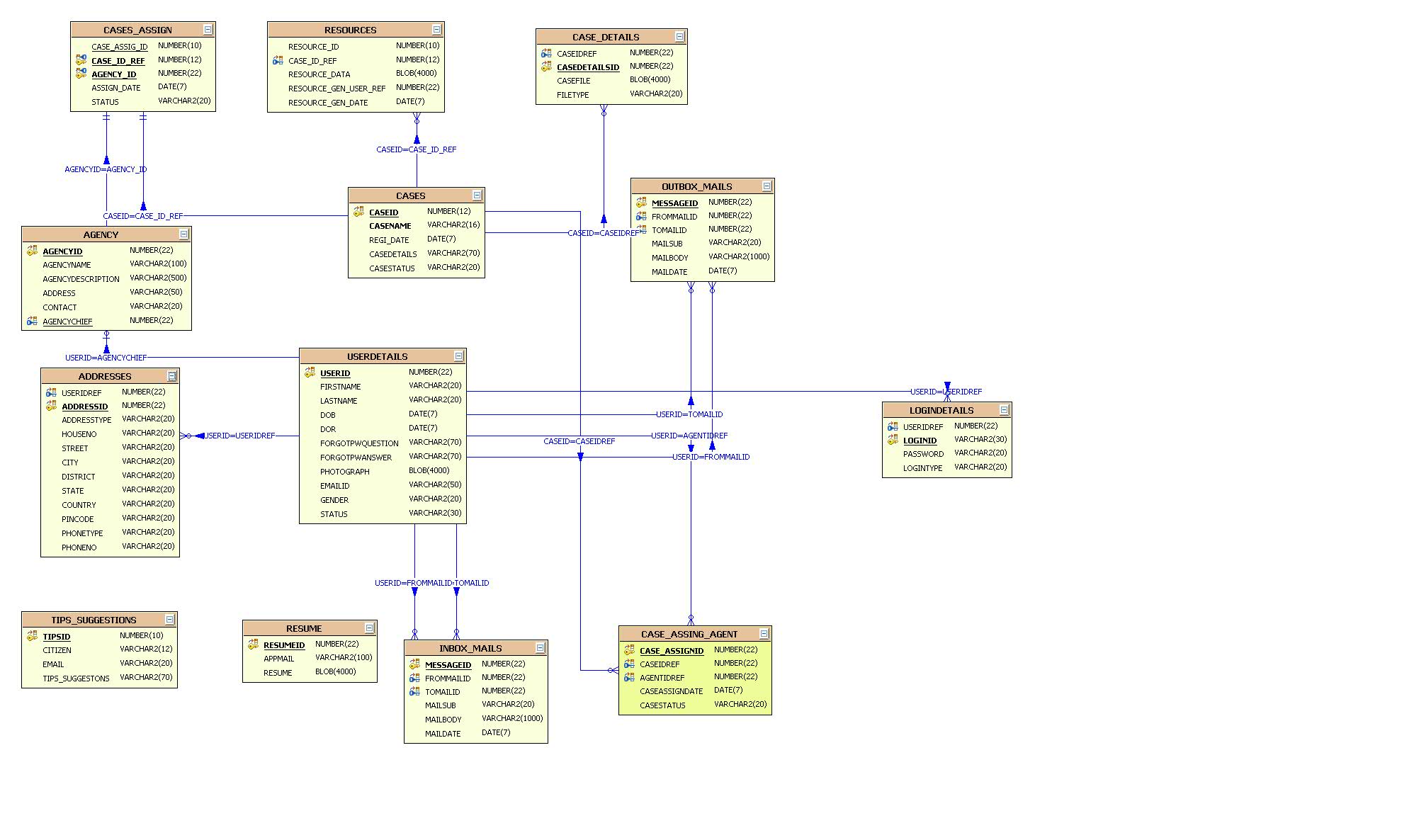
**SYSTEM DESIGN**

**E-R DIAGRAM**

**E - R Diagrams**

****

**DATA FLOW DIAGRAM**

##### DATA FLOW DIAGRAMS:

A graphical tool used to describe and analyze the moment of data through a system manual or automated including the process, stores of data, and delays in the system. Data Flow Diagrams are the central tool and the basis from which other components are developed. The transformation of data from input to output, through processes, may be described logically and independently of the physical components associated with the system. The DFD is also know as a data flow graph or a bubble chart.

DFDs are the model of the proposed system. They clearly should show the requirements on which the new system should be built. Later during design activity this is taken as the basis for drawing the system’s structure charts. The Basic Notation used to create a DFD’s are as follows:

**1. Dataflow:** Data move in a specific direction from an origin to a destination.

**2. Process:** People, procedures, or devices that use or produce (Transform) Data. The physical component is not identified.

**3. Source:** External sources or destination of data, which may be People, programs, organizations or other entities.

**4. Data Store:** Here data are stored or referenced by a process in the System.

**Context Level Data Flow Diagram**



**Level1 Data Flow Diagram For Minister:**



**Level1 Data Flow Diagram for Agent Chief:**



**Level1 Data Flow Diagram for Agent:**



**Authentication Data Flow Diagram:**



**Ministry:**

**Data Flow Diagram**



**Agent Chief Data Flow Diagram Diagram**



**Agent Data Level Diagram**



**UML DIAGRAMS**

**UNIFIED MODELING LANGUAGE DIAGRAMS**

The unified modeling language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic semantic and pragmatic rules.

A UML system is represented using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagram, which is as follows.

**USER MODEL VIEW**

This view represents the system from the users perspective.

The analysis representation describes a usage scenario from the end-users perspective.

**STRUCTURAL MODEL VIEW**

In this model the data and functionality are arrived from inside the system.

This model view models the static structures.

**BEHAVIORAL MODEL VIEW**

It represents the dynamic of behavioral as parts of the system, depicting the interactions of collection between various structural elements described in the user model and structural model view.

**IMPLEMENTATION MODEL VIEW**

In this the structural and behavioral as parts of the system are represented as they are to be built.

**ENVIRONMENTAL MODEL VIEW**

In this the structural and behavioral aspects of the environment in which the system is to be implemented are represented.

UML is specifically constructed through two different domains they are:

UML Analysis modeling, which focuses on the user model and structural model views of the system.

UML design modeling, which focuses on the behavioral modeling, implementation modeling and environmental model views.

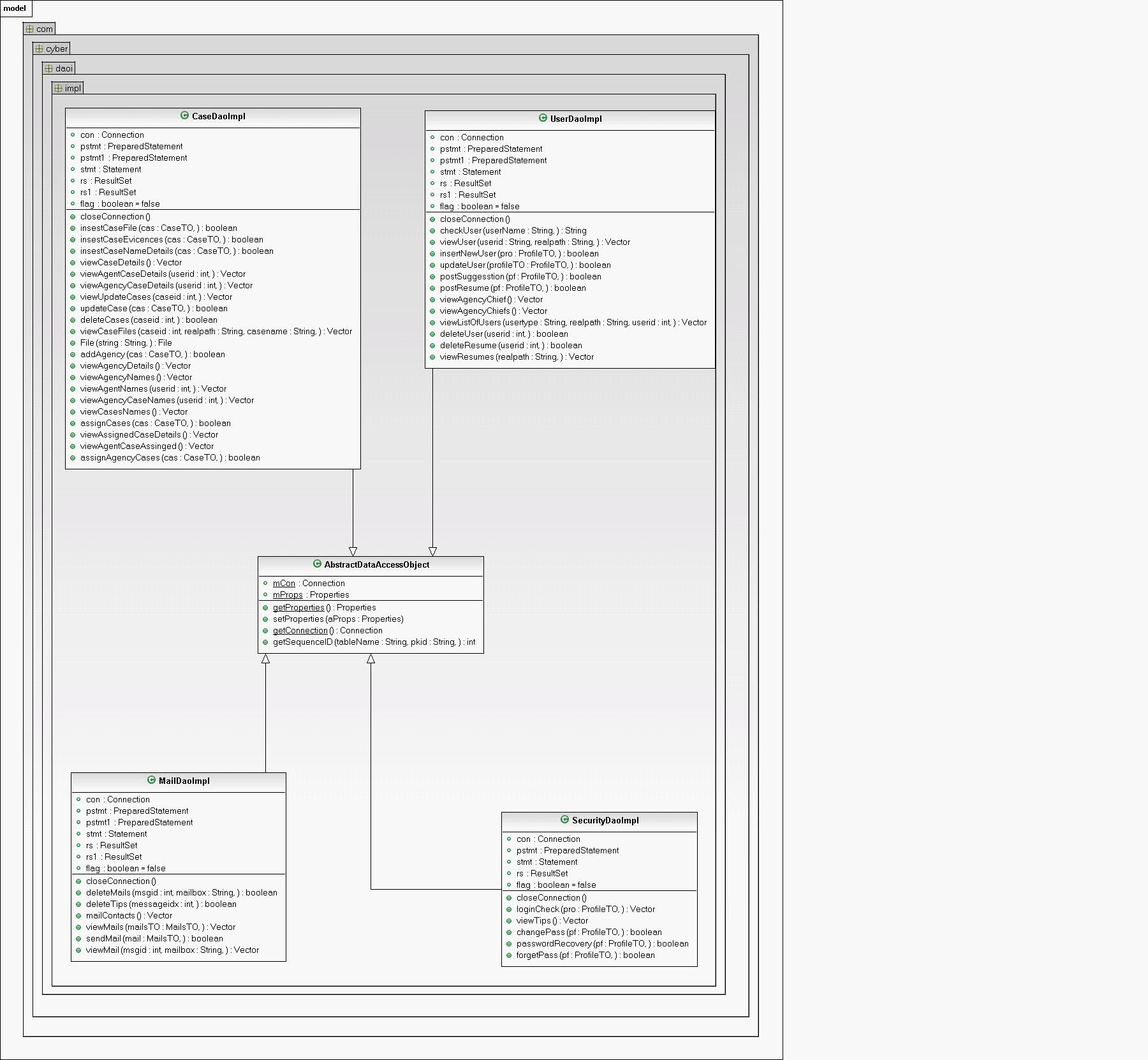
Use case Diagrams represent the functionality of the system from a user’s point of view. Use cases are used during requirements elicitation and analysis to represent the functionality of the system. Use cases focus on the behavior of the system from external point of view.

Actors are external entities that interact with the system. Examples of actors include users like administrator, bank customer …etc., or another system like central database.

**Class Diagram**

**CLASS DIAGRAM**

Class diagrams describe the structure of the system in terms of classes and objects. The servlet api class diagram will be as follows.

****

**Use Case Diagrams**

**UML Diagrams**

**Unified Modeling Language**:

The Unified Modeling Language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic semantic and pragmatic rules.

A UML system is represented using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagram, which is as follows.

* + User Model View
    1. This view represents the system from the users perspective.
    2. The analysis representation describes a usage scenario from the end-users perspective.
  + Structural model view
    1. In this model the data and functionality are arrived from inside the system.
    2. This model view models the static structures.
* Behavioral Model View

It represents the dynamic of behavioral as parts of the system, depicting the interactions of collection between various structural elements described in the user model and structural model view.

* Implementation Model View

In this the structural and behavioral as parts of the system are represented as they are to be built.

* Environmental Model View

In this the structural and behavioral aspects of the environment in which the system is to be implemented are represented.

UML is specifically constructed through two different domains they are:

* UML Analysis modeling, this focuses on the user model and structural model views of the system.
* UML design modeling, which focuses on the behavioral modeling, implementation modeling and environmental model views.

Use case Diagrams represent the functionality of the system from a user’s point of view. Use cases are used during requirements elicitation and analysis to represent the functionality of the system. Use cases focus on the behavior of the system from external point of view.

Actors are external entities that interact with the system. Examples of actors include users like administrator, bank customer …etc., or another system like central database.

**Use Case Diagram for user login :**



1. **system Use Case Diagram**

System

**Agency‘s chief**

**Agent**

**Ministry of Defence**

**Citizen**

**Evidences Transferring in Secret Intelligence Agencies**

**Defence Ministry Use Case Diagram:**



1. **Agent Use Case Diagram :**



**Citizen Usecase Diagram :**

**citizen**

**view success stories**

**view job resources**

**tips & suggesition**

**View services**

**Sequence Diagrams**

**User-Level Sequence Diagrams**

**DefenceMinistry sequence Diagram:**

**AgencyChief sequence Diagram:**

****

**Agent sequence Diagram:**

****

**Login Sequence Diagram:**

****

1 : Execute()

**ChangePassword Sequence Diagram :**

ChangePasswordAction

Securitydelegate

Securityserviceimpl

Securitydaoimpl

dbutil

database

1 :changePass ()

2 : :changePass ()

3 : :changePass ()

4 : getConnection()

5 : getConnection()

6 : getConnection

7 : exexutequery()

8 : queryResult

9 : returnStatus()

10 : returnStatus

11 : Success/failure()

1 : Execute()

**ForgetPassword:**

ForgetPasswordAction

Securitydelegate

Securityserviceimpl

Securitydaoimpl

dbutil

database

1 :forgetPass ()

2 : :forgetPass ()

3 : : forgetPass ()

4 : getConnection()

5 : getConnection()

6 : getConnection

7 : exexutequery()

8 : queryResult

9 : returnStatus()

10 : returnStatus

11 : Success/failure()

1 : Execute()

Registration Secquence Diagram :

RegistrationAction

Userdelegate

Userserviceimpl

UserDaoImpl

dbutil

database

1 : insertNewUser ()

2 : : insertNewUser ()

3 : : insertNewUser ()

4 : getConnection()

5 : getConnection()

6 : getConnection

7 : exexutequery()

8 : queryResult

9 : returnStatus()

10 : returnStatus

11 : Success/failure()

1 : Execute()

Update UserProfile Sequence Diagram :

UpdateUserProfileAction

Userdelegate

Userserviceimpl

UserDaoImpl

dbutil

database

1 : updateUser ()

2 : : updateUser ()

3 : : updateUser ()

4 : getConnection()

5 : getConnection()

6 : getConnection

7 : exexutequery()

8 : queryResult

9 : returnStatus()

10 : returnStatus

11 : Success/failure()

1 : Execute()

view UserProfile Sequence Diagram :

ViewProfileAction

Userdelegate

Userserviceimpl

UserDaoImpl

dbutil

database

1 : viewUser ()

2 : : viewUser ()

3 : : viewUser ()

4 : getConnection()

5 : getConnection()

6 : getConnection

7 : exexutequery()

8 : queryResult

9 : returnStatus()

10 : returnStatus

11 : Success/failure()

1 : Execute()

viewMail Sequence Diagram :

ViewMailAction

Maildelegate

Mailserviceimpl

MailDaoImpl

dbutil

database

1 : viewMails ()

2 : : viewMails ()

3 : : viewMails ()

4 : getConnection()

5 : getConnection()

6 : getConnection

7 : exexutequery()

8 : queryResult

9 : returnStatus()

10 : returnStatus

11 : Success/failure()

1 : Execute()

sendMail Sequence Diagram :

SendMailAction

Maildelegate

Mailserviceimpl

MailDaoImpl

dbutil

database

1 : sendMail ()

2 : : sendMail ()

3 : : sendMail ()

4 : getConnection()

5 : getConnection()

6 : getConnection

7 : exexutequery()

8 : queryResult

9 : returnStatus()

10 : returnStatus

11 : Success/failure()

1 : Execute()

deleteMail Sequence Diagram :

DeleteMailAction

Maildelegate

Mailserviceimpl

MailDaoImpl

dbutil

database

1 : deleteMails ()

2 : : deleteMails ()

3 : : deleteMails ()

4 : getConnection()

5 : getConnection()

6 : getConnection

7 : exexutequery()

8 : queryResult

9 : returnStatus()

10 : returnStatus

11 : Success/failure()

1 : Execute()

AddNewCase Sequence Diagram :

AddNewCaseAction

Casedelegate

Caseserviceimpl

CaseDaoImpl

dbutil

database

1 : insertCaseNameDetails()

2 : : insertCaseNameDetails ()

3 : : insertCaseNameDetails ()

4 : getConnection()

5 : getConnection()

6 : getConnection

7 : exexutequery()

8 : queryResult

9 : returnStatus()

10 : returnStatus

11 : Success/failure()

1 : Execute()

AddAgency Sequence Diagram :

AddAgencyAction

Casedelegate

Caseserviceimpl

CaseDaoImpl

dbutil

database

1 : addAgency()

2 : : addAgency ()

3 : : addAgency ()

4 : getConnection()

5 : getConnection()

6 : getConnection

7 : exexutequery()

8 : queryResult

9 : returnStatus()

10 : returnStatus

11 : Success/failure()

1 : Execute()

ViewAgencyAction

Casedelegate

Caseserviceimpl

CaseDaoImpl

dbutil

database

1 : viewAgencies()

2 : : viewAgencies ()

3 : : viewAgencies ()

4 : getConnection()

5 : getConnection()

6 : getConnection

7 : exexutequery()

8 : queryResult

9 : returnStatus()

10 : returnStatus

11 : Success/failure()

1 : Execute()

**Agent Sequence Collabration Diagrams**

****

**Login Sequence Collabration Diagram :**



Changepassword Sequencediagram :

ChangepasswordAction

Securitydelegate

Securityserviceimpl

Securitydaoimpl

dbutil

database

1 : changePass()

2 :changePass()

3 : changePass()

4 : getConnection()

5 : getConnection()

6 : getConnection

7 : exexutequery()

8 : queryResult

9 : returnStatus()

10 : returnStatus

11 : Success/failure()

ForgetpasswordAction

Securitydelegate

Securityserviceimpl

Securitydaoimpl

dbutil

database

1 : forgetPass ()

2 :forgetPass()

3 : forgetPass ()

4 : getConnection()

5 : getConnection()

6 : getConnection

7 : exexutequery()

8 : queryResult

9 : returnStatus()

10 : returnStatus

11 : Success/failure()

Registration Sequence CollbrationDiagram:

RegistrationAction

Userdelegate

Userserviceimpl

Userdaoimpl

dbutil

database

1 : insertNewUser()

2 : insertNewUser()

3 : insertNewUser ()

4 : getConnection()

5 : getConnection()

6 : getConnection

7 : exexutequery()

8 : queryResult

9 : returnStatus()

10 : returnStatus

11 : Success/failure()

Update UserProfile Sequence Diagram :

UpdateUserProfileAction

Userdelegate

Userserviceimpl

Userdaoimpl

dbutil

database

1 : updateUser ()

2 : updateUser()

3 : updateUser ()

4 : getConnection()

5 : getConnection()

6 : getConnection

7 : exexutequery()

8 : queryResult

9 : returnStatus()

10 : returnStatus

11 : Success/failure()

View UserProfile Sequence Collabration Diagram :

ViewUserProfileAction

Userdelegate

Userserviceimpl

Userdaoimpl

dbutil

database

1 : viewUser ()

2 : viewUser ()

3 : viewUser ()

4 : getConnection()

5 : getConnection()

6 : getConnection

7 : exexutequery()

8 : queryResult

9 : returnStatus()

10 : returnStatus

11 : Success/failure()

SendMail Sequence Collabration Diagram

SendMailAction

Maildelegate

Mailserviceimpl

Maildaoimpl

dbutil

database

1 : sendMail()

2 : sendMail ()

3 : sendMail ()

4 : getConnection()

5 : getConnection()

6 : getConnection

7 : exexutequery()

8 : queryResult

9 : returnStatus()

10 : returnStatus

11 : Success/failure()

ViewMail Sequences Collabration Diagram:

ViewMailAction

Maildelegate

Mailserviceimpl

Maildaoimpl

dbutil

database

1 : viewMails()

2 : viewMails ()

3 : viewMails ()

4 : getConnection()

5 : getConnection()

6 : getConnection

7 : exexutequery()

8 : queryResult

9 : returnStatus()

10 : returnStatus

11 : Success/failure()

deleteMail Sequences Collabration Diagram:

DeleteMailAction

Maildelegate

Mailserviceimpl

Maildaoimpl

dbutil

database

1 : deleteMails()

2 : deleteMails ()

3 : deleteMails ()

4 : getConnection()

5 : getConnection()

6 : getConnection

7 : exexutequery()

8 : queryResult

9 : returnStatus()

10 : returnStatus

11 : Success/failure()

CASES Sequences Collabration Diagram:

AddNewCaseAction

Casedelegate

Caseserviceimpl

Casedaoimpl

dbutil

database

1 : insertNewCaseDetails()

2 : insertNewCaseDetails ()

3 : insertNewCaseDetails ()

4 : getConnection()

5 : getConnection()

6 : getConnection

7 : exexutequery()

8 : queryResult

9 : returnStatus()

10 : returnStatus

11 : Success/failure()

Agency Sequences Collabration Diagram:

AddAgencyAction

Casedelegate

Caseserviceimpl

Casedaoimpl

dbutil

database

1 : insertAgencyDetails()

2 : insertAgencyDetails ()

3 : insertAgencyDetails ()

4 : getConnection()

5 : getConnection()

6 : getConnection

7 : exexutequery()

8 : queryResult

9 : returnStatus()

10 : returnStatus

11 : Success/failure()

**ACTIVITY DIAGRAMS**

**ACTIVITY DIAGRAMS**

**Activity Diagram for Secret Agency’s Chief:**



**Activity Diagram for Agent:**



**Activity Diagram for Defense Ministry:**



**Activity Diagram for Citizen:**

citizen

successfulstories

tips & suggestions

Job information

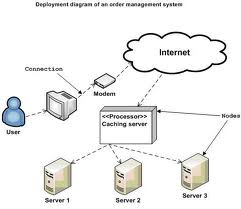
viewStries

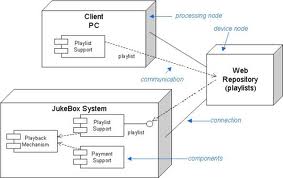
Activity3

tips & suuggestions

**Component Diagram**

**Component Diagram :**





**Deployment Diagram**

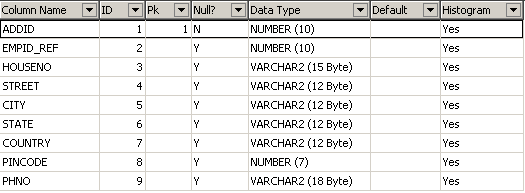
**Deployment Diagram:**

****

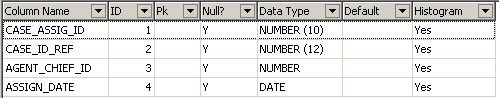
**Data Dictionary**

**Data Dictinory**

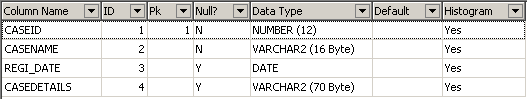
Address



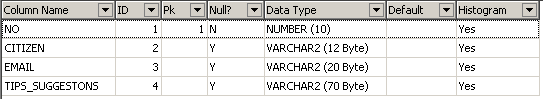
CASES\_ASSIGN\_CHIEF



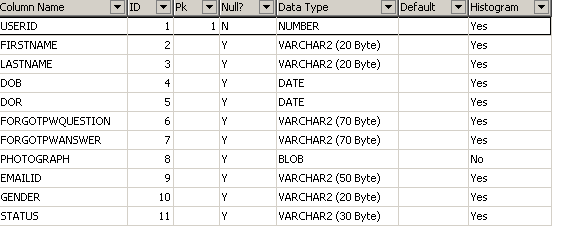
CASSES



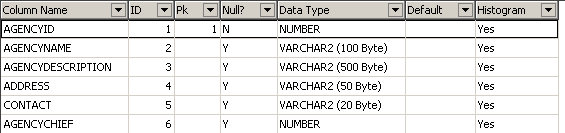
TIPS\_SUGGESTIONS



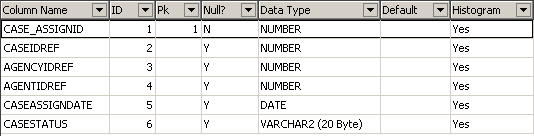
**Userdetails :**



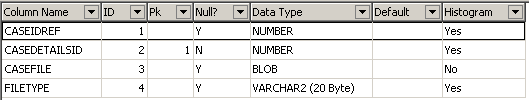
Agency:



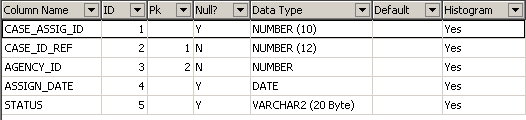
CASES\_ASSIGN\_AGENT



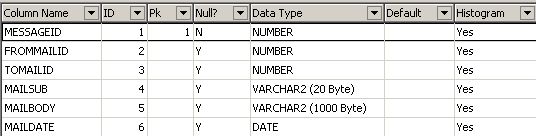
Case details:



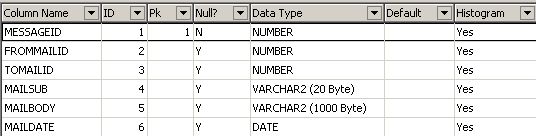
Cases\_ASSIGN



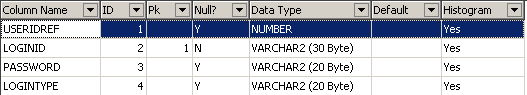
inboxtable



outbox



logindetails



**TECHNOLOGY DESCRIPTION**