Software Requirements and Design Document

for

SocioCult

(Society Management System)

Fatima Mustafa 20i0564

Haissam Bhaur 20i0445

11/27/2022

Table of Contents

1. Introduction

1.1 Purpose

SocioCult (Society Management System) is the latest version of a newly created management system in which students and the administration of FAST university can handle the different issues regarding societies in FAST.

1.2 Product Scope

SocioCult is a system to manage all societies of FAST university. Our system will relieve manual registrations and hassles of both the students and the administration of FAST. With this system, the time taken for a simple process such as registering a society in FAST can be reduced into mere taps in our software. Its user friendly, so that all our audience could easily interact with it without any issues. Through this system administration can reject and approve the allocated or requested budgets of societies; moreover, students can easily view all the currently active societies in FAST and apply as a member in them through simple clicks. Furthermore, students can also apply to different events conducted by different societies and can also decide to create a new society if they wanted to within a few clicks.

1.3 Title

There are a lot of student societies in FAST university and as it gets hard to keep track of all the events being conducted by different societies, most of the students miss their opportunity to register or participate in any event. Our system will ensure that every proposal of a society goes through the authorities and all the queries of handling a society isn't a hassle anymore.

1.4 Objectives

- Manage event proposals
- Schedule interviews
- Display all events to be conducted
- Register a new society
- Events calendar
- Registering students to societies
- Registering in an event
- Admin can manage Societies
- Admin can manage users
- Admin can decide on budget proposals

1.5 Problem Statement

Student societies are a great opportunity to learn something new. But it surely is a hassle when you have to go through different channels to find out about a certain society. What if everything about a

society was provided to you on a single platform. What if societies could recruit people through an automated system. What if societies could send in proposals without any complications. Our system, SocioCult, is the perfect solution.

Our system makes student society management much more intuitive and easier to keep track of. All Student societies will be listed and new societies can register via our system. Event proposals, budget approvals, recruitments of members etc. can be done automatically. Everything that a society does or proposes can be reviewed by the person in charge via our system.

The issues such as manually registering a society and getting in the hassle of going to different departments and asking different authorities to get permission are now resolved through SocioCult as all of these issues can be done through our system without much of a hassle. Having issues of not reaching the director? No worries as through our system, societies can request budget from the administration which can reduce the trips to administration.

2. Overall Description

2.1 Product Perspective

SocioCult is a new self-contained product in which societies of FAST university can be seen and students and administration can achieve many different functionalities.

2.2 Product Functions

Major Functions for:

- Admin:
  - Manage Users
  - Manage Societies
  - Manage Budgets
- Students:
  - Register as member
  - Register/Apply to an upcoming event
  - Create/Form a new Society
  - View Event Calendar
- Society President:
  - Schedule Recruitment drive
  - Schedule Events
  - Request budget

2.3 List of Use Cases

- Schedule Events
- Administer Budget
- Apply to societies
- Register for events
- Register Society
- Manage Societies
- Schedule recruitment drives
- Manage users

2.4 Extended Use Cases

<table>
<tr><td>Use Case Name:</td><td>Schedule Events</td></tr>
<tr><td>Scope:</td><td>Student Society Management System</td></tr>
<tr><td>Level:</td><td>User Goal</td></tr>
<tr><td>Primary Actor:</td><td>Society President</td></tr>
<tr><td>Stakeholders and Interests:</td><td>Society President who wants to schedule event</td></tr>
<tr><td>Main Success Scenario:</td><td>
<table>
<tr><td>Actor Action</td><td>System Response</td></tr>
<tr><td>1. Society President enters event details</td><td></td></tr>
<tr><td></td><td>2. System display calendar to show what days/times are available</td></tr>
<tr><td>3. User selects the days and times they want to schedule</td><td></td></tr>
<tr><td></td><td>4. System allows admin to confirm event after schedule selected</td></tr>
<tr><td></td><td>5. System displays: “Event Scheduled”</td></tr>
</table>
</td></tr>
</table>

Extensions:	1. ***Any time the system halts the progress is saved and the system reconstructs previous state*** 2. ***If there is a clash with another event scheduled, then system will not allow event to be scheduled*** 3. ***No event can be confirmed unless admin has accepted***
Preconditions:	***Society must be registered***
Postconditions:	***Event Scheduled***

Use Case Name:	***Administer Budget***		
Scope:	***Student Society Management System***		
Level:	***User Goal***		
Primary Actor:	***Society President***		
Stakeholders and Interests:	1. ***Society President who requests for budget*** 2. ***Admin who approves/rejects request***		
Main Success Scenario:			
	Actor Action	***System Response***	
	1. Society President clicks on		

	Administer Budget |
	2. System displays finances currently available and budget request
	3. User clicks on budget request |
	4. System displays budget request form
	5. User fills in budget request form and submits it |
	5. System allows admin to accept/reject budget request
	6. System displays "Request accepted" or "Request Rejected"
Extensions:	***Any time the system halts the progress is saved and the system reconstructs previous state*** ***No request can be accepted/rejected without confirmation from admin***
Preconditions:	***Society must be registered***

Postconditions:	***Request accepted/rejected***

Use Case Name:	***Apply to societies***		
Scope:	***Student Society Management System***		
Level:	***User Goal***		
Primary Actor:	***Student***		
Stakeholders and Interests:	***Student who wants to apply to society*** ***Society President who will oversee students who have applied***		
Main Success Scenario:			
	Actor Action	***System Response***	
	1. Student clicks on Apply to Society		

	2. System displays all current societies
3. User selects the society they want to apply to	
	4. System displays the registration form for the society
5. Student fills in the form and submits it	
	6. System allows Society President to view the form and schedule interview
	7. System displays "Applied successfully"

Extensions:	***Any time the system halts the progress is saved and the system reconstructs previous state*** ***No society can be displayed unless it is registered in the system***
Preconditions:	***Student must be registered***
Postconditions:	***Application successful***

2.5 Fatima

Use Case Name:	***Register for events***
Scope:	***Student Society Management System***
Level:	***User Goal***
Primary Actor:	***Student***
Stakeholders and Interests:	***Student who wants to register for the event***
Main Success Scenario:	

Actor Action	*System Response*
1. Student clicks on register for event	
	2. System displays all ongoing and scheduled events
3. Student clicks on the event they want to register for	
	4. System displays the registration form
5. Student fills in the form and submits it	

		6. System displays "Registration successful"
Extensions:	*Any time the system halts the progress is saved and thesystem reconstructs previous state* *Student cannot select event which does not have its registrations opened*	
Preconditions:	*Student must be registered*	
Postconditions:	*Registration successful*	

2.6 Haissam

Use Case Name:	***Register Society***
Scope:	***Student Society Management System***
Level:	***User goal***
Primary Actor:	***Society president***
Stakeholders and Interests:	***Society president registering their new society***
Main Success Scenario:	

<table>
<tr><td></td><td>
<table>
<tr><th>Actor Action</th><th>System Response</th></tr>
<tr><td>1- Society president clicks on new society</td><td></td></tr>
<tr><td></td><td>2- System prompts user to fill form for the registration of new society</td></tr>
<tr><td>3- Society president fills form</td><td></td></tr>
<tr><td></td><td>4- System waits for admin’s approval</td></tr>
<tr><td>5- Admin approves the new society</td><td></td></tr>
<tr><td></td><td>6- System shows newly formed society</td></tr>
</table>
</td></tr>
<tr><td>Extensions:</td><td>- System saves inputted information of user in case of any mishap
- Prompts user to change society’s goals if similar to another society
- Society can’t be registered without admin’s approval</td></tr>
<tr><td>Preconditions:</td><td>Society president should be a student</td></tr>
<tr><td>Postconditions:</td><td>Society registered and created</td></tr>
</table>

<table>
<tr><td>Use Case Name:</td><td>Manage Societies</td></tr>
<tr><td>Scope:</td><td>Student Society Management System</td></tr>
<tr><td>Level:</td><td>User goal</td></tr>
<tr><td>Primary Actor:</td><td>Admin</td></tr>
<tr><td>Stakeholders and Interests:</td><td>Admin could modify society information</td></tr>
<tr><td>Main Success Scenario:</td><td>
<table>
<tr><th>Actor Action</th><th>System Response</th></tr>
<tr><td>1-Admin clicks on manage society</td><td></td></tr>
<tr><td></td><td>2- System prompts to enter the specific societies name</td></tr>
<tr><td>3- Admin enters the society</td><td></td></tr>
<tr><td></td><td>4- System retrieves specific society’s information</td></tr>
</table>
</td></tr>
</table>

	5- Admin changes the needed information such as update, create or delete it.
	6- System updates information
Extensions:	- ***Progress of the admin which they were changing is saved if the system crashes*** - ***System prompts if entered society isn't registered***

Preconditions:	***Society should be registered for modifying society's information***
Postconditions:	***society's information is updated, created or deleted.***

Use Case Name:	***Schedule recruitment drives***
Scope:	***Student Society Management System***
Level:	***User goal***
Primary Actor:	***Society President***
Stakeholders and Interests:	***Society President could schedule their recruitments***

Main Success Scenario:	

Actor Action	*System Response*
1- Society President clicks on schedule recruitment drives	
	2- System retrieves information of all the recruitment drives planned
3- Society President chooses the date and time for recruitment drives for specific clubs in the society	
	4- System prompts the user to set the interviews online or on campus
5- Society President decides and chooses the	
method of taking the interviews	
	6- System updates the event calendar.

Extensions:	- ***Progress of the society president which they were changing is saved if the system crashes*** - ***System prompts if entered date and time aren't available***
Preconditions:	***Society must be registered***
Postconditions:	***Societies planned recruitment drives could be seen in the events calendar***

3. Other Nonfunctional Requirements

3.1 Performance Requirements

The user's authentication is a key component of this system. In this system, user authentication is carried out by login using a user name and password and user type classification. Users will have access to the system based on the classification of their user type's permissions.

3.2 Safety Requirements

Verify that all data you input and is correct and up to date, i.e. personal name, event details etc.

3.3 Security Requirements

In our system, user's that are logged are only allowed the actions they have access to. Moreover, user login info is cross checked just to make sure there aren't any issues between the users of the application.

3.4 Software Quality Attributes

Our application is strong, dependable, and user-friendly. A user can remain logged in for as long as he likes and will still receive the best performance because all the data is contained, shielded from intruders, and maintained at run-time.

3.5 Business Rules

Admin can manage societies, manage users and administer budget proposals.
Student can view societies, view calendar, register for events or register a society.
Society presidents can view calendar, schedule events, schedule recruitment drives and administer society budget.

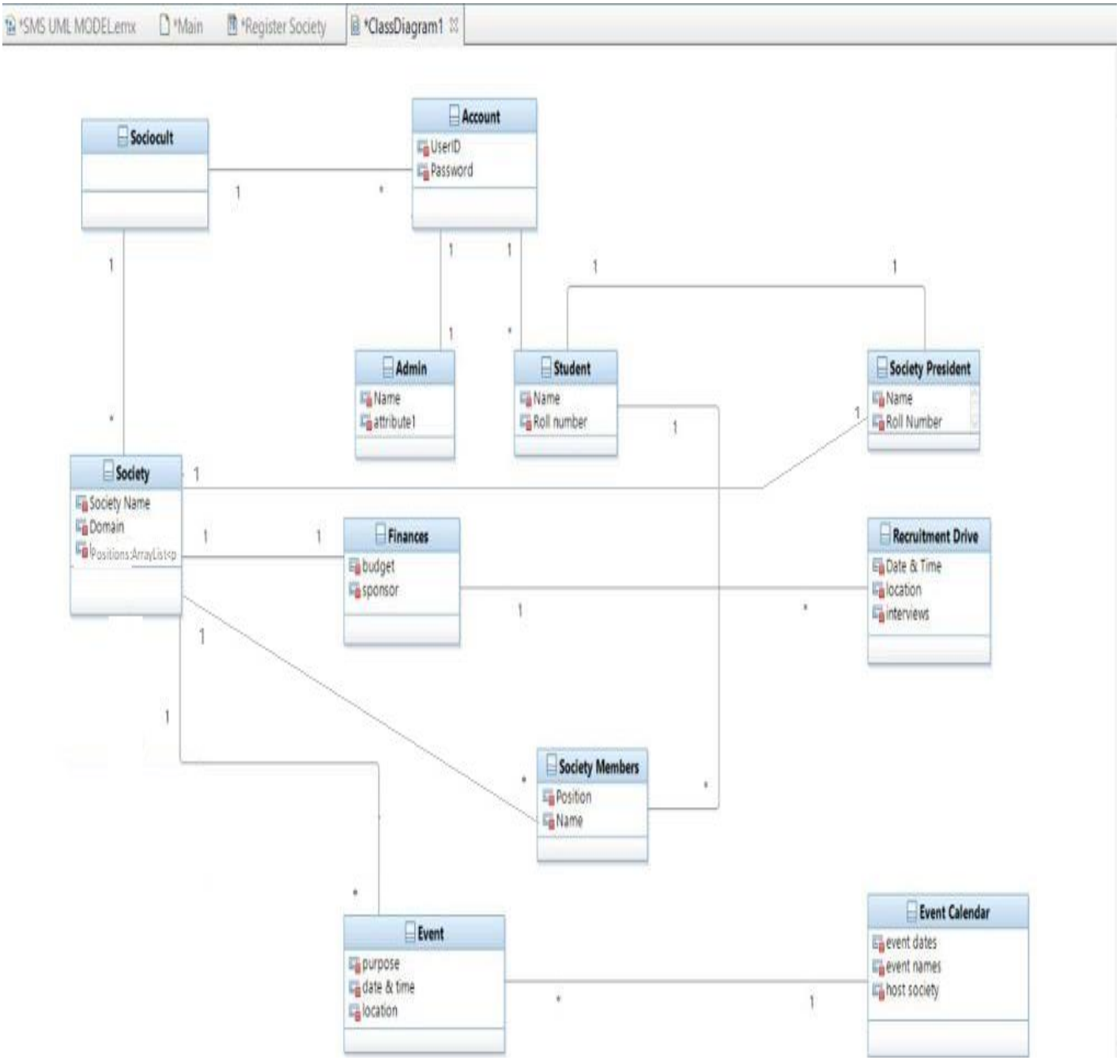
3.6 Operating Environment

The most recent JDBC connectors will be installed inside the project where the application will run in a javafx environment. An editor, such as SceneBuilder, is necessary to edit the fxml pages.

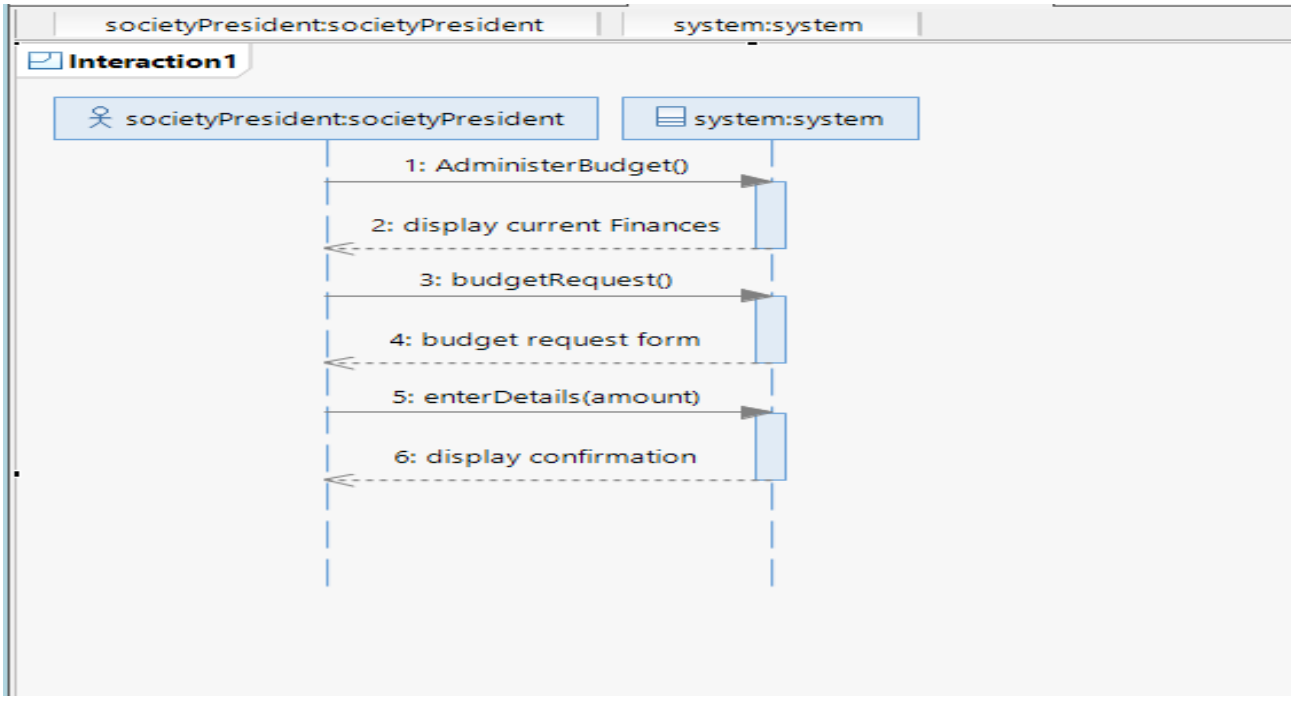
3.7 User Interfaces

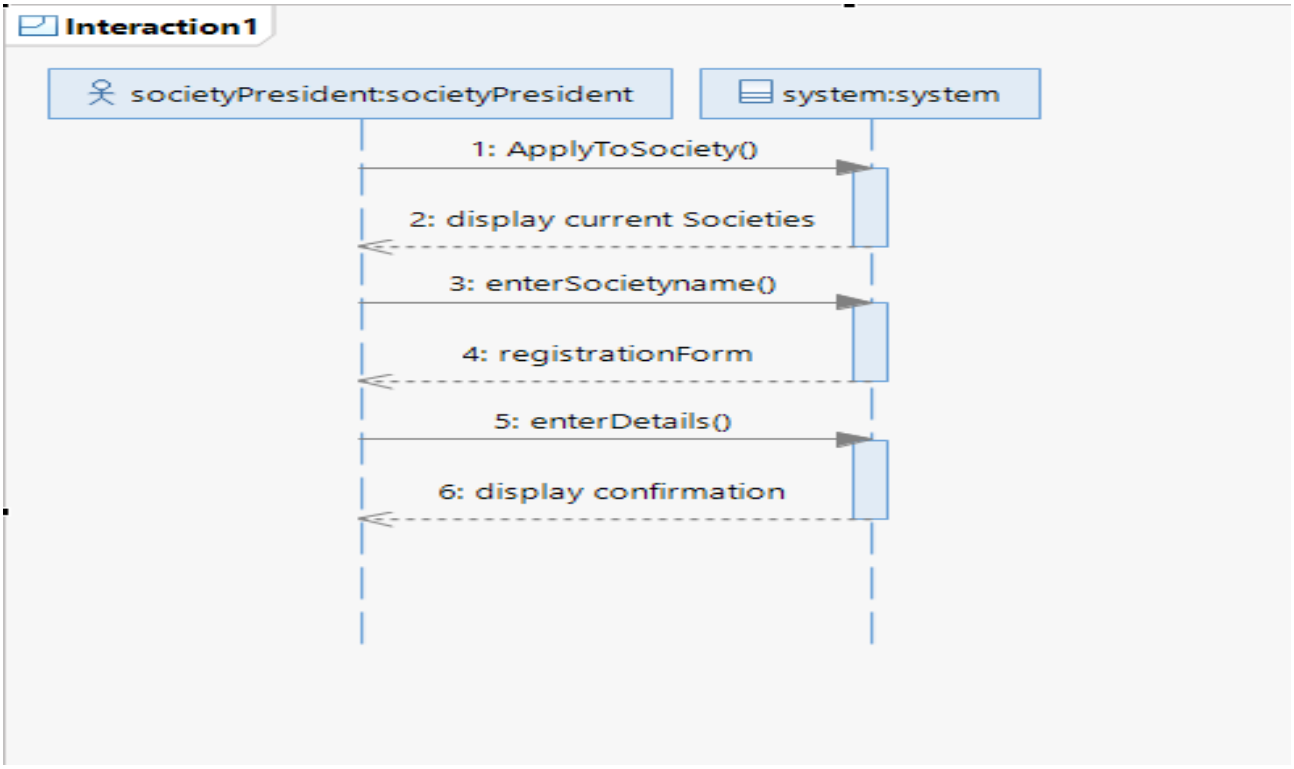
Each page has a consistent header and footer that can be seen. Four buttons are located inside the header, each of which performs a certain function as indicated by its name. Moreover a sliding dashboard opens as well. The system checks for empty submissions.

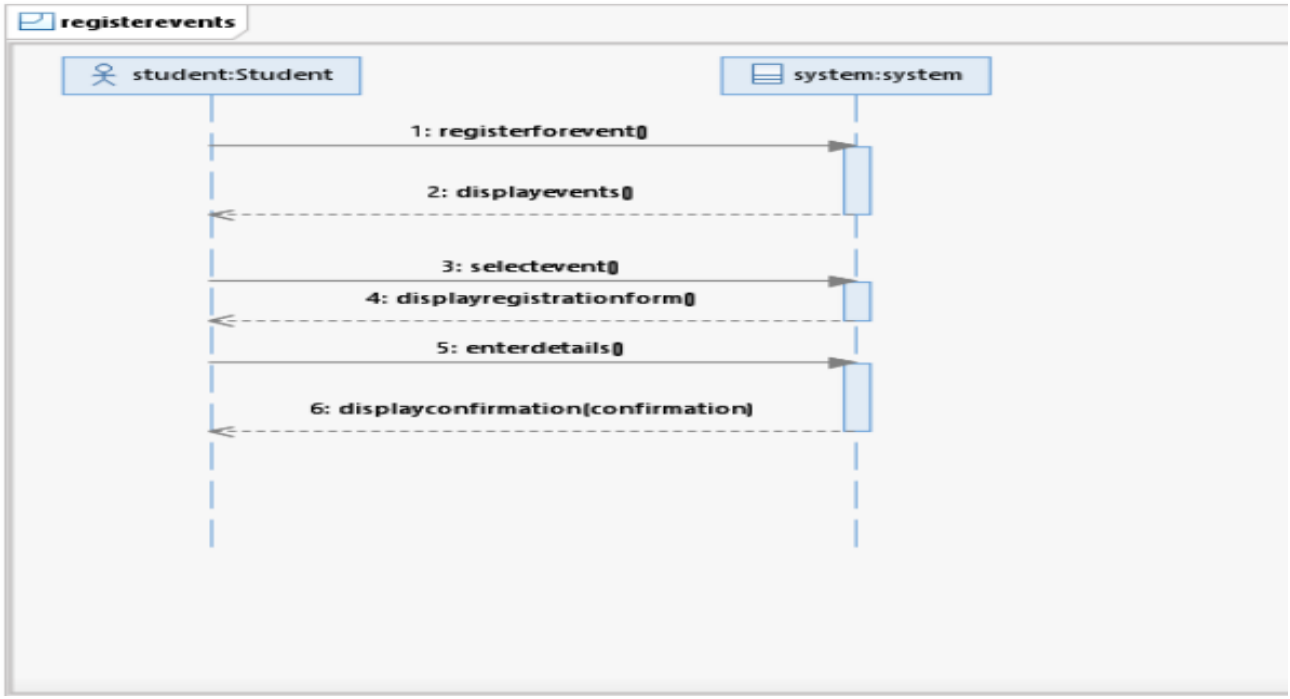
4. Domain Model

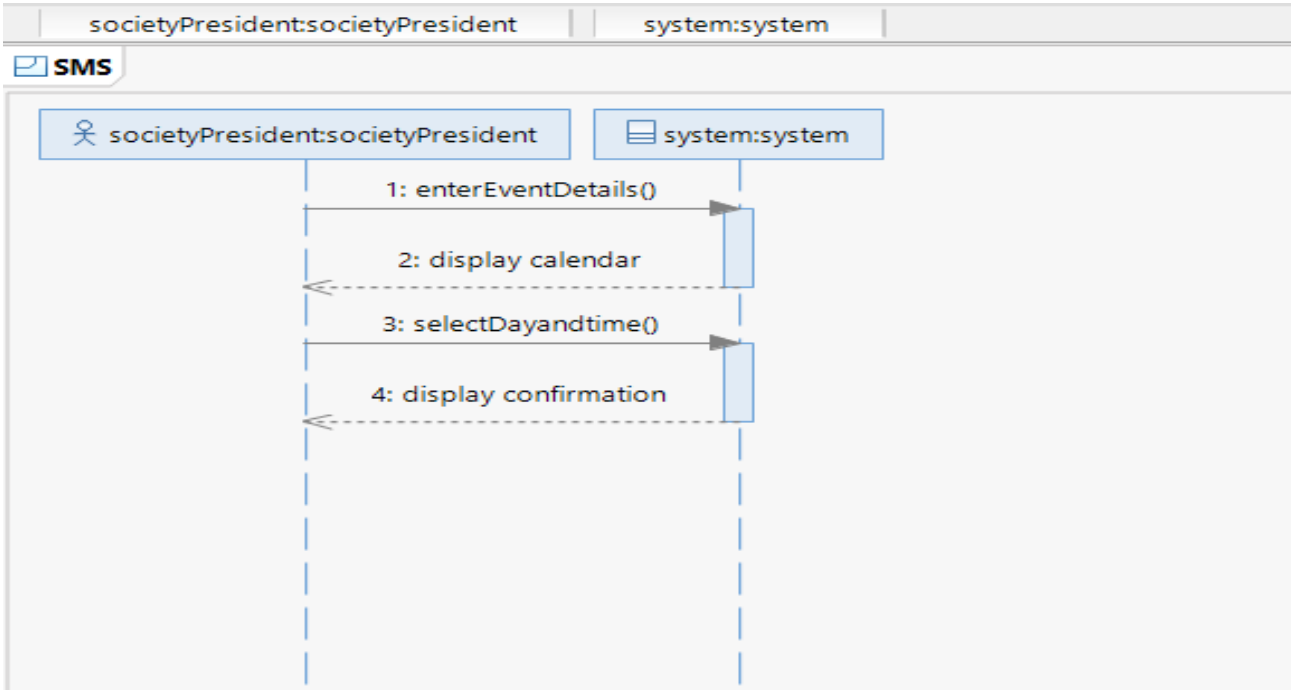


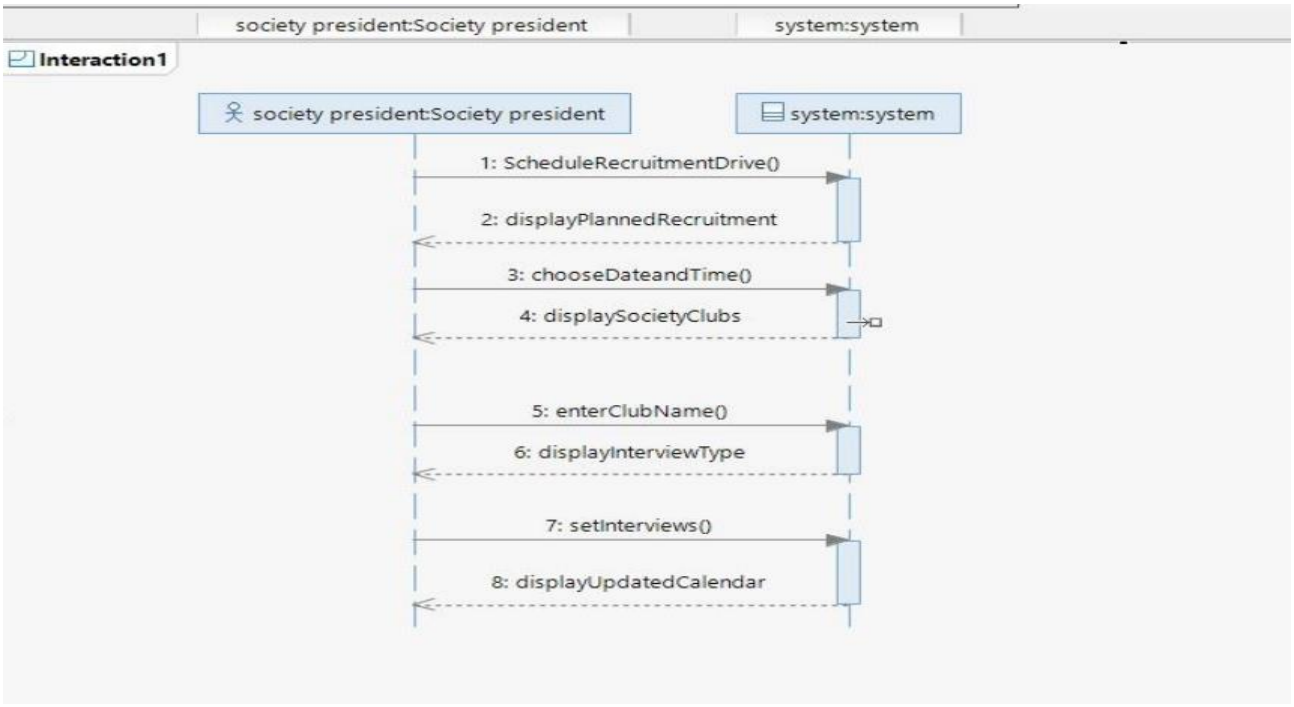
5. System Sequence Diagram

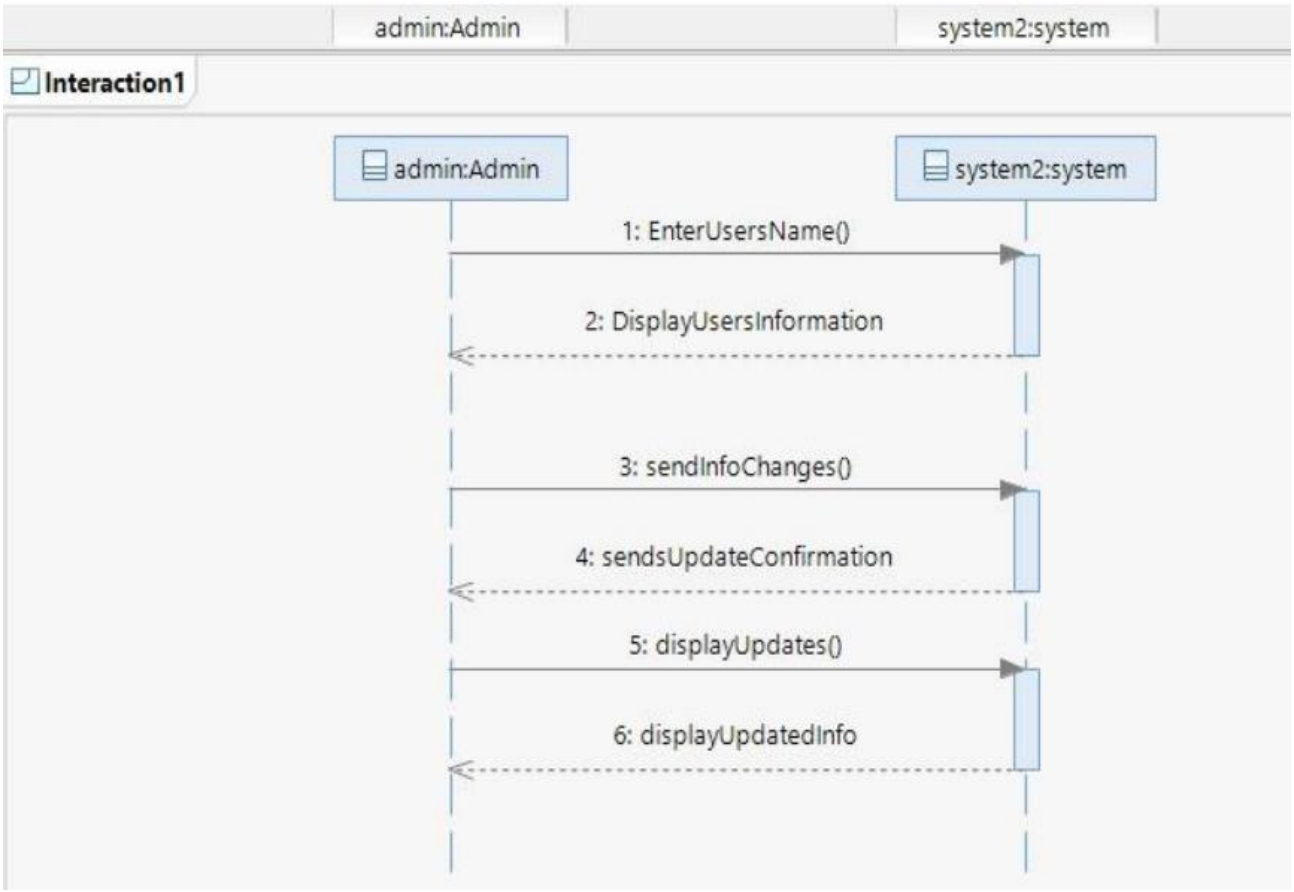


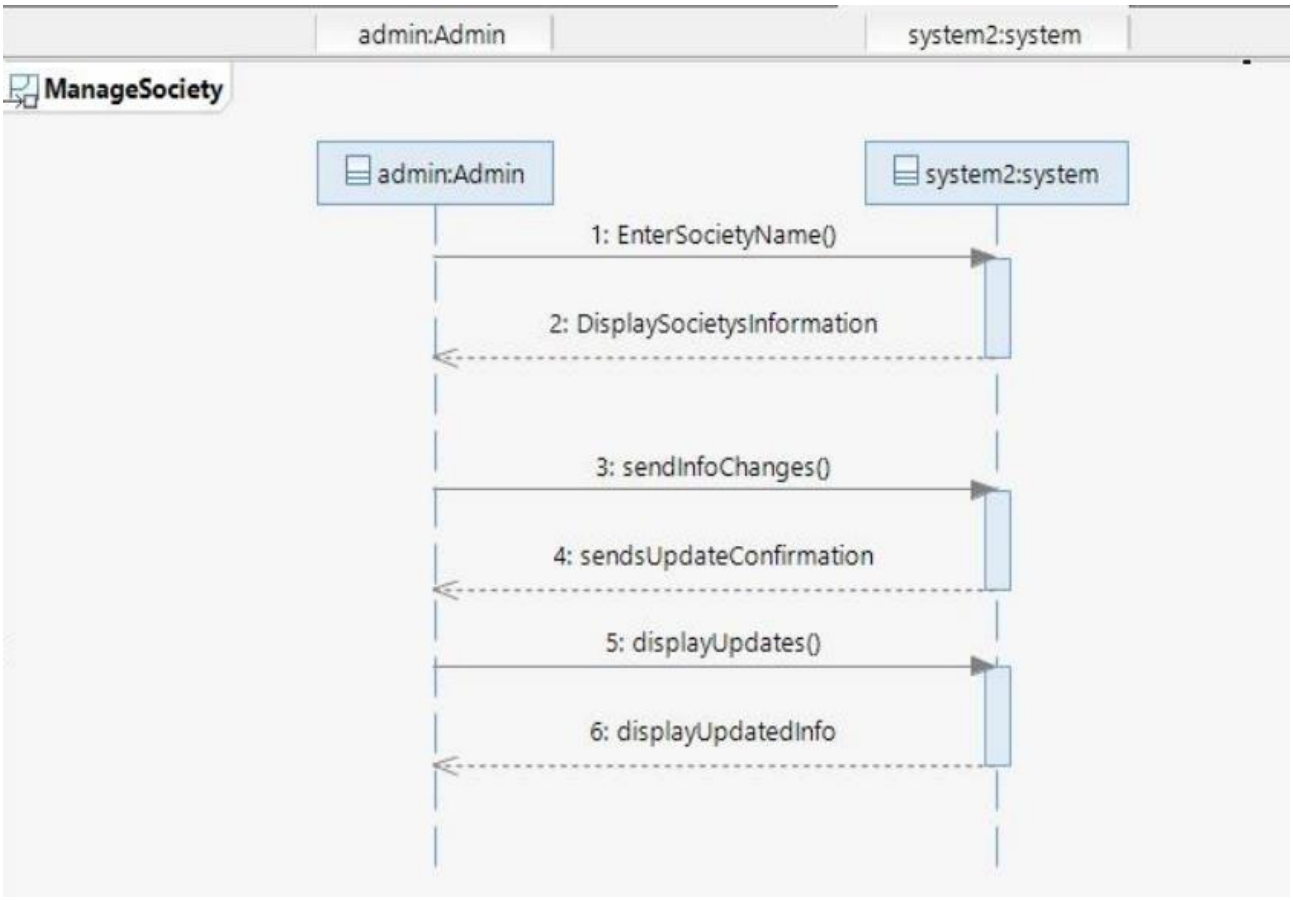


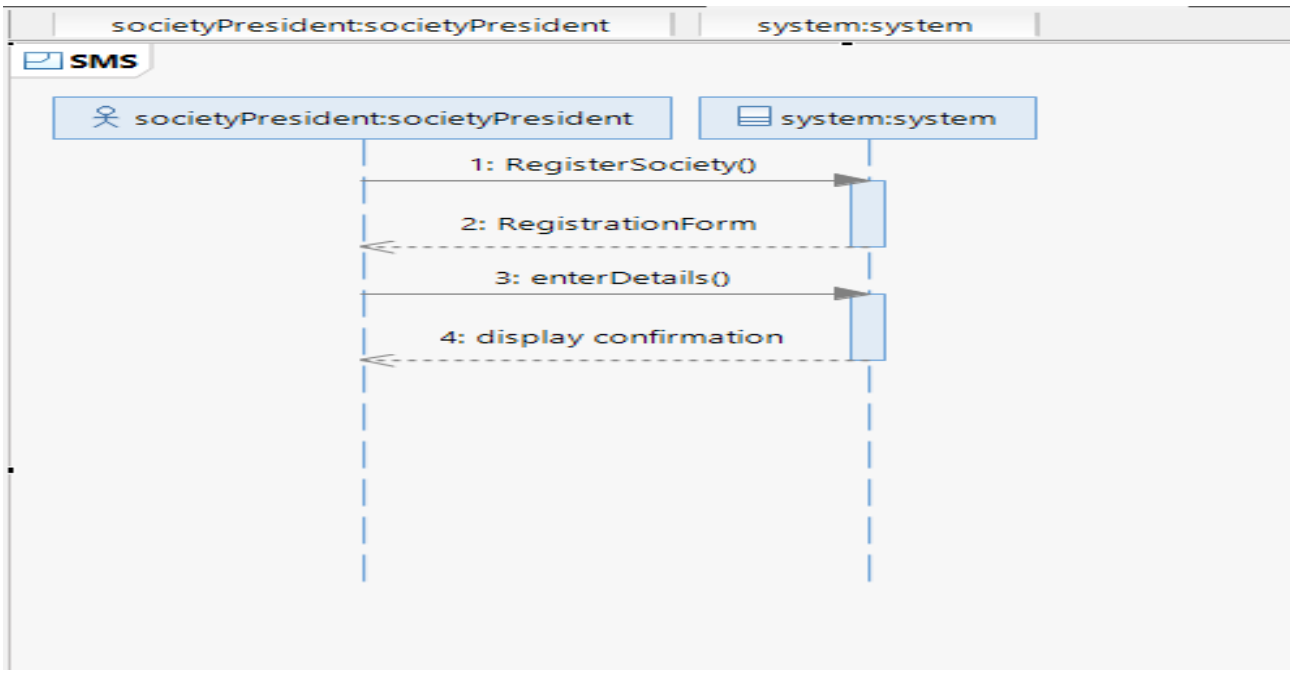

registerevents
student:Student
system:system
1: registerforevent()
2: displayevents()
3: selectevent()
4: displayregistrationform()
5: enterdetails()
6: displayconfirmation(confirmation)


societyPresident:societyPresident
system:system
SMS
societyPresident:societyPresident
system:system
1: enterEventDetails()
2: display calendar
3: selectDayandtime()
4: display confirmation

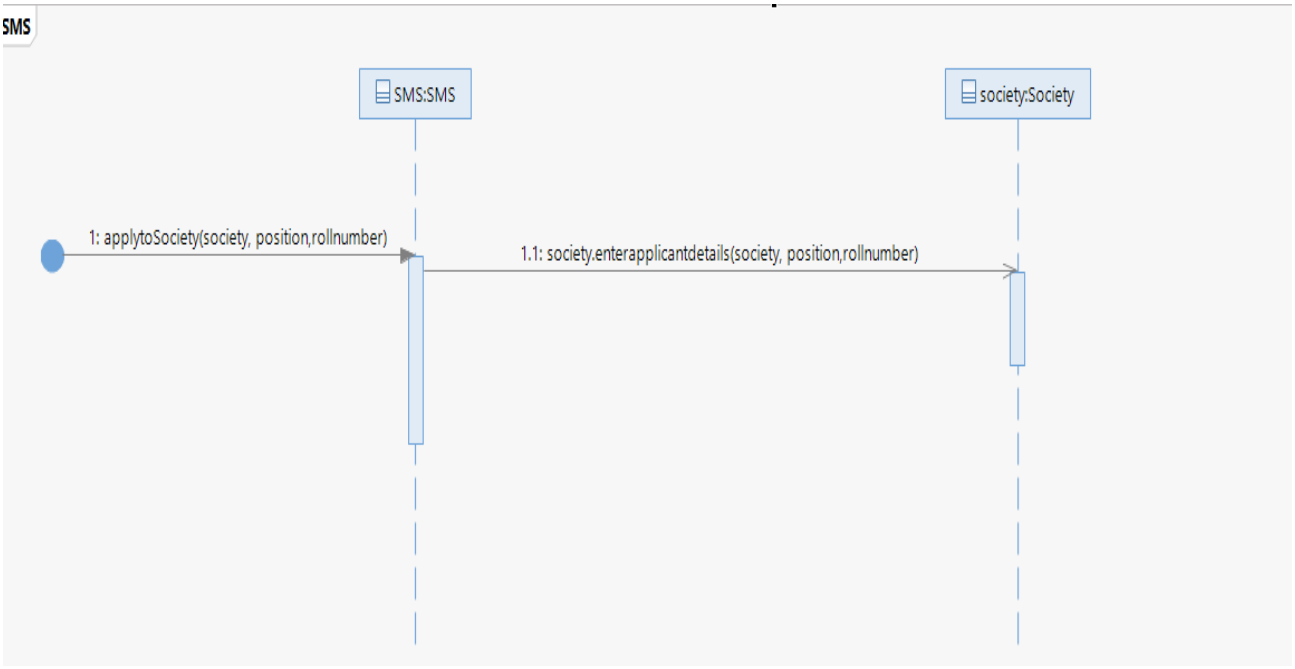

society president:Society president
system:system
Interaction1
society president:Society president
system:system
1: ScheduleRecruitmentDrive()
2: displayPlannedRecruitment
3: chooseDateandTime()
4: displaySocietyClubs
5: enterClubName()
6: displayInterviewType
7: setInterviews()
8: displayUpdatedCalendar

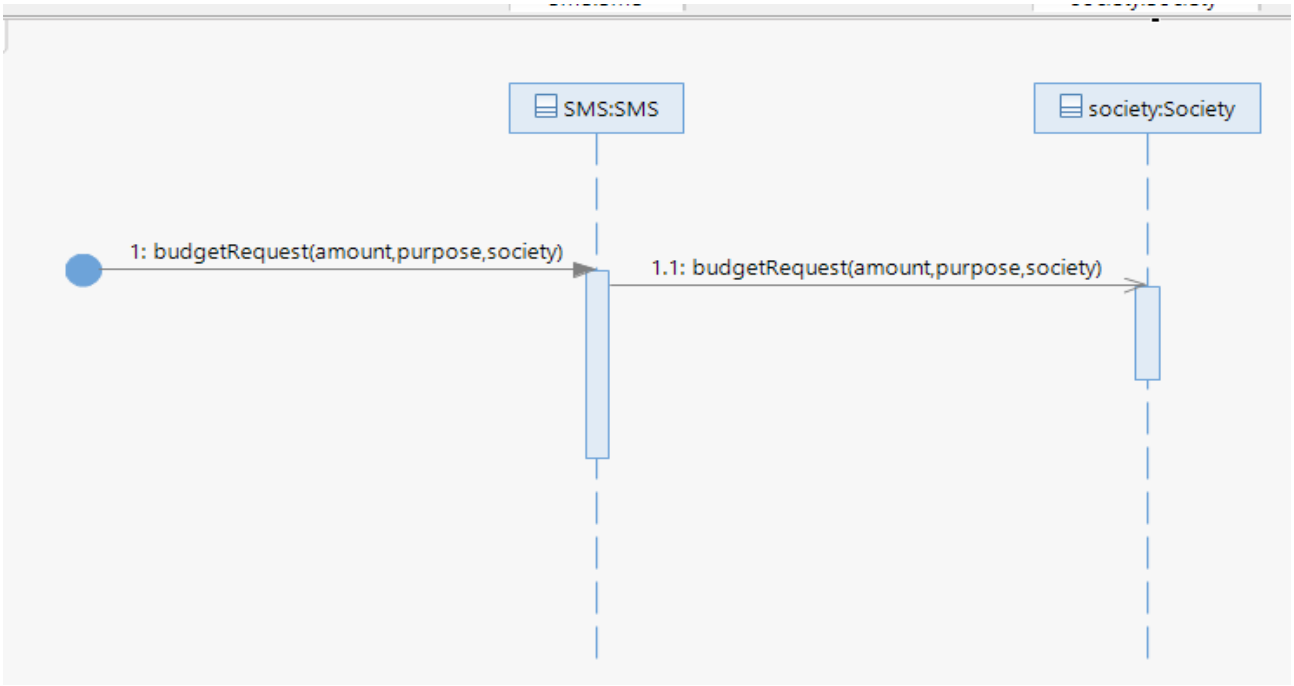

admin:Admin
system2:system
Interaction1
admin:Admin
system2:system
1: EnterUsersName()
2: DisplayUsersInformation
3: sendInfoChanges()
4: sendsUpdateConfirmation
5: displayUpdates()
6: displayUpdatedInfo

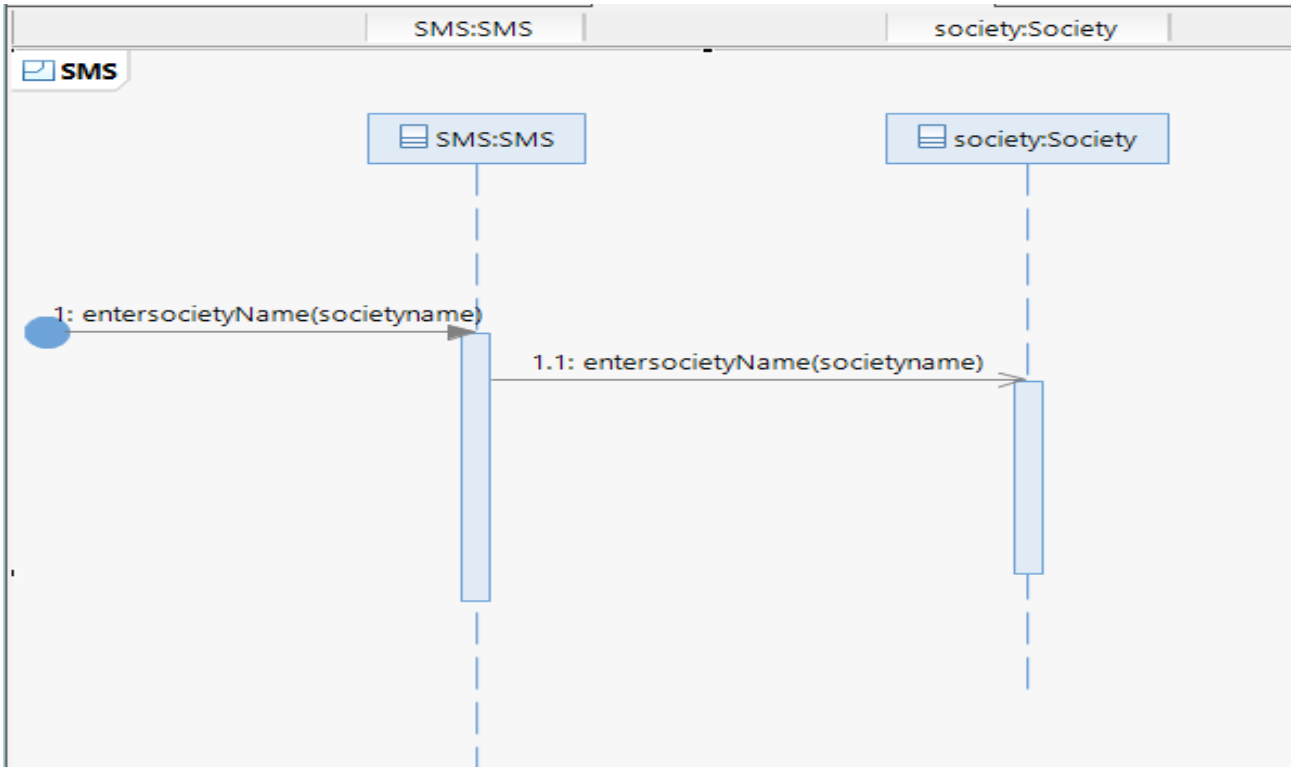

admin:Admin
system2:system
ManageSociety
admin:Admin
system2:system
1: EnterSocietyName()
2: DisplaySocietysInformation
3: sendInfoChanges()
4: sendsUpdateConfirmation
5: displayUpdates()
6: displayUpdatedInfo

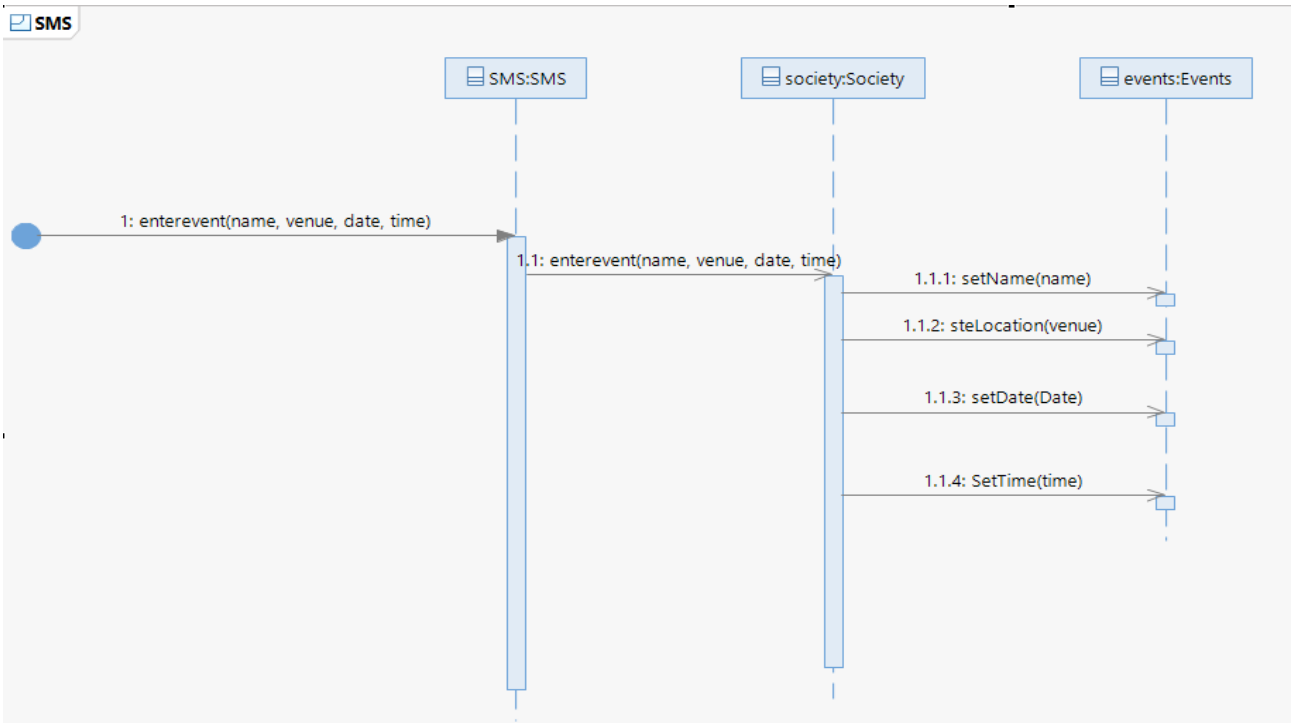

societyPresident:societyPresident
system:system
SMS
societyPresident:societyPresident
system:system
1: RegisterSociety()
2: RegistrationForm
3: enterDetails()
4: display confirmation

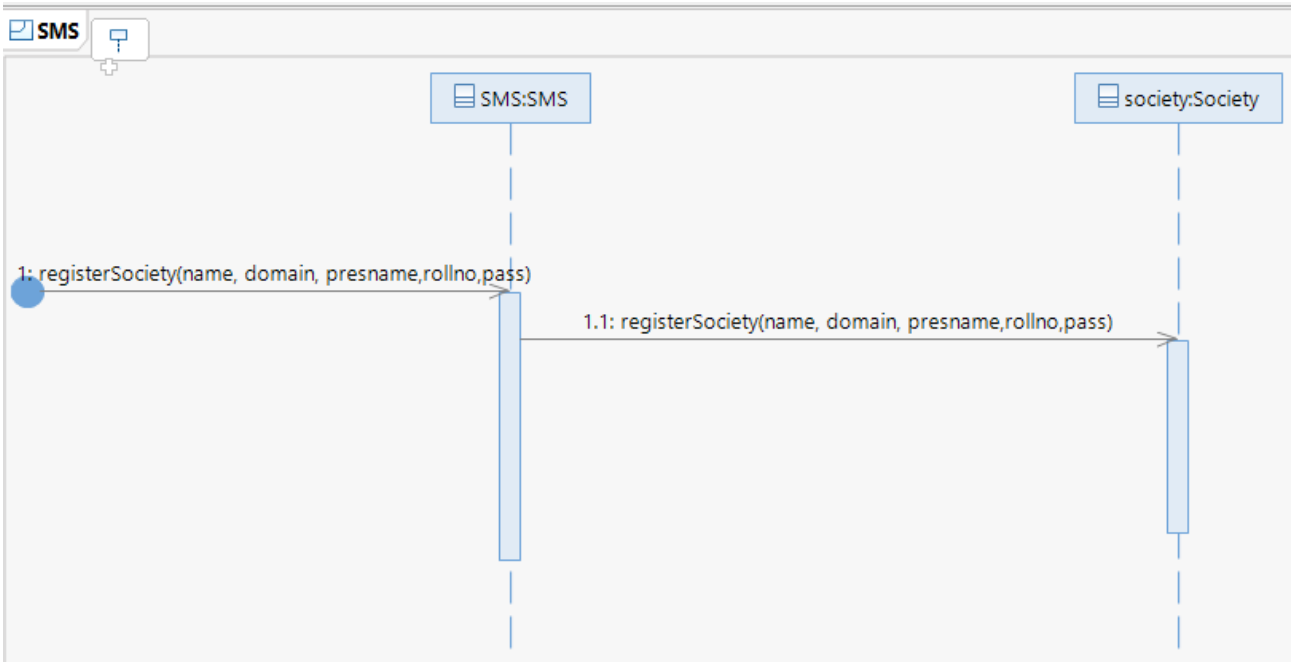
6. Sequence Diagram

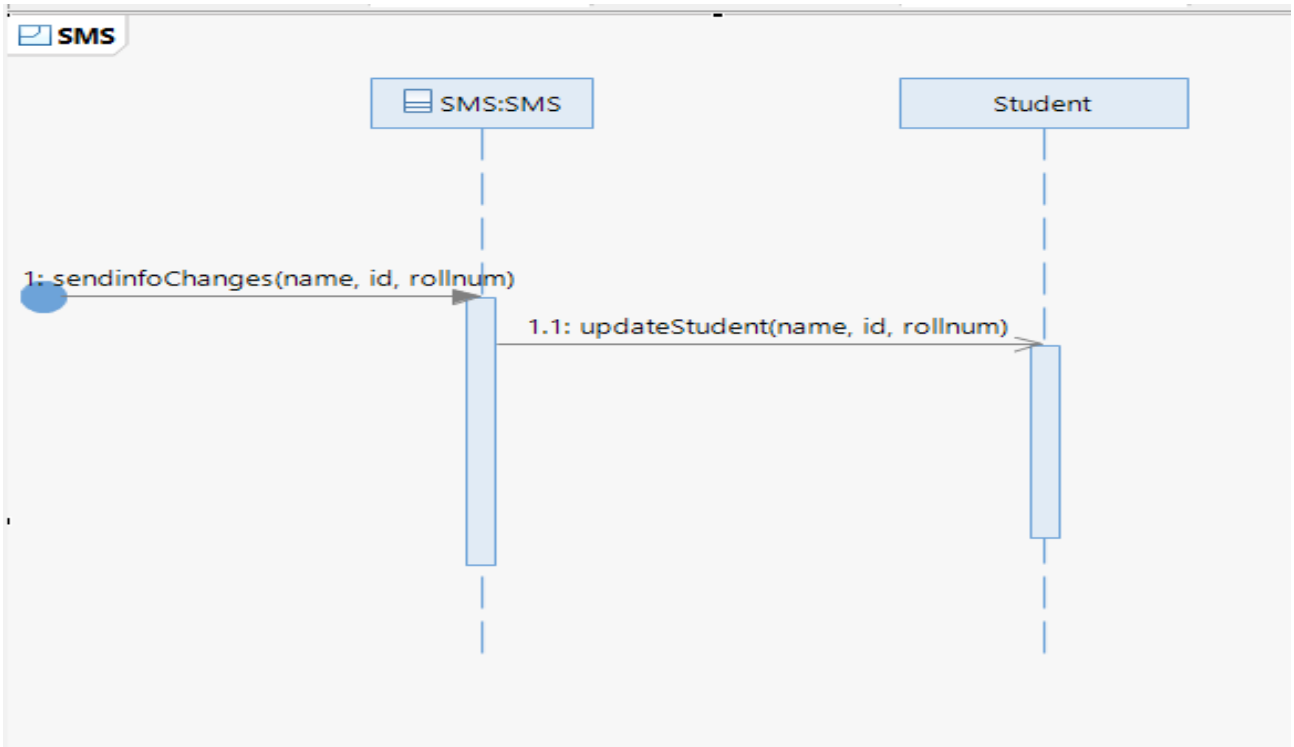


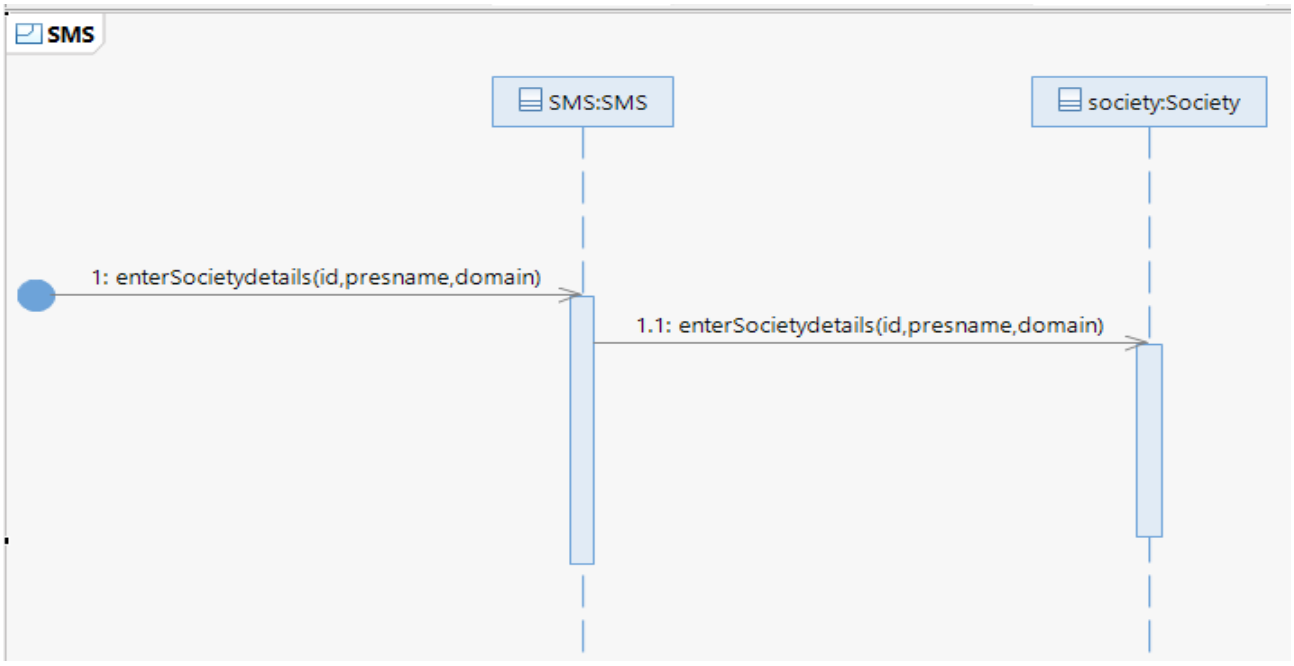


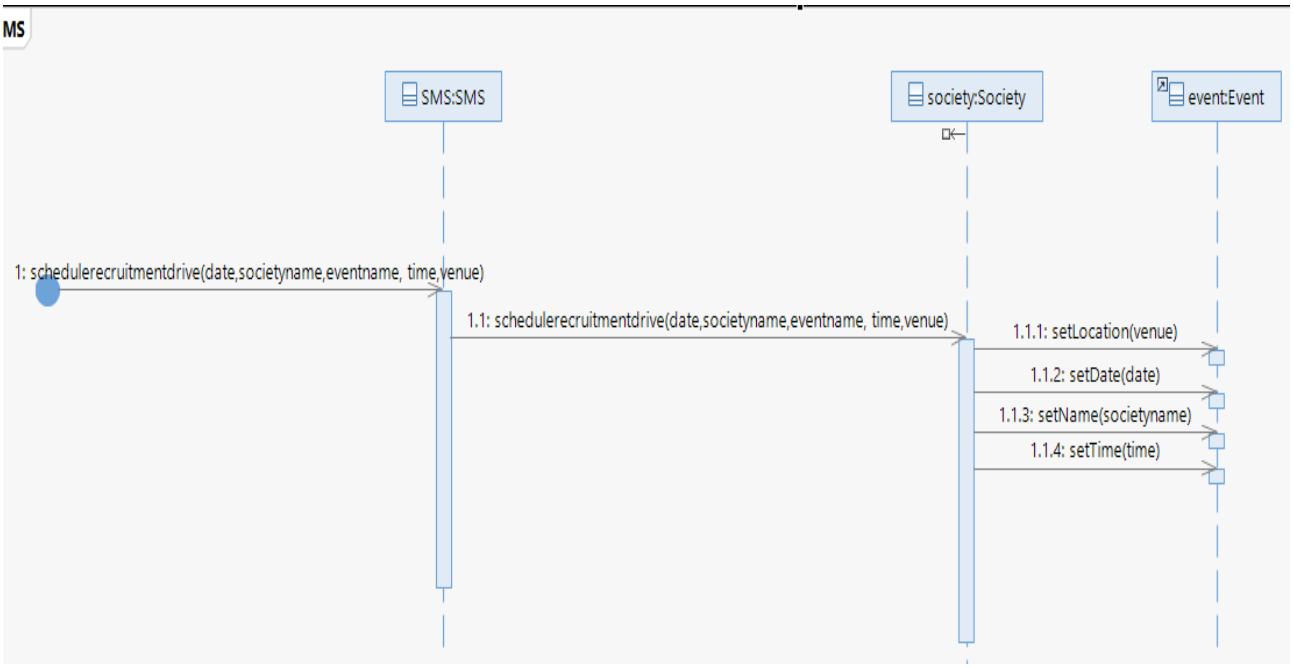

SMS:SMS
society:Society
SMS
SMS:SMS
society:Society
1: entersocietyName(societyname)
1.1: entersocietyName(societyname)

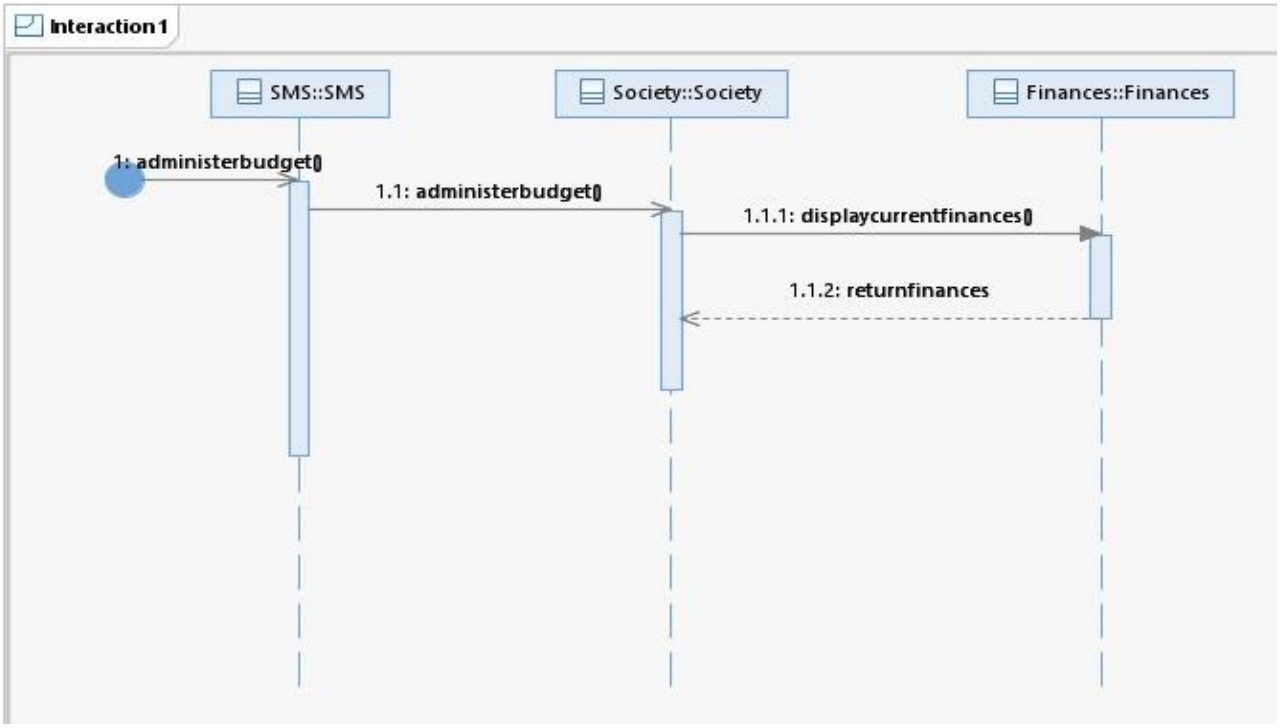

SMS
SMS:SMS
society:Society
events:Events
1: enterevent(name, venue, date, time)
1.1: enterevent(name, venue, date, time)
1.1.1: setName(name)
1.1.2: steLocation(venue)
1.1.3: setDate(Date)
1.1.4: SetTime(time)

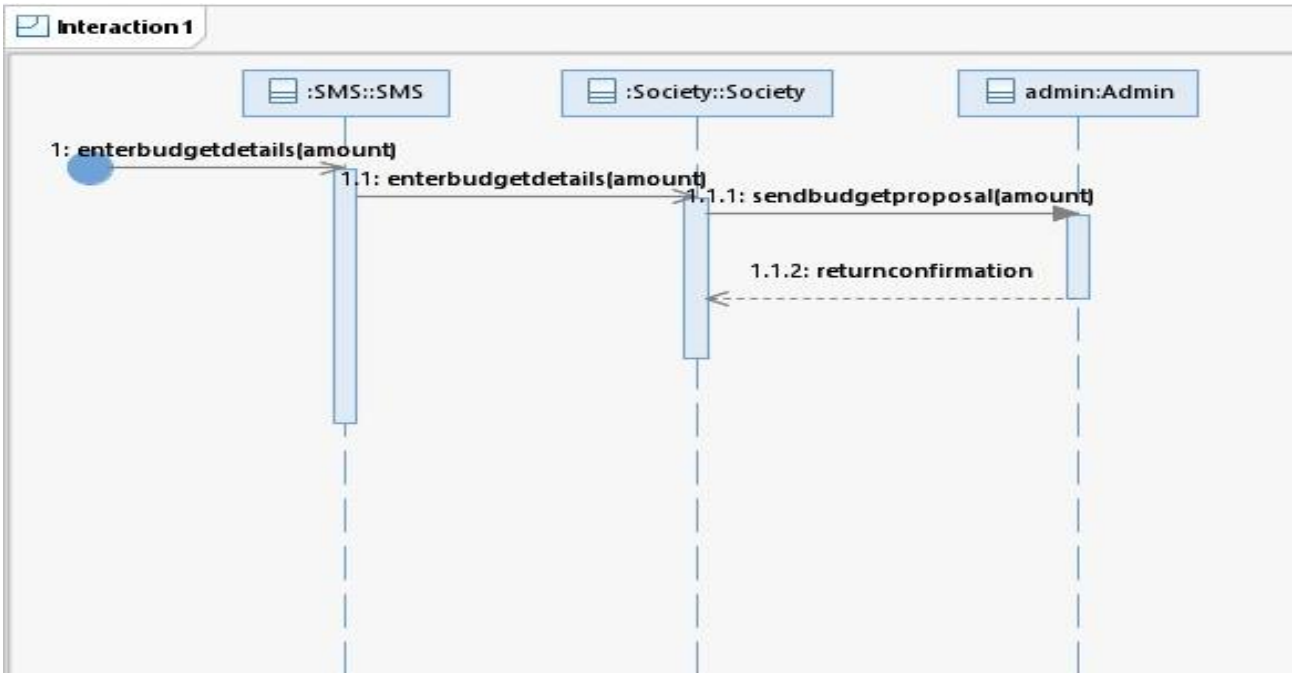

SMS
SMS:SMS
society:Society
1: registerSociety(name, domain, presname,rollno,pass)
1.1: registerSociety(name, domain, presname,rollno,pass)

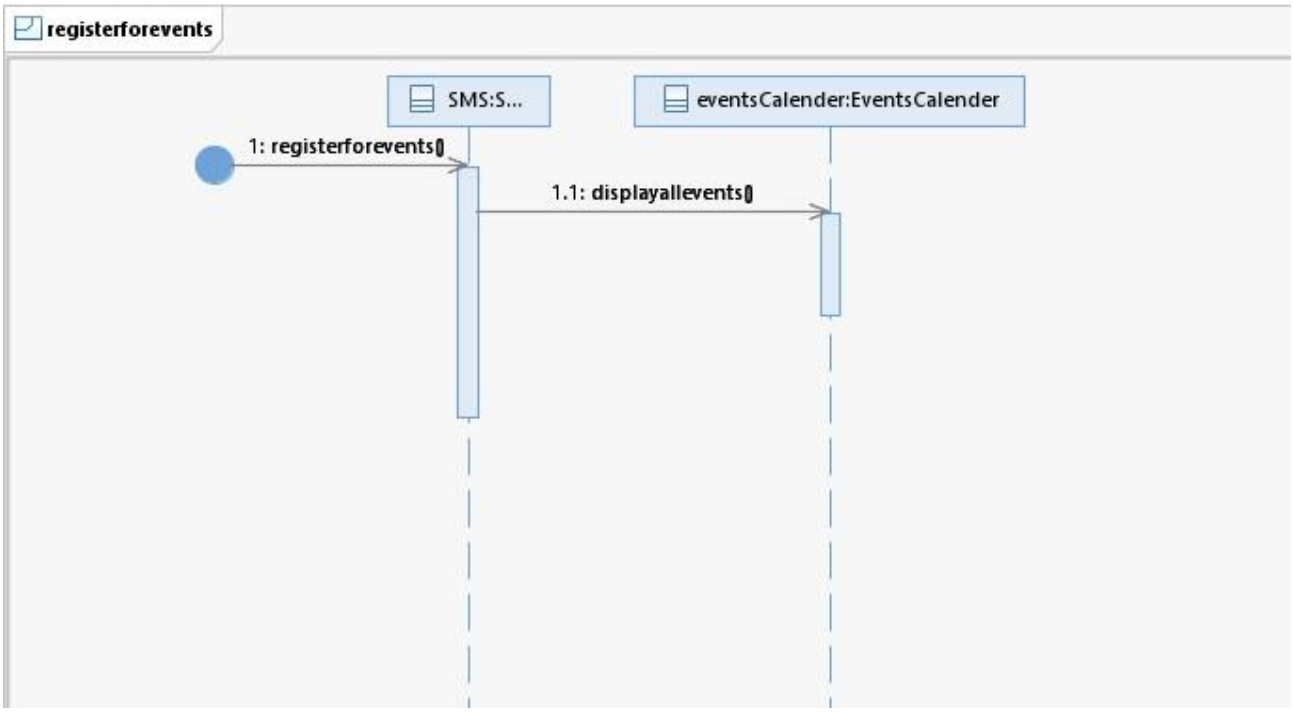

SMS
SMS:SMS
Student
1: sendinfoChanges(name, id, rollnum)
1.1: updateStudent(name, id, rollnum)

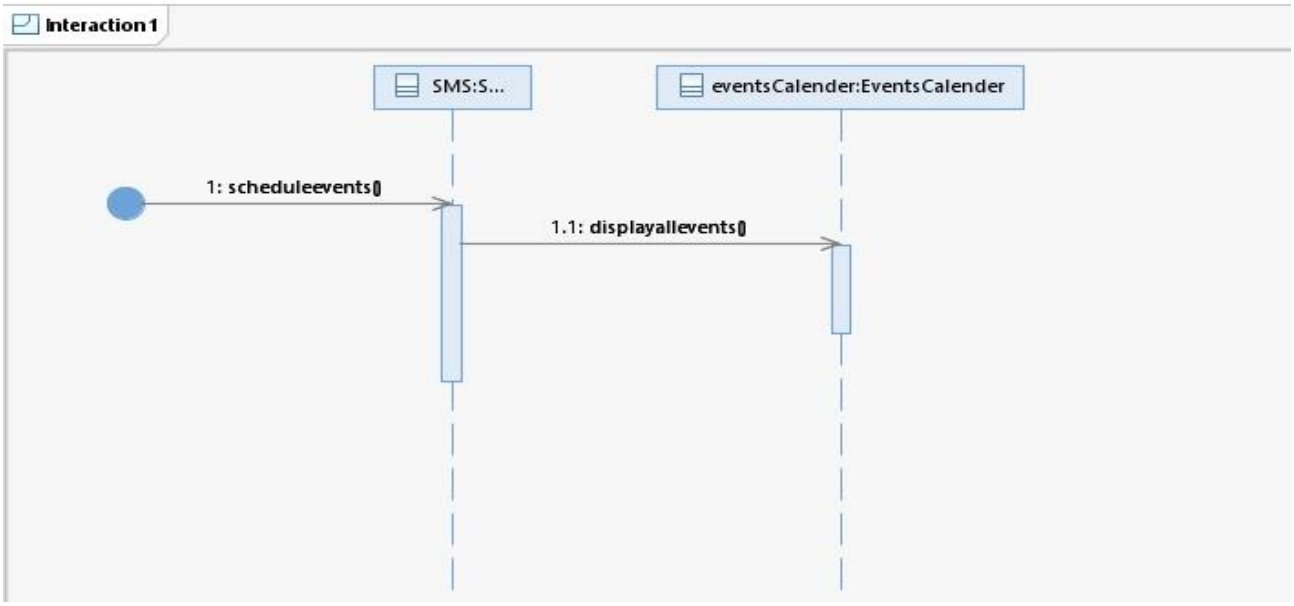

SMS
SMS:SMS
society:Society
1: enterSocietydetails(id,presname,domain)
1.1: enterSocietydetails(id,presname,domain)

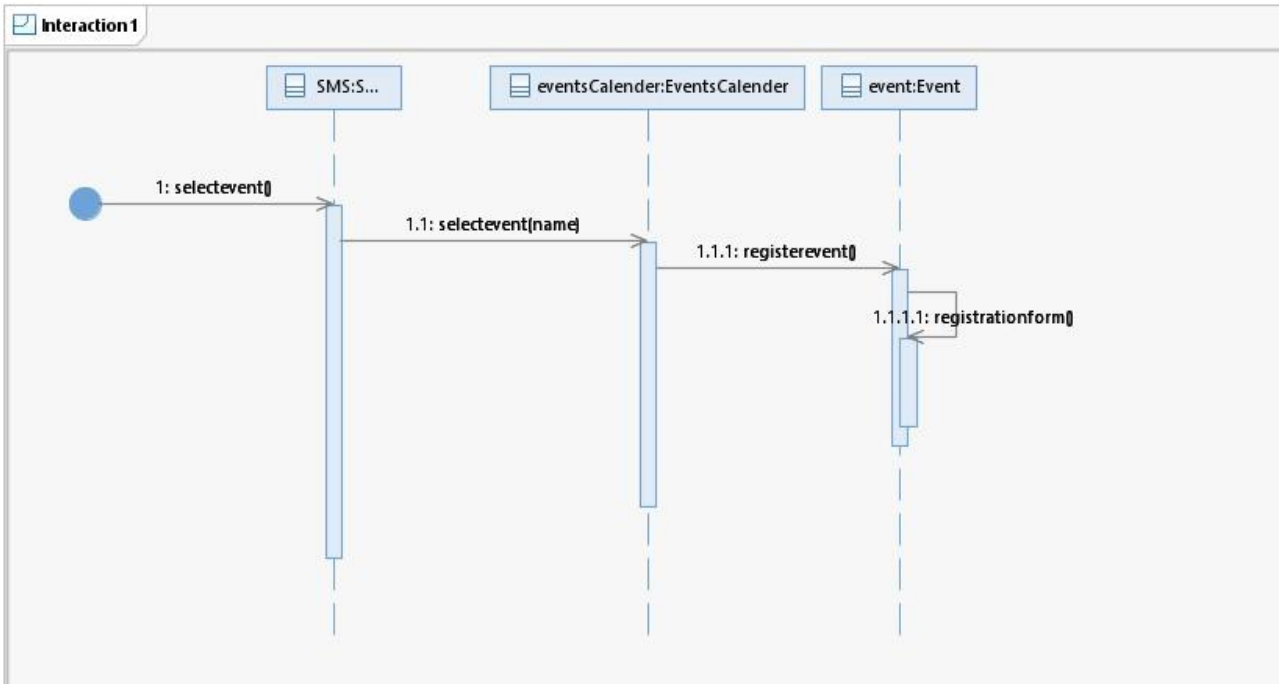

MS
SMS:SMS
society:Society
event:Event
1: scheduleRecruitmentdrive(date,societyname,eventname, time,venue)
1.1: scheduleRecruitmentdrive(date,societyname,eventname, time,venue)
1.1.1: setLocation(venue)
1.1.2: setDate(date)
1.1.3: setName(societyname)
1.1.4: setTime(time)

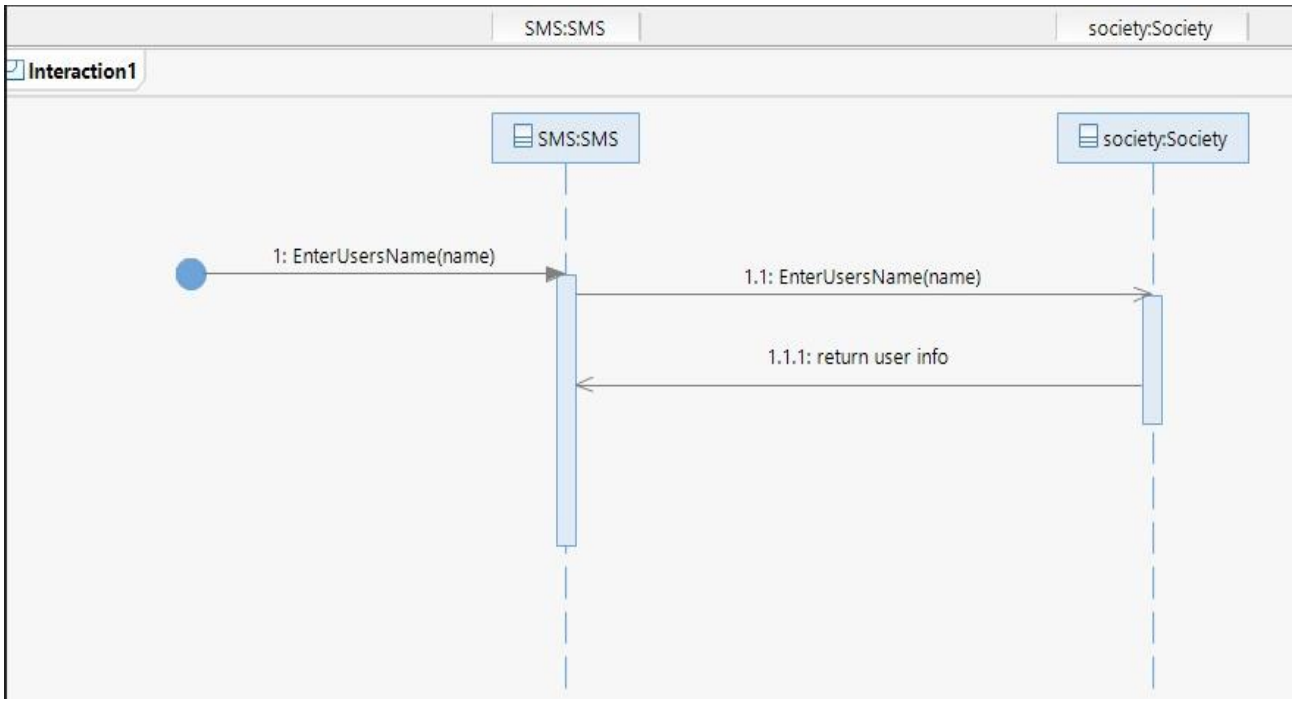

Interaction 1
SMS::SMS
Society::Society
Finances::Finances
1: administerbudget()
1.1: administerbudget()
1.1.1: displaycurrentfinances()
1.1.2: returnfinances


Interaction 1
:SMS::SMS
:Society::Society
admin:Admin
1: enterbudgetdetails(amount)
1.1: enterbudgetdetails(amount)
1.1.1: sendbudgetproposal(amount)
1.1.2: returnconfirmation

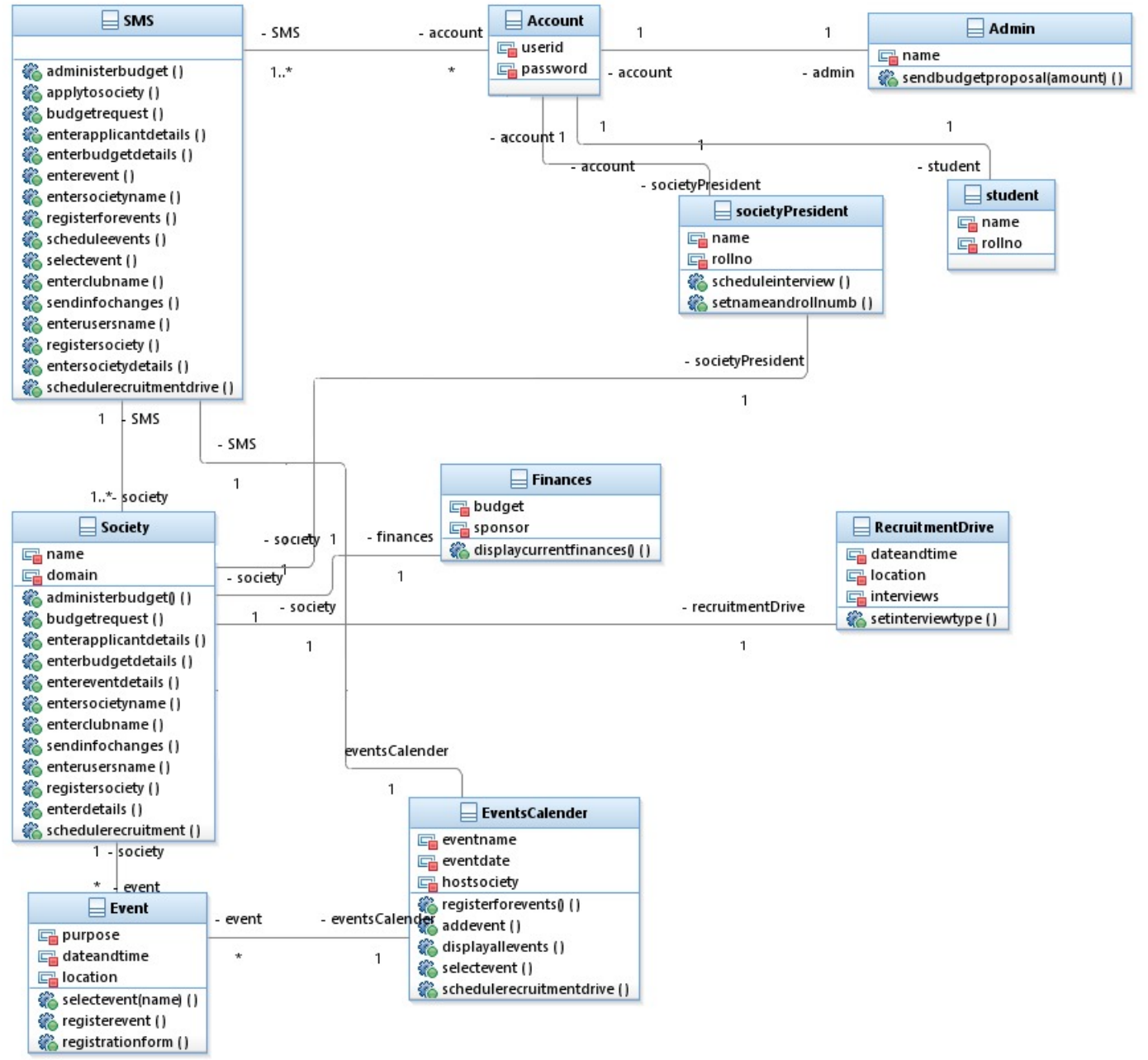

registerforevents
SMS:S...
eventsCalender:EventsCalender
1: registerforevents()
1.1: displayallevents()


Interaction 1
SMS:S...
eventsCalender:EventsCalender
1: scheduleevents()
1.1: displayallevents()

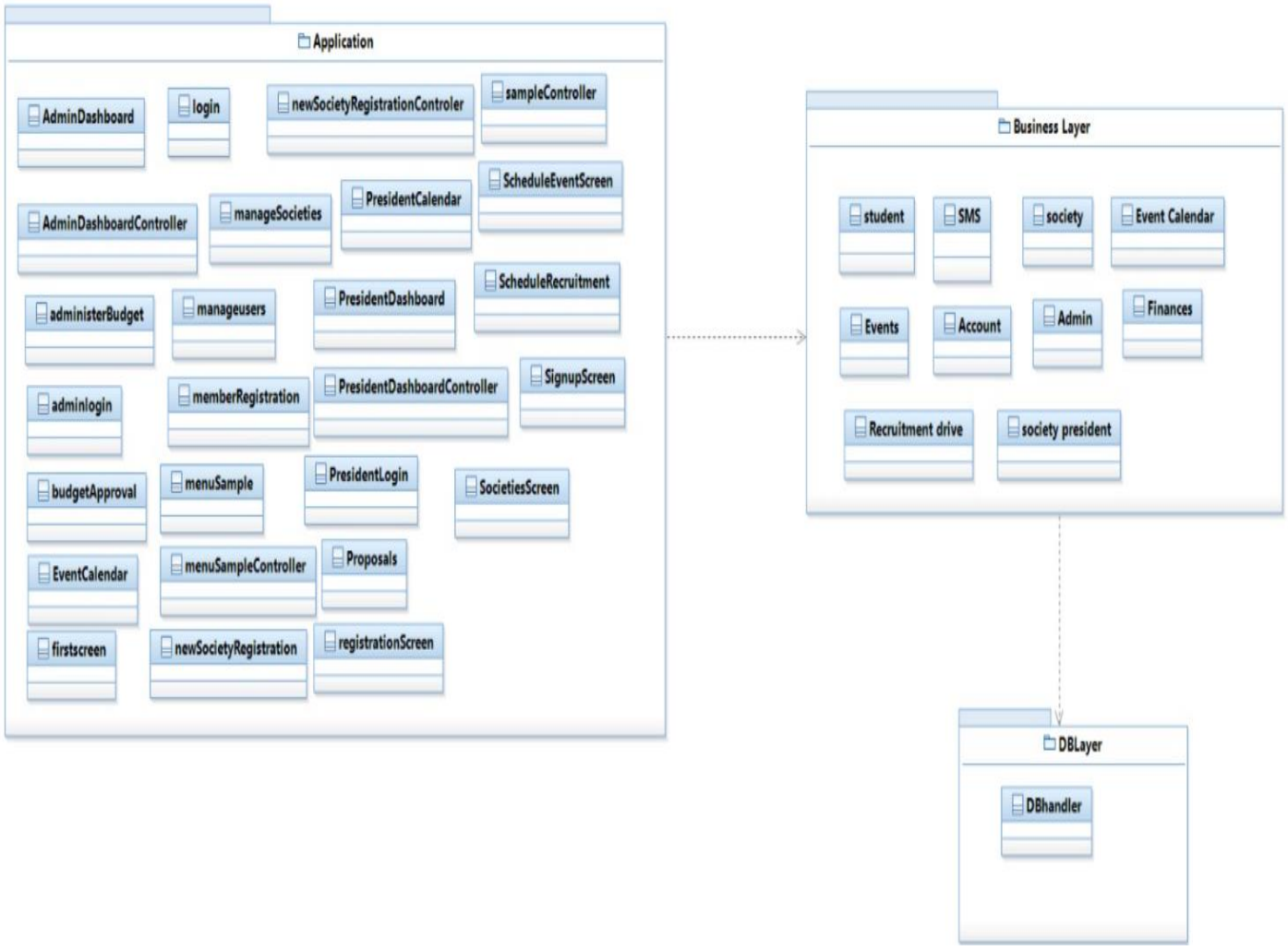

Interaction1
SMS:S...
eventsCalender:EventsCalender
event:Event
1: selectevent()
1.1: selectevent(name)
1.1.1: registerevent()
1.1.1.1: registrationform()


SMS:SMS
society:Society
Interaction1
SMS:SMS
society:Society
1: EnterUsersName(name)
1.1: EnterUsersName(name)
1.1.1: return user info

7. Class Diagram



8. Package Diagram



9. Deployment Diagram