Introduction

Event Management System is a responsive web application that can make all the things easier related to organizing an event for a club and also for the participants who take part in these events. In this application, admins (Organizers) are facilitated with the functions of updating, editing and managing any kind of information related to events. Users of the web-application are able to sign in or sign up and participate in many events using the web-application. Users are facilitated with online payment option also. So, in few words, all the things about an event can be found in only one place, scattered information can be arranged properly, event management can be much easier.

Chosen Sequence Diagrams

1. Sequence Diagram for Sign in Page

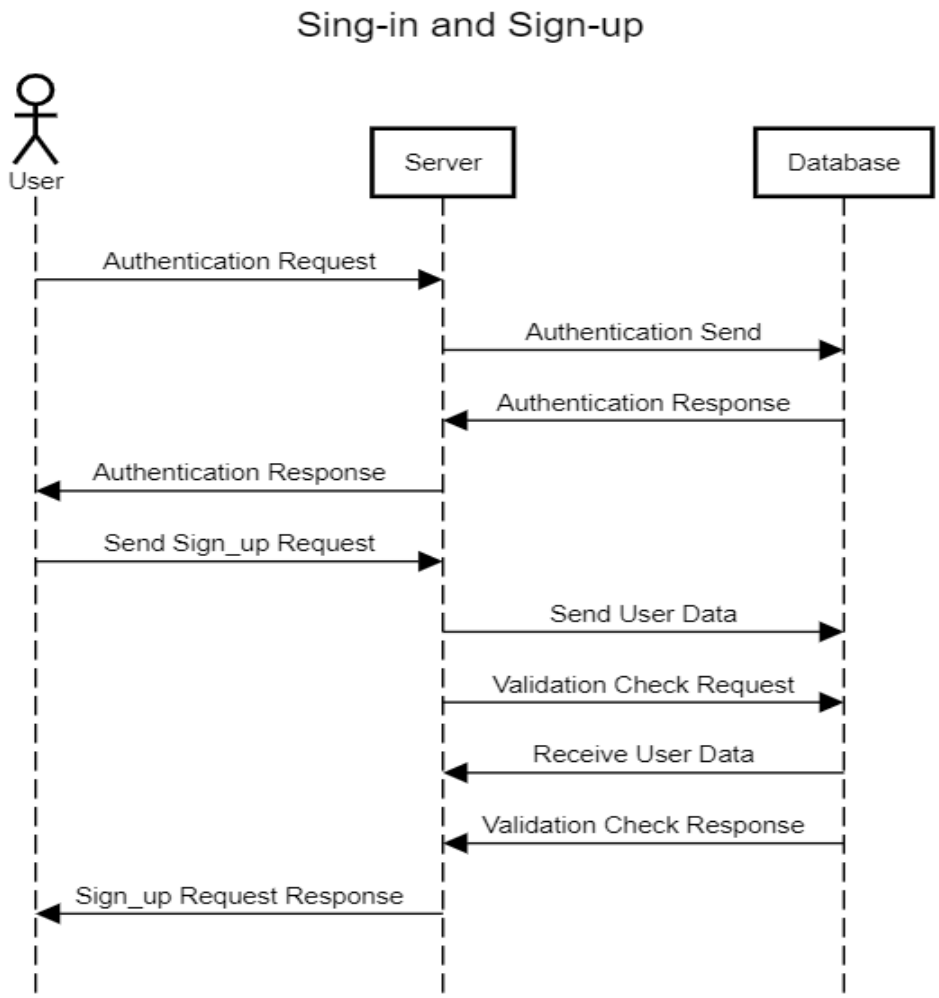

Figure 1: Sequence Diagram for User Sign in

Promela Code for 1st Sequence Diagram:

```
mtype={auth_req, auth_send, send_signup_req, send_usr_dt,
val_chk, ack};
chan user= [2] of {mtype, bit};
chan server= [2] of {mtype, bit};
chan database= [2] of {mtype, bit};

proctype User(chan userCh, serverCh, databaseCh)
{
      bit snd, rcv;
      do
      :: serverCh! auth_req(snd) -> userCh?ack(rcv);
      :: serverCh! send_signup_req(snd)-> userCh?ack(rcv);
      od
}

proctype Server(chan userCh, serverCh, databaseCh)
{
      bit snd, rcv;
      do
      :: serverCh? auth_req(rcv) -> databaseCh! auth_send(snd);
      :: serverCh? ack(rcv) -> userCh! ack(rcv);
      :: serverCh? send_signup_req(rcv) ->
databaseCh!send_usr_dt(snd);
      :: serverCh? send_signup_req(rcv)-> databaseCh!
val_chk(snd);
      :: serverCh? ack(rcv) -> userCh! ack(rcv);
      od
}

proctype Database(chan userCh, serverCh, databaseCh)
{
      bit rcv;
      do
      :: databaseCh? auth_send(rcv) -> serverCh! ack(rcv);
      :: databaseCh? send_usr_dt(rcv) -> serverCh! ack(rcv);
      :: databaseCh? val_chk(rcv) -> serverCh! ack(rcv);
      od
}
```

```
init
{
run User(user, server, database);
run Server(user, server, database);
run Database(user, server, database);
}
```

Automata for user:

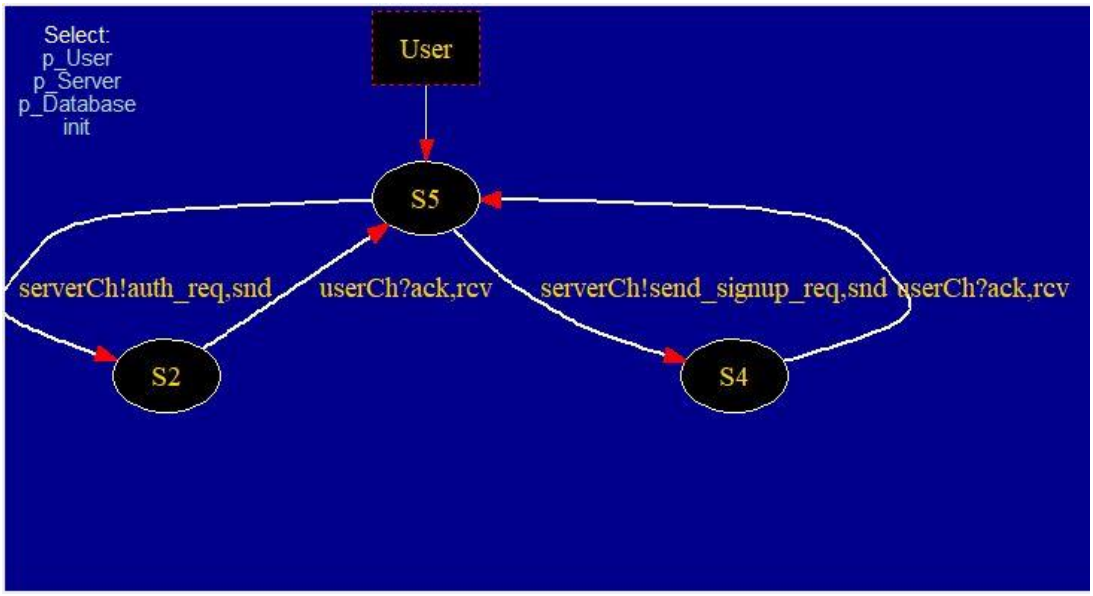

Figure 2: Automata for user (1st Sequence Diagram)

Automata for server:

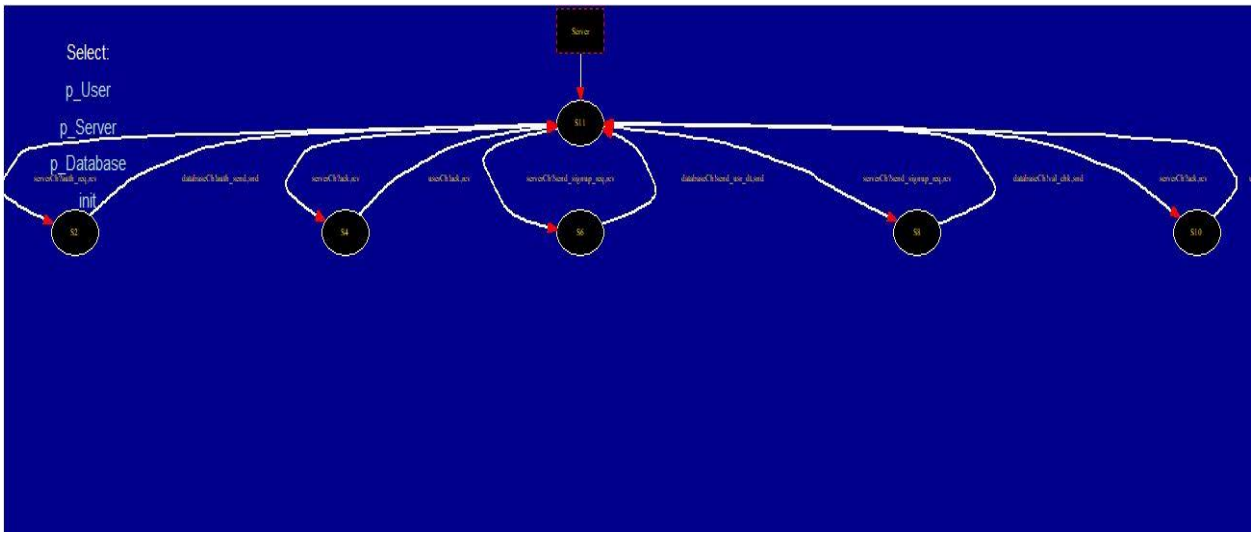

Figure 3: Automata for server (1st sequence diagram)

Automata for database:

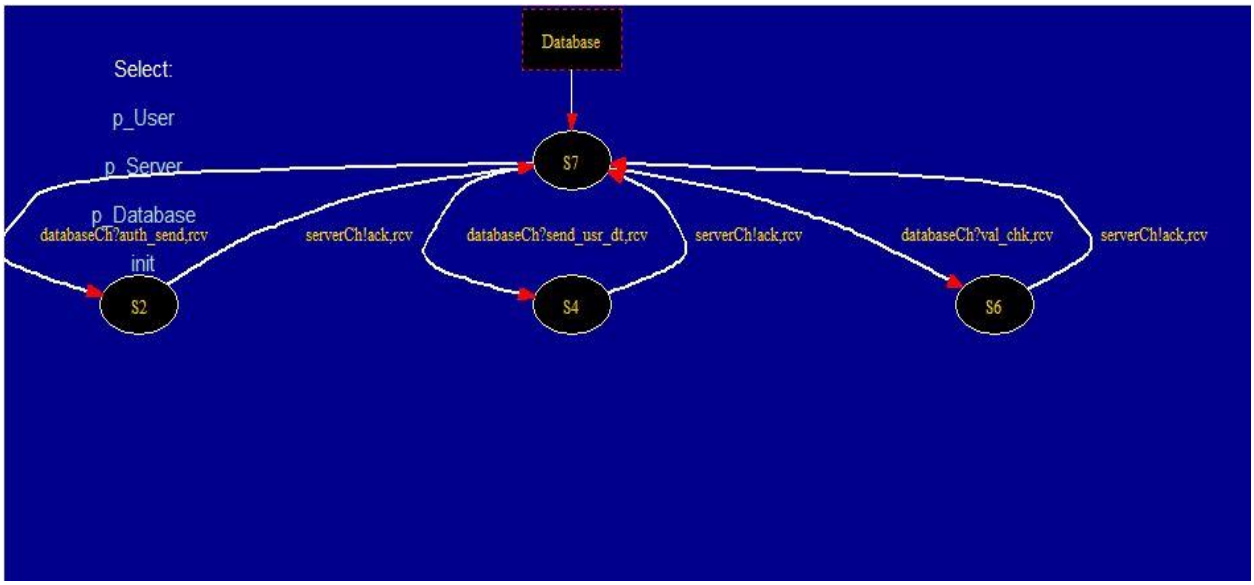

Figure 4: Automata for database (1st Sequence Diagram)

Process Simulation:

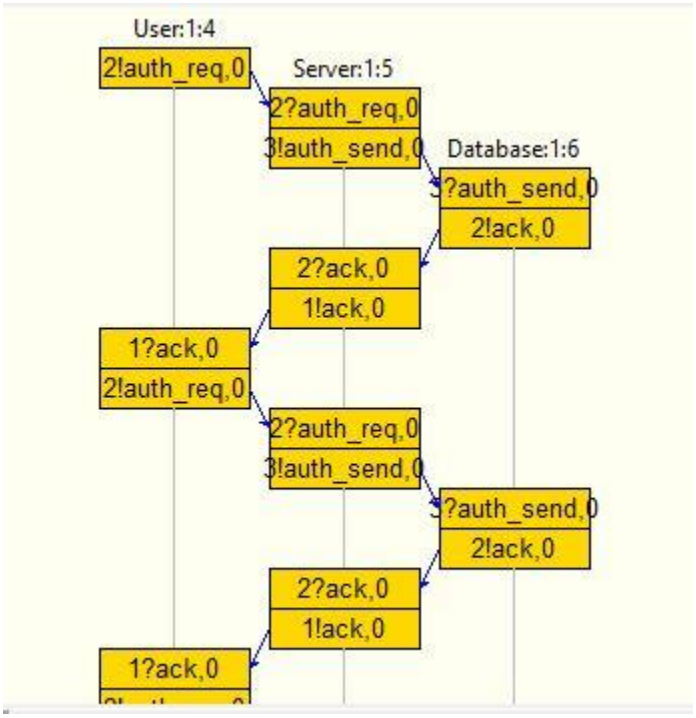

Figure 5: Process Simulation (1st Sequence Diagram)

Process console:

```
using statement merging
Starting User with pid 4
  1:        proc  3 (:init::1) user_auth_signup.pml:39 (state 1) [(run User(user,server,database))]
Starting Server with pid 5
  2:        proc  3 (:init::1) user_auth_signup.pml:40 (state 2) [(run Server(user,server,database))]
Starting Database with pid 6
  3:        proc  3 (:init::1) user_auth_signup.pml:41 (state 3) [(run Database(user,server,database))]
  4:        proc  4 (User:1) user_auth_signup.pml:10 (state 1) [serverCh!auth_req,snd]
  5:        proc  5 (Server:1) user_auth_signup.pml:19 (state 1) [serverCh?auth_req,rcv]
  6:        proc  5 (Server:1) user_auth_signup.pml:19 (state 2) [databaseCh!auth_send,snd]
  7:        proc  6 (Database:1) user_auth_signup.pml:31 (state 1) [databaseCh?auth_send,rcv]
  8:        proc  6 (Database:1) user_auth_signup.pml:31 (state 2) [serverCh!ack,rcv]
  9:        proc  5 (Server:1) user_auth_signup.pml:20 (state 3) [serverCh?ack,rcv]
 10:        proc  5 (Server:1) user_auth_signup.pml:20 (state 4) [userCh!ack,rcv]
Error: receiving from an uninitialized chan databaseCh
 11:        proc  2 (Database:1) user_auth_signup.pml:31 (state 1) [databaseCh?auth_send,rcv]
            transition failed
spin: trail ends after 11 steps
#processes: 7
 11:        proc  6 (Database:1) user_auth_signup.pml:30 (state 7)
 11:        proc  5 (Server:1) user_auth_signup.pml:18 (state 11)
 11:        proc  4 (User:1) user_auth_signup.pml:10 (state 2)
 11:        proc  3 (:init::1) user_auth_signup.pml:42 (state 4)
 11:        proc  2 (Database:1) user_auth_signup.pml:31 (state 1)
 11:        proc  1 (Server:1) user_auth_signup.pml:18 (state 11)
 11:        proc  0 (User:1) user_auth_signup.pml:9 (state 5)
7 processes created
Exit-Status 0
```

Figure 6: Process Simulation Console (1st Sequence Diagram)

Verification:

```
            + Partial Order Reduction

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

State-vector 124 byte, depth reached 10, errors: 1
       11 states, stored
        1 states, matched
       12 transitions (= stored+matched)
        0 atomic steps
hash conflicts:         0 (resolved)

Stats on memory usage (in Megabytes):
    0.001   equivalent memory usage for states (stored*(State-vector + overhead))
    0.289   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
```

Figure 7: Verification (1st Sequence Diagram)

2. Sequence diagram for Participation of Users in Events

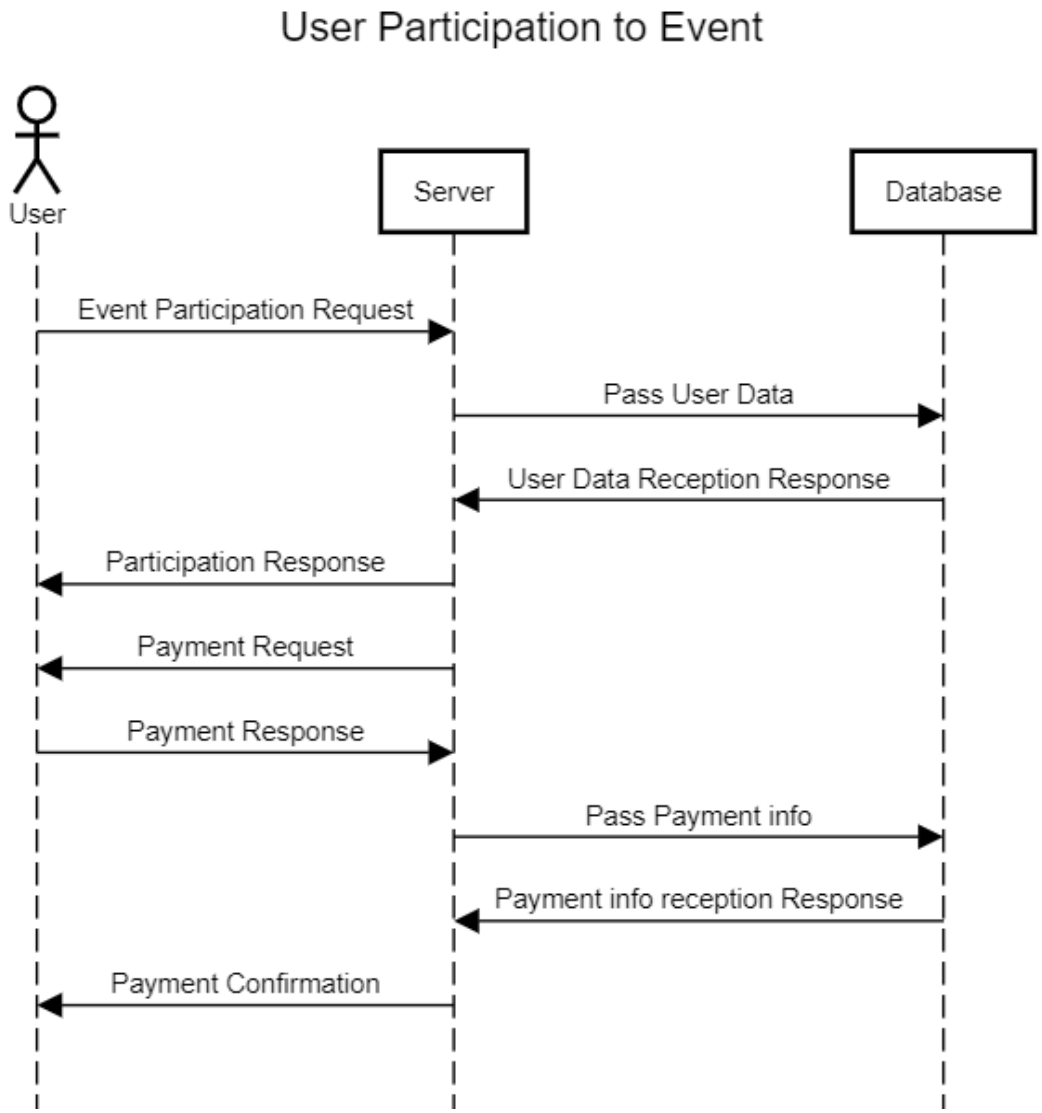

Figure 8: Sequence Diagram for User Event Participation

Next_Page();

Promela Code for 2nd Sequence Diagram:

```
mtype= {event_part_req, pass_usr_dt, payment_req, payment,
pass_payment_info, payment_confirm, ack};
chan user= [2] of {mtype, bit};
chan server= [2]of {mtype, bit};
chan database= [2] of {mtype, bit};

proctype User(chan userCh, serverCh, databaseCh)
     {
          bit snd, rcv;
          do
          :: serverCh! event_part_req(snd) -> userCh?ack(rcv);
          :: userCh? payment_req(rcv) -> server! payment(snd);
          od
     }

proctype Server(chan userCh, serverCh, databaseCh)
     {
          bit snd, rcv;
          do
          :: serverCh? event_part_req(rcv) -> databaseCh!
pass_usr_dt(snd);
          :: serverCh? ack(rcv) -> userCh! ack(rcv);
          :: serverCh? ack(rcv) -> userCh! payment_req(snd);
          :: serverCh? payment(rcv)-> databaseCh!
pass_payment_info(snd);
          :: serverCh? ack(rcv) -> userCh! ack(rcv);
          od
     }

proctype Database(chan userCh, serverCh, databaseCh)
     {
          bit rcv;
          do
          :: databaseCh? pass_usr_dt(rcv) -> serverCh! ack(rcv);
          :: databaseCh? pass_payment_info(rcv) -> serverCh!
ack(rcv);
          od
     }
```

```
init
{
run User(user, server, database);
run Server(user, server, database);
run Database(user, server, database);
}
```

Automata for user:

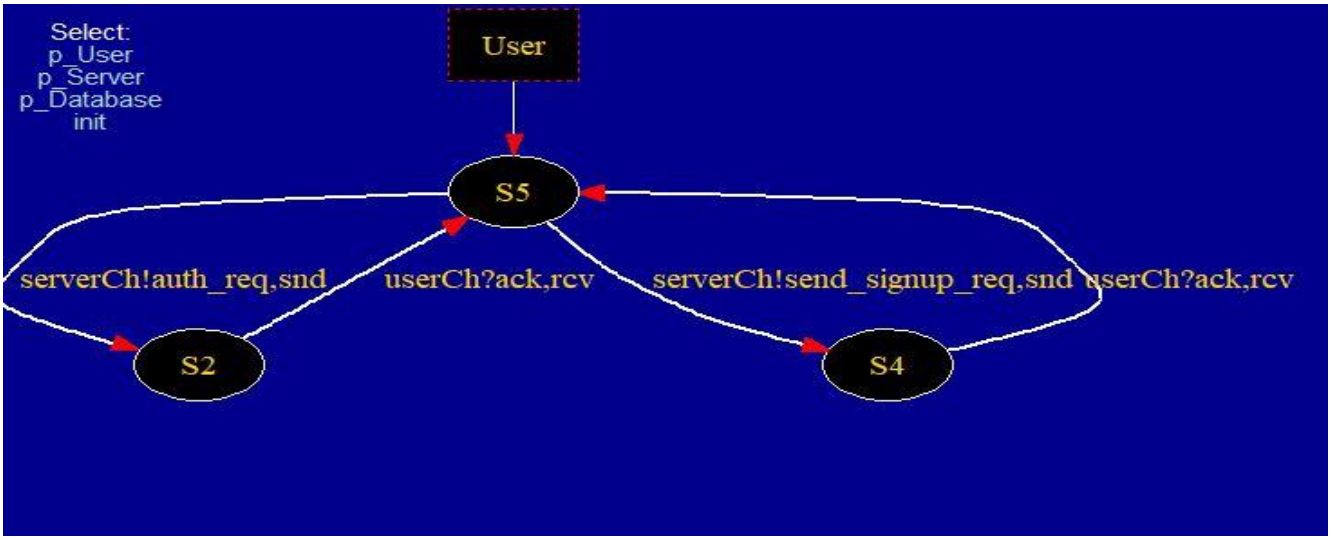

Figure 9: Automata for User (2nd Sequence Diagram)

Automata for Server:

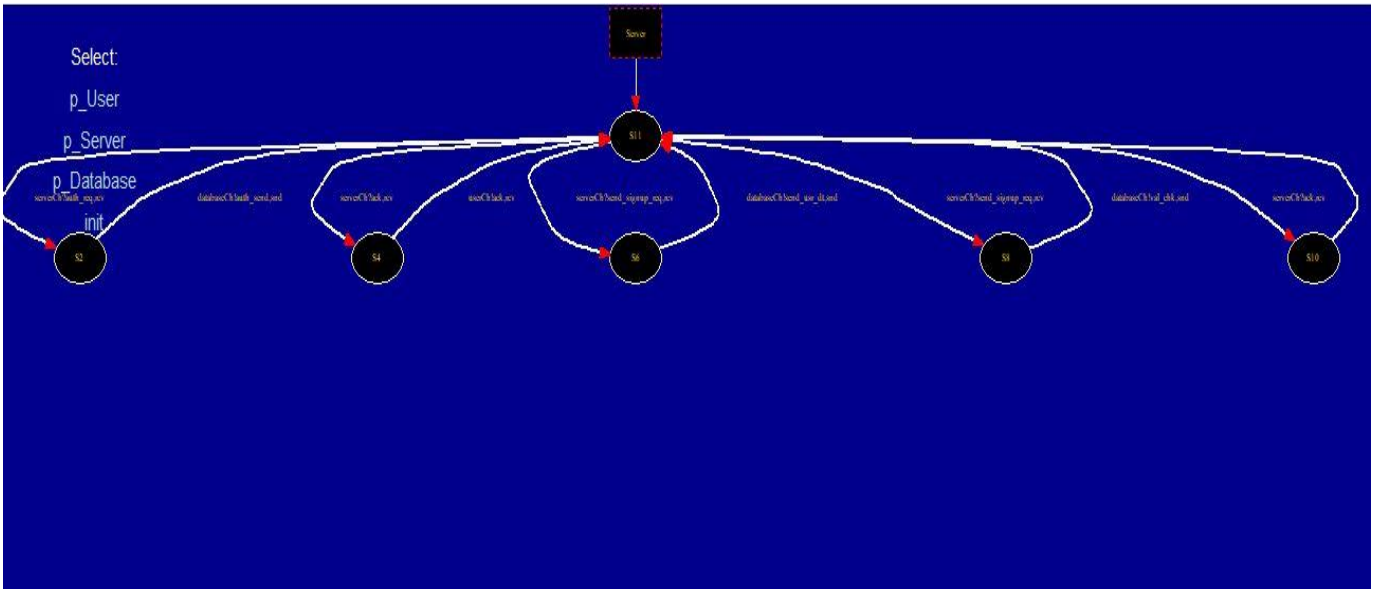

Figure 10: Automata for server (2nd Sequence Diagram)

Automata for Database:

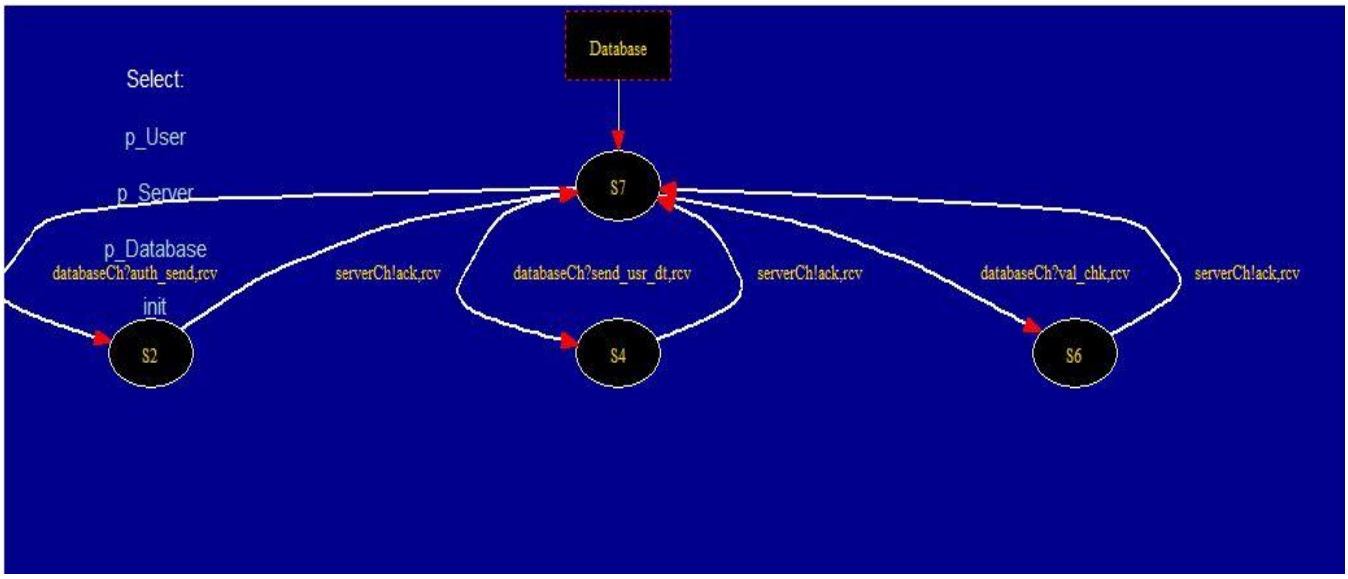

Figure 11: Automata for database (2nd Sequence Diagram)

Process Simulation:

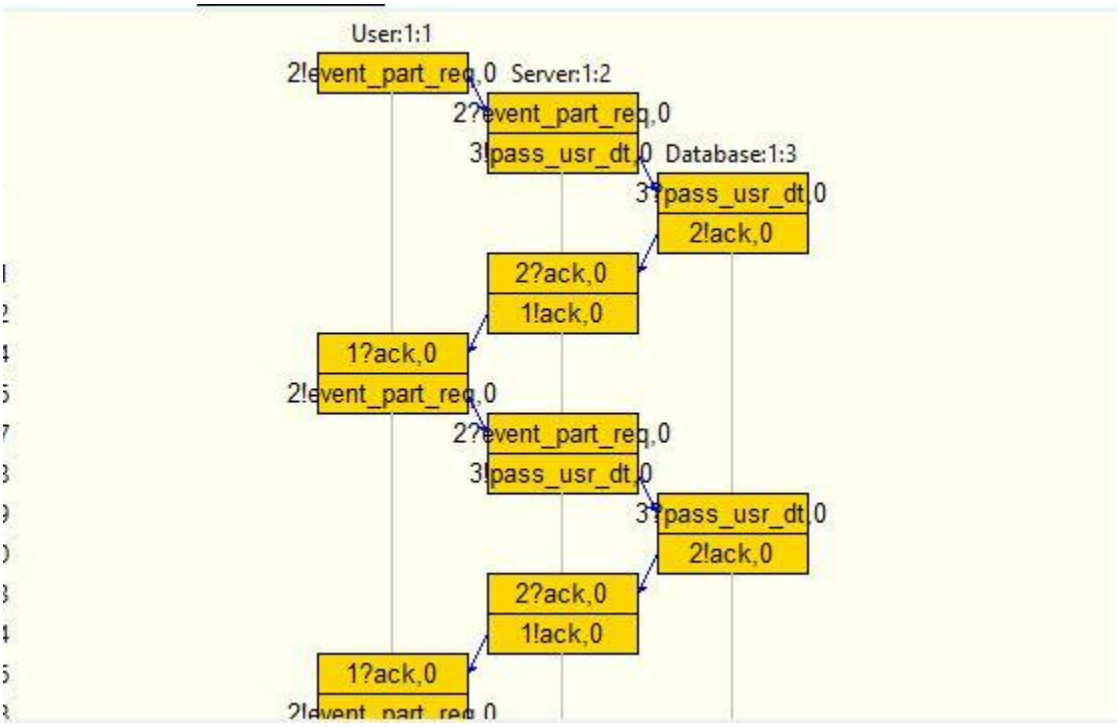

Figure 12: Process Simulation (2nd Sequence Diagram)

Process Console:

```
  0:    proc  - (:root:) creates proc  0 (:init:)
Starting User with pid 1
  1:    proc  0 (:init::1) creates proc  1 (User)
  1:    proc  0 (:init::1) event_participation.pml:35 (state 1) [(run User(user,server,database))]
  2:    proc  1 (User:1) event_participation.pml:10 (state 1) [serverCh!event_part_req,snd]
Starting Server with pid 2
  3:    proc  0 (:init::1) creates proc  2 (Server)
  3:    proc  0 (:init::1) event_participation.pml:36 (state 2) [(run Server(user,server,database))]
  4:    proc  2 (Server:1) event_participation.pml:18 (state 1) [serverCh?event_part_req,rcv]
Starting Database with pid 3
  5:    proc  0 (:init::1) creates proc  3 (Database)
  5:    proc  0 (:init::1) event_participation.pml:37 (state 3) [(run Database(user,server,database))]
  6:    proc  2 (Server:1) event_participation.pml:18 (state 2) [databaseCh!pass_usr_dt,snd]
  7:    proc  3 (Database:1) event_participation.pml:29 (state 1) [databaseCh?pass_usr_dt,rcv]
  8:    proc  3 (Database:1) event_participation.pml:29 (state 2) [serverCh!ack,rcv]
 11:    proc  2 (Server:1) event_participation.pml:22 (state 9) [serverCh?ack,rcv]
 12:    proc  2 (Server:1) event_participation.pml:22 (state 10) [userCh!ack,rcv]
 14:    proc  1 (User:1) event_participation.pml:10 (state 2) [userCh?ack,rcv]
 16:    proc  1 (User:1) event_participation.pml:10 (state 1) [serverCh!event_part_req,snd]
 17:    proc  2 (Server:1) event_participation.pml:18 (state 1) [serverCh?event_part_req,rcv]
 18:    proc  2 (Server:1) event_participation.pml:18 (state 2) [databaseCh!pass_usr_dt,snd]
 19:    proc  3 (Database:1) event_participation.pml:29 (state 1) [databaseCh?pass_usr_dt,rcv]
 20:    proc  3 (Database:1) event_participation.pml:29 (state 2) [serverCh!ack,rcv]
 23:    proc  2 (Server:1) event_participation.pml:22 (state 9) [serverCh?ack,rcv]
 24:    proc  2 (Server:1) event_participation.pml:22 (state 10) [userCh!ack,rcv]
 26:    proc  1 (User:1) event_participation.pml:10 (state 2) [userCh?ack,rcv]
 28:    proc  1 (User:1) event_participation.pml:10 (state 1) [serverCh!event_part_req,snd]
 29:    proc  2 (Server:1) event_participation.pml:18 (state 1) [serverCh?event_part_req,rcv]
 30:    proc  2 (Server:1) event_participation.pml:18 (state 2) [databaseCh!pass_usr_dt,snd]
 32:    proc  3 (Database:1) event_participation.pml:29 (state 1) [databaseCh?pass_usr_dt,rcv]
 33:    proc  3 (Database:1) event_participation.pml:29 (state 2) [serverCh!ack,rcv]
 35:    proc  2 (Server:1) event_participation.pml:22 (state 9) [serverCh?ack,rcv]
 36:    proc  2 (Server:1) event_participation.pml:22 (state 10) [userCh!ack,rcv]
 38:    proc  1 (User:1) event_participation.pml:10 (state 2) [userCh?ack,rcv]
 40:    proc  1 (User:1) event_participation.pml:10 (state 1) [serverCh!event_part_req,snd]
```

Figure 13: Process Simulation Console (2nd Sequence Diagram)

Verification:

```
            assertion violations +
            cycle checks       - (disabled by -DSAFETY)
            invalid end states +

State-vector 100 byte, depth reached 10, errors: 1
       13 states, stored
        1 states, matched
       14 transitions (= stored+matched)
        0 atomic steps
hash conflicts:         0 (resolved)

Stats on memory usage (in Megabytes):
    0.001   equivalent memory usage for states (stored*(State-vector + overhead))
    0.290   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



pan: elapsed time 0.005 seconds
To replay the error-trail, goto Simulate/Replay and select "Run"
```

Figure 14: Verification (2nd Sequence Diagram)

Conclusion

Developing this web-application and verifying the sequence diagrams with "Spin Model Checker" has paved the way to think more and more about endless problems that can arise. Mandatory things have been introduced in the web-application. However, in future, with user's satisfaction and request we are always keen to bring new functions and features in the application to ease many other sides of an event. This project has taught us so many important things about developing a web application. It was just a little step forward to take part in more complex, bigger and more challenging works that lies ahead in our future.