Virtual Educational Assistant System

Jannatul Naim, Tanzina Akter Tani, , Rifah Tatrapi
Course Code: CSE435

Introduction:

Virtual educational assistant is a web system that has been developed so that any student can solve any kind of educational-related problem by contact with the experts who will guide the student virtually. Through this platform, a student can find an expert teacher for a specific topic of a field.

This web system is designed in a way that students and expert teachers can communicate fast. The student and teacher have different roles and activities in this system. When any user creates an account, he or she can specify whether his or her role is student or teacher. If the user is a teacher, he or she needs to add information about his or her expert area. He or she can add the subject that he or she wants to give tuition.

On the other hand, if the role is student, then he or she can choose any subject field that he or she wants to take tuition. He or she can see the list of teachers who is online for the tuition. Students have to pay for taking any teaching help. Whenever a student pays for a specific subject field, he or she gets a meeting link and a confirmation mail of payment. The teacher gets a notification by an SMS that the student wants to meet him or her. So, the teacher will go to his or her user page. He or she then can see a meet link that is automatically generated for both student and teacher. The teacher must join the meeting in 5 minutes. After the teaching assistantship is finished, the student can rate the system and the teacher. Also, any user can give feedback related to any matter and can also edit the profile.

There is also an admin panel where the admin is already registered in the system. After verified login, the admin can see the dashboard where the overall information of the system exists. The admin can also see the student and teacher list. Besides that, he or she can add or remove any subject field and also, sub-field in those subject fields.

This system explores a world of knowledge in front of the student where they can collaborate with highly qualified teachers.

Analysis of Sequence Diagram

General Activity of User (Teacher/Student)

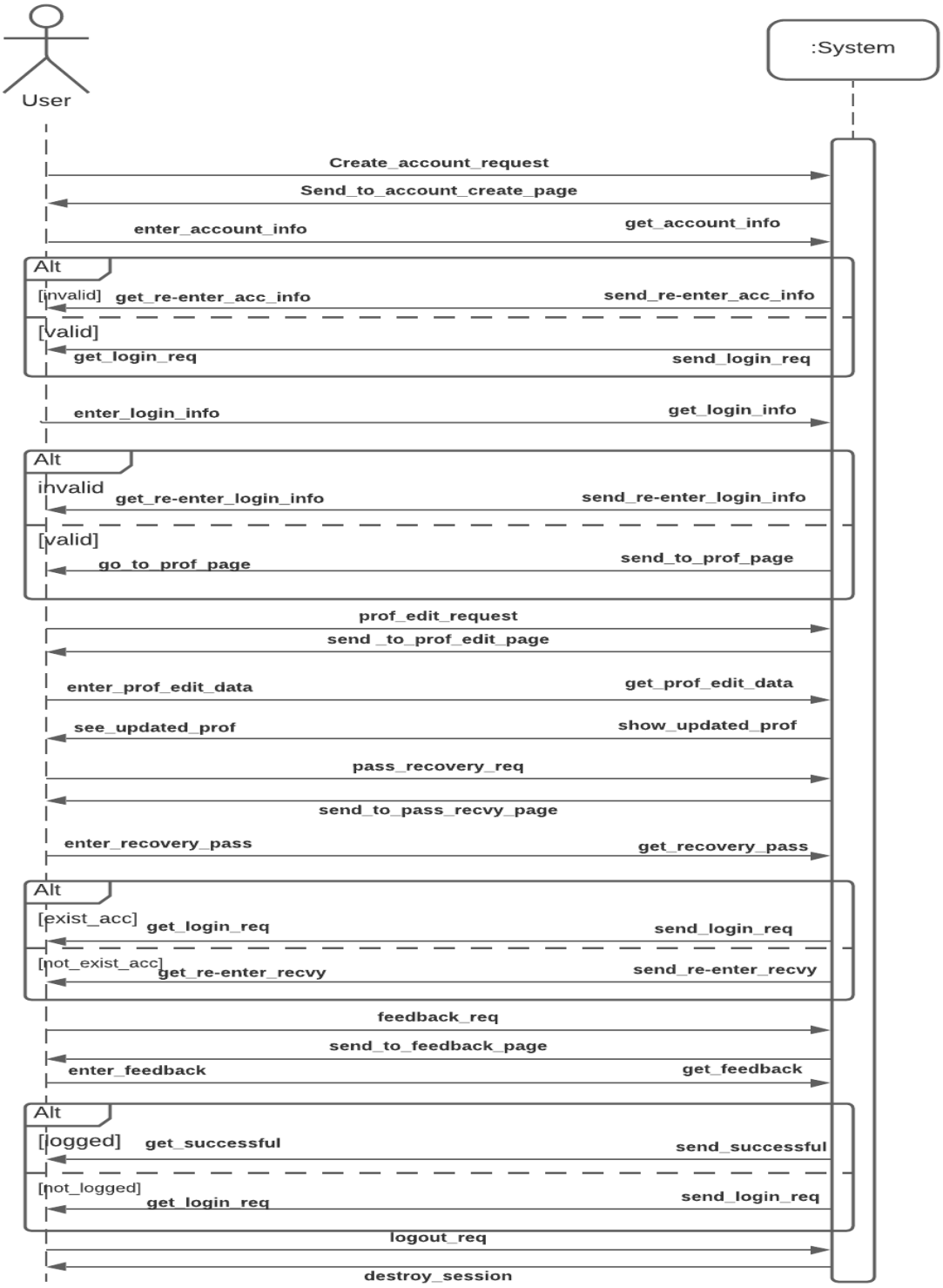

Fig.1 Sequence diagram of general activity of user.

Sequence Diagram Task Description:

When the user creates an account request, the system sends the user to the Account Create Page. Then the user enters data to create an account. If the system gets invalid data, then the system will send a request to the user for re-entering data. If the system gets valid data, the system will send a user to a login request. Then the user can enter data for login or send a request to the system for password recovery. If the user sends a request for password recovery, the system goes to the password recovery page. When the user enters the data for password recovery, the system checks if his or her account already exists or not. If the account already exists, then the password will be recovered successfully otherwise the system will send a create account request to the user. If the user enters data for login, then the system will check the validity of the data. If the login info is correct, then the user gets into the user profile page otherwise the system will send a request to re-enter login data. If the user request for edit profile, the system will take the user to the profile edit page. After the user edits his/her profile, the system will show the updated profile of the user. The user can send a feedback request to the system and in response to it, will get into the feedback page. When the user sends feedback, the system checks if he/she logged in or not. If the user does not log into his/her account, then the system will send a request to the user to log in. Otherwise, the feedback will be submitted successfully. The user can send a logout request after every activity of his/her and in response, the system will destroy the session.

PROMELA code

```
mtype = { crt_acc_req, snd_acc_pg, entr_acc_info, login_req, login_info, prof_pg, prof_edit,
profedit_pg, prof_edit_data, update_prof, pass_rcvy_req,  pass_rcvy_pg, rcvy_data, feedback_req,
feedback_pg, feedback_data, successful, logout_req, destroy_session};

chan to_user = [1] of { mtype };

chan to_system = [1] of { mtype };


active proctype User()
{ bit invalid, logged, feedback, logout;
short recovery;
start:
    atomic {
            invalid= 0;
             logged  = 1;
             feedback = 1;
             logout = 1;
             recovery = 1;
            goto again
          };
s5:
 atomic {
             recovery = recovery+1;
            goto s6
          };
```

```
again: to_system!crt_acc_req;
 to_user?snd_acc_pg;
s1:  to_system!entr_acc_info;
if
::(!invalid)-> to_user?login_req;
            s2: to_system!login_info;
                if
                ::(!invalid)-> to_user?prof_pg;
                               to_system!prof_edit;
                                to_user?profedit_pg;
                                to_system!prof_edit_data;
                                to_user?update_prof;
                                if
                               ::(recovery != 1 ) ->
                                                goto s4;
                               ::(recovery == 1)->
                                              to_system! pass_rcvy_req;
                                                to_user? pass_rcvy_pg;
                                             s3: to_system! rcvy_data;
                                                if
                                                ::(!invalid)->
                                                              to_user?login_req;
                                                                goto s5;
                                                        s6:  goto s2;
                                                :: (invalid) ->
                                                                goto s3
                                                fi;
                                 fi;
                                 s4:
                                 if
                                 ::(feedback)-> to_system! feedback_req;
                                                to_user? feedback_pg;
                                                to_system! feedback_data;
                                                if
                                                ::(logged)-> to_user?successful;
                                                ::(!logged)-> to_user?login_req;
                                                              goto s2;
                                                fi;
                                   fi;
                                   if
                                   ::(logout)->to_system! logout_req;
                                               to_user? destroy_session;
                                   fi;
                :: (invalid) ->
```

```
                    goto s1
                  fi;
:: (invalid) ->
   goto s1
fi;

 goto again
}
active proctype System()
{ bit invalid, logged, feedback, logout;
short recovery;
start:
     atomic {
              invalid= 0;
               logged  = 1;
               feedback = 1;
               logout = 1;
               recovery = 1;
              goto again
            };
s5:
  atomic {
               recovery = recovery+1;
              goto s6
            };
again: to_system?crt_acc_req;
 to_user!snd_acc_pg;
s1:  to_system?entr_acc_info;
if
::(!invalid)-> to_user!login_req;
            s2: to_system?login_info;
                if
                ::(!invalid)-> to_user!prof_pg;
                               to_system?prof_edit;
                                to_user!profedit_pg;
                                to_system?prof_edit_data;
                                to_user!update_prof;
                                if
                                ::(recovery != 1 ) ->
                                                goto s4
                                ::(recovery == 1)->
                                                  to_system? pass_rcvy_req;
                                                   to_user! pass_rcvy_pg;
                                              s3: to_system? rcvy_data;
```

```
                                        if
                                        ::(!invalid)->
                                                    to_user!login_req;
                                                     goto s5;
                                              s6:  goto s2;
                                        :: (invalid) ->
                                                     goto s3
                                        fi;
                              fi;
                              s4:
                              if
                             ::(feedback)-> to_system? feedback_req;
                                        to_user! feedback_pg;
                                        to_system? feedback_data;
                                        if
                                        ::(logged)-> to_user!successful;
                                        ::(!logged)-> to_user!login_req;
                                                    goto s2;
                                         fi;
                               fi;
                               if
                               ::(logout)->to_system? logout_req;
                                          to_user! destroy_session;
                               fi;
                    :: (invalid) ->
                        goto s1
                     fi;
:: (invalid) ->
   goto s1
fi;
 goto again
}
```

Automata for user general activity

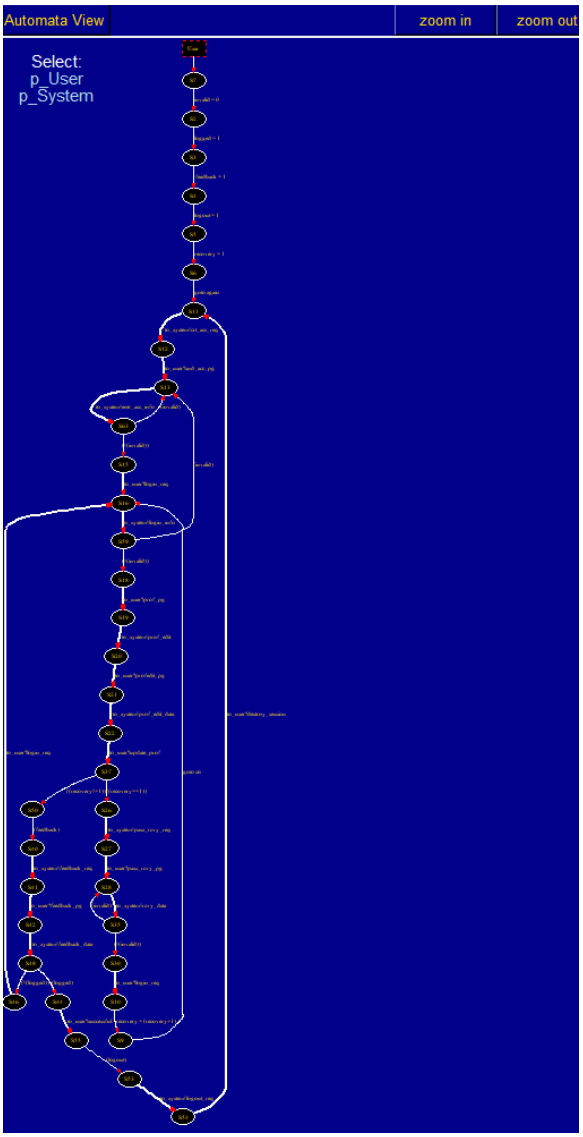

Fig. 2 Automata for user general activity (User)

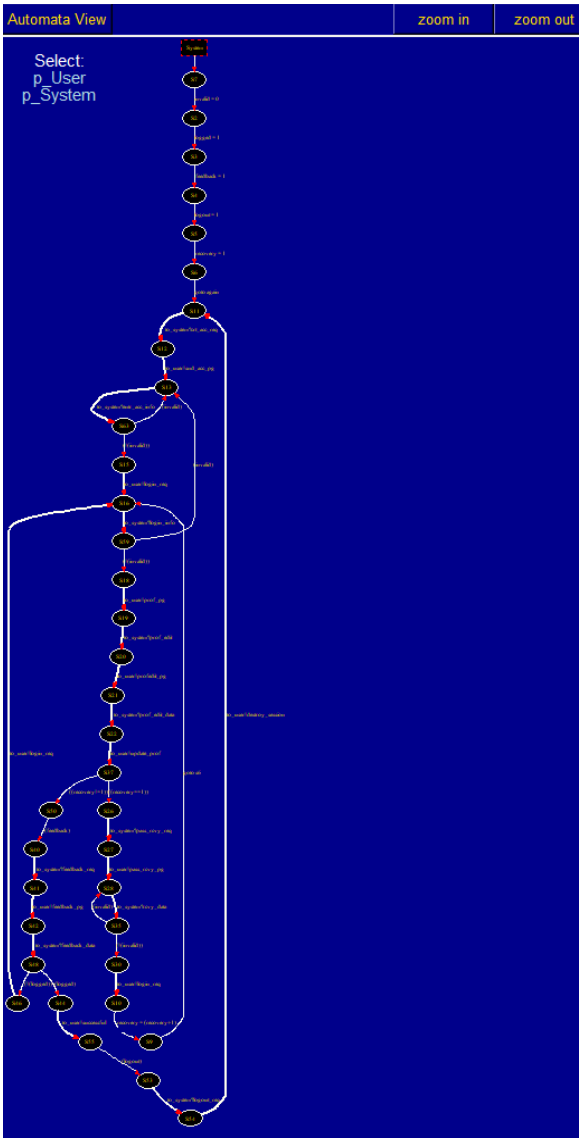

Fig. 3 Automata for user general activity (System)

Process Simulation

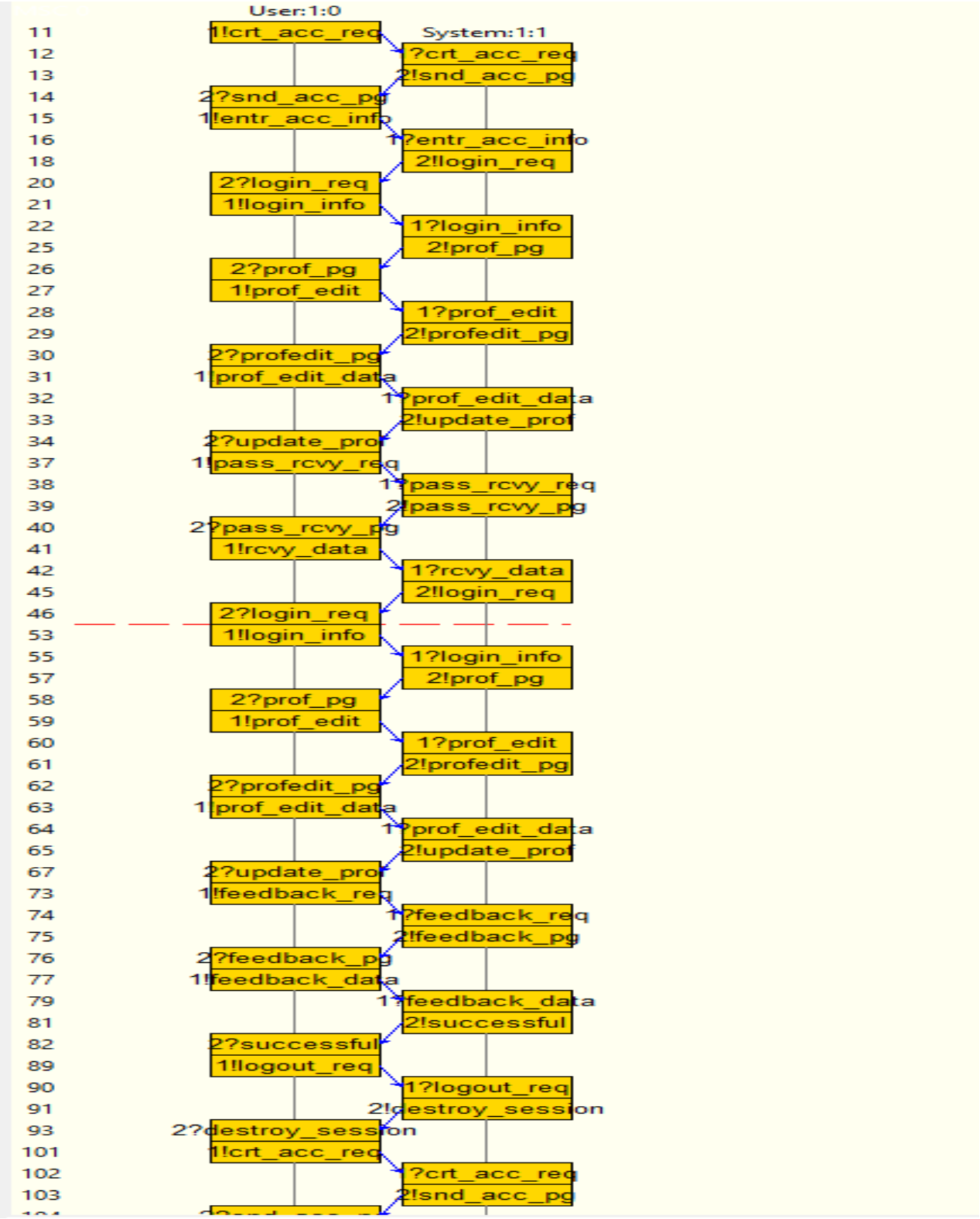

Fig.4 Process Simulation for 1st Sequence Diagram

Verification

```
verification result:
spin -a D:/CSE435/SPIN/Bin/spin651_windows64.exe/main_final.pml
gcc -DMEMLIM=1024 -O2 -DXUSAFE -DSAFETY -DNOCLAIM -w -o pan pan.c
./pan -m10000
Pid: 5336

(Spin Version 6.5.1 -- 20 December 2019)
        + Partial Order Reduction

Full statespace search for:
        never claim         - (not selected)
        assertion violations +
        cycle checks        - (disabled by -DSAFETY)
        invalid end states +

State-vector 40 byte, depth reached 83, errors: 0
       84 states, stored
        1 states, matched
       85 transitions (= stored+matched)
        0 atomic steps
hash conflicts:         0 (resolved)

Stats on memory usage (in Megabytes):
    0.004   equivalent memory usage for states (stored*(State-vector + overhead))
    0.282   actual memory usage for states
   64.000   memory used for hash table (-w24)
    0.343   memory used for DFS stack (-m10000)
   64.539   total actual memory usage


unreached in proctype User
        D:/CSE435/SPIN/Bin/spin651_windows64.exe/main_final.pml:56, state 46, "to_user?login_req"
        D:/CSE435/SPIN/Bin/spin651_windows64.exe/main_final.pml:72, state 66, "-end-"
        (2 of 66 states)
unreached in proctype System
        D:/CSE435/SPIN/Bin/spin651_windows64.exe/main_final.pml:125, state 46, "to_user!login_req"
        D:/CSE435/SPIN/Bin/spin651_windows64.exe/main_final.pml:141, state 66, "-end-"
        (2 of 66 states)

pan: elapsed time 0.001 seconds
No errors found -- did you verify all claims?
```

Fig.5 Verification (1st Sequence Diagram)

Process console:

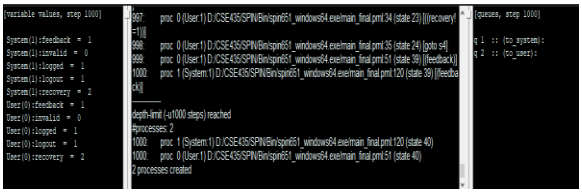

Fig.6 Process Simulation Console (1st Sequence Diagram)

Teaching Service between Student and Teacher

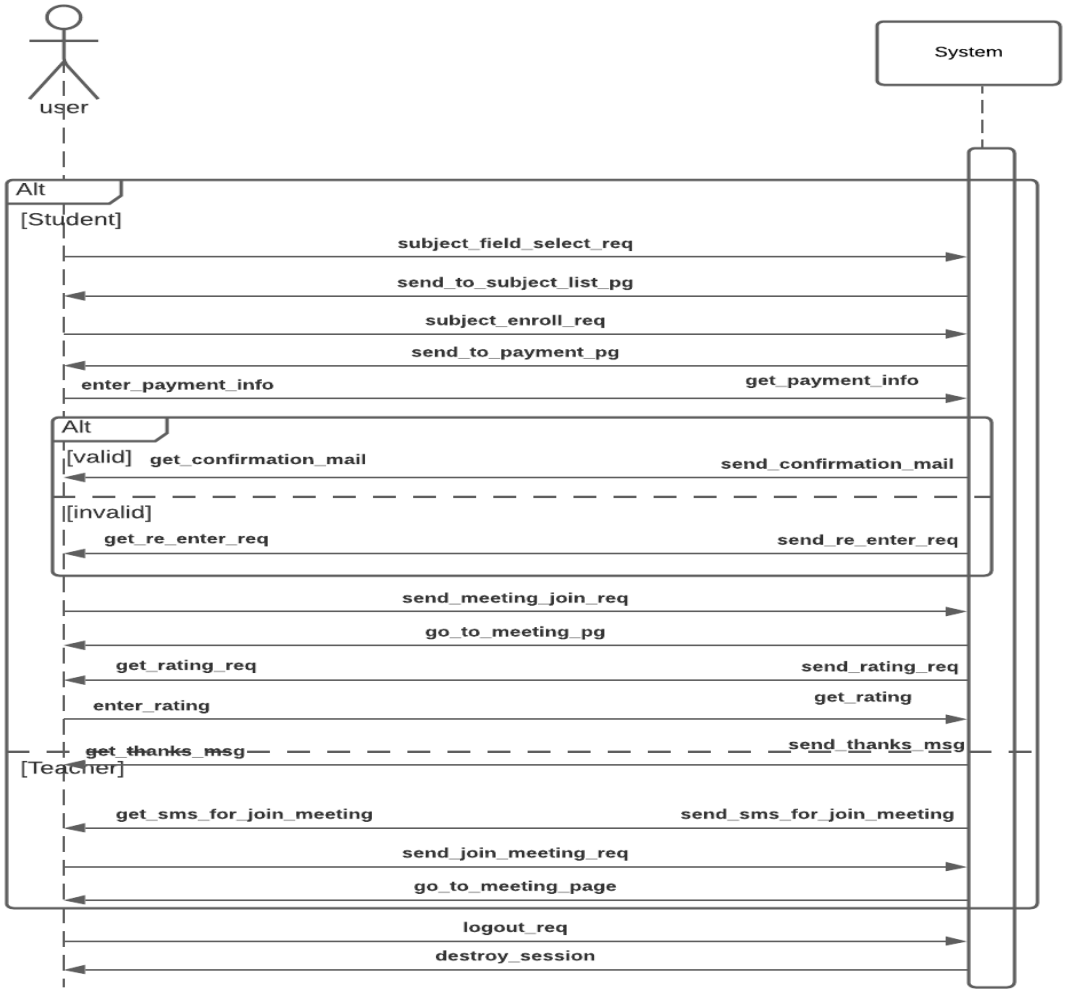

Fig. 7 Sequence diagram for Teaching Service between Student and Teacher

Sequence Diagram Task Description:

There are two users for this web system- student and teacher. Two users have different activities in the web system. A student user can send a subject filed select request to the system and in response from the system, he/she will be sent to the subject list page. On the subject list page, the student user can send enroll request for a subject. In response, the system will send the student user to the payment page where he/she have to fill up the payment info. After the student user enters payment info and the system gets the info, there takes place a validation if the info is correct or not. If the info is incorrect, then the student user will get re-enter request from the system. Otherwise, the student user will get a confirmation mail and a meet link from the system. After getting the link, the user sends a join request to the system and in response go to the meeting page. When the student user finished meeting with the teacher, he/she gets a rating request from the system.

After getting a rating, the system will send the student user a thanking message. Now if the user is a teacher, then he/she will get an SMS for joining the meeting after the student user confirmed payment. The teacher then can send a joining meeting request from the link he gets from his/her profile. The system will take the teacher to the meeting page where he/she meet the student user. The student and teacher anyone can log out or go to the home page after doing any action.

PROMELA code:

```
mtype = { field_select, subject_list, enroll_req, payment_pg, entr_payment_info, confirmed_mail, meeting_join_req, meeting_pg, rating_req, entr_rating, msg, join_meeting_sms, logout_req, destroy_session};

chan to_user = [1] of { mtype };

chan to_system = [1] of { mtype };

active proctype User()
{ bit invalid, student, teacher;
start:
    atomic {
            invalid          = 0;
              student  = 1;
              teacher = 1;
            goto again
          };
again:
if
::(student)-> to_system! field_select;
                to_user?subject_list;
                to_system!enroll_req;
                to_user?payment_pg;
           s1: to_system!entr_payment_info;
                if
                ::(!invalid)-> to_user?confirmed_mail;
                                to_system!meeting_join_req;
                                to_user?meeting_pg;
                                to_user?rating_req;
                                to_system!entr_rating;
                                to_user?msg;
                 :: (invalid) ->
                                goto s1
                  fi;
fi;

if
::(teacher)-> to_user?join_meeting_sms;
                to_system!meeting_join_req;
```

```
                to_user?meeting_pg;
fi;
to_system! logout_req;
to_user? destroy_session;
 goto again
}
active proctype System()
{ bit invalid, student, teacher;
start:
     atomic {
              invalid          = 0;
                student  = 1;
                teacher = 1;
              goto again
            };
again:
if
::(student)-> to_system? field_select;
                to_user!subject_list;
                to_system?enroll_req;
                to_user!payment_pg;
             s1: to_system?entr_payment_info;
                if
                ::(!invalid)-> to_user!confirmed_mail;
                                to_system?meeting_join_req;
                                to_user!meeting_pg;
                                to_user!rating_req;
                                to_system?entr_rating;
                                to_user!msg;
                 :: (invalid) ->
                                goto s1
                  fi;
fi;

if
::(teacher)-> to_user!join_meeting_sms;
                to_system?meeting_join_req;
                to_user!meeting_pg;
fi;
to_system? logout_req;
to_user! destroy_session;
 goto again
}
```

Automata for teaching service between Student and Teacher

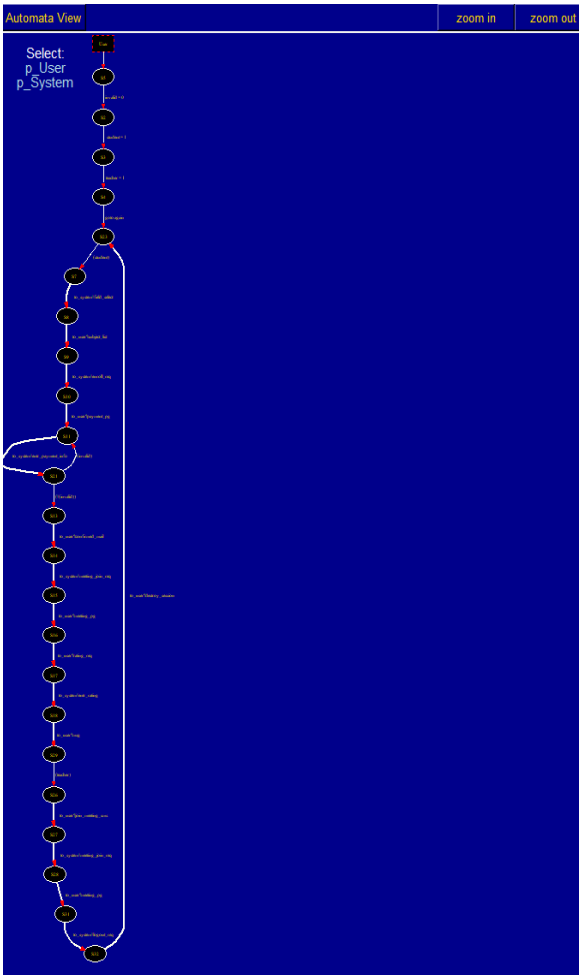

Fig. 8 Automata for teaching service between Student and Teacher (User)

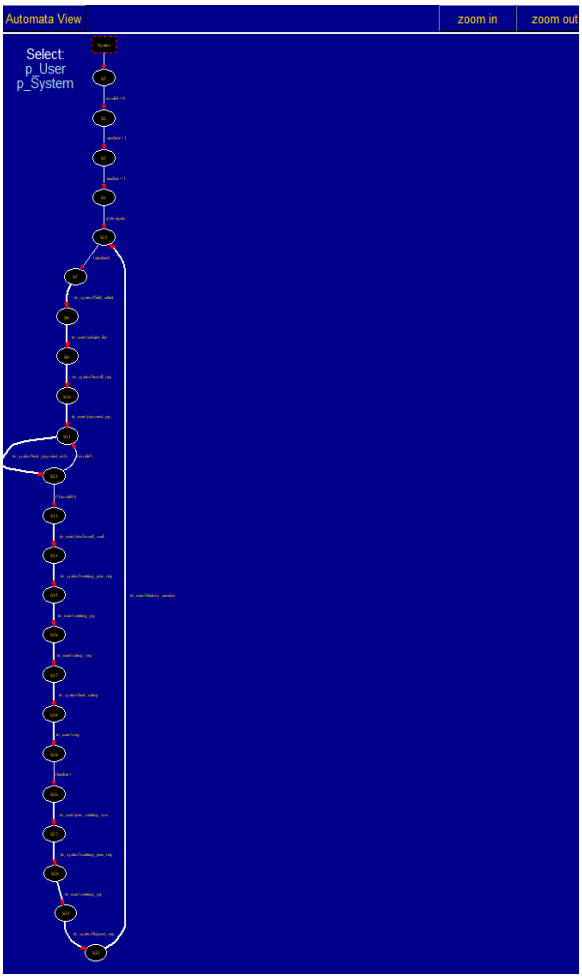

Fig. 9 Automata for teaching service between Student and Teacher (System)

Process Simulation:

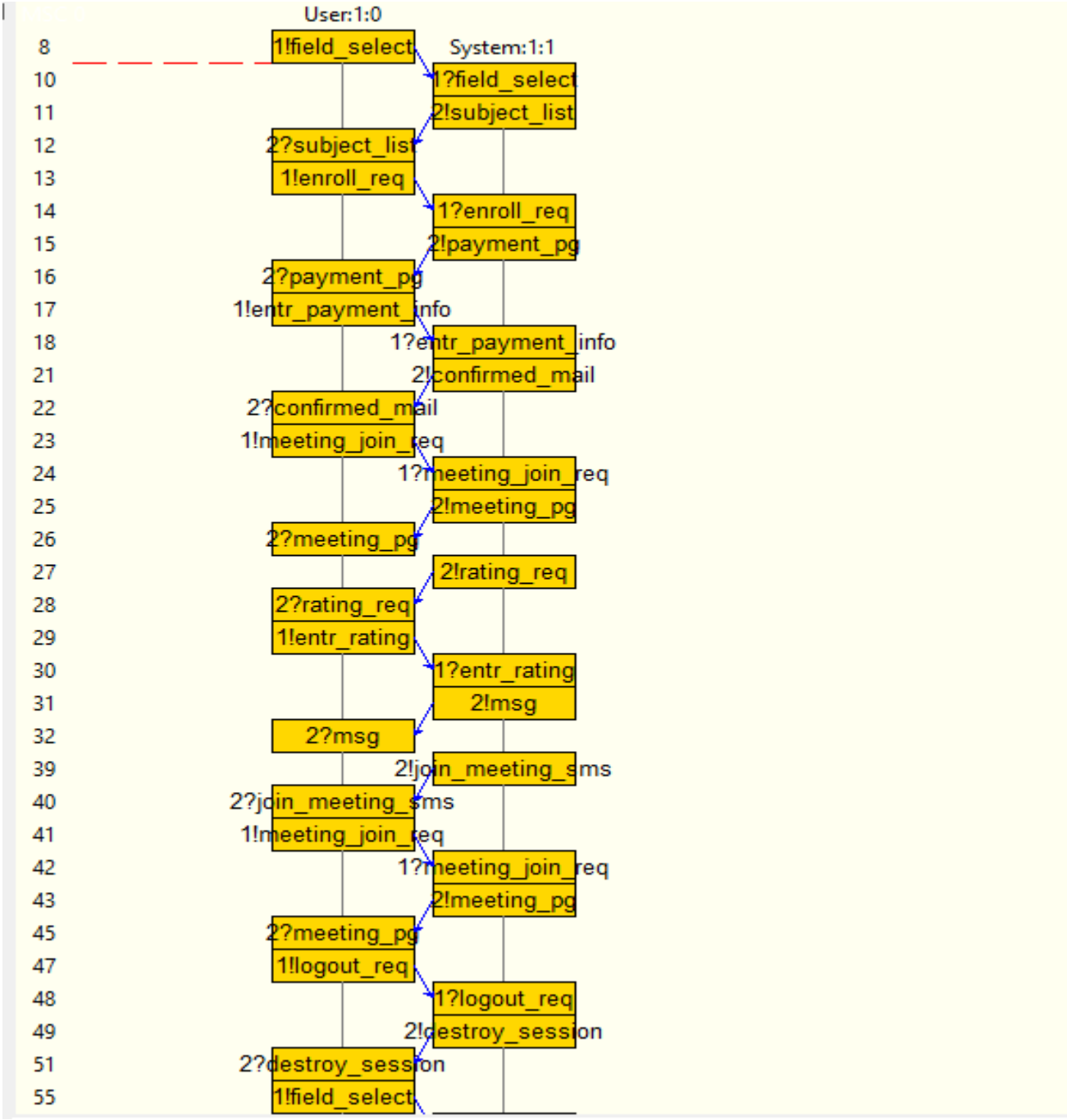

Fig.10 Process Simulation for 2nd Sequence Diagram

Verification

```
verification result:
spin -a  teacher.pml
gcc -DMEMLIM=1024 -O2 -DXUSAFE -DSAFETY -DNOCLAIM -w -o pan pan.c
./pan -m10000
Pid: 15024

(Spin Version 6.5.1 -- 20 December 2019)
        + Partial Order Reduction

Full statespace search for:
        never claim         - (not selected)
        assertion violations +
        cycle checks        - (disabled by -DSAFETY)
        invalid end states +

State-vector 32 byte, depth reached 40, errors: 0
     44 states, stored
      3 states, matched
     47 transitions (= stored+matched)
      0 atomic steps
hash conflicts:         0 (resolved)

Stats on memory usage (in Megabytes):
   0.002   equivalent memory usage for states (stored*(State-vector + overhead))
   0.288   actual memory usage for states
  64.000   memory used for hash table (-w24)
   0.343   memory used for DFS stack (-m10000)
  64.539   total actual memory usage


unreached in proctype User
        teacher.pml:40, state 34, "-end-"
        (1 of 34 states)
unreached in proctype System
        teacher.pml:77, state 34, "-end-"
        (1 of 34 states)

pan: elapsed time 0 seconds
No errors found -- did you verify all claims?
```

Fig.11 Verification (2nd Sequence Diagram)

Process console:

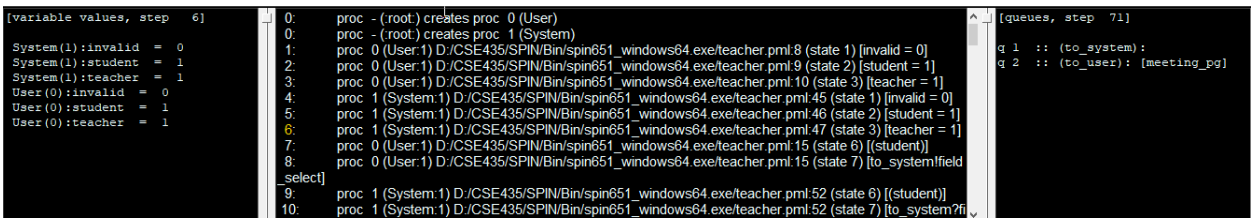

Fig.12 Process Simulation Console (2nd Sequence Diagram)

Admin Activity:

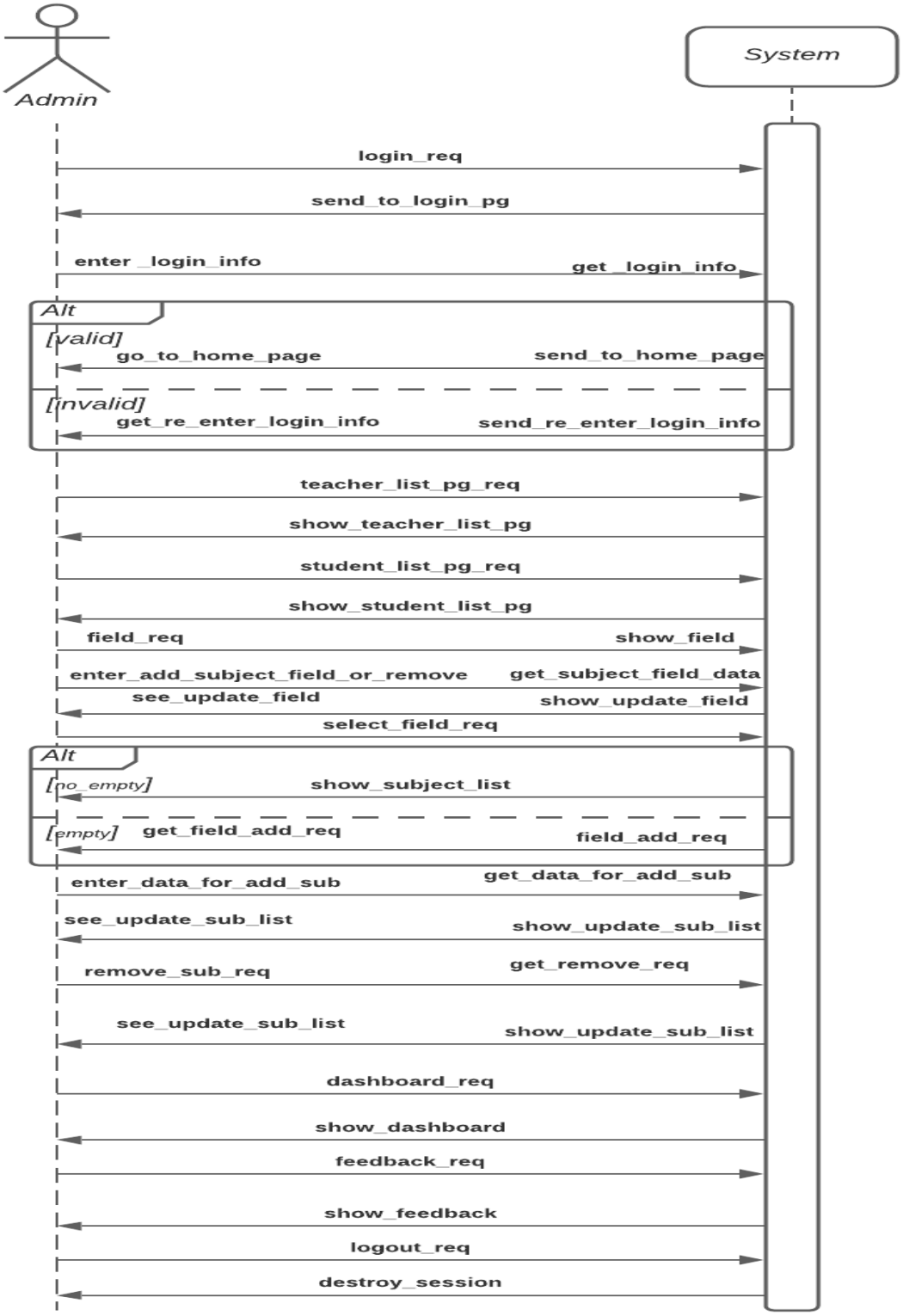

Fig. 13 Sequence diagram for Admin activity.

Sequence Diagram Task Description:

The admin of this web system has already created account. So, when the admin wants to do any activity, he/she will send a login request to the system. In response, the user gets into the login page where he/she enter the login info. When the system gets the login info from the user, it will check if the info is valid or invalid. If it is valid, then the user will be sent to the home page otherwise he/she will get re-enter login info message from the system. The admin can send teacher or student list page requests to the system to show the information of all teachers and students in the web system. The admin can send a field request to the system to add or remove any subject field for the user. After updating the subject field, the system will show the updated field to the admin. When the admin request for Select field, he/she will see a subject field list. If there is no subject field in the list, then the admin will get a request to add a subject field from the system. The subject field can only have subfields. So, the admin can enter a subfield to add or remove only if the subject field already exists. The system will show the updated sub-field list to the admin whenever there is an action for this field. The activity of admin like add or remove subject field will update the web-system for both user and admin. Admin can request to see the dashboard and feedback to the user. Additionally, need to mention, Admin can send a logout request after every activity of his/her and in response, the system will destroy the session.

PROMELA code:

```
mtype = { login_req, login_pg, login_info, prof_pg, teacher_list_req, teacher_list, student_list_req,
student_list, field_req, add_or_remove_field, updated_field, select_field, subject_list, add_subject,
updated_subject_list, remove_subject, dashboard_req, dashboard, feedback_req, feedback, logout_req,
destroy_session};

chan to_user = [1] of { mtype };

chan to_system = [1] of { mtype };

active proctype User()
{ bit invalid, empty_f;
start:
    atomic {
            invalid = 0;
             empty_f  = 0;
            goto again
          };
again:
to_system!login_req;
to_user?login_pg;
s1: to_system!login_info;
if
::(!invalid)-> to_user?prof_pg;
              to_system!teacher_list_req;
               to_user?teacher_list;
              to_system!student_list_req;
```

```
            to_user?student_list;
            to_system!field_req;
        s2: to_system!add_or_remove_field;
            to_user? updated_field;
            to_system!select_field;
             if
             ::(!empty_f)-> to_user?subject_list;
                            to_system!add_subject;
                            to_user?updated_subject_list;
                            to_system!remove_subject;
                            to_user?updated_subject_list;
             :: (empty_f) ->
                             goto s2
              fi;
              to_system! dashboard_req;
              to_user? dashboard;
              to_system! feedback_req;
              to_user? feedback;
              to_system! logout_req;
              to_user? destroy_session;

:: (invalid) ->
   goto s1
fi;
 goto again
}
active proctype System()
{ bit invalid, empty_f;
start:
     atomic {
            invalid = 0;
             empty_f  = 0;
            goto again
           };
again:
to_system?login_req;
to_user!login_pg;
s1: to_system?login_info;
if
::(!invalid)-> to_user!prof_pg;
              to_system?teacher_list_req;
               to_user!teacher_list;
              to_system?student_list_req;
               to_user!student_list;
```

```
            to_system?field_req;
        s2: to_system?add_or_remove_field;
            to_user! updated_field;
            to_system?select_field;
             if
             ::(!empty_f)-> to_user!subject_list;
                            to_system?add_subject;
                            to_user!updated_subject_list;
                            to_system?remove_subject;
                            to_user!updated_subject_list;
             :: (empty_f) ->
                             goto s2
             fi;
             to_system? dashboard_req;
             to_user! dashboard;
             to_system? feedback_req;
             to_user! feedback;
             to_system? logout_req;
             to_user! destroy_session;

:: (invalid) ->
   goto s1
fi;
 goto again
}
```

Automata for Admin activity

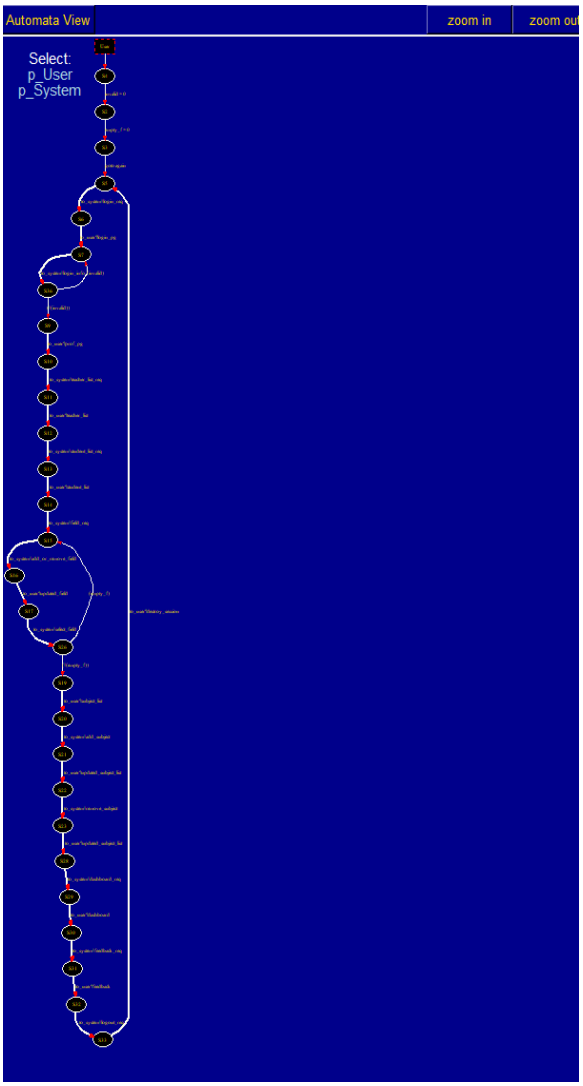

Fig. 14 Automata for Admin activity (User)

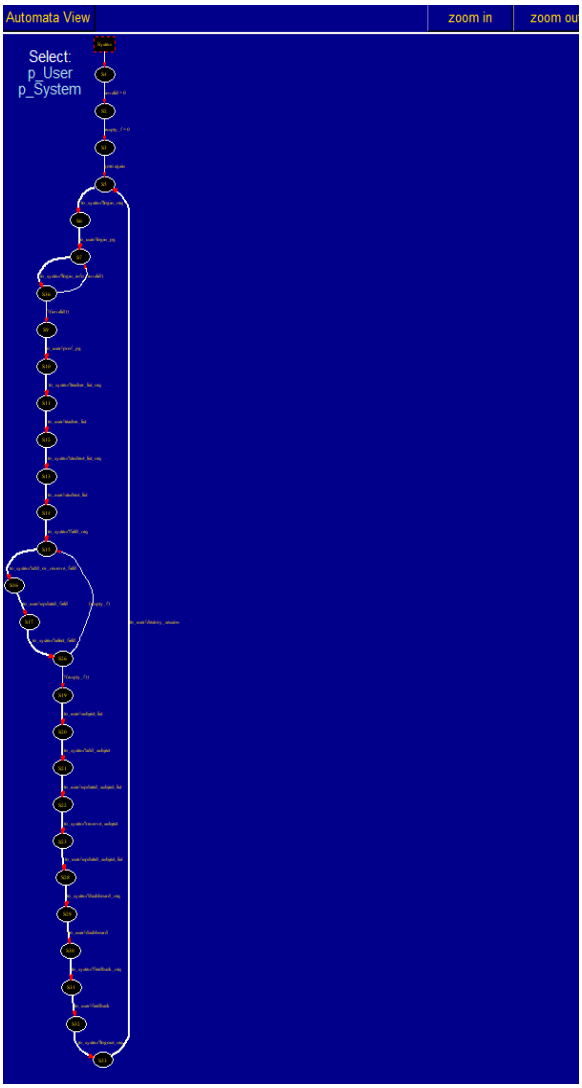

Fig. 15 Automata for Admin activity (System)

Process Simulation:

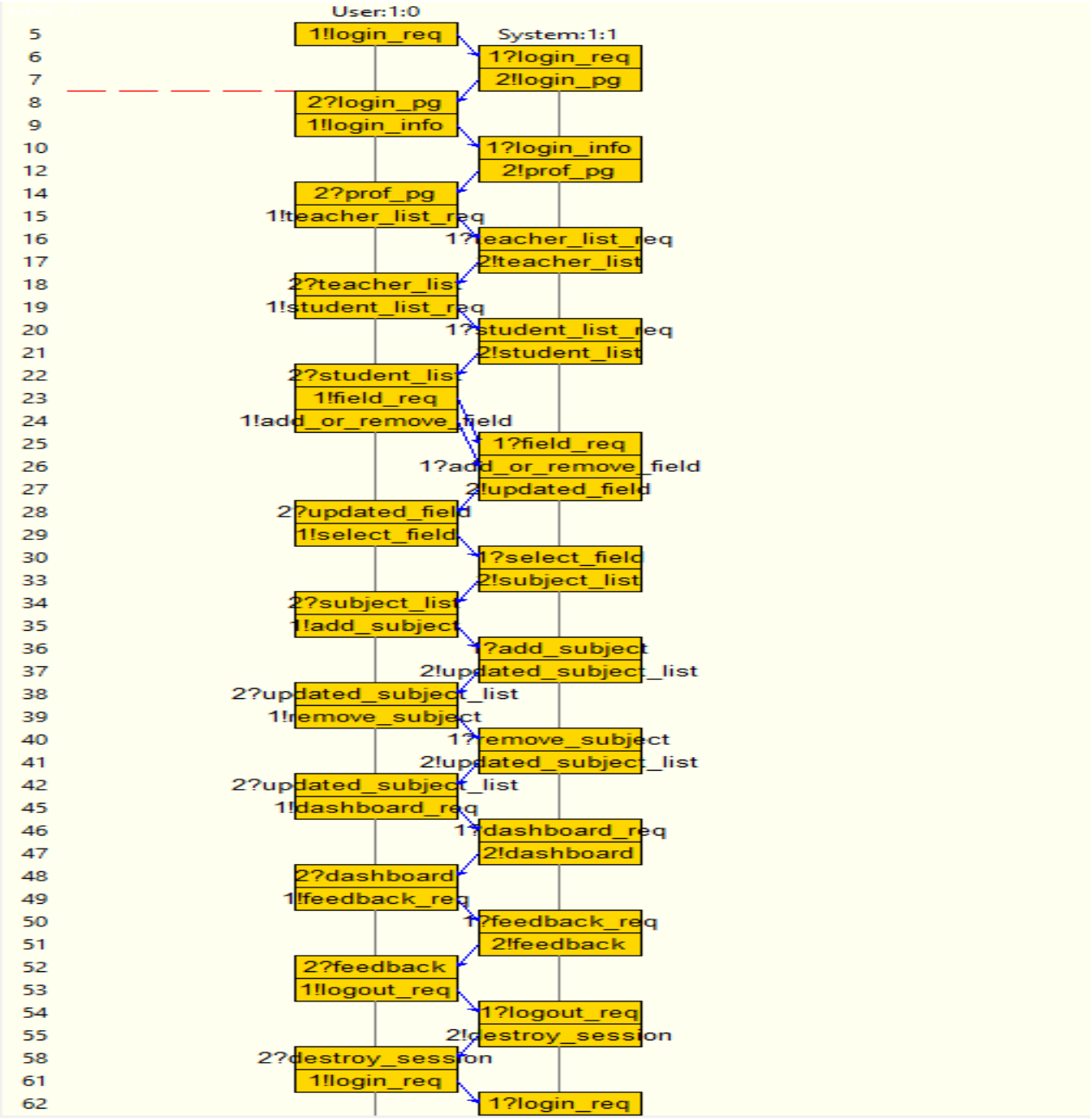

Fig.16 Process Simulation for 3rd Sequence Diagram

Verification

```
verification result:
spin -a  admin.pml
gcc -DMEMLIM=1024 -O2 -DXUSAFE -DSAFETY -DNOCLAIM -w -o pan pan.c
./pan -m10000
Pid: 11968

(Spin Version 6.5.1 -- 20 December 2019)
        + Partial Order Reduction

Full statespace search for:
        never claim         - (not selected)
        assertion violations +
        cycle checks        - (disabled by -DSAFETY)
        invalid end states +

State-vector 32 byte, depth reached 51, errors: 0
     53 states, stored
      2 states, matched
     55 transitions (= stored+matched)
      0 atomic steps
hash conflicts:         0 (resolved)

Stats on memory usage (in Megabytes):
   0.002   equivalent memory usage for states (stored*(State-vector + overhead))
   0.287   actual memory usage for states
  64.000   memory used for hash table (-w24)
   0.343   memory used for DFS stack (-m10000)
  64.539   total actual memory usage


unreached in proctype User
        admin.pml:46, state 39, "-end-"
        (1 of 39 states)
unreached in proctype System
        admin.pml:89, state 39, "-end-"
        (1 of 39 states)

pan: elapsed time 0 seconds
No errors found -- did you verify all claims?
```

Fig.17 Verification (3rd Sequence Diagram)

Process console:

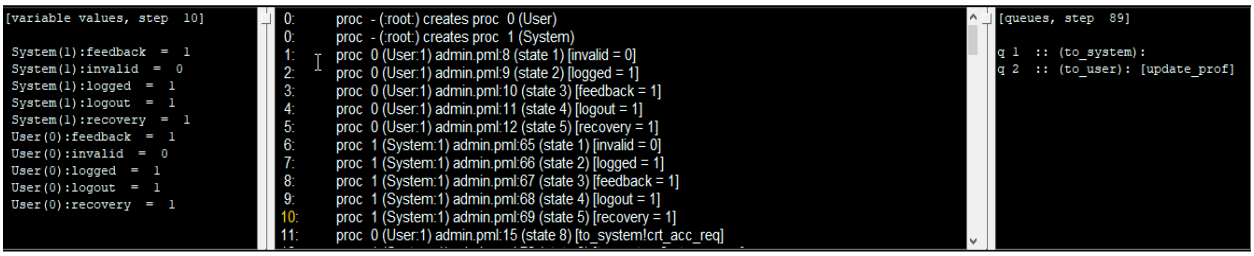

Fig.18 Process Simulation Console (3rd Sequence Diagram)

---------- **End** ----------