EAST WEST UNIVERSITY

Course Name: Software Quality Assurance

Course Code: CSE435(sec-3)

Summer21

Project Report Part-2

Submitted to:

Dr. Shamim H Ripon

Professor,Department of CSE

East West University

Submitted By:

1.Najia Anjum (ID: 2018-1-60-125)

2. Nayma Alam (ID: 2018-1-60-180)

3. Aftab Miraj Swachchha (ID:2017-2-60-074)

EWU-CONNECT

Project Description:

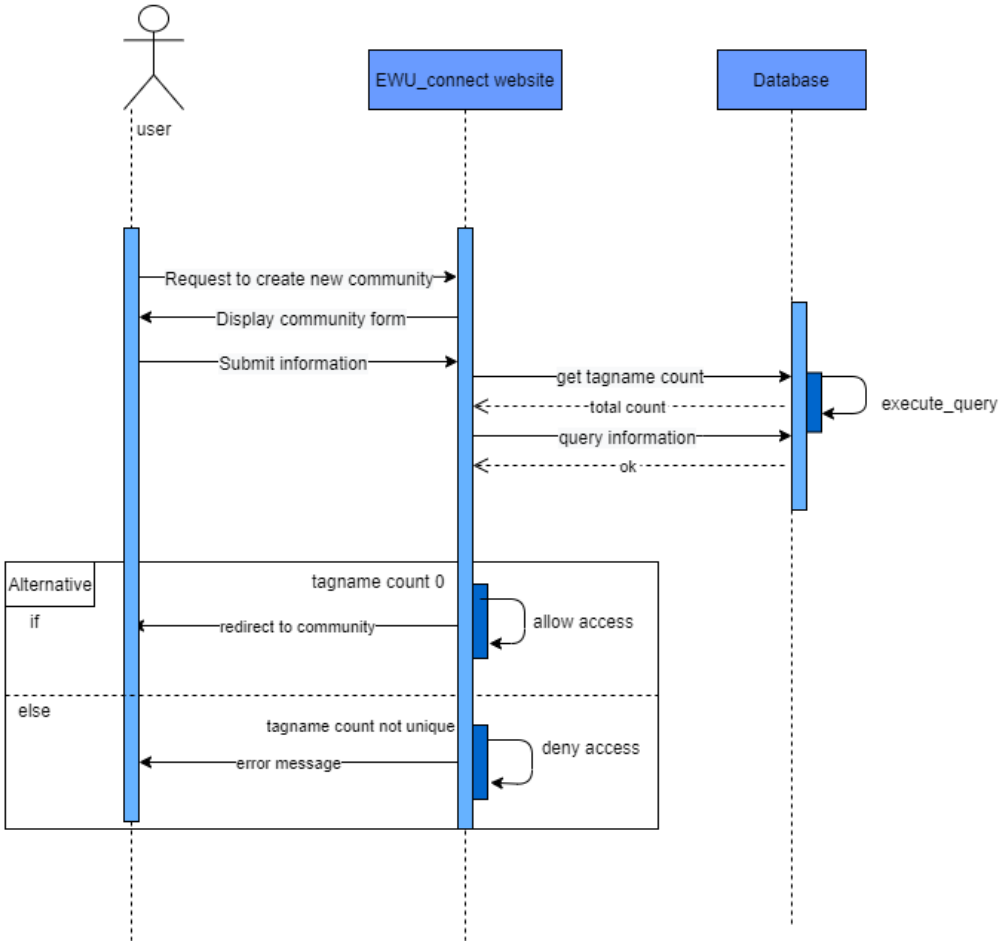
EWU CONNECT is a virtual community platform designed exclusively for the students of East West University where they can connect with their faculties, clubs, communities and the university as a whole. Only EWU members can be registered and registered members can submit links, text posts, images. which are then voted up or down by other members in the same community. Posts are arranged by tops into a user-created group called "community" which cover up many topics like news, EWU club activity, events, movies, video games, music, books and image-sharing. Post that has more up-votes appears at the top of their community, if the post receives enough upvotes, eventually on the site's front page. Even with strict rules prohibiting harassment, Community administrators can moderate the site.

Sequence Diagrams:

1.Sequence Diagram of Create Community:

User requests to create a new community, then the website will display community form. After submitting information, the website will request tagname count to the database. Then the database will execute a query for tagname and return it to the website. After getting the tagname website will query for saving new communities to the database, then the database will save the information and return ok requests to the website. If the tagname count is equal to 0 the website will allow user access and redirect user to community, else tagname count is not unique, website will deny access and show error message to the user.

Create community



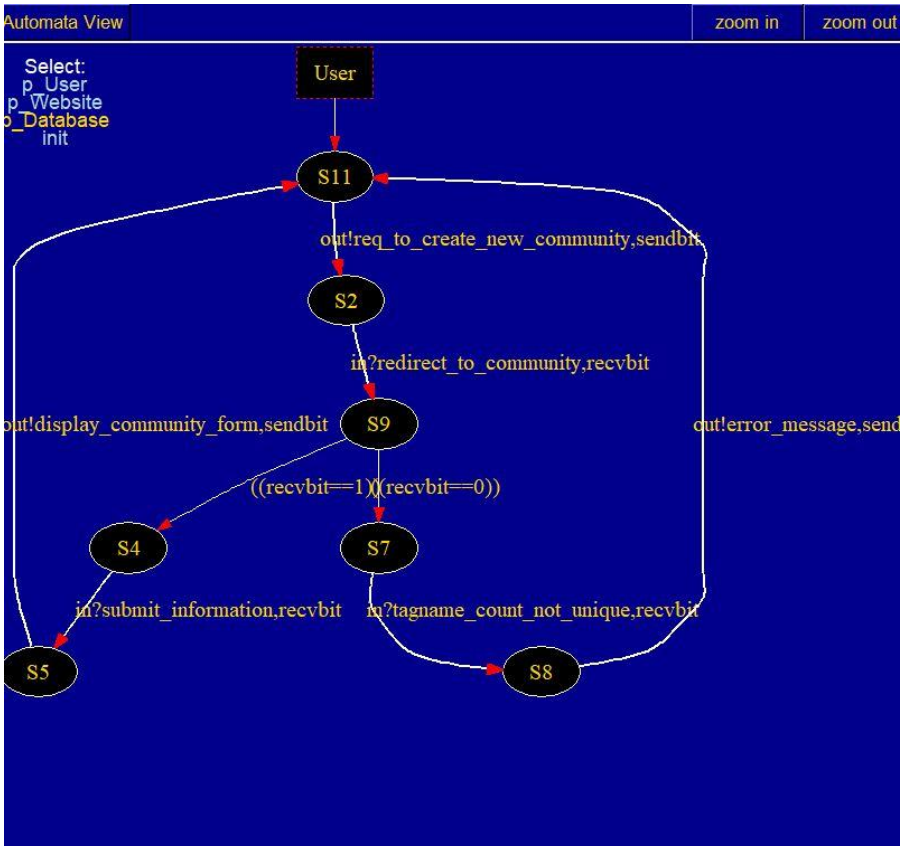
PROMELA Code:

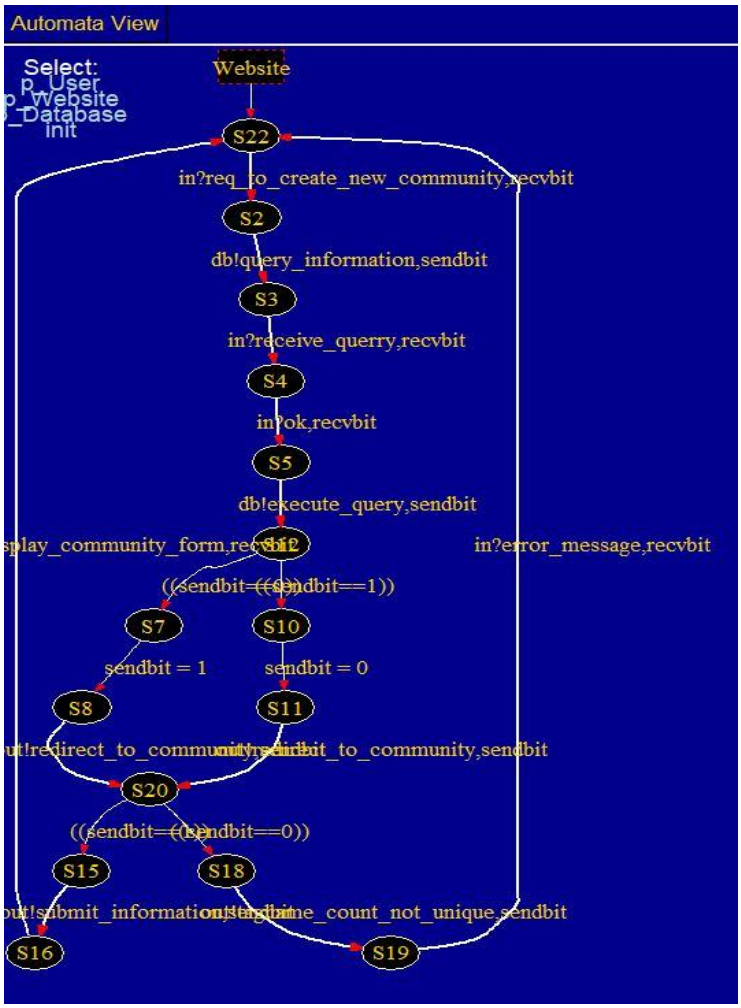
mtype = {req_to_create_new_community, display_community_form, submit_information, error_message, tagname_count, query_information, execute_query,redirect_to_community, allow_access, tagname_count_not_unique, receive_querry, website_login, ok, deny_access};

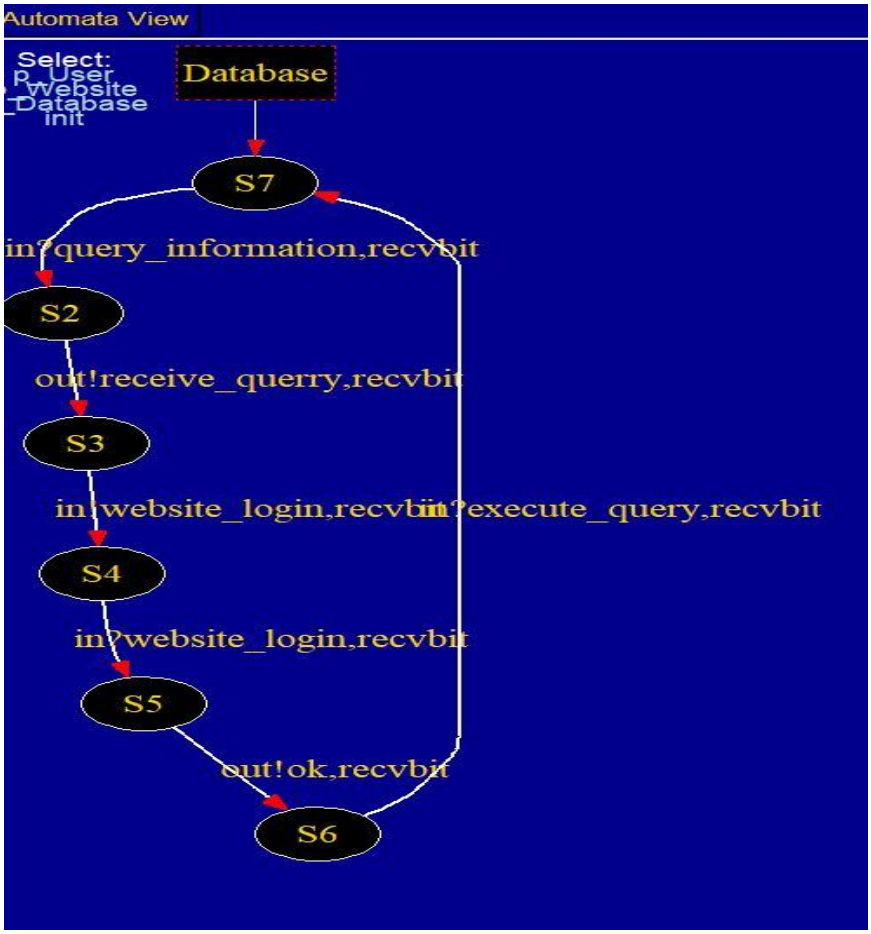
```
chan toUser = [1] of {mtype,bit};
chan toWebsite = [1] of {mtype,bit};
chan toDatabase = [1] of {mtype,bit};
bool success = 1;
```

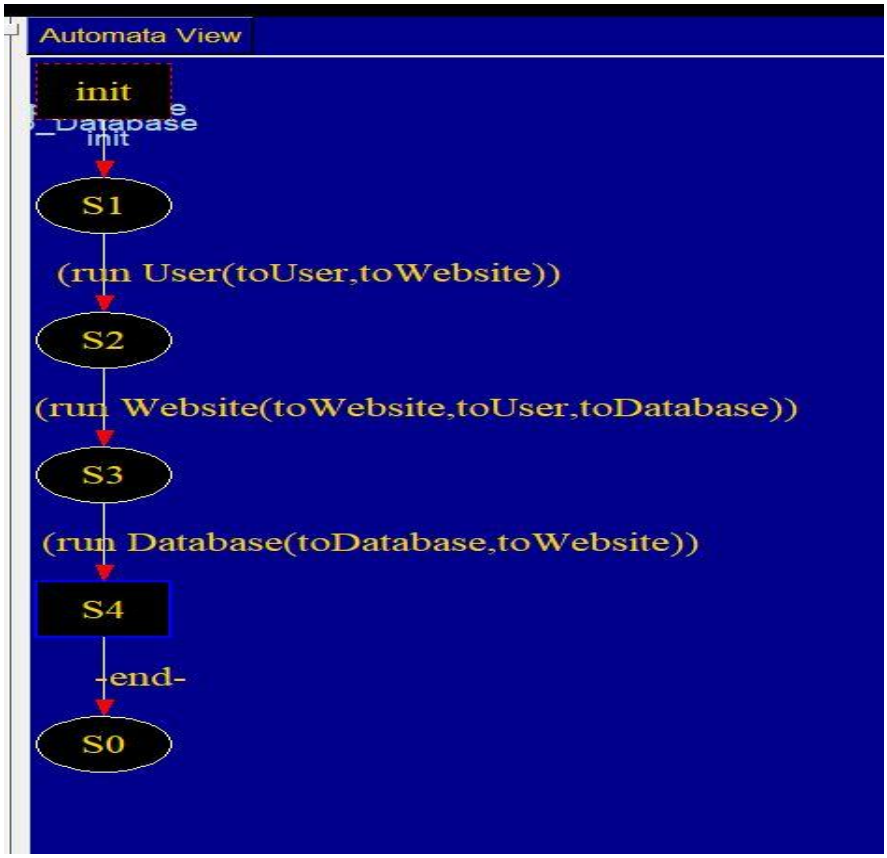
```
proctype User(chan in, out)
{
   bit sendbit, recvbit;
   do
   :: out ! req_to_create_new_community, sendbit;
   in ?  redirect_to_community, recvbit;
   if
   :: recvbit == 1 -> in ? submit_information, recvbit ->
      out ! display_community_form, sendbit;
   :: recvbit == 0 -> in ? tagname_count_not_unique, recvbit ->
      out ! error_message, sendbit;
   fi
   od
}
proctype Website(chan in, out, db)
{
   bit sendbit, recvbit;
   do::
   in ? req_to_create_new_community(recvbit);
   db ! query_information, sendbit ->
   in ? receive_querry, recvbit;
   in ? ok, recvbit ->
   db ! execute_query, sendbit;
   if
   :: sendbit == 0 ->
   sendbit = 1; out !  redirect_to_community(sendbit);
   :: sendbit == 1 ->
   sendbit = 0; out !  redirect_to_community(sendbit);
   fi
   if
   :: sendbit == 1 -> out ! submit_information(sendbit) ->
```

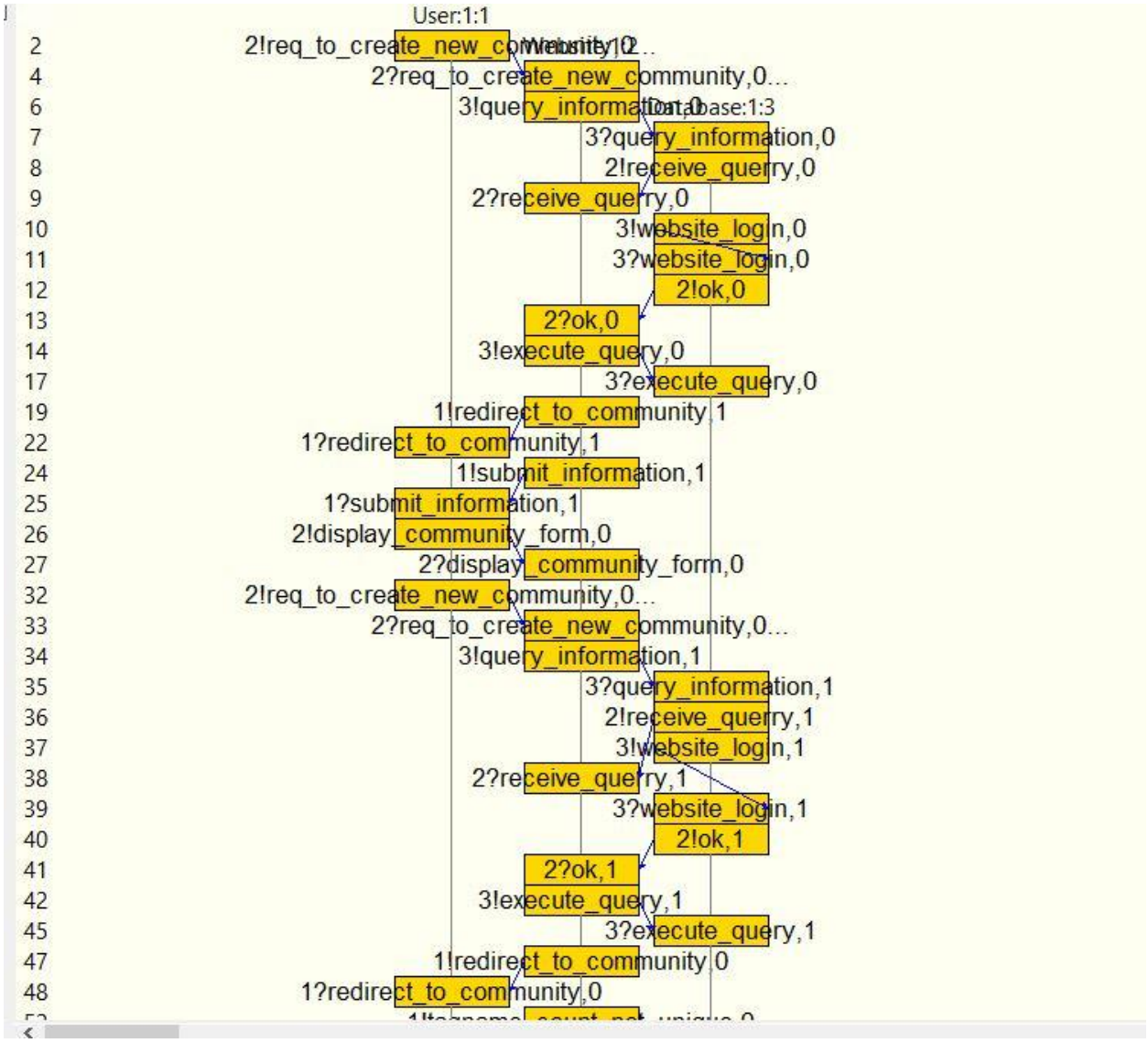
```
    in ? display_community_form(recvbit);
    :: sendbit == 0 -> out ! tagname_count_not_unique(sendbit) ->
    in ? error_message(recvbit);
    fi
    od
  }
  proctype Database(chan in, out)
{
    bit recvbit, sendbit;
    do::
    in ? query_information(recvbit) ->
    out ! receive_querry(recvbit);
    in ! website_login(recvbit);
    in ? website_login(recvbit);
    out ! ok(recvbit) ->
    in ? execute_query(recvbit);
    od
  }
  init
  {
    run User(toUser, toWebsite);
    run Website(toWebsite, toUser, toDatabase);
    run Database(toDatabase, toWebsite);
  }
```


Automata View
zoom in
zoom out
Select:
p_User
p_Website
p_Database
init
User
S11
out!req_to_create_new_community,sendbit
S2
in?redirect_to_community,recvbit
out!display_community_form,sendbit
S9
out!error_message,send
((recvbit==1)(recvbit==0))
S4
S7
in?submit_information,recvbit
in?tagname_count_not_unique,recvbit
S5
S8


Automata View
Select:
p_User
p_Website
_Database
init
Website
S22
in?req_to_create_new_community,recvbit
S2
db!query_information,sendbit
S3
in?receive_querry,recvbit
S4
in?ok,recvbit
S5
db!execute_query,sendbit
in?error_message,recvbit
((sendbit==0))
((sendbit==1))
S7
S10
sendbit = 1
sendbit = 0
S8
S11
S20
((sendbit==0))
((sendbit==0))
S15
S18
S16
S19


Automata View
Select:
p_User
_Website
_Database
init
Database
S7
in?query_information,recvbit
S2
out!receive_querry,recvbit
S3
in!website_login,recvbit
in?execute_query,recvbit
S4
in?website_login,recvbit
S5
out!ok,recvbit
S6

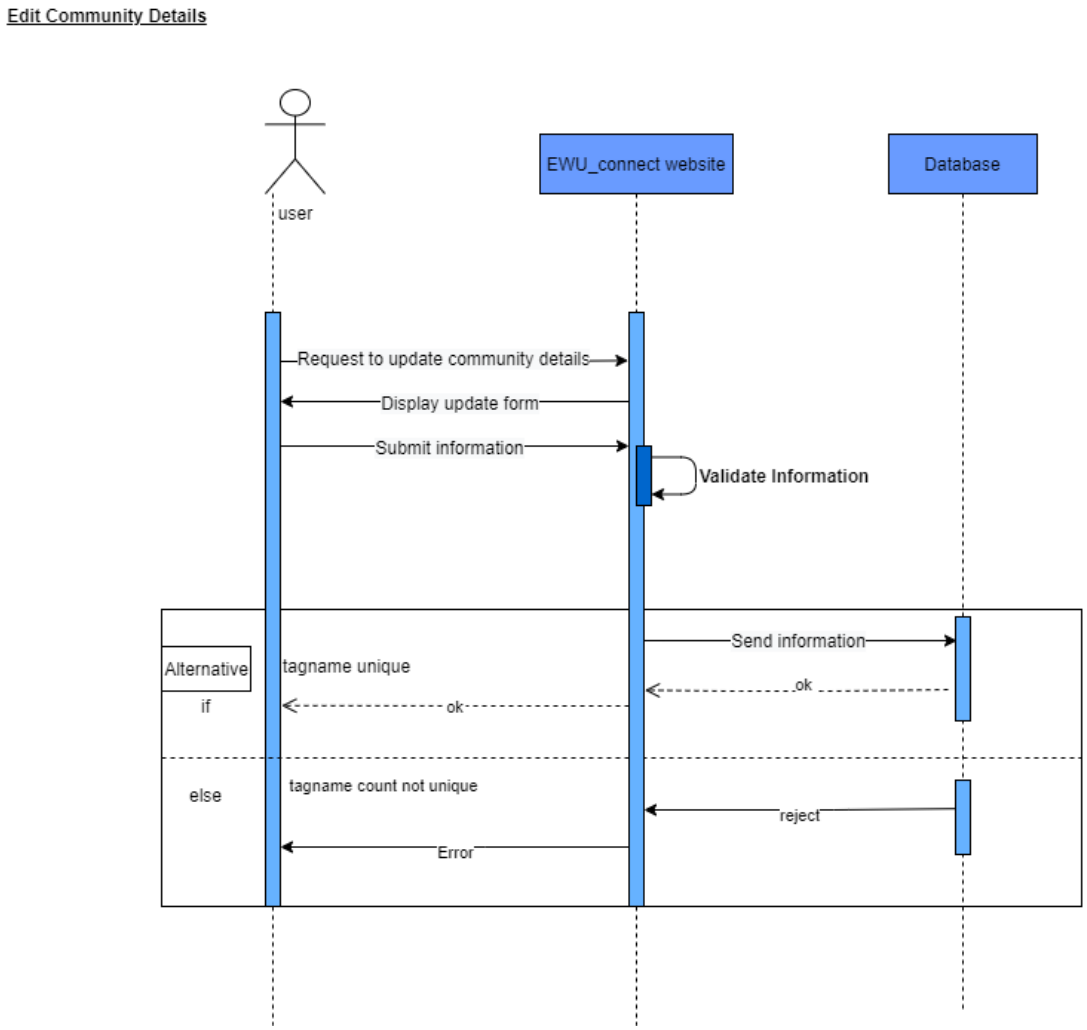

Automata View
init
init
S1
(run User(toUser,toWebsite))
S2
(run Website(toWebsite,toUser,toDatabase))
S3
(run Database(toDatabase,toWebsite))
S4
-end-
S0



2.Sequence Diagram of Edit Community Details:

User requests to update the community database to the website, it will display the form. After submitting information, the website will validate that information. If the tagname is unique, the website will send information to the database. Then the database will store the data and return an

‘ok’ response to the website and it will show an ‘ok’ message to the user. If the tagname count is not unique, the website will show an error message to the user.



PROMELA code:

mtype = {req_to_update_community, display_update_form , submit_information, validate_info, receive_input, tag_name_unique,send_confirnation_user, database_update, send_confirmation_website, send_error}

chan toUser = [1] of {mtype,bit};

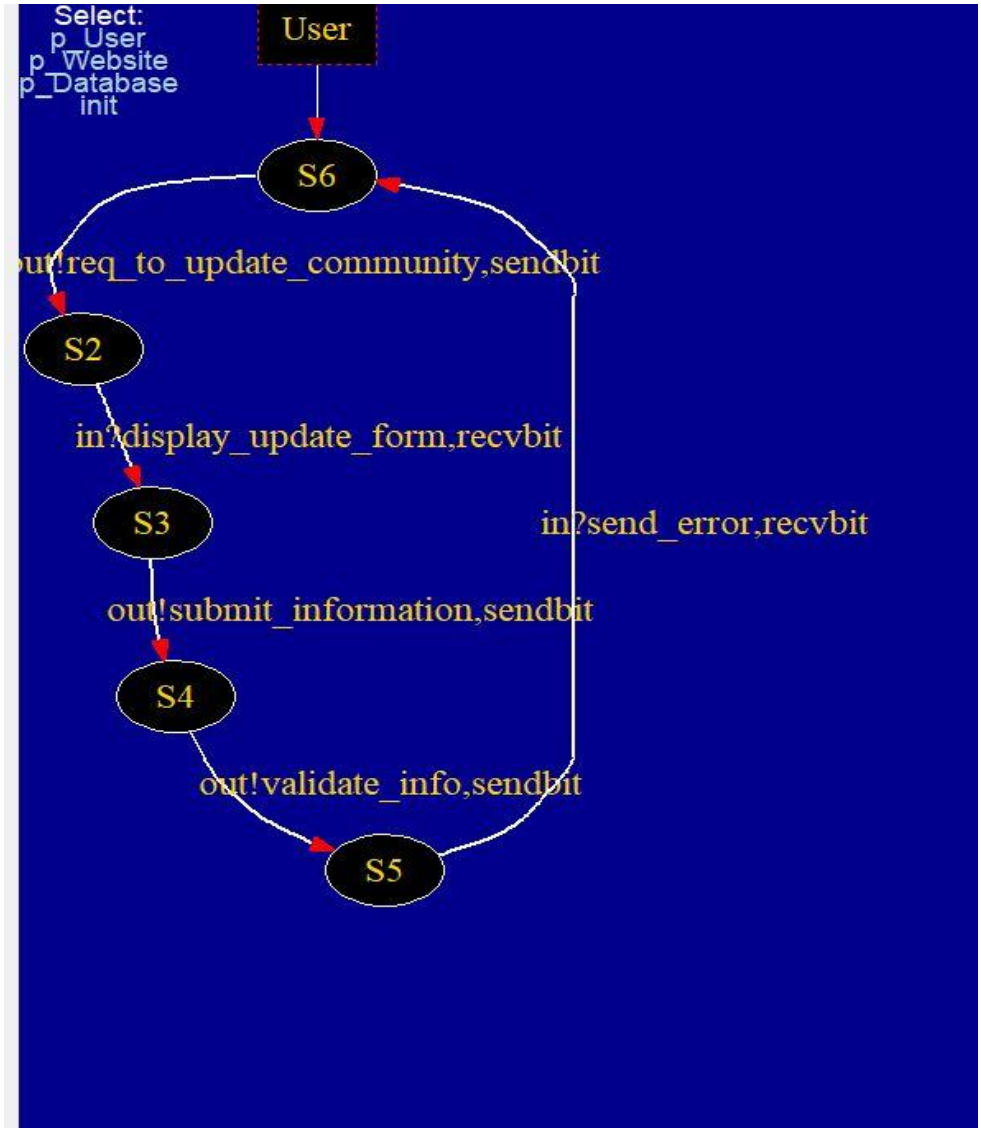
chan toWebsite = [1] of {mtype,bit};

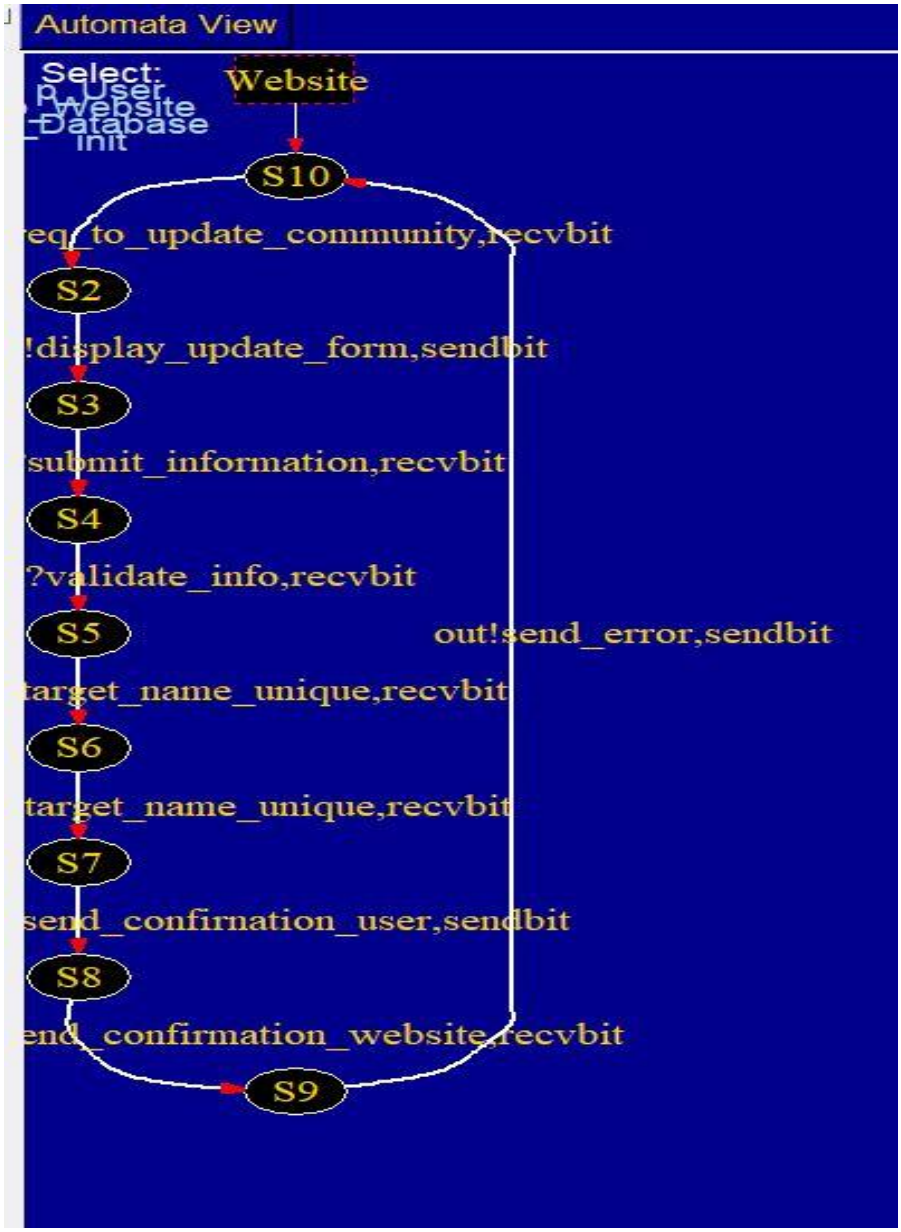
chan toDatabase = [1] of {mtype,bit};

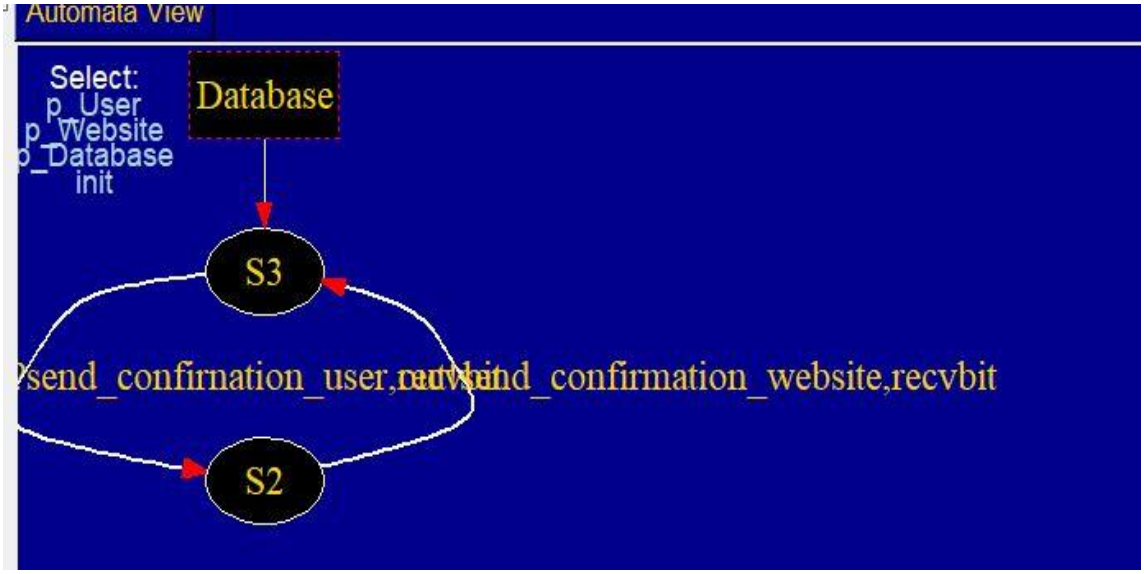
proctype User(chan in, out)

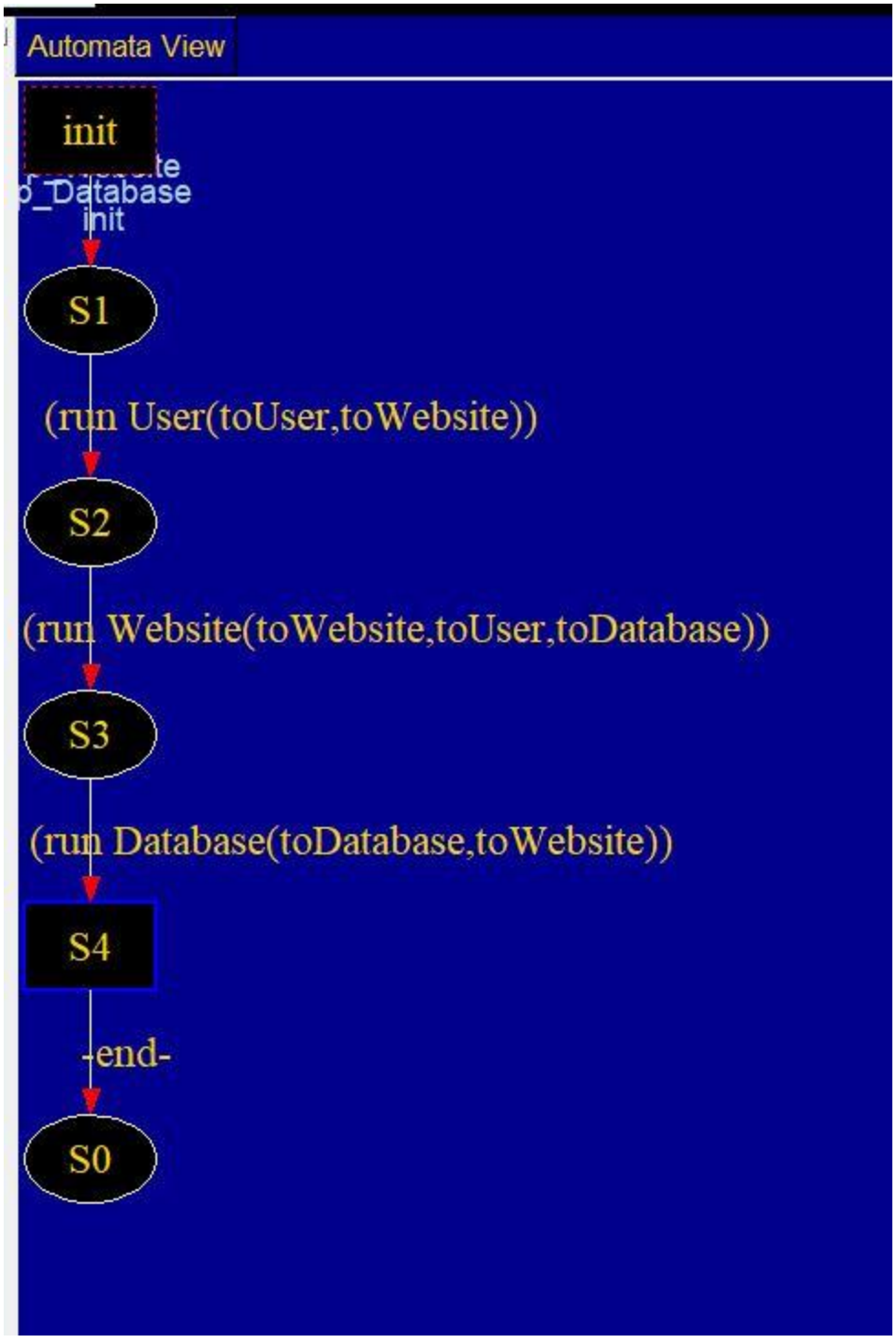
```
{
   bit sendbit, recvbit;

   do
   :: out ! req_to_update_community, sendbit;
   in ? display_update_form, recvbit;
   out ! submit_information, sendbit;
   out ! validate_info, sendbit;
   in ? send_error, recvbit;
   od

}
proctype Website(chan in, out, db)
{
   bit sendbit, recvbit;

   do::
   in ? req_to_update_community(recvbit);
   out ! display_update_form(sendbit);
   in ? submit_information(recvbit);
   in ? validate_info(recvbit);
   in ! target_name_unique(recvbit);
   in ? target_name_unique(recvbit);
   db ! send_confirnation_user, sendbit ->
   in ? send_confirmation_website, recvbit;
   out ! send_error(sendbit);

   od

}
proctype Database(chan in, out)
```

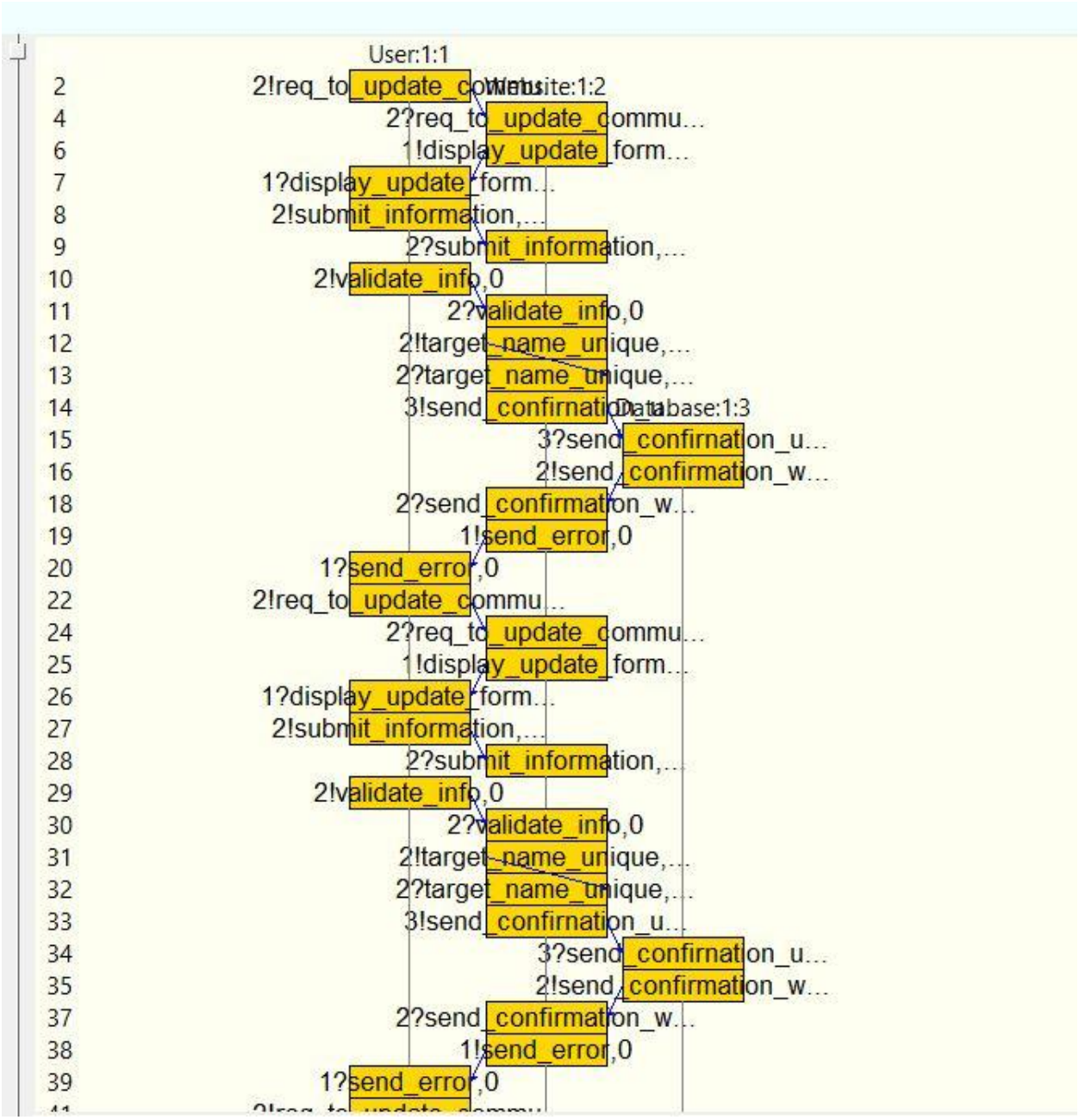
```
{
   bit recvbit, sendbit;
   do
   :: in ? send_confirnation_user(recvbit) ->
   out ! send_confirmation_website(recvbit);
   od

}
init
{
   run User(toUser, toWebsite);
   run Website(toWebsite, toUser, toDatabase);
   run Database(toDatabase, toWebsite);
}
```


Select:
p_User
p_Website
p_Database
init
User
S6
out!req_to_update_community,sendbit
S2
in?display_update_form,recvbit
S3
in?send_error,recvbit
out!submit_information,sendbit
S4
out!validate_info,sendbit
S5


Automata View
Select:
User
Website
Database
Init
Website
S10
eq_to_update_community,recvbit
S2
!display_update_form,sendbit
S3
submit_information,recvbit
S4
?validate_info,recvbit
S5
out!send_error,sendbit
target_name_unique,recvbit
S6
target_name_unique,recvbit
S7
send_confirmation_user,sendbit
S8
end_confirmation_website,recvbit
S9

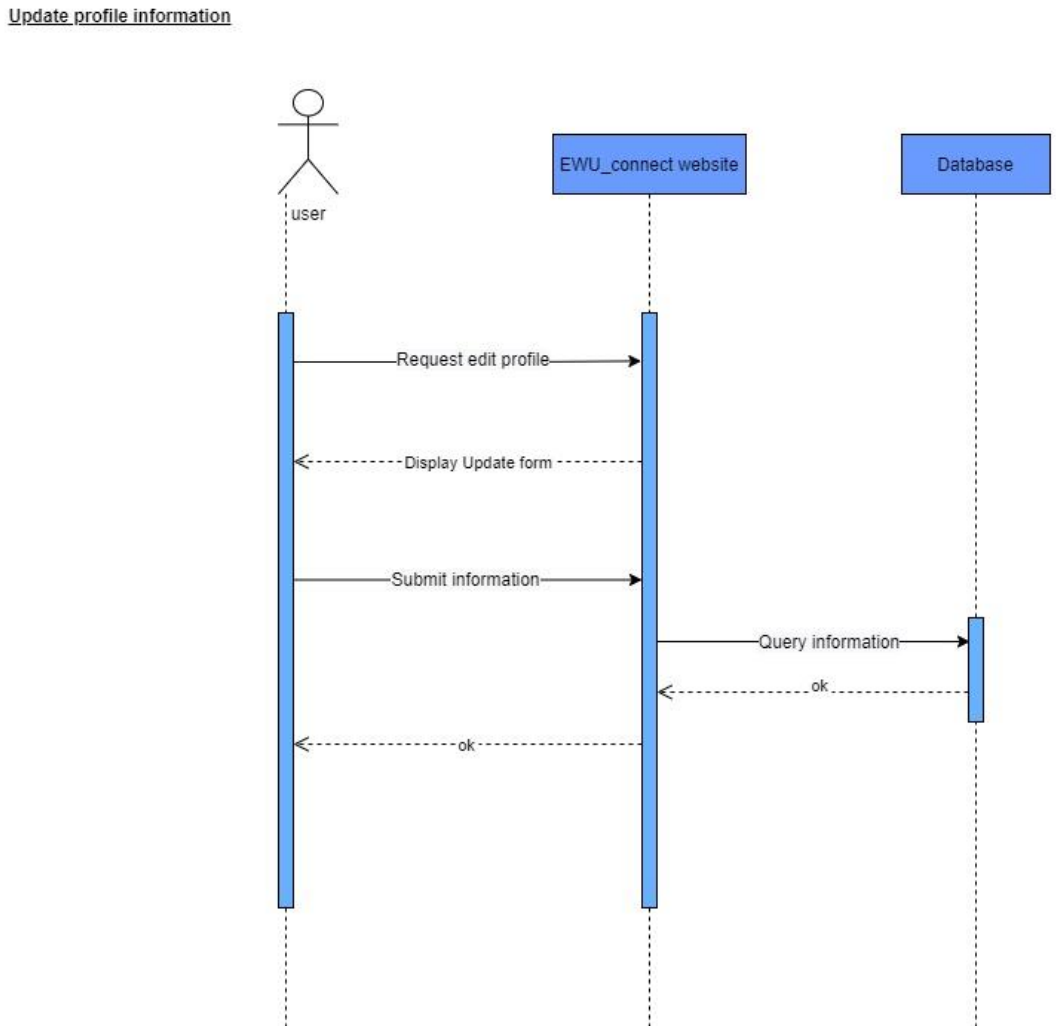

Automata View
Select:
p_User
p_Website
p_Database
init
Database
S3
send_confirmation_user,recvbit
nd_confirmation_website,recvbit
S2


Automata View
init
init
S1
(run User(toUser,toWebsite))
S2
(run Website(toWebsite,toUser,toDatabase))
S3
(run Database(toDatabase,toWebsite))
S4
-end-
S0



3.Sequence Diagram of Update Profile Information:

User will request for editing his/her profile on the website. It will return the profile edit form to the user. After filling up and submitting the form, the website will query to the database with new information. After successfully saving that information on the database, it will return a success response to the website, then the website will show a success message to the user.



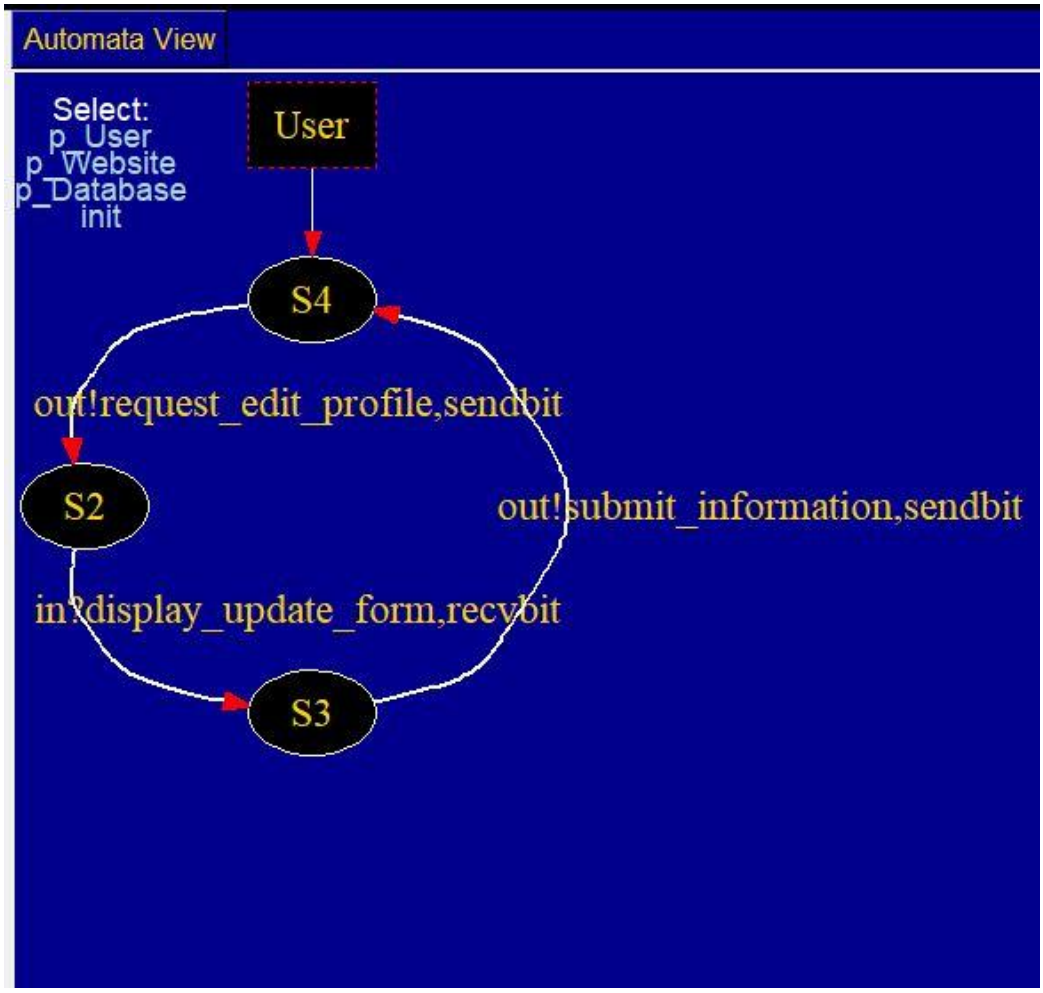
PROMELA Code:

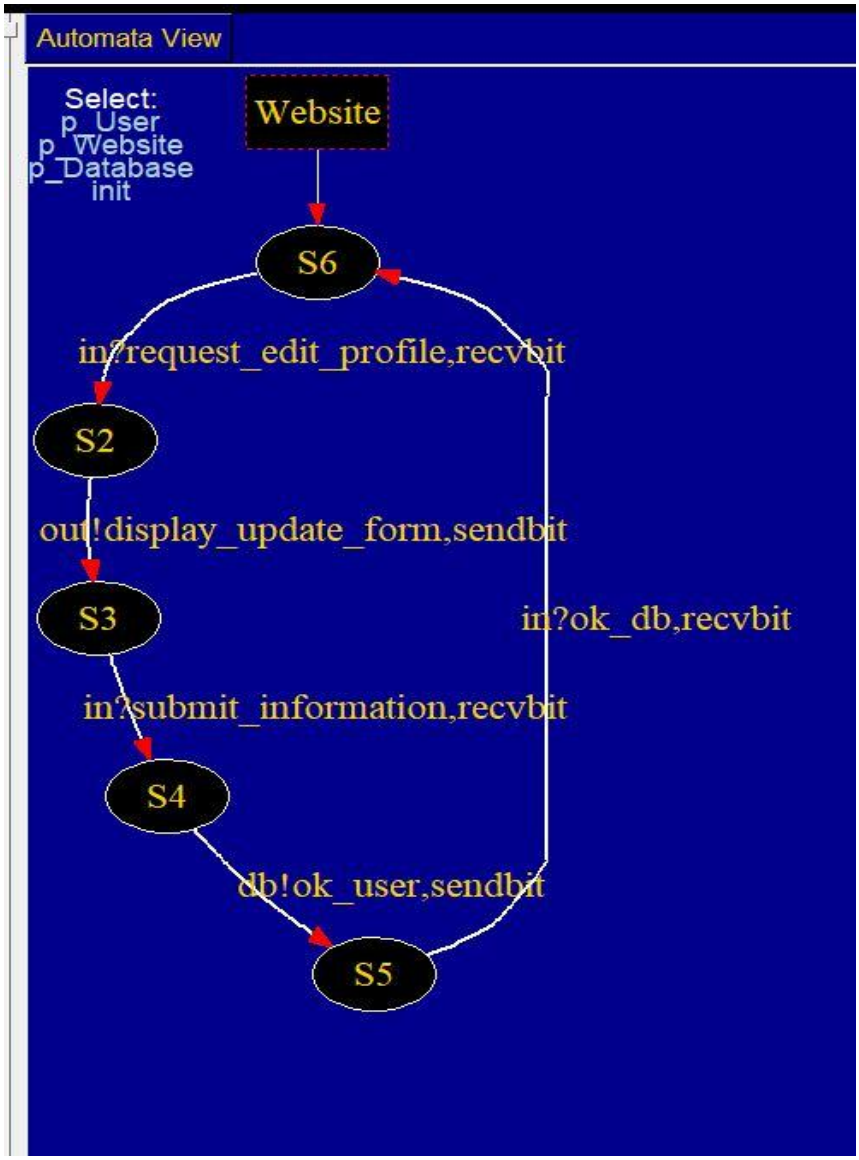
mtype = {request_edit_profile, display_update_form , submit_information, query_information,ok_user, ok_db}

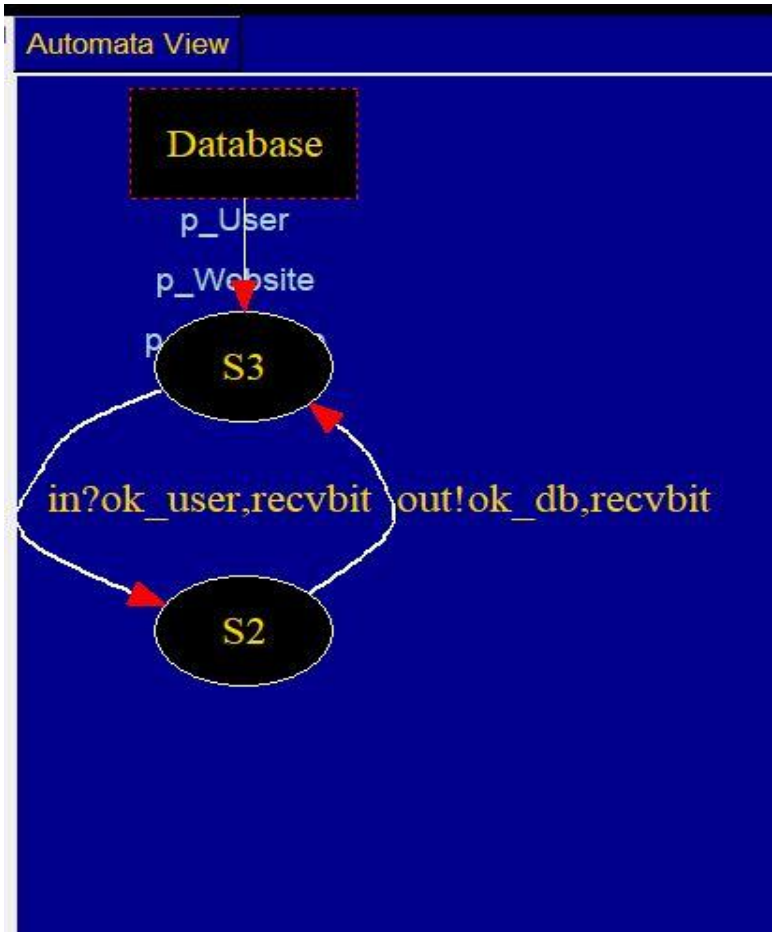
chan toUser = [1] of {mtype,bit}

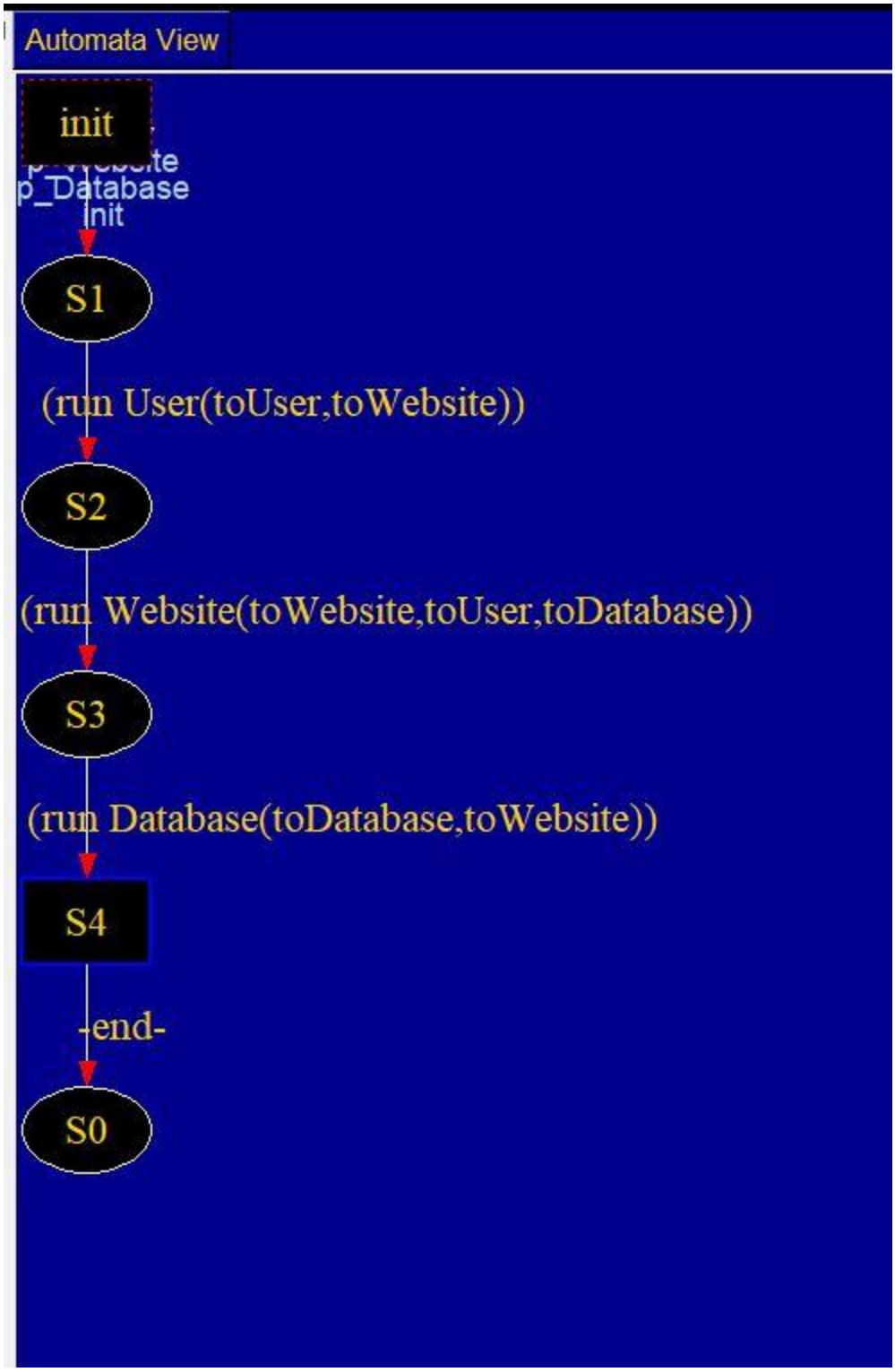
```
chan toWebsite = [1] of {mtype,bit};
chan toDatabase = [1] of {mtype,bit};
proctype User(chan in, out)
{
   bit sendbit, recvbit;

   do
   :: out ! request_edit_profile, sendbit;
   in ? display_update_form, recvbit;
   out ! submit_information, sendbit;


   od

}
proctype Website(chan in, out, db)
{
   bit sendbit, recvbit;

   do::
   in ? request_edit_profile(recvbit);
   out ! display_update_form(sendbit);
   in ? submit_information(recvbit);

   db ! ok_user, sendbit ->
   in ? ok_db, recvbit;


   od

}
```

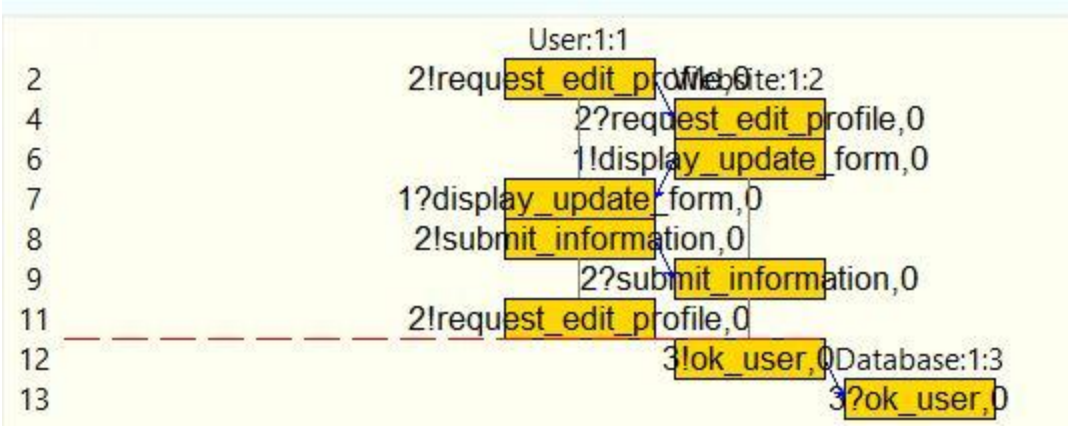
```
proctype Database(chan in, out)
{
   bit recvbit, sendbit;
   do
   :: in ? ok_user(recvbit) ->
   out ! ok_db(recvbit);
   od

}
init
{
   run User(toUser, toWebsite);
   run Website(toWebsite, toUser, toDatabase);
   run Database(toDatabase, toWebsite);
}
```


Automata View
Select:
p_User
p_Website
p_Database
init
User
S4
out!request_edit_profile,sendbit
S2
out!submit_information,sendbit
in?display_update_form,recvbit
S3


Automata View
Select:
p_User
p_Website
p_Database
init
Website
S6
in?request_edit_profile,recvbit
S2
out!display_update_form,sendbit
S3
in?ok_db,recvbit
in?submit_information,recvbit
S4
db!ok_user,sendbit
S5


Automata View
Database
p_User
p_Website
S3
in?ok_user,recvbit
out!ok_db,recvbit
S2


Automata View
init
p_Database
init
S1
(run User(toUser,toWebsite))
S2
(run Website(toWebsite,toUser,toDatabase))
S3
(run Database(toDatabase,toWebsite))
S4
-end-
S0


User:1:1
Website:1:2
Database:1:3
2
2!request_edit_profile,0
4
2?request_edit_profile,0
6
1!display_update_form,0
7
1?display_update_form,0
8
2!submit_information,0
9
2?submit_information,0
11
2!request_edit_profile,0
12
3!ok_user,0
13
3?ok_user,0