East West University

Course Title : **Software Quality Assurance**
Course Code : CSE 435
Section : 01
Semester : Fall 2020

Project topic: Online Ambulance Service

Submitted By

Nasif Wasek Fahim
2017-1-60-037
Mueem Nahid Ibn Mahbub
2017-1-60-089
Md. Jubayer Hossain Abir
2017-1-60-084

Submission Date

09/12/2020

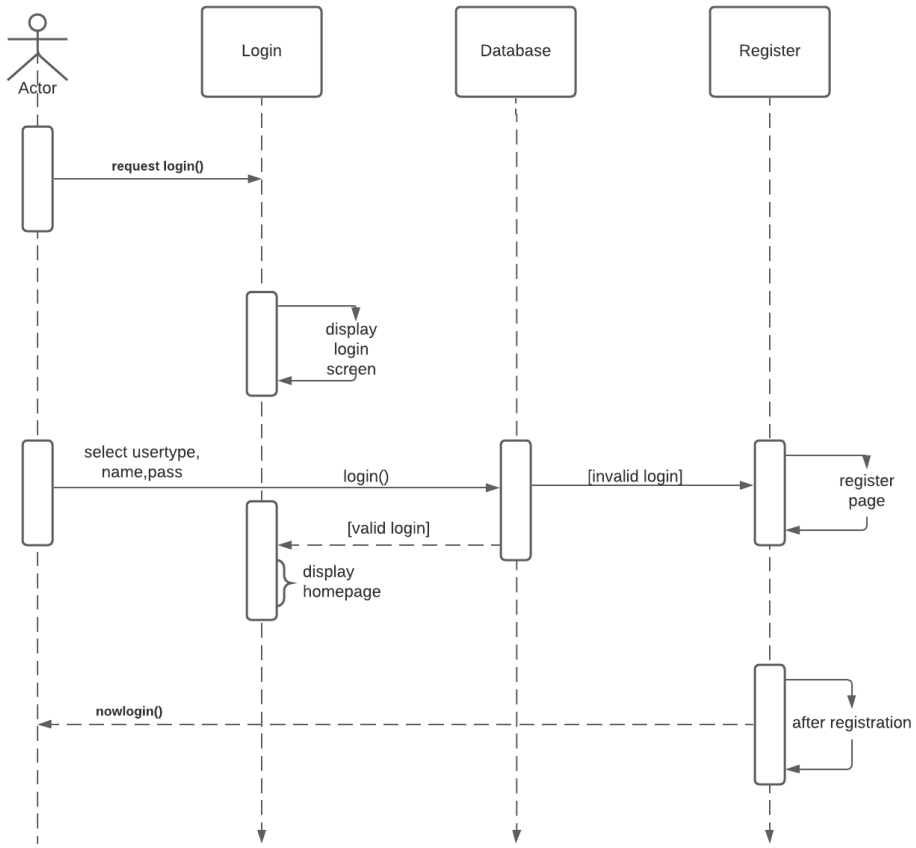
Online Ambulance Service

Introduction:

This is an integrated service that provides all information about the location of the ambulance and its routes for the public. The proposed system is a web-based application that provides information regarding ambulance quality, routes, fair. The system contains an online order system where service recipients can order any nearby ambulance service and service providers can receive the given order. There is also an admin module where the admin can add ambulance, scrap ambulance, and any update. The admin is a panel that will be consisted of a group of authorized persons.

1st Sequence Diagram:

Login System Sequence Diagram:



The functionality of Login Sequence Diagram:

In this system actor (admin, service receiver, service provider) would try to log in to the system and for that, the actor would enter user type, username, and password. After this, the information would be verified and checked into the database. If the information is legit then the actor would be able to enter the homepage. Otherwise, the system would take the user to the registration page for registration.

FSP Code for the Login System:

Login = (request_login -> receive_validity -> display_Homepage -> Login).

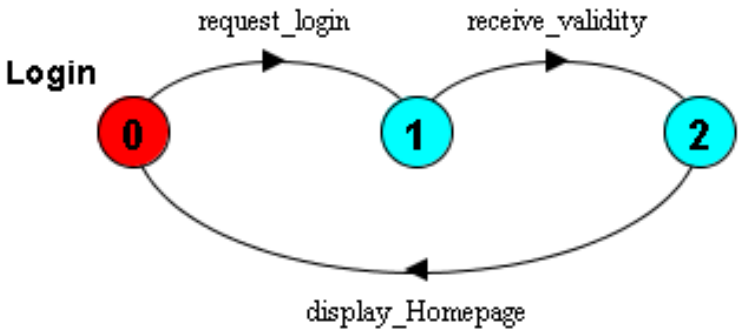
Database = (login_request -> check_validity -> send_validation -> Database).

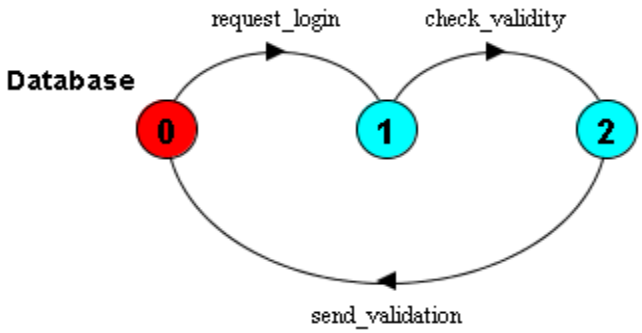
Register = (invalid_login -> do_registration -> response -> Register).

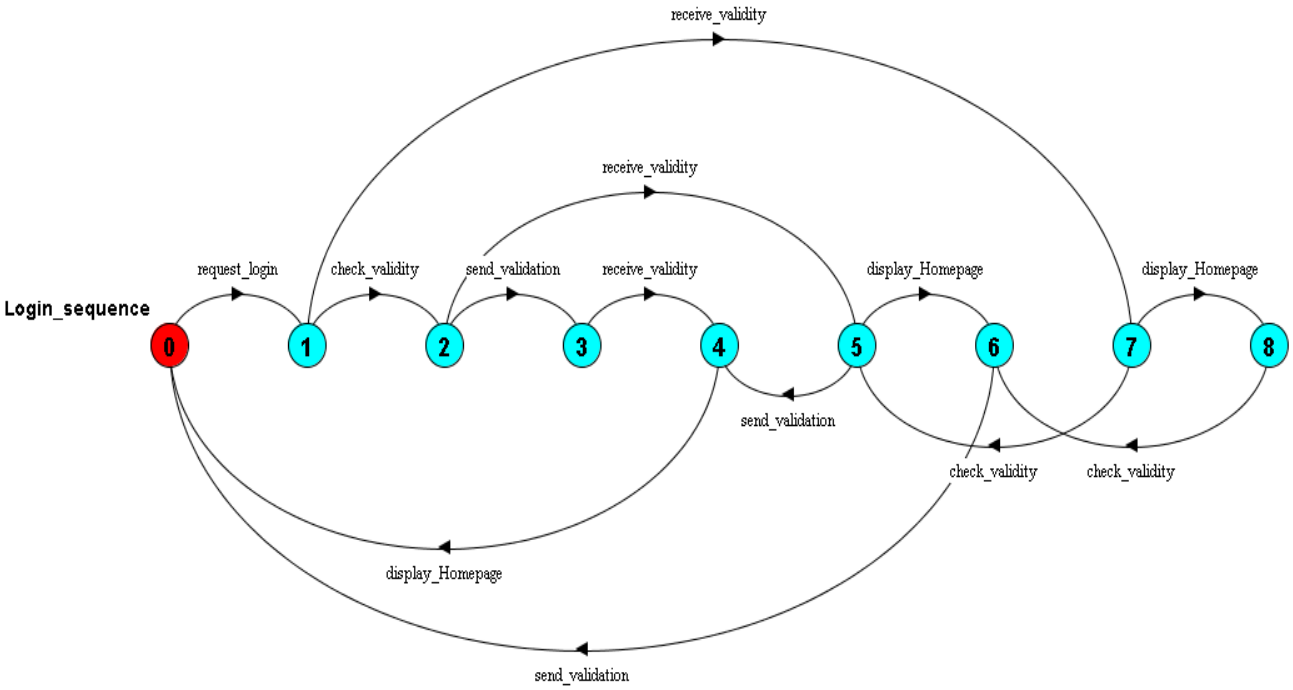
||Login_sequence = (Login || Database)

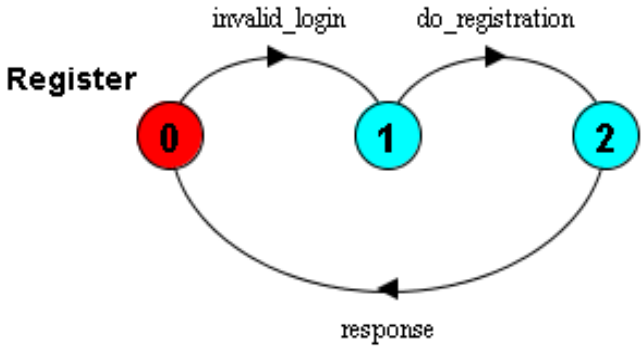
/{request_login / login_request, receive_validity / send_validity}.

State Transition Diagram for Login System:



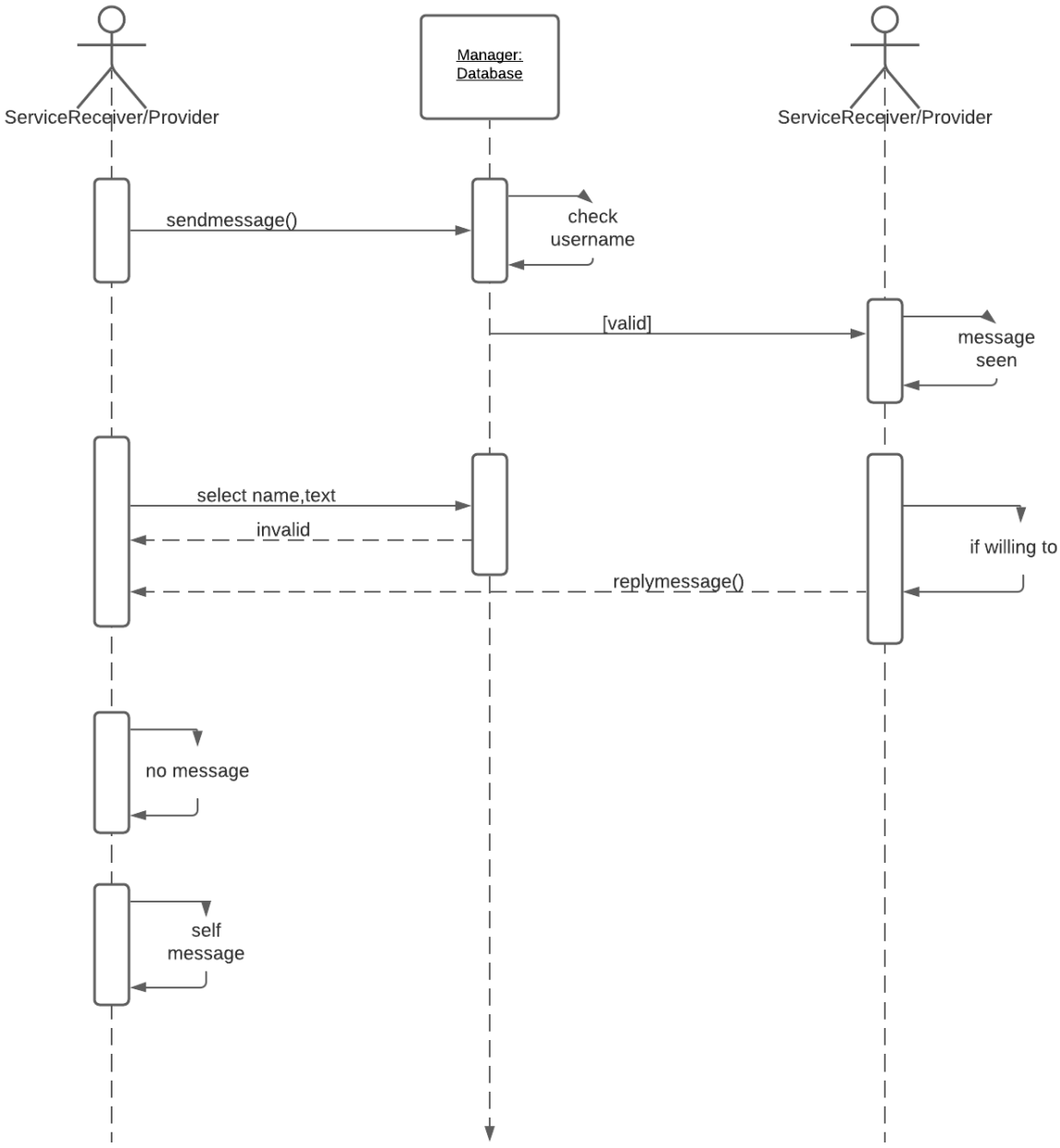

request_login
check_validity
Database
0
1
2
send_validation


receive_validity
receive_validity
request_login
check_validity
send_validation
receive_validity
display_Homepage
display_Homepage
Login_sequence
0
1
2
3
4
5
6
7
8
send_validation
check_validity
check_validity
display_Homepage
send_validation


invalid_login
do_registration
Register
0
1
2
response

2nd Sequence Diagram:

Inbox System Sequence Diagram:



Sequence diagram of
Inbox System

The functionality of Inbox Sequence Diagram:

In this system, the service receiver can send messages to the service provider in case of any need and vice versa. The system also allows service receivers to send message to other service receivers and the other way around. There is also a feature where the user can message oneself. First, a user selects a username to text and sends the text. After the message is sent by the user, the user will have to wait for a response. After the message is seen by the recipient then the recipient can reply, and the sender would get the message.

FSP Code for the Inbox System:

Actor1= (select_name_text -> send_message -> wait_for_response -> receive_reply -> Actor1).

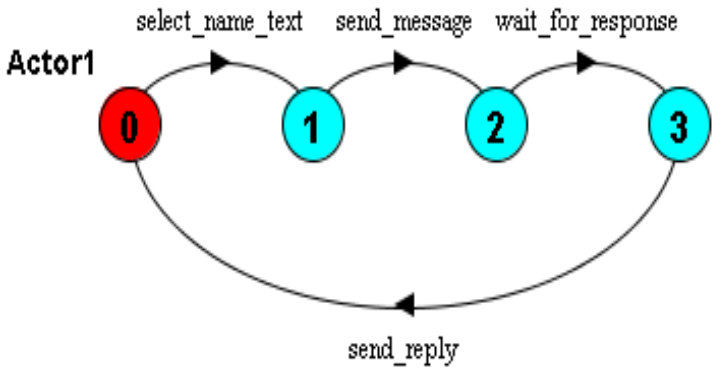
Actor2= (receive_message -> select_text -> send_reply -> check_message_sent -> Actor2).

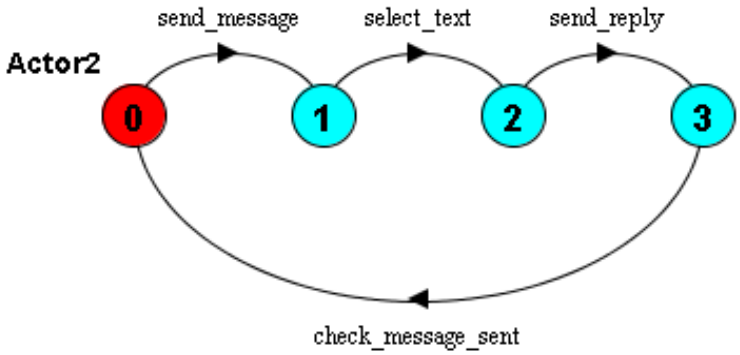
Manager =(receive_request -> check_validity -> send_response -> Manager).

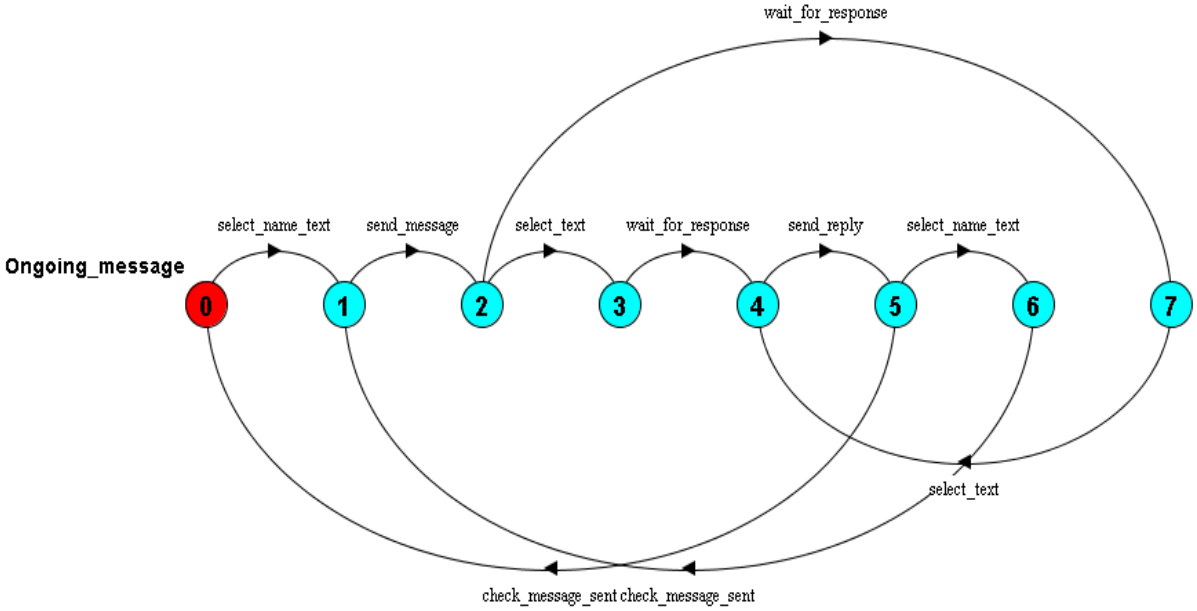
||Ongoing_message = (Actor1 || Actor2)

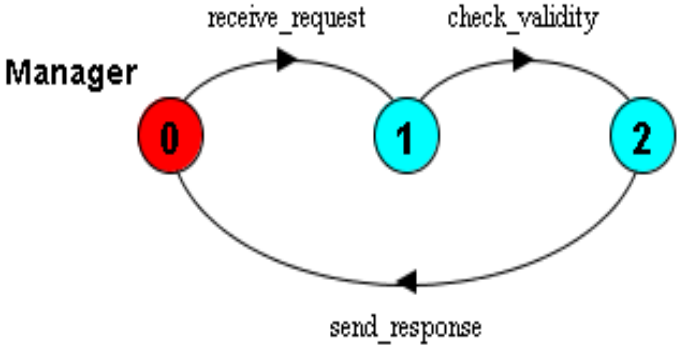
/{send_message / receive_message, send_reply / receive_reply}.

State Transition Diagram for Inbox System:



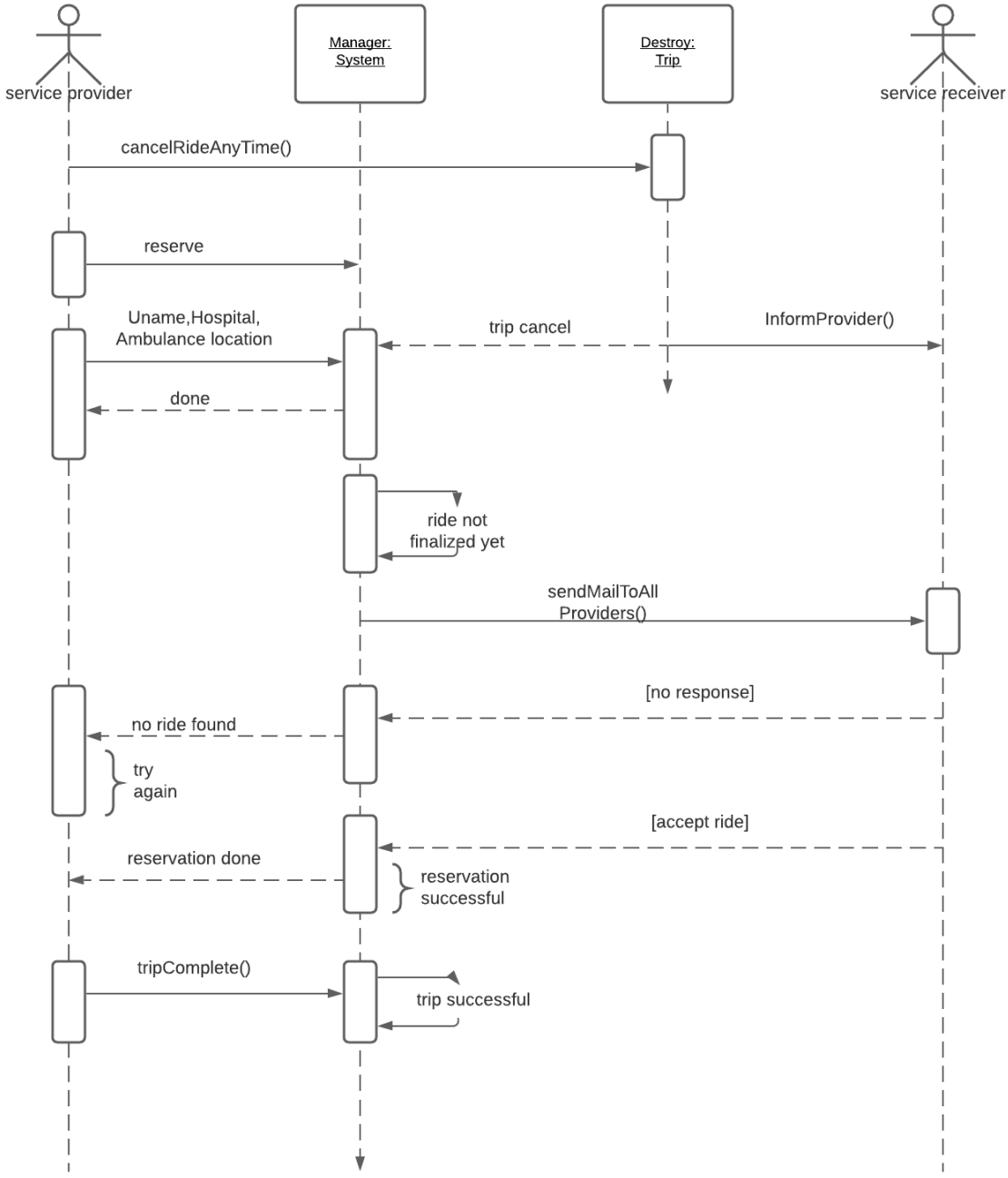

send_message
select_text
send_reply
Actor2
0
1
2
3
check_message_sent


wait_for_response
select_name_text
send_message
select_text
wait_for_response
send_reply
select_name_text
Ongoing_message
0
1
2
3
4
5
6
7
select_text
check_message_sent
check_message_sent


receive_request
check_validity
Manager
0
1
2
send_response

3rd Sequence Diagram:

Reservation System Sequence Diagram:



Reservation System

The functionality of Reservation System Sequence Diagram:

This system allows a service receiver to book an ambulance in time of need. First, the service receiver fills up the required location for the trip like providing the current location, destination, type of ambulance, etc. After submitting this information, this would go to the database and the database would send mail to all possible providers for informing about the trip. The available provider would click on the agree button and the response would go to the sender. After the trip is complete, the completion record would be stored on the database. If no rider was found, then the sender would be asked to try again. A service receiver can cancel a ride at any time and ride cancelation would notify the service provider.

FSP Code for the Reservation System:

Service_receiver = (select_hospital_name_location -> reserve -> wait_for_response -> receive_response -> reservation_successful -> ride_complete -> Service_receiver).

Service_provider = (notification_for_new_trip -> accept_ride -> trip_successful -> trip_complete -> Service_provider).

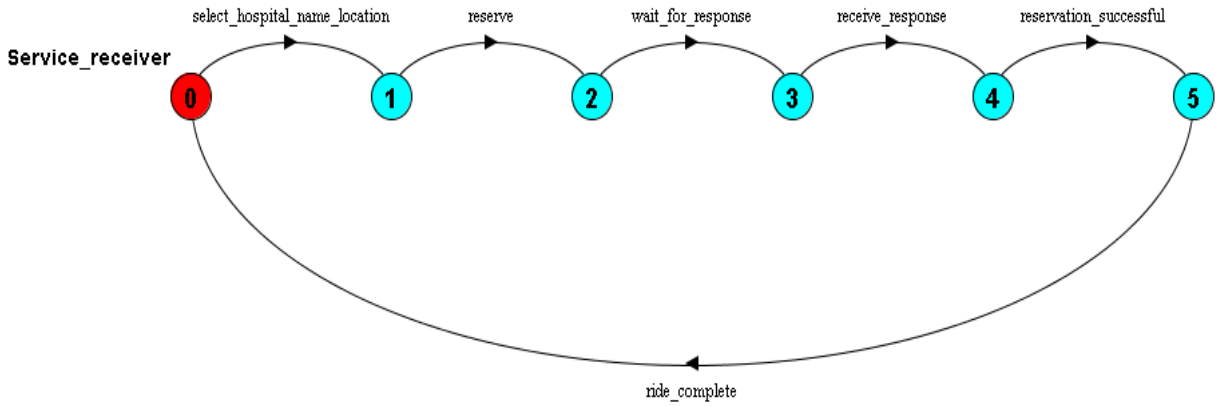
System = (getting_request -> ride_not_finalized_yet -> send_mail_to_all_providers -> rider_accepted -> store_record -> System).

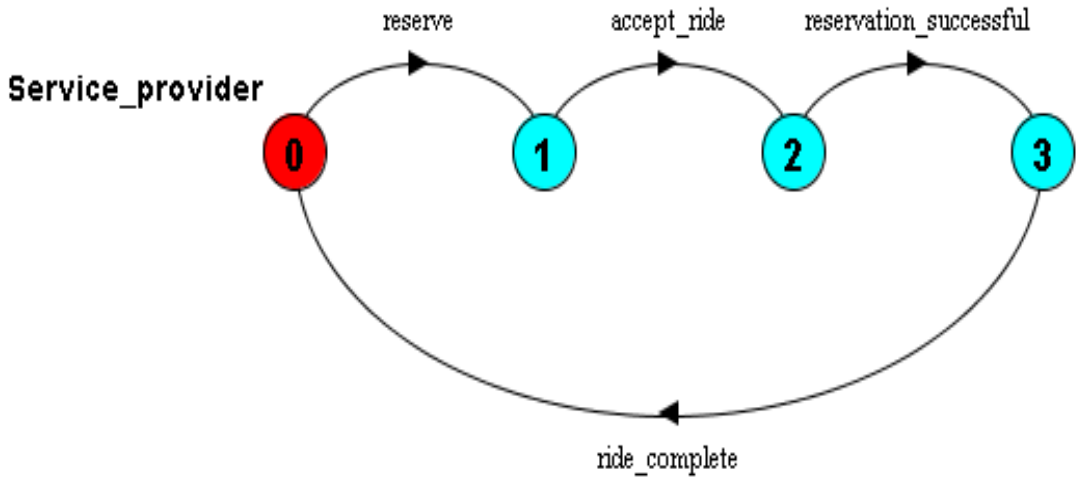
Destroy_trip = (cancel_ride -> inform_provider -> store_info -> Destroy_trip).

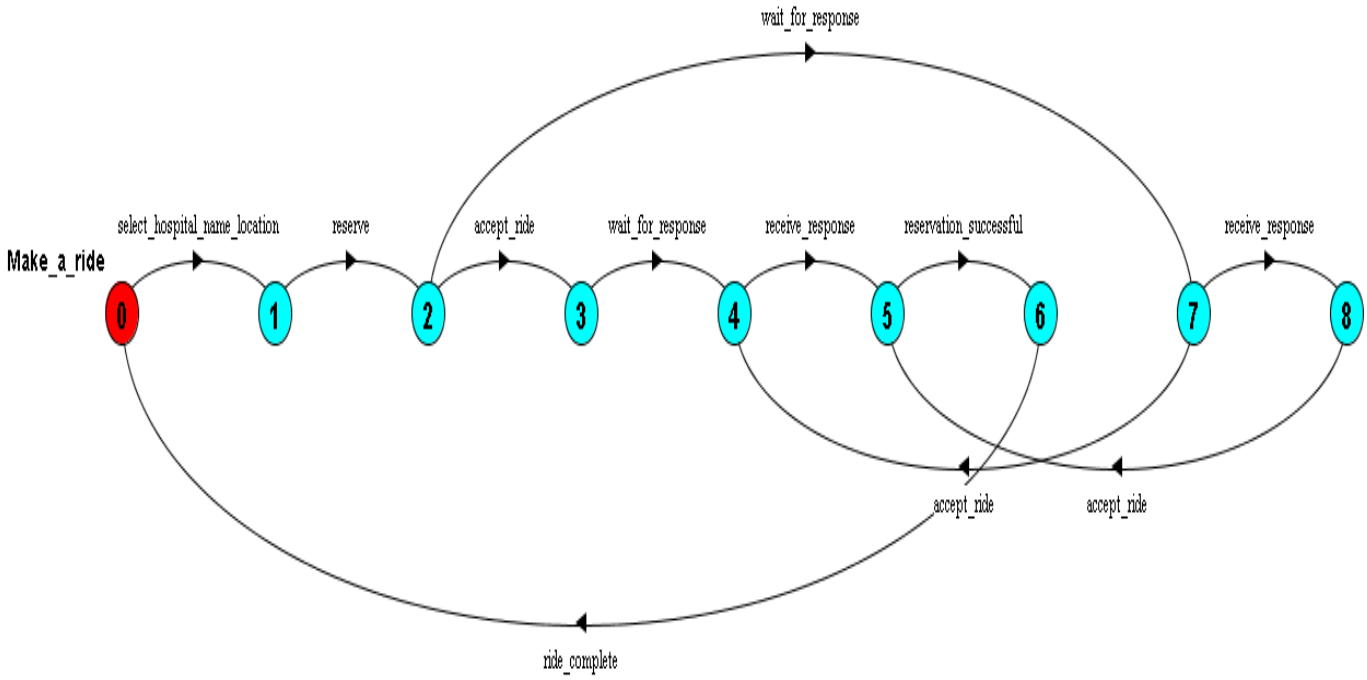
||Make_a_ride = (Service_receiver || Service_provider)

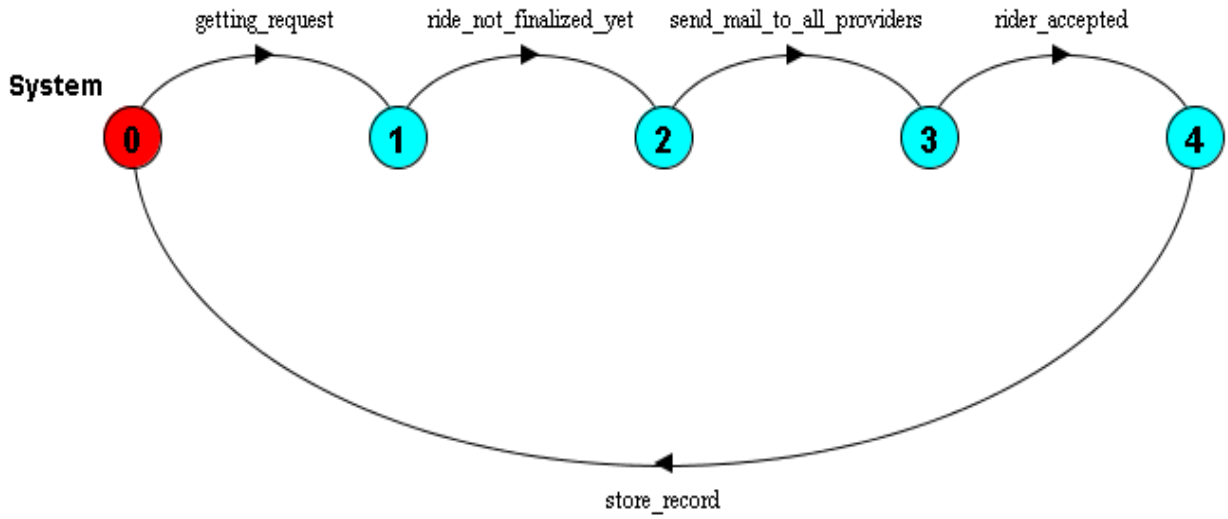
/{reserve/notification_for_new_trip,accept_ride/receive_resposnse,
reservation_successful/trip_successful,
ride_complete/trip_complete}.

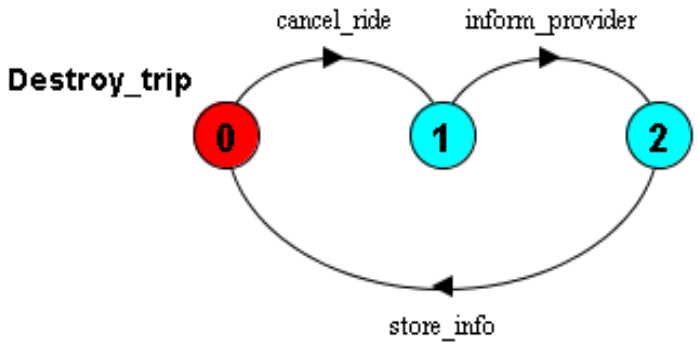
State Transition Diagram for Reservation System:






Make_a_ride
0
1
2
3
4
5
6
7
8
select_hospital_name_location
reserve
accept_ride
wait_for_response
receive_response
reservation_successful
wait_for_response
receive_response
accept_ride
accept_ride
ride_complete


System
0
1
2
3
4
getting_request
ride_not_finalized_yet
send_mail_to_all_providers
rider_accepted
store_record


Destroy_trip
cancel_ride
inform_provider
0
1
2
store_info