professor/admin) 2. Form validation is done, and if invalid admin must fill it again until it is correct. 3. After a correct form filling is done, the data is sent to the database where a new BaseUser object is created and returned.
Alternative Path	No alternative path
Post Condition	The user is created successfully.
Exception Paths	All parts of the form must be filled.
Others	None

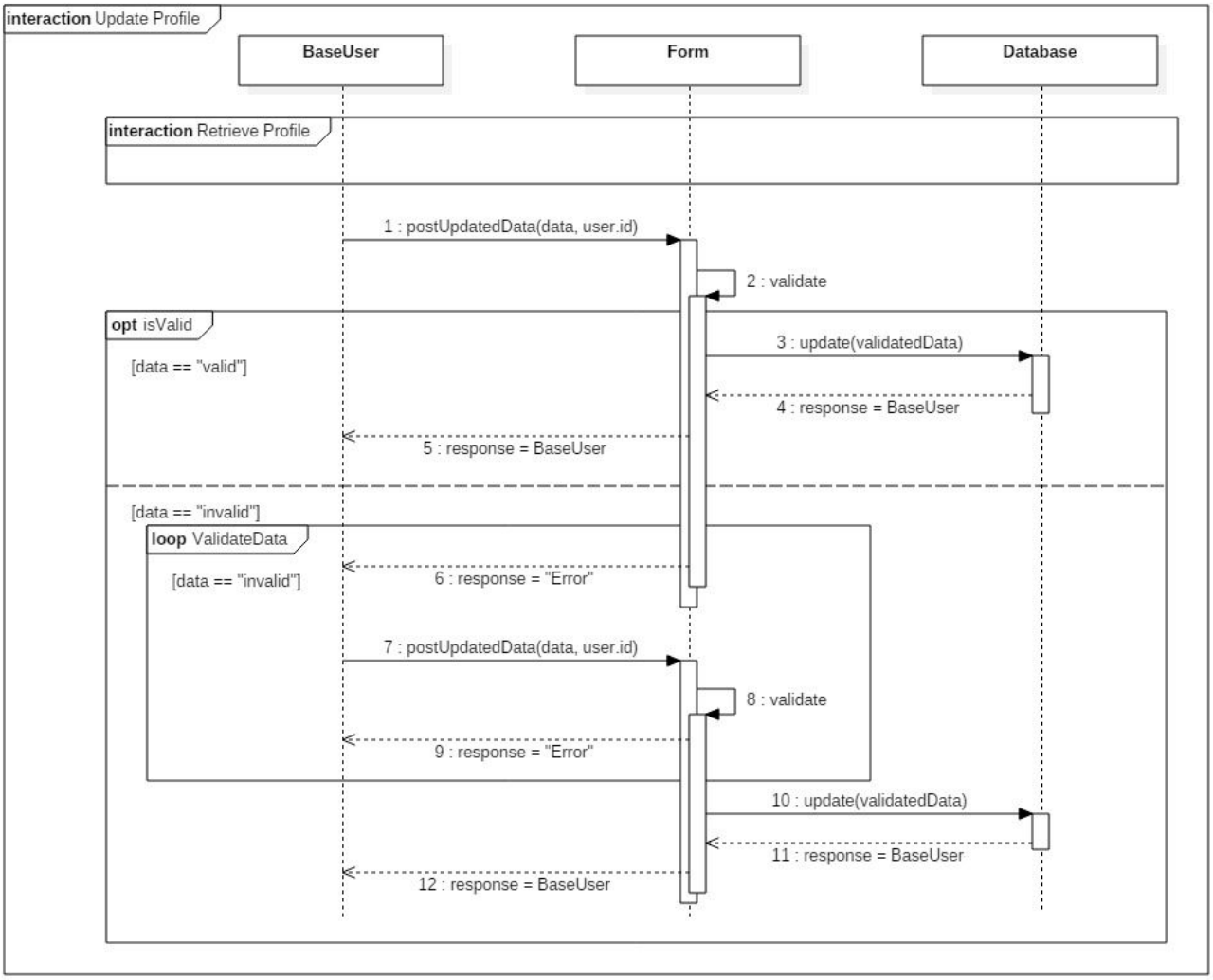




3. Update account

Use case Name	Update
Actor	BaseUser, Database
Trigger	The user chooses update profile from his/her profile options.
Precondition	The BaseUser must be logged in appropriately.

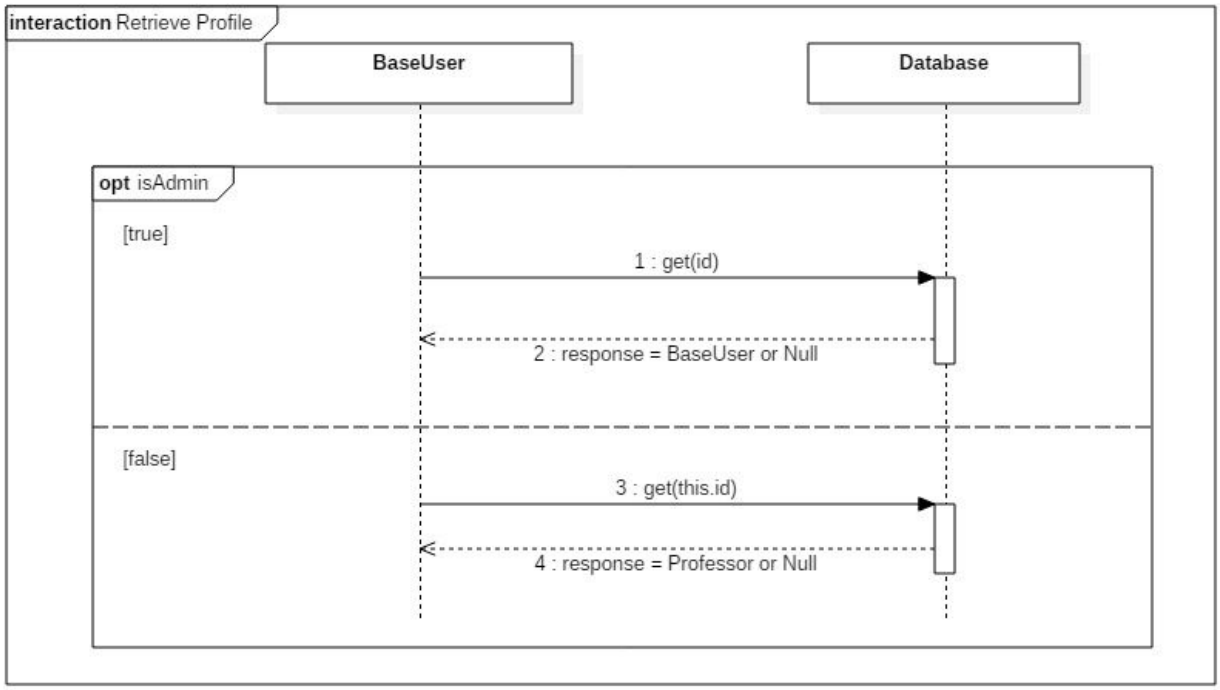
Basic Path	1. i. If the current logged in BaseUser is an admin, he needs to identify the BaseUser object to be changed whose current details are then displayed. 2. Then admin enters the updated details of the BaseUser, which he needs to update. He can't change details of another admin. 3. ii. If the current logged in BaseUser is a Professor, he needs to enter the updated details for himself. 4. Form validation is done, and if invalid admin must fill it again until it is correct. 5. After a correct form filling is done, the data is sent to the database where the BaseUser object is updated and returned.
Alternative Path	No alternative path
Post Condition	The user is taken (back) to his profile page.
Exception Paths	All fields must be filled.
Others	None



4.Retrieve Account Details

Use case Name	Retrieve
Actor	BaseUser, Database
Trigger	BaseUser chooses Retrieve details options from profile menu.
Precondition	The BaseUser must be appropriately logged in.
Basic Path	1. If the BaseUser is a professor, the database simply returns the current logged in object which also contains the details of the professor. 2. If the BaseUser is an admin, they identify the object whose information needs to be retrieved. The

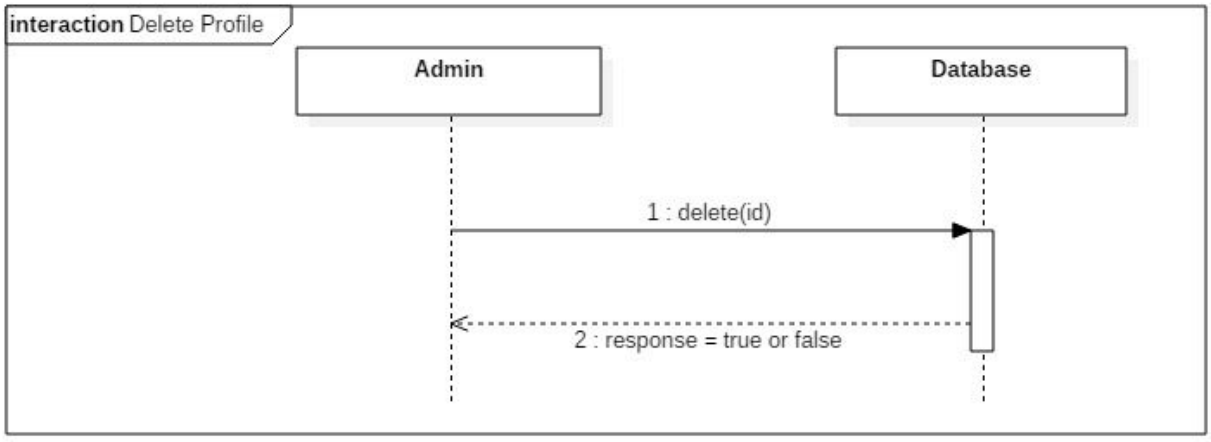
	database returns appropriate object with the details 3. The details are then printed.
Alternative Path	No alternative path.
Post Condition	The BaseUser is taken to back to his profile page.
Exception Paths	All fields must be filled.
Others	None



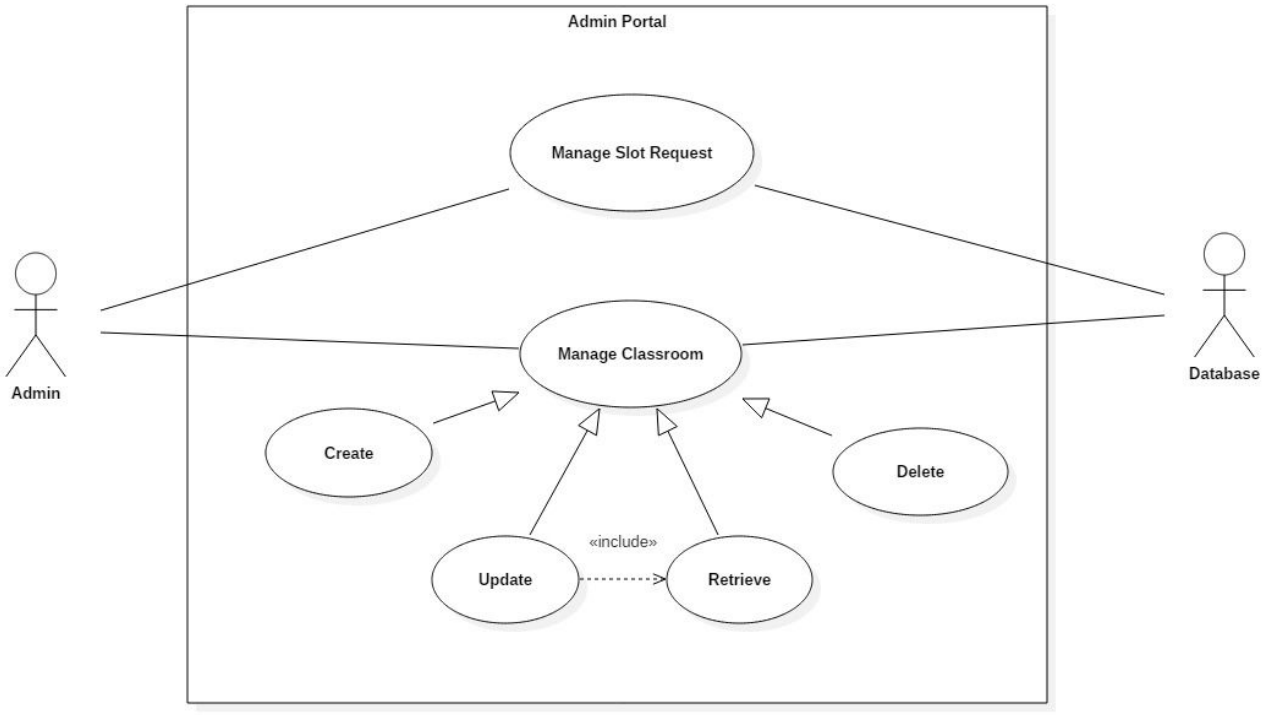
5.Delete Account

Use case Name	Delete
Actor	Admin, Database
Trigger	Admin chooses the delete account option from profile menu.
Precondition	The Admin must be logged in appropriately.

Basic Path	1. Admin identifies the BaseUser whose account needs to be deleted. 2. The database then deletes the object corresponding to the particular BaseUser's account. An admin cannot delete himself/herself and another admin can delete some admin object.
Alternative Path	No alternative path.
Post Condition	The user is taken back to his profile page.
Exception Paths	All fields must be filled.
Others	None



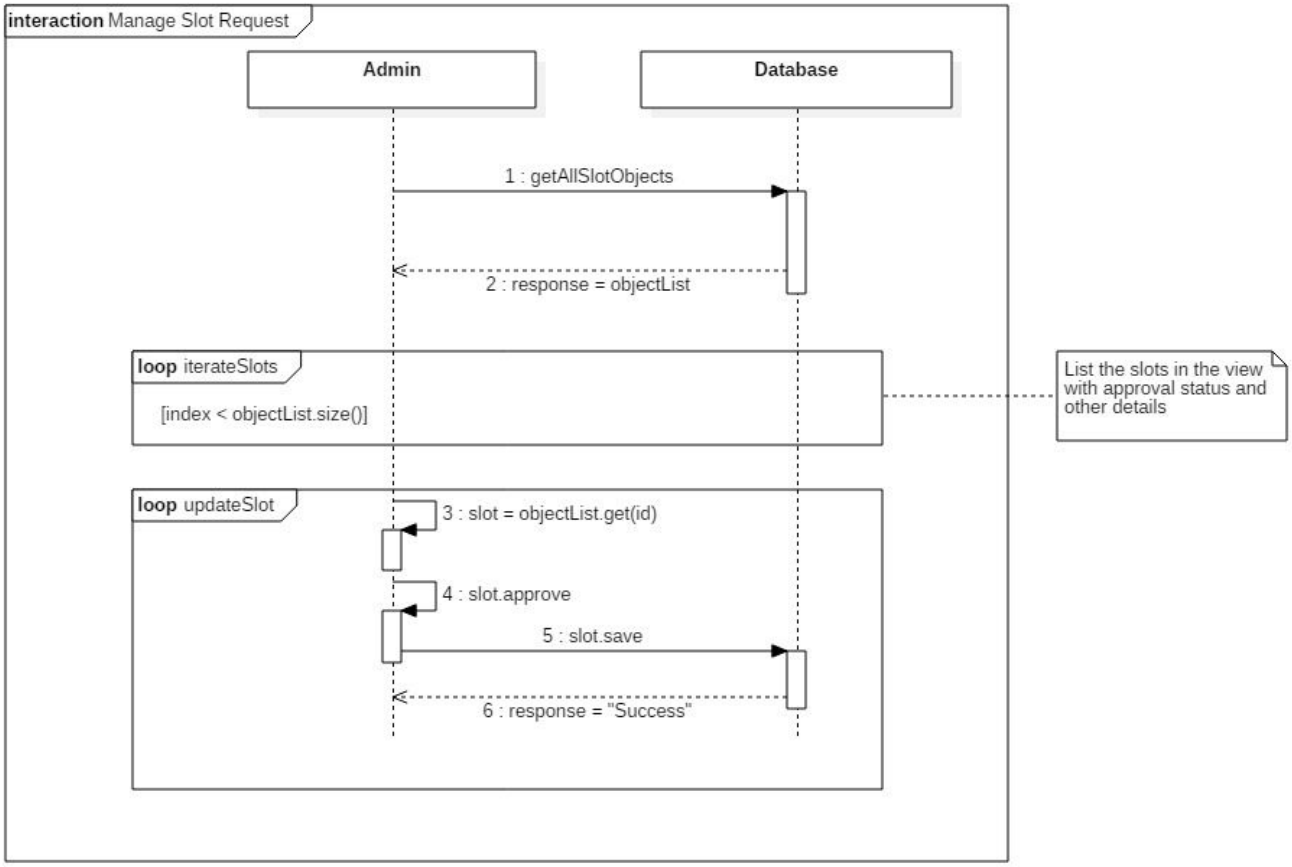
Admin Portal



1. Manage Slot Requests

Use case Name	Manage Slot Requests
Actor	Admin, Database
Trigger	Admin chooses show notifications option from profile menu.
Precondition	The admin must be appropriately logged in.
Basic Path	1. Database returns a list of all Slots(approved and unapproved). Only the ones which are not approved and are new are shown. 2. Now the admin has the power to approve/disapprove a particular slot. 3. If the slot is saved successfully, a success response is received by the admin

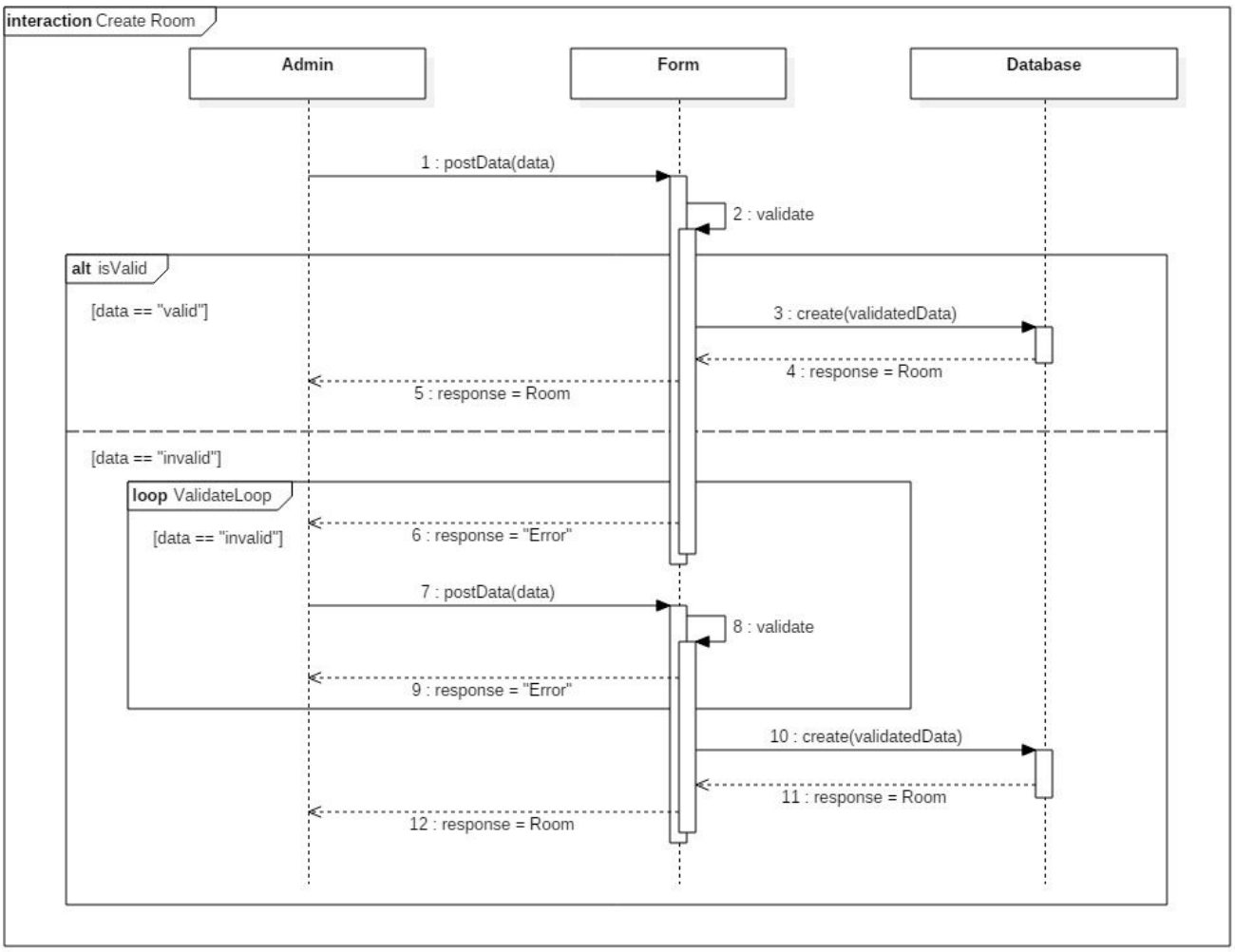
Alternative Path	No alternative path.
Post Condition	The admin is taken back to his profile page.
Exception Paths	All fields must be filled.
Others	None



2. Create Classroom

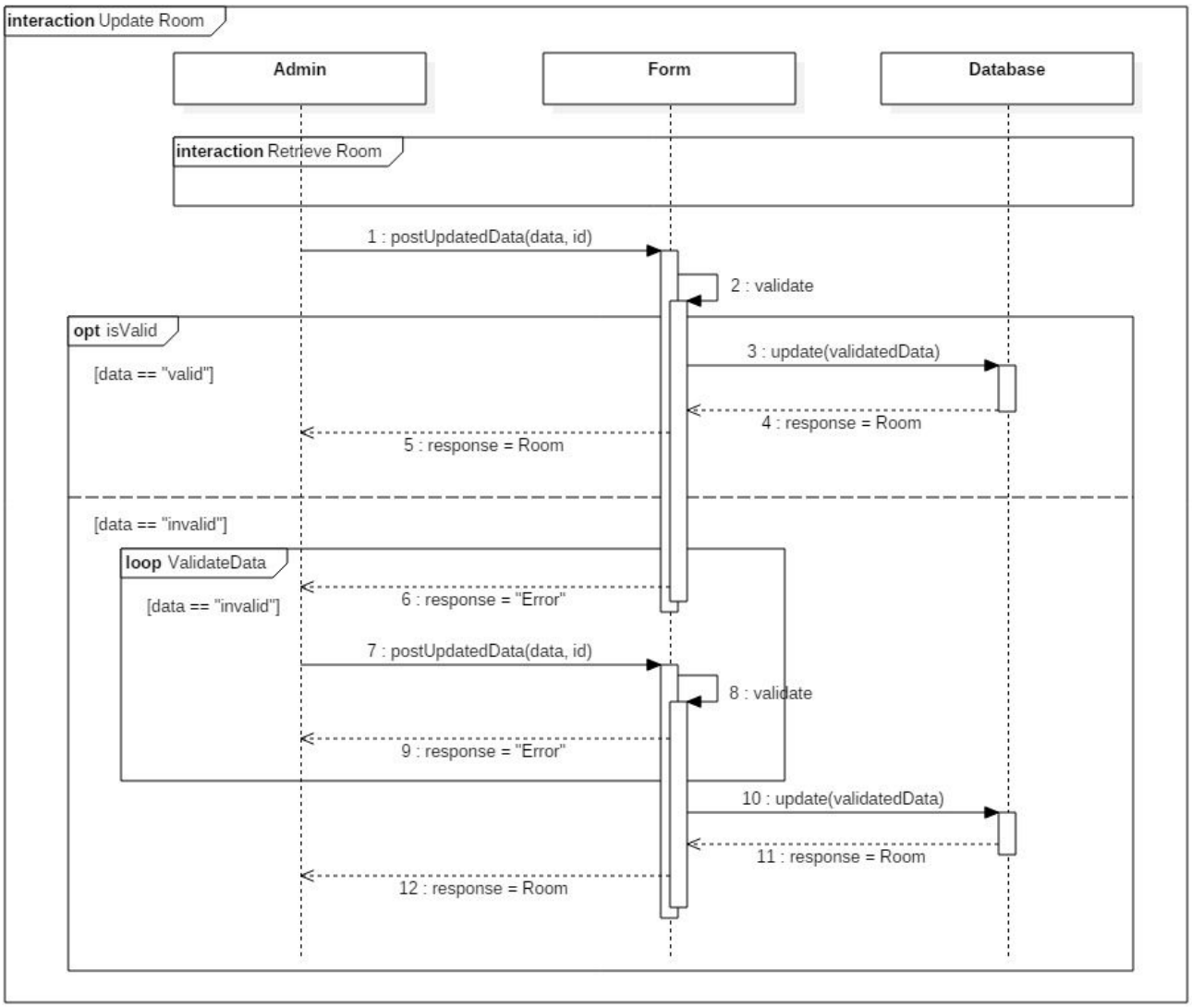
Use case Name	Create
Actor	Admin, Database
Trigger	Admin chooses create new class option from profile menu.
Precondition	The admin must be appropriately logged in.

Basic Path	1. Admin fills data related to new classroom. 2. Data is validated by the Form class and until data is entered according to correct format, Form class iterates and prompts for correct data to be entered. 3. Once valid data is sent to the Database, a Room object is created and returned with the same attribute details as recorded above.
Alternative Path	No alternative path.
Post Condition	The admin is taken back to his profile page.
Exception Paths	All fields must be filled.
Others	None



3. Update Class details

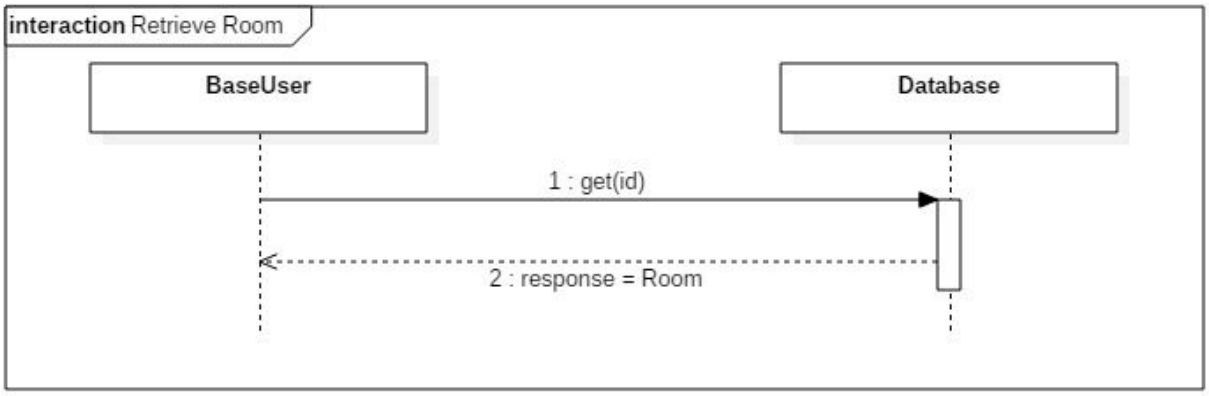
Use case Name	Update
Actor	Admin, Database
Trigger	Admin chooses update classroom option from profile menu.
Precondition	The admin must be appropriately logged in.
Basic Path	1. The room to be updated is identified and the corresponding object is retrieved. 2. Updated data once filled by the admin is validated by the Form class and once validated is sent to the Database. 3. The Database makes the changes in the object and returns an updated object of the classroom updated.
Alternative Path	No alternative path.
Post Condition	The admin is taken back to his profile page.
Exception Paths	All fields must be filled.
Others	None



4. Retrieve Classroom Information

Use case Name	Retrieve
Actor	Admin, Database
Trigger	Admin chooses show class details option from profile menu.
Precondition	The admin must be appropriately logged in.
Basic Path	1. Admin identifies the class whose information is to be retrieved. If no such class exists, a null object is returned. 2. However if the class exists, Database returns an

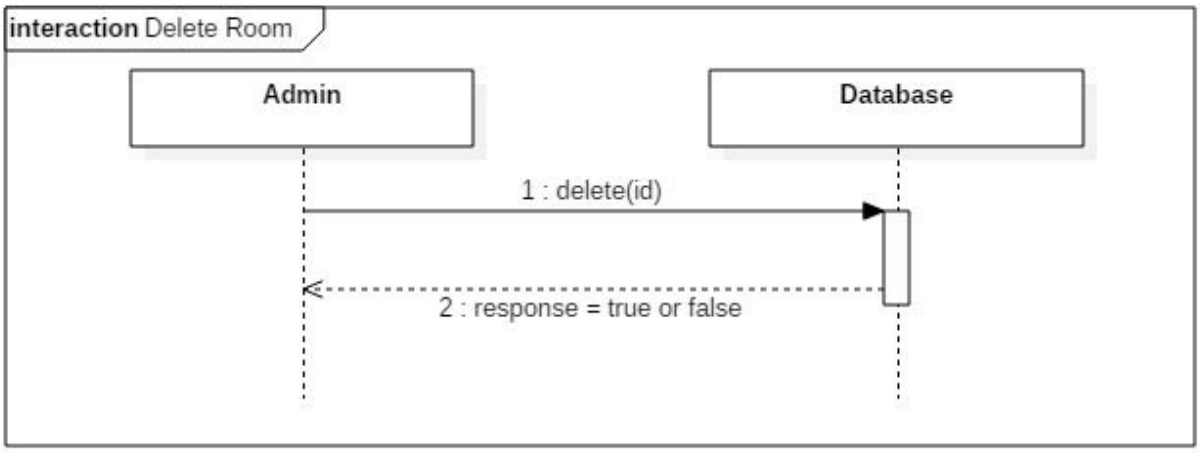
	object of the corresponding class. 3. The details related to that class are then printed.
Alternative Path	No alternative path.
Post Condition	The admin is taken back to his profile page.
Exception Paths	All fields must be filled.
Others	None



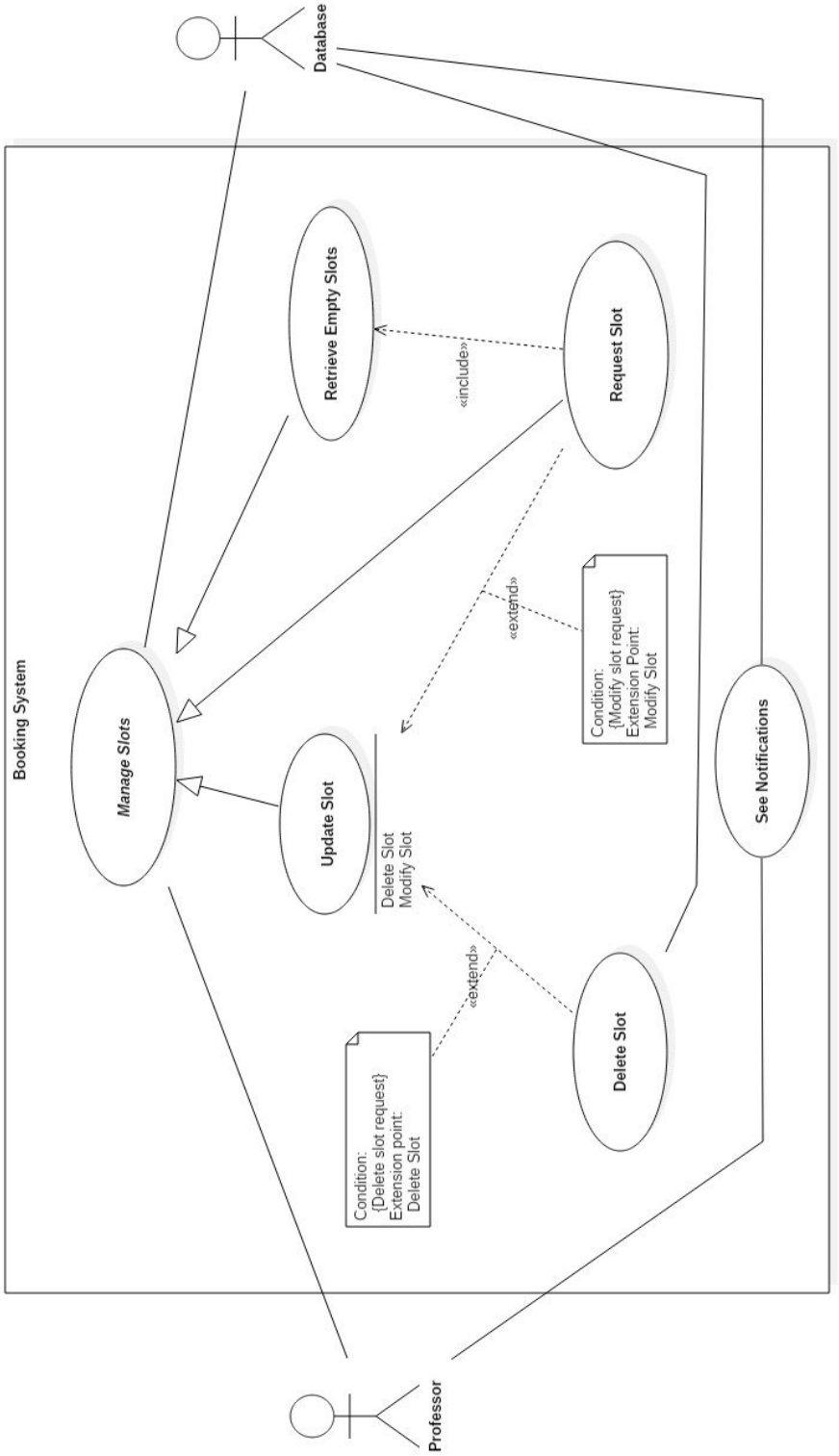
5. Delete Classroom

Use case Name	Delete
Actor	Admin, Database
Trigger	Admin chooses delete classroom option from profile menu.
Precondition	The admin must be appropriately logged in.
Basic Path	1. The admin identifies the class to be deleted. 2. If the class is successfully identified, Database deletes it and sends a success(true) response. If the classroom does not exist, a failure(false) response is sent.
Alternative Path	No alternative path.
Post Condition	The admin is taken back to his profile page.
Exception Paths	All fields must be filled.

Others	None

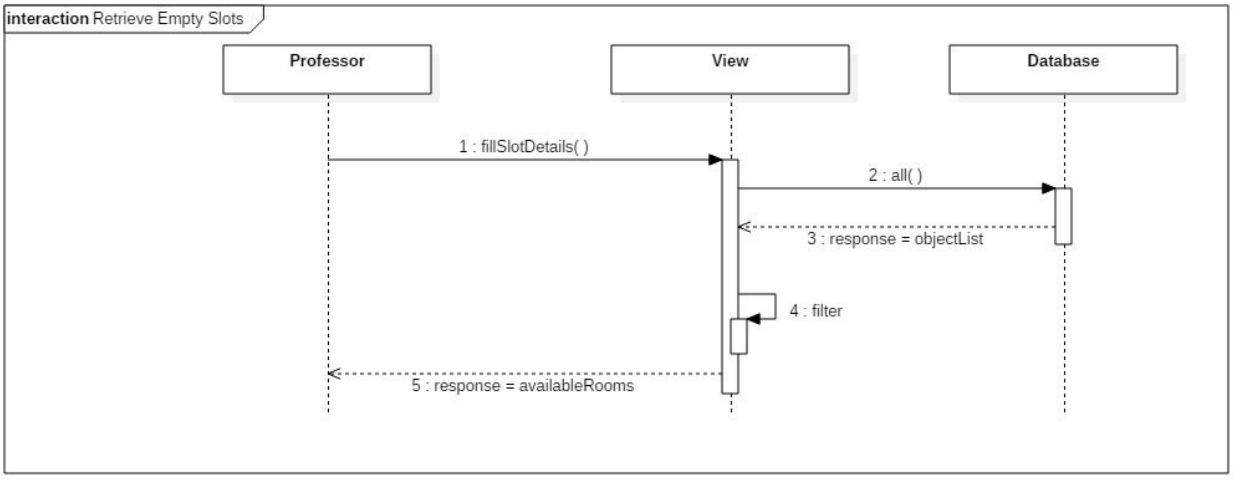


Booking System



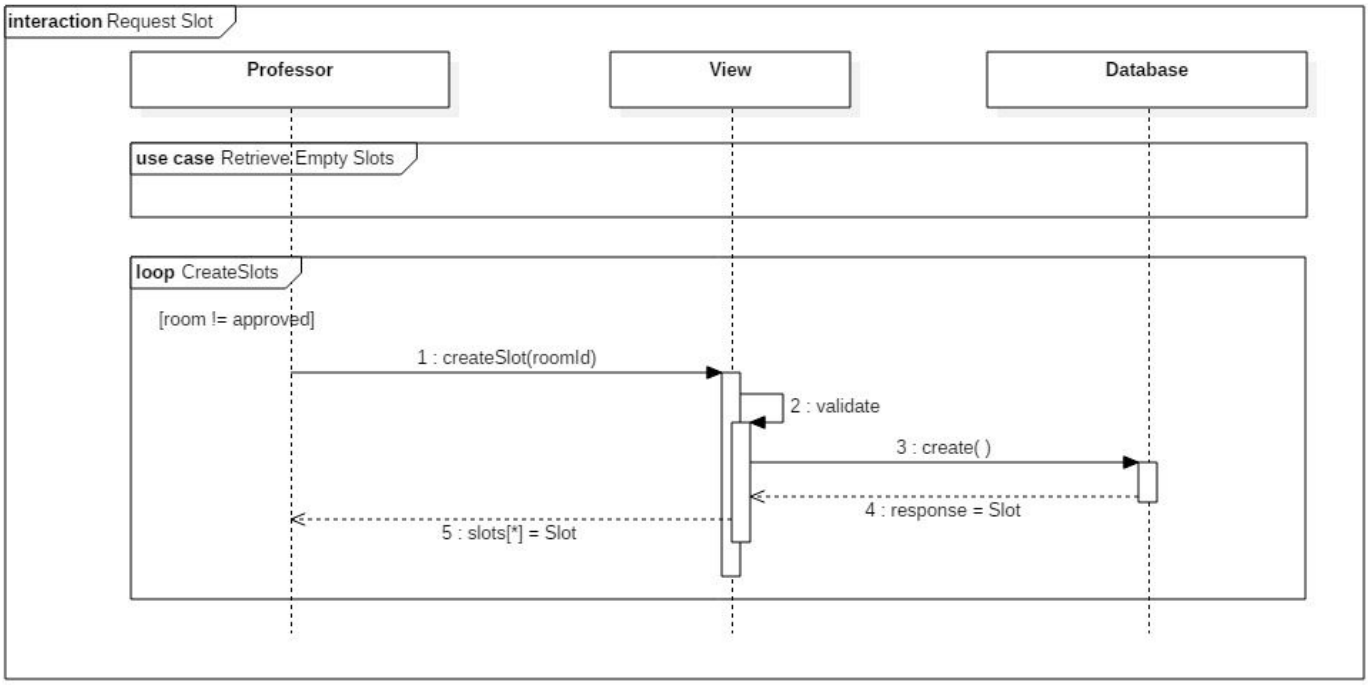
1. Retrieve Empty Slots

Use case Name	Retrieve Empty Slots
Actors	Professor, Database
Trigger	Professor chooses show empty classes option.
Precondition	The professor must be appropriately logged in.
Basic Path	1. Professor gives a time slot for which he wants to enquire about. 2. Database returns all slot requests(approved and unapproved) along with all classroom objects. 3. The free classes are the ones in the list which do not have any slot(approved) against them which intersects with the queried timeslot. 4. The free classroom list is then returned to the professor.
Alternative Path	No alternative path.
Post Condition	The Professor is taken back to his profile page.
Exception Paths	All fields must be filled.
Others	None



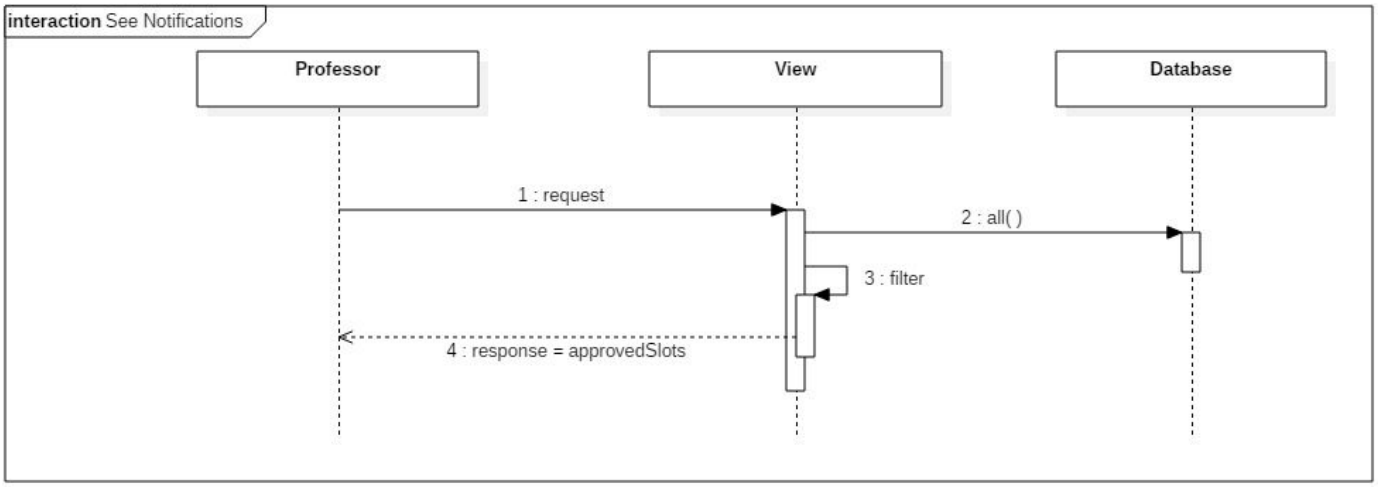
2. Request Slot

Use case Name	Request Slot
Actor	Professor, Database
Trigger	Professor chooses the request a class option from the profile menu.
Precondition	The professor must be appropriately logged in.
Basic Path	1. The professor enters the date and time for which he needs a classroom along with other details(strength, AV needed or not) 2. Retrieve Empty Classroom use-case is thus invoked which returns a list of the empty classrooms for that given slot. 3. The Professor then chooses if he wants to view another timeslot or requests current slot. 4. If the current slot is requested, the Database creates a new Slot object corresponding to each class which is free. 5. All the Slots are initially in the unapproved state(new).
Alternative Path	No alternative path.
Post Condition	The Professor is taken back to his profile page.
Exception Paths	Date is requested until a valid date is entered.
Others	None



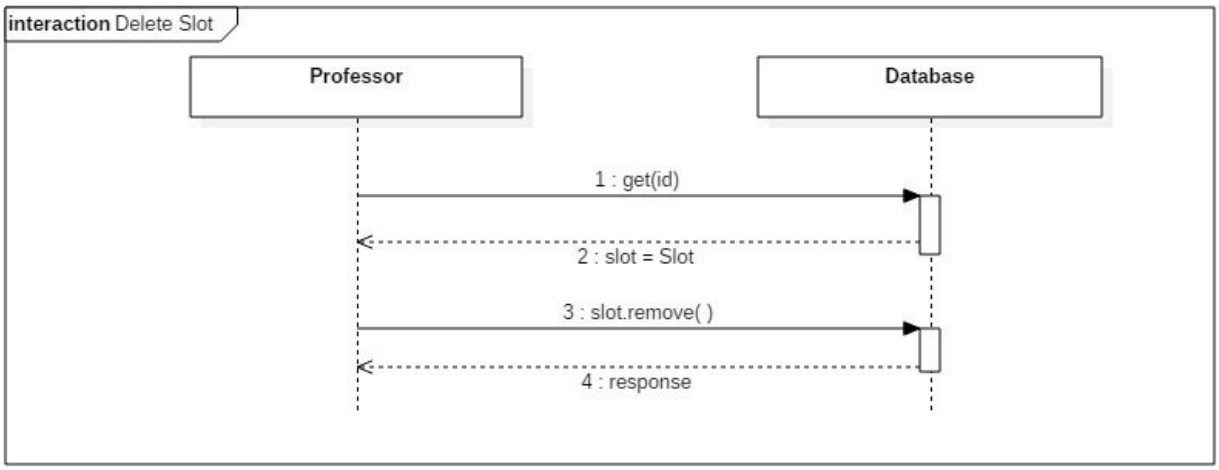
3. See Notifications

Use case Name	See Notifications
Actor	Professor, Database
Trigger	Professor chooses show notifications option from profile menu.
Precondition	The professor must be appropriately logged in.
Basic Path	1. Database returns a list of all Slot objects associated with the currently logged in professor 2. All the slots associated to the currently logged in professor are printed if they are in approved state.
Alternative Path	No alternative path.
Post Condition	The Professor is taken back to his profile page.
Exception Paths	All fields must be filled.
Others	None



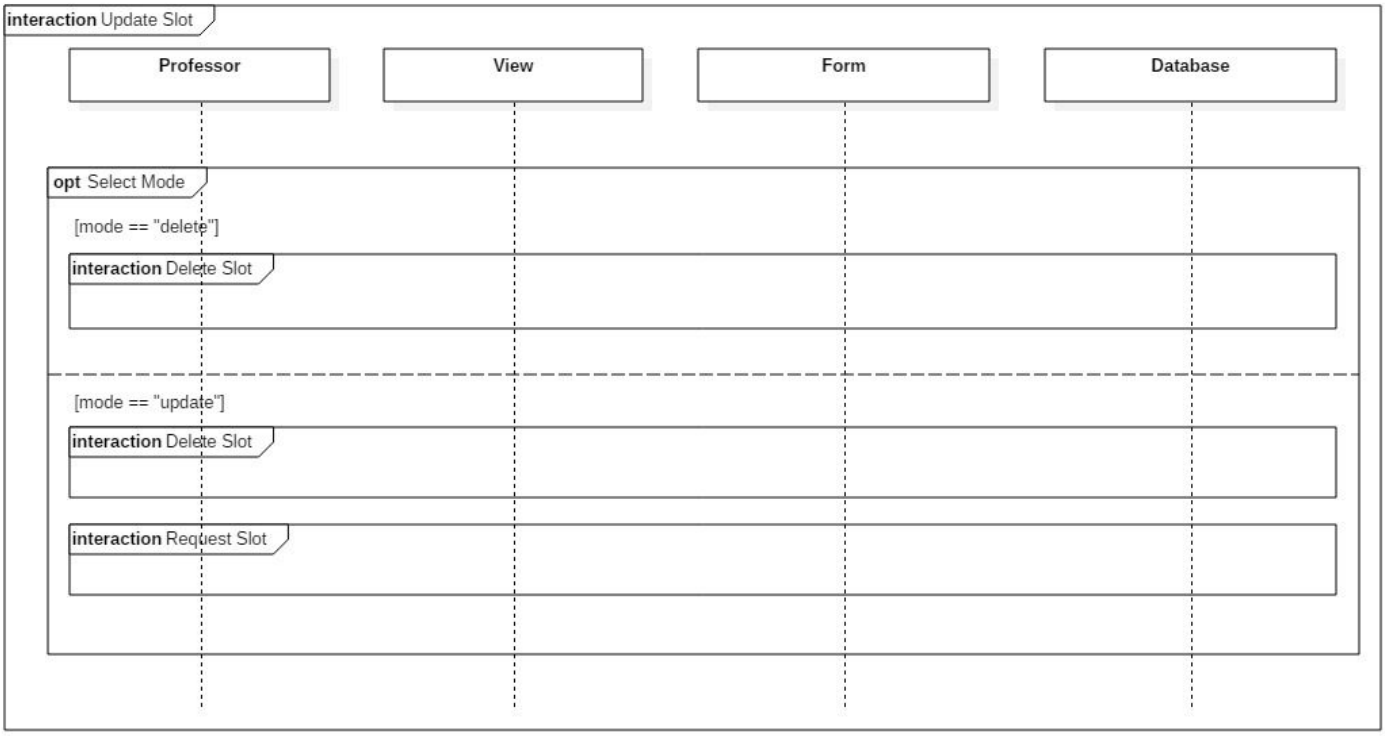
4. Delete Slot Request

Use case Name	Delete
Actor	Professor, Database
Trigger	Professor chooses Delete Slot option from profile menu.
Precondition	The Professor must be appropriately logged in.
Basic Path	1. Professor enters the slot he would like to delete 2. The slot object is then deleted from the Database
Alternative Path	No alternative path.
Post Condition	The Professor is taken back to his profile page.
Exception Paths	The slot to be deleted is valid
Others	None



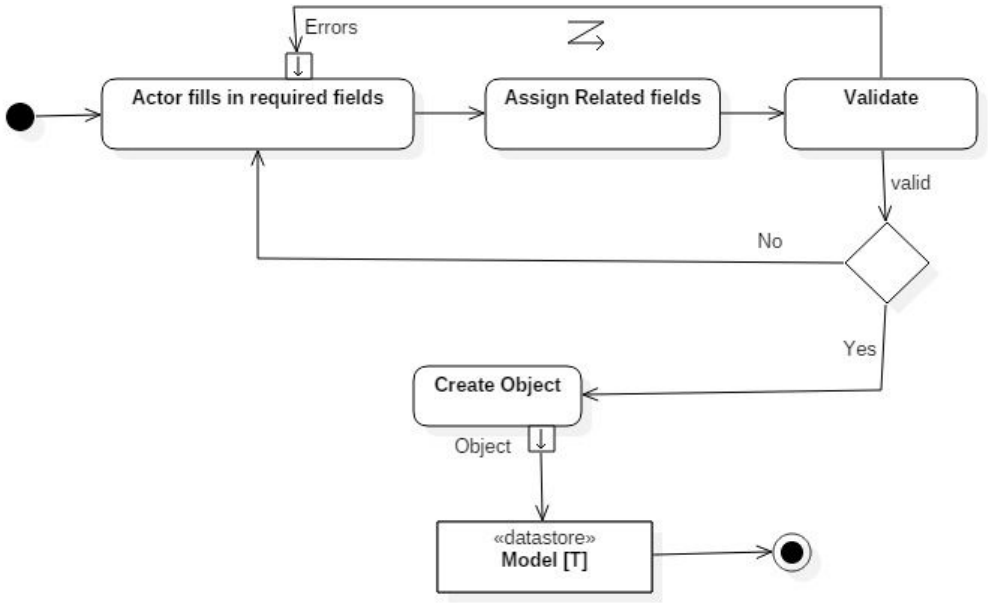
5. Update Slot Requests

Use case Name	Update Slot Request
Actor	Professor, Database
Trigger	Professor chooses Update/Delete Slot option from profile menu.
Precondition	The Professor must be appropriately logged in.
Basic Path	1. Database returns a list of all the Slot objects. 2. A list of those objects which are associated with the currently logged in class is made. 3. Professor chooses to Delete or Modify the details of a slot. 4. The Delete slot request use-case is now invoked for the selected slot 5. If the request was to update slot, the request new slot use-case is invoked and a new slot object is created.
Alternative Path	No alternative path.
Post Condition	The Professor is taken back to his profile page.
Exception Paths	The slot to be updated must be a valid slot.
Others	None

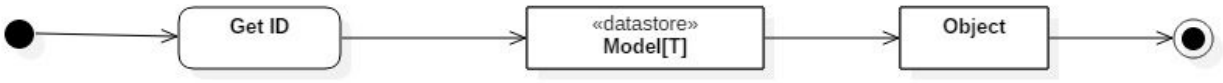


Appendix A: CRUD based Generic Activity Diagrams

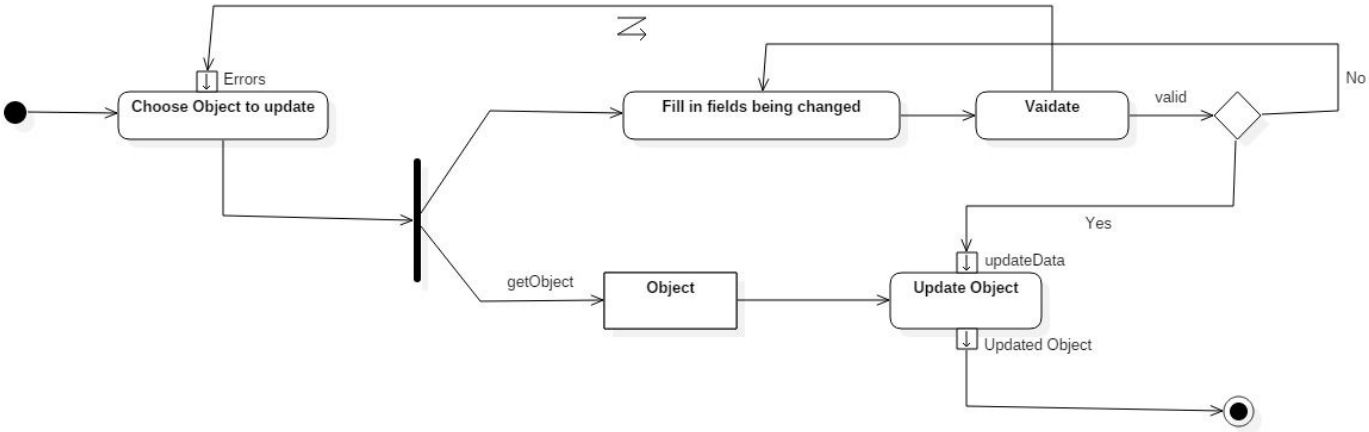
1. Create Activity Diagram



2. Retrieve Activity Diagram



3. Update Activity Diagram



4. Delete Activity Diagram