TOUR MANAGEMENT

Previously people wishing to visit places had to manually search for available accommodation at the visiting places. Also they themselves had to make reservation. People hardly had any knowledge of which are the worth seeing places and about its history. Such procedure was time consuming and energy wasting.

Tour Reservation System has made life very easy for such visitors by saving both their time and energy.
Visitor requests for scheme to check the availability of the desired tour package. This information is stored in Tour Information System.
System will check whether the customer is existing or new. New user will enter his personal and tour details for reservation. In turn he/she will be provided with system generated unique ID and password. This login information could be used for further transactions.

When customer is satisfied with tour package he/she would request for reservation of tour. Personal details of new customer is stored in cust_info while the details regarding the tour selected by particular customer is stored in tour_info and the details regarding it would be restructured in Tour Information System.
Existing customer can update his/her personal details in cust_info and cancel reservation for tour from tour_info and changes regarding it are also reflected in Tour Information System.
After confirming the tour package the customer will make payment either online or through staff by personally going at the office. Customer can make payment by cash, credit card or by cheque.
System checks for the validity of staff. Once the payment is done by customer, valid staff will make Ticket Reservation System. .

Reserved customer will be able to view details about reservation by providing login information from cust_info and tour_info system.
Administrator can add, delete or modify tour schemes from Tour Information System.

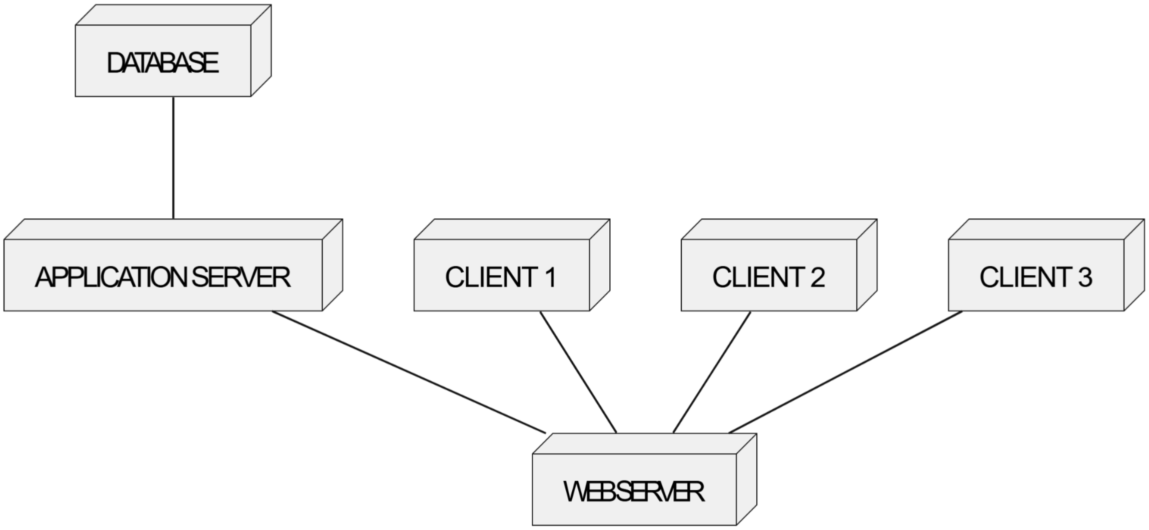
UML DIAGRAMS:

1) DEPLOYMENT DIAGRAM:

CODE:

```
@startuml
node "DATABASE" as DB
node "APPLICATION SERVER" as AppServer
node "WEB SERVER" as WebServer
node "CLIENT 1" as Client1
node "CLIENT 2" as Client2
node "CLIENT 3" as Client3

DB -- AppServer
AppServer -- WebServer
Client1 -- WebServer
Client2 -- WebServer
Client3 -- WebServer
@enduml
```

DIAGRAM:



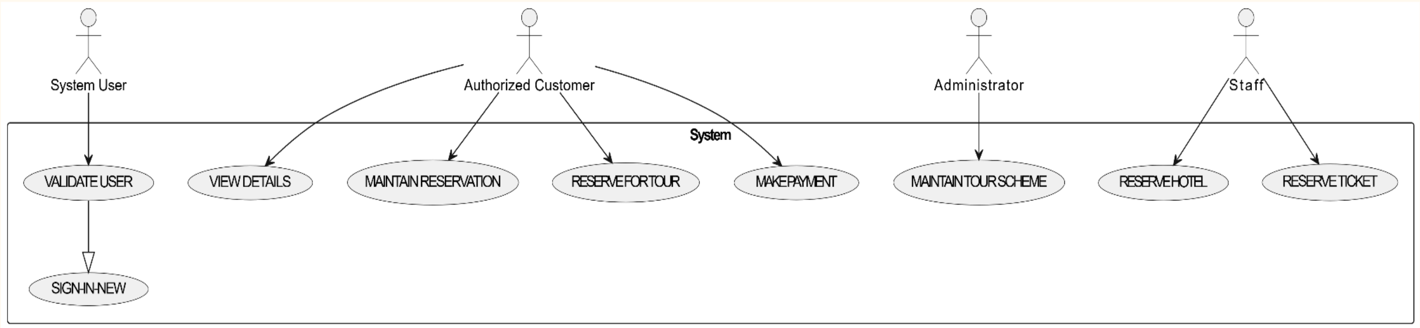
2)USE CASE DIAGRAM:

CODE:

```
@startuml
actor "System User"
actor "Authorized Customer"
actor "Administrator"
actor "Staff"

rectangle System {
  usecase "SIGN-IN-NEW" as signin
  usecase "VALIDATE USER" as validate
  usecase "RESERVE FOR TOUR" as reserve
```

```
 usecase "MAINTAIN RESERVATION" as maintain
 usecase "VIEW DETAILS" as view
 usecase "MAKE PAYMENT" as payment
 usecase "MAINTAIN TOUR SCHEME" as maintainScheme
 usecase "RESERVE TICKET" as reserveTicket
 usecase "RESERVE HOTEL" as reserveHotel

 "System User" --> validate
 validate --|> signin
 "Authorized Customer" --> reserve
 "Authorized Customer" --> maintain
 "Authorized Customer" --> view
 "Authorized Customer" --> payment
 "Administrator" --> maintainScheme
 "Staff" --> reserveTicket
 "Staff" --> reserveHotel
}
@enduml
```

DIAGRAM:



3)CLASS DIAGRAM:

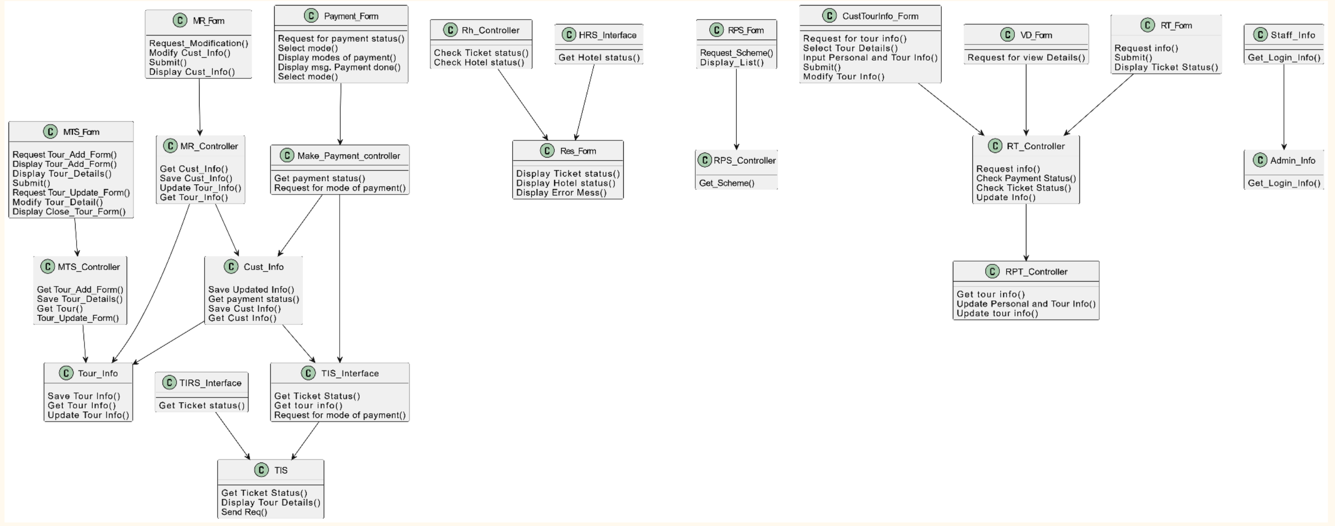
Code:

```
@startuml
class MR_Form {
   Request_Modification()
   Modify Cust_Info()
   Submit()
   Display Cust_Info()
}

class MR_Controller {
   Get Cust_Info()
   Save Cust_Info()
   Update Tour_Info()
   Get Tour_Info()
}

class MTS_Form {
   Request Tour_Add_Form()
   Display Tour_Add_Form()
   Display Tour_Details()
   Submit()
   Request Tour_Update_Form()
   Modify Tour_Detail()
   Display Close_Tour_Form()
}

class MTS_Controller {
   Get Tour_Add_Form()
   Save Tour_Details()
   Get Tour()
   Tour_Update_Form()
}

class Make_Payment_controller {
   Get payment status()
   Request for mode of payment()
}

class Payment_Form {
   Request for payment status()
   Select mode()
   Display modes of payment()
   Display msg. Payment done()
   Select mode()
}

class Rh_Controller {
   Check Ticket status()
   Check Hotel status()
}

class Res_Form {
   Display Ticket status()
   Display Hotel status()
   Display Error Mess()
}

class RPS_Form {
   Request_Scheme()
   Display_List()
}

class RPS_Controller {
   Get_Scheme()
}

class CustTourInfo_Form {
   Request for tour info()
   Select Tour Details()
   Input Personal and Tour Info()
   Submit()
   Modify Tour Info()
}

class VD_Form {
   Request for view Details()
}

class RT_Form {
   Request info()
   Submit()
   Display Ticket Status()
}

class RT_Controller {
   Request info()
   Check Payment Status()
   Check Ticket Status()
   Update Info()
}

class RPT_Controller {
   Get tour info()
   Update Personal and Tour Info()
   Update tour info()
}
```

```
class Cust_Info {
  Save Updated Info()
  Get payment status()
  Save Cust Info()
  Get Cust Info()
}

class Tour_Info {
  Save Tour Info()
  Get Tour Info()
  Update Tour Info()
}

class TIS_Interface {
  Get Ticket Status()
  Get tour info()
  Request for mode of payment()
}

class TIS {
  Get Ticket Status()
  Display Tour Details()
  Send Req()
}

class Staff_Info {
  Get_Login_Info()
}

class Admin_Info {
  Get_Login_Info()
}

class HRS_Interface {
  Get Hotel status()
}

class TIRS_Interface {
  Get Ticket status()
}

MR_Form --> MR_Controller
```

```
MTS_Form --> MTS_Controller
Payment_Form --> Make_Payment_controller
CustTourInfo_Form --> RT_Controller
VD_Form --> RT_Controller
RT_Form --> RT_Controller
RT_Controller --> RPT_Controller
MR_Controller --> Cust_Info
MR_Controller --> Tour_Info
MTS_Controller --> Tour_Info
Make_Payment_controller --> Cust_Info
Make_Payment_controller --> TIS_Interface
Rh_Controller --> Res_Form
RPS_Form --> RPS_Controller
Cust_Info --> TIS_Interface
Cust_Info --> Tour_Info
TIS_Interface --> TIS
Staff_Info --> Admin_Info
HRS_Interface --> Res_Form
TIRS_Interface --> TIS
```

DIAGRAM:

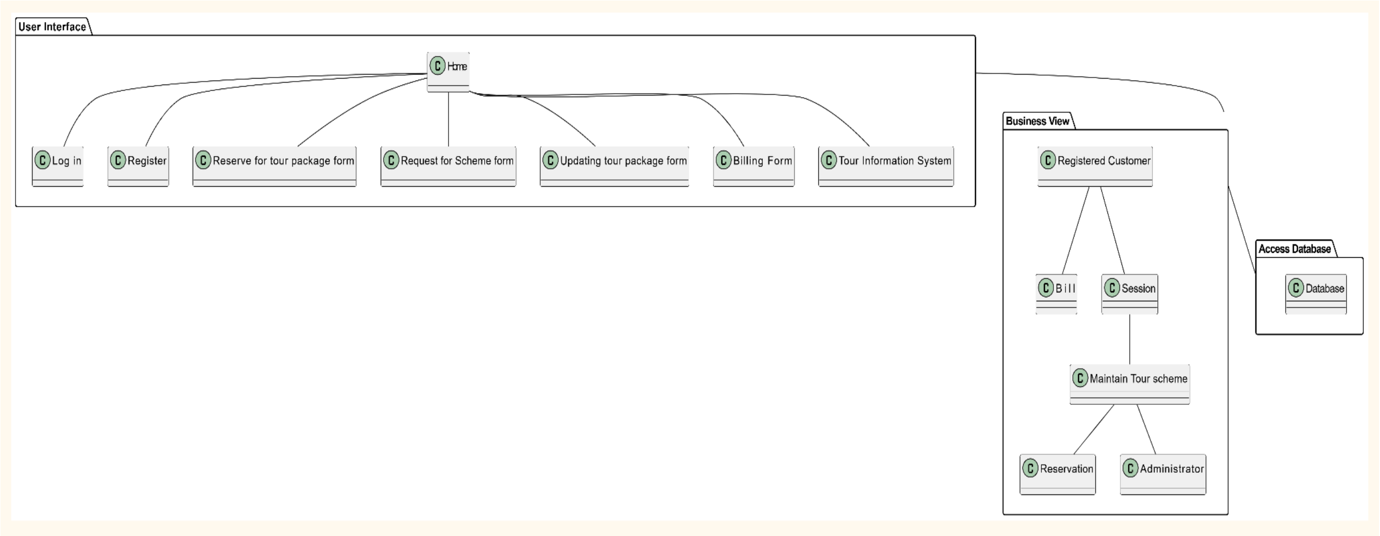


4) COMPONENT DIAGRAM:

CODE:

```
@startuml
package "User Interface" {
 class Home
 class "Log in"
 class "Register"
 class "Reserve for tour package form"
 class "Request for Scheme form"
 class "Updating tour package form"
 class "Billing Form"
 class "Tour Information System"

 Home -- "Log in"
 Home -- "Register"
 Home -- "Reserve for tour package form"
 Home -- "Request for Scheme form"
 Home -- "Updating tour package form"
 Home -- "Billing Form"
 Home -- "Tour Information System"
}

package "Business View" {
 class "Registered Customer"
 class "Bill"
 class "Session"
 class "Maintain Tour scheme"
 class "Reservation"
 class "Administrator"

 "Registered Customer" -- "Bill"
 "Registered Customer" -- "Session"
 "Session" -- "Maintain Tour scheme"
 "Maintain Tour scheme" -- "Reservation"
 "Maintain Tour scheme" -- "Administrator"
}

package "Access Database" {
 class "Database"
}

"User Interface" -- "Business View"
"Business View" -- "Access Database"
@enduml
```

DIAGRAM:

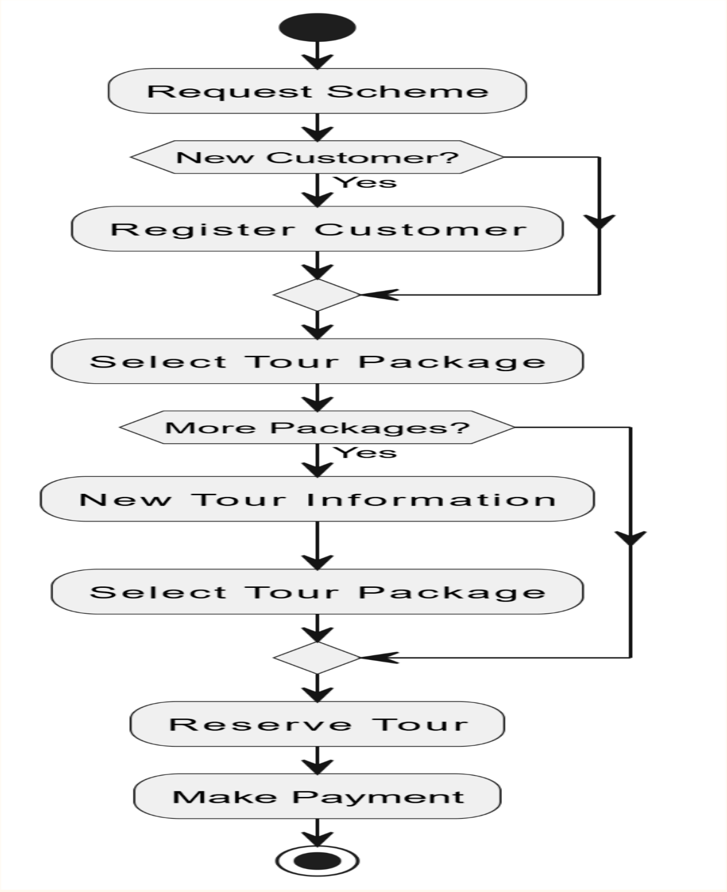


5) STATE DIAGRAM:

CODE:

```
@startuml
start
:Request Scheme;
if (New Customer?) then (Yes)
  :Register Customer;
endif
:Select Tour Package;
if (More Packages?) then (Yes)
  :New Tour Information;
  :Select Tour Package;
endif
:Reserve Tour;
:Make Payment;
stop
@enduml
```

DIAGRAM:



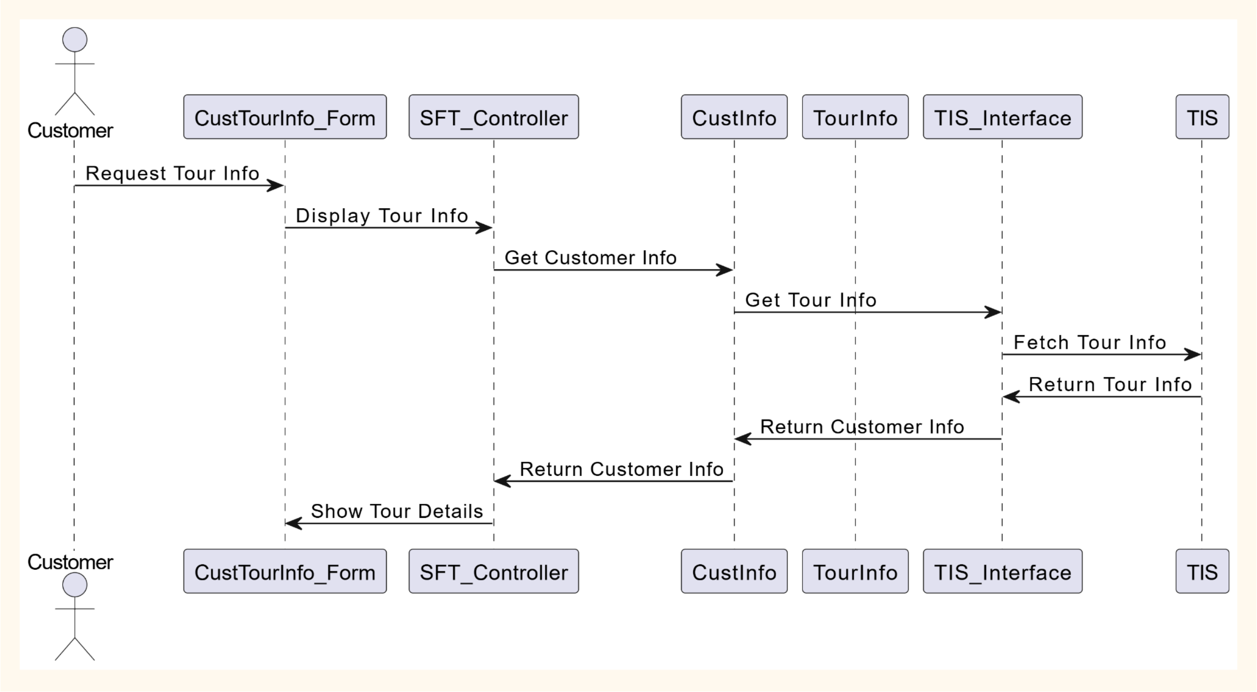
6) SEQUENCE DIAGRAM:

CODE:

```
@startuml
actor Customer
participant "CustTourInfo_Form"
participant "SFT_Controller"
participant "CustInfo"
participant "TourInfo"
participant "TIS_Interface"
participant TIS

Customer -> "CustTourInfo_Form": Request Tour Info
"CustTourInfo_Form" -> "SFT_Controller": Display Tour Info
"SFT_Controller" -> "CustInfo": Get Customer Info
"CustInfo" -> "TIS_Interface": Get Tour Info
"TIS_Interface" -> TIS: Fetch Tour Info
```

```
TIS -> "TIS_Interface": Return Tour Info
"TIS_Interface" -> "CustInfo": Return Customer Info
"CustInfo" -> "SFT_Controller": Return Customer Info
"SFT_Controller" -> "CustTourInfo_Form": Show Tour Details
@enduml
```

DIAGRAM:



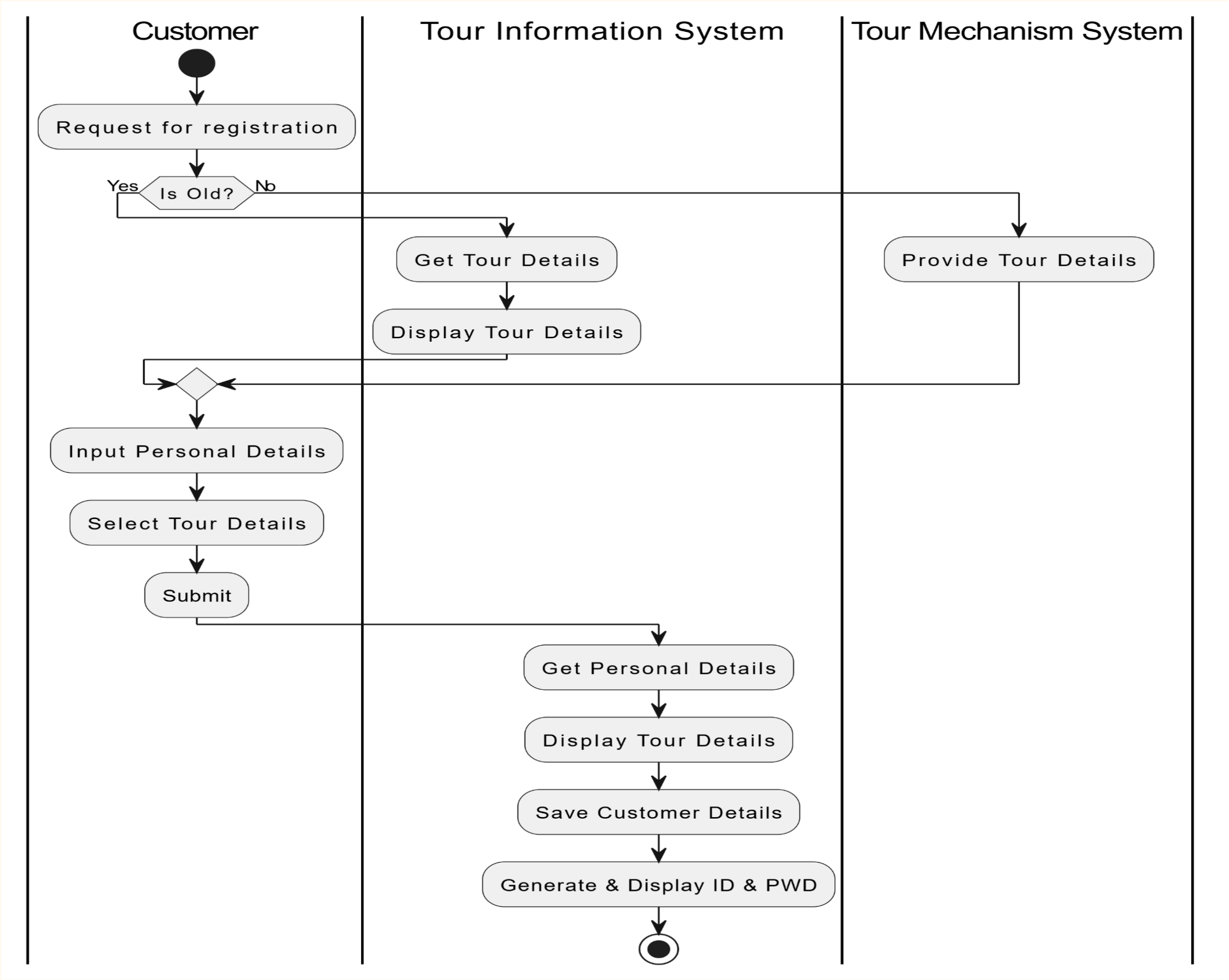
7) ACTIVITY DIAGRAM:

CODE:

```
@startuml
|Customer|
start
:Request for registration;
if (Is Old?) then (Yes)
  |Tour Information System|
  :Get Tour Details;
  :Display Tour Details;
else (No)
  |Tour Mechanism System|
  :Provide Tour Details;
endif
```

```
|Customer|
:Input Personal Details;
:Select Tour Details;
:Submit;

|Tour Information System|
:Get Personal Details;
:Display Tour Details;
:Save Customer Details;
:Generate & Display ID & PWD;

stop
@enduml
```

DIAGRAM:


Customer
Tour Information System
Tour Mechanism System
Request for registration
Yes
Is Old?
No
Get Tour Details
Provide Tour Details
Display Tour Details
Input Personal Details
Select Tour Details
Submit
Get Personal Details
Display Tour Details
Save Customer Details
Generate & Display ID & PWD