##### 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 Administrator:**



**Level1 Data Flow Diagram for User:**



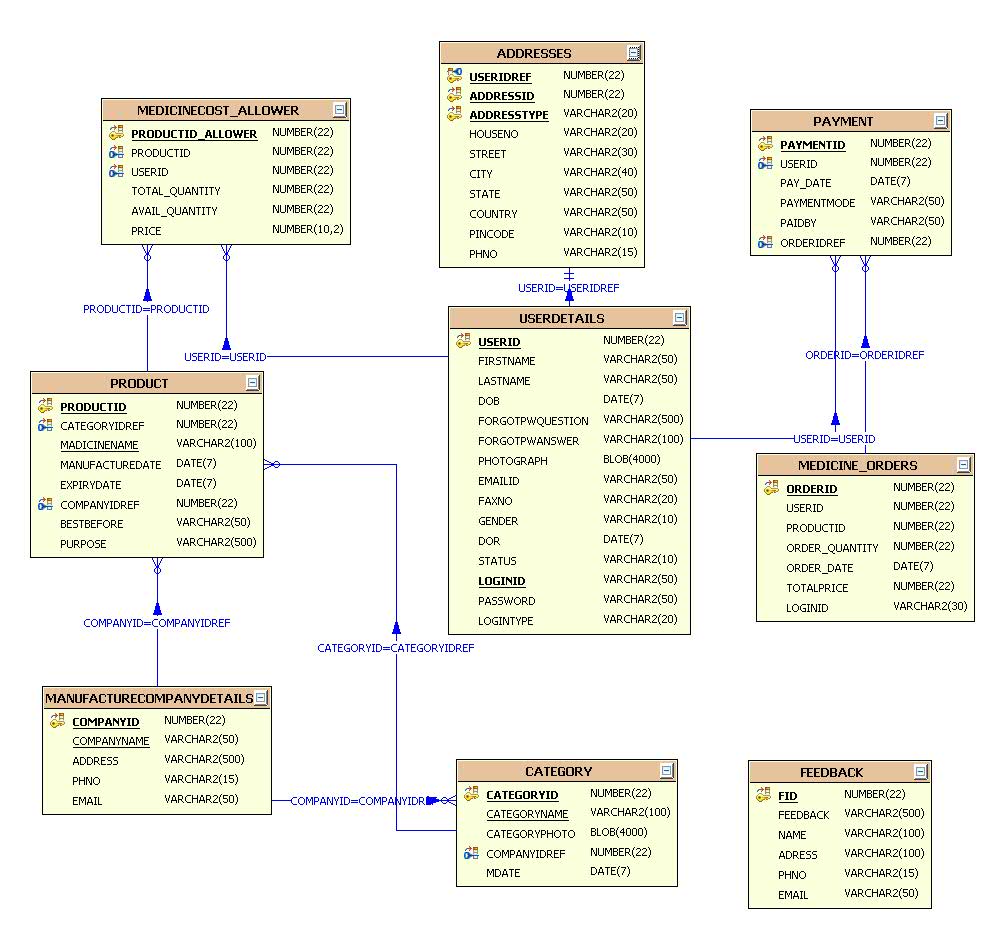
**Level1 Data Flow Diagram for Employee:**



**Authentication Data Flow Diagram:**



**E-R 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.

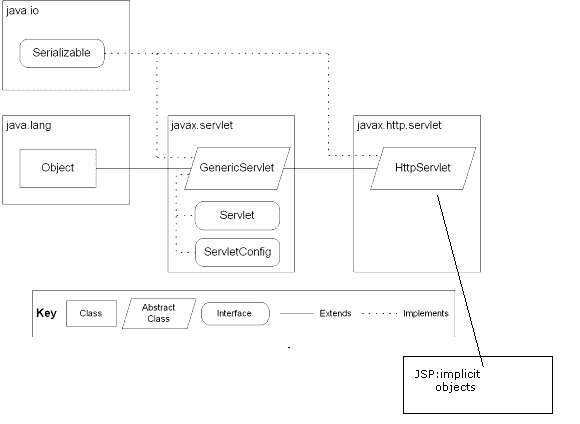
Use case Diagrams represent the functionality of the system from a user’s point of view. Use cases are used during requirements elicitation and analysis to represent the functionality of the system. Use cases focus on the behavior of the system from external point of view.

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**

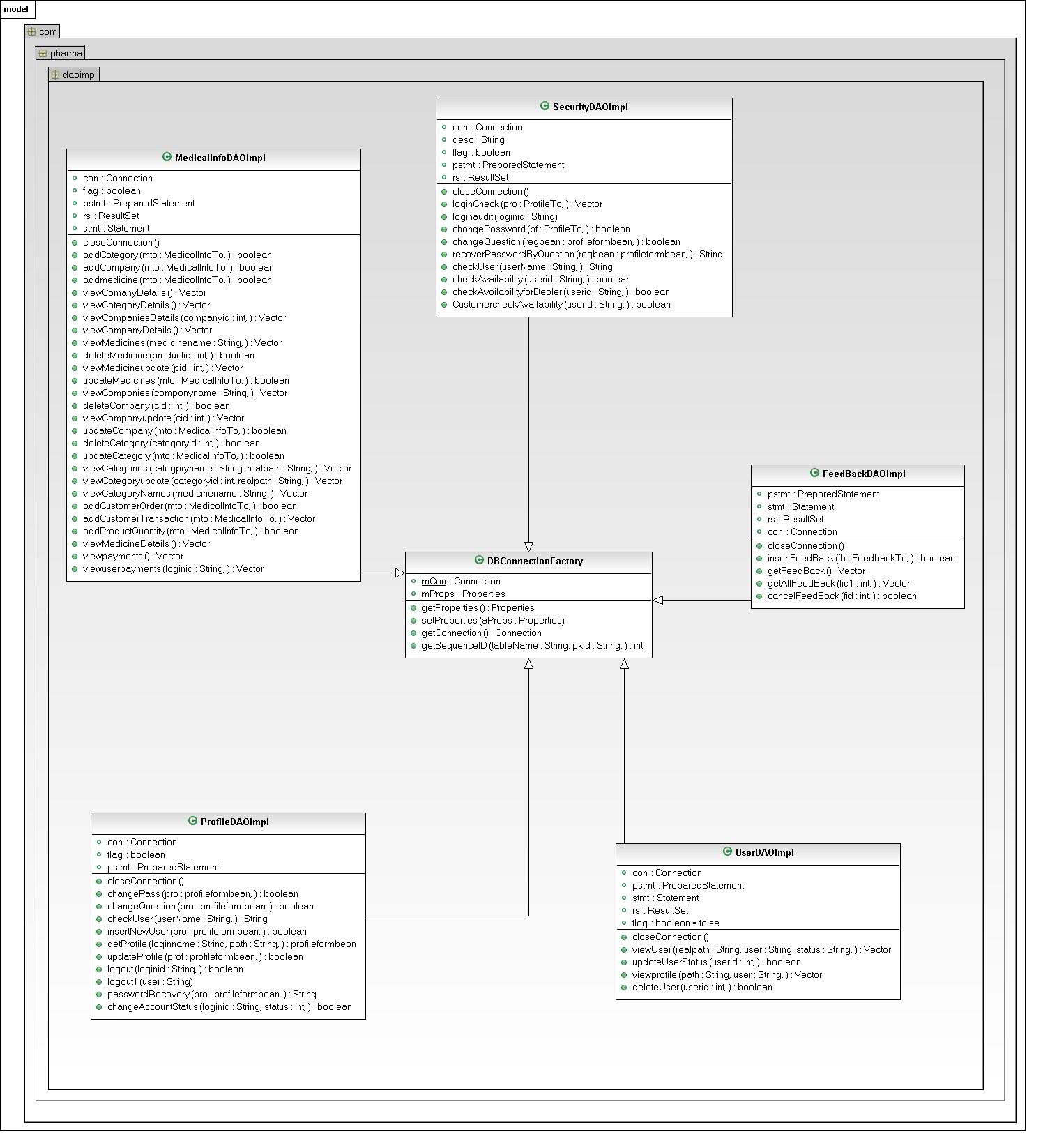
**CLASS DIAGRAM**

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



**Class Collaboration Diagrams**

**Class Collaboration Diagram**



**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.

1. **system Use Case Diagram**

System

**Admin**

**Customer**

**Employee**

Easy and Safe Drug Monitoring System for Patients

Admin Use Case

****

Employee Use Case



Customer Use Case



**Sequence Diagrams**

RegisterAction

ProfileMgrDelegate

ProfileServiceImpl

ProfileDAOImpl

1 : Execute()

2 : insertNewUser()

3 : insertNewUser()

4 : insertNewUser()

5 : flag

6 : flag

7 : flag

Login Check Sequence Diagram

LoginAction

SecurityMgrDelegate

SecurityServiceImpl

SecurityDAOImpl

1 : Execute()

2 : LoginCheck()

3 : LoginCheck()

4 : LoginCheck()

5 : flag

6 : flag

7 : flag

Check User Sequence Diagram

CheckUserAction

SecurityMgrDelegate

SecurityServiceImpl

SecurityDAOImpl

1 : Execute()

2 : CheckUser()

3 : CheckUser()

4 : CheckUser()

5 : flag

6 : flag

7 : flag

Change Password Sequence Diagram

ChangePasswordAction

SecurityMgrDelegate

SecurityServiceImpl

SecurityDAOImpl

Execute()

2 : ChangePassword()

3 : ChangePassword()

4 : ChangePassword()

5 : flag

6 : flag

7 : flag

View Users Sequence Diagram

ViewUserAction

ProfileMgrDelegate

UserServiceImpl

UserDAOImpl

1 : Execute()

2 : ViewUser()

3 : ViewUser()

4 : ViewUser()

5 : vector

6 : vector

7 : vector

Sequence for View Company Details



Add medicine Sequence



View medicine Sequence



Payment for medicines Sequence



Collaboration Diagram for Registration



Collaboration Diagram for Login



Collaboration Diagram for Check User



Collaboration Diagram for View User



Collaboration Diagram for Change Password



Collaboration for View companies



Collaboration for Add medicines



Collaboration for View medicines



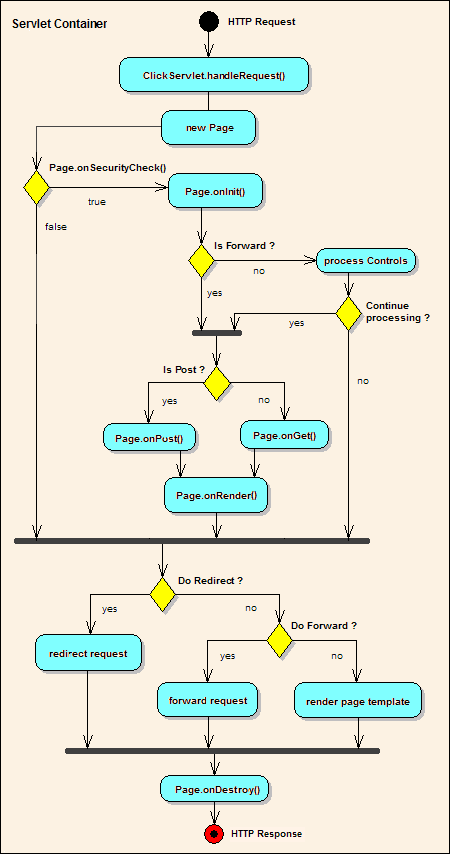
Collaboration Payment for medicines



**ACTIVITY DIAGRAMS**

**ACTIVITY DIAGRAMS**

1. **Servlet Container**

****

Admin Activity



Employee Activity



Customer Activity



**Component Diagram**

**Component Diagram :**

****

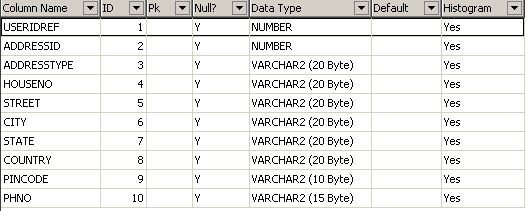
**Deployment Diagram**

**Deployment Diagram:**

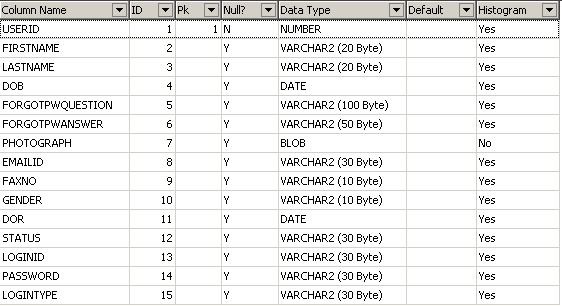
****

**Data Dictionary**

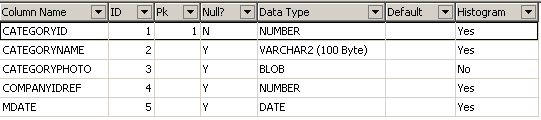
**ADDRESSES**



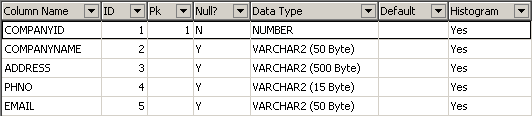
**USERDETAILS**

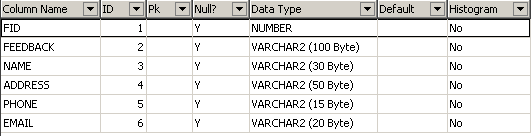
****

CATEGORY

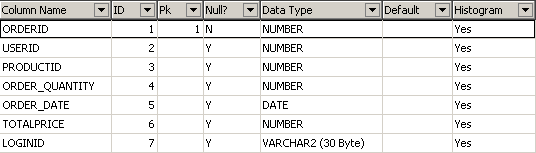


MANUFACTURECOMPANYDETAILS

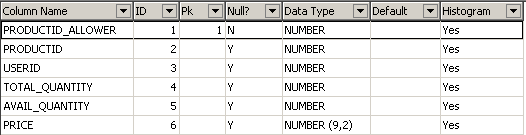


**FEEDBACK**

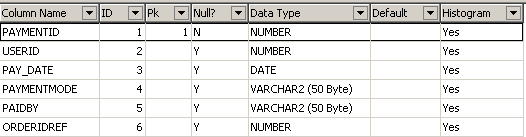
MEDICINE\_ORDERS



MEDICINECOST\_ALLOWER



PAYMENT



PRODUCT

