Software Design Specification

For

Hostel Asset Management

Prepared by

Jobin Joseph (U101116FCS053)
Mamidi Lokesh (U101116FCS063)
PVNSSK Chaitanya (U101116FCS085)
PSK Vamsi (U101116FCS088)
Pradeep Yadav (U101116FCS089)

Software Engineering
NIIT University

30th October 2018

1. Introduction

1.1 Purpose of this document

The main purpose of this document is to provide a software development team with overall guidance to the architecture of the software project. This document would help any individual software developer or a software development team to understand the architecture of the project and show the working of a individual software parts as well as the whole software.

1.2 Scope of the development project

For a University or campus scale of asset management, it is usually found to be uneasy to follow the flow of assets and their management.

With proper flow control, we are in a process to guarantee hour deep measure of information of a particular asset.

1.3 Definitions, acronyms, and abbreviations

IEEE: Institute of Electrical and Electronics Engineers

SDS: Software Design Specification

1.4 References

- The whole document is built by taking IEEE - SDS template (IEEE 1016)
  as a reference.

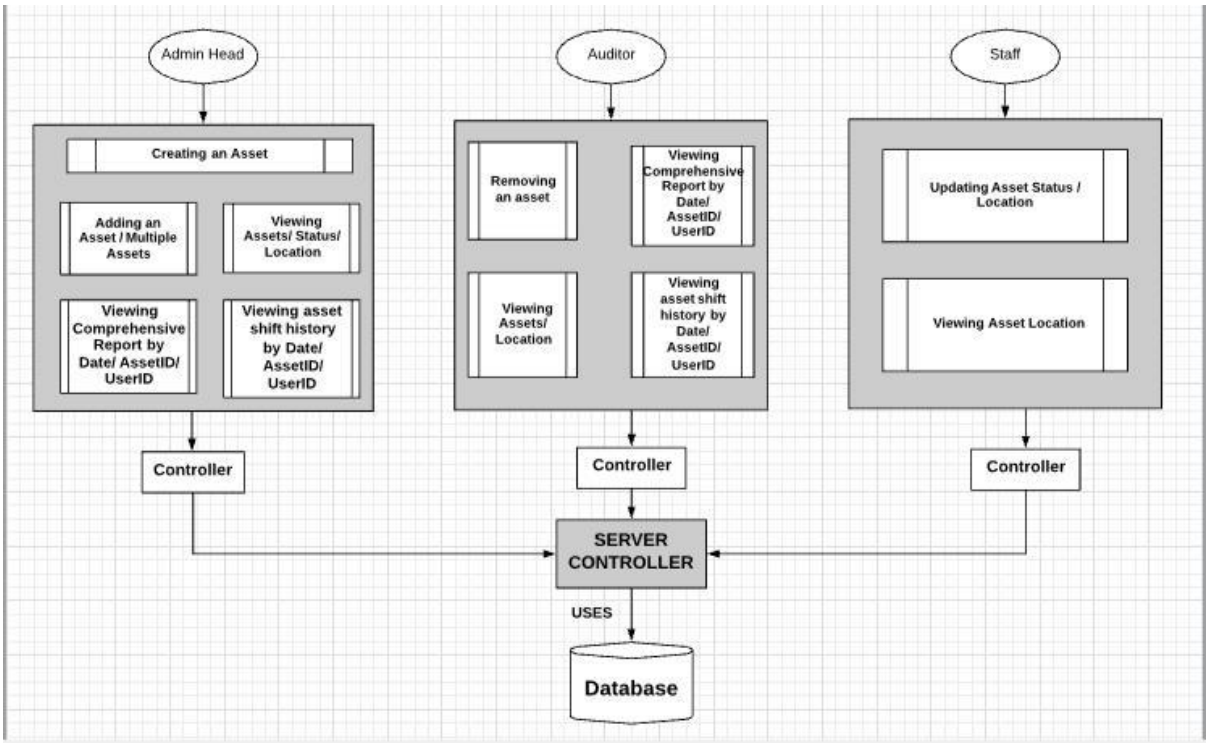
1.5 Overview of document

The SDS is divided into five sections with various sub-sections. The sections of the Software Design Document are:

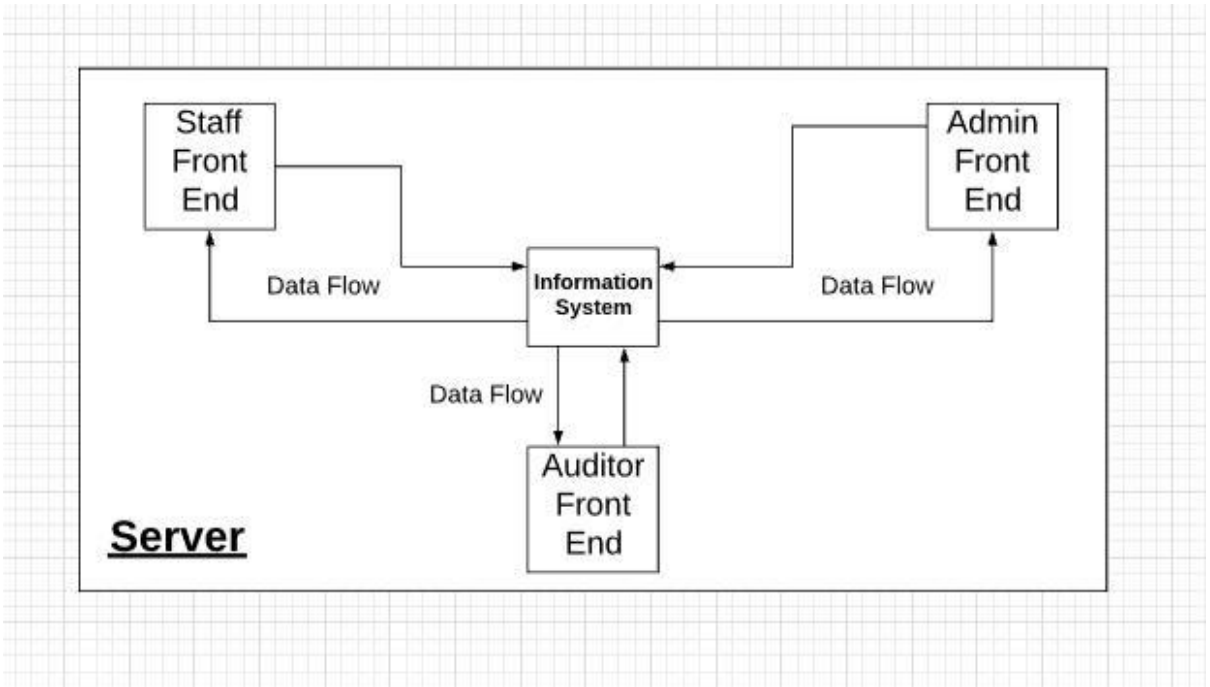
1. Introduction
2. Conceptual Architecture/ Architecture Diagram
3. Logical Architecture
4. Execution Architecture
5. Design Decisions and Trade-offs

2. Conceptual Architecture/Architecture Diagram

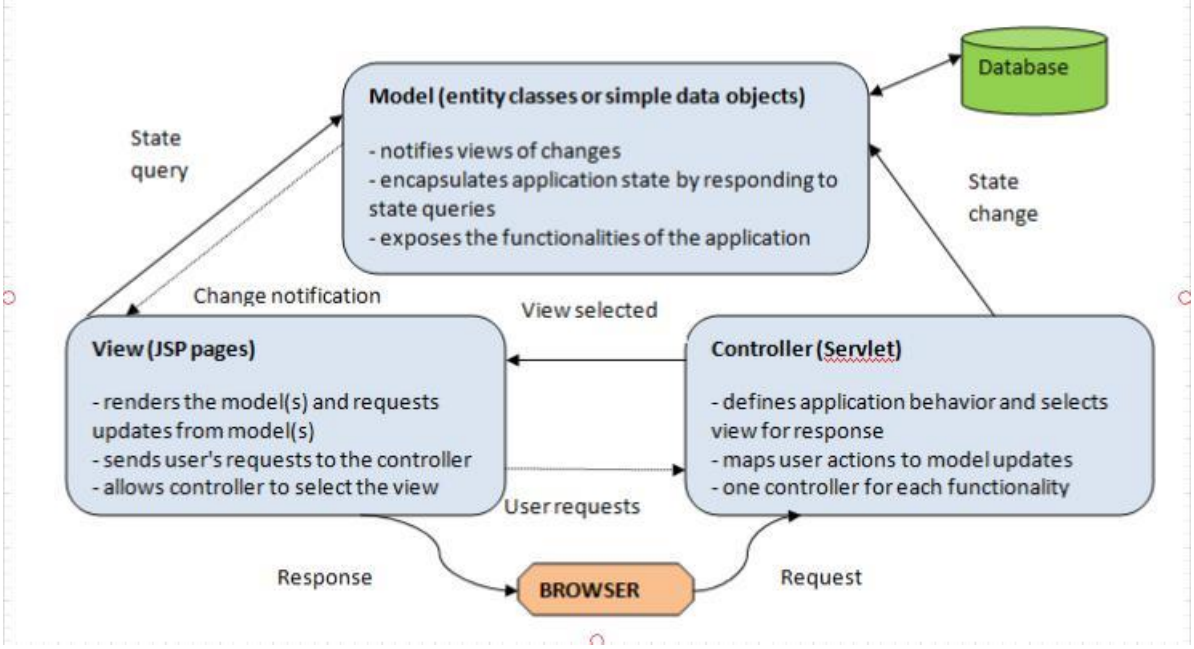
Architecture Diagram 1:



Architecture Diagram 2:

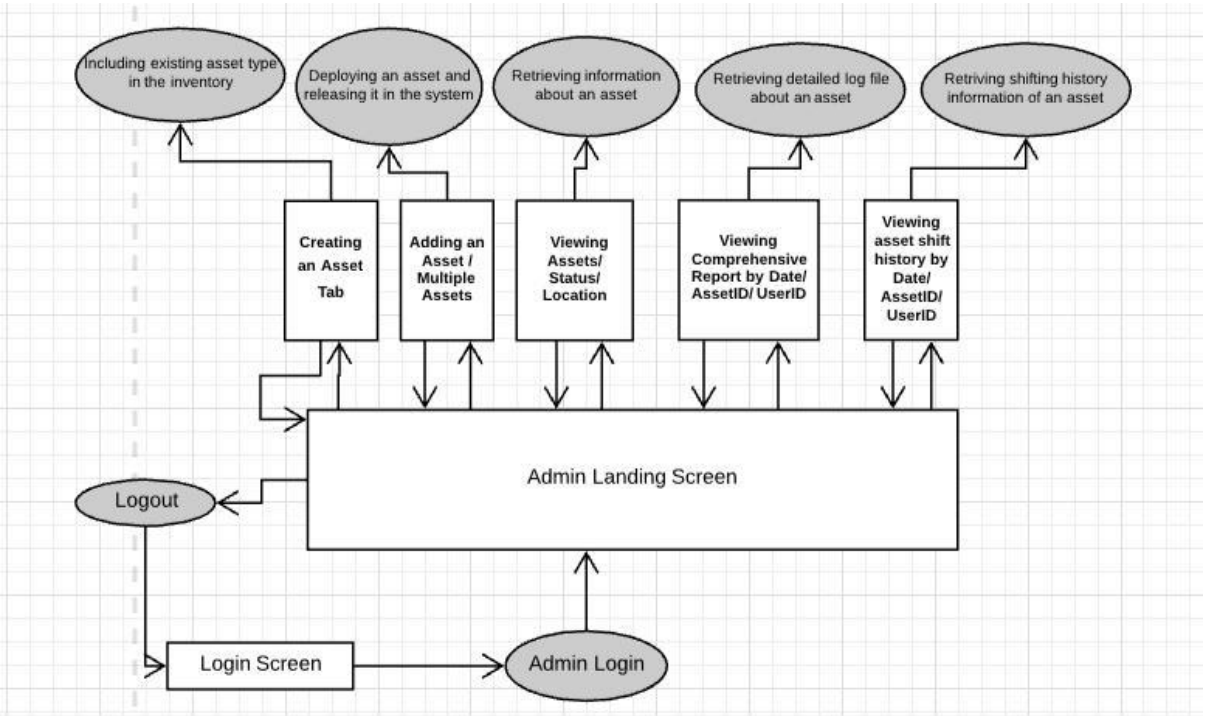


2.1 Overview of modules / components

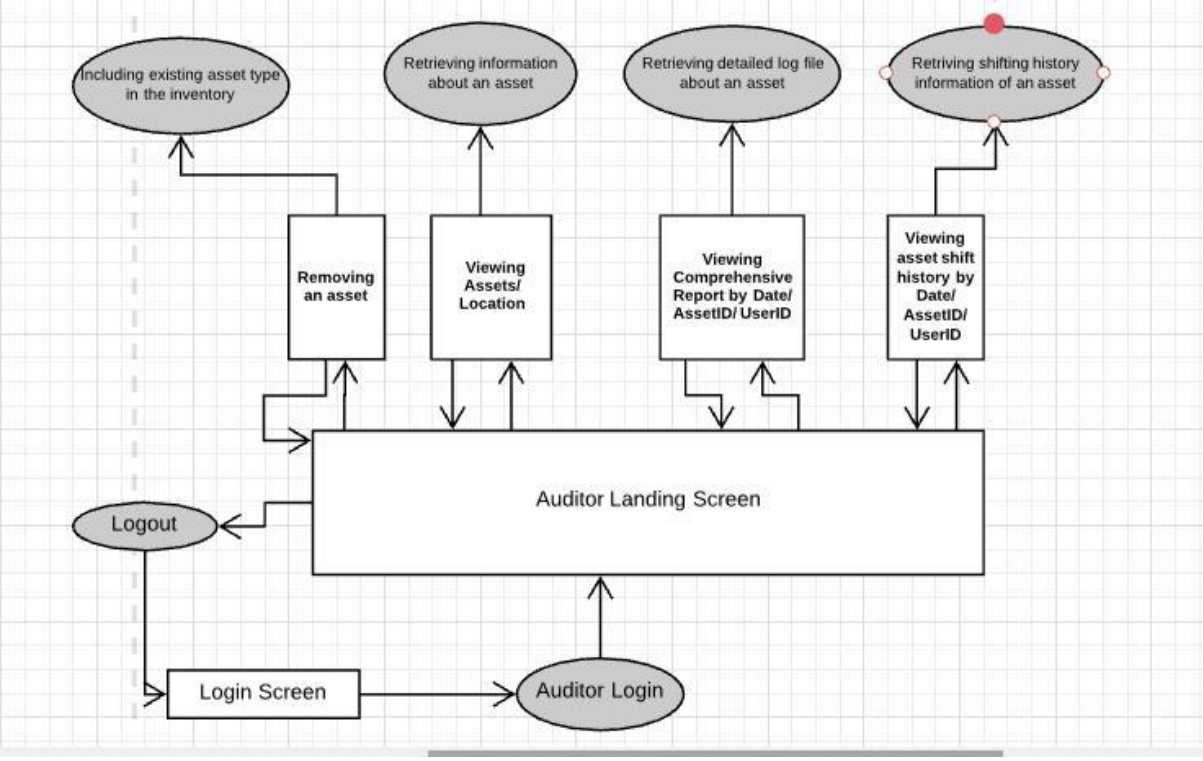


2.2 Structure and relationships

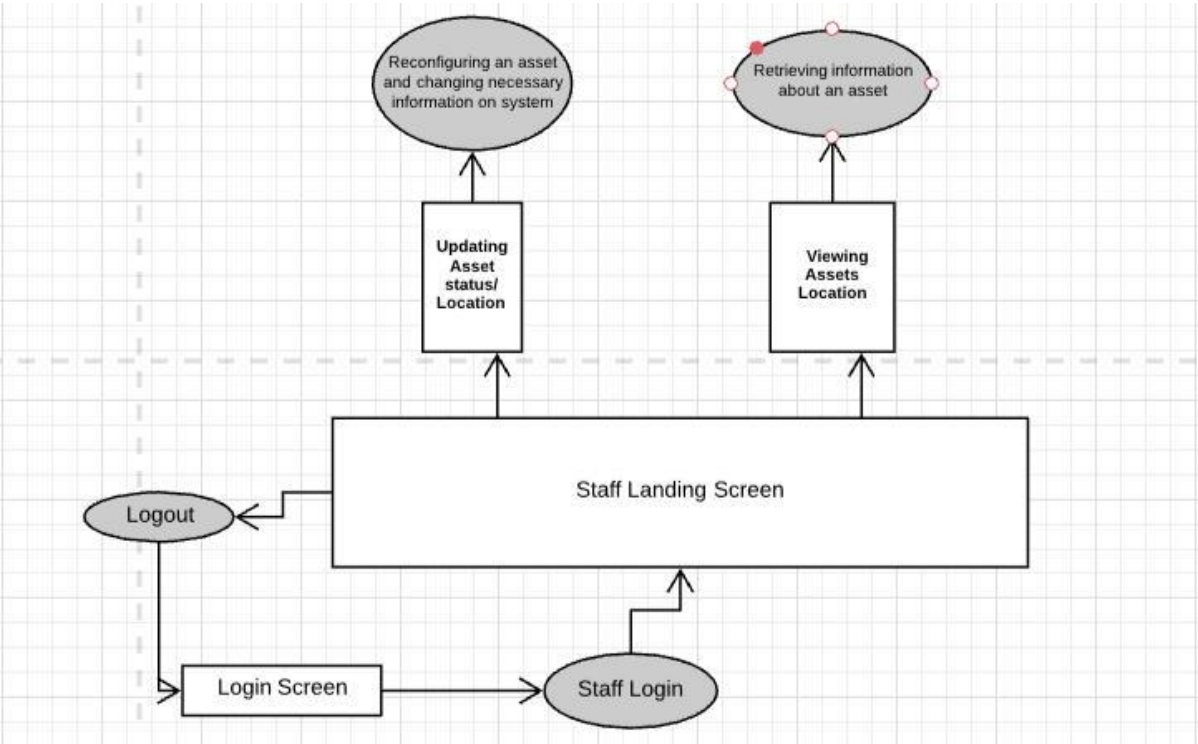
2.2.1 Admin's Side



2.2.2 Auditor's Side



2.2.3 Staff's Side



2.3 User interface issues

This section will address User Interface issues as they apply to the following hypothetical users of the Hostel Asset Management System.

- User X is a 33-year-old male, member of staff, assistant professor at ECE Department at NIIT University. He has been using various computing devices and has a somewhat applicable knowledge of Web-Applications, although he is not "smooth user", he can apply the basic information/instructions given on the user interface to go through the process of viewing asset and updating asset status and/or location and to make the process(es) easier, a hierarchical chain of command and execution oriented user interface is required, aided by messages at stages of required briefing about its actions.

Often this work may seem rigorous and a bit too dynamic hence the user interface is made friendly to the user with careful exposure to the architecture, that is preventing the intimidation and confusion caused by non-supervised release of information.

- User Y is a 36-year-old female employee of the non-teaching staff, who oversees the audit part of the different infrastructural needs at NIIT University.

Her work begins after operations are performed by staff member and/or the admin. Though she isn't proficient at technological advances, she understands the architecture well enough to do her job, this is where we become the middlemen to provide her with ease of access and constructive design of our web-application so as to avoid shortcomings and out of order foundations of the architecture and at the same time being user friendly and less rhetorical.

- User Z is a 45-year-old female, head at administrative task force, who oversees all operations under staff and auditor actors and the managing of the Asset management system as whole.

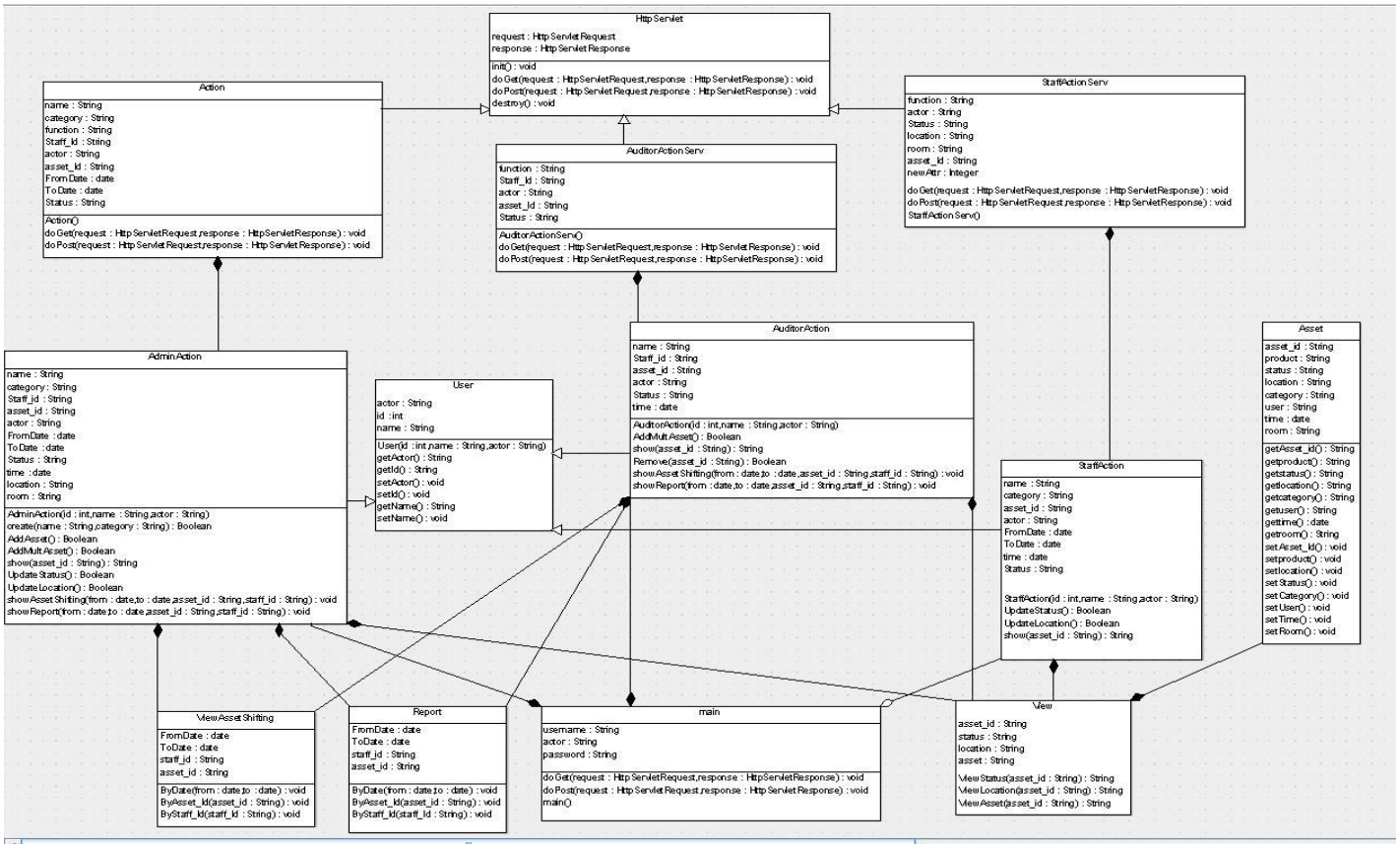
Due to responsibilities and the need of the productive throughput the occurrence of error i.e. unprecedented features or loopholes and system failure is burdening and the reliability of the system is expected to be at par with all acting factors, form-of-actions and feasible forecasts of functionality with administrative clearances and job-functions independent from the descendants (Auditor and Staff).

As the administrator is answerable and responsible to the actions and decisions of the system we tried to make the filing and handling part to be effortless, giving the

administrator space to feel free and get a hold of the architecture of Hostel Asset Management System.

3. Logical Architecture Description

3.1.1 Class Diagram:



Class Diagram Description:

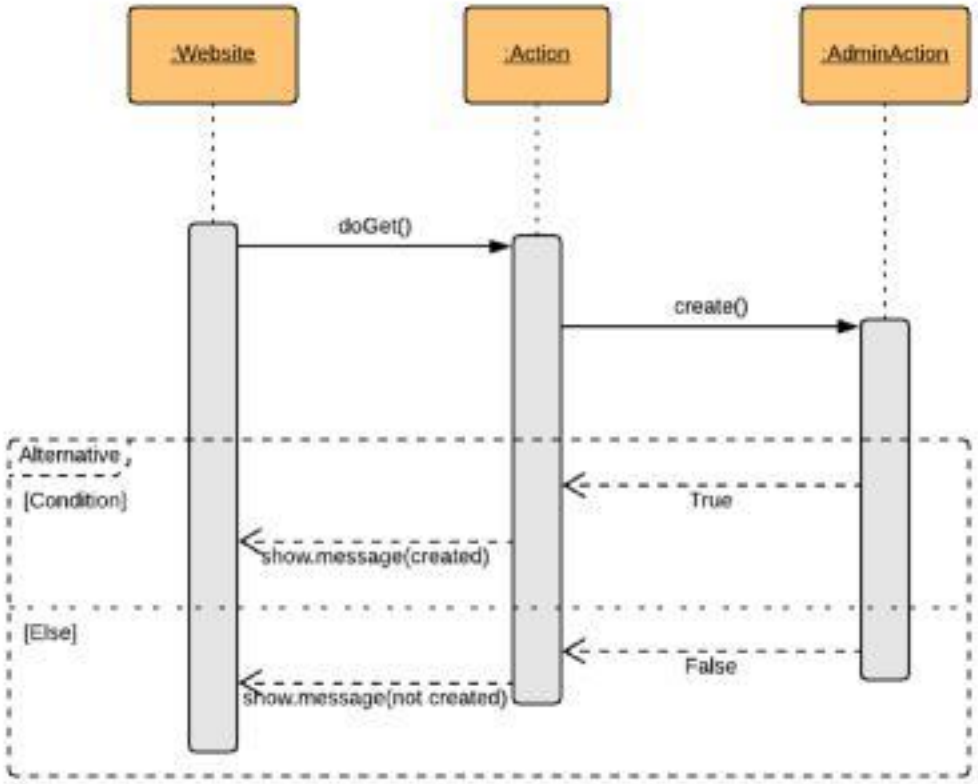
Here, the HttpServlet class is inherited by all the Servlet Classes. User class is inherited by Java classes which have the methods for different actions by different users.

All the methods are discussed class wise in Section 3.2. The class description as well as the different methods along with their Input and output objects and a detailed description in mentioned. The Class diagram here acts as the basic framework as well as the design architecture in the development of the software.

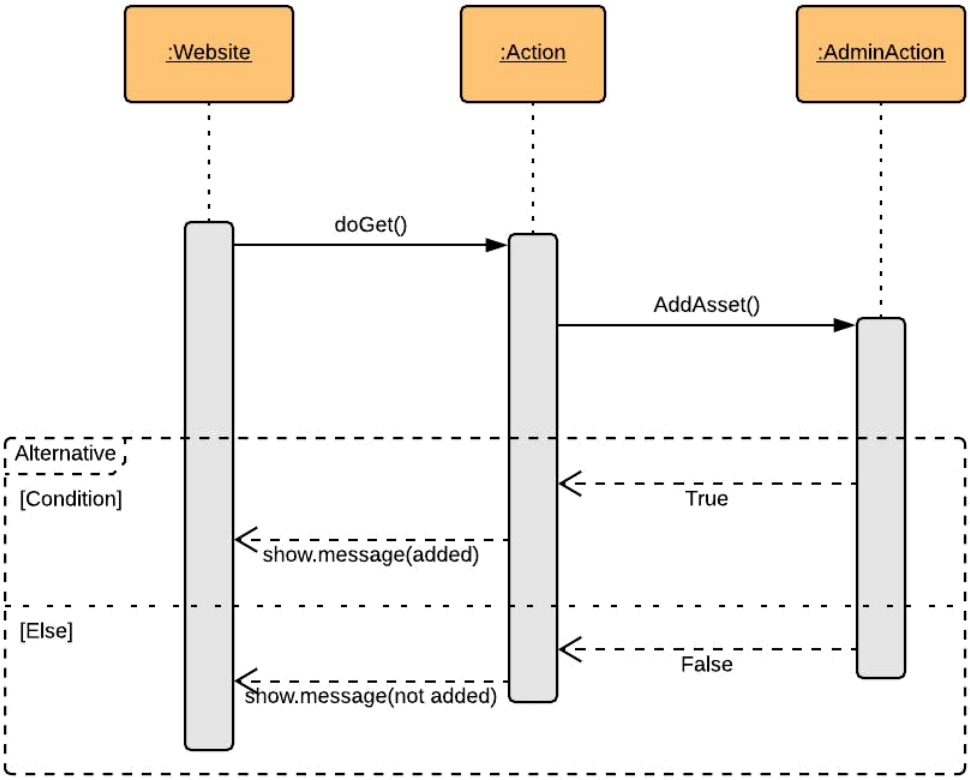
A total of 13 classes are to be observed with different functionality for each class. These classes' methods are called whenever we a specific action is to be performed.

3.1.2 Sequence Diagram:

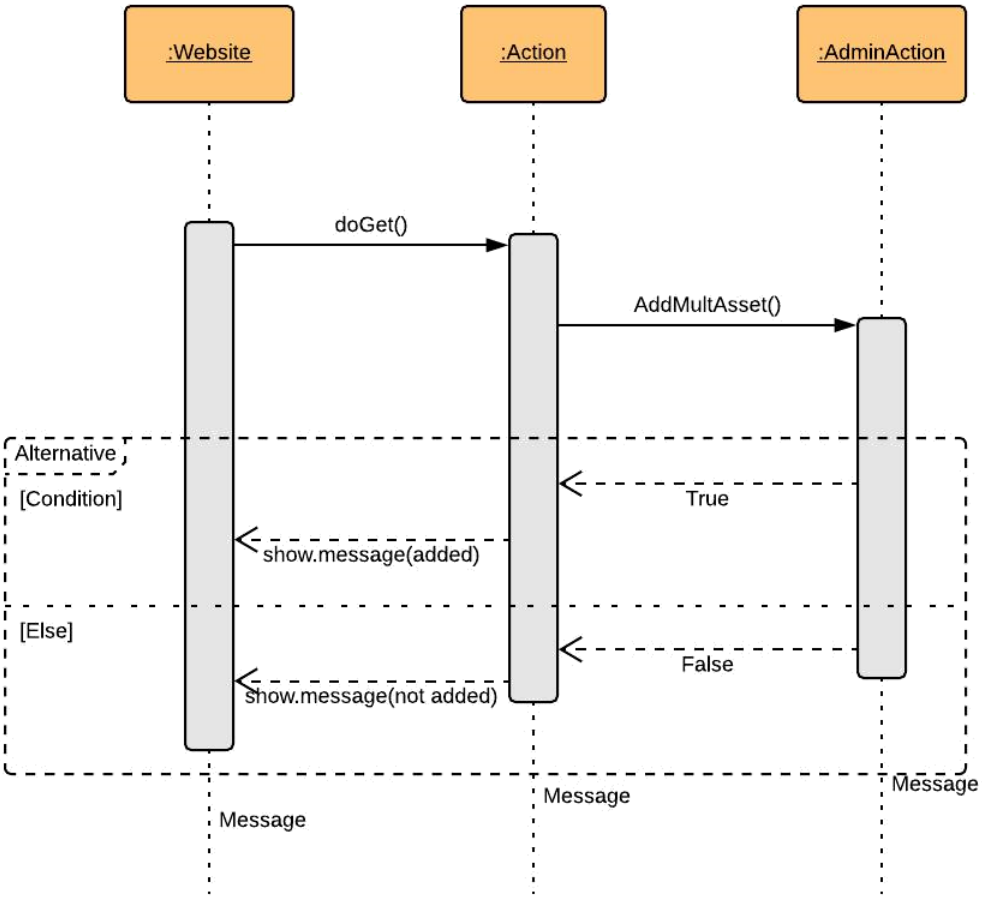
Arrow line signifies there is a send message taken place. Response is being shown by dotted arrows.



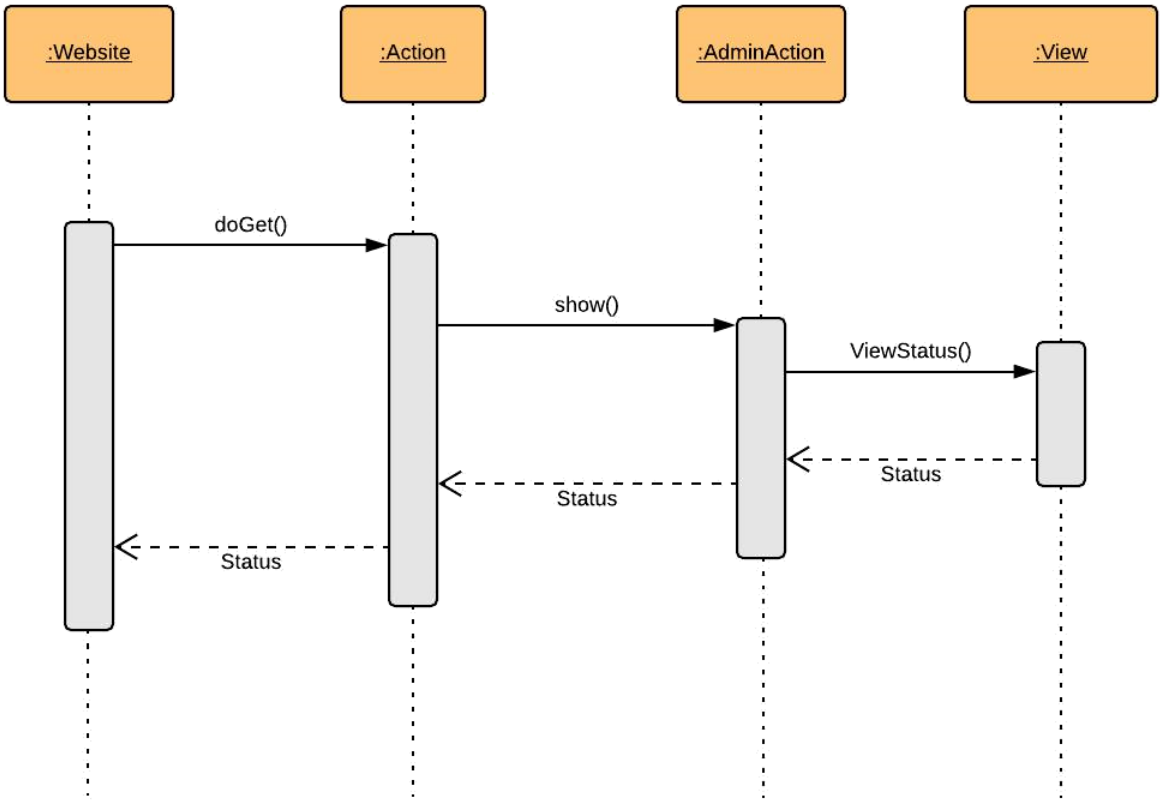
3.1.2.1 Create Asset: The doGet() method calls the create() method which creates an asset and returns a true if created and false in any other case. This same message is shown to the user through a dialog box.



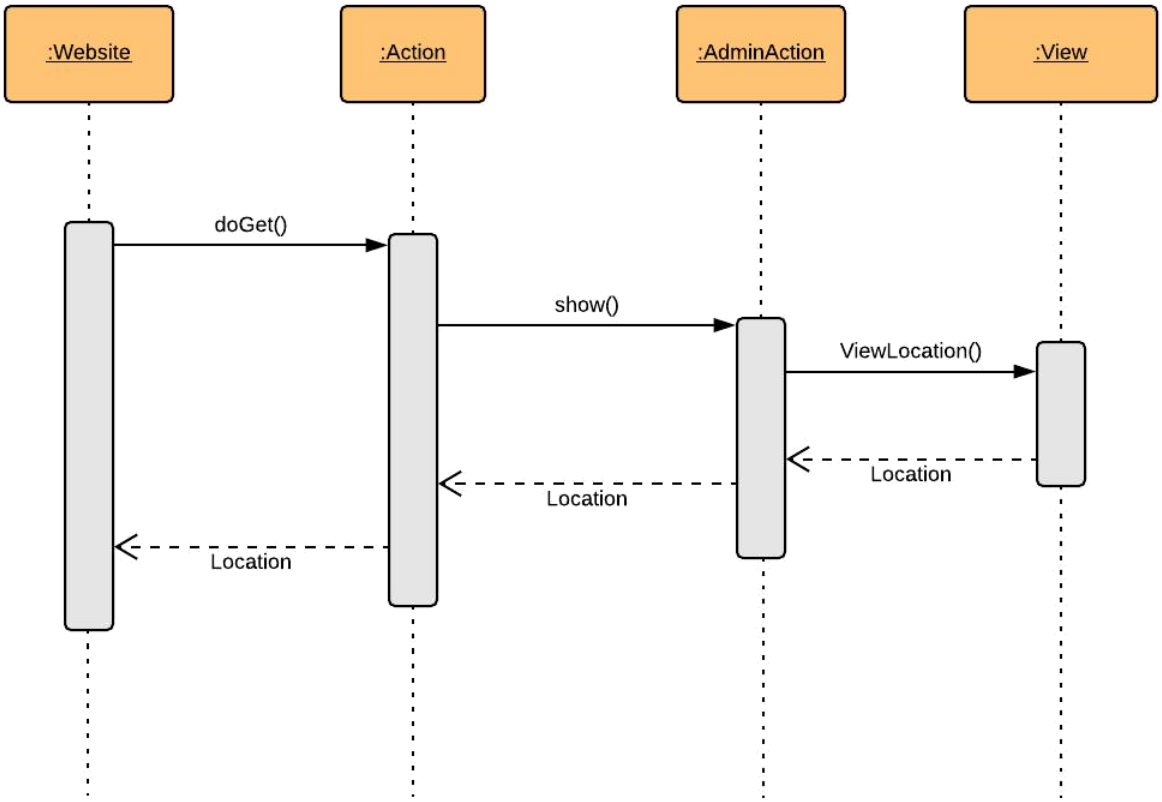
3.1.2.2 Add Asset: The doGet() method calls the AddAsset() method of AdminAction class which adds the details of the asset to the database and returns a message as the previous section.



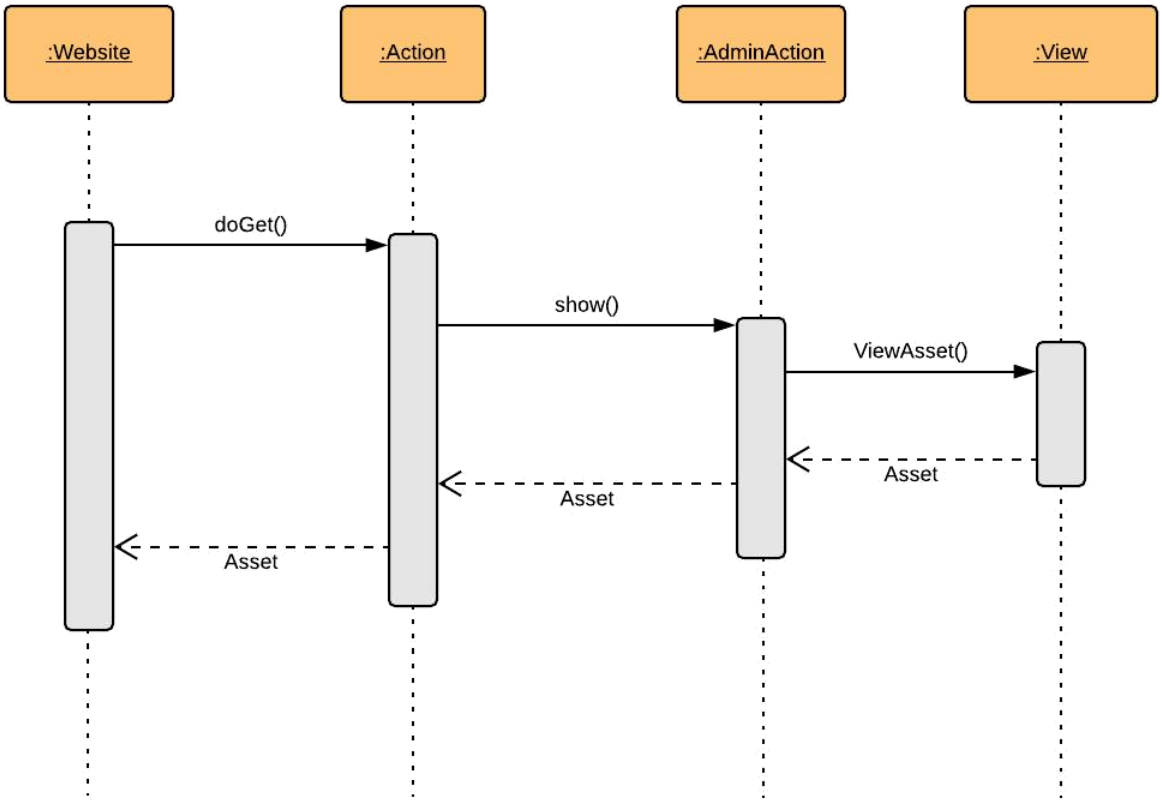
3.1.2.3 Add Multiple Asset: The doGet() method calls the AddMultAsset() method of AdminAction class which adds the details of the assets to the database and returns a message as the previous section.



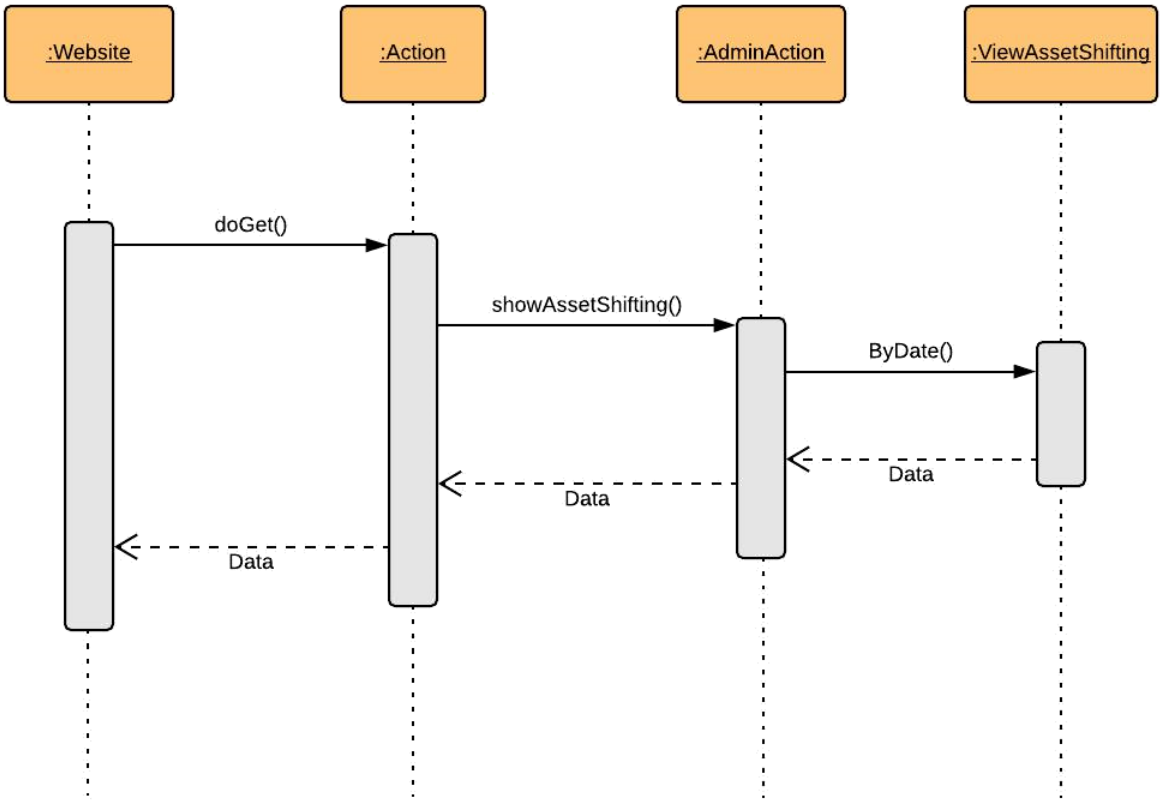
3.1.2.4 View Asset Status: The doGet() method of Action class is called which calls the show() method of AdminAction class which in turn calls the ViewStatus() method of View class. It returns a string with the status of the asset whose assetID is provided.



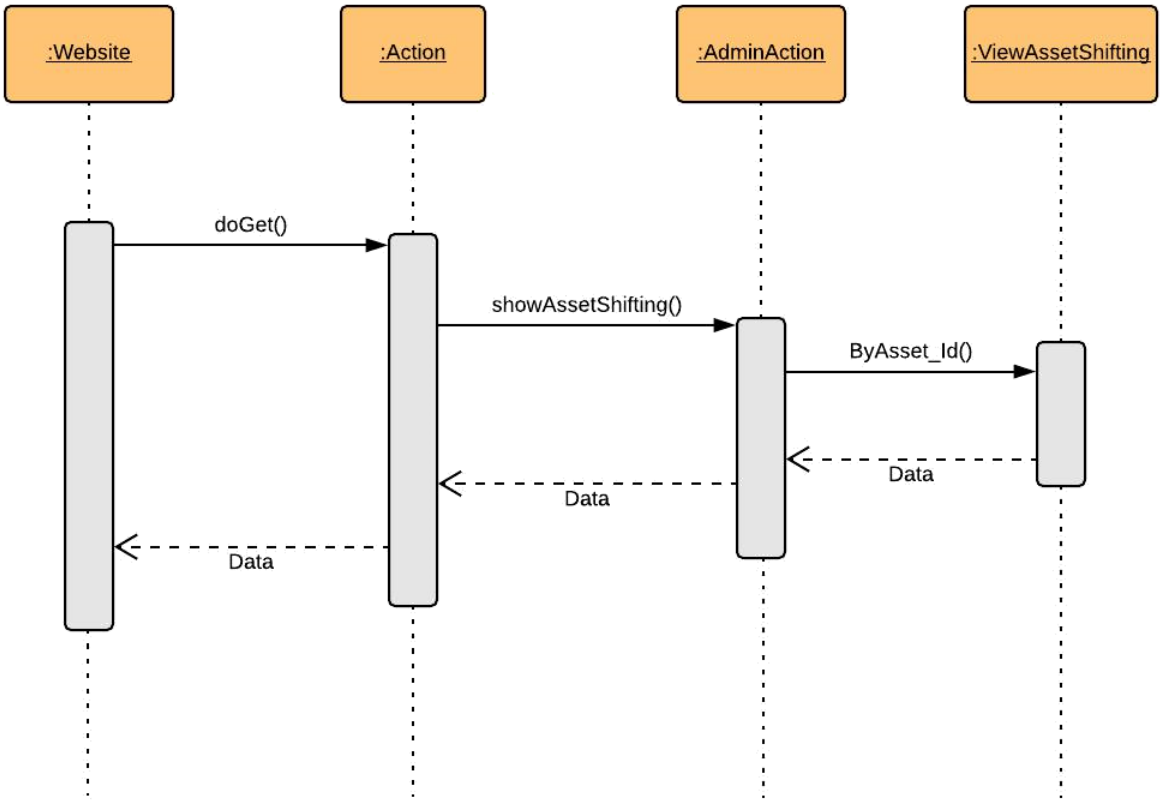
3.1.2.5 View Asset Location: The doGet() method of Action class is called which calls the show() method of AdminAction class which in turn calls the ViewLocation() method of View class. It returns a string with the location of the asset whose assetID is provided.



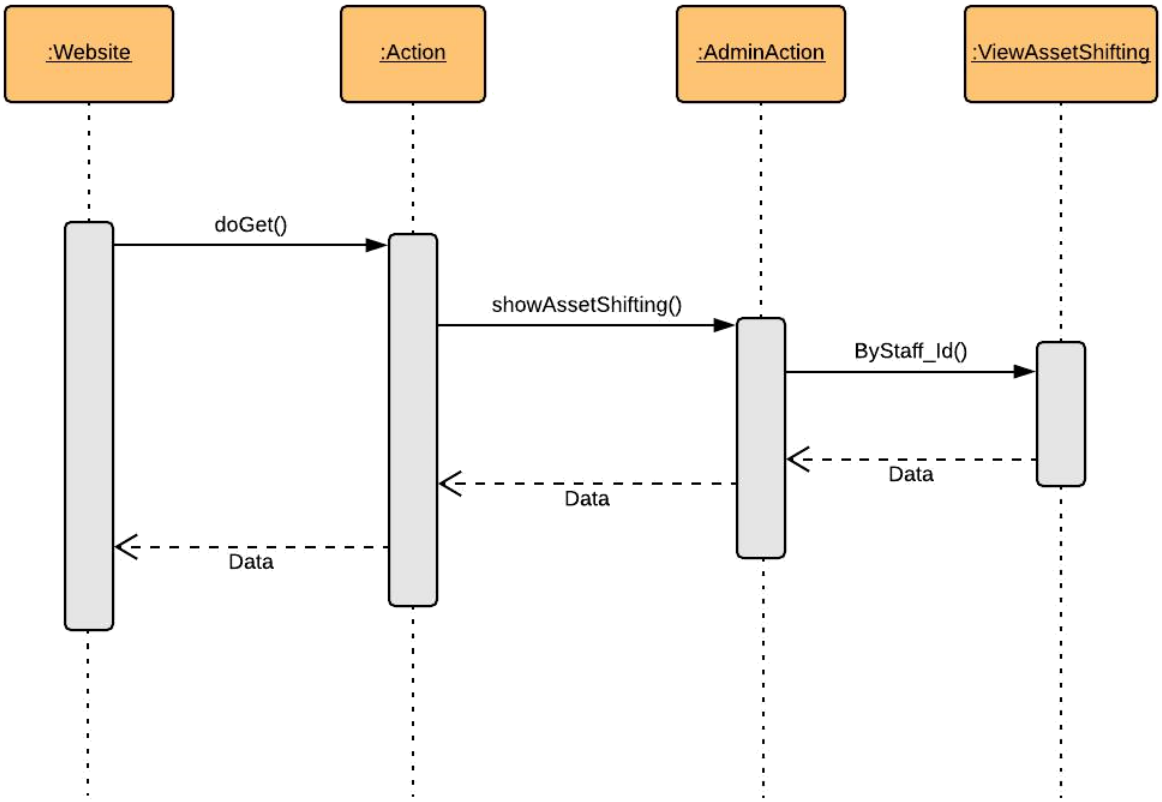
3.1.2.6 View Asset: The doGet() method of Action class is called which calls the show() method of AdminAction class which in turn calls the ViewAsset() method of View class. It returns a string Asset which has all the details of the asset whose assetID is provided.



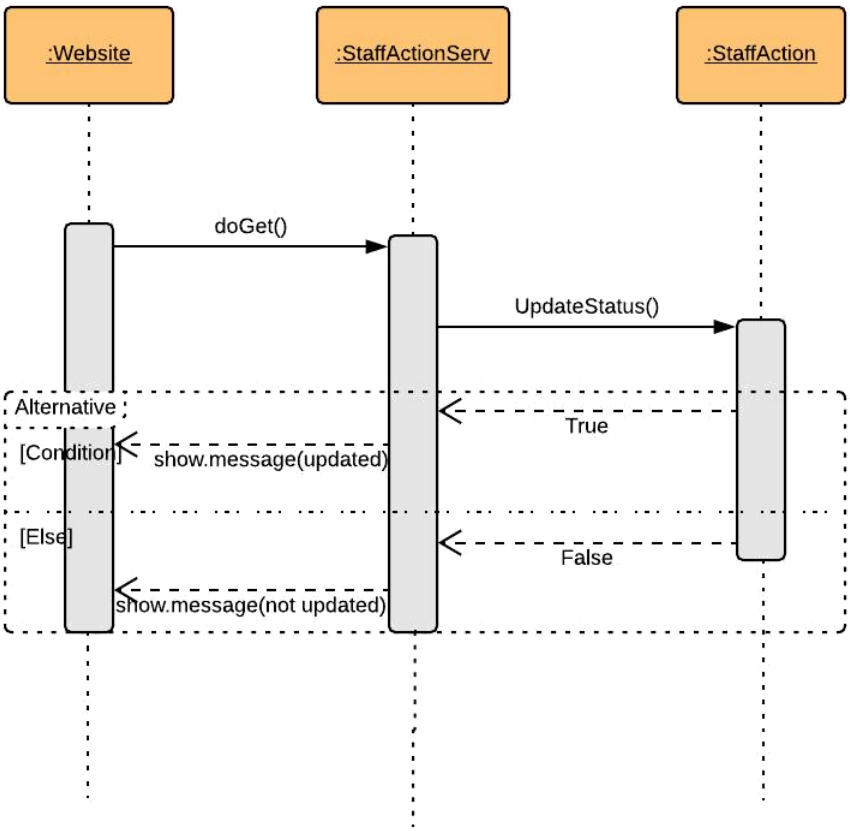
3.1.2.7 View Asset Shifting History by Date: The doGet() method of Action class is called which calls the showAssetShifting() method of AdminAction class which in turn calls the byDate() method of ViewAssetShifting class. It returns a table data which has all the shifting operations from the dates provided.



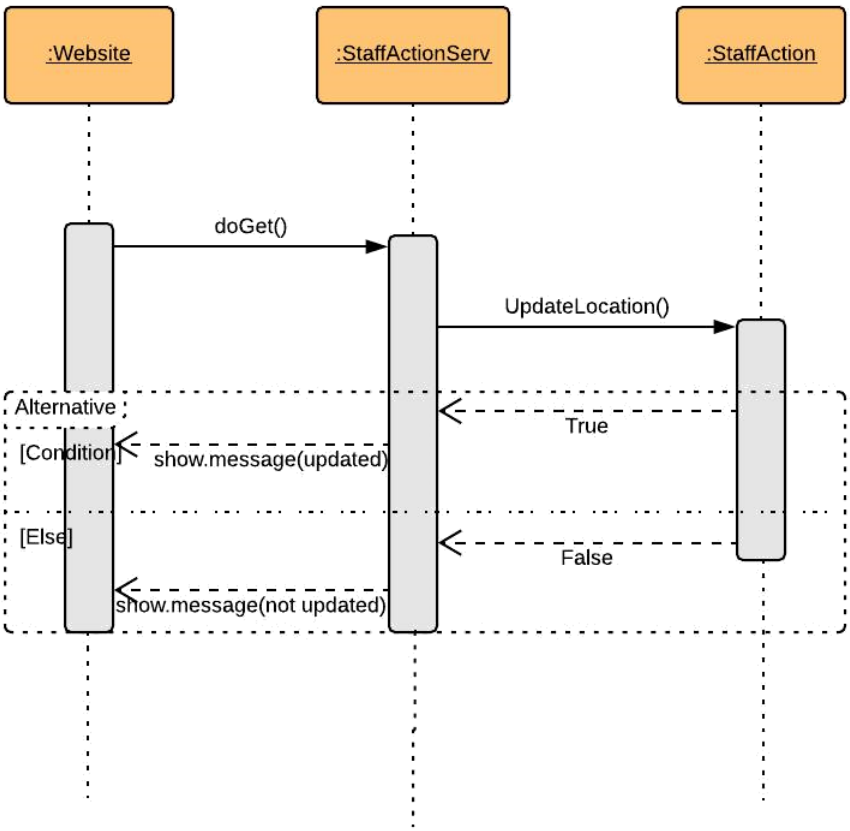
3.1.2.8 View Asset Shifting History by assetID: The doGet() method of Action class is called which calls the showAssetShifting() method of AdminAction class which in turn calls the byassetID() method of ViewAssetShifting class. It returns a table data which has all the shifting operations on a single asset.



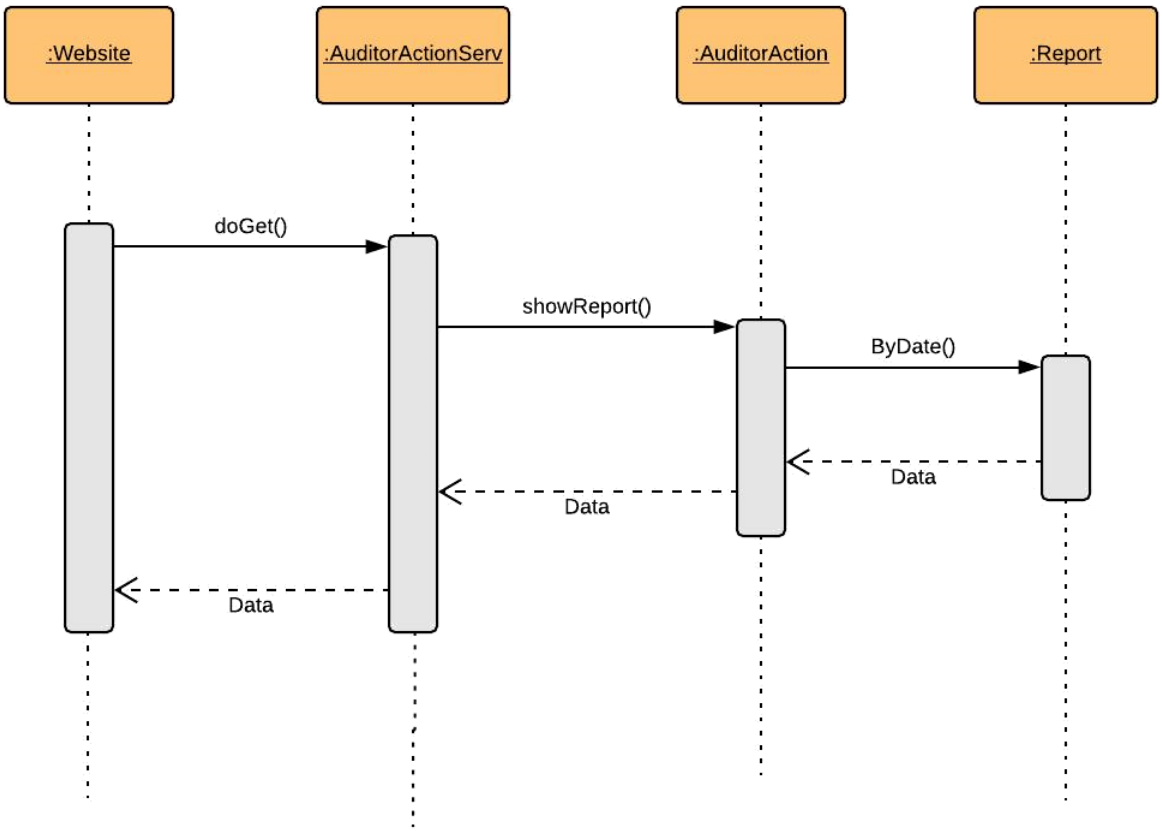
3.1.2.9 View Asset Shifting History by staffID: The doGet() method of Action class is called which calls the showAssetShifting() method of AdminAction class which in turn calls the bystaffID() method of ViewAssetShifting class. It returns a table data which has all the shifting operations by the staff member whose ID is provided.



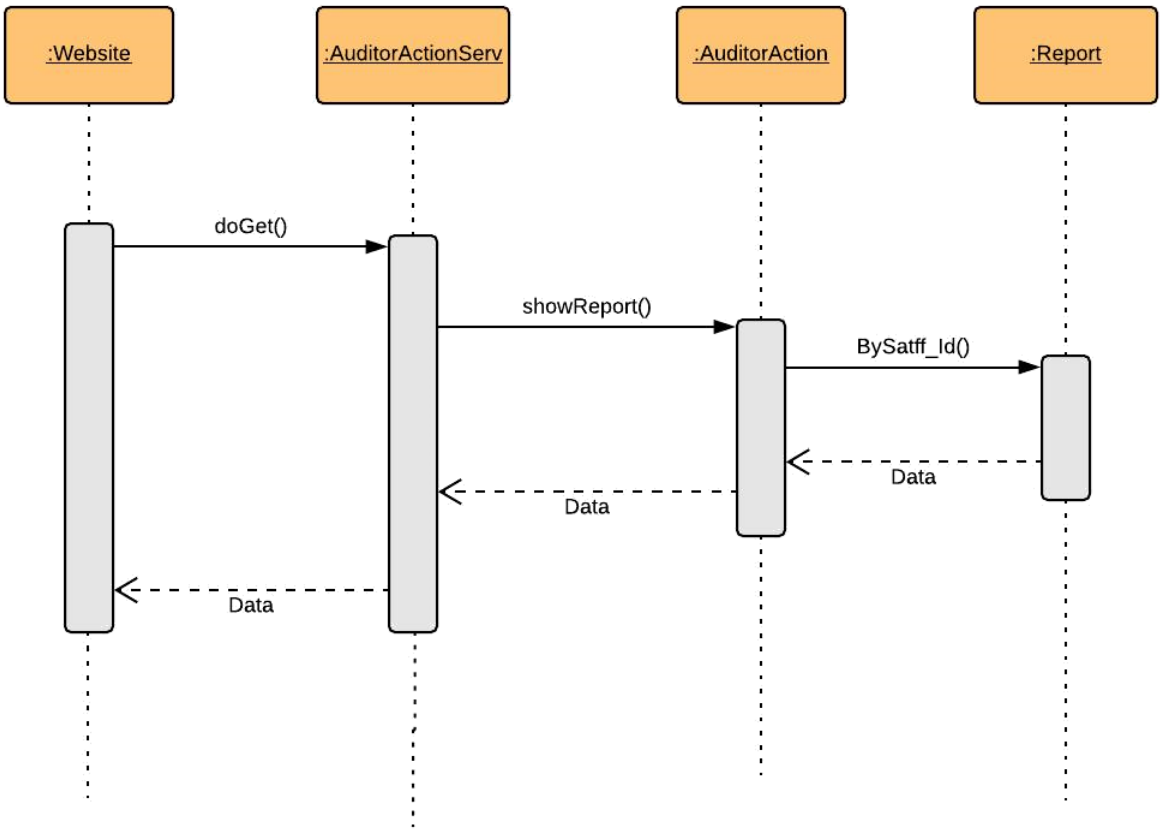
3.1.2.10 Update Asset Status: The doGet() method of StaffActionServ class is called which in turn calls the UpdateStatus() method of StaffAction Class. If the update is successful then, a TRUE is returned else, a FALSE is returned. These values decide the output to the user.



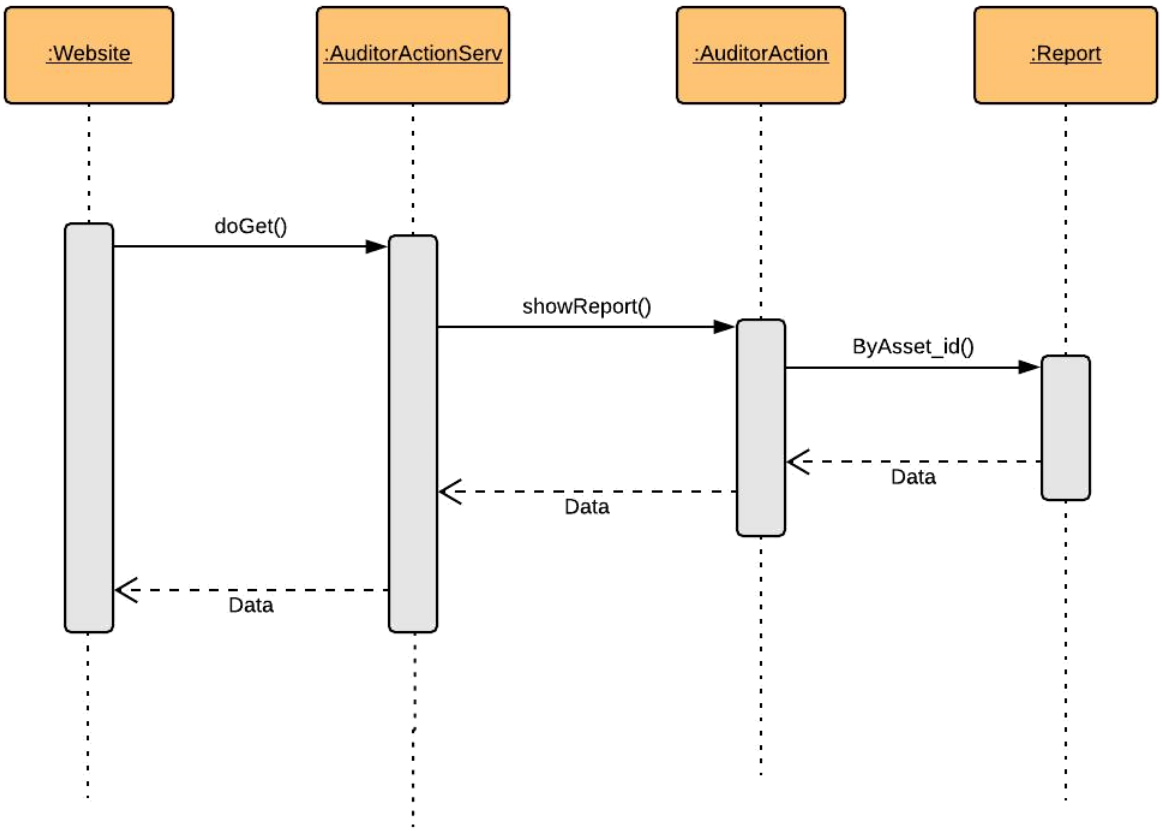
3.1.2.11 Update Asset Location: The doGet() method of StaffActionServ class is called which in turn calls the UpdateLocation() method of StaffAction Class. If the update is successful then, a TRUE is returned else, a FALSE is returned. These values decide the output to the user.



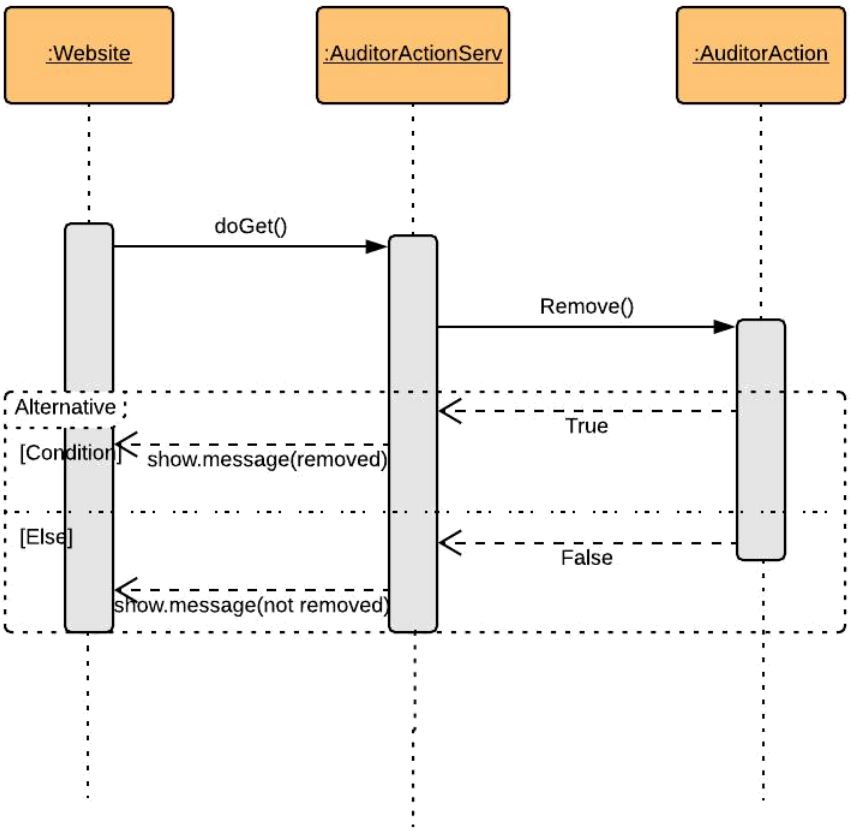
3.1.2.12 Show Report by Date: The doGet() method of AuditorActionServ class is called which calls the showReport() method of AuditorAction Class which in turn calls the byDate() method of the Report class. It returns a table of all the operations between the provided dates.



3.1.2.13 Show Report by StaffID: The doGet() method of AuditorActionServ class is called which calls the showReport() method of AuditorAction Class which in turn calls the byStaffID() method of the Report class. It returns a table of all the operations by the staff member from day of joining to last operation.



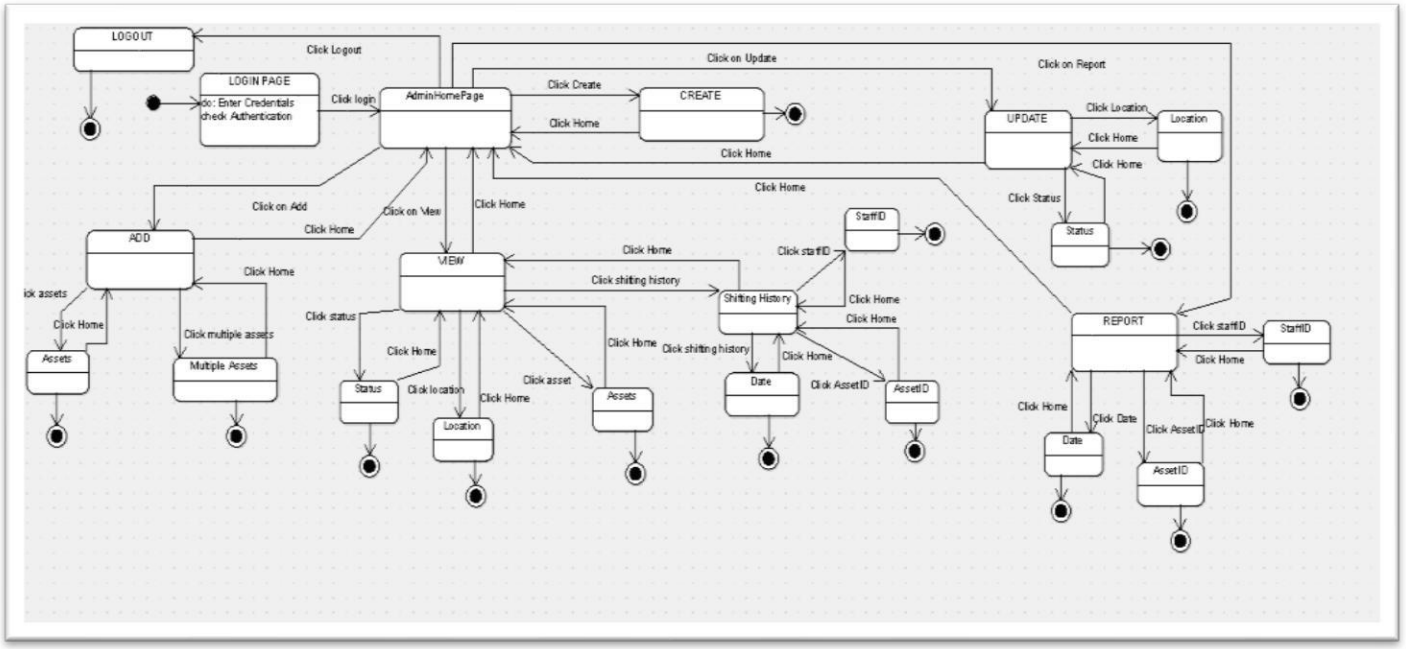
3.1.2.14 Show Report by AssetID: The doGet() method of AuditorActionServ class is called which calls the showReport() method of AuditorAction Class which in turn calls the byAssetID() method of the Report class. It returns a table of all the operations on the asset with the assetID assigned.



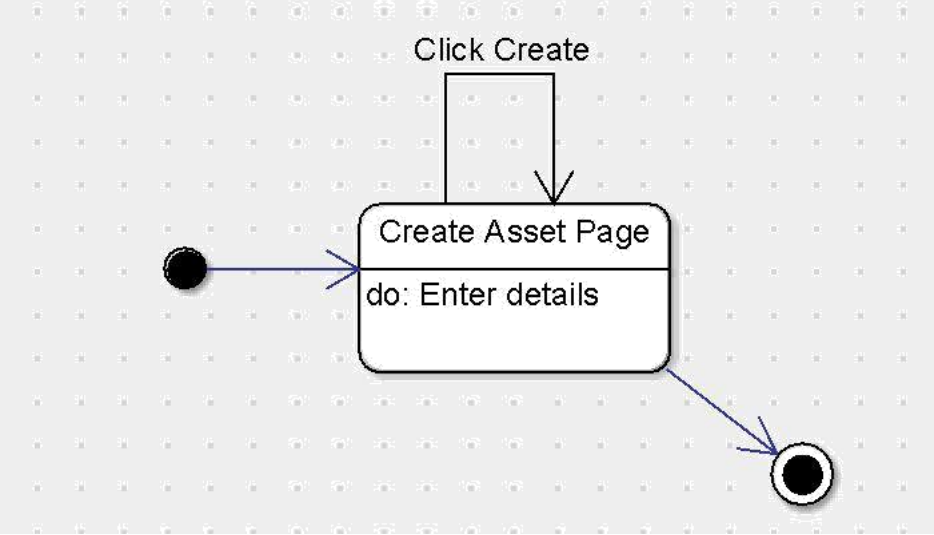
3.1.2.15 Remove Asset: The doGet() method of AuditorActionServ class is called which calls the remove() methods of AudtiorAction class. It removes the asset from the database and returns TRUE if the operation was successful else returns a FALSE.

3.1.3 State Diagram:

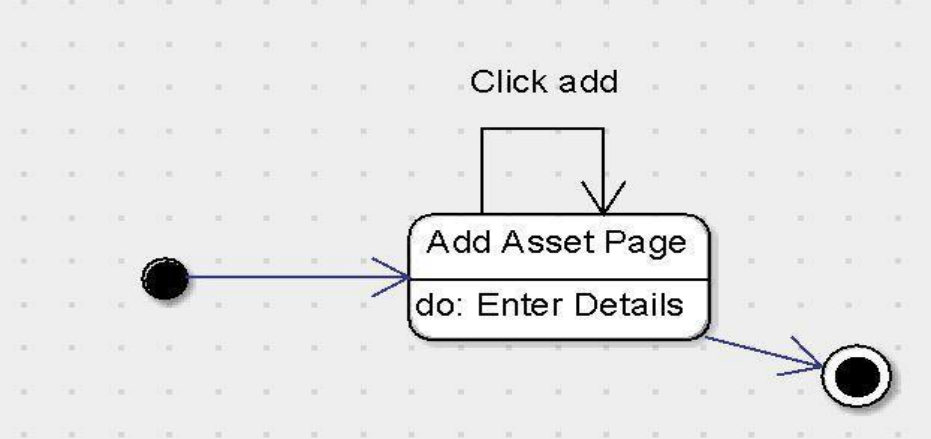
Initial state is being shown by starting with a black dot. Final State is being shown by the black dot surrounded by an empty circle.



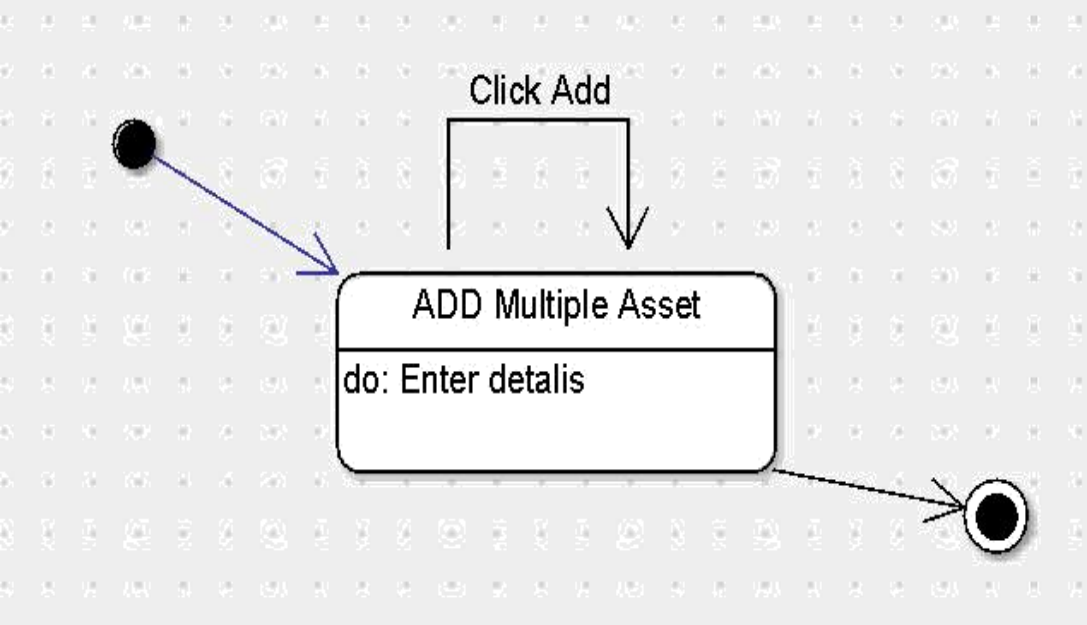
3.1.3.01 Admin Landing Page: This page has all the operations for the user. Here, the user is directed to each and every new page depending the function or use of the application. User can even logout from here. Which is actually signing off the application.



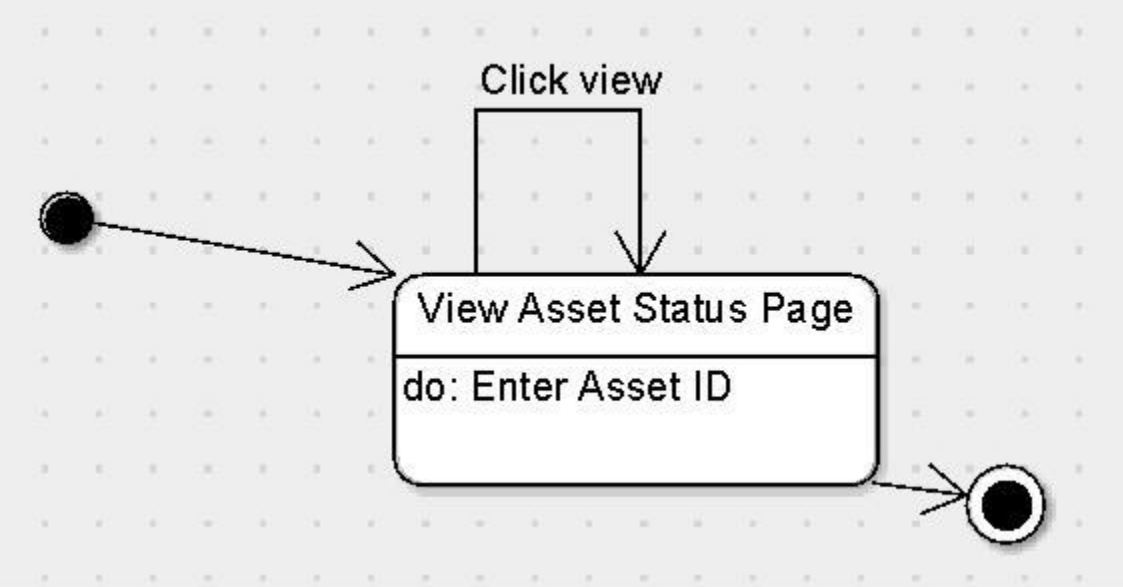
3.1.3.02 Create Asset: It allows user to enter the name as well as the category of the asset which will be added later. Entering data into this page tells to the system that there is a new type of asset coming in. Later, the assets purchased will be added using the add asset page.



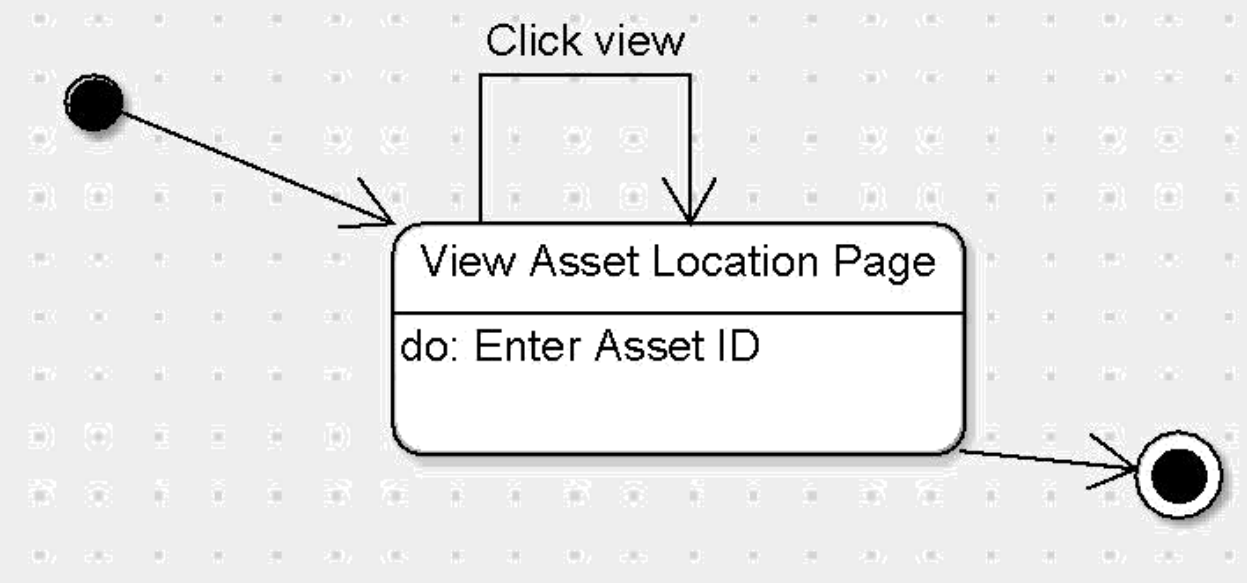
3.1.3.03 Add Asset: It allows user to add an asset which was already created. The user can only add an asset and get the Asset ID of the asset for future reference.



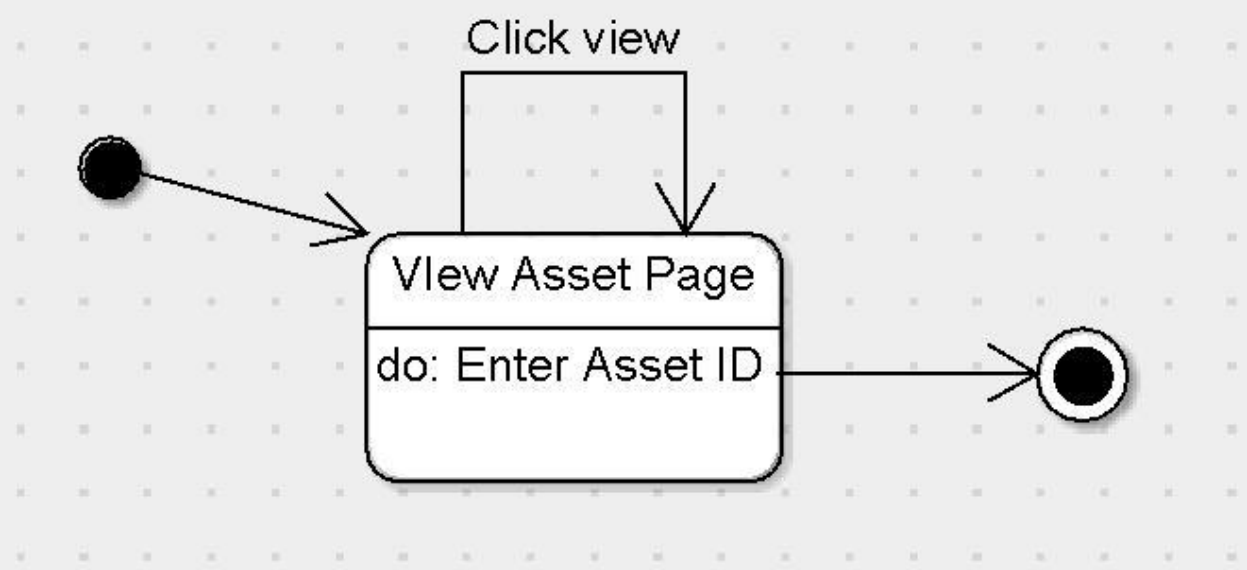
3.1.3.04 Add Multiple Asset: It allows user to more than one asset of same type at a single click. The user provides the Product ID (which is different for each type of asset) and the number of assets to be added. The output would be a set of Asset IDs for each asset for future reference.



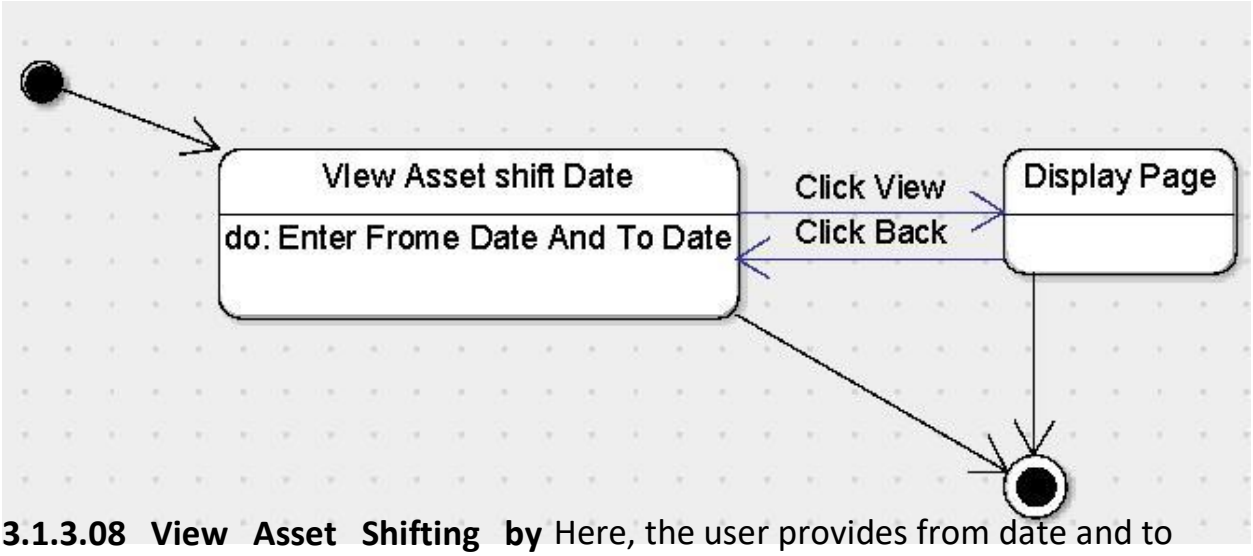
3.1.3.05 View Asset Status: The user gets the status of the asset i.e., broken, can't be repaired, to be repaired, beyond economic rate and many more just by providing the Asset ID of the asset and clicking View button.



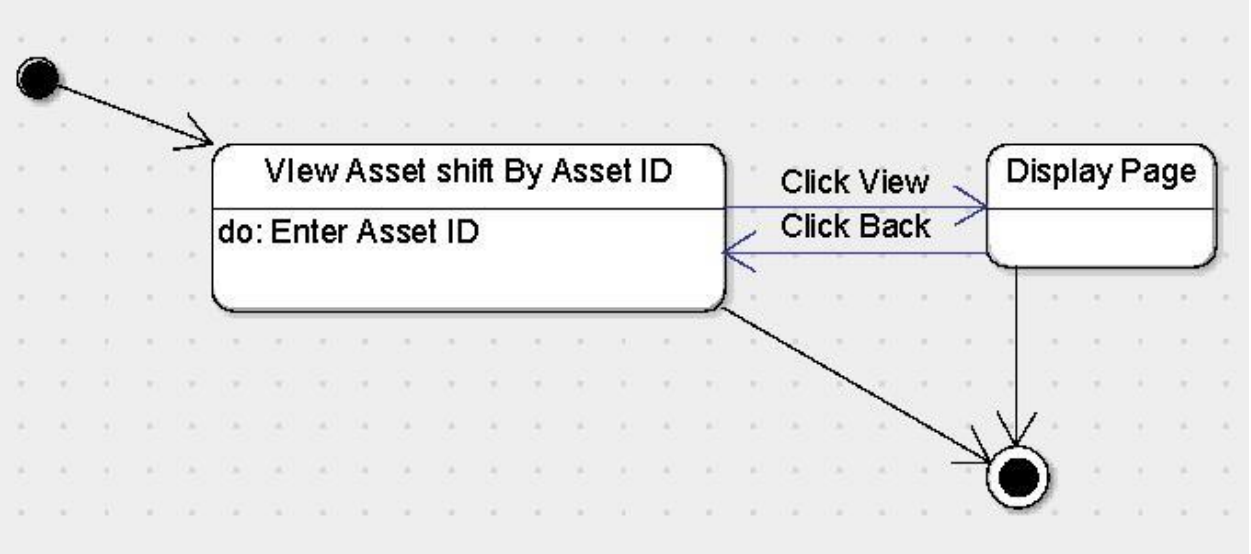
3.1.3.06 View Asset Location: The user gets the location of an asset just by providing the Asset ID of the asset and clicking View button.



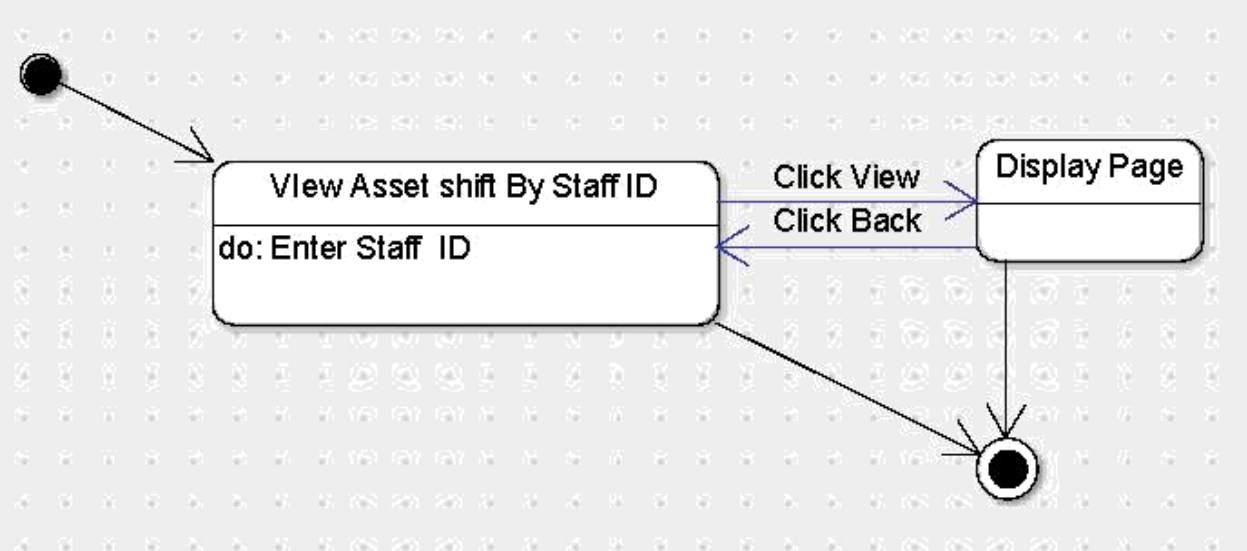
3.1.3.07 View Asset: The user gets the whole details and description i.e., status, location, date of purchase and many more just by providing the unique asset ID and clicking View button.



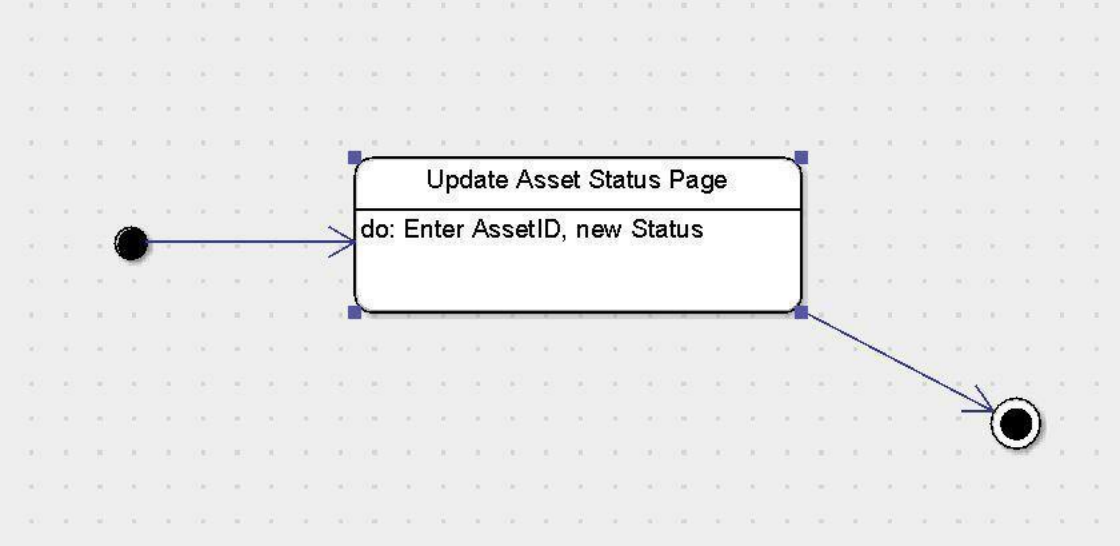
3.1.3.08 View Asset Shifting by Date: Here, the user provides from date and to date of all the assets done in the duration and gets all the shifting operations clicking the View button. just by



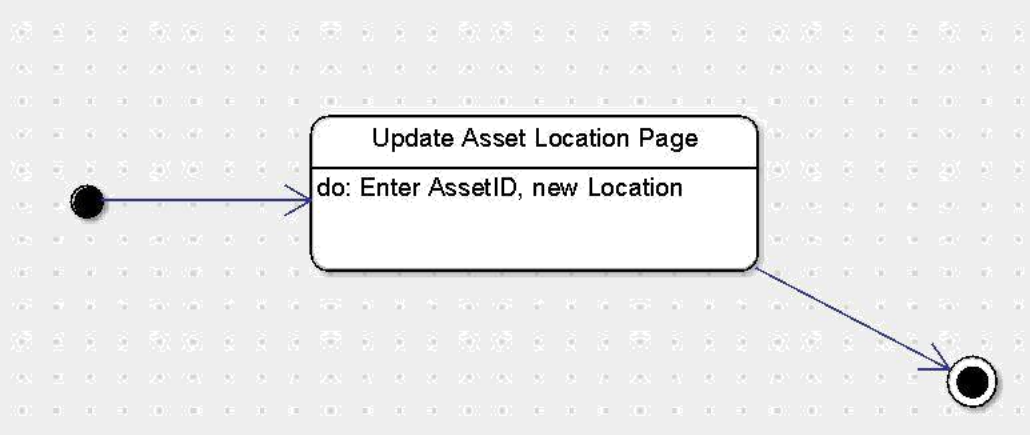
3.1.3.09 View Asset Shifting by Asset ID: Here, the user provides the Asset ID of an asset and gets all the shifting operations done to the asset from the time of purchase just by clicking the View button.



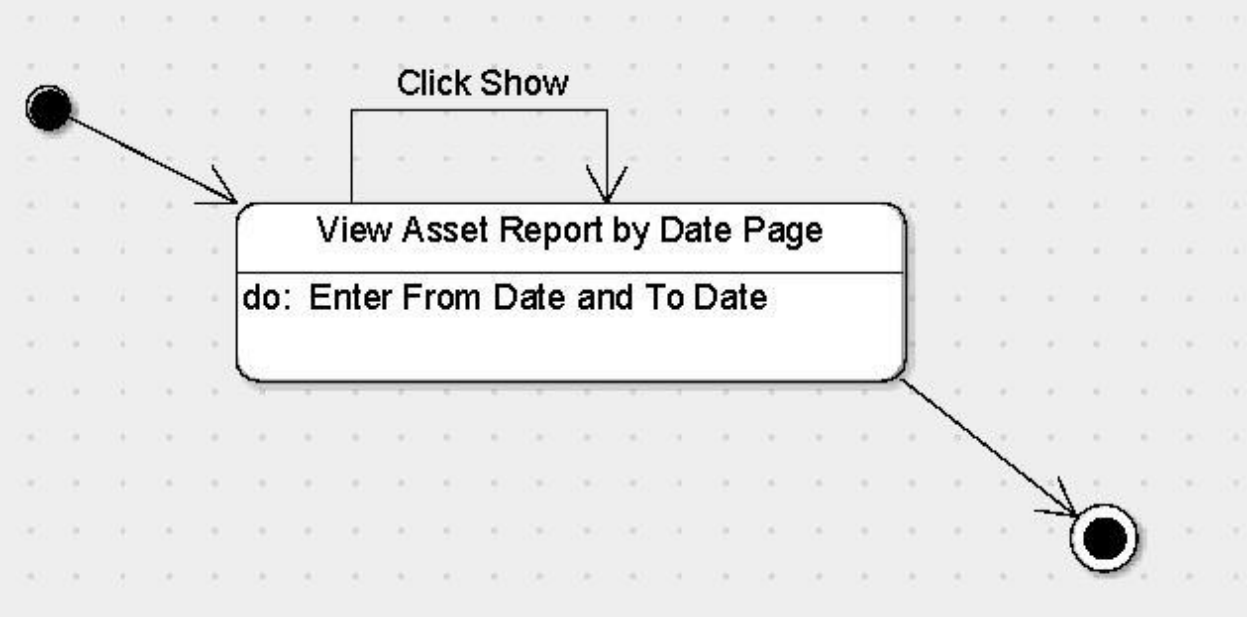
3.1.3.10 View Asset Shifting by Staff ID: Here, the user provides the Staff ID of a particular staff and gets all the shifting operations done by the staff from the time of joining the organization just by clicking the View button.



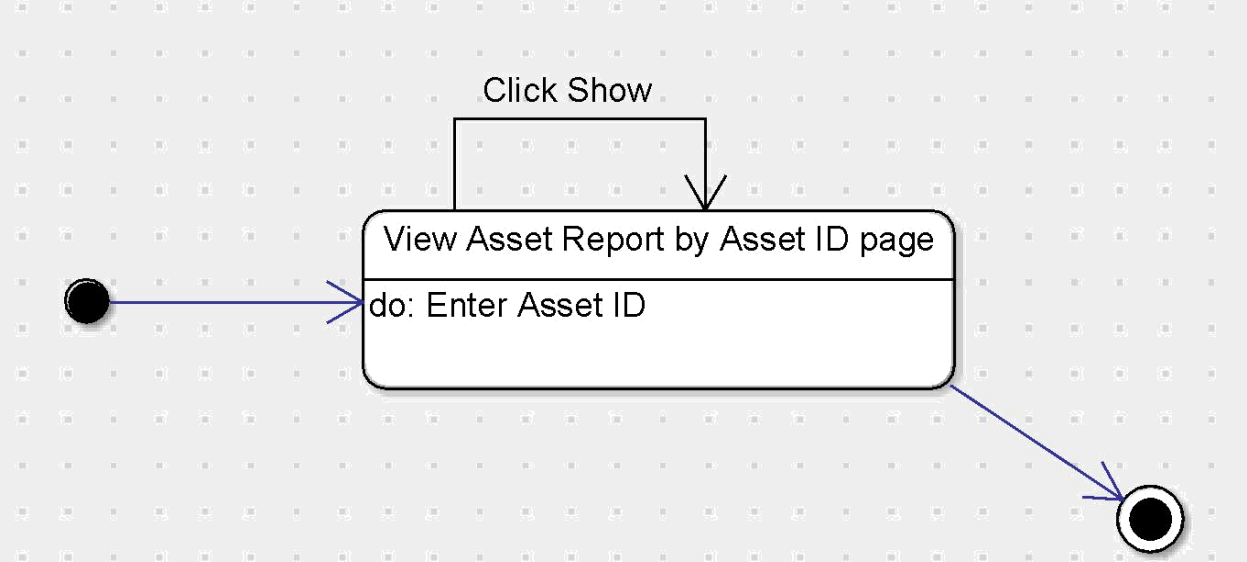
3.1.3.11 Update Asset Status: The user provides the asset ID of the asset and the new status which are predefined and by clicking the update button the status of the asset is updated.



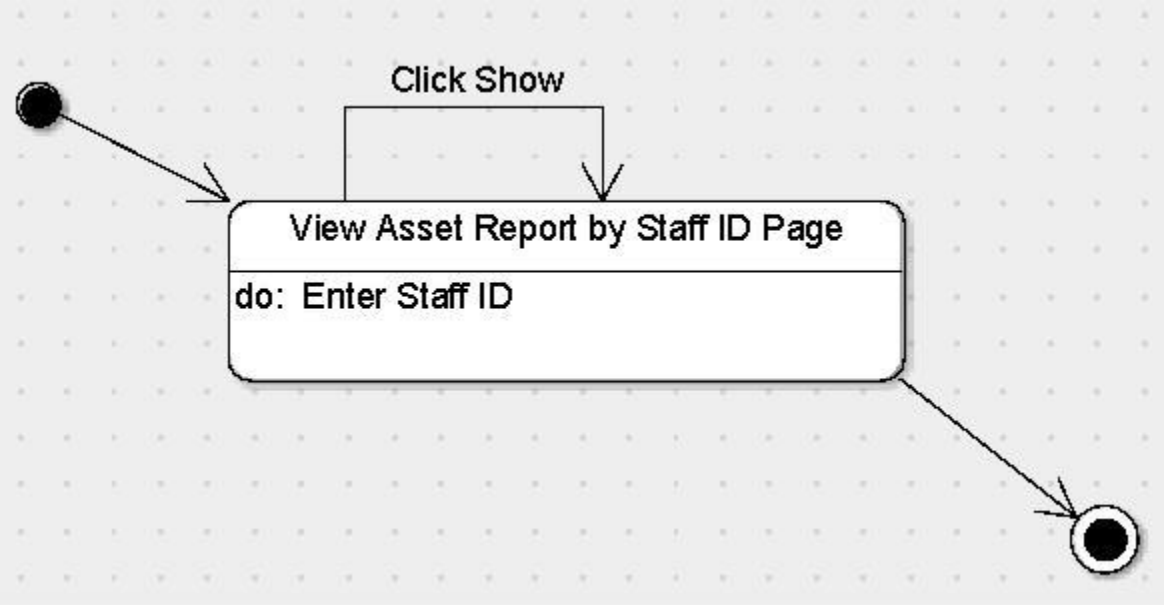
3.1.3.12 Update Asset Location: The user provides the asset ID of the asset, the new location and room to which the asset is being shifted to and by clicking the update button the location of the asset is updated.



3.1.3.13 View Report by Date: The user here gets the whole transactions occurring the whole-time frame between from date and to date. The output is a list of all the transactions in a tabular format in a different page.



3.1.3.14 View Report by Asset ID: The user here gets the whole transactions occurring to a single asset like adding, updating status, updating location and many more operations on a single asset just by providing the asset ID and clicking the Report button.



3.1.3.15 View Report by Staff ID: The user here gets the whole transactions done by a staff member like adding assets, updating asset status and locations and many other operations performed by a specific member of the team just by providing the staff ID and clicking the report button.

3.2 Class name: HttpServlet

Description: This is an in-built class in java servlet package which is used in almost all the web-based projects built using java. It has a few basic functions which are can be used for easy programming and fewer code lines.

3.2.1 Method 1: doGet()

Input: HttpServletRequest request, HttpServletResponse response

Output: void

Method Description:

Every action to be done passes through this function. The data which comes from the website from the form is stored in request object irrespective of the type and number of inputs. This method has response object which actually creates the response to the user in the form of dynamic web pages and puts it in a presentable format. All this is done in the doGet method and directing between web pages and .jsp files for appropriate functioning of the application.

3.2.2 Method 2: doPost()

Input: HttpServletRequest request, HttpServletResponse response

Output: void

Method Description:

This function is generally used in the place of doGet() method in special cases where you prefer hiding the url of the server and the data you send from one page/jsp file to other page/jsp file.

3.2.3 Method 3: init()

Input: void

Output: void

Method Description:

This is the servlet inbuilt function which actually builds the servlet. Extra functionality like starting the servlet with some objects and some other initial actions can be performed in this function.

3.2.4 Method 4: destroy()

Input: void

Output: void

Method Description:

This is the servlet inbuilt function which runs at the end of any servlet. This function can print some message, free some heap area or some other actions which are to be performed at the end of the servlet. Like, the operations to be performed just before the servlet dies are to be written in this function.

3.3 Class name: Action

Description: This is a servlet which inherits the HttpServlet class using the doGet() and doPost() methods of the parent class. This servlet is where the admin is headed after login. After login which includes authentication of the user, the user has a set of options to choose from. The user can then choose one of these just by hovering and clicking the option required. The servlet then forwards to the concerned class/jsp/servlet to perform the actions as per user's need.

3.3.1 Method 1: doGet()

Input: HttpServletRequest request, HttpServletResponse response

Output: void

Method Description:

Every action to be done passes through this function. The data which comes from the website from the form is stored in request object irrespective of the type and number of inputs. This method has response object which actually creates the response to the user in the form of dynamic web pages and puts it in a presentable format. All this is done in the doGet method and directing between web pages and .jsp files for appropriate functioning of the application.

3.3.2 Method 2: doPost()

Input: HttpServletRequest request, HttpServletResponse response

Output: --

Method Description:

This function is generally used in the place of doGet() method in special cases where you prefer hiding the url of the server and the data you send from one page/jsp file to other page/jsp file.

3.4 Class name: AuditorActionServ

Description: This is a servlet which inherits the HttpServlet class using the doGet() and doPost() methods of the parent class. This servlet is where the auditor is headed after login. After login which includes authentication of the user, the user has a set of options to choose from. The user can then choose one of these just by hovering and clicking the option required. The servlet then forwards to the concerned class/jsp/servlet to perform the actions as per user's need.

3.4.1 Method 1: doGet()

Input: HttpServletRequest request, HttpServletResponse response

Output: void

Method Description:

Every action to be done passes through this function. The data which comes from the website from the form is stored in request object irrespective of the type and number of inputs. This method has response object which actually creates the response to the user in the form of dynamic web pages and puts it in a presentable format. All this is done in the doGet method and directing between web pages and .jsp files for appropriate functioning of the application.

3.4.2 Method 2: doPost()

Input: HttpServletRequest request, HttpServletResponse response

Output: void

Method Description:

This function is generally used in the place of doGet() method in special cases where you prefer hiding the url of the server and the data you send from one page/jsp file to other page/jsp file.

3.5 Class name: StaffActionServ

Description: This is a servlet which inherits the HttpServlet class using the doGet() and doPost() methods of the parent class. This servlet is where the staff member is headed after login. After login which includes authentication of the user, the user has a set of options to choose from. The user can then choose one of these just by hovering and clicking the option required. The servlet then forwards to the concerned class/jsp/servlet to perform the actions as per user's need.

3.5.1 Method 1: doGet()

Input: HttpServletRequest request, HttpServletResponse response

Output: void

Method Description:

Every action to be done passes through this function. The data which comes from the website from the form is stored in request object irrespective of the type and number of inputs. This method has response object which actually creates the response to the user in the form of dynamic web pages and puts it in a presentable format. All this is done in the doGet method and directing between web pages and .jsp files for appropriate functioning of the application.

3.5.2 Method 2: doPost()

Input: HttpServletRequest request, HttpServletResponse response

Output: void

Method Description:

This function is generally used in the place of doGet() method in special cases where you prefer hiding the url of the server and the data you send from one page/jsp file to other page/jsp file.

3.6 Class name: User

Description: This is a servlet which is a result of Action servlet created earlier.

3.6.1 Method 1: User()

Input: int id, String actor, String name

Output: void

Method Description:

This is the constructor of the class used to create the object of the class. The object of the class is created using this function with pre-defined parameters a user object can be created which can be easily inserted in database and actions of the user can also be defined.

3.7 Class name: AdminAction

Description: This is a java class which calls functions of other classes to perform each action. Every request from Action class gets distributed here and the whole application moves in a specific direction to perform a specific action.

3.7.1 Method 1: create()

Input: String name, String category

Output: Boolean created

Method Description:

This method adds the new type of asset created to the database. The assets to this type of asset are to be added using add asset function. This function just creates the type of the new asset.

3.7.2 Method 2: AddAsset()

Input: String productID

Output: Boolean added

Method Description:

This method actually adds the asset to the database where its type is already created. The method takes the input as product ID which is given to it by the Action servlet which is in its request object. It just gives a pop-up which tells the created assetID.

3.7.3 Method 3: AddMultAsset()

Input: String productid, int number

Output: Boolean added

Method Description:

This method actually adds multiple assets to the database where its type is already created. The method takes the input as product ID and number of assets to be added which is given to it by the Action servlet which is in its request object. It just gives a pop-up which tells the created assetIDs.

3.7.4 Method 4: show()

Input: String assetID

Output: String out

Method Description:

This method deals with 3 different actions. These three actions are performed by the methods in View class. This method just calls the concerned method of the class.

3.7.5 Method 5: updateStatus()

Input: String assetID, String stat

Output: Boolean out

Method Description:

This method takes in assetID and new status as input and updates the status of the object in the database.

3.7.6 Method 6: updateLocation()

Input: String assetID, String loc

Output: Boolean out

Method Description:

This method takes in assetID and new location as input and updates the location of the object in the database.

3.7.7 Method 7: showAssetShifting()

Input: void

Output: void

Method Description: This method deals with 3 different actions. These three actions are performed by the methods in ViewAssetShifting class. This method just calls the concerned method of the class.

3.7.8 Method 8: showReport()

Input: void

Output: void

Method Description:

This method deals with 3 different actions. These three actions are performed by the methods in Report class. This method just calls the concerned method of the class.

3.8 Class name: AuditorAction

Description: This is a java class which calls functions of other classes to perform each action. Every request from Action class gets distributed here and the whole application moves in a specific direction to perform a specific action.

3.8.1 Method 1: show()

Input: String assetID

Output: String out

Method Description:

This method deals with 3 different actions. These three actions are performed by the methods in View class. This method just calls the concerned method of the class.

3.8.2 Method 2: showAssetShifting()

Input: void

Output: void

Method Description:

This method deals with 3 different actions. These three actions are performed by the methods in ViewAssetShifting class. This method just calls the concerned method of the class.

3.8.3 Method 3: showReport()

Input: void

Output: void

Method Description:

This method deals with 3 different actions. These three actions are performed by the methods in Report class. This method just calls the concerned method of the class.

3.8.4 Method 4: remove()

Input: String assetID

Output: Boolean out

Method Description:

When this method is called, all the records of the asset corresponding to the asset ID provided are deleted.

3.9 Class name: StaffAction

Description: This is a java class which calls functions of other classes to perform each action. Every request from Action class gets distributed here and the whole application moves in a specific direction to perform a specific action.

3.9.1 Method 1: show()

Input: String assetID

Output: String out

Method Description:

This method deals with 3 different actions. These three actions are performed by the methods in View class. This method just calls the concerned method of the class.

3.9.2 Method 2: updateStatus()

Input: String assetID, String stat

Output: Boolean out

Method Description:

This method takes in assetID and new status as input and updates the status of the object in the database.

3.9.3 Method 3: updateLocation()

Input: String assetID, String loc

Output: Boolean out

Method Description:

This method takes in assetID and new location as input and updates the location of the object in the database.

3.10 Class name: Asset

Description: This is a java class which creates an object for each asset object we deal with in each operation in order to produce faster results. This class's object is generally used to insert data into database and access any asset's data in the database easily and efficiently.

3.11 Class name: ViewAssetShifting

Description: This is a java class which is use to perform 3 actions

- View Asset Shifting History by date
- View Asset Shifting History by Asset ID
- View Asset Shifting History by Staff ID

These three actions are performed by 3 different methods. The methods are called by AdminAction or AuditorAction or StaffAction class to perform the above-mentioned actions. These functions of the class are called by showAssetShifting() method of the above mentioned classes.

3.11.1 Method 1: byDate()

Input: String from, String to

Output: void

Method Description:

This method takes in a from date and a to date and provides a history of all the Shifting operations done from the first date to the other. The output of this method is just a table of all the shifting history between those dates.

3.11.2 Method 2: byAssetID()

Input: String assetID

Output: void

Method Description:

This method takes in an assetID and provides a history of all the Shifting operations done to the asset whose assetID is provided. The output of this method is just a table of all the shifting history of an asset.

3.11.3 Method 3: byStaffID()

Input: String staffID

Output: void

Method Description:

This method takes in a staffID and provides a history of all the Shifting operations done by the staff member whose staffID is provided. The output of this method is just a table of all the shifting history done by a single staff member.

3.12 Class name: Report

Description: This is a java class which is use to perform 3 actions

- View Comprehensive report by date
- View Comprehensive report by Asset ID
- View Comprehensive report by Staff ID

These three actions are performed by 3 different methods. The methods are called by AdminAction or AuditorAction or StaffAction class to perform the above-mentioned actions. These functions of the class are called by showReport() method of the above mentioned classes.

3.12.1 Method 1: byDate()

Input: String from, String to

Output: void

Method Description:

This method takes in a from date and a to date and provides a history of all the operations done from the first date to the other. The output of this method is just a table of all the operations between those dates.

3.12.2 Method 2: byAssetID()

Input: String assetID

Output: void

Method Description:

This method takes in an assetID and provides a history of all the operations done to the asset whose assetID is provided. The output of this method is just a table of all the operations of an asset.

3.12.3 Method 3: byStaffID()

Input: String staffID

Output: void

Method Description:

This method takes in a staffID and provides a history of all the operations done by the staff member whose staffID is provided. The output of this method is just a table of all the operations done by a single staff member.

3.13 Class name: View

Description: This is a java class which is use to perform 3 actions

- View Status of an asset
- View Location of an asset
- View Asset

These three actions are performed by 3 different methods. The methods are called by AdminAction or AuditorAction or StaffAction class to perform the above-mentioned actions. These functions of the class are called by show() method of the above mentioned classes.

3.13.1 Method 1: viewAssetStatus()

Input: String assetID

Output: void

Method Description:

This method takes in an assetID and provides present status of the asset like, broken, can't be repaired, to be repaired, beyond economic rate and many more pre-defined states.

3.13.2 Method 2: viewAssetLocation()

Input: String assetID

Output: void

Method Description:

This method takes in an assetID and provides present location of the asset like, UG1, UG2, PG1, PG2, AC1, AC2 and many more pre-defined locations.

3.13.3 Method 3: viewAsset()

Input: String assetID

Output: void

Method Description:

This method takes in an assetID and provides all the details about the asset like, category, name, date of purchase, status, location and other details.

4.0 Execution Architecture

Runtime environment required is any device supporting Java and an OS of Windows 7 or later versions.

The whole project was built using the MVC architecture in OOPs for a better code reusability as well as better understanding. Testing and development of code is easier and can be done any non-member of the group easily jus y reading this document as well as Software Requirements document provided earlier.

5.0 Design Decisions and Tradeoffs

The earlier design of the landing page had icons spread all over the page and the user could be directed to the concerned page. Thanks to JSP pages which can help us embed all the functionality easily in all the pages easily and with a simple use a tag. Now, the user can jump between pages easily just hovering on to the options seen at the top of every page and clicking it. Now, we have a comprehensive toolbar like headline for every page which can help the user to switch between actions and pages easily.

6.0 Pseudo Codes

Create():

1. Make connection with database
2. Sql1="Select prod_id from products;”
3. PreparedStatement ps=(PreparedStatement) con.prepareStatement(sql1);
4. ResultSet rs=ps.executeQuery();
5. id=rs.getString(1);
6. id=id+1;
7. Execute the following statement to store the new product details in database
8. Insert into products values(id,name,category);

ViewStatus():

1. Make connection with database
2. Take asset_id1 as input
3. sql1="Select status from assets where asset_id= asset_id1;”
4. rs = ps.executeQuery();
5. While rs.next() is TRUE repeat Step 6
6. status=rs.getString(1);

ViewLocation():

1. Make connection with database
2. Take asset_id1 as input
3. sql1="Select location and room from assets where asset_id= asset_id1;”
4. rs = ps.executeQuery();
5. While rs.next() is TRUE repeat Step 6
6. location=rs.getString(1)+" "+rs.getString(2);

ViewAsset():

1. Make connection with database
2. Take asset_id1 as input
3. sql1="Select * from assets where asset_id=asset_id1";
4. rs = ps.executeQuery();
5. While rs.next() is TRUE repeat Step 6
6. asset=rs.getString(1)+" "+rs.getString(2)+" "+rs.getString(3)+" "+rs.getString(4)+" "+rs.getString(5)+" "+rs.getString(6)+" "+rs.getTimestamp(7);

Remove():

1. Make connection with database
2. Take asset_id
3. Sql=”Remove assets where assetId=asset_id;”
4. Executing the Sql statement will remove the asset from database.

UpdateStatus():

1. Make connection with database
2. Take asset_id and status1 as inputs
3. Sql=”Update assets set status=status1 where assetId=asset_id;”
4. Sql=insert into status values(trans_id, prod_id,asset_id, status1,user_id, time)
5. Executing these two Sql statements will update the database.

UpdateLocation():

1. Make connection with database
2. Take asset_id , room1 and location1 as inputs
3. Sql=”Update assets set location=location1 and room=room1 where assetId=asset_id;”
4. Sql=insert into location values(trans_id, prod_id,asset_id, toLocation,toRoom, user_id, time)
5. Executing these two Sql statements will update the database.

ViewAssetShifting ByAssetID():

1. Make connection with database
2. Take asset_id1 as input
3. sql1="Select tolocation,toroom,time from location where asset_id=asset_id1;”
4. rs = ps.executeQuery();
5. While rs.next() is TRUE repeat Step 6
6. Print(rs.getString(1)+" - "+rs.getString(2)+" - "+rs.getString(3));

ByStaffID():

1. Make connection with database
2. Take Staff_id as input
3. sql1="Select prod_id,asset_id,tolocation,toroom,time from location where UserId=Staff_id;”
4. rs = PS.executeQuery();
5. While rs.next() is TRUE repeat Step 6
6. Print(rs.getString(1)+" - "+rs.getString(2)+" - "+rs.getString(3)+" - "+rs.getString(4)+" - "+rs.getString(5));

ReportByDate():

ReportByAssetId():

1. Make connection with database
2. Take asset_id1 as input
3. sql1="Select tolocation,toroom,time from location where asset_id=asset_id1;”
4. rs = ps.executeQuery();
5. Print(“ Shifting”);
6. While rs.next() is TRUE repeat Step 7
7. Print(rs.getString(1)+" - "+rs.getString(2)+" - "+rs.getString(3));
8. sql1="Select status,time from status where asset_id=?;”
9. rs = ps.executeQuery();
10. Print(“ Status”);
11. While rs.next() is TRUE repeat Step 12
12. Print(rs.getString(1)+" - "+rs.getString(2))

ReportByStaffId():

1. Make connection with database
2. Take staff_id1 as input
3. sql1="Select tolocation,toroom,time from location where asset_id=staff_id1;”
4. rs = ps.executeQuery();
5. Print(“ Shifting”);
6. While rs.next() is TRUE repeat Step 7
7. Print(rs.getString(1)+" - "+rs.getString(2)+" - "+rs.getString(3));
8. sql1="Select status,time from status where staff_id=?;”
9. rs = ps.executeQuery();
10. Print(“ Status”);
11. While rs.next() is TRUE repeat Step 12
12. Print(rs.getString(1)+" - "+rs.getString(2));

Logout():

1. Get session object from request
2. Session.invalidate();

Login():

1. Make connection with database
2. Take user_name and password as input
3. sql="Select * from UserN where UserID=user_name and password=aes_encrypt(password,key);”
4. rs = ps.executeQuery();
5. While rs.next() is TRUE Repeat Steps 6 to Step 11
6. actor=rs.getString(2);
7. name=rs.getString(3);
8. if(actor==null)
9. Print(“username or password is wrong”);
10. Else
11. Print(“Successful login”);