Technical Report

on

Online Medical Center Management System (SUST)

version 1.0

Course No. : CSE-300

Submitted

By

Md Mokarrom Hossain

Reg. No. 2007331039

June 2011

PURPOSE

The purpose of this document is to gather requirements, outlines the technical design and provides a comprehensive architectural overview for **Online Medical Center Management System version 1.0** (SUST medical center automation). The objective of this document is to give detail understand of the system which can be understand by managers and programmers.

PROBLEM STATEMENT

Currently, Our Medical Centre uses manual (primitive) Management System for maintaining the patient demography and distributing medicine to the patient. In existing system, doctors and other employees have to spend a lot of time to provide services to the patient because a lot of paper is used to record information. Here, prescribing patient, delivering medicine, maintaining medicine stock, retrieving records etc. everything is tremendously manual and paper dependent. So an automated online management system needs to be developed.

PROJECT OVERVIEW

The proposed system will provide a graphical user interface to maintain the total system, including prescribing patient, delivering medicine, maintaining medicine stock etc. Moreover, the new system will be accessible from terminals within the Medical Center and also through the internet from computers outside the Medical Center. Besides, the patients (both student and staff) can view their prescription through internet from anywhere.

USER CLASSIFICATION AND CHARACTERISTICS

There are two generic type of user in the system.

1) **General user (Patient):** There are two type of general user.

    a) **Staff:** All valid employees of the university who will familiar in web browsing.
    b) **Student:** All current students of the university who will familiar in web browsing.

2) **Administrative user:** There are four type of administrative user who will interact to the system both using Medical Center terminal and the Internet.

a) **Level 1: Doctors** (Medical Officer/Senior Medical Officer/Junior Medical Officer): All doctors who have a little bit knowledge in web browsing.
b) **Level 2 : Senior Pharmacist/Store Officer**
An employee who have a knowledge on web browsing and medicine. Senior pharmacist have little bit technical knowledge on about the system.
c) **Level 3: Medicine Distributor**.
An employee who have a knowledge on web browsing. Also have some knowledge about the medicine.

REQUIREMENT ANALYSIS

Software Requirement Specification (SRS):

1. **Stock Management** (for medicine)**:**
    a. Due to the maintenance simplicity and management security stock of medicine is divided in two sub category ***Central Stock*** and ***Sub Stock.*** All records of stock medicines are maintained through two subcategory central-store and sub-store.
    b. At first, newly bought medicine are added to the ***Central Stock*** and then transfer these into ***Sub Stock*** as needed.

2. **Entry New Purchased Medicine:**
    a. Pharmacist entry the newly purchased medicine at ***Central Stock Ledger*** providing Company Name, Purchased Date, Commission, Medicine Type, Medicine Name, Medicine Quantity, Per piece price, Manufacturing Date, Expiring Date.
    b. Pharmacist also entry the location of medicine at central stock.

3. **Stock Transfer :**
    a) Pharmacist can transfer medicine from ***Central Stock*** to ***Sub Stock*** as per vacancy of medicine at ***Sub Stock*** with the permission of **Chief Medical Officer.**
        i. Pharmacist can transfer a single medicine less or equal to the available quantity at Central ***Stock.***

ii. S/he can transfer multiple medicines simultaneously less or equal to the available quantity at Central ***Stock***.

b) Pharmacist can also update medicine location at sub stock.

4. <u>Stock View and Monitor</u> :

a. Pharmacist can view both ***Central Stock*** and ***Sub Stock*** and also monitor the flow of medicine.

b. Medicine distributor can only view ***Sub Stock.***

5. <u>Patient Detection and Authentication</u> :

a. **Students:** Doctors authenticate and detect students as legal patient by Digital Image, Registration Number, Department Name, Semester, Address, and Age.

b. **Staff:** Doctors authenticate and detect employee as legal patient by Digital Image, Employee Code, Department Name, Address, Age, and Designation.

6. <u>Prescribe Patients</u> :

a. Doctors prescribe the patient by providing following information to system

i. Diagnosis Details :

- Chief Complains
- On Examination
- Investigation

ii. Medicine Details :

- Medicine Type
- Medicine Name
- Time interval
- Amount per interval
- Medication instruction
- Quantity
- Duration

iii. **General Advice:** General advice can be selected from the drop down list as per demand.

iv. **Re-consultation Date:** The doctor select the interval in day/month/year for re-consulting and system will automatically calculate the re-consultation date.

7. **<u>Distribute Medicine</u> :**

a. Medicine distributor deliver medicine (suggested by doctor) to the patient which is available to ***Sub Stock.***

b. If a particular medicine is not currently available at ***Sub Stock*** but available at ***central Stock*** then ---

i. S/he said to patient to re-collect the medicine after a certain period.

ii. S/he send a request to Pharmacist to transfer the particular medicine from ***Central Stock*** to ***Sub Stock*** as early as possible.

8. **<u>View Prescription</u> :**

a. Students:

i. All students can view their all prescription (from the first prescription to last prescription like an **archive**) by logging their personal account.

ii. If a particular medicine is not provided by medical center it is showed specially for purchasing from outside.

b. Employee :

i. All employees can view their all prescription (from the first prescription to last prescription like an **archive**) by logging their personal account.

ii. All employee can view their payment amount of medicine (which is taken from Medical Center) of current month as well as the previous all month (from **archive**).

iii. If a particular medicine is not provided by medical center it is showed specially for purchasing from outside.

c. Doctors: Doctor can view all prescription prescribed by him/her and also view any prescription by dint of searching feature.

9. **<u>Searching</u> :**

   a. **Medicine Location :**

      - The location of a particular medicine at ***Central Stock***.
      - The location of a particular medicine at ***Sub Stock.***

   b. **Medicine Information :**

      - Find the quantity, manufacturing date, expiring date of a particular medicine.
      - Find all medicine to a particular generic group.
      - Find all expired medicine with their location both in ***Central Stock*** to ***Sub Stock.***

   c. **Search Prescription:** In a Graphical User Interface a doctors can search any prescription by selecting some searching criteria. S/he can search by date, diagnosis, patient id, doctor's id.

10. **Bill Generation and Payment :**

   a. Pharmacist can generate a bill for newly purchased medicine (which already enrolled to the system) for individual pharmaceutical company (i.e. medicine supplier).

11. **Possible Medicine List** (for purchasing):

   a. A proposal or suggestion of new medicine purchase list (which should be purchase in upcoming month) with quantity will be generated based on the previous record of database.

   b. An estimated budget for new medicine list will also be generated based on the previous price rate from database.

12. **<u>Expired Medicine</u> :**

   All expired medicine are shown in a new tab so that operator can take necessary steps to process them. In both ***Central Stock*** and ***Sub Stock*** expired medicines are showed separately and these are not deliverable.

13. **Payment for Medicine** (only for employee) :
    - **a.** Employee have to pay for their received medicine at Medical Center. An equivalent amount money of their delivered medicine will be discarded form their monthly salary.
    - **b.** Employee can view his/her total amount of payment for medicine.

14. **Medicine Location :**
    a) Every medicine has a predefine location both in the ***Central Stock*** to ***Sub Stock*** (e.g. Napa: Room no-203, Almera no-101 , Shelf no-5).
    b) Operator can change the location of a particular medicine and also able to define a new location for a newly arrived medicine.
    c) Medicine distributor will able to see the location and also able to find the location of a particular medicine.

15. **Notification and Alert :**
    - **a.** When the quantity of a particular medicine fall below of certain amount both in ***Central Stock*** and ***Sub Stock,*** a notification will be generated automatically.
    - **b.** When the expiration date of a particular medicine is close, then send an alert to the pharmacist so that s/he can distribute this medicine.

16. **Report Generation:**
    a. Different type of report will be generated based on the criteria (e.g. flow of a particular medicine at a particular period). A third party software (e.g. **Crystal Reports**) may be used or integrated to the system.

Technical Analysis

In developing the Online *Medical Center Management* project we have to face a lot of technical problem. Such as, one of critical problem is storing and formatting the patient's diagnosis details. It is very tedious and complex job storing and retrieving the patient diagnosis details in relational database. To avoid such kind of hazard we use a modern technique **JSON** for processing patient diagnosis details.

JSON acronym for ***JavaScript Object Notation*** is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language. It is derived from the JavaScript scripting language for representing simple data structures and associative arrays, called objects. Despite its

relationship to JavaScript, it is language-independent, with parsers available for most scripting languages.

Here is a simple example of a diagnosis details of one prescription of a particular patient

```
Var  diagnosis details {
                    “Chief Complaints” :
                        {
                            “ Fever “  :  “5 Days ”  ,
                            “Throught  Pain” : “7 Days ”   ,
                            “ Headache ” : “ 3 Days ”
                        }  ,
                    “On Examinatio ” :
                        {
                            “ Temperature “   :   “ 103 ” ,
                            “ Abdonem ”  :  “ Soft + ”   ,
                            “ Blood Pressure  ” :  “ 180/90 ”
                        }   ,
                    “Investigation ” :
                            [“ ECG ” , “Lipidprofile ” , “ CMR “ ]
}
```

We use JSON parser for converting a JSON data into a string and vice versa.

- JSON.parse(strJSON) - converts a JSON string into a JavaScript object.
- JSON.stringify(objJSON) - converts a JavaScript object into a JSON string.

Another technical problem is dynamically update one drop down list based on the selection of another drop down list. We use Ajax request and response for solving this problem.

System Interfaces:

- Client on Internet: Web Browser, Operating System (any)
- Client on Intranet: Client Software, Web Browser, Operating System (any)
- Web Server: Apache Tomcat, Operating System (any)
- Data Base: Mysql Server.

User Interfaces:

User interfaces for all users are graphical user interfaces (GUI). These GUI will be both web based and also desktop base which is connect to the medical central terminal. The user interfaces for this system have to be simple and clear. Most importantly, the ages must be easy to read, easy to understand and accessible.

Communication Interfaces:

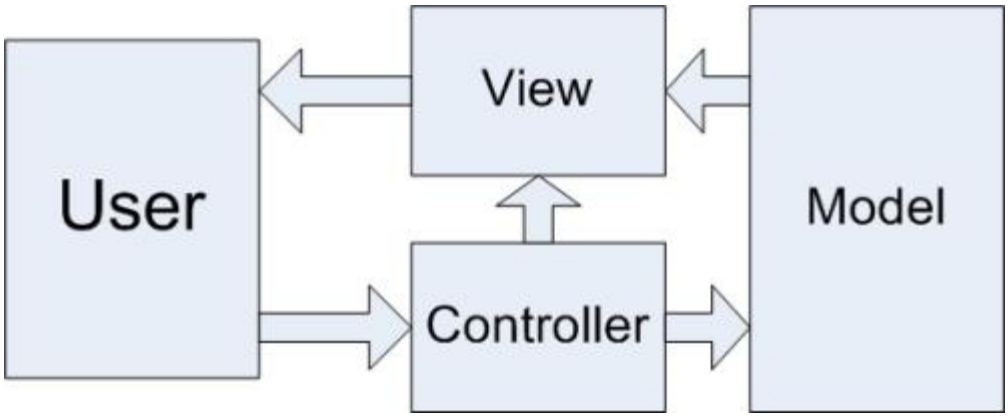
- Client on Internet will be using HTTP/HTTPS Protocol.
- Client on intranet will be using TCP/IP protocol.

ARCHITECTURE

Model-view-controller (MVC) is a software architecture concept considered as an architectural pattern in software engineering. The main aim of the MVC architecture is to separate the business logic and application data from the presentation data to the user. The pattern isolated domain logics from the user interfaces(GUI), resulting in an application where it is easier to modify either the visual appearance of the application or the underlying business rules without affecting the other. Decoupling the domain logic from the user interface permit independent development, testing and maintenance of each. It is usually called the separation of concerns.

Input → Processing → Output

Controller → Model → View



MVC architecture consists of the following three components.

Model

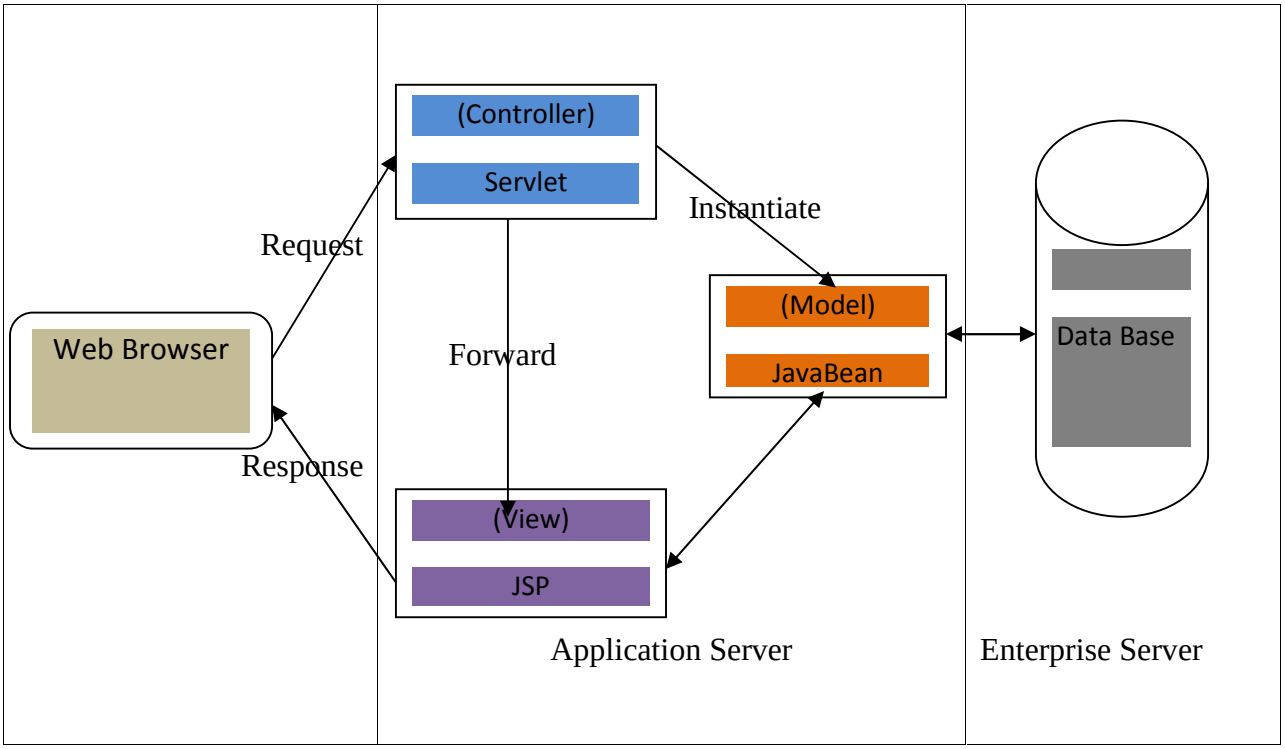
The model object knows about all the data that need to be displayed. It is the model who is aware about all the operations that can be applied to transform that object. It only represents the data of an application. The model represents enterprise data and the business rules that govern access to and updates of this data. Model is not aware about the presentation data and how that data will be displayed to the browser. Many applications use a persistent storage mechanism such as a database to store data. JavaBeans and any other similar type of technology may be used for implementing the Model.

View

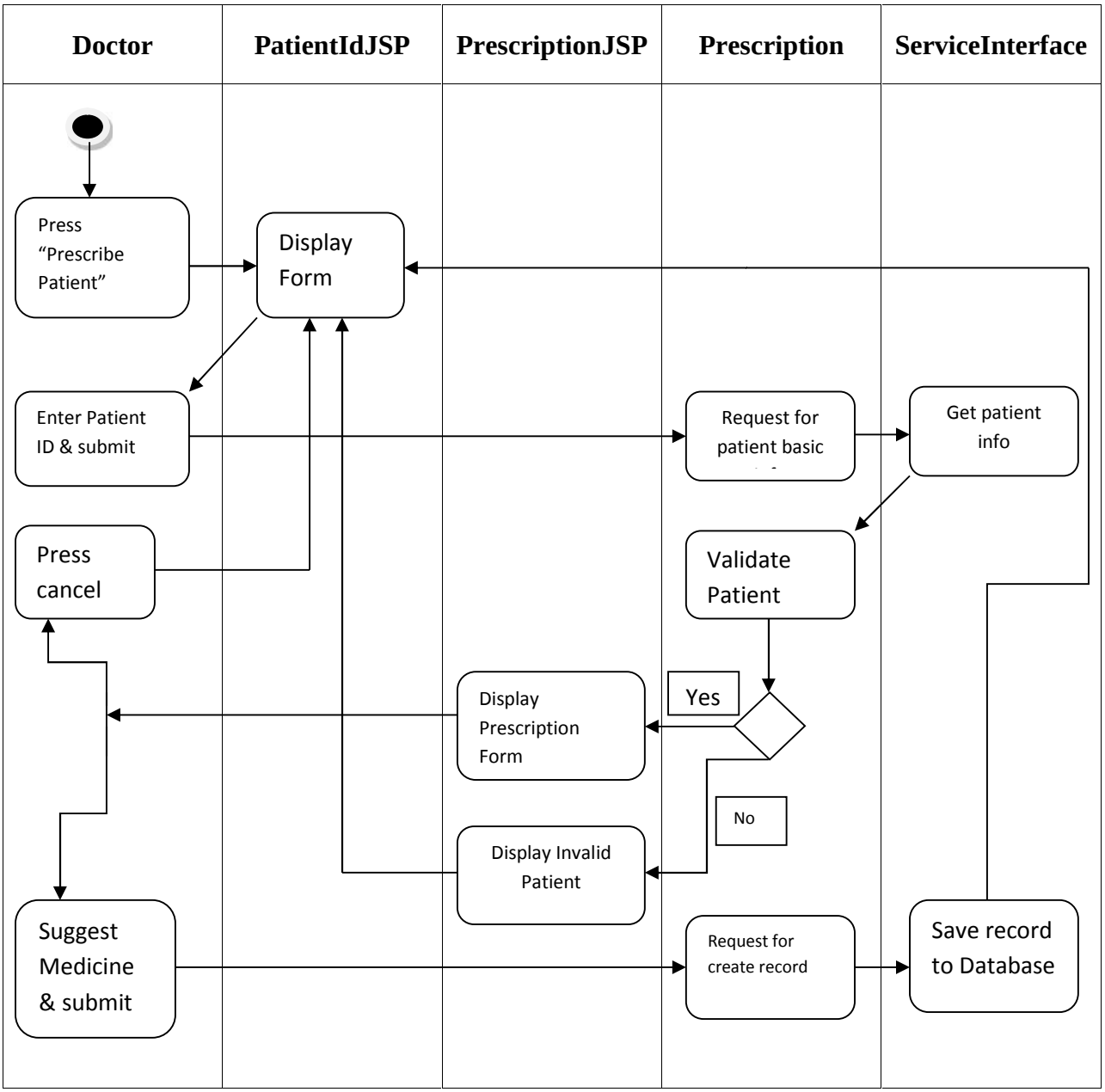
The view represents the presentation of the application. The view object refers to the model. It uses the query methods of the model to obtain the contents and renders it. The view is independent from the application logic. It remains same if there is any modification in the business logic. View is the interface the user sees and interacts with. There is no real processing happening in the view; it serves only as a way to output data and allow the user to act on that data. The GUI layer can use techniques like HTML, CSS, JSP, JHTML, DHTML, jQuery, Ajax etc.

Controller

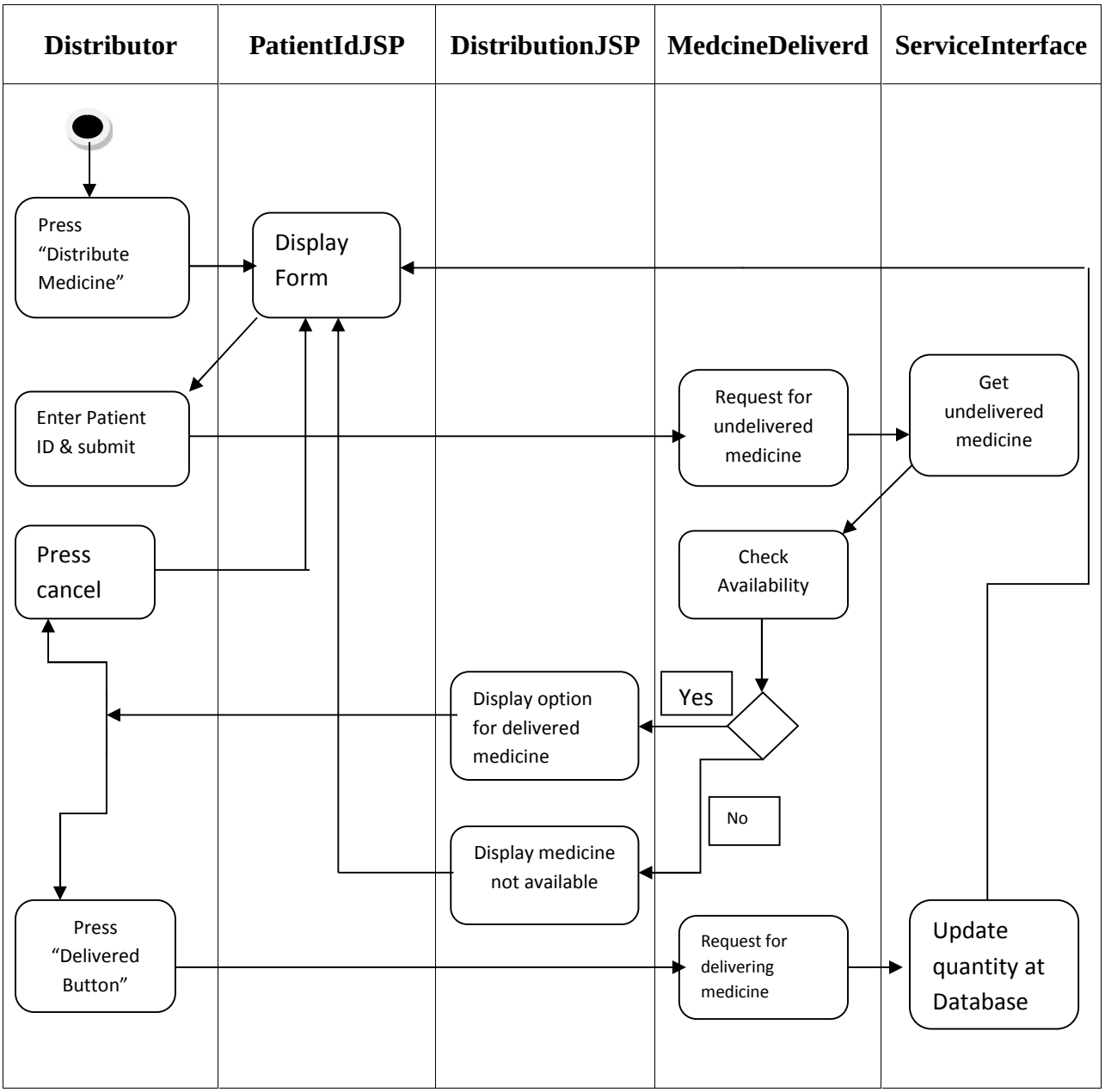
Whenever the user sends a request for something then it always go through the controller. The controller is responsible for intercepting the requests from view and passes it to the model for the appropriate action. After the action has been taken on the data, the controller is responsible for directing the appropriate view to the user. In GUIs, the views and the controllers often work very closely together. Actually Controller Processes and responds to events (typically user actions) and may indirectly invoke changes on the model. Servlet and any other similar type of technology may be used in the controller.


(Controller)
Servlet
Instantiate
Request
(Model)
Web Browser
Forward
Data Base
JavaBean
Response
(View)
JSP
Application Server
Enterprise Server

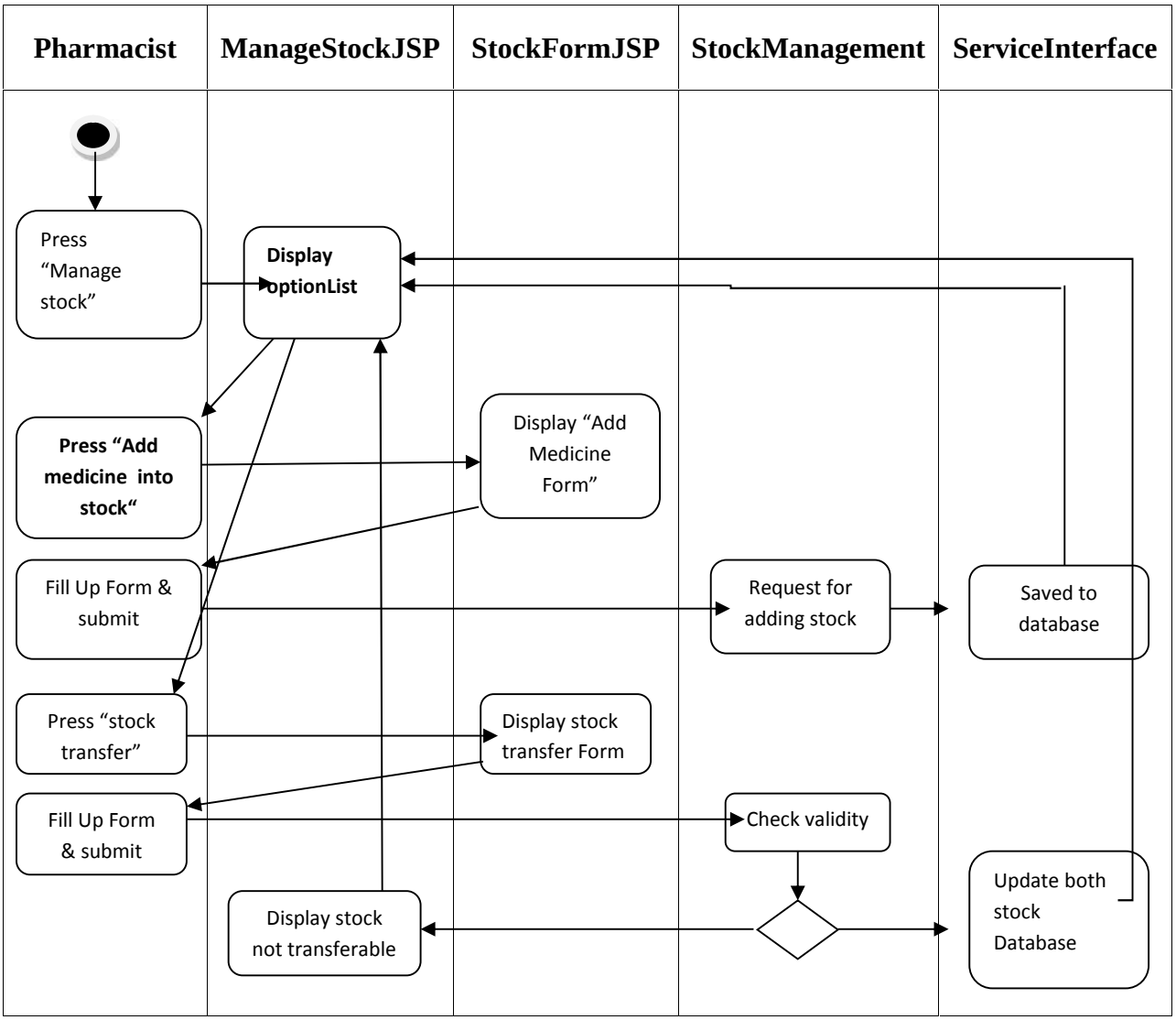
Activity diagram representing how doctor serve a patient.



Activity diagram representing how Medicine distributor distribute medicine to a patient.

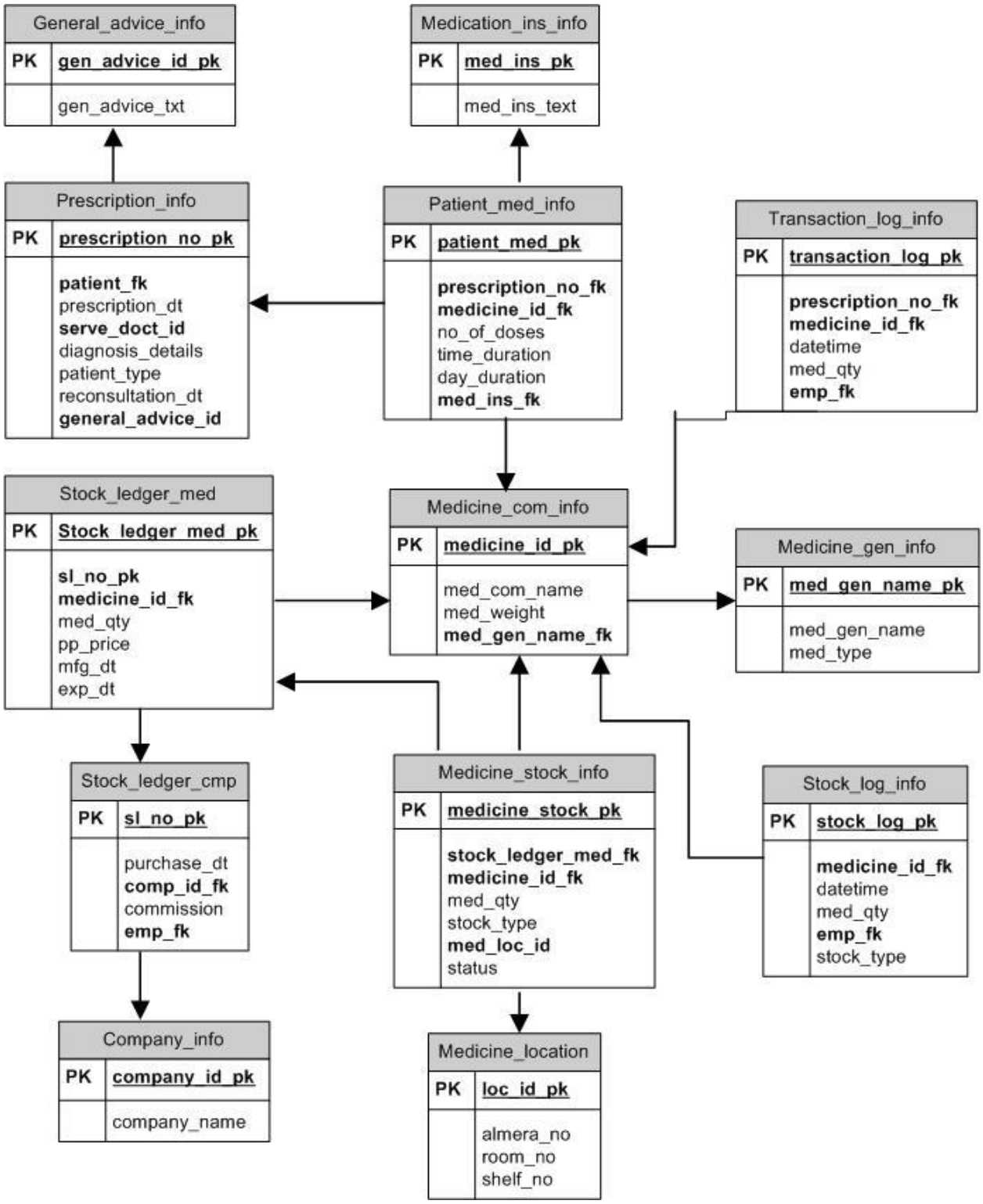


Activity diagram representing how **Pharmacist** add medicine into stock and transfer medicine from ***Central Stock*** to ***Sub Stock***.



DATABASE MODEL

Data Model is a method for describing data structures and a set of operations used to manipulate and validate that data. Data Model for the Online Medical Center Management System is as shown below (in the model primary key is indicated by **bold** & **underline**, foreign key is indicate by only **bold** and arrow indicate the relationship) -


General_advice_info
PK
gen_advice_id_pk
gen_advice_txt
Medication_ins_info
PK
med_ins_pk
med_ins_text
Prescription_info
PK
prescription_no_pk
patient_fk
prescription_dt
serve_doct_id
diagnosis_details
patient_type
reconsultation_dt
general_advice_id
Patient_med_info
PK
patient_med_pk
prescription_no_fk
medicine_id_fk
no_of_doses
time_duration
day_duration
med_ins_fk
Transaction_log_info
PK
transaction_log_pk
prescription_no_fk
medicine_id_fk
datetime
med_qty
emp_fk
Stock_ledger_med
PK
Stock_ledger_med_pk
sl_no_pk
medicine_id_fk
med_qty
pp_price
mfg_dt
exp_dt
Medicine_com_info
PK
medicine_id_pk
med_com_name
med_weight
med_gen_name_fk
Medicine_gen_info
PK
med_gen_name_pk
med_gen_name
med_type
Stock_ledger_cmp
PK
sl_no_pk
purchase_dt
comp_id_fk
commission
emp_fk
Medicine_stock_info
PK
medicine_stock_pk
stock_ledger_med_fk
medicine_id_fk
med_qty
stock_type
med_loc_id
status
Stock_log_info
PK
stock_log_pk
medicine_id_fk
datetime
med_qty
emp_fk
stock_type
Company_info
PK
company_id_pk
company_name
Medicine_location
PK
loc_id_pk
almera_no
room_no
shelf_no

CODING

The coding phase needs deep knowledge about Java programming and object oriented technique. This includes traditional Java standard and JSP programming. Programming technique using Struts 2 is also very important. For sake of better understanding on MVC element of system, the following discussion will be divided into subsection explaining about how to implement Model, View and Controller using Struts 2.

View Layer Implementation:
The classes in View layer has been implemented in JSP with combination of Struts 2 User Interface (UI) tag. This tag must be used so that classes in view layer can communicate with Struts 2 controller. Knowledge of other technology such as HTML and Cascade Style Sheets (CSS), JavaScript, Ajax is very helpful in developing View Layer.

Controller Layer Implementation:
The communication between JSP in View Layer and classes in Action packages of Model Layer is done using Struts 2 controller. Basically this controller is internal component of Struts 2 and there is no programming needed. However the communication need to be configured using XML file named as struts.xml.

Model Layer Implementation:
Model layer consist of Action, PersistenceService and Entity packages. As a name suggest, in struts 2 point of view, the classes in Action packages is the implementation of struts 2 action class framework. Classes in Action packages have been developed in traditional Java standard. Java class must implement Struts 2 framework in order to comply with the architecture.

TESTING AND DEBUGING:

Unit Testing: On receiving system design documents, the work is divided into different modules/units. The system will be developed first in small programs called units in the coding phase, which will be integrated in the next phase. Each unit is developed and tested for its functionality; this is referred to as Unit Testing. Unit testing mainly verifies if the modules/units meet their specifications. The requirement traceability will come in handy to verify if the earlier requirement has been fulfilled.

Unit testing is focused on verifying small portions of functionality. For example, an individual unit test case might focus on verifying that the correct data has been saved to the database when the Submit button on a particular page is clicked.

Integration and System Testing: As specified above, the system is first divided in units which are developed and tested for their functionalities. These units are integrated into a complete

system during Integration phase and tested to check if all modules/units coordinate between each other and the system as a whole behaves as per the specifications. After successfully testing the software, it is delivered to the customer.

Functionality Testing: Functionality testing is to validate behaviors, error handling, manipulations, services levels, order of functionality, links, content of web page & backend coverage's. Make sure all the fields that collect information from the user, can gracefully handle any value that is entered.

a. Create and run a unit test that tests the login by:
- starting the web application
- entering invalid data into the fields
- pressing the login button
- checking that the login page is shown again
- entering valid data into the fields
- pressing the login button again
- checking that now the index page is shown

b. Now that we can login into the web application, it is time that we explore it. The individual parts of the web application are reached by the menu on the left side. After performing a login, click the link in the menu on the left side for listing all words, and then assert that the corresponding page has been loaded.

c. Log in into the system as doctor and perform the following task
- Enter valid/invalid patient id and check it.
- Verify the patient basic info.
- Make sure that all data of prescription insert into database properly.
- Make sure search prescription data is correct.

d. Log in into the system as pharmacist and perform the following task
- Enter medicine into database manually and check it displayed properly.
- Transfer the medicine quantity grater then available quantity and check it.
- Add different quantity of medicine and validate it.

e. Check the browser back button, reload button and check by pressing twice all buttons in UI.

User Acceptance Testing: Test every user interface (UI) for validation & user friendliness individually. By performing user acceptance testing, you are making sure your web application fits the use for which it was intended. Simply stated, you are making sure your web application makes things easier for the user and not harder. One effective way to handle user acceptance testing is by setting up a beta test for your web application.

Browser compatibility testing: There are two main aspects of verifying the validity of our HTML. First, we want to make sure that our syntax is correct, such as verifying that all opening

and closing tags match, etc. Secondly, we want to verify how our pages look in different browsers, at different screen resolutions, and on different operating systems. Create a profile of our target audience and make some decisions on what browsers you will support, on which operating systems, and at what screen resolutions.

Load testing: We have to conduct a real-world load testing tips to avoid bottlenecks when our system goes live. The only variable our servers really care about in relation to load is the number of requests per second. The number of virtual users and their think times are combined to produce that load. The following formula calculates the load generated (requests/second) by real users accessing the site:

$$\frac{\text{Requests}}{\text{Second}} = \left(\frac{\text{Users}}{\text{Download Time+Think Time}}\right)\left(\frac{\text{Requests}}{\text{Page}}\right)$$

Stress Testing: Stress testing, which is a specialized form of performance testing, is similar to destructive testing in other fields of engineering. The goal of stress testing is to crash the application by increasing the processing load past performance degradation until the application begins to fail due to saturation of resources or the occurrence of errors. Stress testing helps to reveal subtle bugs that would otherwise go undetected until the application was deployed. Since such bugs are typically the result of design flaws, stress testing should begin early in the development phase on each area of the application.

Testing Security: With the large number of highly skilled hackers in the world, security should be a huge concern for anyone building a web application. We need to test how secure our web application is from both external and internal threats. We have to concern about several security threat like Password cracking, SQL Injection, Cross Site Scripting (XSS)**,** Cookie poisoning, Command injection **etc.**

After performing your initial security testing, make sure to also perform ongoing security audits to ensure your web application remains secure over time as people and technology change.