EAST WEST UNIVERSITY

PROJECT REPORT

Course Code : CSE 435

Course Title : Software Quality Assurance

Submitted to

Dr. Shamim H Ripon

Professor, Dept. Of CSE

Submitted by

Shishir Zaman

ID : 2017-2-60-141

Department of Computer Science and Engineering

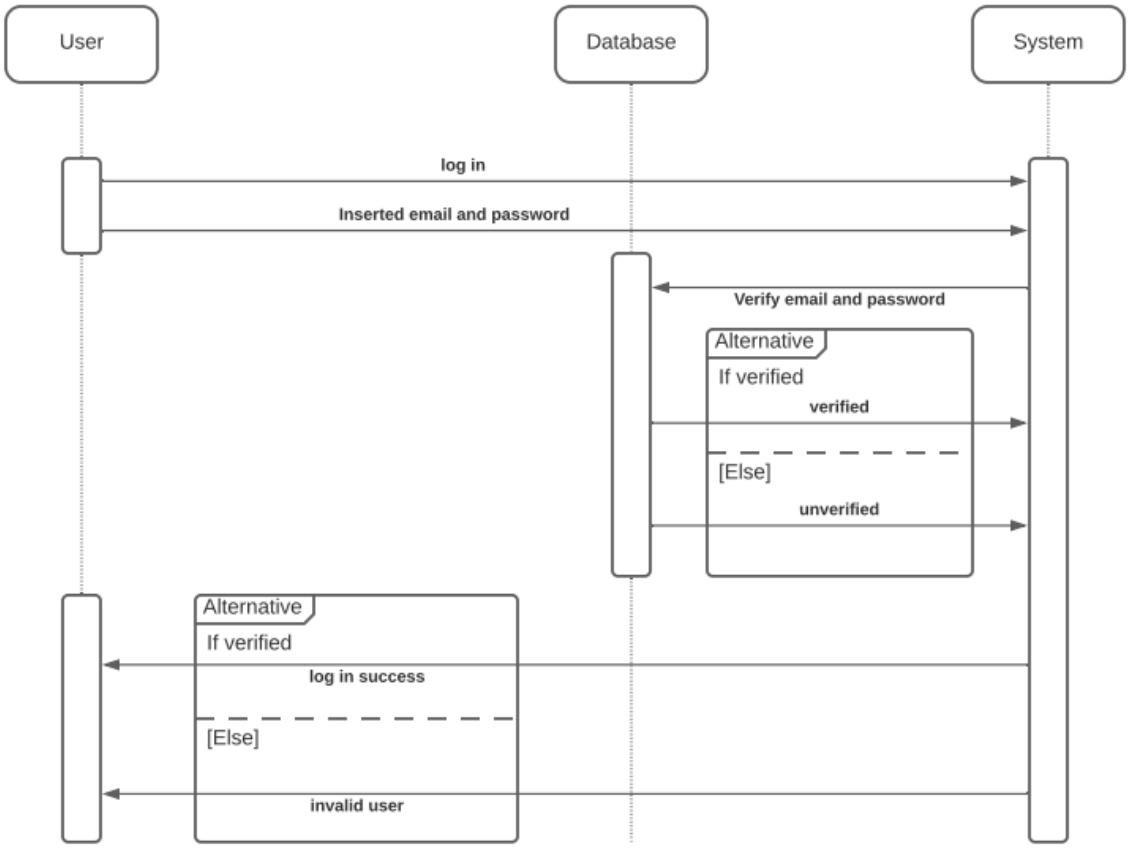
Date of Submission : 02-06-2021

Introduction:

"Corona Updates and lab test management system" is an online computerize system where a user can visualize (different charts are used) worldwide and Bangladesh corona updates news and can reserve a date a for corona lab test whenever he/she want to test himself/ herself. Through this online system anyone can visualize corona updates news but for booking a date for lab test at a test center user need to register himself/ herself to the system. And after testing, his/ her test result will be informed through this system.

Chosen Sequence Diagram:

Sequence diagram of Login:



Promela code for Login Sequence diagram:

```
      mtype = {login, insert_email_pass, inserted_email_pass, verify_email_pass,
 verified_email_pass, unverified_email_pass,

        login_success, invalid_user};

chan user = [2] of {mtype, bit};
chan system = [2] of {mtype, bit};
```

```
chan database = [2] of {mtype, bit};
bool verify = 1;
bool valid = 1;
bool verified = 1;


proctype User(chan userCh, systemCh, databaseCh)
{
    bit snd,rcv;
    do
    :: userCh!login(snd)->systemCh?insert_email_pass(rcv);
    userCh!inserted_email_pass(snd);
    if
    ::valid==1->
        userCh?login_success(rcv);valid=0;
    ::valid==0->
        userCh?invalid_user(rcv);valid=1;
    fi
    od
}

proctype System(chan userCh, systemCh, databaseCh)
{
    bit snd,rcv;
    do
    ::userCh?login(rcv)->systemCh!insert_email_pass(rcv);
    userCh?inserted_email_pass(rcv)->systemCh!verify_email_pass(rcv);
    if
    ::verified==1->
        databaseCh?verified_email_pass(rcv)-
>userCh!login_success(rcv);verified=0;
    ::verified==0->
        databaseCh?unverified_email_pass(rcv)-
>userCh!invalid_user(rcv);verified=1;
    fi
    od
}

proctype Database(chan userCh, systemCh, databaseCh)
{
    bit rcv;
    do
    ::if
    ::verify==1->
```

```
        system?verify_email_pass(rcv)-
>databaseCh!verified_email_pass(rcv);verify=0;
    ::verify==0->
        system?verify_email_pass(rcv)-
>databaseCh!unverified_email_pass(rcv);verify=1;
    fi
    od
}

init
{
run User(user, system, database);
run System(user, system, database);
run Database(user, system, database);
}
```

Automata for User:

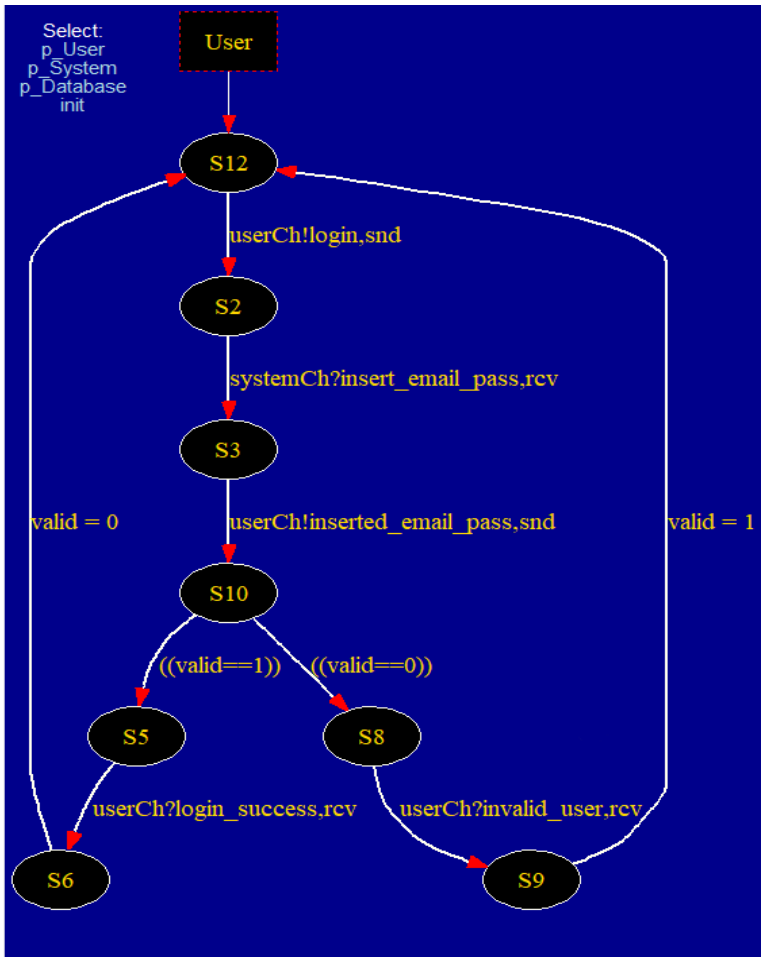


Fig: automata for user.

Automata for System:

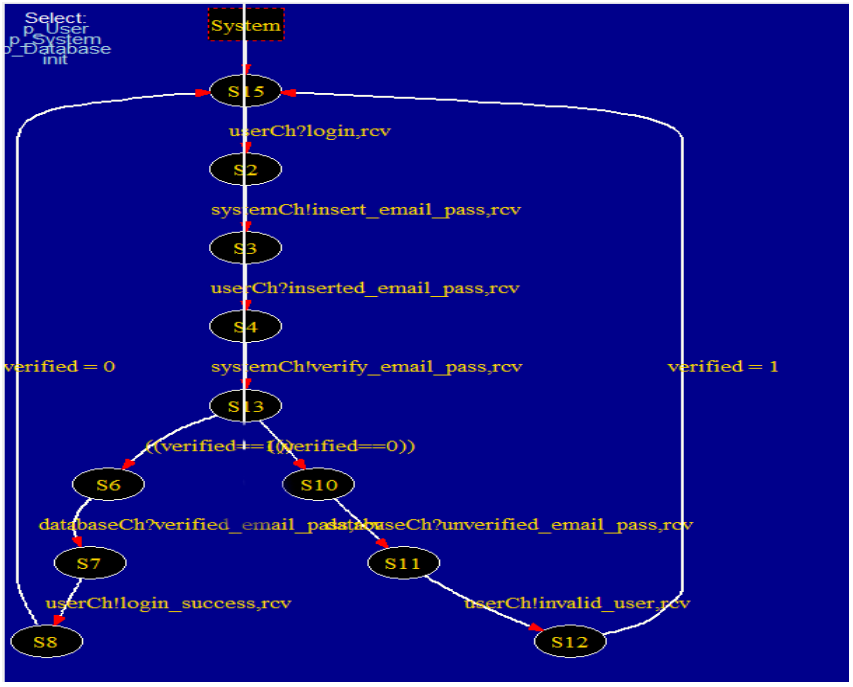


Fig: Automata for system.

Automata for Database:

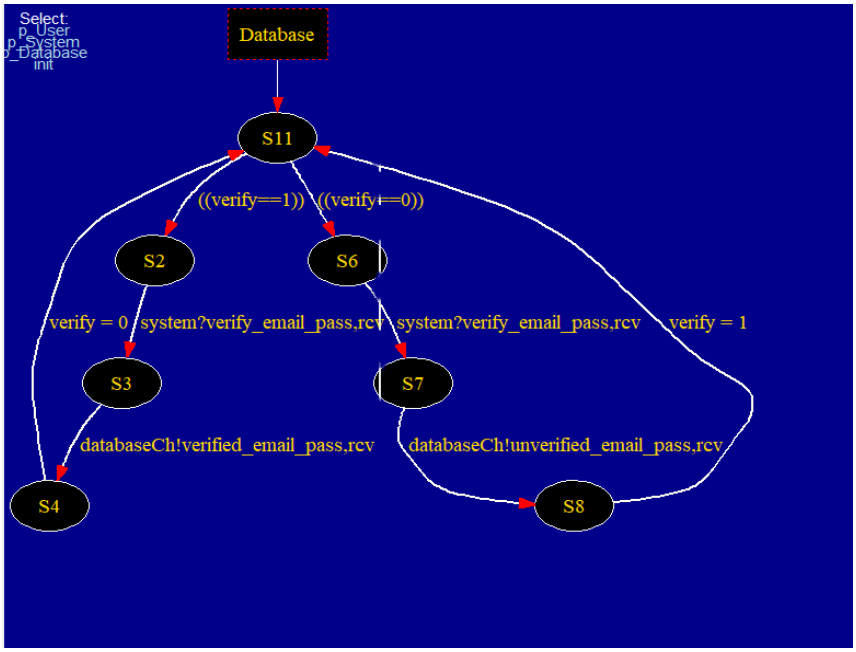


Fig: Automata for Database.

Process Simulation:

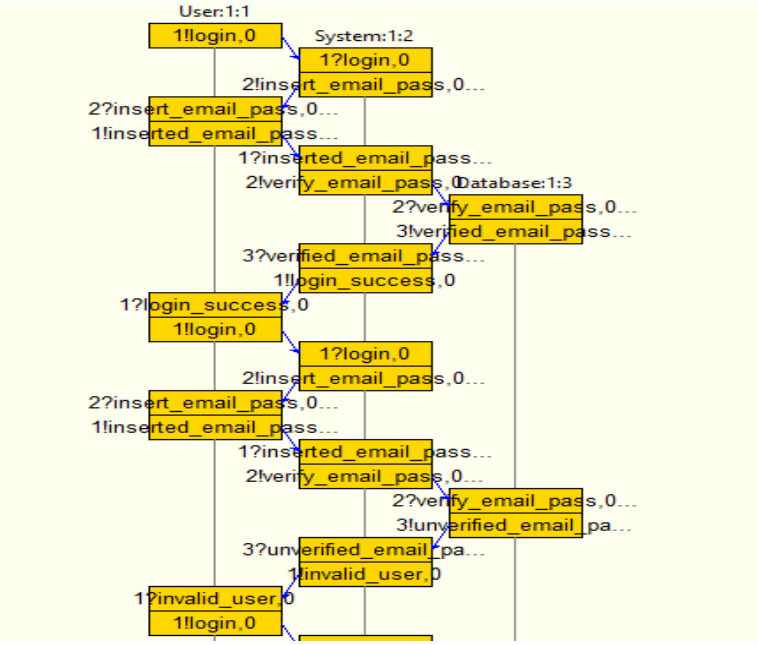


Fig: Process simulation.

Process console:

```
 0:      proc  - (:root:) creates proc  0 (:init:)
spin: login.pml:0, warning, proctype System, 'bit   snd' variable is never used (other than in print stmnts)
Starting User with pid 1
 1:      proc  0 (:init::1) creates proc  1 (User)
 1:      proc  0 (:init::1) login.pml:57 (state 1) [(run User(user,system,database))]
 2:      proc  1 (User:1) login.pml:16 (state 1) [userCh!login,snd]
Starting System with pid 2
 3:      proc  0 (:init::1) creates proc  2 (System)
 3:      proc  0 (:init::1) login.pml:58 (state 2) [(run System(user,system,database))]
 4:      proc  2 (System:1) login.pml:31 (state 1) [userCh?login,rcv]
Starting Database with pid 3
 5:      proc  0 (:init::1) creates proc  3 (Database)
 5:      proc  0 (:init::1) login.pml:59 (state 3) [(run Database(user,system,database))]
 6:      proc  3 (Database:1) login.pml:47 (state 1) [((verify==1))]
 7:      proc  2 (System:1) login.pml:31 (state 2) [systemCh!insert_email_pass,rcv]
 8:      proc  1 (User:1) login.pml:16 (state 2) [systemCh?insert_email_pass,rcv]
 9:      proc  1 (User:1) login.pml:17 (state 3) [userCh!inserted_email_pass,snd]
10:      proc  2 (System:1) login.pml:32 (state 3) [userCh?inserted_email_pass,rcv]
11:      proc  1 (User:1) login.pml:19 (state 4) [((valid==1))]
12:      proc  2 (System:1) login.pml:32 (state 4) [systemCh!verify_email_pass,rcv]
13:      proc  2 (System:1) login.pml:34 (state 5) [((verified==1))]
14:      proc  3 (Database:1) login.pml:48 (state 2) [system?verify_email_pass,rcv]
15:      proc  3 (Database:1) login.pml:48 (state 3) [databaseCh!verified_email_pass,rcv]
16:      proc  3 (Database:1) login.pml:48 (state 4) [verify = 0]
```

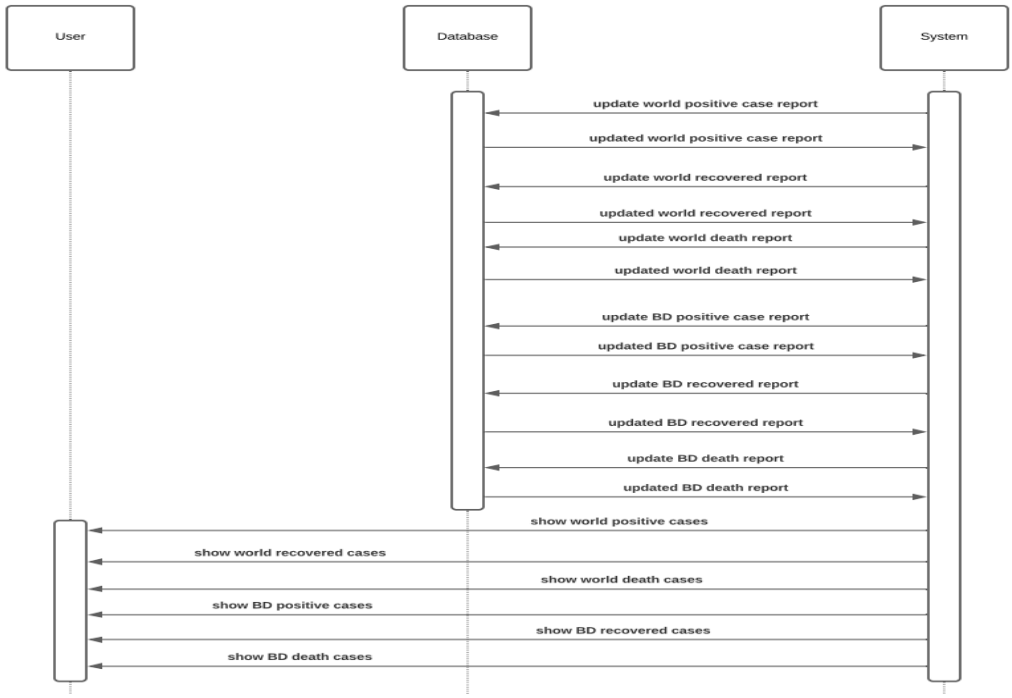
Fig: Process console.

Verification:

```
verification result:
spin -a  login.pml
gcc -DMEMLIM=1024 -O2 -DXUSAFE -DSAFETY -DNOCLAIM -w -o pan pan.c
./pan -m10000
Pid: 2308

(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 100 byte, depth reached 110, errors: 0
     173 states, stored
     179 states, matched
     352 transitions (= stored+matched)
       0 atomic steps
hash conflicts:         0 (resolved)

Stats on memory usage (in Megabytes):
   0.018  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


unreached in proctype User
        login.pml:25, state 15, "-end-"
        (1 of 15 states)
unreached in proctype System
        login.pml:40, state 18, "-end-"
        (1 of 18 states)
unreached in proctype Database
        login.pml:53, state 14, "-end-"
        (1 of 14 states)
unreached in init
        (0 of 4 states)

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

Fig: Verification for login sequence diagram.

Sequence diagram of Covid News update:



Promela code for Covid News Update Sequence diagram:

```
mtype = {upWoP_req, upWoP_ack, upWoRe_req, upWoRe_ack, upWoDe_req, upWoDe_ack,
     upBdP_req, upBdP_ack, upBdRe_req, upBdRe_ack, upBdDe_req, upBdDe_ack,
     shoWoP_ack, shoWoRe_ack, shoWoDe_ack, shoBdP_ack, shoBdRe_ack, shoBdDe_ack,
     shoWoP, shoWoRe, shoWoDe, shoBdP, shoBdRe, shoBdDe};

chan user = [2] of {mtype, bit};
chan system = [2] of {mtype, bit};
chan database = [2] of {mtype, bit};

proctype System(chan userCh, systemCh, databaseCh)
{
    bit snd,rcv;
    do
    ::systemCh!upWoP_req(snd)->databaseCh?upWoP_ack(rcv);
    systemCh!upWoRe_req(snd)->databaseCh?upWoRe_ack(rcv);
    systemCh!upWoDe_req(snd)->databaseCh?upWoDe_ack(rcv);
    systemCh!upBdP_req(snd)->databaseCh?upBdP_ack(rcv);
    systemCh!upBdRe_req(snd)->databaseCh?upBdRe_ack(rcv);
    systemCh!upBdDe_req(snd)->databaseCh?upBdDe_ack(rcv);
    systemCh!shoWoP(snd)->userCh?shoWoP_ack(rcv);
    systemCh!shoWoRe(snd)->userCh?shoWoRe_ack(rcv);
    systemCh!shoWoDe(snd)->userCh?shoWoDe_ack(rcv);
    systemCh!shoBdP(snd)->userCh?shoBdP_ack(rcv);
    systemCh!shoBdRe(snd)->userCh?shoBdRe_ack(rcv);
    systemCh!shoBdDe(snd)->userCh?shoBdDe_ack(rcv);
    od
}
proctype Database(chan userCh, systemCh, databaseCh)
{
    bit rcv;
    do
    ::systemCh?upWoP_req(rcv)->databaseCh!upWoP_ack(rcv);
    systemCh?upWoRe_req(rcv)->databaseCh!upWoRe_ack(rcv);
    systemCh?upWoDe_req(rcv)->databaseCh!upWoDe_ack(rcv);
    systemCh?upBdP_req(rcv)->databaseCh!upBdP_ack(rcv);
    systemCh?upBdRe_req(rcv)->databaseCh!upBdRe_ack(rcv);
    systemCh?upBdDe_req(rcv)->databaseCh!upBdDe_ack(rcv);
    od
}
proctype User(chan userCh, systemCh, databaseCh)
{
    bit rcv;
    do
```

```
    ::systemCh?shoWoP(rcv)->userCh!shoWoP_ack(rcv);
    systemCh?shoWoRe(rcv)->userCh!shoWoRe_ack(rcv);
    systemCh?shoWoDe(rcv)->userCh!shoWoDe_ack(rcv);
    systemCh?shoBdP(rcv)->userCh!shoBdP_ack(rcv);
    systemCh?shoBdRe(rcv)->userCh!shoBdRe_ack(rcv);
    systemCh?shoBdDe(rcv)->userCh!shoBdDe_ack(rcv);
    od
}
init
{
run System(user, system, database);
run Database(user, system, database);
run User(user, system, database);
}
```

Automata for User:

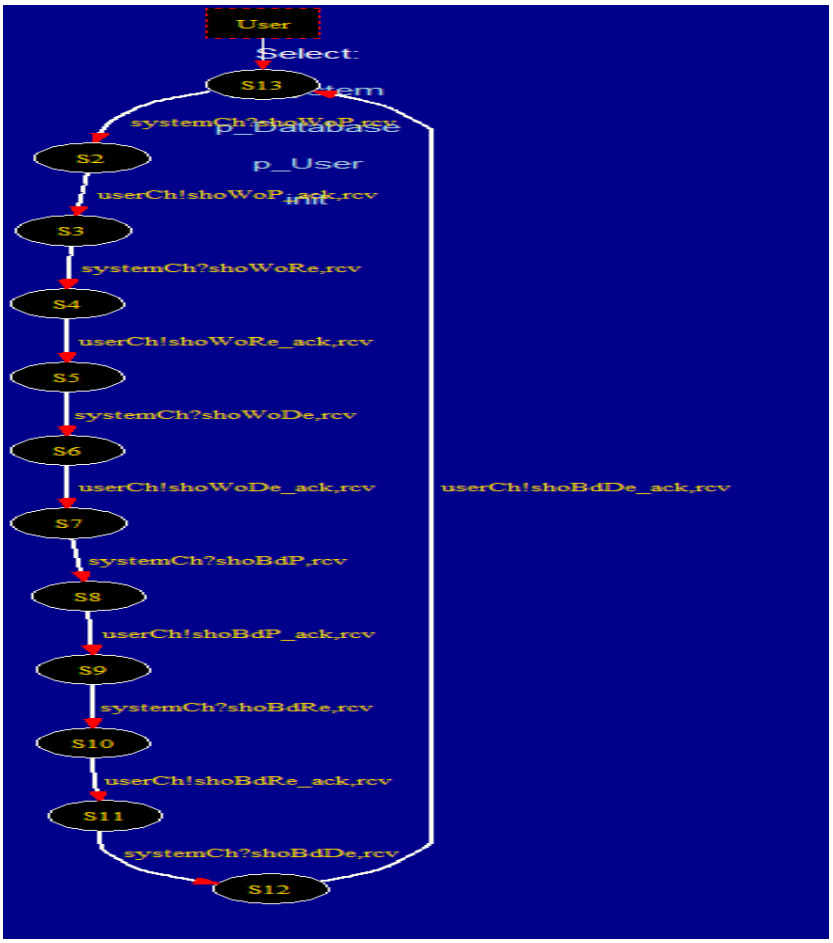


Fig: Automata for User.

Automata for System:

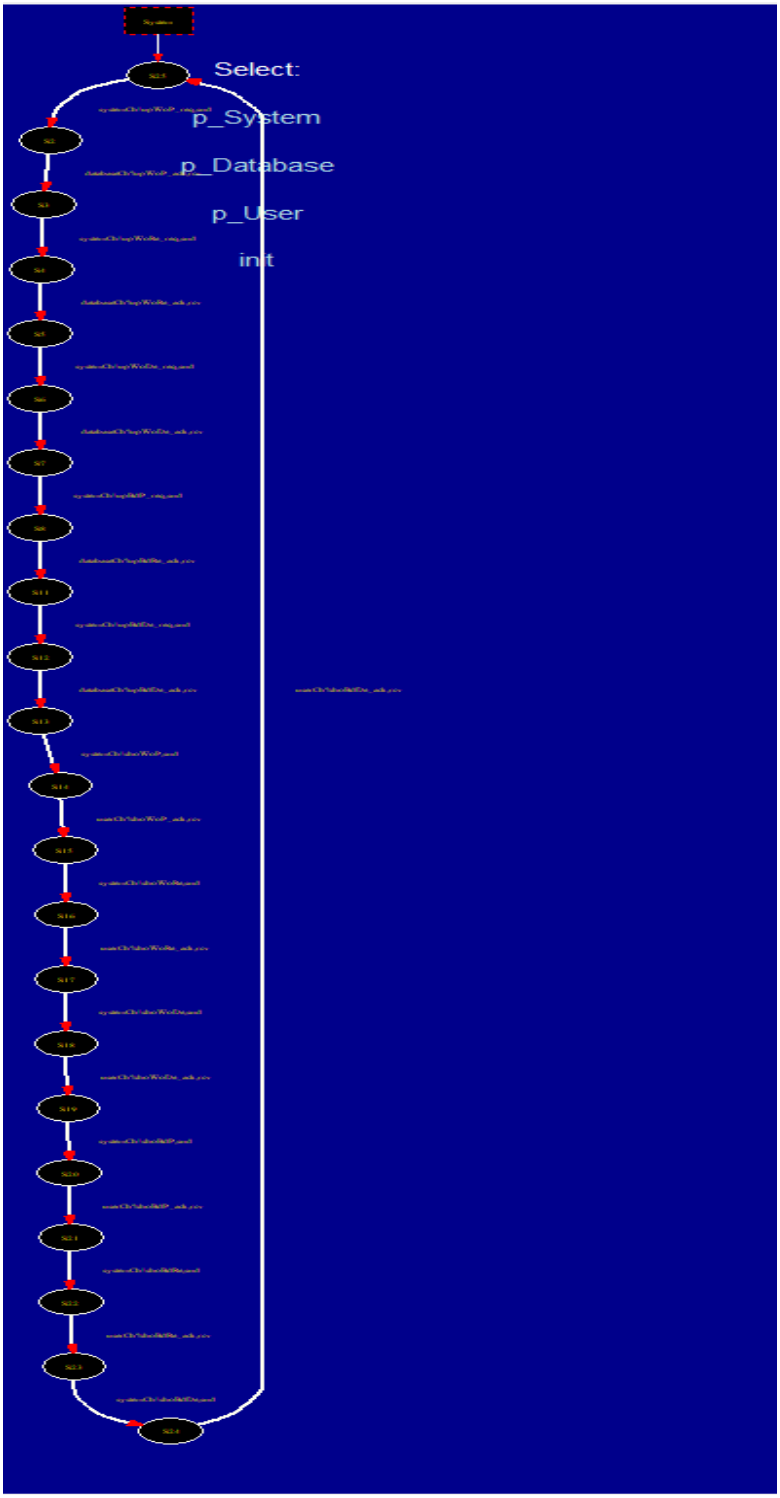


Fig: Automata for System.

Automata for Database:

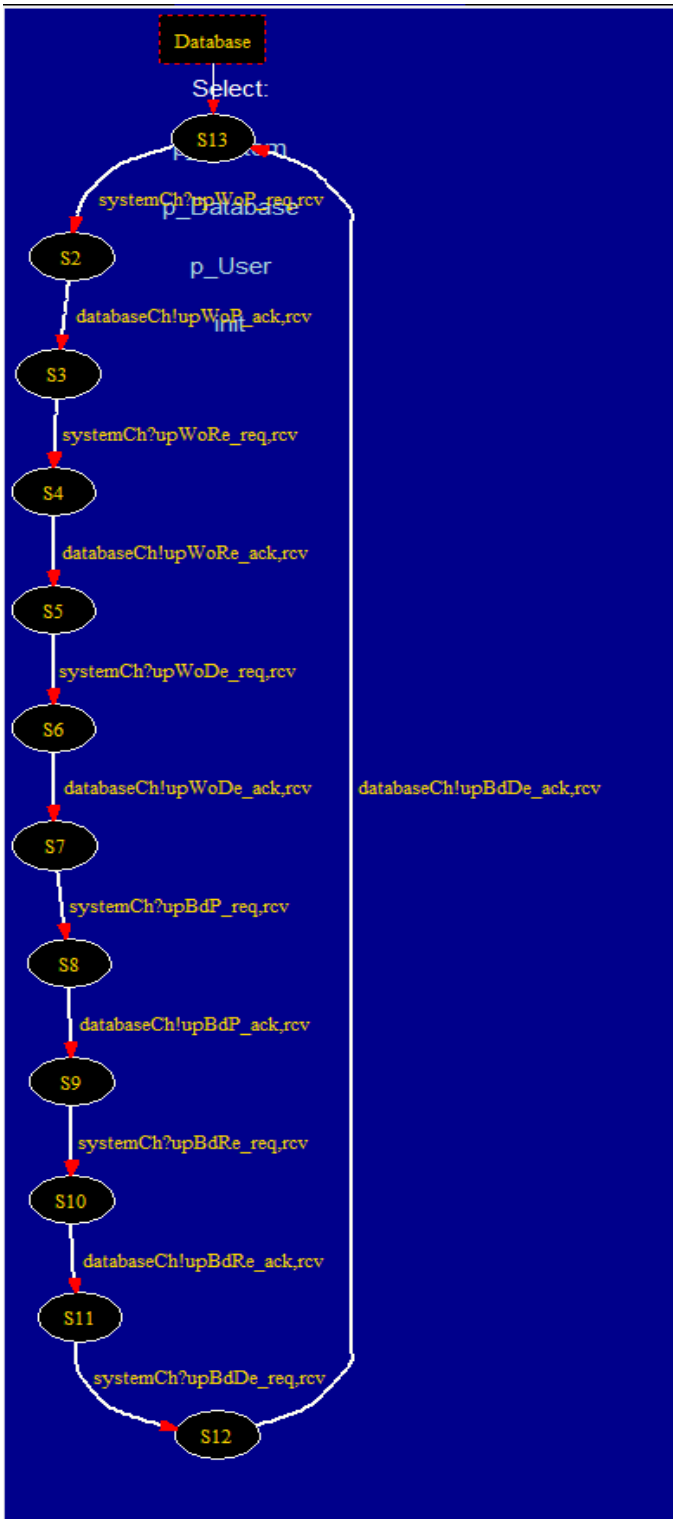


Fig: Automata for Database.

Process Simulation:

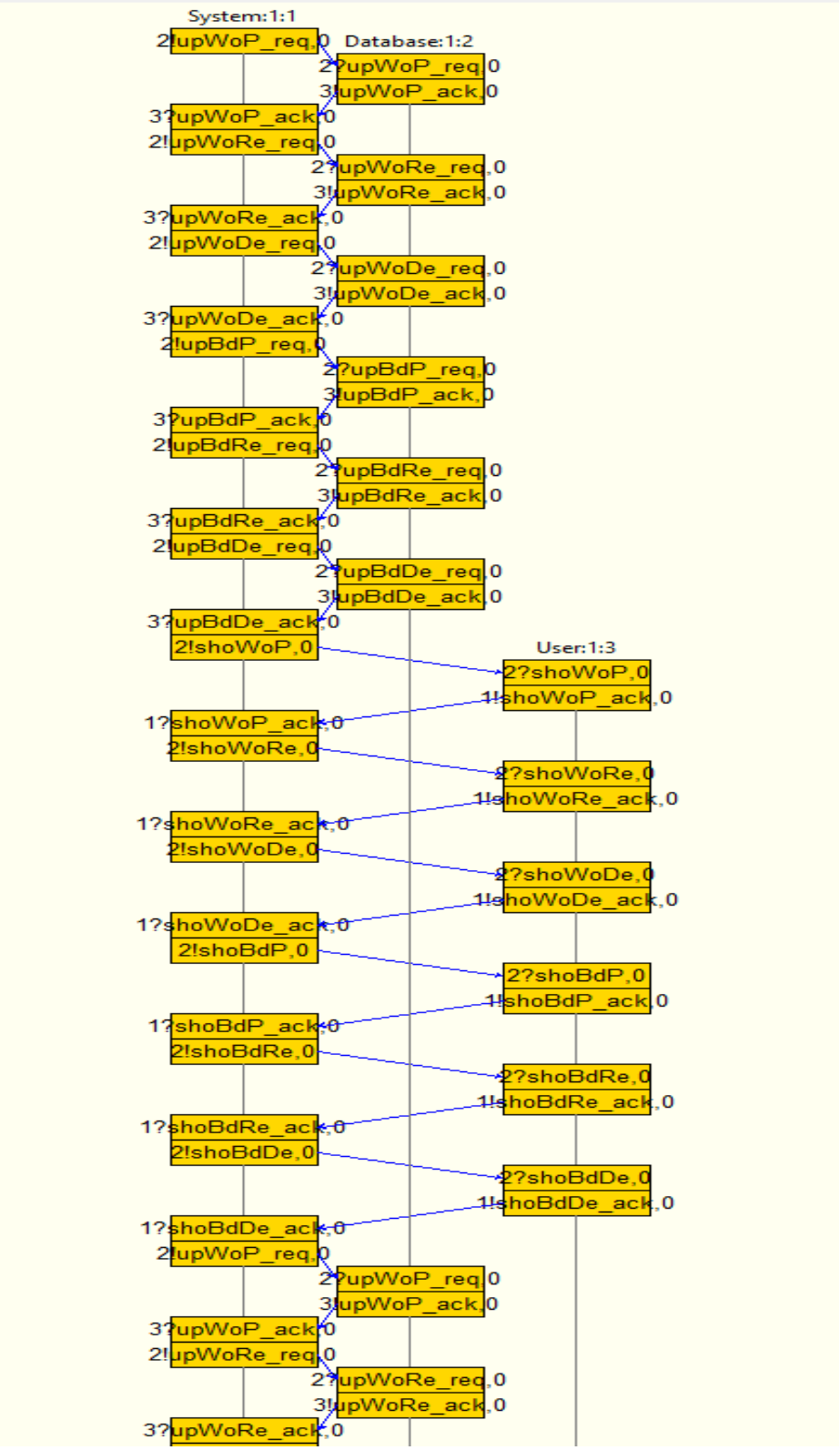


Fig: Process Simulation for Covid News update sequence diagram.

Process Console:

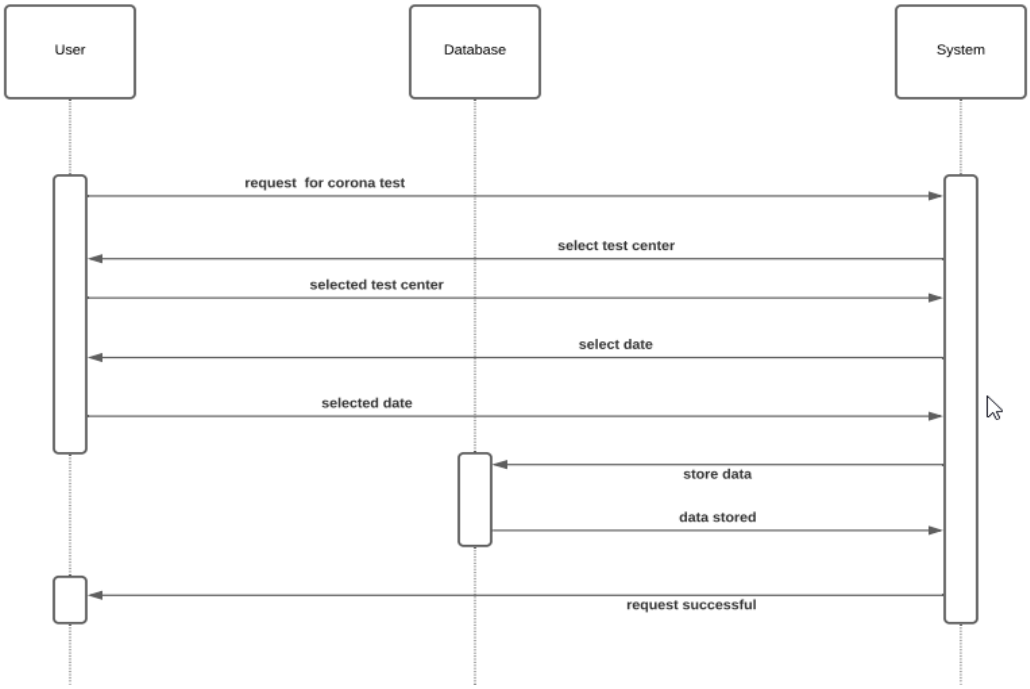
```
 0:     proc  - (:root:) creates proc  0 (:init:)
Starting System with pid 1
 1:     proc  0 (:init::1) creates proc  1 (System)
 1:     proc  0 (:init::1) covid_news.pml:53 (state 1) [(run System(user,system,database))]
 2:     proc  1 (System:1) covid_news.pml:13 (state 1) [systemCh!upWoP_req,snd]
Starting Database with pid 2
 3:     proc  0 (:init::1) creates proc  2 (Database)
 3:     proc  0 (:init::1) covid_news.pml:54 (state 2) [(run Database(user,system,database))]
 4:     proc  2 (Database:1) covid_news.pml:31 (state 1) [systemCh?upWoP_req,rcv]
Starting User with pid 3
 5:     proc  0 (:init::1) creates proc  3 (User)
 5:     proc  0 (:init::1) covid_news.pml:55 (state 3) [(run User(user,system,database))]
 6:     proc  2 (Database:1) covid_news.pml:31 (state 2) [databaseCh!upWoP_ack,rcv]
 7:     proc  1 (System:1) covid_news.pml:13 (state 2) [databaseCh?upWoP_ack,rcv]
 8:     proc  1 (System:1) covid_news.pml:14 (state 3) [systemCh!upWoRe_req,snd]
 9:     proc  2 (Database:1) covid_news.pml:32 (state 3) [systemCh?upWoRe_req,rcv]
18:     proc  2 (Database:1) covid_news.pml:34 (state 8) [databaseCh!upBdP_ack,rcv]
19:     proc  1 (System:1) covid_news.pml:16 (state 8) [databaseCh?upBdP_ack,rcv]
20:     proc  1 (System:1) covid_news.pml:17 (state 9) [systemCh!upBdRe_req,snd]
21:     proc  2 (Database:1) covid_news.pml:35 (state 9) [systemCh?upBdRe_req,rcv]
22:     proc  2 (Database:1) covid_news.pml:35 (state 10) [databaseCh!upBdRe_ack,rcv]
23:     proc  1 (System:1) covid_news.pml:17 (state 10) [databaseCh?upBdRe_ack,rcv]
24:     proc  1 (System:1) covid_news.pml:18 (state 11) [systemCh!upBdDe_req,snd]
25:     proc  2 (Database:1) covid_news.pml:36 (state 11) [systemCh?upBdDe_req,rcv]
26:     proc  2 (Database:1) covid_news.pml:36 (state 12) [databaseCh!upBdDe_ack,rcv]
```

Verification:

```
verification result:
spin -a  covid_news.pml
gcc -DMEMLIM=1024 -O2 -DXUSAFE -DSAFETY -DNOCLAIM -w -o pan pan.c
./pan -m10000
Pid: 10688

(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 100 byte, depth reached 50, errors: 0
     51 states, stored
      1 states, matched
     52 transitions (= stored+matched)
      0 atomic steps
hash conflicts:         0 (resolved)

Stats on memory usage (in Megabytes):
   0.005   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


unreached in proctype System
        covid_news.pml:26, state 28, "-end-"
        (1 of 28 states)
unreached in proctype Database
        covid_news.pml:38, state 16, "-end-"
        (1 of 16 states)
unreached in proctype User
        covid_news.pml:50, state 16, "-end-"
        (1 of 16 states)
unreached in init
        (0 of 4 states)

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

Fig: Verification for Covid News Update Sequence Diagram.

Sequence diagram of Test Request:



Promela code for Test Request Sequence diagram:

```
mtype = {req_for_corona_test, select_test_center, selected_test_center,
        select_date, selected_date, store_data, data_stored,
        request_successful, ack};

chan user = [2] of {mtype, bit};
chan system = [2] of {mtype, bit};
chan database = [2] of {mtype, bit};

proctype User(chan userCh, systemCh, databaseCh)
{
    bit snd,rcv;
    do
    :: userCh!req_for_corona_test(snd)->systemCh?select_test_center(rcv);
    userCh!selected_test_center(snd)->systemCh?select_date(rcv);
    userCh!selected_date(snd)->systemCh?request_successful(rcv);

    od
}

proctype System(chan userCh, systemCh, databaseCh)
{
    bit snd,rcv;
```

```
    do
    ::userCh?req_for_corona_test(rcv)->systemCh!select_test_center(rcv);
    userCh?selected_test_center(rcv)->systemCh!select_date(rcv);
    userCh?selected_date(rcv)->systemCh!store_data(rcv);
    databaseCh?data_stored(snd)->systemCh!request_successful(rcv);
    od
}

proctype Database(chan userCh, systemCh, databaseCh)
{
    bit rcv;
    do
    ::systemCh?store_data(rcv)->databaseCh!data_stored(rcv);
    od
}

init
{
run User(user, system, database);
run System(user, system, database);
run Database(user, system, database);
}
```

Automata for User:

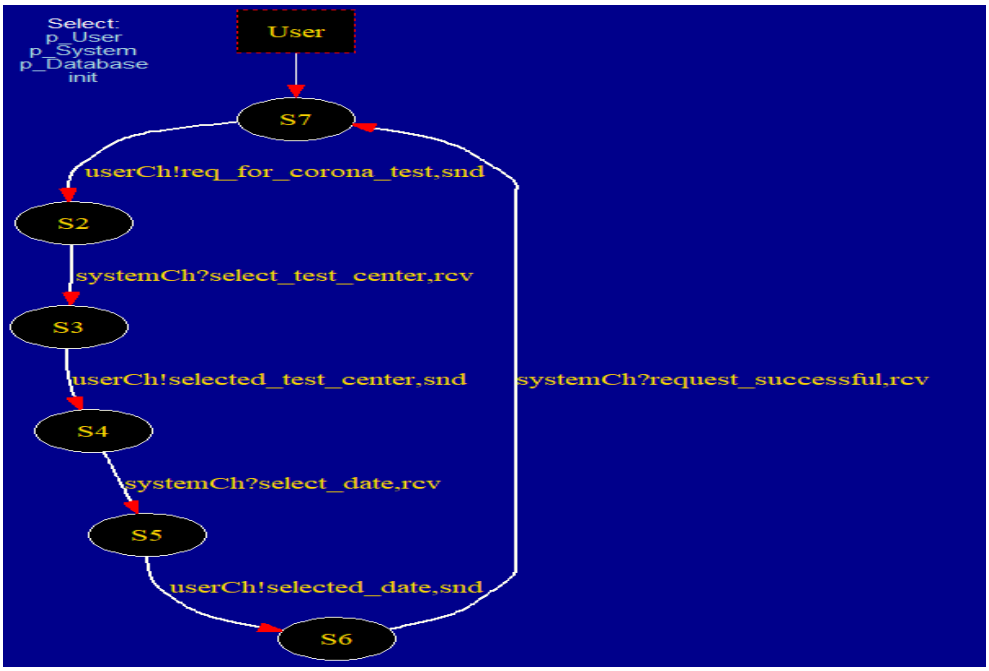


Fig: Automata for user.

Automata for System:

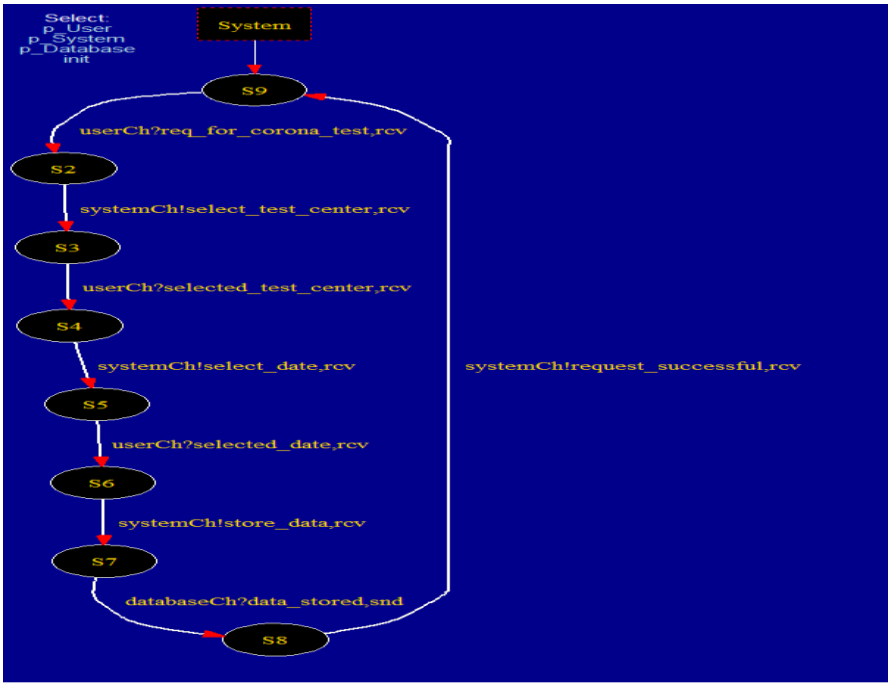


Fig: Automata for system.

Automata for Database:

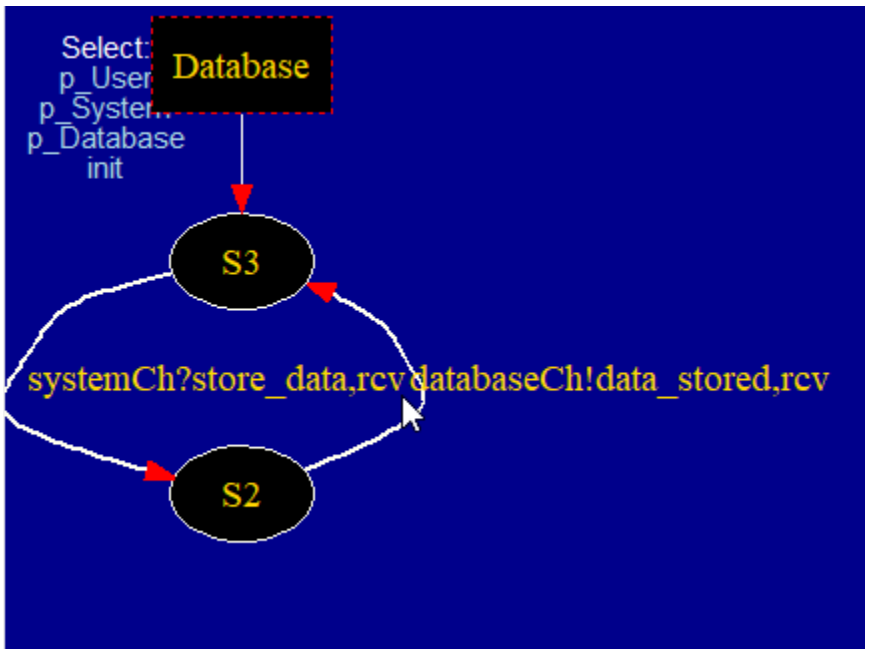


Fig: Automata for Database.

Process Simulation:

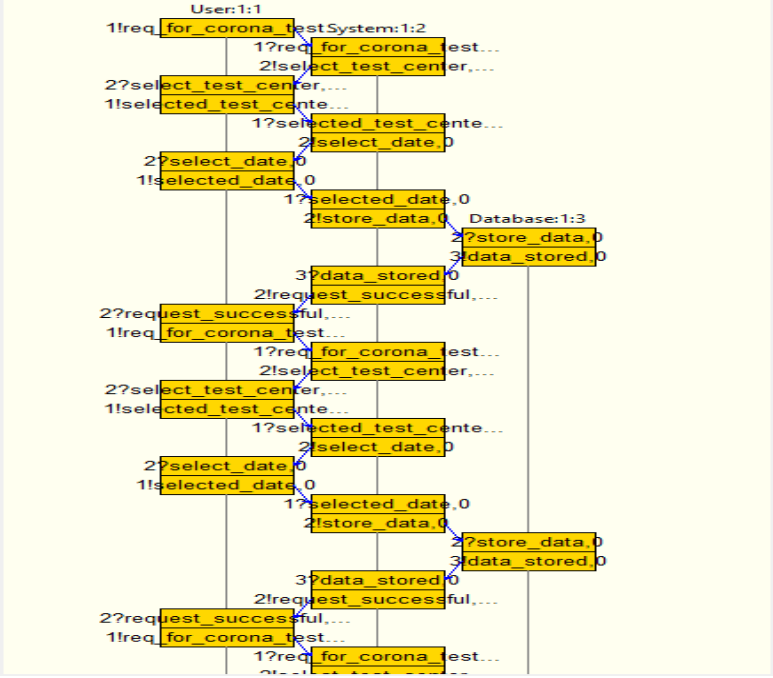


Fig: Process Simulation for Test Request Sequence diagram.

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) testcenter.pml:39 (state 1) [(run User(user,system,database))]
 2:      proc  1 (User:1) testcenter.pml:11 (state 1) [userCh!req_for_corona_test,snd]
Starting System with pid 2
 3:      proc  0 (:init::1) creates proc  2 (System)
 3:      proc  0 (:init::1) testcenter.pml:40 (state 2) [(run System(user,system,database))]
 4:      proc  2 (System:1) testcenter.pml:22 (state 1) [userCh?req_for_corona_test,rcv]
Starting Database with pid 3
 5:      proc  0 (:init::1) creates proc  3 (Database)
 5:      proc  0 (:init::1) testcenter.pml:41 (state 3) [(run Database(user,system,database))]
 6:      proc  2 (System:1) testcenter.pml:22 (state 2) [systemCh!select_test_center,rcv]
 7:      proc  1 (User:1) testcenter.pml:11 (state 2) [systemCh?select_test_center,rcv]
 8:      proc  1 (User:1) testcenter.pml:12 (state 3) [userCh!selected_test_center,snd]
 9:      proc  2 (System:1) testcenter.pml:23 (state 3) [userCh?selected_test_center,rcv]
 10:     proc  2 (System:1) testcenter.pml:23 (state 4) [systemCh!select_date,rcv]
 11:     proc  1 (User:1) testcenter.pml:12 (state 4) [systemCh?select_date,rcv]
 12:     proc  1 (User:1) testcenter.pml:13 (state 5) [userCh!selected_date,snd]
 13:     proc  2 (System:1) testcenter.pml:24 (state 5) [userCh?selected_date,rcv]
 14:     proc  2 (System:1) testcenter.pml:24 (state 6) [systemCh!store_data,rcv]
 15:     proc  3 (Database:1) testcenter.pml:33 (state 1) [systemCh?store_data,rcv]
 16:     proc  3 (Database:1) testcenter.pml:33 (state 2) [databaseCh!data_stored,rcv]
 17:     proc  2 (System:1) testcenter.pml:25 (state 7) [databaseCh?data_stored,snd]
 19:     proc  2 (System:1) testcenter.pml:25 (state 8) [systemCh!request_successful,rcv]
 20:     proc  1 (User:1) testcenter.pml:13 (state 6) [systemCh?request_successful,rcv]
```

Fig: Process console for Test Request sequence diagram.

Verification:

```
verification result:
spin -a  testcenter.pml
gcc -DMEMLIM=1024 -O2 -DXUSAFE -DSAFETY -DNOCLAIM -w -o pan pan.c
./pan -m10000
Pid: 2720

(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 100 byte, depth reached 18, errors: 0
      19 states, stored
       1 states, matched
      20 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))
   0.291   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
        testcenter.pml:16, state 10, "-end-"
        (1 of 10 states)
unreached in proctype System
        testcenter.pml:27, state 12, "-end-"
        (1 of 12 states)
unreached in proctype Database
        testcenter.pml:35, state 6, "-end-"
        (1 of 6 states)
unreached in init
        (0 of 4 states)

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

Fig: Verification for Test Request 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.